

Module 1: AI and Risk - Introduction and Overview

Learning Objectives

The application of artificial intelligence (AI) introduces a novel set of risks to all organizations that use it. This module offers a historical perspective on AI, an overview of both machine learning methodologies and generative AI, and an introduction to the risks associated with using AI/ML.

After completing this module, you should be able to:

- Explain some central principles of Classical AI, including search methods and recursion.
- Describe, at a high level, how reinforcement learning works.
- Describe, at a high level, how neural networks work, and how they differ from Classical AI systems.
- Articulate the potential and limitations of deep learning.
- Identify key breakthroughs leading to advances in AI and ML.
- Compare and contrast reinforcement, supervised, and unsupervised learning, and identify practical applications for each technique.
- Discuss risks associated with inscrutability in AI and ML.

- Discuss risks associated with over-reliance on AI systems.
- Discuss ways in which AI exposes individuals, organizations, and society to risk.
-

1.0 Classical AI

Intelligence – or “reason” – has long been seen as the human quality that, more than any other, distinguishes us from other species and gives us our unique power over the world. But there has been speculation through the ages about the possibility of duplicating the power of human reason with artificial technology. This began to seem practically feasible after the industrial revolution, and in the early 19th century, Charles Babbage conceived of a mechanical, gear-driven “difference engine” that could replace error-prone human calculation in the efficient production of mathematical tables. His more-ambitious design for an “analytical engine” is widely seen as anticipating the digital computer, though bringing it to successful completion would be a huge feat of engineering, even with modern technology.

The crucial breakthrough occurred when the digital computer emerged as the brainchild of Alan Turing in 1936, although his “computing machine” (universally known as a “Turing machine”) was a theoretical invention – designed to prove fundamental results about mathematics and what we now call theory of computation – rather than a practical device. Turing argued convincingly that such a machine would in principle be able to follow any given recipe for calculation that we are able to

devise, and in that sense would be a universal programmable computer.¹ These ideas strongly informed his work at Bletchley Park during the Second World War, devising mechanical methods of decrypting the Nazi Enigma codes, which gave a massive impetus to the development of the electronic computer after the war (on both sides of the Atlantic). Turing himself had visionary thoughts about the possibility and potentially tremendous power of intelligent machines, which he famously presented in his paper, "Computing Machinery and Intelligence" of 1950. This centers on the idea of an "imitation game" – now known as a "Turing Test" – in which a computer proves itself to be intelligent if it can generate textual conversation of a quality indistinguishable from that of an intelligent human, ranging over any topic chosen by the human "interrogator." Though highly controversial philosophically, the Turing Test remains influential, and has become particularly salient recently, as "chatbots" based on large language models (such as ChatGPT) have provided – for the first time – examples of computer programs that are relatively plausible contenders.

¹ Turing's seminal paper, entitled "On Computable Numbers, with an Application to the *Entscheidungsproblem*", appeared in the *Proceedings of the London Mathematical Society* 42 (1936-37), pp. 230-65. For links to relevant online talks and papers, see <https://www.philocomp.net/videos/turing.htm>, and for a set of Oxford University lectures on "Alan Turing on Computability and Intelligence", see https://www.philocomp.net/videos/turing_lectures.htm.

1.1 Specific versus General AI

Media discussion of AI, and futuristic speculation, is often concerned with the question of whether an artificial system could surpass human *general* intelligence. This may in part be a legacy of the Turing Test, but it probably also reflects a natural concern

about challenges to our primacy in reason and understanding – abilities that are so closely bound up with human identity. Such concerns are also commonly associated with worries about existential risk arising from an artificial general intelligence (AGI). It is far less threatening to consider that machines might outperform us in very specific domains, such as calculating mathematical tables or generating complex statistics. But it is the latter kind of machine – designed to be *intelligent at solving specific problems* – that will mostly concern us in what follows.

Some would argue that a system can only properly count as “intelligent” if it has the sort of general ability that we do. It has become commonplace, however, to talk of intelligent or “smart” systems that are relatively limited. Even Turing’s legacy does not point very heavily toward a purely general interpretation of AI. In 1950, his aim was to devise a thought experiment that demonstrated the possibility of AI by imagining a computer system that could not reasonably be denied being intelligent. In that context, it made good sense to consider a conversational system of comparable versatility and skills to a human, as a sort of conceptual existence proof of that possibility.² But once we have been convinced that there is no objection *in principle* to artificial intelligence, it seems odd to deny that a system can be intelligent in addressing specific tasks, without having more general competence.

² For more on this understanding of the Turing test, see the 2021 paper linked from footnote 1.

1.2 Good Old-Fashioned AI (GOFAI): Logic and Games

The term “artificial intelligence” was coined in 1956 by John McCarthy of Dartmouth College, who along with others (Marvin Minsky of Harvard, Nathan Rochester of IBM, and Claude Shannon of Bell Laboratories) organized a seminal two-month summer conference on the topic.³ Echoing Turing’s notion of universality, this was based on “the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.”⁴ Here the paradigm of intelligence was accordingly *explicit, precise processing*, as typified by symbolic logic or other formal rule-governed systems. And in arguing for the universality of his computing machines in 1936, Turing had emphasized their ability to mimic logical proofs, constructed step-by-step from a given set of “axioms,” and systematically applying such rules.

Following in this tradition, in 1955 Allen Newell, Herbert Simon, and Cliff Shaw (all working at the Rand Corporation) started developing their *Logic Theorist* program, designed to generate proofs in *propositional logic*, the simplest and most fundamental form of modern symbolic logic. Newell and Simon presented this program at the Dartmouth conference (though not, apparently, to great acclaim at the time), and they later used it to confirm most of the theorems in Chapter 2 of the monumental *Principia Mathematica* by Alfred North Whitehead and Bertrand Russell.⁵ This famous three-volume work had aimed to demonstrate the *logicist* thesis, that all of mathematics could be deduced from suitable axioms by pure logic, so it was an excellent choice for a high-profile application of machine intelligence, made all the more impressive when *Logic Theorist* was able to find a new and more elegant proof (of Theorem 2.85).

Another paradigm aspiration of AI was to create a program capable of playing “intellectual” games such as chess at a high level. This again went back to Alan Turing, although his attempt to implement such a program proved impractical on the hardware then available, and he never got beyond a “paper” implementation of chess-playing rules. Chess would eventually be conquered with the advent of much more powerful machines and sophisticated techniques, culminating with the defeat of world champion Garry Kasparov by *Deep Blue* in 1997. But by far the most prominent early success in this direction came with the program developed in the late 1950s by Arthur Samuel of IBM, which focused on the much simpler game of checkers (or draughts).

Quite apart from their intrinsic interest (especially to mathematically minded thinkers) and their promotional value in raising public awareness of the possibility of AI, such intellectual games provided an attractive testbed for the development of AI for at least three related reasons. First, they are relatively *simple*, compared with the complexities of normal life, and hence could plausibly be tackled even when computing resources would have been severely stretched in even recording the details of a physical environment, let alone taking on the immensely complicated challenge of reasoning about it. Secondly, such games are straightforwardly *rule-governed*, with clear and explicit regulations regarding the setup of the game, how the pieces are permitted to move, the effects of these moves on the board position, and criteria for winning, losing, and drawing. Thirdly, they are *competitive*, and we are already very familiar with the idea that players can be rated according to their relative skill. Hence such games lend themselves very naturally to the *assessment* of that

skill, so that it is straightforward to judge researchers' progress as they endeavor to produce a program capable of competing at ever higher levels.

³ McCarthy and Minsky are widely considered to be among the "founding fathers" of AI. In 1959 they co-founded the AI Project at MIT, which later became the AI Lab (and ultimately CSAIL, the Computer Science and Artificial Intelligence Laboratory).

McCarthy went on to devise the AI computer language LISP in 1960, and to found the Stanford Artificial Intelligence Lab (SAIL) in 1963.

⁴ Quoted from the first paragraph of "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence", by McCarthy, Minsky, Rochester, and Shannon, and dated August 31, 1955 (a transcript of which can be found at <https://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>).

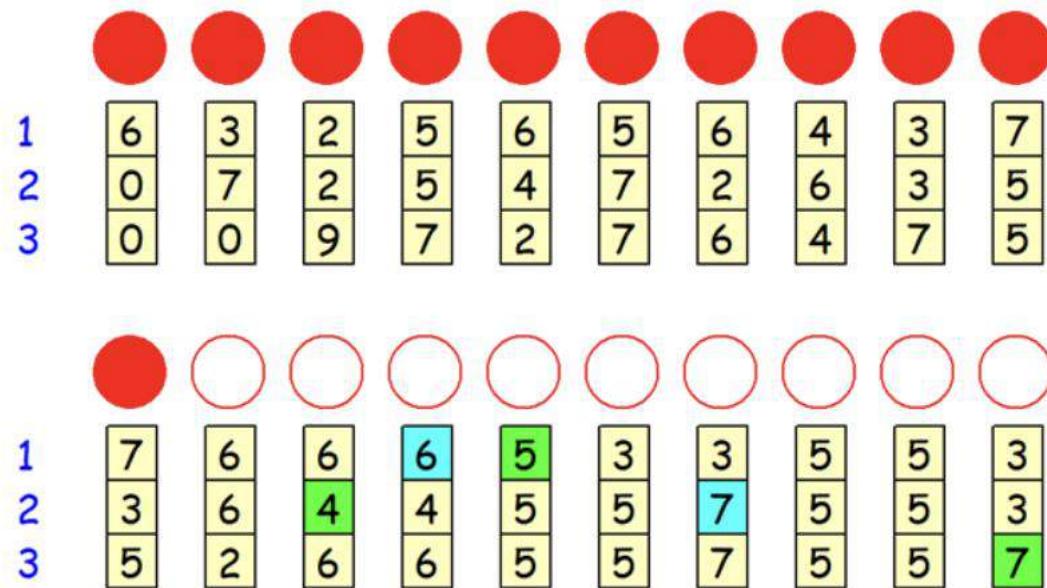
⁵ Alfred North Whitehead and Bertrand Russell, *Principia Mathematica*, 3 vols (1910, 1912, 1913), Cambridge: Cambridge University Press.

1.3 Simple Reinforcement Learning

The three aforementioned characteristics also combine to make it possible to develop a program that can *learn* to play such a game, at least in principle. Indeed, Samuel's program did this, thus becoming the first substantial example of *machine learning* (a term he coined). The first technique is reinforcement learning, illustrated with the game of Nim, a game of strategy in which two players take turns removing (or "nimming") objects from the game board. There are several versions of Nim, but this one starts with 20 coins being placed on a table. Now taking turns, each player removes either 1, 2, or 3 coins, and the winner is the player who gets to remove the last coin(s) from the table. There is, in fact, a simple winning strategy for the second player in this game: Depending on whether player *A* has removed 1, 2, or 3 coins on his turn, player *B* should respectively remove 3, 2, or 1 on his turn. A total of 4 coins is

removed by A and B together, and so after 5 rounds all 20 coins will have been removed, with B taking the last group.

Suppose, however, that we are new to the game, with no clue what strategy to adopt, and we want to write a machine-learning program that will learn to play it for us. One easy way of doing this is to set up a program that, in any given position, will choose its move *randomly* based on a given set of *weights*, and that learns by adjusting those weights according to how well each move turns out in practice. For simplicity, suppose that these weights can vary between 0 and 9, and we set them all up initially to be 5. Then we play against the system, and depending on which player wins or loses each game, the weights are changed to reflect that success or failure. [Figure 1.1](#), for example, is how things might look after a few games, in the middle of a game in which there are currently 11 coins remaining and the computer is about to move:⁶



The computer will choose on these probabilities:

1 coin: $7/15$

2 coins: $3/15$

3 coins: $5/15$

Figure 1.1

In this ongoing game, the human player started by removing 3 coins (indicated by the green square on the far right, in what we might call the $20/3$ position). Then, with 17 coins remaining, the computer chose to remove 2 coins (indicated by the cyan

square in the 17/2 position). The next three moves (human taking 1, computer taking 1, human taking 2) are shown by the other green and cyan squares. Here, as we see, the computer is choosing its move randomly based on the three *weights* displayed under the highest-numbered remaining coin (i.e. coin 11), so that it has probability $7/15$ of choosing 1 coin, $3/15$ of choosing 2, and $5/15$ of choosing 3 (15 being the sum of the three weights). So it is most likely to remove 1 coin, reflecting that this has so far been the most successful move when 11 coins remain. This process of alternate moving continues until the game ends, after which reinforcement learning takes place as follows. Whichever player *wins*, all of his chosen moves that led to that victory will be *positively* reinforced by having their weight incremented by one (with a maximum of 9), whereas the moves not chosen in those positions will be *negatively* reinforced by having their weight decremented by one (with a minimum of 0). Conversely, for the player who *loses*, all of his chosen moves that led to that loss will be *negatively* reinforced by having their weight decremented by one (with a minimum of 0), whereas the moves not chosen in those positions will be *positively* reinforced by having their weight incremented by one (with a maximum of 9).⁷

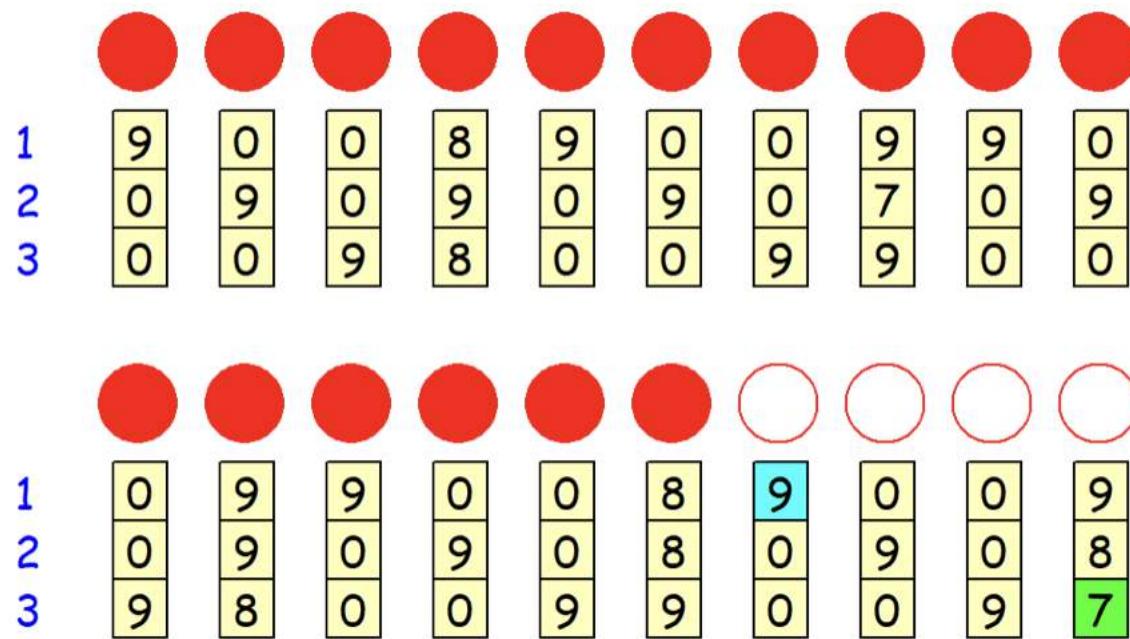


Figure 1.2

Though extremely simple, we find that this mechanism quickly discovers the winning strategy, especially if we replace the human player with an automated opponent following the same learning strategy.⁸ After around 100 games, we find that the weights have been adjusted in the sort of way shown below. In this particular game ([Figure 1.2](#)), the human player started by removing 3 coins, and the computer responded by removing 1 coin, leaving 16 coins remaining for the human's next move. Note that the computer's last move was inevitable, given the 9-0-0 weights that it has learned when 17 coins remain, and note that a similar point applies when

18 or 19 coins remain – in all three cases, there is a 9/9 probability that the computer will reduce the remaining coins to 16. The same pattern can be seen also in the rest of the table: *whenever it can do so, the computer will reduce the coins remaining to a multiple of 4*. And thus eventually, as explained earlier, the number reduces to 0, with the computer player winning by removing the last coin(s). The program has thus learned for itself the optimal winning strategy by reinforcement learning. As the various possible moves were tried out, those that ended up leading to victory in any particular game were *positively reinforced*, in that the weights were adjusted to make them more probable in the future, whereas moves that led to loss were *negatively reinforced*, by being made less likely in the future.

This very crude kind of reinforcement learning – accumulating experience of the results that arise from the different available moves in very specific positions in a game – can only be applied to relatively simple games in which the number of possible positions is limited (although we see later in this module how reinforcement learning can be extended to more complex games such as checkers).

⁶ Image created by author using his own *Turtle System* software, available free from www.turtle.ox.ac.uk. Interested readers can thus try out these programs, and examine or adapt them if desired.

⁷ In the current case, for example, the weights at the initial 20-coin position are 3/3/7, with me having chosen the last of these, while the weights at the 17-coin position are 3/7/7, with the computer having chosen the second of these. So if the computer now ends up beating me in this game, the 20-coin weights will change to 4/4/6 – downweighting the choice I made, and upweighting the other two – while the 17-coin weights will change to 2/8/6 – upweighting the choice it made, and downweighting the other two.

⁸ Learning is much slower if the automated “experimental” opponent plays randomly, because then the system will often end up winning games that it should have lost due to an early error, while the opponent will lose games that it should have won, resulting in misleading reinforcement. The best opponent for quick learning is one that is good at punishing errors, so that the system learns quickly to avoid those errors.

1.4 Lookahead

In our Nim-learning program, the computer has not at any point “looked ahead” to see what outcomes are likely to emerge depending on its possible choices. The program has ultimately developed a perfect winning strategy, but it has done so by a sort of evolutionary process, whereby successful moves have been made progressively more probable as future choices, and unsuccessful moves have been made progressively less probable. It now plays perfectly, but one might hesitate to call it genuinely “intelligent” when it is just blindly choosing moves according to specified probabilities, without any more-sophisticated “lookahead” or planning. Such intelligence as it displays seems to lie more in the learning process that devised its playing method than in the method itself.

Lookahead becomes essential when our aim is to form a multistep plan to achieve some goal, within a context where there is a very large number of possible situations overall, but with a relatively constrained and predictable range of options in any particular situation. A familiar example would be attempting to solve a Sudoku puzzle, where we are unsure whether to write a 4 or 7 in a particular cell, so we start looking ahead to identify the consequences that will follow: “If I put 4 here, then that means this other cell will have to be 7, in which case this third cell cannot be 7 and must be 2, and then ...”. Here we are following a sequence of implications, with the idea that eventually we will either find a contradiction (in which case, the original cell must be

7 rather than 4, so we rewind – or “backtrack” – the entire sequence), or generate a complete solution.

For another example, consider the Tower of Hanoi puzzle (see [Figure 1.3](#)).⁹ In this case, the tower consists of nine differently sized and differently colored disks, which were originally piled in order – from largest to smallest – on the leftmost pillar (where the three largest disks remain). There are two other pillars, and *disks can be moved from one pillar to another, but only one at a time, and never putting a disk on top of a disk that is smaller than itself*. Subject to these rules, the original task was to move the entire tower from the leftmost pillar to the rightmost pillar, and this picture is a snapshot taken part-way through the execution of this task. Suppose, then, that we encounter this unfinished task and wish to complete it. Here we have just three possible legal moves: (a) moving disk 1 (the smallest, red one) from back to left; (b) moving disk 1 from back to right; and (c) moving disk 4 (the purple one) from right to left. But which do we choose? Here we might try tactical lookahead to narrow down the sequences of options that look plausible (e.g., that give the appearance of making progress and don’t involve “wasted” moves). But more promising is a strategic planning approach, whereby we try to identify what shorter-term outcomes would usefully contribute to the completion of our task. Here we might notice that to move, say, the light blue disk 7 from the leftmost pillar, we need first to pile up disks 1 to 5 on the orange disk 6, so as to leave the rightmost pillar clear. To achieve this, we need to move green disk 5 onto orange disk 6, and to achieve that, we need first to pile up disks 1 to 4 (the purple one) on the leftmost pillar. So we can conclude that the best move here is to

move the purple disk from right to left, and similar thinking will enable us to work out the rest of the puzzle.

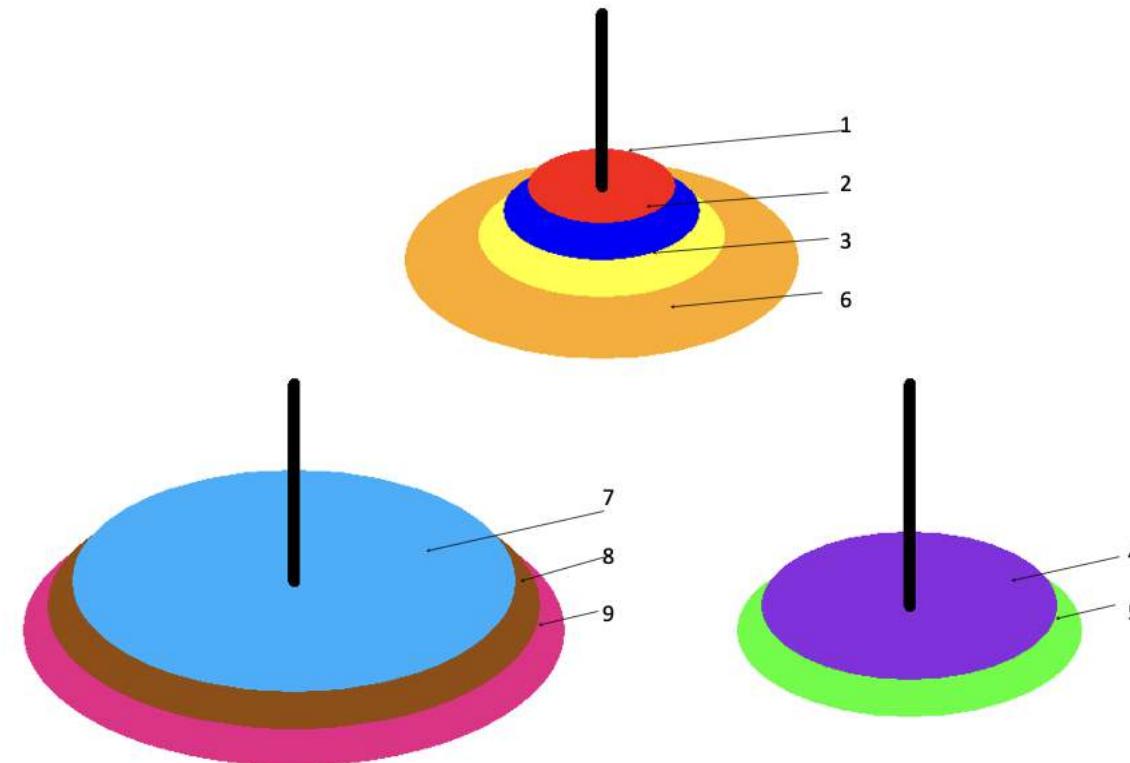


Figure 1.3

⁹ Image created by author using his own *Turtle System* software, available free from www.turtle.ox.ac.uk.

1.5 Search

Notice that in working through these various options, we are conducting a search through the possible outcomes to find one that will best achieve our aims. And indeed, generating sets of possible outcomes and searching through them is one of the most fundamental techniques in AI (see a worked-out example of searching through a lookahead game tree in the Noughts-and-Crosses algorithm of section 1.7 below).

The first of these involves a number-guessing program, which provides a simple illustration of “binary search,” whereby a search space is progressively divided in two until the desired target has been found. The user is invited to choose a number within a given range (say, between 1 and 1,000), and the program’s job is to guess this number as quickly as possible. To accomplish this, we use two numeric variables *lowest* and *highest* to keep track of the current range, and having initialized these to 1 and 1,000, respectively, we enter a loop in which the program guesses the rounded mean of *lowest* and *highest*, and then asks the user to say whether this is *correct*, *too high*, or *too low*. If it is too high, then *highest* is revised downward to one less than the guess, whereas if it is too low, then *lowest* is revised upward to one greater than the guess. So whenever a guess is unsuccessful, the “search space” is reduced by at least half, removing the part that has been ruled out by the user’s response. Assuming the responses are consistent, this will find the correct value within at most 10 guesses (since $2^{10} = 1,024$, and there are 1,000 possibilities).

A more complicated example is attempting to find the optimal route from one city to another by taking an appropriate sequence of journeys (by train, let us suppose), with no obvious best combination to reach our destination. In the simplest case, we are attempting to optimize on just one criterion (e.g., distance, though it might also be cost or time), and we want our program to search through all the potentially optimal routes, finding the best of them. There are several different ways in which this can be undertaken – for example, it might use breadth first search, in which we keep track of all the different possible routes as we build them up stage by stage; or it might use depth first search, in which we follow each possible route to the end before moving on to compare the next one. Both have serious disadvantages: For example, breadth-first search is inefficient and expensive in memory, whereas depth-first search may waste time focusing on poor options initially while neglecting short and straightforward paths that breadth-first search would have found far more quickly. Usually better than either of these is to employ methods of heuristic search, where we take advantage of some additional information (e.g., the crow-flies distance between cities) to help us decide which options are worth exploring first. The best-known example of this kind of search technique is the A* ("A star") algorithm, which for each location reached so far in our search, keeps track of both the distance that has been travelled so far to reach that location, and an (optimistic) estimate of the distance still to go to our ultimate destination (for example, the crow-flies distance).¹⁰ The locations are kept ordered according to the sum of these two values, and exploration of ongoing routes always starts from the location that currently has the shortest sum.

Heuristic techniques such as these can indeed help considerably in facilitating many search problems, but unfortunately in more complex or demanding cases – for example the task of finding the *perfectly optimal* route around a list of cities (the so-called “travelling salesman problem”) – the search space grows so enormously as the number of cities increases – a combinatorial explosion as the relevant numbers of possible choices get multiplied together – that certainty of solution quickly becomes unfeasible (see sections 1.8 and 1.9 below). Much the same applies to complex games such as chess, where searching for optimality also involves additional complications because it is adversarial, in that we are trying to maximize our own advantage while our opponent is trying to minimize it. An essential preliminary is the important concept of recursion.

¹⁰ It is crucial that the estimate be *optimistic* if the A* algorithm is to be guaranteed to find the best route, because this means that potential routes that *might* prove to be better than any yet found are “kept in the game” (and the search halts when a complete route is found which is shorter than any estimated route remaining). Thus the crow-files or Euclidean distance is eminently suitable as an estimate, since no train journey can be shorter. This would not be true if the algorithm were to use what is called Manhattan distance (i.e. sum of differences in x and y coordinates on some predefined grid, reminiscent of distance through the streets of Manhattan). But the latter is useful in various machine learning algorithms.

1.6 Recursion

The discussion of lookahead might have given the impression that any algorithmic solution to the Tower of Hanoi problem is bound to be highly complex, with sophisticated lookahead and comparison of possible future lines of play. But in fact the problem is extremely easy to solve using a powerful technique called *recursion*, which is of huge importance within classical AI (and, indeed, in computer science

generally). The Tower of Hanoi is one of the most famous and elegant illustrations of the recursive technique, whereby a complex problem is solved by being progressively *reduced* to simpler instances of the same kind of problem.

We have already implicitly noticed how this sort of reduction is implied in the Tower of Hanoi problem, because whenever we want to move one of the larger disks – say disk 5 – from one pillar to another, we need first to pile up all of the disks that are smaller than it – here disks 1 to 4 – onto a single pillar, so that disk 5 will be free to move without violating the rule that it cannot be put on top of a smaller disk. It immediately follows that if we want to move *the entire pile of disks 1 to 5* from pillar *A* to pillar *B*, using pillar *C* as an intermediary, then we have to perform the following three subtasks:

To move the pile of disks 1-5 from *A* to *B*

- Move the pile of disks 1-4 from *A* to *C*
- Move disk 5 from *A* to *B*
- Move the pile of disks 1-4 from *C* to *B*

The second of these subtasks is trivial, just involving a single disk transfer. But what about the first and third subtasks: How are those to be achieved? Well, we can apply *exactly* the same kind of reasoning that we have just applied to the problem of moving the 5-pile from *A* to *B*, to the problem of moving the 4-pile from *A* to *C* (and later from *C* to *B*). Thus the first subtask itself divides into three:

To move the pile of disks 1-4 from *A* to *C*

- Move the pile of disks 1-3 from *A* to *B*
- Move disk 4 from *A* to *C*
- Move the pile of disks 1-3 from *B* to *C*

And, in fact, this simple line of reasoning provides a complete solution to the Tower of Hanoi, dividing every problem of moving a pile of n disks into two smaller problems of moving a pile of $n-1$ disks (plus one move of disk n), so that in the end everything is reduced down to single disk movements. Moving the entire pile of 9 disks by this method takes 511 steps altogether (calculated as $2^9 - 1$).¹¹

¹¹ This can be neatly proved by the method of mathematical induction, as follows. Moving one disk obviously requires just 1 move, which is $2^1 - 1$. Now suppose – in line with the hypothesis that we are trying to prove – that moving k disks involves $2^k - 1$ moves. Well, in that case, moving $k + 1$ disks will require moving the pile of k disks to the intermediate pillar ($2^k - 1$ moves), then moving disk $(k + 1)$ to the destination pillar (1 move), then moving the pile of k disks from the intermediate to the destination pillar ($2^k - 1$). Thus the total number of moves required to move $k + 1$ disks will turn out to be $2 \times (2^k - 1) + 1 = (2^{k+1} - 2) + 1 = 2^{k+1} - 1$, exactly fitting our hypothesis. The upshot is that if our hypothesis is true for $k = 1$ (as it is), then it must be true for $k = 2$, and then it must be true for $k = 3$, and so on for all positive integers.

1.7 Recursive Adversarial Tree Search – Solving Noughts and Crosses (Tic-Tac-Toe)

Recursion can also be applied to solve the relatively complex problem of playing an adversarial game, by combining with lookahead and search. Here our task is to design a program that plays a perfect game of Noughts-and-Crosses (also known as Tic-Tac-Toe), in the sense that it will never lose and will always find a winning sequence of play if one exists (though it won't necessarily find the quickest path to a win). Crucial to this will be devising a function that can assess the *value* of any position *to the player whose turn it is to move in that position* (i.e., whether it is a position that, with best play on both sides, will lead to a win for that player [value +1]; a loss for that player [value -1];

or a draw [value 0]). Once we have such a function, it will be easy to construct our program.

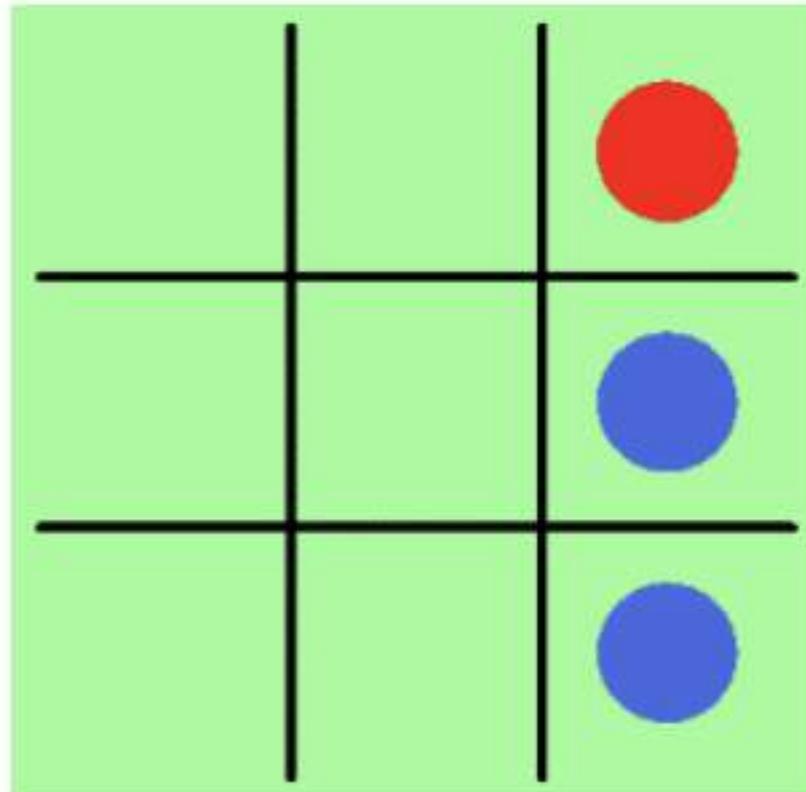


Figure 1.4

We start by considering a game between Blue (male) and Red (female) which has reached the position shown in [Figure 1.4](#). It is Red's turn to move, and she has 6 choices of empty squares, so we call this a "6-space-position" or "6-position" for

short. After Red makes her move, Blue will be faced with a 5-position in which to make his reply. Now suppose, hypothetically, that Red had some infallible method – some *oracle* – for assessing the value *to Blue* of any 5-position he might face: How could *Red* use this now to work out her best move? Clearly, Red should choose a move that gives Blue the *lowest possible value* in the resulting 5-position. In the current case, that means choosing one of the top two empty squares, either of which will yield a 5-position that is losing for Blue (i.e., with a value for Blue of -1).¹²

We now have a clear method for calculating the theoretical value of this 6-position to Red: We work through all the possible 5-positions that could result, pick the *lowest value* of any of these positions (here -1), and invert it (here $+1$) on the basis that -1 to Blue is $+1$ to Red (and vice-versa). This is called adversarial search, because Red is supposed to look for whichever move is *least* advantageous for her opponent.¹³

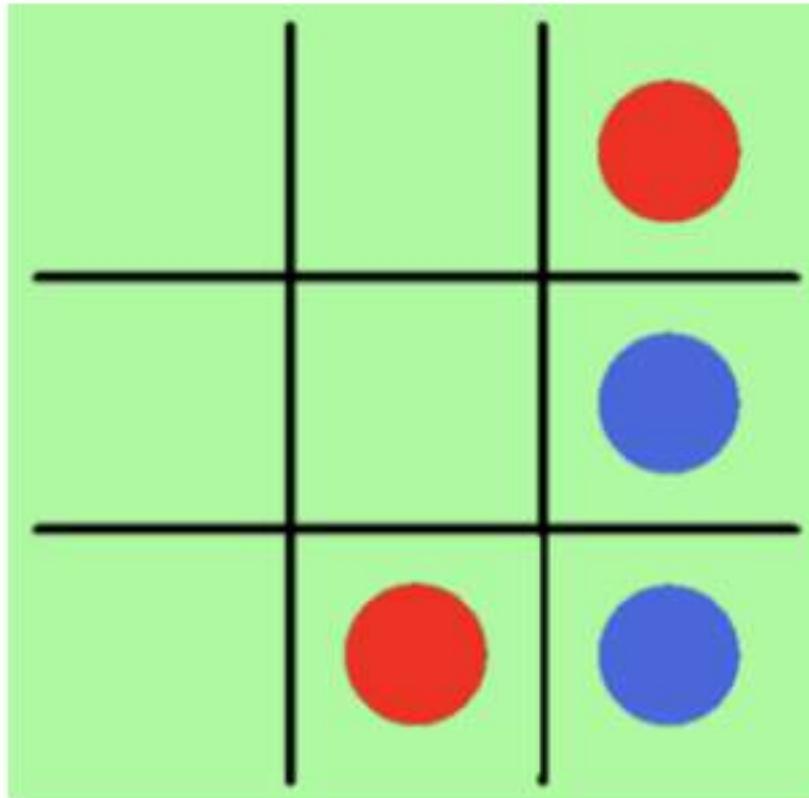


Figure 1.5

But all this depended on having an oracle that could tell us the value of any 5-position, and of course we have no such thing. But now we can repeat the same sort of reasoning at the next level, as we can see if we consider a possible (suboptimal) continuation from the position above, as shown in [Figure 1.5](#). This is a 5-position with Blue's turn to move, and after he makes his move, Red will be faced with a 4-position

in which to make her reply. Much as before, we imagine what Blue could do if he had some oracle for assessing the value (to Red) of any 4-position. In that case, he could work through all the possible 4-positions that could result from the current position, use the oracle to assess their value to Red, pick the *lowest* of those values (here -1 , in the case where Blue moves in the middle square), and invert it to yield the value to himself of the current position (i.e., $+1$).

Note, importantly, that the same sort of reasoning will apply to *any* 5-position, *except where Blue has an immediate winning move*, so that one of the resulting 4-positions contains a line of three Blues – then the value to Red of that position is clearly -1 , without requiring any oracle.

Spelling out the consequences of this line of reasoning:

- If we can evaluate any 5-position, then we can use these assessments to evaluate any 6-position;
- If we can evaluate any 4-position, then we can use these assessments to evaluate any 5-position;
- This pattern continues, with the upshot that if we can evaluate any 0-position (i.e., a position with no remaining moves to be played), then we can evaluate any 1-position, and hence any 2-position, and hence any 3-position, and hence any 4-position, and so on, all the way back to the initial 9-position.

- Apart from this, we need to be able to recognize when a game has ended owing to completion of a line (yielding a position of value -1 to the player whose turn it would otherwise be).

We can now specify our complete algorithm for evaluating a Noughts-and-Crosses (Tic-Tac-Toe) position in which it is player x 's turn to move:

1. Has x already lost (because the opponent has a line of three in the current position)? If so, the x -value of the position is -1 .
2. Is the position full up (with all nine spaces filled)? If so, its x -value is 0 (i.e., drawn).
3. If neither of these, then construct in turn all of the positions (with y to move) that can arise from the current position (i.e., try all of x 's possible moves in the current position in turn).

Evaluate each of these new positions *from Y's point of view using this very same algorithm* (resulting in evaluations Y_1 , Y_2 , and Y_3), and then assign – as x -value of the *current* position – the *inverse* of the *lowest* of Y_1 , Y_2 , and Y_3 .

When its turn comes, the computer should select a move that leads to a position with the lowest available evaluation (from its opponent's point of view).

¹² By contrast, if Red moves to either of the bottom two empty squares, then Blue should win (so the value of the resulting 5-position will be $+1$ to the player whose turn it is to move in that position); and if Red moves to either of the two middle empty squares, then the game should be drawn (value 0).

¹³ In the specific context of two-player games, this is called minimax search, whereby when looking through the available moves, I presume that my opponent will choose whichever reply *maximises* their benefit, and on that assumption, I choose the move which *minimizes* that maximum benefit – i.e. I choose the move which is worst for them, on the presumption that they will be responding optimally. The same principle applies at every level, and the recursive algorithm described below ensures this.

1.8 Complexity, Heuristics, and Refinement by Reinforcement Learning

It is very surprising that an assessment algorithm that is capable of exhaustively analyzing all of the possible lines of play in a game of Noughts-and-Crosses, implicitly creating a “search tree,” and keeping track of those positions’ value can be implemented so simply (in fewer than 25 short lines of computer code). But there is a catch – the curse of *exponential complexity*, leading to a *combinatorial explosion*. Noughts-and-Crosses is practically solvable in this way only because the number of possible lines of play is so relatively small, with 9 possible first moves, 8 possible responses, then 7, 6, 5, and so on, making an upper limit of 9 factorial (i.e., 362,880, though the true figure will be significantly less because so many games finish sooner). If we try to solve complex games like go, chess, or even checkers in this way, then the depth and exponential growth of the search tree (with an average branching factor per move of around 250, 30, and 10, respectively) means that at any plausibly achievable speed of computation, the Earth will have been swallowed up by an aging Sun long before the program even makes its first move.

As Samuel recognized, therefore, a practicable checkers program could not possibly work by exhaustive search through all possible combinations of moves. Instead, in any position it must limit its exhaustive search to, say, the first n moves (e.g., just 3 in the case of Samuel’s program, perhaps 8 [i.e., 4 moves for each player] in a modern program).¹⁴ Then to assess the positions reached through this analysis (and

thus to select which path is best to follow), it would use some set of *heuristics* – in other words, calculable criteria or rules of thumb that can usefully guide the choice of position to aim for, even though they cannot be guaranteed to reach an optimal solution. In the case of checkers, perhaps the most obvious criterion for assessing a position is based on material balance of our pieces as compared with our opponent's. A second plausible criterion might be the relative mobility of those pieces. A third could be how far advanced those pieces are (on the way to becoming “kings”), and a fourth the relative number of “kings” already achieved. Suppose, then, that we have come up with six such plausible criteria,¹⁵ and have worked out a method of calculation for each of them such that, for any given position on the board, we are able to calculate a single number (say, c_1 to c_6) representing how far each criterion is achieved or not (e.g., in the case of material balance, this might be simply a positive or negative integer, representing the number of my pieces minus the number of yours). To compare one position with another overall, we also need to assign some relative *weight* to these six criteria (say, w_1 to w_6), and we can then calculate a *value* for each position by calculating the sum:

$$v = w_1 c_1 + w_2 c_2 + w_3 c_3 + w_4 c_4 + w_5 c_5 + w_6 c_6$$

Choosing a move in a particular position now becomes a matter of examining the positions that would arise from all the analyzed *possible* sequences of n moves in that position (i.e., all of the n -move sequences that are available, given the rules of the game), calculating the *value* of each resulting position, and selecting whichever move yields the highest value position, using the same kind of adversarial search algorithm that we saw in section 1.7 above. Where several moves seem equally good (or

approximately so), we choose randomly among them. This has the twin advantages of making the program less predictable (and hence less easy to beat consistently), and providing a more-varied basis for learning.

Human judgement – informed by practical experience with the game – is involved here in identifying plausible criteria for assessing positions, which indeed might seem fairly straightforward for an expert player. But how much relative weight should we give to each of these criteria, when they tell in different directions? That is far more difficult to assess, but fortunately there is no need to rely on human judgement when selecting the crucial *weights* that feed into our move-selection algorithm, for this is where reinforcement learning can play a role. To start with a very crude method, we might initially assign w_1 to w_6 an equal value (say, 1,000). This defines our initial *current* strategy. We then make some random variation to these weights (e.g., by assigning a random change between +100 and –100 to each of them). This generates a *competing* strategy. Then we play a sequence of games between these two strategies (e.g., four games with *current* moving first, and four with *competing* moving first), either played out to completion or at least to a stage where both strategies agree that one player has a significant advantage. If *competing* does better overall than *current* in these games, we then replace *current* with *competing* (so that becomes our new *current* strategy).¹⁶ Either way, we then go on to generate a new *competing* strategy randomly as before. Continuing iteratively in this way, the weights in our *current* strategy are progressively refined (by simulated evolution, so to speak), and we hope ultimately to achieve something close to an optimal set of weights, and thus the best possible strategy of this kind. Depending on the game and suitable

choice of criteria, this sort of machine-learning technique can enable a program to learn to play better than its designer.¹⁷

¹⁴ However, as Samuel observed ("Some Studies in Machine Learning Using the Game of Checkers", 1959, p. 538), it is vital to extend the analysis beyond n moves when a capturing combination or piece exchange is in progress, since otherwise the position will give a misleading impression of the material balance. Hence analysis would normally continue until the resulting position is *quiescent*. Note in passing that because the counting of "moves" is potentially ambiguous (e.g. does the chess sequence "1. e4 e5 2. f4 exf4" count as two moves or four?), it is standard to use instead the term "ply", so that one move for one player is "1 ply", while two moves for each of two players (as in this case) is "4 ply".

¹⁵ Samuel devised 38 criteria (in addition to piece advantage), of which 16 would be used at any time within the position scoring function (1959, p. 544).

¹⁶ Of course there is scope for far more subtlety here – for example, the weights might be adjusted more or less, depending on how emphatically *competing* defeats *current*.

¹⁷ Note that in the context of a complex game, which cannot possibly be analysed to its end owing to the *combinatorial explosion* of possible lines of play, it is not possible to find the optimal weights by mathematical optimisation methods – practical experiment and feedback from the eventual game results is crucial.

1.9 Limits of Classical AI

We have now reviewed some of the main ideas and techniques of classical AI, illustrated mainly within the simple context of familiar rule-governed games. And indeed, it was within such contexts that classical AI would ultimately prove particularly effective, its most conspicuous success coming in 1997 with the sensational defeat of World Chess Champion Garry Kasparov – then considered by many to be the strongest human chess player of all time – by *Deep Blue*. In less well-disciplined contexts, however, classical AI techniques struggled to fulfil their apparent early promise. One serious difficulty was how to represent the characteristics and relationships of things in the world, and especially the "rules" that govern their

behavior. Even human experts find it very hard to elucidate the implicit background assumptions that guide their judgements, and spelling out our ordinary common-sense understanding of things proves to be extraordinarily difficult.¹⁸ A related issue was the so-called “frame problem,” keeping track of which aspects of a situation change when some action is performed, but also which aspects stay the same. Codifying these things in detail added yet more fuel to the sort of combinatorial explosion that we have already mentioned, which quickly resulted when attempts were made to apply the general-purpose search mechanisms of classical AI to problems of any serious complexity. This became all the worse if the data in question were at all ambiguous or uncertain, requiring yet more complex calculation if *probabilities* were to be taken into account (quite unlike the situation in games of “perfect information” such as chess and checkers, which had been such popular testbeds within classical AI). Acquiring real-time data about physical things was inevitably very error-prone, with computer vision systems relying on tailor-made algorithms first to identify such things as edges, shapes, textures, and colors, and then to synthesize these features into representations of familiar objects. Such synthesis also proved to be immensely difficult except within specific domains (for example where a limited range of objects could be expected, conforming to familiar templates and with known dimensions of variation).

In response to such limitations, the focus of much classical AI research moved – especially from the 1980s – toward so-called *expert systems*, which were designed to capture knowledge within specific domains, often using *logic programming* as a general-purpose reasoning mechanism.¹⁹ These systems were sometimes very successful, but

their functionality could not easily be generalized beyond those specific domains without hitting the same old problems. Indeed, AI systems in general were notoriously “brittle,” failing in unexpected ways when applied beyond their familiar narrow boundaries, or to situations that had not been explicitly foreseen by their designers. So, while progress continued in many areas where precise codification of facts and rules was achievable, and computational techniques were refined to achieve increasingly impressive results in those areas, a pessimistic consensus emerged that Alan Turing’s dream of a *general artificial intelligence* was likely to remain unfulfilled, at least for the foreseeable future.

¹⁸ For example, Douglas Lenat’s decades-long *Cyc* project, running from 1984, attempted to codify human common-sense knowledge, and by 2017 was reckoned to have absorbed over 1,000 person-years of effort, with around 25 million rules and assertions.

¹⁹ The boom in expert systems (also known as *intelligent knowledge-based systems*) was provoked in particular by the Japanese government’s massive investment in the *Fifth Generation Computer Systems Project*, which aimed to harness the logic programming language PROLOG as the inferential medium. Part of the impetus behind this came from the idea that a distinction could be drawn between a topic-specific “knowledge base” and a more general “inference engine”, making it easier to for “knowledge engineers” to extend the technology to new domains by gathering and codifying expert knowledge about those domains.

2.0 Neural Nets

Alan Turing had been contemplating, discussing, and writing about the possibility of AI as early as 1941,²⁰ while he was at Bletchley Park engaged in the even more important project of helping to defeat the Nazis. But what is often considered the first published contribution to modern AI development appeared soon after, in the *Bulletin of Mathematical Biology* of 1943, in the form of “A logical calculus of the ideas imminent in

nervous activity” by Warren McCulloch and Walter Pitts. This paper, intended as a contribution to theoretical neurophysiology, described the activity of neurons in the brain, and proposed that the behavior of networks of these neurons could be analyzed in terms of propositional logic.²¹ Ironically, in view of the later contrast that would be drawn between logical and neural approaches, the paper talks of neural nets as equivalent to Turing machines, and hence as limited by Turing computability (p. 129), saying that even “in psychology, … the fundamental relations are those of two-valued logic” (p. 131).

Donald Hebb’s 1949 book *The Organisation of Behaviour*, however, put aside such approaches from “the application of mathematics” and instead found inspiration in biology (pp. xi-xii). He suggested a way in which associative learning could take place within networks of neurons, based on a “neurophysiological postulate” that “*When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased*” (p. 62). Frank Rosenblatt later took this further, proposing a specific computational model for how the brain learns and stores information in which, rather than treating memories as encoded representations of experiences, he instead favored the Hebbian approach. The latter, he suggested, “stems from the tradition of British empiricism” and treats “the central nervous system … as an intricate switching network, where retention takes the form of new connections” (1958, p. 386). Rejecting any symbolic or algorithmic approach, he accordingly formulated his new model – which he called the Perceptron – “in terms of probability theory rather than symbolic logic” (pp. 387-8).

²⁰ See for example Jack Copeland, "Artificial Intelligence", p. 353 in his excellent edited collection of Turing's writings, *The Essential Turing* (Oxford University Press, 2004).

²¹ McCulloch and Pitts use the notation of Russell and Whitehead (whose influence was evident in §1.2 above), together with symbolism from Rudolf Carnap (a student of Gottlob Frege who had devised the foundations of modern logic). Their only other explicit reference is to Hilbert and Ackermann's famous work in theoretical logic, to which Turing's 1936 paper had been in part responding. So overall, although their paper concerns neuropsychology, it is absolutely immersed in the same logical background as Turing's.

2.1 Artificial Neurons

Rosenblatt's Perceptron led to a standard model of an artificial "neuron," which takes a number of numerical inputs (i_1, i_2, i_3 etc.) and outputs a single value v . Each input is multiplied by some weight (w_1, w_2, w_3 etc.) and these are added together (potentially with some constant bias term w_0) to make the net input. This is then fed into an activation function f to determine the output value v :

$$v = f(w_0 + w_1 i_1 + w_2 i_2 + w_3 i_3 + \dots)$$

The activation function is chosen to be non-linear, and often approximates to a step function that gets triggered when the net input is greater than some threshold (so the neuron's output is "all or nothing").²² This non-linearity is crucial if a network is to be able to learn non-linear behavior.

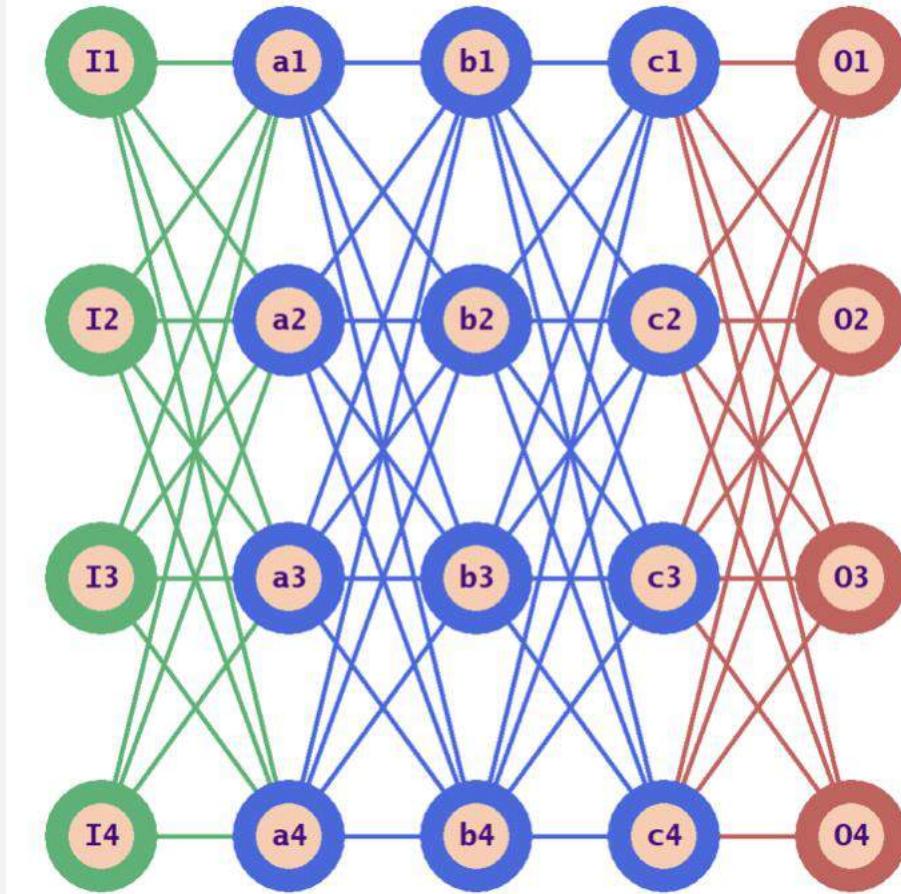


Figure 1.6

In recent years, artificial neurons of this general kind have been powerfully linked together in “deep” networks that have multiple additional layers between the input and output layers. **Figure 1.6** is a simplified diagram of a neural net with five layers:

the input layer (labelled “I”), three hidden layers (labelled “a”, “b,” and “c”), and the output layer (labelled “O”). Each neuron in the input layer is given a specific numeric level of activation, in such a way that the overall pattern of activations represents the input data. For instance, each neuron’s activation might store the color value of one of the pixels in an image (in which case, the actual number of input neurons will be far larger than the mere four shown here). In this example, every neuron in the input layer is connected to every neuron in the first hidden layer (i.e., the a-layer), so that the level of activation of the a-neurons will depend – in the way described above – on their activation function f_a as applied to the net input that they receive from the input layer. In the same way, every neuron in the second hidden layer (i.e., the b-layer) will depend on their activation function f_b as applied to the net input that they receive from the a-layer, and so on. Typically, the activation functions will be consistent across the neurons, but *the weights of the individual connections will vary*, thus influencing the propagation of activity through the network from the input layer, through the hidden layers, to the output layer O. It is these changing weights that represent the learning of the network. Hence in a successfully trained network, the weights will have evolved in such a way that the activation of the output neurons does indeed provide the desired response to the relevant input. In a network trained for recognition of digits, for instance, there are likely to be ten output neurons – one for each of the digits from “0” to “9” – whose activation should tell us which digit is represented by the image whose pixel values are reflected in the activation of the input neurons.

²² For example, the sigmoid and hyperbolic tangent functions have commonly been used. In deep networks, however, the Rectified Linear Unit (ReLU) function – which does not closely approximate a step function – has become more popular, partly because it is more computationally efficient. This makes $f(x)$ equal to x if x is positive, and zero otherwise.

2.2 Connectionism and its Early Challenges

Rosenblatt's work inspired widespread interest in artificial neural networks, and an appreciation of some of their particular strengths as learning mechanisms. But in 1969, Marvin Minsky and Seymour Papert published a fierce critique in their book *Perceptrons*, which severely undermined this enthusiasm. In particular, they showed that Perceptrons were incapable of learning some simple logical functions (notably anything depending on an "exclusive or"), thus apparently wrecking any prospect that they might provide a route toward "intelligent" information processing of any complexity. But it later turned out that this critique applied only to neural networks consisting of just a *single layer* (as in Rosenblatt's original design), and that more-complex functions could be learned by arranging artificial neurons (such as Perceptrons) into multiple layers. A network with a large number of such layers is called a deep neural network (see section 2.1).

Adding additional layers, however, significantly complicated the process of learning, and so this route was largely ignored until the back-propagation learning algorithm rose to prominence in the 1980s. To illustrate how this works, suppose that we have a network of the type described in section 2.1, with input data representing the pixels of an image, and ten output neurons intended to identify the pictured digit. In this setup, we would hope that if an image of a "6" is input, then the output neuron corresponding to "6" would have maximal activation (close to 1) and the other nine

would have minimal activation (close to 0). If the actual pattern of activation is very different, however, then learning is required, so we calculate how much a small change in the weights of the final (output) layer would contribute (positively or negatively) to improving the match.²³ Back-propagation is then the process of working back through the layers, doing a similar calculation for all the other weights in the network.²⁴ Once this has been done, the weights are adjusted by a small amount, in the direction that would bring improvement. This process is then repeated with other images – potentially thousands or millions of times – and so the network learns by iteratively adjusting its weights.

The discovery and success of back-propagation led to a resurgence of interest in what became known as “connectionism,” encouraged in 1986 by the two-volume *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* by David Rumelhart, James McClelland, and the PDP Research Group. This displaced the pessimism of Minsky and Papert, demonstrating how layers of simple interacting “neurons” could achieve learning of complex, cognitively relevant functions. The approach was attractive for at least four reasons. First, it was biologically inspired, and consequently had the potential to shed light on how we think. Second, and relatedly, it seemed to learn in much the way that we do, through association and feedback in response to success or failure. Third, this method of learning was quite general, and could apparently be applied to a huge variety of tasks and domains, without requiring any special programming. Finally, the storage of this learned information, rather than being explicitly represented, was obscurely “distributed” through the network of weights, which made the learning process more robust in response to “noisy” or ambiguous

data, better able to generalize, and less liable to break down entirely as the quality of data declines (i.e., it exhibited “graceful degradation”).

These advantages became well known, but enthusiasm for connectionism steeply declined in the 1990s as research failed to deliver the sort of practical breakthroughs that had been expected. But widespread turning away from neural networks would again prove to be premature, for their day would come once computer technology was able to deliver the computational power required by multilevel learning.

²³ Technically, this involves calculating the partial derivative of the loss function (which measures the magnitude of the mismatch) with respect to the given weight.

²⁴ Suppose we have already calculated the partial derivative of the loss function for every weight in layer n . Then back-propagation involves using the *chain rule* of differentiation to calculate – based on these layer- n results – the relevant partial derivatives for layer $n-1$. Thus we work backwards through the network.

2.3 Deep Learning Proves its Potential

Work on neural networks continued, with a particular focus on image and speech recognition, problems that very much played to their strengths. Consider again the familiar task of identifying digits (and letters) in print or handwriting – something of great potential practical value in applications such as sorting mail or processing bank checks. Here an important breakthrough came in 1998, when Yann LeCun and colleagues published their work on LeNet-5, a *convolutional* neural network designed for recognition of digits.²⁵ This is a deep network with a number of “convolutional” layers, each of which has the effect of applying a local filter (or “kernel”) repeatedly to points across the grid, which in this case is a grid of pixel values in a (grayscale) digitized

image. The filter consists of a small matrix of numerical “weights,” and each of these weights is multiplied by the pixel activation value at its position, with the sum of these products providing the corresponding activation value in the convolutional layer (at the filter’s central point). The weights within the filter matrix are initially set randomly, and subsequently “learned” in the same sort of way as other weights in the network. The virtue of such a convolutional layer is to enable efficient identification of local features in an image, which can then feed into the remainder of the network. LeNet-5 consisted of 8 layers, starting with the input layer that stores the pixel values of the digital image, then two sequences of a convolutional layer, followed by a “pooling” layer (which reduces the dimensions of the “feature map” by dividing it into 2×2 squares and averaging), then two further layers, and finally the output layer that classifies the digit.²⁶

Startling vindication of the amazing power of deep networks for image recognition came in 2012, when AlexNet – developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton – soundly thrashed the other competitors in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a high-profile international competition to identify images taken from the large ImageNet collection built up by Fei Fei Li.²⁷ The winning score in 2011 had been an error rate of 25.8%; AlexNet in 2012 had an error rate of 15.3%! Compared with LeNet-5, AlexNet was gigantic, nearly 100 times bigger with almost 900,000 neurons. The computational power required had become possible thanks to the rapid development of graphics processing units in response to the demand for high-quality animated computer games. To enhance speed of animation, these are designed to process image pixels *in parallel* (with recent

versions having as many as 10,000 cores), and this has turned out to be perfect for the sorts of huge matrix operations that are required by neural networks.

²⁵ Yann Lecun, Léon Bottou, Yoshua Bengio and Patrick Haffner (1998), "Gradient-based learning applied to document recognition", *Proceedings of the IEEE* 86 (11), pp. 2278–2324.

²⁶ In a bit more detail, the input image was 32×32 pixels, which after convolution with a 5×5 filter reduces to 28×28 (because the center point of the filter never reaches the pixels within 2 of the boundary). In fact 6 different filters were used here, so now there are 6 "channels" (i.e. 6 neurons for each point of the 28×28 "feature map"). In the next "pooling" layer the values are averaged pairwise, reducing the size to $14 \times 14 \times 6$, after which a further convolution with 5×5 filters reduces the size to 10×10 (though now with 16 channels), a pooling layer takes this down to $5 \times 5 \times 16$, and the final three layers have respectively 120, 84 and 10 neurons. Altogether this adds up to 9,118 neurons in the 8 layers.

²⁷ Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton (2012), "ImageNet Classification with Deep Convolutional Neural Networks", *Communications of the ACM*, 60 (6), pp. 84-90. Fei Fei Li started building ImageNet in 2006, crowdsourcing images and their "labels" (i.e. categorization) using Amazon's Mechanical Turk. By 2009 it consisted of 3.2 million images labelled into 5,247 categories; at the time of writing it has around 14.2 million images, in nearly 22,000 categories.

2.4 Deep Learning Beats Symbolic AI at its Own Game

Another major deep-learning landmark came in 2013, when the London company DeepMind (founded in 2010) announced its success in programming a deep convolutional network to learn to play vintage Atari video games from the 1970s, including *Breakout* and *Pong*, on which it learned to perform vastly better than an expert human. The remarkable thing about this achievement is that the system was not given any information about the *goal* of the games, the *significance* of the screen images, or the *effect* of the user's actions (e.g., pressing buttons). It had to learn what to do entirely on the basis of knowing that a certain number of distinct actions were available, and trying these out in response to pixel information about the changing images and the game score. Everything else was done by deep reinforcement

learning based on the score feedback, so in a sense, the system was teaching itself how to play from scratch.

In 2014 DeepMind was acquired by Google, and soon after another of its deep-learning programs – named *AlphaGo* – made even more headlines when it defeated European champion Fan Hui 5-0 in a match of Go, a game that until then had been considered too subtle and complicated for computer algorithms to master in the foreseeable future. The following year, *AlphaGo* sensationally went on to defeat World Go Champion Lee Sedol, 4-1. And in 2017 DeepMind revealed *AlphaZero*, a more generalized program capable of teaching itself new games, including both Go and chess. It did not quite do this “from scratch” in the way of the 2013 Atari program, for it was given the starting position, the moves of all the pieces, the various rules (including how to identify a win, draw, and loss), and had efficient recursive adversarial tree searching (as in section 1.7 above, but hugely optimized) built into it. But in contrast to sophisticated traditional AI programs like *Deep Blue* (which had scored a spectacular triumph by defeating Garry Kasparov 20 years before), *AlphaZero* learned how to play by competing against itself (in much the same way as the Nim program and Samuel’s checkers program) without any input from human experts or game databases. Self-training in this way for only a few hours, it was even able to defeat the 2017 world champion chess program *Stockfish*. And now *Deep Blue*’s 1997 achievement looked relatively modest, for it (like *Stockfish*) had been specifically programmed to play chess, with input from expert players and using algorithms that were finely tuned to reflect established theory. *AlphaZero* was an altogether more impressive system, teaching itself new skills to an even higher level than human

ingenuity had been able to achieve, and thus representing massive and potentially frightening progress toward Artificial General Intelligence.

2.5 The Inscrutability of Deep Learning

Part of the promise and threat of deep learning lies precisely in its ability to represent all kinds of information in ways that it works out for itself in response to training data and feedback. This information is stored implicitly within the weights of a neural network that consists of layers of artificial neurons. As we saw in section 2.1, the input layer of the network is set up to reflect the specific problem case, so for example the activation level of each input neuron might represent the state of one square in a game position (in which a move is to be chosen), or the color of one pixel in a digital image (which is to be classified). The output layer is set up to signal the corresponding solution, respectively the chosen move or the image label (e.g., “cat” or “dog” if the task is to classify images of pets). But in a deep network there may be many intermediate layers, with patterns of activation passing through the layers from the input layer to the output layer.²⁸ Beyond the input layer, the activation of each neuron depends on the inputs it receives from the neurons to which it is connected in the previous layer (possibly involving convolutions or other kinds of local processing, as briefly described in the case of LeNet-5 above). These inputs depend both on the level of activation of those previous neurons, but also – crucially – on the weights given to the relevant connections (as described in section 2.1). These weights are

adjusted during the reinforcement learning process (by back-propagation or refinements thereof), which typically involves going iteratively through the training data, assessing the resulting outputs, and gradually refining the relevant weights until a sufficient match between inputs and outputs has been achieved. Neither the weights nor the roles of the neurons in the intermediate layers are predetermined when the learning process starts. By the end, the immensely complex pattern of weights implicitly represents what the network has learned, but in a way that an unaided human will find impossible to interpret, and whose behavior they will be able to predict only by experience.

All this can look almost magical, enabling a deep network to solve problems that we have no idea how to address, and using calculations that are beyond our comprehension even when they have been discovered. But we should not be misled to suppose that the machines themselves “understand” what they are doing in any reflective way, and not only because they are completely non-sentient (and hence have no awareness or conscious understanding of anything). In conspicuous contrast to Classical AI techniques, their way of working is far more closely analogous to our own unconscious pattern-recognition than it is to how we think when explicitly calculating or reasoning about something. And we are unlikely to consider that we really understand *how* we ourselves operate when recognizing an animal as a dog, or a symbol as the letter “f” – we too learn these things by examples, building up neural structures of which we are completely unaware, and whose results we find hard to articulate. Likewise in expert chess-playing – often considered a paradigm of explicit calculation – many moves are decided by a sort of trained instinct, involving implicit

recognition of a position's characteristics and appropriate strategies, to the extent that a grandmaster will often choose a move with negligible calculation, and if asked, may initially have little more to say than "in this sort of position, that is the right move to play."²⁹ There is an obvious analogy here to a neural network's method of position evaluation by pattern recognition. Of course, the grandmaster also needs to be able to tell when explicit calculation is needed, for example when tactical combinations are imminent. But as noted above, *AlphaZero*, too, can perform explicit calculation, exploring options down the "game tree" of branching possibilities. Thus, within its carefully specified and rule-governed context, it was able to generate its own training data and feedback by playing lots of different games against itself, learning by experience which patterns were most conducive to success. But unlike Samuel's checkers program, discussed above, *AlphaZero* was not primed in advance with the relevant patterns – it learned them all for itself.

²⁸ In *recurrent* networks activation can also pass up the hierarchy of levels, but we ignore those here.

²⁹ This is not to deny that the grandmaster could probably expand on the strategic significance of the move given longer to think about it, but the point is that such deeper reflection would often not feature when initially choosing the move within the game.

2.6 The Dawn of Artificial General Intelligence?

Since 2022, we have witnessed a new "AI shock" that seems even more significant and widespread than any that has come before, with the release of OpenAI's chatbot *ChatGPT*, quickly followed by similar types of systems from rival companies. These again have been developed using techniques of deep learning, but this time

with colossal human textual input in the form of around 300 billion words from various sources on the internet. And now we suddenly – and unexpectedly – are faced with a technology that seemingly has the potential to pass the “Turing Test” in its full generality, to converse plausibly, flexibly, coherently, and informatively about a vast range of topics, and without relying on pre-prepared outputs.

How should we assess *ChatGPT* and its cousins from a theoretical point of view? The system was developed by applying statistical analysis on those 300 billion words of textual data to create a large language model (LLM) that – to simplify somewhat – records the probability that any given sequence of words (within its broader context) will be continued in different ways. And accordingly, *ChatGPT*’s primary method of working is to predict which individual words are most likely to follow in any particular textual context, and then to choose one of these words (but not always the most likely word, since an element of randomness enables it to respond differently if prompted again in the same way). That chosen word is then added to the existing text, and the next word chosen in the same way, and so on. Alongside the purely automated learning that generated the (reportedly) trillion or so parameters (i.e., stored probabilities) in the LLM (reflected in the deep network’s weights, as explained earlier), human users improved the system’s responses through supervised fine tuning, in which it was given a wide range of typical prompts (i.e., potential user inputs), together with suitable human-crafted responses. Then a stage of reinforcement learning from human feedback was applied, whereby the system generated a range of responses for each given prompt, and humans assessed the relative suitability of those responses. This feedback was then statistically analyzed to

generate a “reward model,” which could in turn be used to rate responses in general, and thus assist in selecting the most appropriate.

This is, of course, a hugely simplified description of how *ChatGPT* was created and how it works. But two points here are particularly worthy of emphasis. First, its method of working is based entirely on imitating – with variation – the sorts of responses that it found in the massive textual resources that were used to train it. Second, the information it stores implicitly within its trillion or so network weights has been tuned to reflect *the characteristics of the textual data*, rather than *the characteristics of whatever domain the text might concern*. Hence, for example, if you play chess against *ChatGPT* using a common opening (e.g., a main line Sicilian Defence), it will initially respond with some of the same sensible moves that dominate its training data, and thus give the impression that it understands what is happening on the chessboard. But it stores no internal model of the board position, has no mechanisms of analysis or “lookahead” (unlike *AlphaZero*), and is therefore subject to absurd errors, such as overlooking obvious captures, or allowing your bishop to capture one of its own pieces even when the diagonal between them is blocked. Play a less common opening, and the problems will appear much more quickly (e.g., losing track of a pawn on the fourth move of a King’s Gambit). Ask it to solve a simple chess puzzle and it will likely have no clue. It seems likely that before long these particular foibles will be dealt with by linking *ChatGPT* to a dedicated chess engine that really has a model of the board together with relevant analytical algorithms. But the key point here is that its apparent *general intelligence* – its ability to converse “intelligently” about a vast range of topics where it has not been augmented with specific assistance – is based on the

illusion that it has some internal understanding of what is being discussed. But unless the language model in some domain is able to generate indirectly a reliable model of the reality (which looks most plausible in domains that are primarily conceptual or expressive), it will have nothing like such internal understanding, and can quite appropriately be described as a “stochastic parrot.”³⁰

³⁰ The term “stochastic parrot” was coined by Emily M. Bender in 2021. Recent work by my Oxford colleagues Philipp Koralus and Vincent Wang indicates that the most recent generations of ChatGPT have become better at mimicking *good* human thinking, but also *fallacious* human thinking (arXiv:2303.17276v1). This again emphasises how the system is working by modelling *human language* rather than *the relevant domain*.

3.0 Machine Learning and Its Risks

We have now seen in general terms how AI rose to prominence, initially as a relatively conventional branch of computer science (with well-understood data structures and algorithmic methods), but then took a radically different trajectory with the dramatic, accelerating, and previously unexpected rise of contemporary machine learning over the last quarter-century. Understanding this contrast is crucial to appreciating why AI is now seen as posing quite distinctive and novel risks in a way that previous computer systems did not. But before moving on to review these risks, it will be helpful to impose a bit more structure by distinguishing among four main types of machine learning, whose methods, benefits, and potential risks differ quite significantly.

3.1 Four Types of Machine Learning

So far, we have focused mostly on **reinforcement learning**, which is typically involved in situations where a sequence of actions or decisions is to be made within a changing environment to achieve some goal. Such learning involves “trial and error,” with positive feedback where things turn out well, and negative feedback where they turn out badly. This is most easily illustrated in the case of competitive games, such as the Nim example in section 1.3 and the Atari, *AlphaGo*, and *AlphaZero* game-learning programs discussed in section 2.4. Reinforcement learning is also valuable in many other domains, for example:

- Robotics – teaching robots or autonomous vehicles to move and manipulate objects effectively, to interact with humans and learn from human feedback
- Healthcare – monitoring effects of treatment both for groups and individuals
- Finance – trading algorithms
- Optimization – broadly, any area in which optimization problems can be addressed by automated learning from experience

Second, there is **supervised learning**, where the aim is to learn from a set of *labelled examples* (the *training data*) either how to categorize further examples of the same kinds of things, or to predict some characteristic. As with reinforcement learning, this was long part of the repertoire of classical computation, in the form of methods such as linear

regression, decision trees, and support vector classification. But the kind of supervised learning that is today seen as generating special risks is derived from *deep learning*, particularly because of its inscrutability and hidden basis (see section 3.3). The most prominent example of such deep learning applied to problems of *categorization* is perhaps the classification of images, as we saw in the case of LeNet-5 and AlexNet, and which has since become ubiquitous in applications from recognition of handwriting, faces, and road signs, to analysis of medical scans and visual scenes. Other applications include *speech recognition*, *machine translation*, and *recommendation systems*. Learned *prediction* (often called *regression* when the relationships involved are relatively straightforward) operates differently, in that now the “labels” are typically numerical values rather than categories (though the numbers can represent *probabilities* of category membership), and the aim of the learning process is to be able to predict the corresponding value for new instances. Here prominent examples include:

- Financial analytics of likely stock prices or future demand
- Asset valuation of house prices
- Credit scoring probability of default
- Weather prediction of wind speed, rainfall, or temperature

This can also be applied to domains already mentioned (e.g., medical diagnosis and prediction), and broadly, to any area in which a wealth of past data can be harnessed to predict future outcomes.

The third main type of learning, of which we have yet to discuss any examples, is **unsupervised learning**, where the aim is to identify patterns within the given data, but without the help of human input in the form of specified “labels.” The two most prominent forms of such learning focus on cluster analysis and dimensionality reduction. Cluster analysis is a learning technique in which groups of similar objects are automatically identified without needing a training set. Dimensionality reduction, on the other hand, is a technique in which the main “dimensions” of variation among the objects are identified, enabling their range and relationships (e.g., comparative closeness) to be more easily grasped. This yet again is a form of learning that can be done using classical methods as well as using deep networks, and here – for ease of comprehension – we look at one of those classical methods that nicely illustrates both cluster analysis and dimensionality reduction. This is the important technique of *Principal Component Analysis* (PCA), and we briefly review two such applications, first to the geographical layout of a village, and then to the analysis of written texts.

The fourth type of machine learning is **semi-supervised learning**, which is a combination of supervised and unsupervised learning. It is used in cases where only part of the available data is labeled. The unlabeled data is used to determine patterns within the explanatory variables, or it is fitted with “pseudo-labels” determined by estimating a model on the labeled part of the data. We discuss this technique in Module 2.

3.2 Examples of Unsupervised Learning (Principal Component Analysis)

To aid our imagination, we start with an example from *physical* space rather than a virtual space of numerical values. Suppose, then, that we plot the position of houses in a village, starting from coordinates in terms of latitude, longitude, and height above sea-level. PCA will then transform these coordinates into a different framework, so that the various dimensions – instead of lining up with latitude, longitude, and height – will instead line up with *whatever directions best discriminate between the data items*. Thus, if the village has grown up along a fairly straight road on mainly flat land, then the dimension that gives most discriminating information (i.e., along which the houses are most spaced out) will be horizontal and in the direction parallel to the road – so that is our first “vector.” To provide a proper coordinate frame, the second vector must be orthogonal (i.e., at right-angles) to the first,³¹ and to preserve most information, we choose another *horizontal* vector (because the houses will vary more in distance from the road than they do in height above sea-level or in any diagonal direction).³² Our third and last vector must be orthogonal to both of the others, and hence must be vertical. Note that unless the road runs north-south, the resulting coordinate frame will be different from the original latitude/longitude/height frame. The coordinates of each house within the new frame can be calculated from the original coordinates, and we then imagine looking into the space from different directions. To see most information, we want to view distances according to the first

two coordinates, so we look from above – as indeed maps standardly do.³³ Thus the PCA analysis yields exactly what we would wish: the most informative presentation of the layout of the village.

For a very different example, we can look to classic work in stylometry by John Burrows, published in 1982, and replicated in [Figure 1.7](#) using the *Signature* software system.³⁴ Burrows set out to explore whether "The Memoirs of a Lady of Quality" (denoted as "Quality" in [Figure 1.7](#)), published in 1751 as part of Tobias Smollett's *Adventures of Peregrine Pickle*, was in fact written by Smollett. With this in mind, he applied PCA to the relevant texts of fifteen early eighteenth-century novelists, including Smollett himself. This method starts by counting the occurrences of all words in the entire textual corpus of these authors, ordering them by overall frequency, and selecting the most common 50 ("the," "of," "and," "I," "my," "to," "a," etc.) for further processing. Then the fifteen authorial texts – together with the "Memoirs of a Lady of Quality" – are individually re-read, in each case counting the occurrences of these common words and dividing by the text's total word-count. This yields 50 fractional values for each of the 16 texts, representing the relative frequency of each word within it (e.g., if a word occurs 61 times in a text of 10,000 words, then that text's value for that word is 0.0061).

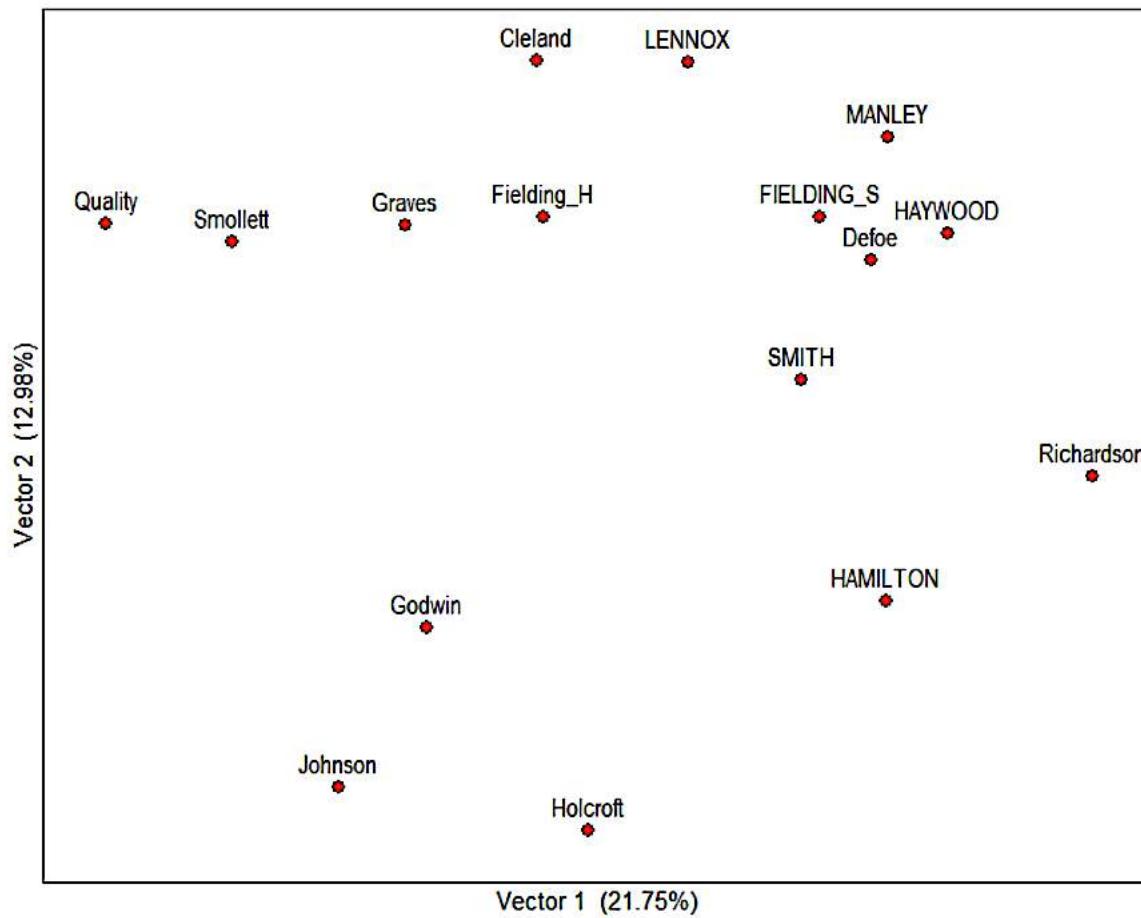


Figure 1.7

We then imagine these 16 texts placed within a 50-dimensional space (one equally extended dimension for each word), in which the 50 coordinates for each text are proportional to these relative frequencies. Thus, each of the 16 texts is represented

by a point within this space, in such a way that two texts whose points are close together have broadly similar patterns of word occurrence (i.e., the same words tend to be relatively common in both texts, and the same words tend to be relatively uncommon, as compared with the other 14 authorial texts). Just as with the village map example above, PCA now gives us a way of looking into that space to see a two-dimensional projection of it *from whichever direction gives most information* – that is, whichever direction shows the texts maximally spaced out in a two-dimensional projection (so that relative similarities and differences can be most clearly seen).³⁵ Extensive calculation is needed to discover which direction gives the best projection, something that would be hard and extremely tedious to do by hand.³⁶ Hence, this technique is commonly considered as falling under the umbrella of “machine learning,” although as Burrows’ example makes clear, it significantly pre-dates the recent machine-learning revolution.

What is striking about this method – and further illustrated also by other examples in Burrows’ interesting paper of 1982 – is that it somehow manages to cluster the authors in such a way as to identify patterns of similarity between them, even though it has been done entirely automatically. In the current case, for example, we see clustering of the female authors (whose names are capitalized), with “Memoirs of a Lady of Quality” both distant from this cluster, and very close indeed to Smollett himself, thus yielding a fairly decisive positive verdict on his authorship. But although this is clear, it is hard to understand exactly what these similarities and differences are. The graph represents them as distances within a vector space, but there is no simple way of explaining what “Vector 1” and “Vector 2” signify, because they do not

represent simple word frequencies, but rather, complicated linear functions of 50 different word frequencies.

To sum up, then, PCA provides a powerful method of *dimensionality reduction*, whereby a large array of points in multidimensional space can be “reduced” to a graph in two dimensions, while effectively representing the most significant measures of closeness and distance within that multidimensional space. For the same reason, it can highlight clusters of similar items to serve as a basis for future investigation. And even though it might often be hard to pin down exactly the basis of this similarity, PCA does not reach the extreme level of inscrutability that we find with deep network learning, because here at least we can understand in general what the dimensions of variability represent, and explore their basis relatively straightforwardly.

³¹ Technically, the mathematical processing involved in PCA identifies a sequence of *mutually orthogonal vectors* that form a *basis* of the multi-dimensional space, ordered so that those vectors that preserve most information about overall distances come before those that are less informative. For a rigorous but relatively accessible presentation of the mathematics involved and some applications, see Ian T. Jolliffe and Jorge Cadima, ‘Principal component analysis: a review and recent developments’, *Philosophical Transactions of the Royal Society A* 374 (2016).

³² Note that if the ground slopes systematically in any direction, then this second vector will also slope accordingly, because the sloping distance between the houses will be greater than their horizontal distance.

³³ Other possibilities would be to focus on the first and third coordinates, thus looking from the side, or on the second and third, thus looking from one end of the road; but each of these gives far less discriminating information about the layout of the houses than looking from above.

³⁴ John Burrows, “Computers and the Study of Literature”, in Christopher S. Butler (ed.), *Computers and Written Texts* (Oxford: Blackwell, 1992), pp. 167–204. An early version of the *Signature* software is available from <https://www.philocomp.net/texts/signature.htm>. Also documented there are the present author’s stylometric investigations of a malicious rumour regarding Barack Obama’s *Dreams from My Father* shortly before his 2008 election, and the authorship of J. K. Rowling’s pseudonymous *The Cuckoo’s Calling* of 2013. Both of these adventures, rather bizarrely, ended up on the front page of the *Sunday Times* newspaper!

³⁵ If this seems hard to imagine, start with just a three-dimensional space (e.g. based on relative frequencies of ‘her’, ‘him’, and ‘me’ within the texts), and imagine looking into it from different directions. Here it might help also to compare with the more familiar three-dimensional example of village houses, as given above.

³⁶ John Burrows did the calculations with a 1972 statistical package called *Minitab*. That is clearly preferable to hand calculation, but still far more tedious and error-prone than using software designed for the purpose.

3.3 Risks of Inscrutability

The inscrutability of deep network machine learning is problematic for a number of reasons, which can be understood in general terms given what we have discussed. First, when classification or prediction is based on supervised learning with deep networks, it is likely that any bias that exists in the labelling of the training data (e.g., owing to historical prejudice against particular categories of people) will be implicitly learned by the model and perpetuated, *but in such a way that its presence is hidden and hard to eradicate*. Suppose, for example, that a company has for many years prejudicially favored men over women, and it decides to create a computational tool to filter job applicants, based on data about its past recruitment (e.g., in the form of applicants’ CVs and their acceptance or rejection). Clearly the resulting model is likely to exhibit a significant bias in favor of men, because such a bias assists it in correctly predicting the “labels” in the training data that it is attempting to match. Nor can this bias be removed by the simple expedient of removing information about the applicants’ gender from the CVs before processing them, because a significant proportion of the other information in those CVs is likely to act as an effective “proxy” for gender. Lots of features of our lives, from the subjects we choose at school, to the sports and

hobbies we pursue, to the careers we embark on, are strongly correlated with gender, and in societies that exhibit gender bias, these correlations are even stronger. So even if *explicit* gender information is removed from the training data, the trained models are likely to learn other correlations that have a similar effect. Another example of this type would be if financial decisions (e.g., approving a loan) have traditionally been racially biased, within a city whose areas are – not coincidentally – radically segregated to a significant extent. If information about applicants' race is completely expunged from the training data, the learning algorithm may instead come to base its judgements on the strong correlation between postal code and historical decisions, thus again unwittingly perpetuating that long-standing prejudice. Another related risk is to *privacy*, because this deep implicit entanglement of complex information within the learned network can easily hold clues to personal characteristics that we would prefer to keep secret. In 2013, for example, researchers at Cambridge University found that Facebook "Likes" could be used "to automatically and accurately predict a range of highly sensitive personal attributes including: sexual orientation, ethnicity, religious and political views, personality traits, intelligence, happiness, use of addictive substances, parental separation, age, and gender."³⁷ This also carries significant risks of *manipulation*, in both the commercial and political sphere, because those who are able to identify our personal characteristics and foibles may also be able to play on them through the now familiar phenomenon of targeted advertising.

All of these problems arising from the inscrutability of deep network models are rendered more intractable because their complex incomprehensibility also makes it

very hard to identify – with a view to eradication – the source of any such hidden clues or biases. Unlike a traditional expert system, for example, the deep network model can provide no *explanation* of how it reaches its decisions and predictions. The true explanation is hidden in that vast web of weights,³⁸ possibly billions in number, which cannot possibly be rendered humanly comprehensible. Finding ways of dealing with this and achieving some sort of explainable AI while retaining the power of machine learning is a major challenge.³⁹ Companies that employ deep learning systems without addressing these issues carry huge reputational risk when things go wrong.

³⁷ See Michal Kosinski, David Stillwell, and Thore Graepel, "Private traits and attributes are predictable from digital records of human behavior", *PNAS* 110 (2013), pp. 5802-5.

³⁸ Plus convolutions, poolings, biases, and activation functions etc., as sketched in notes 18 and 21 above.

³⁹ This challenge is sometimes expressed as that of achieving interpretable machine learning by finding ways of transforming *black box models* into *glass box models*.

3.4 Risks of Over-Reliance

Inscrutability can also engender over-reliance, as we come to depend on a system whose mode of operation seems completely mysterious to us, and yet appears to be impressively expert at what it does – far faster (and typically more confident) than we could possibly aspire to be, yet also far more sophisticated in its processing and comprehensive in the data on which that processing is founded. This can undermine our autonomy as responsible individuals and lead to serious dangers as we neglect to develop or apply our own judgment and fail to realize when the system is getting

things wrong. As noted, neural network models have long been considered more robust than classical AI alternatives in the context of noisy, ambiguous, or incomplete data. But to the contrary, it has recently become clear that deep learning models are commonly vulnerable to deliberately designed “adversarial” examples, whereby – for example – two images that differ imperceptibly (as far as a human is concerned) may be categorized by the system quite differently, and even very confidently. One notorious, and very worrying, illustration of this was revealed by researchers at New York University who discovered that adding a sticker to road signs could cause them to be misidentified (so that a stop sign, for example, was taken to be a mere speed limit sign).⁴⁰

The dangers of over-reliance can be seriously exacerbated by the hype surrounding conspicuous developments in AI, as for example the recent excitement about *ChatGPT* and its supposed general intelligence. One such area of concern is software development, where the apparent ability of *ChatGPT* to perform impressively in programming exercises might tempt companies to rely on it for code production, and thus hugely economize on their software engineering teams. But as in routine chess openings (see section 2.6), the reason why *ChatGPT* does so well in those exercises is not that it rigorously analyzes some internal model of what is going on in the code that it generates, but rather, that its own implicit textual model is based on analysis of the many millions of programming examples that it has been given, and it accordingly generates code by mimicking, with what appear to be appropriate textual variations of the dominant patterns. Therefore, if you have a programming task that is very standard, or which deviates from a standard task in relatively common ways

(e.g., when teachers have devised non-standard programming tasks for their students, such as sorting an array of numbers but with odds preceding evens), then *ChatGPT* might well come out with a correct solution. But if the task involves any nuances (or combinations of factors) that are less common or entirely novel, then there is a serious risk that you will end up with code that has the double disadvantage of looking very plausible – and quite possibly works well in most cases – while actually being incorrect, thus making the errors especially hard to identify.⁴¹ None of this is to deny that in expert hands, *ChatGPT* can be a useful programming assistant, providing quick and targeted search through the mountains of online programming resources to generate “first-draft” answers that are often very helpful. But the use of *ChatGPT* in programming by those who are unaware of its foibles, or insufficiently expert to check carefully what it produces, carries the risk of generating false confidence, both in the code that it generates and in the programmers who rely on it (whose lack of understanding could be masked by their success in turning out plausible code). Programmers may find themselves spending a considerable proportion of their time fixing “bugs,” so the use of generative AI to generate code doesn’t necessarily guarantee quality code or time savings and could expose a firm to financial risk if managed poorly.

⁴⁰ See Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg, “Badnets: Evaluating Backdooring Attacks on Deep Neural Networks”, *IEEE Access*, 2019.

⁴¹ It is well known that current large language models are prone to so-called “hallucination”, in which they generate textual information which is completely false (yet often plausible). Recent work by Ziwei Xu, Sajay Jain and Mohan Kankanhalli (<https://arxiv.org/abs/2401.11817>) demonstrates, as their title states, that “Hallucination is Inevitable: An Innate Limitation of Large Language Models” (though this does not preclude avoidance of such errors by use of other systems in combination with the LLM).

3.5 Risks to Individuals, Organizations, and Society

The development and deployment of artificial intelligence exposes individuals, organizations, and society to a wide spectrum of risks.

Individual risk: Risks posed to individuals vary widely and may originate with organizations or institutions with whom they interact or through their own behavior. Individuals routinely interact with institutions and organizations (e.g., banks, insurers, medical care providers) where decisions are made that have an impact on the individual. If AI systems contributing to that decision making are poorly designed or trained using inappropriate data, then one could suffer because of faulty, biased, or unfair decision making. In terms of how individuals interact with technology, automation bias, or the tendency of humans to over-rely on automated systems, can lead to complacency and reduced vigilance, which can potentially impact an individual in the form of reduced decision-making autonomy or a risk to safety and well-being. AI algorithms that draw on large amounts of information about individual behaviors, activities, and preferences create the potential for manipulation (e.g., by advertisers or interest groups) by allowing them to make predictions about future actions and influence our behavior. Inferences made from that information can also threaten individual privacy.

Organizational risk: Risks posed to organizations could come in a variety of forms. As cited previously, over-reliance on AI, and the false confidence that it engenders, can

lead to bad decision making, customer dissatisfaction, and/or commercial loss. Depending on the impact of the over-reliance, an organization could also be exposed to reputational, regulatory, or legal risk. Reputational risk could arise as a result of AI practices that hurt individuals or groups (e.g., a poorly designed AI algorithm that lacks explainability, breaches privacy, or yields biased results). Although AI-specific regulations are still emerging, business practices that rely on AI are still subject to existing privacy laws and model governance regulations, so in cases where such practices hurt individuals or groups, organizations are potentially subject to regulatory risk and legal risk.

Societal risk: As with prior industrial and technological “revolutions,” advancements in AI create the potential for risks at the societal level. These include job losses in fields where AI-driven applications take the place of human workers, a widening wealth gap between those who have the skills to work alongside AI and those who do not, and the exacerbation of global inequality between countries well positioned to take advantage of AI (i.e., wealthier, more industrialized countries) and those less well positioned. As with many technologies, AI can be leveraged by bad actors for ill. One such example is the creation and spread of misinformation (e.g., deep fakes) via generative AI to mislead or influence public opinion. Finally, discussions of AI in recent years have often raised a fear of “existential” risk to humanity, especially in relation to concerns about superintelligence and the possibility that an AI system, such as an autonomous weapons system, will make decisions autonomously and against the interests, values, or priorities of its creators or humanity. It should be

noted, however, that there is currently no consensus regarding the nature or extent of existential risk posed by AI.

Navigating the Remainder of this Course

Our exploration here has laid the groundwork for understanding the opportunities and challenges of AI. Yet, true power lies not just in abstract concepts, but in the ability to apply and govern these emerging technologies effectively. Therefore, the remainder of this course focuses on two key areas: mastering the AI toolbox itself and navigating the ethical, regulatory, and risk-based complexities where those tools are deployed.

In our **Tools and Techniques** module, we take a deeper dive into the core building blocks of AI. We categorize machine learning algorithms for supervised, unsupervised, and reinforcement learning tasks. You'll become fluent in data preparation techniques, transforming raw information into insights fit for modeling. We cover feature selection, visualization, and the nuanced art of balancing model accuracy with explainability – a vital skill for ensuring AI aligns with critical business decision-making.

Although mastering individual tools is vital, our **Risks and Risk Factors** module takes a broader perspective. We dissect the potential unintended consequences of AI, from the amplification of existing biases to new threats to autonomy and safety. You'll gain strategies for mitigating risks related to fairness, transparency, and the complex ways

errors can translate into real-world harm. As AI applications evolve, we confront the reputational risks stemming from public mistrust and the importance of proactive AI governance for building a trustworthy organization in the face of potential missteps. In the domain of **Ethical and Responsible AI**, we move beyond risk mitigation toward principled implementation. Here, we analyze frameworks from consequentialism, deontology, and virtue ethics, applying them to real-world scenarios with AI at their core. You'll gain a thorough understanding of the evolving legal landscape, including the landmark EU AI Act and emerging US regulatory initiatives. This equips you to be proactive as regulations evolve, not merely reactive – a key leadership quality in a field where the rules are still being shaped.

Governance will be a recurring theme – not as a bureaucratic burden, but a critical lever for successful implementation. In the **Governance** module, you'll gain a holistic picture of both data and model governance. You'll learn strategies like designing a robust data strategy, ensuring data quality, and establishing clear processes for model development, testing, and ongoing monitoring. These principles ensure your AI efforts yield reliable insights, not unexpected liabilities or operational weaknesses. Finally, we explore a variety of **Case Studies**, **Practitioner Perspectives**, and other content to help bring these concepts to life. The case studies examine specific applications in finance and risk management. The other content illustrates how machine learning transforms traditional risk modeling, while discussing practical strategies to ensure explainability and robustness, particularly in high-stakes, regulated environments. Throughout this course, the focus is on application and action. Your goal is not to become a data scientist – but rather a business leader who can speak their language

and bridge the gap between technical possibilities and real-world business needs. Mastering the interplay among AI, ethics, and risk management is critical to unlocking the transformative potential of this powerful technology in a sustainable, responsible, and ultimately profitable way.

AI and Risk - Introduction and Overview: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

1. What are the four most basic forms of machine learning?

Reinforcement learning, supervised learning, unsupervised learning, and semi-supervised learning.

2. What are some reasons why inscrutability of deep network machine learning is problematic?

The presence of bias can be hidden and hard to eradicate; the risk to privacy, because the deep implicit entanglement of complex information within the learned network can easily hold clues to personal characteristics that we would prefer to keep secret; risks of manipulation, in both the commercial and political sphere, because those who are able to identify our personal characteristics and foibles may also be able to play on them through the now familiar phenomenon of targeted advertising.

3. What is the risk associated with overreliance on AI systems?

Overreliance on AI systems, also called automation bias, can lead to complacency and reduced vigilance, which in turn can undermine our autonomy as responsible individuals. This can create the potential for a dynamic in which one neglects to develop or apply one's own judgment and fails to realize when the system is getting things wrong.

4. True/False: The fact that AI-specific regulations are still emerging results in organizations having little or no AI-related regulatory risk.

False. Although AI-specific regulations are still emerging, business practices that rely on AI are still subject to existing privacy laws and model governance regulations, so in cases where such practices hurt individuals or groups, organizations are potentially subject to regulatory risk and legal risk.

5. In the context of reinforcement learning, why might Lookahead be applied?

Lookahead becomes essential when our aim is to form a multistep plan to achieve some goal, within a context where there is a very large number of possible situations overall, but with a relatively constrained and predictable range of options in any particular situation.

6. Differentiate between cluster analysis and dimension reduction.

Both are unsupervised learning techniques. Cluster analysis is a learning technique in which groups of similar objects are automatically identified without needing a training set. Dimensionality reduction, on the other hand, is a technique in which the main "dimensions" of variation among the objects are identified, enabling their range and relationships (e.g., comparative closeness) to be more easily grasped.

7. What are the advantages of principle component analysis (PCA)?

PCA provides a powerful method of dimensionality reduction, whereby a large array of points in multidimensional space can be “reduced” to a graph in two dimensions, while effectively representing the most significant measures of closeness and distance within that multidimensional space. For the same reason, it can highlight clusters of similar items to serve as a basis for future investigation. And even though it might often be hard to pin down exactly the basis of this similarity, PCA does not reach the extreme level of inscrutability that we find with deep network learning, because here at least we can understand in general what the dimensions of variability represent, and explore their basis relatively straightforwardly.

8. True/False: There is currently broad consensus regarding existential risk posed to humanity by AI.

False. There is currently no consensus regarding the nature or extent of existential risk posed by AI.

Module 2: Tools & Techniques

Chapter 1: Introduction to Tools & Techniques

Learning Objectives

Machine learning (ML) is an umbrella term used to cover a range of techniques for training models to recognize data patterns for a variety of applications, including prediction and classification. This chapter offers a high-level introduction to ML techniques and applications, the potential benefits, and considerations for implementation.

After completing this chapter, you should be able to:

- Differentiate between machine-learning techniques and classical econometrics.
- Differentiate among unsupervised, supervised, semi-supervised, and reinforcement learning models.
- Distinguish between different data types.
- Describe how to encode categorical variables.
- Describe how to clean data and the benefits of cleaning.
- Describe data preparation techniques and their benefits.
- Apply transformations to a set of data.
- Discuss how principal components analysis (PCA) is used to reduce the dimensionality of a data set.
- Explain the differences between the training, validation, and test data subsamples, and how each is used.

1.1 Machine Learning

Machine learning (ML) is an umbrella term that covers a range of techniques in which a model is trained to recognize patterns in data to suit a range of applications, including prediction and classification. As an aspect of artificial intelligence (AI) and a set of tools for data analysis and building models, it has gained significant popularity in recent years as the techniques themselves have developed alongside advances in computing power and huge increases in the amount of data available to analysts.

Machine learning is used everywhere, in applications including stock selection, image recognition, game playing, operation of autonomous vehicles, medical research, and many risk-management situations, such as credit scoring and fraud detection.

Machine learning influences virtually all financial decisions. In fact, the sheer amount of information available nowadays and the complexity of the problems under study have exposed the limitations of traditional statistical techniques.

Machine learning offers a large and growing suite of techniques that can be useful for analysis and often offers advantages over traditional econometrics methods in areas such as:

- **Handling big data.** Ranging from clustering algorithms to neural networks, there are ML approaches that can handle very large amounts of data more

effectively than traditional econometric methods. These techniques are highly useful for making use of available data that is growing exponentially in volume, as well as an increase in dimensionality with the rise in digitization of the global economy.

- **Handling non-linearity.** There are many non-linear relationships and patterns in data that traditional econometric techniques might miss, which machine learning tools, such as decision trees, random forests, and neural networks, can help identify and model.
- **Reducing dimensionality.** Machine learning tools such as principal components analysis (PCA) and feature selection can be particularly helpful for prediction and classification tasks when the number of variables is large and when the number of variables is greater than the number of observations, a situation with which standard econometric methods tend to have great difficulty.
- **Handling missing data.** There are ML techniques, such as k-nearest neighbors, that can be used to handle missing values in large data-sets in a flexible way.

1.1.1 Machine Learning, Classical Statistics, and Econometrics

The paradigm that ML provides for data analysis is very different compared to classical statistics and econometrics. For the latter, it is usually hypothesized that the

data-generating process can be approximated based on some economic or financial theory. The analyst decides on the model and the variables to include, and the computer algorithm's role is generally limited to estimating the parameters and testing whether they are significant. Based on the results, the analyst decides whether the data support the pre-specified theory. In contrast, machine learning treats the data-generating process as unknown and uses techniques such as regularization to select the relevant predictors.

Model selection is at the heart of the empirical design in ML applications, and tuning, the process of searching through many models to identify the top performers, is a common characteristic of all ML methods. In contrast to traditional statistics, the focus is not on inference, but on the ability to produce reliable predictions out-of-sample.

Therefore, tools such as measures of out-of-sample prediction accuracy and an understanding of the bias-variance trade-off play more important roles than traditional statistics such as R-squared, t-values, and p-values¹.

The terminology in ML also tends to deviate from that of classical statistics. For instance, data points are called examples or instances; the dependent variable is called the target, response variable, output, or label (in classification problems); and the independent variables are called features, inputs, or signals. The estimation process is often called training or learning and, accordingly, the estimator is generally called a learner (or a classifier, in classification problems) or simply an

algorithm. **Table 1.1** summarizes the main terminological differences between machine learning and traditional statistics.

Table 1.1 Differences in the terminology between machine learning and traditional statistics / econometrics.

| Statistics | Machine Learning |
|--|---|
| Data point | Example, instance |
| Dependent variable, explained variable, predicted variable, regressand | Output, outcome, label, target, response variable, ground truth |
| Independent variable, explanatory variable, predictor, regressor | Feature, signal, input, attribute |
| Estimation | Training, learning |
| Estimator | Learner (classifier), algorithm |

¹ In regression analysis, R^2 (the coefficient of determination) is the proportion of the variation in the dependent variable that is explained by the independent variables. Both t -statistic and p -value are used to test the significance of the coefficients. The t -statistic is the ratio of the regression coefficient to its standard error. The p -value tests the null hypothesis that the regression coefficient for an independent variable is zero. A low p -value indicates that the coefficient is unlikely to be zero.

1.2 Types of Machine Learning

Machine learning is a broad area that covers several algorithms for classification, pattern recognition, prediction, and decision making. Machine-learning methodologies

can be categorized as unsupervised, supervised, semi-supervised, and reinforcement learning.

Unsupervised Learning

Unsupervised learning is concerned with recognizing patterns in data. In unsupervised learning problems, for each observation we have a vector of features, but no corresponding output value to predict. In such a situation, we can try to understand the relationship between the variables or between the observations. Unsupervised learning problems generally involve clustering the data or finding a small number of factors that explain the data.

By definition, unsupervised machine learning is not used to generate predictions and, at first sight, might appear not to be very worthwhile. However, it can be an extremely useful technique to characterize a dataset and learn its structure. For example, unsupervised learning is sometimes used for anomaly detection when a bank is trying to identify the features of transactions that might be suspicious and worthy of further investigation. In such cases, the bank cannot identify *a priori* which variables are important, but highlighting the characteristics that might make some transactions distinct from the others can constitute valuable information that the bank can then use to train and develop a fraud model. Another application of unsupervised learning consists of using clustering algorithms to identify groups of stocks that share similar characteristics (e.g., in terms of size, liquidity, market exposure, etc.).

Dimensionality-reduction techniques such as principal component analysis (PCA; discussed in section 1.3) are also examples of unsupervised learning.

Supervised Learning

Supervised learning is concerned with prediction and classification. When the value of a numerical variable (e.g., the price of a house) is to be predicted, this is called a *prediction* problem. When an observation is to be classified (e.g., a loan is to be classified as “will repay” or “will default”), this is called a *classification* problem.

In supervised learning problems, for each observation in the data set, we have a vector of attributes and an associated output or label. For example, in the case of valuing a house, the data would consist of the various characteristics of houses (lot size, square feet of living space, etc.) and their selling prices (the labels, or targets). In the case of loans, the data would consist of the different loan metrics (income of borrowers, credit scores, etc.) and labels indicating whether they defaulted or not. The algorithm learns from these “labeled” data with the aim of producing accurate predictions of the target value for new, unseen, and unlabeled instances.

There are many possible applications of supervised learning. The prediction of the value of a variable could be in a time-series context (e.g., forecasting gross national product or the value of the S&P 500 index next year) or making a cross-sectional prediction for a data point not in the sample (e.g., if my neighbors put their apartment up for sale, how much would it be worth?). In the financial realm, supervised ML found early applications in algorithmic trading and high-frequency

trade execution. A successful example of classification is in credit decisions where a lender classifies potential borrowers according to whether they are acceptable credit risks.

Semi-Supervised Learning

In semi-supervised learning the objective, as in supervised learning, is to make predictions. But only part of the available data is labeled (i.e., provides values for the variable that is to be predicted). The rest is used to determine patterns within the explanatory variables, or it is fitted with “pseudo-labels” determined by estimating a model on the labeled part of the sample.

Reinforcement Learning

Reinforcement learning is concerned with making a series of decisions to achieve a goal. The environment in which the learning is occurring may be static, i.e., it does not change as the learning process evolves. Or the environment may be dynamic, with changes occurring in it as the process evolves. In reinforcement learning, there are no explicit labels. Instead, feedback is provided in the form of a “reward” during the learning process, which encourages a desired behavior but without giving explicit instructions to the learner. In simple words, reinforcement learning uses a trial-and-error approach where the desired behavior is rewarded. The technique is advantageous when decisions need to be made repeatedly so that the algorithm can learn based on the rewards or sanctions received in previous rounds. Unlike both

unsupervised and supervised learning, the “output” from reinforcement learning applications is a recommended action given the circumstances rather than a prediction, classification, or cluster.

Reinforcement learning has found plenty of applications in risk management and portfolio optimization. For example, it is used to determine the optimal way to buy or sell a large block of shares, to determine how a portfolio should be managed, and to hedge derivatives portfolios.

Parametric vs. Non-Parametric Methods

Machine learning algorithms can be also divided between *parametric* and *non-parametric* approaches. Parametric methods require the modeler to make an assumption about the functional form of the relationship between the features and labels (i.e., the *map* that associates the features to the output). This map can be a linear function, as in the case of the linear regression model, or a non-linear, highly complex one. The parameters that describe this map are then estimated (or learned) using the available data. In contrast, non-parametric methods do not make any explicit assumption about the functional form of the map or relationship between the features and the output. Parametric methods carry the risk that the chosen functional form is wrong, and therefore the resulting model will not fit the data well. On the contrary, non-parametric methods are very flexible and capture very complex data patterns; however, they require many observations to obtain an accurate estimation of the map.

1.3 Exploratory Data Analysis

Exploratory data analysis (EDA) is an integral part of any ML project. EDA is the process of collecting, cleaning, visualizing, and analyzing data. It is an essential step prior to building a machine-learning model. Collecting data is the necessary first step in building any ML model. The data may be collected from single or multiple sources. It may be available in different formats, which will require conversion to a more usable format. The data is then cleaned to correct any errors, removing duplicate entries and removing or filling in missing observations. The data is then visualized and analyzed to understand any relationships contained in it. Histograms, statistical summaries, correlation matrices, scatter plots, and quantile vs. quantile (Q-Q) plots are some of the tools employed at this stage. Visualization and basic statistical analysis will be useful in selecting the features for building a robust ML model.

1.3.1 Data Collection and Preparation

Data represents the fundamental raw material of every ML application. Data analysts often spend up to 80% of their time cleaning the data, and good data cleaning can

make all the difference between successful and unsuccessful ML projects. Analysis can only be as good as the data it is based on, a concept often summarized by the acronym GIGO, which stands for *garbage in, garbage out*. Before venturing into a discussion of data preparation and pre-processing tasks, it is useful to draw some terminological distinctions.

Structured, Semi-Structured, and Unstructured Data

Raw data can be structured, semi-structured, or unstructured. Structured data are organized in rows and columns where, typically, each column represents one attribute, and the rows contain different observations for the attributes.² Census data or the database containing the characteristics of the borrowers of a bank (e.g., age, gender, income, job title, education, amount of borrowed money, etc.) are examples of structured data. In contrast, unstructured data are not arranged according to a preset data model. Examples of unstructured data are sensor data, image data, web logs, network traffic, and texts. Unstructured data are inherently more difficult to analyze, as they have to be represented first in a format that is readable by a machine. Semi-structured data refers to data that is partly structured and partly unstructured. Luckily, most datasets in the realm of finance and risk management come in a structured form; however, alternative datasets such as geo-satellite images or textual data (see Chapters 9 and 10) are gaining momentum.

Numerical and Categorical Data

Structured datasets may contain two types of attributes. Attributes such as age and income are numerical and have a natural ordering such that, for instance, an income of USD 50,000 is larger than an income of USD 40,000. Such attributes are called *continuous*, *numeric*, or *quantitative*. Attributes like “marital status” (e.g., single, married, widowed, or divorced) are said to be *categorical* and can take discrete values. A special case of categorical data is binary data, which can take only one of two values. An example is a variable that indicates whether a rolling credit facility has been granted (“YES” or “NO”). Categorical data may or may not have a natural order. For example, there is no inherent ordering between the categories “male” and “female,” but there is typically an ordering related to income categories, credit ratings, and the like.

Longitudinal and Cross-Sectional Data

Another useful distinction is between *longitudinal data* and *cross-sectional data*. The characteristics of a pool of mortgages loans is an example of cross-sectional data, where each loan observation is independent of the other observations. On the other hand, longitudinal data are related to each other temporally, spatially, or through network relations. *Time-series* data are the most common example of such data in finance. The returns of the S&P 500 over the past 50 years are time-series data. In this case, the data are the result of a measurement over time and therefore the observations cannot be understood independently from one another. Also, in contrast to cross-sectional data, where the order usually does not matter, time-series data

have a chronological sense that must be considered when modeling and analyzing the data.

Textual and Other Data

Machine learning is used to analyze documents containing textual data. The natural language processing (NLP) techniques that are used to process textual data are presented in Chapter 9.

In addition to textual data, there is increased use of other data such as audio, video, photographs, maps, and drawings in ML projects.

The classification of different data types is shown in [Table 1.2](#), and [Box 1.1](#) presents different scales of data measurement. The data types are not mutually exclusive because the primary data types are contained in the other data types listed in the table.

Table 1.2 Classification of Different Data Types

| Classification Basis | Data Type | Sub Type | Example |
|----------------------|----------------|-----------------|---|
| Primary data Types | Numerical data | Continuous data | Income, age, temperature |
| | | Discrete data | Number of daily emails received Number of pages in books |

| Classification Basis | Data Type | Sub Type | Example |
|-----------------------------|---|--|--|
| | Categorical data | Nominal data: Data with no natural ordering implied. | Marital status, gender |
| | | Ordinal data: Data with implied natural ordering | Educational level, credit ratings, Likert scale responses (strongly agree, agree, neutral, disagree, strongly disagree) |
| Organization of data | Structured: Data that can be organized in rows and columns | | Census data, mortgage loan origination, and performance data |
| | Unstructured: Unorganized data. Most alternative data belong to this category | | Textual data: Prospectus, books, news releases, transcripts of interviews and earnings calls, etc. Audiovisual: Podcast recordings, voice messages, webinars recordings, maps, photographs, paintings |
| | Semi-structured: Data has some structure, but it is not a fixed format. | | Email, CSV files, HTML files etc. |

| Classification Basis | Data Type | Sub Type | Example |
|---------------------------------------|--|-----------------|--|
| Longitudinal vs. Cross-sectional data | Longitudinal data: Contains data spanning several time periods, or connected spatially or through network relationships | | Stock prices and other financial data series, Economic data series |
| | Cross-sectional data: Data is for a single point in time or time period. It provides a snapshot of a specific moment or time period. | | 2024 annual income of different individuals Product sales in May 2024 for different companies |

Box 1.1: Scales of Data Measurement

Scales of measurement refer to the different levels of measurement. There are four different scales: Nominal, Ordinal, Interval, and Ratio.

- Nominal scales represent data without any inherent order. Examples of nominal scales are eye color (black, blue, brown, etc.) and blood type (A, B, O, etc.).
- Ordinal scales represent categories with a meaningful order but not having clear intervals between values. An example of an ordinal scale is customer satisfaction ratings (dissatisfied, neutral, satisfied).
- Interval data has a meaningful order with a consistent interval between values. But it does not have a true zero-point, implying that a value of zero

does not represent absence of the quantity being measured. Examples of interval scales are temperature measurements (20°C , -15°C), calendar dates, longitude, and latitude.

- Ratio data implies a natural ordering with a clear interval between values and it has a true zero point. Examples of ratio scales are height in centimeters and distance traveled. In this scale, a height of 0 cm represents the absence of height.

² In machine learning, attributes represent different characteristics of the data being modeled. On the other hand, features are attributes themselves, or quantities derived from attributes.

1.3.2 Data Cleaning

Data cleaning is important because of the errors potentially associated with the collection process. The main reasons for data cleaning are:

- **Inconsistent recording.** For data to be read correctly, it is important that all data are recorded in the same way.
- **Unwanted observations.** Observations not relevant to the task at hand should be removed.
- **Duplicate observations.** These should be removed to avoid biases.
- **Outliers.** Outliers are observations on a feature that are significantly different from the remaining data (e.g., they are several standard deviations from the mean), such that suspicion arises that they were generated by a different

underlying process. Outliers should be checked carefully, as they can have a big effect on results. They could also be due to substitution of values for missing data. For instance, a daily stock return of 2500% is very likely to be a mistake. However, sometimes outliers contain useful information. For instance, fraud-detection systems are generally based on the identification of unusual patterns in the data. Therefore, there is no one-size-fits-all treatment for outliers, and what to do depends on the specific application. Not all predictive models are sensitive to outliers. For instance, tree-based classification models, or support vector machines for classification (see Chapter 4) are deemed to be robust to the presence of outliers. If the model is sensitive to outliers, data scaling can often minimize the problem. Outlier detection is discussed in Chapter 3.

- **Missing data.** This is the most common problem encountered during the data-preparation stage. Missing data can be structurally missing or just unavailable. For instance, the education of a mutual fund manager will be missing if the fund is managed by a team. This is an example of structurally missing information. When values are missing, it is crucial to understand why they are missing and if the pattern of missing data is associated with the outcome. For instance, information about credit card use can only be collected for bank clients that have a credit card. However, if the intent is to predict whether a loan will become delinquent, the absence, or *missingness*, of credit card data is associated with the outcome if only more solvent clients apply for a credit card. This is called *informative missingness* and can induce

significant bias in the model. If missingness is not informative, the removal of a small number of observations with missing data from a large sample is not a problem. Otherwise, one approach is to replace missing observations on a feature with the mean or median of the observations on the feature that are not missing. This technique is called *imputation*. Alternatively, the missing observations can be estimated in some way from observations on other features. For instance, a few predictive models, such as tree-based techniques (see Chapter 4) can account for missing data.

1.3.3 Data Visualization

The next step in exploratory data analysis is data visualization. It is the practice of translating information into a visual context, such as a graph. Data visualization can be of great help during the data preparation phase to grasp patterns and identify potential problems, such as outliers.

During the exploratory phase, data visualization techniques can be used to achieve a reasonable understanding of the shape of the distribution of one or more variables. This enables the analyst to detect, for instance, if the data are highly skewed and need to be transformed, something that discussed in greater detail below. An obvious tool to visualize the shape of a variable's distribution is a *histogram*. A histogram is a count of how many observations fall within specified divisions (*bins*) of the x -axis.

Figure 1.1 shows a histogram of 521 weekly observations of US market returns collected between April 2010 and March 2020.³ To realize the plot, the data have been split into 100 bins. On the y -axis, we read the density, that is, the proportion of data per bin. Hence, the total area of the histogram is equal to 1. Visually, the distribution is not symmetric but has a longer left “tail.” It can be seen clearly by comparing with the normal distribution, which is plotted as a line on the same plot. When the distribution of a variable is not symmetric, that is, the probability that a data point is at the left of the mean is not equal to the probability that it is at its right, we say that the variable is *skewed*. In the example below, returns are evidently negatively skewed (i.e., the left tail is longer), a characteristic that should be unsurprising to anyone with some experience in the stock market: On the other hand, a distribution is said to be positively skewed when the right tail is longer. Large negative returns are rare but still more common than equally large positive returns in the stock market.

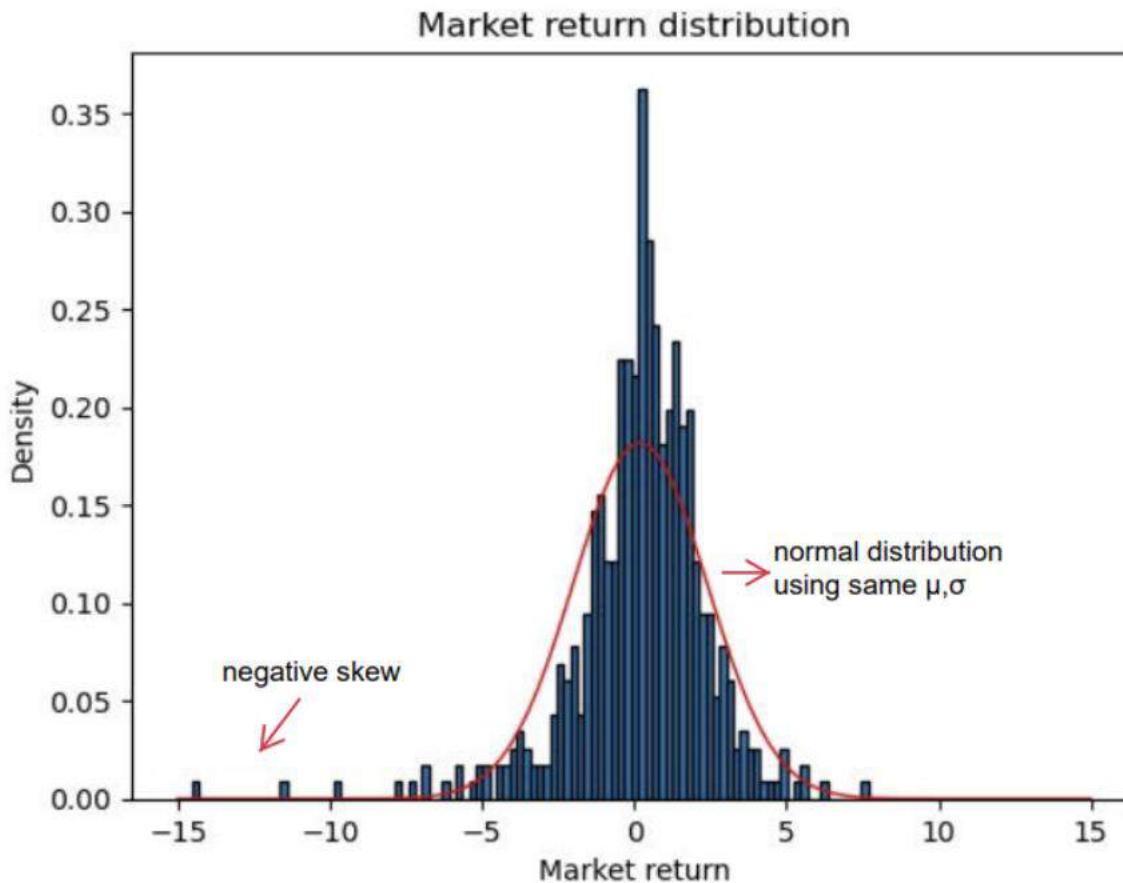


Figure 1.1 The distribution of US weekly market returns for a sample spanning 2010 to 2020.

A statistical summary is a useful tool to capture quickly the primary statistics of the data at hand. A summary of the returns data is presented in [Table 1.2](#).

Table 1.2 Statistical Summary of Weekly Market Returns (%) from 2010 to 2020

| Item | Value |
|---------------|--------|
| Mean | 0.19 |
| Median | 0.33 |
| St. Deviation | 2.19 |
| Skewness | -1.35 |
| Minimum | -14.54 |
| Maximum | 7.68 |

Some applications may require an assumption about the distribution of one or more variables. Quantile-versus-quantile (Q-Q) plots are generally employed to verify whether the empirical distribution matches the distribution that is assumed.⁴ A point on the Q-Q plot corresponds to one of the quantiles of the empirical distribution (y -coordinate) plotted against the same quantile of the theoretical distribution (x -coordinate). If the empirical and theoretical distributions are similar, the points should lie on the identity line ($y=x$) on the cartesian plane. Of particular interest is the normal Q-Q plot, which is used to assess whether the data are normally distributed. [Figure 1.2](#) shows a normal Q-Q plot for the weekly returns discussed above. The red line is the identity line ($y = x$). If returns were normally distributed, all the data points should lie approximately on that line. It is easy to see that the tails of the empirical distribution depart from the normal distribution. This is unsurprising as there is plenty of empirical evidence that returns display “fat-tailed” distributions.

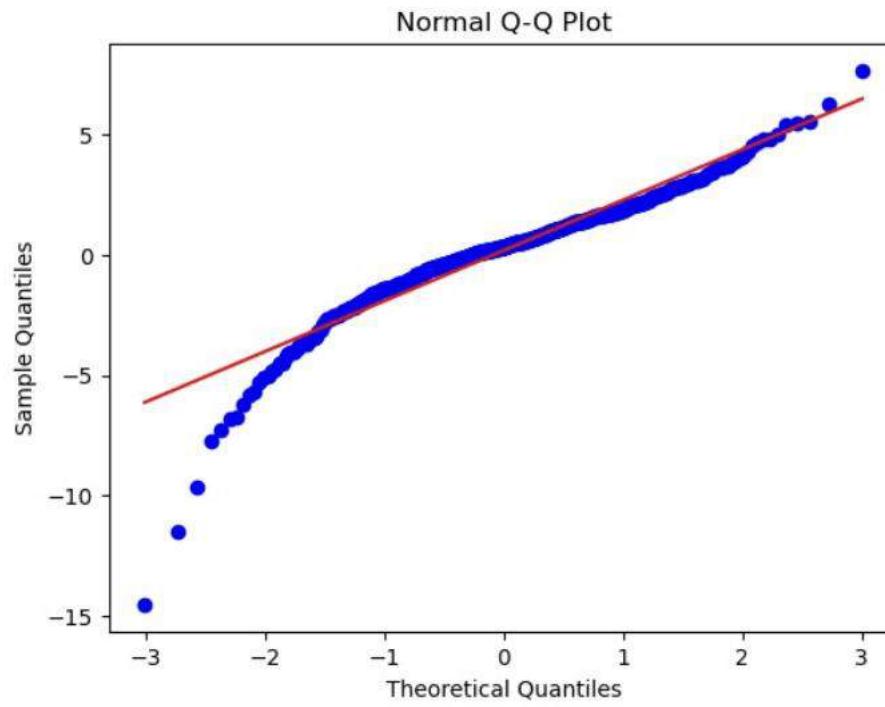


Figure 1.2 A normal Q-Q plot of US weekly market returns for a sample spanning from 2010 to 2020

Finally, a *box-and-whiskers plot*, sometimes simply called *boxplot*, can be employed to detect outliers. This is a graphical summary of a distribution. [Figure 1.3](#) is a boxplot of the weekly market returns. The box indicates the first and third quartiles (the “hinges” of the box) and the median (the bold line in the middle). The distance between the two quartiles is also known as the interquartile range. The lines (or whiskers) outside the box denote the smallest or largest observation that falls within 1.5 times the box size⁵ from the nearest hinge. All the observations that lie farther away are considered

outliers and are depicted separately. It clearly appears from the plot that the data contain a few extreme returns, especially negative ones.

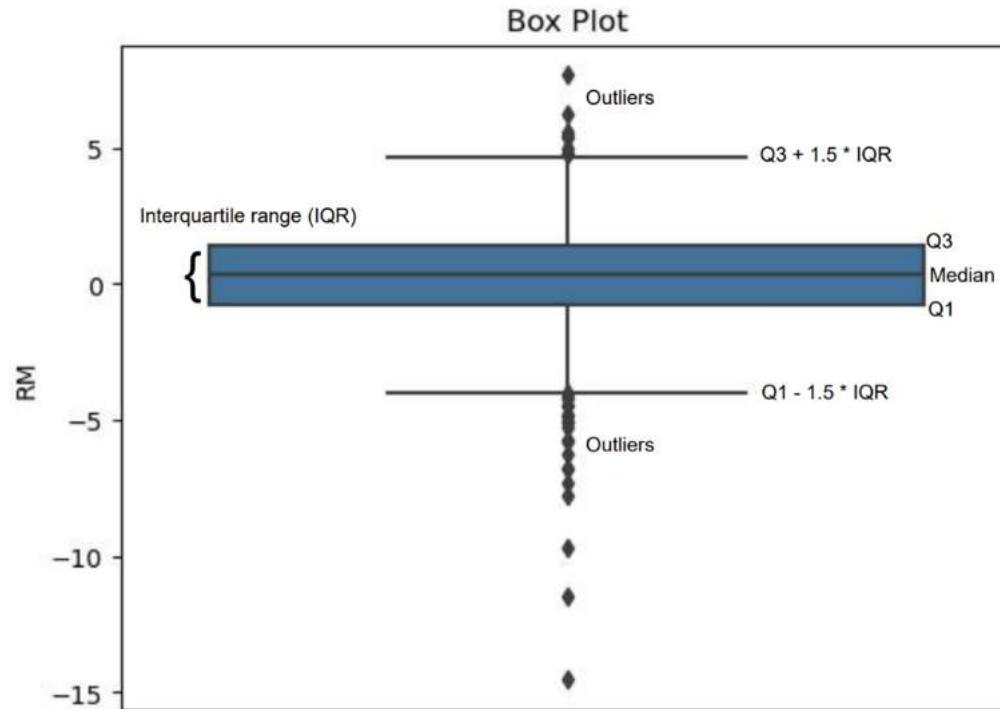


Figure 1.3 A box-and-whiskers plot of US weekly market returns for a sample spanning from 2010 to 2020.

In addition to individual attributes by themselves, the relationships among different attributes are also explored. Scatter plots showing the relationship among different attributes, and correlation matrices are typically used for this purpose. An example of a scatter plot is [**Figure 1.4**](#). It shows the relationships between the daily changes in different parts of the US Treasury yield curve. The corresponding correlation matrix is

in **Table 1.3**. It can be seen from the scatter plot and the correlation matrix that there are strong correlations present between the daily movements in the yield curve.

Table 1.3 Correlation Matrix Showing the Relationships Between Daily Changes in Different Parts of the Treasury Yield Curve⁶

| Treasury Maturity | 1Y | 2Y | 3Y | 5Y | 7Y | 10Y | 20Y | 30Y |
|-------------------|------|------|------|------|------|------|------|------|
| 1Y | 1.00 | 0.85 | 0.80 | 0.72 | 0.65 | 0.58 | 0.45 | 0.40 |
| 2Y | 0.85 | 1.00 | 0.96 | 0.90 | 0.83 | 0.76 | 0.61 | 0.54 |
| 3Y | 0.80 | 0.96 | 1.00 | 0.96 | 0.91 | 0.84 | 0.70 | 0.63 |
| 5Y | 0.72 | 0.90 | 0.96 | 1.00 | 0.97 | 0.93 | 0.81 | 0.74 |
| 7Y | 0.65 | 0.83 | 0.91 | 0.97 | 1.00 | 0.98 | 0.89 | 0.83 |
| 10Y | 0.58 | 0.76 | 0.84 | 0.93 | 0.98 | 1.00 | 0.95 | 0.90 |
| 20Y | 0.45 | 0.61 | 0.70 | 0.81 | 0.89 | 0.95 | 1.00 | 0.95 |
| 30Y | 0.40 | 0.54 | 0.63 | 0.74 | 0.83 | 0.90 | 0.95 | 1.00 |

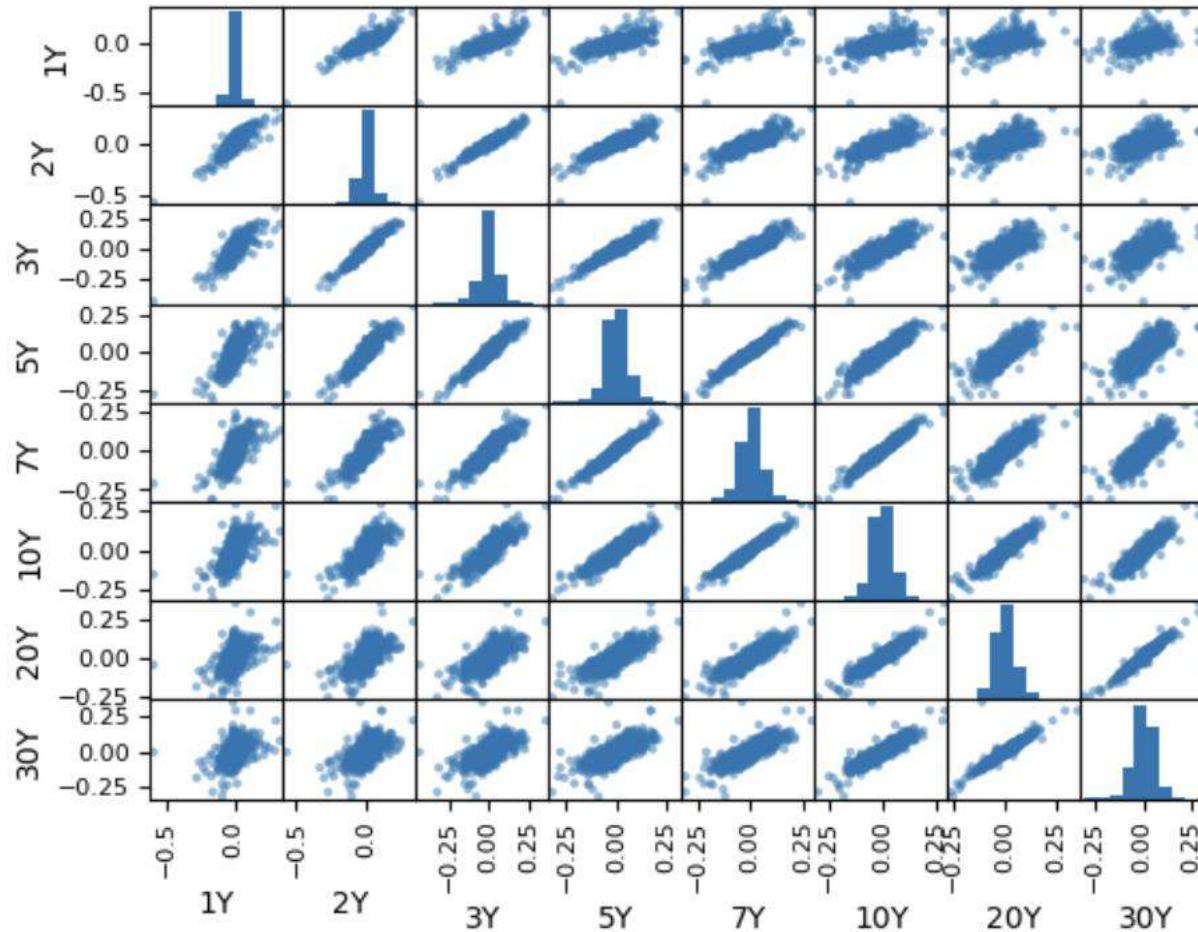


Figure 1.4 Scatter Plot Showing the Relationships Between Daily Changes in Different Parts of the Treasury Yield Curve⁷

³ The dataset is obtained from Prof. Ken French's website:

https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

⁴ Quantiles are points partitioning the observations in a data set into equal proportions.

⁵ The box size is the difference between 3rd and 1st quartiles.

⁶ The correlation matrix is for the daily changes in 1, 2, 3, 5, 7, 10, 20 and 30 year Treasury yields for the November 2018 to October 2023 period. Data obtained from FRED, Federal Reserve of St. Louis, Mo.

⁷ The scatter plot is for the daily changes in 1, 2, 3, 5, 7, 10, 20 and 30 year Treasury yields for the November 2018 to October 2023 period. Data obtained from FRED, Federal Reserve of St. Louis, Mo.

1.3.4 Feature Extraction

The next step is to go from the attributes collected in the datasets to features that can be used for analysis. This process is called *feature extraction* or *feature engineering*. Although quantitative data can be directly used as input to a model, qualitative information needs to be transformed in a way that is suitable for statistical analysis. The process of transforming non-numeric information into numbers is sometimes termed *encoding*. This procedure is discussed in detail in Chapter 9 in the context of textual data.

Categorical variables are often used to capture qualitative information so that it can be used in a model. Examples of categorical variables include marital status, geographic region, and the like. In all these cases, the categories do not initially have a numerical value associated with them, so we assign such values. It would be possible to set up a single variable taking different values for each of the categories (in the case of the marital status example, this could be 0 for “single,” 1 for “married,” 2 for “divorced,” 3 for “other categories,” etc.). This type of variable is known as a *dummy variable*. However, this representation would not work well because there is no natural ordering in the categories – this would be known as nominal data.

In addition, the model would assume that a move in the value of the variable from 0 (single) to 1 (married) would have the same impact on y as a move from 1 to 2, which would probably not be the case.

Instead, we represent categorical features by using separate binary dummy variables for each category. For instance, we would construct a variable called $SINGLE_i$, which would take the value 1 if person i were single and zero otherwise; a variable $MARRIED_i$, which would take the value 1 if person i were married; and so on. This use of 0-1 dummy variables is known as *one hot encoding* or binarization because each individual observation will only be in a single category, having a value 1 for that category and 0 for all other categories.

As a further example of a categorical variable, suppose we were developing a model to determine whether applications for credit cards should be accepted, and a piece of information we wish to include in the model relates to the applicant's region of residence in the US. Suppose further that we have five categories: Pacific, Rocky Mountain, Midwest, Northeast, and South. It might be tempting to think that we could set up a single variable taking values such as Pacific = 0; Rocky Mountain = 1; Midwest = 2; Northeast = 3; and so on. However, because the information has no natural ordering, it would be inappropriate to code it as if it did.

Again, the correct approach is to set up a separate 0-1 dummy variable for each category. Then, for each individual applicant, the dummy variables corresponding to the four categories that do not apply would take the value 0, whereas the one that applies would take the value 1.

However, including dummy variables for all categories in regression-based models can lead to the introduction of multicollinearity, a phenomenon known as the dummy variable trap. This situation arises when at least two dummy variables become perfectly correlated, resulting in inaccurate calculations of regression coefficients and standard errors.

To address this issue, it is necessary to impose additional constraints on the parameters of regression equations. Two commonly used constraints are (a) omitting one of the dummy variables from the equation, or (b) setting the constant term (bias) of the equation to zero.⁸

A slightly different situation is where there is a natural ordering for the categorical data (i.e., the variable is *ordinal*). Examples are company size and education level. For instance, a company's size could be specified as small, medium, or large. We could then use a variable that equals 0 for small firms, 1 for midsize firms, and 2 for large firms. Such a variable could be used as a feature but not as the output unless an estimation technique that can account for this was employed such as ordered logit (see Chapter 3).

On some occasions, we might be interested in converting numeric attributes into categorical ones. For instance, suppose that an analyst is given a dataset containing the regulatory capital of a pool of banks but only cares about whether the capital is above or below the regulatory threshold. The analyst can transform the continuous attribute into a categorical feature by creating a dummy variable that takes the value 1 if capital is above the threshold and 0 otherwise. Equivalently, if the analyst wanted

to split stocks into different categories based on market capitalization, the numeric attribute could be divided into several ranges, as described in the previous paragraph. This process is called *discretization*.⁹

Table 1.4. Examples of Different Data Conversions

| Data Type Converted | Example | Mapped to |
|-----------------------|-----------------------------------|---|
| Categorical - Nominal | Marital Status | Single (0/1), Married (0/1), Divorced (0/1) etc. |
| Categorical - Ordinal | Education Level Credit Ratings | No High School Diploma (0), High School Diploma (1) College (2), Post Graduate (3), Doctorate (4), Professional (5) AAA (6), AA(5), A(4), BBB (3), BB (2), B (1), Unrated (0) |
| Numerical | Firm size (market capitalization) | 0, Micro-Cap (<\$250MM) 1, Small (\$250MM -\$2B) 2, Medium (\$2B - \$10B) 3, Large (\$10B-\$200B) 4, Mega Cap (\$200B and above) |

⁸ Suits, Daniel B. "Use of dummy variables in regression equations." *Journal of the American Statistical Association* 52.280 (1957): 548-551.

⁹ Also referred to as binning.

1.3.5 Data Scaling

In practice, features are often measured on very different scales, and some can be several orders of magnitudes larger than the others. Many machine-learning approaches require all the variables to be measured on the same scale; otherwise, the technique will not be able to determine the parameters appropriately and the results will be dominated by the feature with the largest magnitude.

Standardization and Normalization

There are broadly two methods to achieve this rescaling, most commonly referred to as *standardization* and *normalization*. Standardization involves subtracting the sample mean of each variable from all observations on that variable and dividing by its standard deviation. Mathematically, the j^{th} observation on the i^{th} variable, x_{ij} , would be changed to:

$$, \quad (1.1)$$

where \bar{x}_i and s_i are the estimated mean and standard deviation, respectively, of the sample observations on variable i . This process creates a new variable that can take on any value but has a mean of zero and a variance of one.

Normalization (sometimes called the *min-max transformation*) takes a slightly different tack, creating a variable that is bounded between zero and one, but that will usually not have a mean of zero or unit variance:

,(1.2)

where $x_{i,min}$ and $x_{i,max}$ are the minimum and maximum of the observations on variable i , respectively.

Which method is preferable will depend on the characteristics of the data.

Standardization is preferred when the data contain outliers, because normalization would squash the data points into a tight range that is uncharacteristic of the original data.

There are primarily three reasons of performing standardization and normalization:

1. Numerical stability of the learning algorithm. The difference in the scale of the variables could cause overflow/underflow easily.
2. Ease of interpretation of the model parameter estimates. If the scales of the variables differ significantly, so do those of the parameter estimates and it would make evaluation of the importance of each estimate difficult.
3. Determining whether the out-of-sample prediction is within the range of the training data. If the out-of-sample data corresponds to a normalized value greater than 1 or smaller than 0, that means the prediction is an extrapolation that may correspond to a large prediction error.

Regardless of whether normalization or standardization of the variables in the dataset is undertaken, it is applied to all the input numerical data (not categorical). However, it is not necessary to rescale the variables that are being predicted.

1.3.6 Data Transformation

When the data have highly skewed features, it is a good practice to transform them. As a rule of thumb, if the ratio of the highest value to the lowest value is larger than 10 (in absolute value), we consider the variable to be highly skewed. A common practice with highly skewed data is to transform the data by using the natural log of the data, instead of directly using it. Although this usually does not produce a symmetric distribution, the data are better behaved. Alternative transformations include the square root or the inverse transformations, but they are less commonly used in financial applications.

Although scaling data does not change the shape of underlying distributions and correlations between different attributes or features of the data, transformation of data will result in changes to the shape of distributions and correlations.

[**Figure 1.5**](#) illustrates the distribution of 1,089 weekly observations of trading volume (expressed in millions) of the S&P500 collected between 1990 and 2010.¹⁰ On the left, the histogram shows the distribution of the volume, which displays a positive skew equal to 1.61. The maximum value is 9.33 and the minimum value is 0.09, yielding a

ratio of approximately 103 between the two. On the right, the histogram shows the distribution of the natural log of the volume. Although the distribution is not exactly symmetric, the values range approximately between -2 and 2 , with a maximum value of 2.23 and a minimum value of -2.44 . The skewness is now much closer to zero, exactly 0.05 . Hence, the log transformation has drastically reduced the skewness in the data.

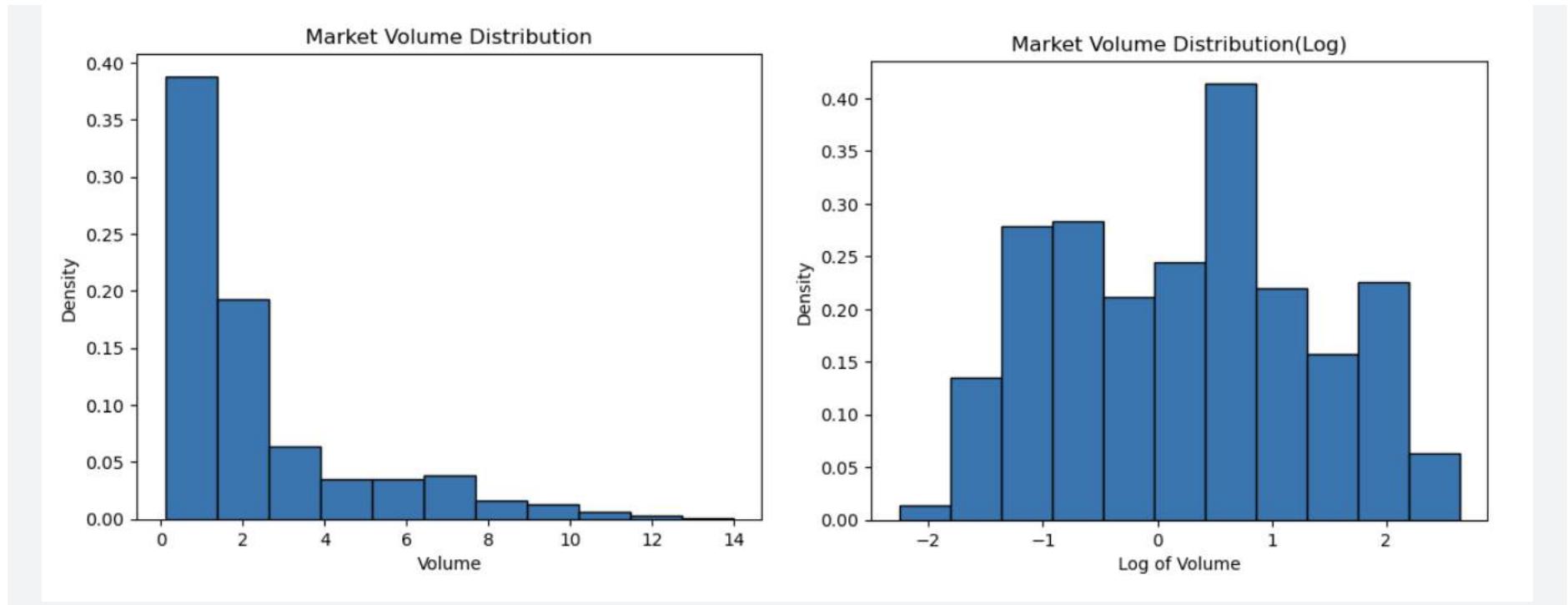


Figure 1.5 The distribution of the trading volume of the S&P500 from 1990 to 2010 (on the left) and of its log (on the right).

¹⁰ The dataset is obtained from Yahoo Finance's GSPC Weekly Historical Data, <https://finance.yahoo.com/quote/%5EGSPC/history>

1.4 Dimensionality Reduction Techniques

Large datasets can often be represented more compactly, which makes the application of sophisticated and computationally intensive algorithms easier. Techniques such as principal component analysis (PCA) use the correlations in the data to represent it in a smaller number of dimensions. These techniques, which also belong to the realm of unsupervised learning, are often used to obtain a smaller number of (uncorrelated) features from a larger number of correlated variables.

1.4.1 Principal Components Analysis

PCA is the most commonly used dimensionality reduction technique. The idea behind this method is to find linear combinations of the original predictors that summarize most of the variability in the data. These linear combinations are called principal components (PCs). The first PC is the linear combination of the original predictors that captures the most variability in the data among all possible linear combinations. The second PC is the linear combination of the predictors that captures the most variability that is *not already explained* by the first PC. This second PC is constructed to be *orthogonal* to the first PC, that is, the two PCs are uncorrelated. The process continues in the same way to find the third PC and so on, until all the variability in the data has been explained, which will provide the same number of PCs as features. The j^{th} principal component can be represented as follows:

(1.3)

where P_i denotes each of the original predictors (after standardization or transformation), m is the number of original predictors, and b_{ij} is the weight assigned to P_i in PC_j , which captures the importance of predictor i for the principal component j . The coefficients $b_{1j}, b_{2j}, \dots, b_{mj}$ are also called *component weights*, and they help us understand which features are the most important to explain the variability of the data. An example application of PCA to the Treasury yield curve is given in [Box 1.2](#).

Box 1.2: An application of PCA

A typical application of PCA is to reduce a set of daily changes in the US Treasury yield-curve to a small number of explanatory components. Suppose, for instance, that we have data on the daily changes in yields with one-year, two-years, three-years, five-years, seven-years, ten-years, twenty-years and thirty-years maturity for a sample period from November 2018 to October, 2023¹¹. Using PCA, an analyst can find a small number of uncorrelated variables that describe the Treasury yield curve.

Table 1.5 Principal component weights for 8 US Treasury yield series: one-year, two-year, three-year, five-year, seven-year, ten-year, twenty-year, and thirty-year

| | Principal Component | | | | | | | |
|-------------------|---------------------|-------|-------|-------|-------|-------|-------|-------|
| Treasury Maturity | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 0.29 | -0.54 | 0.73 | -0.29 | 0.06 | 0.02 | 0.02 | 0.00 |
| 2 | 0.35 | -0.40 | -0.15 | 0.60 | -0.35 | -0.46 | -0.03 | -0.03 |

| | Principal Component | | | | | | | |
|-------------------|---------------------|-------|-------|-------|-------|-------|-------|-------|
| Treasury Maturity | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 0.37 | -0.27 | -0.30 | 0.19 | 0.15 | 0.76 | -0.27 | -0.04 |
| 5 | 0.38 | -0.07 | -0.33 | -0.22 | 0.24 | -0.07 | 0.67 | 0.43 |
| 7 | 0.38 | 0.08 | -0.23 | -0.37 | 0.14 | -0.24 | -0.04 | -0.76 |
| 10 | 0.38 | 0.23 | -0.07 | -0.30 | -0.04 | -0.25 | -0.64 | 0.49 |
| 20 | 0.34 | 0.42 | 0.20 | -0.06 | -0.71 | 0.30 | 0.25 | -0.05 |
| 30 | 0.32 | 0.48 | 0.38 | 0.50 | 0.51 | -0.05 | 0.05 | -0.03 |

Table 1.5 shows the principal component weights obtained from daily changes in eight Treasury yields between November 2018 and October 2023 (1249 data points). The daily data was scaled before running the principal component analysis. To explain the movements fully, all eight components are necessary. However, when the actual movements are expressed as a linear combination of the components, the first component explains most of the variation (82%), and the first three components explain more than 98% of the variation. This is because there is a high degree of correlation between the yield movements, and the bulk of the information contained in them can be captured by a small number of PCs, which are then used as explanatory variables in subsequent regression models rather than the yields themselves.

Looking at the components' weights reported in the above table, we can see that the first PC loads positively and almost equally on all the interest rate variables. This would imply that all the eight yields tend to move in the same direction. Therefore, we can interpret the first PC as movements in the level of the yield curve. The second PC loads negatively on short-term yields (up to five-year) and positively on long-term yields, implying that as the shorter-term yields rise or fall, the longer-term yields tend to move in the opposite direction. We can interpret the second PC as movements in the slope of the yield curve. The third PC loads positively on one-year, negatively from two to ten-years and positively on long-term yields. We can interpret the third PC as a twist in the yield curve.

PCA is particularly useful when the original predictors are highly correlated with each other. In fact, besides reducing the dimensionality of the predictor set, PCA helps solve problems associated with multicollinearity (discussed in Chapter 3) and improves the numerical stability of models that require low correlation among the predictors. In fact, as explained above, PCs are built to be uncorrelated with each other and this appealing feature has contributed to their popularity.

Importantly, it must be understood that PCA seeks to find linear combinations to explain the variability in the predictors without any understanding of the predictors' measurement scale or their distribution (e.g., if they are skewed). For instance, if the predictors are on measurement scale that differs in orders of magnitude (think, for instance, of income and age among the demographic variables that explain the creditworthiness of borrowers), PCA will first summarize the predictors with more

variation, that is those with the largest magnitude. To avoid the problem that PCA summarizes distributional or scale differences in the data, the analyst ought to remove skewness and then center the data using the scaling methods discussed in Section 1.2.

PCA is often used in high dimensional problems to choose a lower number of predictors before applying another method (such a regression model or a more sophisticated technique) to predict the outcome. The choice of how many PCs to retain must be decided by the researcher. Unfortunately, there is no objective way to decide how many principal components are “enough,” or “too many.”

A heuristic method that is often used to support the decision is to construct a plot that contains the ordered component number on the x -axis and the amount of variability explained by each of them on the y -axis. The plot usually shows that the marginal amount of explained variability decreases as the number of components increases. The point at which the proportion of variance explained by each subsequent principal component drops off is often referred to as an elbow in the plot. The analyst then retains the components before this point.

Figure 1.6 shows the plot for the PCs obtained in Table 1.3. Visibly, the amount of variance explained drops after the first component, and an elbow is formed after the first two components. Overall, the first three components can explain virtually all the variability in the series, and therefore only these three need to be retained.

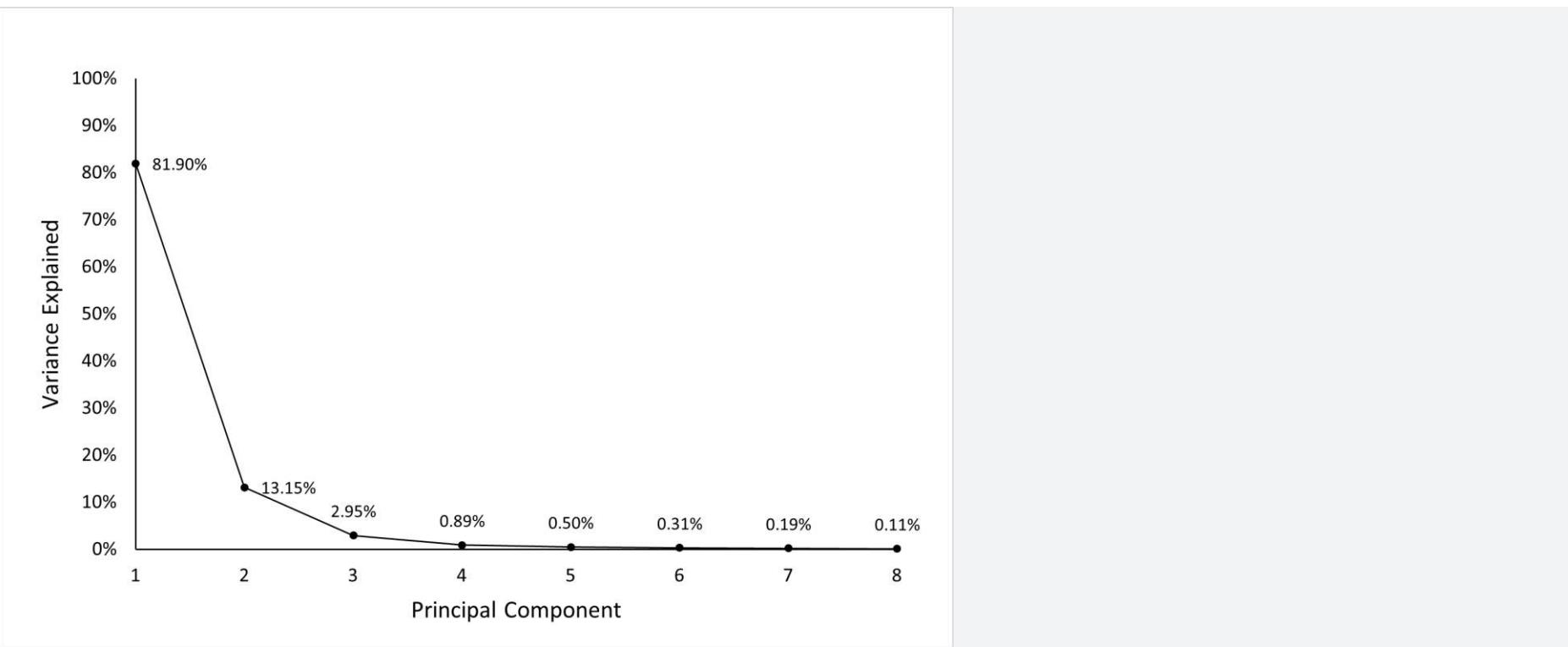


Figure 1.6 The plot of the variance explained by each principal component obtained from the daily movements of US Treasury yields.

¹¹ The Treasury data is obtained from Federal Reserve Economic Data | FRED | St. Louis Fed (stlouisfed.org)

1.5 Training, Validation, and Testing

After understanding and processing the data, the next step is to build and evaluate the model using it. As already stated, machine learning is often used with the goal of

making predictions or classifying observations.¹² When a dataset is used for prediction, the analyst wants the model to be able to generalize well to data that have not been used to estimate it. For this purpose, in conventional econometrics, it is common, although not universal, to retain part of a data sample for testing the fitted model and determining how well it can predict observations on the dependent variable that it has not seen. This leads to a distinction between in-sample (model estimation) and out-of-sample (sometimes known as a hold-out sample) parts of the data.

The use of a hold-out sample is even more crucial in machine learning, as there is typically little economic or financial intuition behind the modeling assumptions and the risk of choosing a complex model that accurately fits the dataset at hand but does not *generalize* well to unseen data is high. The estimation of a model that is too complex and captures the noise in the dataset at hand rather than the true nature of relationships between the features and the output(s) is generally known as *overfitting*. On the other hand, *underfitting* occurs when significant patterns in data are not captured by the model.

In ML, the analyst is generally not just interested in testing one model, but wants to try several models, choose between them, and to test the accuracy of the selected model. Therefore, the overall sample is usually split into three parts: training, validation, and test data sets. The validation set plays a crucial role in the choice of the model in machine learning.¹³

The *training set* is employed to estimate model parameters (e.g., the intercept or bias and weights or slopes in a regression)—this is the part of the data from which the computer learns how best to represent its characteristics.

The *validation set* is used to select between competing models. We are comparing alternative models to determine which one generalizes best to new data. Once this model selection has been undertaken, the validation set has already been “contaminated” and is no longer available for a genuinely independent test of the model’s performance.

Therefore, a third sample known as the *test set* is retained to determine the final chosen model’s effectiveness. A good model will be able to generalize, which means that it will fit almost as well to the test sample as to the training sample because the machine has learned the crucial elements of the relationship without fitting to unimportant aspects (the noise) that would not likely repeat in the test set.

¹² For simplicity, in what follows, we will use the term “prediction” generically for referring to both prediction and classification problems. We will draw a distinction between the two with reference to accuracy measurement in Chapter 8.

¹³ In contrast, in econometrics the data is divided in to in-sample (training) and out-of-sample (test) parts only since only one model is fitted.

1.5.1 Sample Splitting and Preparation

An obvious question is, how much of the overall data available should be used for each sub-sample? There is no definitive answer, and different researchers are liable

to make different choices. One rule of thumb is that roughly two-thirds of the sample is used for training, with the remaining third equally split for validation and testing. Naturally, if the overall number of data points is large, this separation will be less crucial. If the training sample is too small, this can introduce biases into the parameter estimation, whereas if the validation sample is too small, model evaluation can be inaccurate so that it is hard to identify the best specification.

If the output data in the sample have no natural ordering (i.e., they are cross-sectional), then the three samples should be drawn randomly from the total dataset. On the other hand, if the data are time-series, it is common for the training data to be the first part of the sample, then the validation data, with the test data being at the end. This sample split has the advantage of allowing the model to be tested on the most recent data.

Ideally, the overall dataset will be sufficient to allow for reasonably sized training, validation, and test samples. When this is not the case, cross-validation can be deployed to use the data more efficiently. Cross-validation involves combining the training and validation data into a single sample, with only the test data held back. Then the combined data are split into equally sized sub-samples, with the estimation being performed repeatedly and one of the sub-samples left out each time. The technique, known as k -fold cross-validation, splits the combined training and validation data available, n , into k samples, with the test data excluded from the combined sample. We return to cross-validation in Chapter 7, where we delve deeper into the details of model training and selection.

1.6 Software for Machine Learning

Programming languages are probably among the most important tools in the toolbox of a data analyst. Unfortunately, there is no such thing as a universally recognized “best language” or “best software package” for machine learning. Analysts often disagree on the choice, which remains in many cases a matter of personal preference. The best approach for someone who is interested in learning a new programming language is to select one that is flexible with a rich ecosystem of third-party open-source libraries¹⁴ and a vibrant community for continued software development and support when code does not work or fails to run as expected. Examples of such languages are R and Python, which are both open-source scripting languages that run on Windows, macOS, and Linux platforms and are commonly employed by data analysts for machine learning tasks. Both R and Python can perform virtually every data analysis task, have an easy-to-read syntax, and are relatively easy to learn. On the one hand, R tends to be preferred by statisticians as it is great for data visualization and has a rich environment of statistical packages. On the other hand, Python is a general-purpose language, and it is better at non-statistical tasks such as web scraping and textual analysis. Overall, they are both equally suitable for machine learning tasks and we will refer to both for our purposes.

The most important Python toolboxes for a data analyst are NumPy, SciPy, Pandas, Scikit-Learn, Tensorflow and Keras. NumPy is the building block for many other toolboxes in Python as it provides the framework to perform operations with

multidimensional arrays and to support fundamental linear algebra functions. SciPy is a collection of numerical algorithms that is used, for instance, to solve optimization problems, a crucial task in machine learning applications. Scikit-Learn is a machine learning library built on NumPy and SciPy. It offers tools to perform pre-processing tasks, model training, selection, and testing for several machine learning methods. Finally, Pandas is a toolbox for data manipulation providing high performance functions for merging, aggregating, and reshaping the data as well as ways to deal with missing values.

A popular package for building and evaluating machine learning models in R is *caret* (which is short for classification AND regression Training). It contains tools for data splitting, pre-processing, feature selection, model tuning and variable importance estimation. A list of other R libraries for machine learning tasks can be found at <https://cran.r-project.org/web/views/MachineLearning.html>.

¹⁴ Libraries (also known as packages or toolboxes) are pre-programmed sets of routines for accomplishing particular tasks. Libraries are usually “called” when required rather than being automatically available for use, which would unnecessarily take up a lot of memory.

Introduction to Tools & Techniques: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

1.1 For each of the following terms used in classical statistics, provide the equivalent term in machine learning parlance:

- Intercept

- Slope
- Explanatory variable
- Dependent variable
- In-sample period
- Out-of-sample period
- *Intercept – bias*
- *Slope – weight*
- *Explanatory variable – feature*
- *Dependent variable – output or label*
- *In-sample period – training data*
- *Out-of-sample period – test data*

1.2

A. What are the main differences between machine learning and more conventional econometric techniques?

Under conventional econometric approaches, the researcher selects a particular model or hypothesis and tests whether it is consistent with the available data. The emphasis is on inference and the main tools are t-statistics, p-values, and R-squares. Under machine-learning approaches, the emphasis is on letting the data decide the features to include in the model, with very few assumptions or theory. Inference is less important while the focus is on the model's prediction or classification accuracy out-of-sample.

B. For what kinds of problems would machine learning likely be more suitable than conventional econometric modeling?

Machine-learning techniques have advantages when applied to problems where there is little theory regarding the nature of a relationship or which features are relevant. It is used when the number of data points and the number of features are large (big data or wide data, as opposed to tall data where the number of predictors is strictly smaller than the number of observations). Machine learning might also be preferable when the relationships between features (and targets) are nonlinear.

1.3 What are the main differences between supervised and unsupervised machine learning methods?

In unsupervised learning problems, for each observation we have a vector of features, but no corresponding output value to predict. Therefore, the focus is on identifying patterns in the data. On the contrary, in supervised learning problems, a set of labeled examples is provided. In other words, there are instances for which values of the predictors and the outcome variables are available. The goal is to predict the outcome for new, unseen, unlabeled instances.

1.4 Give an example of:

A. a classification problem.

In a classification problem, the label is of categorical nature. An example is to discriminate among credit applicants who will default and those who will not. The predictors could be the characteristics of the borrowers (e.g., age, income, education, gender, etc.) while the output is a label indicating whether they have defaulted.

B. a prediction problem.

A prediction problem concerns the prediction of a numeric value. An example is the prediction of the sale price of a house. The predictors could be characteristics of the house (size, location, age of construction, proximity to schools, etc.) while the target variable is the sale price.

1.5 What is an outlier and how should it be treated?

Outliers are observations on a feature that are significantly different from the remaining data, e.g., they are several standard deviations from the mean, such that suspicion arises that they were generated by a different mechanism. Their treatment depends on the problem at hand. Sometimes they are the fruit of errors in the collection or transcription of

the data; other times (e.g., fraud detection) they convey useful information about the data. Remedies include using algorithms that are robust to the presence of outliers or data transformation.

1.6 What are the benefits of using principal components analysis (PCA)?

Principal components analysis involves projecting a feature dataset onto a smaller number of components. For instance, if the dataset involves ten features, the first five components might be used, which would then reduce the number of input variables by half. This is particularly useful in situations where the features are highly correlated or very numerous and estimating a model containing them could be challenging; by construction, the principal components are uncorrelated. The technique is straightforward to implement, no matter how many features or data points there are, because the components are simply linear combinations of the features.

1.7 Consider the dataset below, containing the income for a few credit applicants:

| Applicant ID | Income (\$) |
|--------------|-------------|
| 1 | 50,000 |
| 2 | 30,000 |
| 3 | 27,000 |
| 4 | 88,000 |
| 5 | 36,000 |
| 6 | 47,000 |
| 7 | 18,000 |

- A. Standardize each data point.

The first step in standardization is to compute the sample mean and standard deviation of the variable. In this case, the sample mean is \$42,286, and the sample standard deviation is \$23,041. Then each data point is standardized by subtracting the sample mean and dividing by the sample standard deviation, that is:

B. Normalize each data point.

The first step in normalization is to identify the minimum and maximum values among the sample data, which are 18,000 and 88,000 respectively. Then each data point is normalized by subtracting the minimum and dividing by the difference between the maximum and the minimum:

C. Comment on the values you obtain in parts a and b of the question.

The results are reported below:

| Applicant ID | Income (\$) | Standardized Income | Normalized Income |
|---------------------|--------------------|----------------------------|--------------------------|
| 1 | 50,000 | 0.33 | 0.46 |
| 2 | 30,000 | -0.53 | 0.17 |
| 3 | 27,000 | -0.66 | 0.13 |
| 4 | 88,000 | 1.98 | 1.00 |
| 5 | 36,000 | -0.27 | 0.26 |
| 6 | 47,000 | 0.20 | 0.41 |
| 7 | 18,000 | -1.05 | 0.00 |

As expected given the calculation methods, the some of the standardized figures are positive (those whose raw values were above the mean), while some are negative. If we calculated the mean and variance of the standardized incomes, they would be 0 and 1 respectively. On the other hand, the mean and standard deviation of the normalized incomes are not 0 and 1 but all the figures lie between 0 and 1. This occurs by construction and will always be the case whatever the original data.

Chapter 2: Unsupervised Learning

Learning Objectives

Unsupervised learning is associated with a model's use of unlabeled data to develop insights or pattern recognition with no specific guidance or rules. This chapter introduces clustering analysis, or segmentation, a common application of unsupervised learning that separates data points into groups based on the "closeness" of their features. Focusing on K-means clustering, a widely used approach, the chapter outlines how the method works, how to select the optimal number of clusters, how its performance can be evaluated, and the strengths and weaknesses in its application.

After completing this chapter, candidates should be able to:

- Differentiate between clustering techniques.

- Illustrate how the K-means algorithm separates data into clusters, and describe the advantages/disadvantages of K-means clustering.
- Describe performance measures such as Within the Cluster Sum of Squares (WCSS) and Between the Clusters Sum of Squares (BCSS).
- Apply different methods to determine the optimal number of clusters in unsupervised learning.
 - Describe the construction and uses of a dendrogram.

As outlined in the previous chapter, unsupervised learning involves analyzing a dataset with no labels or outputs, only a set of features. A common application of unsupervised learning is *clustering analysis*, also known as *segmentation*, which aims to separate data points into groups based on the “closeness” of their features. Clustering places points that are similar into the same group and points that are dissimilar into different groups. There are many applications of clustering analysis including:

- Organizing customer data into groups to determine the characteristics that separate their purchasing behaviors.
- Investigating whether subsets of bank accounts can be clustered into groups likely and unlikely to involve fraudulent transactions or money laundering, which is an example of anomaly detection.
- Creating clusters of documents or newswires with similar content.

In addition to the purposes illustrated in the examples above, clustering can also be helpful for identifying the structure of a set of features prior to conducting a classification or prediction task. In other words, even where we have labelled data, we might choose to deliberately ignore the labels initially to focus first on better understanding the characteristics of the features.

All clustering applications are based on measures of distance, as explained below, and consequently it is essential that the data are normalized or standardized before analysis. Otherwise, if one of the features has a larger scale than the others it will end up dominating the measures so that other features are rendered irrelevant.

There are several different families of clustering techniques. One set of techniques is hierarchical clustering, where we begin with either one cluster containing all points and successively separate them into sub-clusters, or alternatively we begin with each data point in a separate cluster and successively combine them together. The former approach is known as divisive clustering, while the later approach is called agglomerative clustering. A dendrogram is an example of a hierarchical clustering technique. A second set of clustering techniques is partitional clustering which separates the dataset by locating observations based on their centroids, which are the center-points of the clusters. The K-means clustering algorithm is an example of such techniques. A final set of clustering techniques is based on the density of points in feature space, of which DBSCAN and SNN are examples.

We begin the remainder of this chapter by a discussion of the most popular approach within the second category, which is known as K-means clustering – outlining how the

method works, how to choose the optimal number of clusters, and how its performance can be evaluated. It is followed by a presentation on hierarchical clustering techniques and a brief discussion on density-based clustering techniques.¹

¹ Autoencoder is an unsupervised learning technique based on neural networks. It is discussed in Chapter 4.

2.1 The K-means Clustering Algorithm

The K -means algorithm is a straightforward, unsupervised algorithm to separate N observations into clusters. The number of required clusters, K , is determined at the outset by the analyst. Often, analysts try several different values of K and then aim to choose the most appropriate from among them as described below.

The algorithm, sometimes also known as Lloyd's algorithm, proceeds as follows:

1. Randomly choose initial values for the *centroids*, $\mu_j, j = 1, \dots, K$, which are the centers of the clusters.
2. Allocate each data point, $X_i, i = 1, \dots, N$, to its nearest centroid.
3. Recalculate the centroids to be at the centers of all the data points assigned to them.
4. Repeat steps 2 and 3 until the centroids no longer change.

Instead of centroids, the clusters can be determined with reference to medoids, which are the most frequently occurring points, if the features are categorical rather than taking continuous values.

Steps 2 and 3 require a definition of the distance of each observation to the centroids. There are two commonly used measures. The first is the Euclidean ("as the crow flies") distance, and the second is the Manhattan distance measure.²

To provide a simple illustration, suppose that we have two features, x_1 , and x_2 , and two observations on each of them, represented by the points P and Q in [Figure 2.1](#), which have coordinates (x_{1P}, x_{2P}) , and (x_{1Q}, x_{2Q}) , respectively. The Euclidean distance, d_E , between the two points would be calculated as the square root of the sum of the squares of the distances in each dimension (the diagonal line drawn onto [Figure 2.1](#)), sometimes known as the L^2 -norm:

$$d_E = \sqrt{(x_{1Q} - x_{1P})^2 + (x_{2Q} - x_{2P})^2} \quad (2.1a)$$

The measurement would be constructed in the same fashion if there were more than two dimensions (i.e., when there are more than two features). If there were m features for two points P and Q , the distance would be the square root of the sum of the squares of the distances:

$$d_E = \sqrt{\sum_{j=1}^m (x_{jQ} - x_{jP})^2} \quad (2.1b)$$

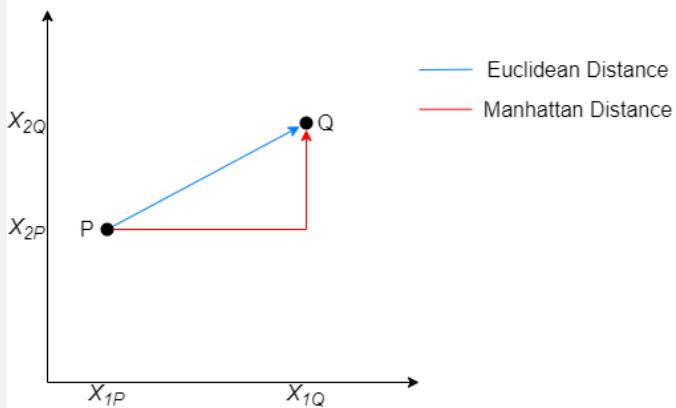


Figure 2.1 Two points, P and Q , with their coordinates for the calculation of the Euclidean distance.

Note that, for simplicity, the formulae above describe the distance between one point P and another point Q . However, the purpose of K -means is not to minimize the distance between points, but rather to minimize the distance between each point and its centroid.

To see how K -means works in practice, consider ten fictional data points reported in [Table 2.1](#). Assume that $K = 2$.

Table 2.1 Ten fictional data points with two features, x_1 and x_2 .

| i | x_1 | x_2 |
|-----|-------|-------|
| 1 | 6.8 | 12.6 |
| 2 | 3.3 | 11.6 |
| 3 | 3.8 | 9.6 |
| 4 | 4.4 | 7.8 |

| i | x_1 | x_2 |
|-----|-------|-------|
| 5 | 7.6 | 17.4 |
| 6 | 6.5 | 19.9 |
| 7 | 4.5 | 2.7 |
| 8 | 8.4 | 6.9 |
| 9 | 6.6 | 7.8 |
| 10 | 4.5 | 7.3 |

Although the k -means algorithm should be applied on data that underwent a rescaling of some form, for the sake of this example we skip that step, given that the two features are measured on a similar scale. To make the problem more concrete, we could imagine x_1 to be the outstanding balance on a credit card and x_2 the monthly income of the borrower, both measured in thousands of US dollars.

Let us arbitrarily select the points $[3.3, 11.6]$ and $[6.6, 7.8]$ as starting values for the centroids and use the Euclidean distance to assign the points to their closer centroid.

The Euclidean distance between P_1 and the first centroid is given by:

while the distance between P_1 and the second centroid is given by

Table 2.2 reports the Euclidean distance between each point and each of the two centroids (the fourth column reports the distance of a point from the centroid of the first cluster and the fifth column reports the distance of a point from the centroid of the second cluster). Each point is assigned to the closest centroid, with the assignment shown in the sixth column.

Table 2.2 Ten fictional data points with two features, x_1 and x_2 and their Euclidean distance from random centroids

| i | x_1 | x_2 | Distance to k_1 | Distance to k_2 | j |
|-----|-------|-------|-------------------|-------------------|-----|
| 1 | 6.8 | 12.6 | 3.6 | 4.8 | 1 |
| 2 | 3.3 | 11.6 | 0.0 | 5.0 | 1 |
| 3 | 3.8 | 9.6 | 2.1 | 3.3 | 1 |
| 4 | 4.4 | 7.8 | 4.0 | 2.2 | 2 |
| 5 | 7.6 | 17.4 | 7.2 | 9.7 | 1 |
| 6 | 6.5 | 19.9 | 8.9 | 12.1 | 1 |
| 7 | 4.5 | 2.7 | 9.0 | 5.5 | 2 |
| 8 | 8.4 | 6.9 | 6.9 | 2.0 | 2 |
| 9 | 6.6 | 7.8 | 5.0 | 0.0 | 2 |
| 10 | 4.5 | 7.3 | 4.5 | 2.2 | 2 |

Obviously, the points from where we started are no longer centroids of these two newly formed clusters. Therefore, we should now compute the new centroids based on our assignment in the first step. The centroids are computed as the averages of the features of the points allocated to that cluster. For the first cluster, the average of x_1 is:

and the average of x_2 is:

Therefore, the centroid of cluster one is now [5.6, 14.2]. In the same fashion, we can obtain that the centroid of cluster two is [5.7, 6.5]. We can now compute the distances of each data point to the new centroids and assign each observation to the cluster with the closest centroid. The algorithm stops when the clusters no longer change. In this case, in the following iteration the point [3.8, 9.6] moves from cluster one to cluster two. At the next iteration, the clusters do not change, and they appear as in [**Figure 2.2**](#) below. The diamonds belong to cluster two, and the circles to cluster one. The centroids of each cluster, which are not among the original data points, are depicted in red.

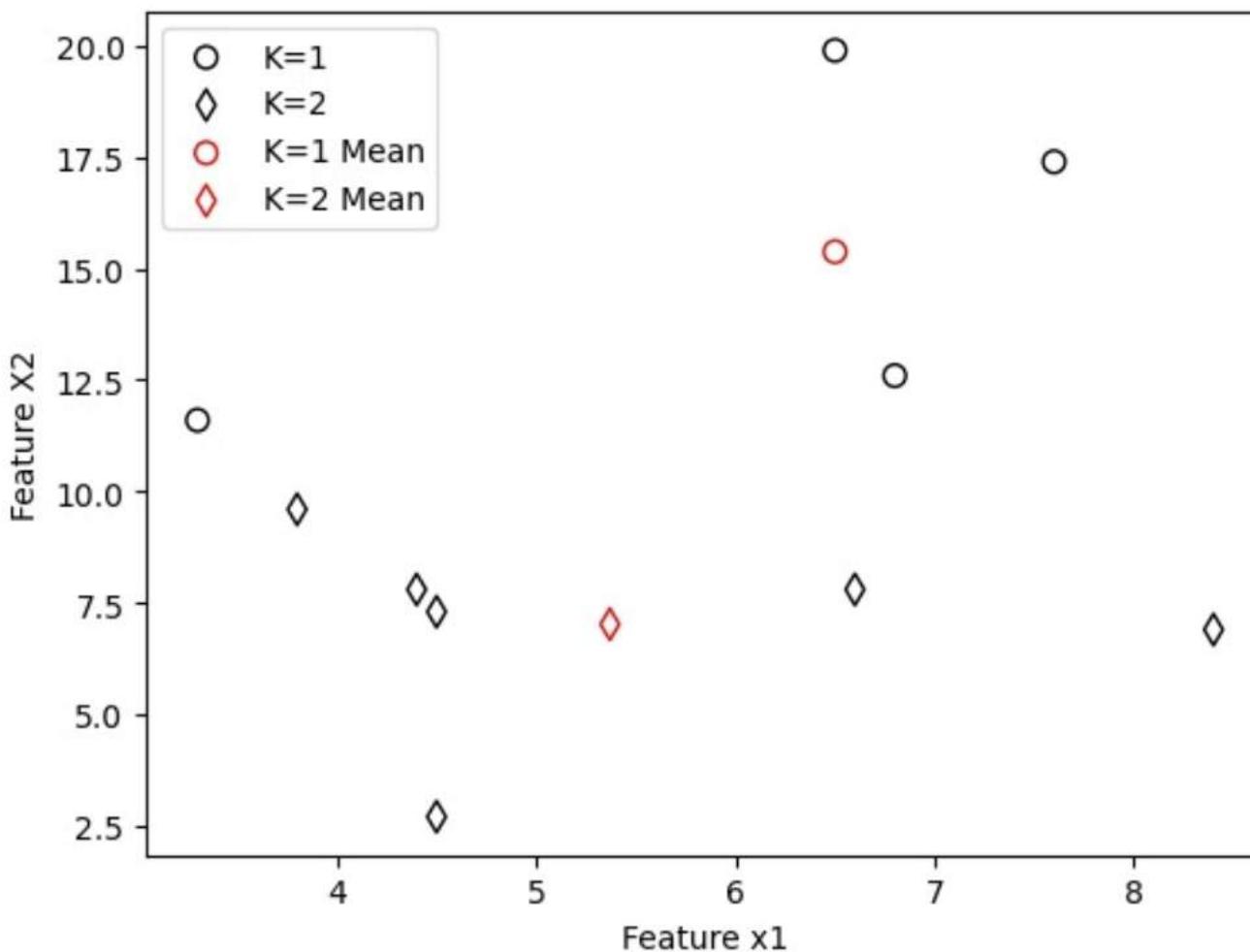


Figure 2.2 A plot of the 10 fictional points reported in [Table 2.2](#) on the Cartesian plane, where feature x_1 is on the horizontal axis and feature x_2 is on vertical axis.

² Manhattan distance and other distance measures are discussed in Appendix 2.A.

2.1.1 Performance Measurement for K-means

Evaluating the performance of a clustering application is more challenging than in the context of supervised learning due to the absence of a “right answer”. With supervised learning, the predictions for the training, validation and test sets can each be compared with the outputs, and a range of possible evaluation measures constructed, as described in Chapter 8. But even if we retained some observations in a test sample, this still would not help with evaluation. So, a different approach is required, and there are several possibilities.

The first option is to rely on a measure called *inertia*. The better the model’s fit, the closer the data points will be, collectively, to their respective centroids. We calculate the Euclidian distance measure, d_j between a data point j and the centroid to which it has been allocated. We can then define the inertia, or *within-cluster sum of squares* (*WCSS*) for that cluster as:

$$WCSS_k = \sum_{j=1}^{n_k} d_j^2 \quad (2.2a)$$

where n_k is the number of datapoints within the k^{th} cluster. The total WCSS for all the clusters is given by:

$$WCSS = \sum_{k=1}^K WCSS_k \quad (2.2b)$$

where K is the total number of clusters.

The lower the inertia, the better each cluster fits the data and the more separation there would be between the clusters, and so the within cluster sum of squares should be minimized. Another quantity called the *between cluster sum of squares (BCSS)*, which is the sum of squares of the distances of each centroid to the centroid of all data points weighted by the number of data points in each cluster. This measure, which increases as the number of clusters grows, gives another indication of how well the clusters are separated.

A slightly more sophisticated approach to performance measurement would be to use the Variance Ratio Criterion (VRC) which is also known as the Calinski-Harabasz index:

$$VRC = \frac{BCSS(N - K)}{WCSS(K - 1)} \quad (2.3)$$

Where N is the total number of data points. A higher VRC implies a model with better grouping of clusters.

2.1.2 Selection of the Starting Positions of the Centroids

The initial positions of the centroids are usually chosen randomly. If we choose a different set of initial locations, it's likely that we get a different set of solutions,

particularly for large values of K . To mitigate this dependency of the outcome on the initial random choices, it is common to run K -means several times with different random initializations, and then to select the outcome with the lowest inertia. A further alternative approach is to use “ K -means++”, which involves establishing the initial positions of the centroids far from each other in feature space, so that the position of only one of the centroids is chosen randomly. This can lead to better outcomes and faster convergence to the optimal solution. In fact, with randomly chosen initial centroid locations, it is possible that their initial positions are close together, making it hard to identify distinct clusters. This means that it can take many iterations to reach convergence when the numbers of features and data points are large.

2.1.3 Selection of K

In the same way that R^2 will never fall when more explanatory variables are added to a regression model, the inertia will never rise as the number of centroids increases. In the limit, as K reaches its maximum possible value of N , the total number of data points, each data point will have its own cluster and the inertia will fall to zero. Such a model with $K = N$ would clearly be of no value even though it would fit the data perfectly, and therefore, choosing K optimally is an important practical consideration. A straightforward rule of thumb is to set K equal to the integer closest to the square root of half the number of data points, $\sqrt{\frac{N}{2}}$. However, this is rather arbitrary and will

not vary depending on the characteristics of the dataset and how useful it would be to create an additional cluster. Two more sophisticated approaches based on inertia and silhouette scores are now presented.

Scree Plots

One approach is to calculate the value of the $wCSS$ for different values of K and plot the result; in other words, one can plot the within-cluster sum of squares against the number of clusters. This is sometimes known as a scree plot, which can also be used to determine the number of components to use in PCA as discussed in Chapter 1. We would then examine the figure to determine whether there is an obvious point at which the $wCSS$ starts to decline more slowly as K is further increased (a so-called “elbow”), which we would then choose as the optimal number of clusters. Because by construction the inertia will scale with the square of the features, we cannot argue that a particular value of it is “good” or “bad”; rather, we can only compare across models.

Silhouette Scores

An alternative way to choose K is to compute what is termed the *silhouette coefficient* or *silhouette score*. This compares the average distance of each observation from other points in its own cluster (known as *cluster cohesion*, denoted a_i) with its average distance from points in the closest other cluster (known as *cluster separation*, denoted b_i). The silhouette score for a particular data point i would be:

$$s_i = \frac{b_i - a_i}{\max[b_i, a_i]} \quad (2.4)$$

Given the form of the equation, s_i will lie within the range $[-1, 1]$. By averaging the silhouette scores for all the points within a cluster, we can calculate one silhouette score, U_j , for each of the K clusters. It is common to average these scores:

$$s = \frac{1}{K} \sum_{j=1}^K U_j \quad (2.5)$$

The quantity s measures the mean difference between the intra-cluster and inter-cluster distances, normalized by the maximum of the two. The best value of K is the one that gives the highest silhouette score s , which implies that the points within a cluster are closest together but furthest away from other clusters. The limit case $s = 1$ would imply that all the points allocated to each cluster were exactly on the centroid of their respective clusters whereas $s \approx 0$ implies that clusters are significantly overlapping; $s < 0$ is empirically unlikely but would correspond to points being mis-clustered. An example of the use of the silhouette method is given in Appendix 2.B.

2.1.4 K -means Selection of K Example

We now apply the K -means clustering algorithm to annual value-weighted stock index returns (all stocks on the NYSE, Amex, and NASDAQ) and Treasury bill yields³ from 1927 to 2021.

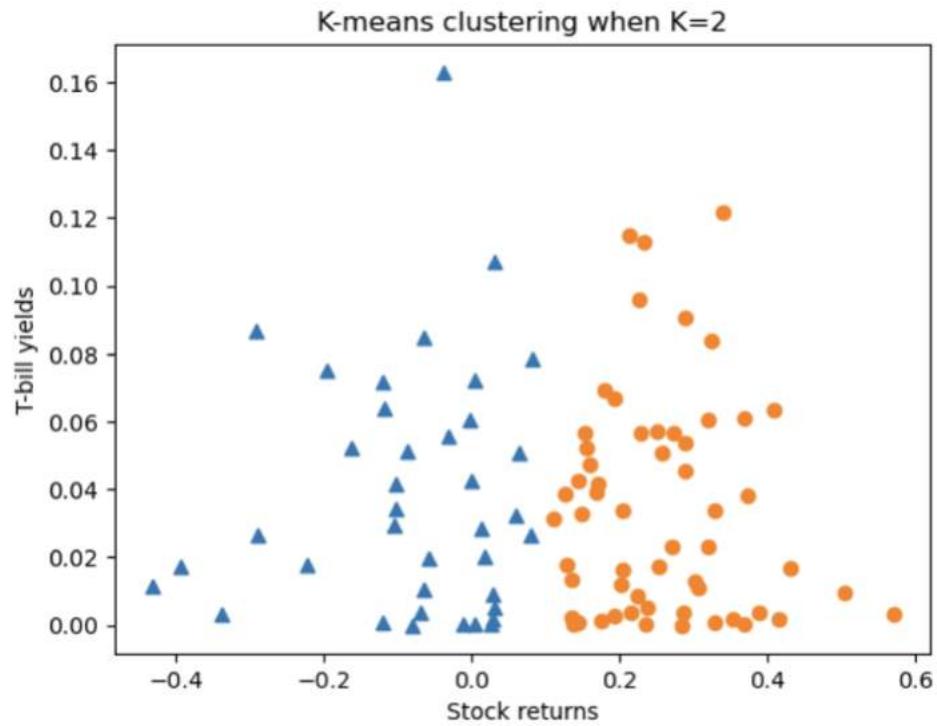


Figure 2.3 K-Means clustering for annual time-series of Treasury bill yields and stock returns, 1927–2021 – with two clusters (setting K = 2)

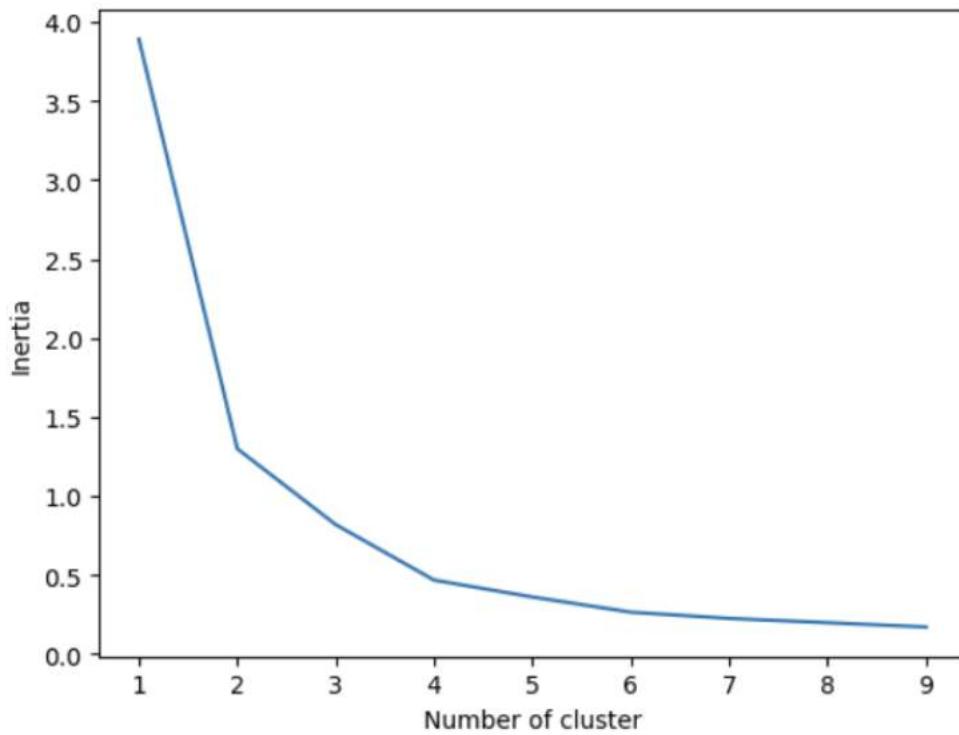


Figure 2.4 Plot showing the K -means fit versus number of clusters. This figure plots the number of clusters from 1 to 10 against the inertia for a K -means fit to annual time-series of Treasury bill yields and stock returns, 1927–2021.

It is apparent that the algorithm has identified two separate clusters based on separating the stock returns into “boom” and “bust” regimes. The optimal value of K is investigated in [Figure 2.4](#) for this data, which presents a scree plot of the number of clusters from 1 to 10 against the inertia (within-cluster sum of squares). An elbow is visible at $K = 2$, indicating that this value might be optimal.

³ The data are obtained from Ken French’s website: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html. Reprinted with permission of Ken French.

2.1.5 Advantages of K-means

k -means is widely applied in finance and many other areas. Compared with other clustering techniques, its advantages include that it:

- Makes intuitive sense and is easy to visualize if there are up to three features.
- Is quite fast and scales well to large numbers of data points.
- Will always converge to a solution (even though it might not always be the most appropriate one).

2.1.6 Problems With K-means

k -means clustering is straightforward to understand and implement, but has several disadvantages.

1. **Non-spherical clusters.** As well as the need to specify an *a priori* number of clusters, a further disadvantage of the technique is that because it is based on distances from a centroid, it tends to produce spherical clusters. Some of the clusters that arise in practice have non-spherical shapes, and this can cause problems.

Figure 2.5 shows two “pathological cases” where k -means will fail to work in determining the clusters. In both diagrams, there are two groups of points that are visibly separate from one another. But in the left-hand diagram, the two optimally positioned centroids are approximately in the same place in the middle of the inner circle with the data comprising two rings, one of which is located entirely inside the other. Therefore, the distances will be identical whichever of the two centroids each point is allocated to and the algorithm will be unable to form two distinct clusters.

In the right-hand diagram of **Figure 2.5**, there are again two visibly separate clusters but consider specifically the orange highlighted point on the right-hand side of the left-hand (short, wide) oval cluster. If the optimal centroids of the two clusters are roughly in the centers of the ovals, then the orange point is closer to the centroid of the other cluster than to its own cluster, which would lead it to be misallocated. In fact, k -means as described above might not lead to the specification of appropriate clusters in any application where the data space does not have an approximately spherical shape.

Possible solutions to these issues are either to apply a non-linear transformation of the input data, known as a *kernel function*, or to switch to a different technique that can cope better with a wider variety of data patterns.

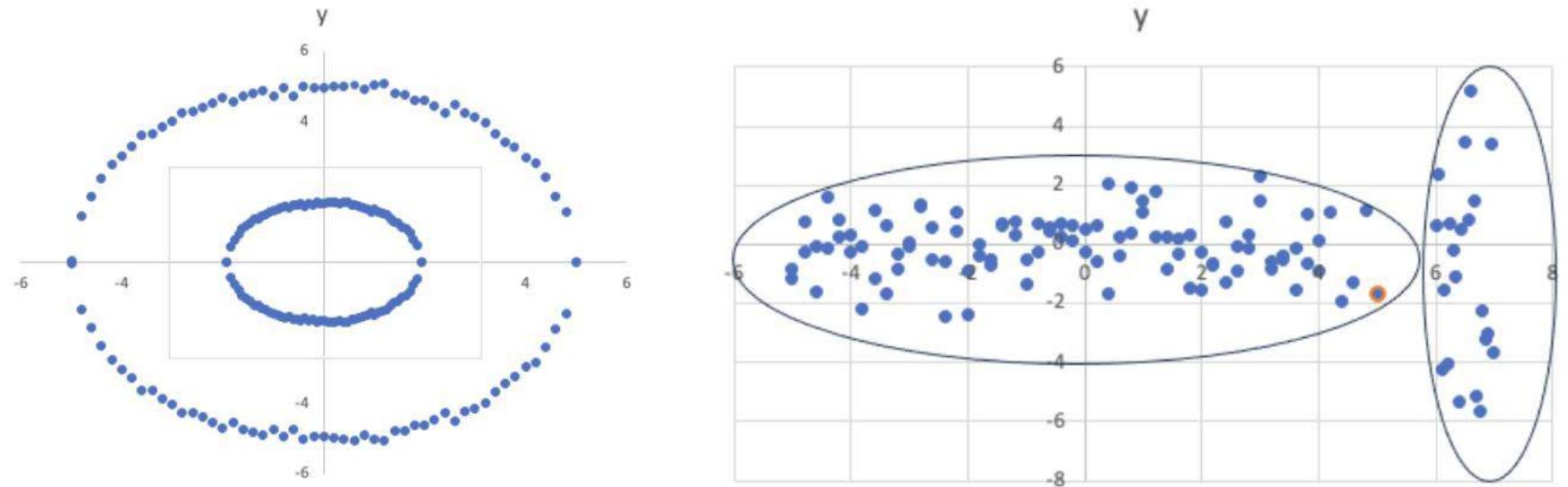


Figure 2.5 Some data patterns that cause K -means problems

2. **The presence of outliers.** K -means can be highly sensitive to outliers, which can either cause the centroids to become distorted or, in extreme cases, outliers might end up with their own cluster. This issue is particularly likely if the model is overfitted with too large a value of K . Standardizing the data using the min-max transformation will mitigate this issue to some extent, although from this perspective (and noting the disadvantages of doing so that will be outlined in Chapter 3) it may be preferable to remove outlying data points entirely from the sample prior to beginning the analysis.

3. The curse of dimensionality. Although k -means scales well as the number of data points increases, it tends to perform poorly when the number of features is large, converging very slowly to an optimal solution. The reason is that, as the number of features increases for a fixed sample size, the data points become increasingly sparse ("spread out") in feature space. Although the distances between each point and a centroid will increase as features are added to an equation such as (2.2), because it is a sum of squares, the distances between a given point and different centroids will become increasingly similar, which is known as *distance concentration*. This will make it hard for k -means to identify the clusters. Moreover, k -means struggles to construct the clusters if the features are highly correlated. To avoid this issue, there are several possibilities:

- Some features can be removed from the analysis.
- The features can be transformed to a smaller subset using principal components analysis or a similar technique.
- A distance measure that does not always increase with respect to the number of features can be employed instead of the Euclidean distance, such as the cosine distance.⁴

⁴ This measure is discussed in Appendix 2.A

2.1.7 Fuzzy K -means

The approach described so far in this chapter involves *hard clustering*, where data points are allocated uniquely to each centroid, meaning that a particular data point is either in cluster j or it is not. Each point is allocated to one and only one cluster, which is known as an exclusive or “winner takes all” clustering regime. An alternative framework is *soft clustering*, also known as fuzzy clustering or *fuzzy C-means*. Here, each data point can belong to more than one cluster, with a probability assigned to each cluster. For instance, a point i might have a 0.1 probability of being in cluster 1, a 0.6 probability of being in cluster 2, and a 0.3 probability of being in cluster 3. Implementing this requires a modification of the equation for inertia so that instead of calculating the direct summation of d_{ij}^2 , the equation incorporates a probability, raised to a power f , where f is a fuzziness coefficient. The larger the value of f , the greater the extent to which the clusters will overlap (i.e., they will be fuzzier).

2.2 Hierarchical Clustering

A limitation of k -means clustering is that it necessitates the number of clusters being specified *a priori*. Hierarchical clustering does not have this requirement, and instead it utilizes every possible value of K (from 1 to N) as an integral part of the process.

There are two types:

1. **Divisive:** This begins with just one cluster and then sequentially partitions the data by adding another cluster until every data point has its own cluster.

2. Agglomerative: This begins with each point having its own cluster, and then the closest two clusters in feature space are merged. The process ends when all points have been merged into a single cluster.

In either case, a definition of the proximity between sets of points is required. Here, we will need to capture the distance between one cluster and another cluster and compare that with the distance between one cluster and a single point not in that cluster. Instead of a distance measure, a *linkage criterion* is used, of which there are several variants. The most straightforward of these is a *single linkage*, which simply calculates the distance between clusters based on the distance between two data points from those clusters that are closest to one another. Another possibility is to use a *complete linkage*, which calculates the distance between the points in the clusters that are furthest apart from one another. These two linkage methods are illustrated in [**Figure 2.6**](#).

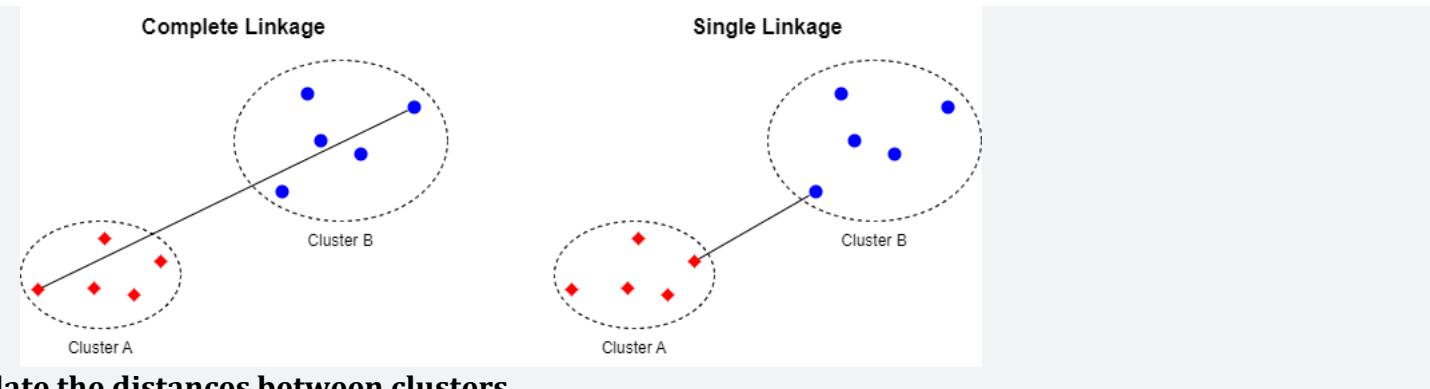


Figure 2.6 Two ways to calculate the distances between clusters

When the process of division or agglomeration is completed, the output is presented in a *dendrogram*, which has the data points on the *x*-axis and distances on the *y*-axis.

For illustrative purposes, consider that six “objects” (call them A, B, C, D, E, and F, and imagine that they are six stocks, and their features are the size and the book-to-market ratio) must be clustered. Let us adopt an agglomerative approach and start with each object being a cluster of its own. The symmetric matrix in [Table 2.3](#), also known as *distance matrix*, gives us the distances between the objects.

Table 2.3 The distance matrix for six fictional objects

| | A | B | C | D | E | F |
|---|----|----|----|----|----|----|
| A | 0 | 5 | 7 | 4 | 8 | 10 |
| B | 5 | 0 | 3 | 9 | 14 | 6 |
| C | 7 | 3 | 0 | 12 | 13 | 17 |
| D | 4 | 9 | 12 | 0 | 16 | 15 |
| E | 8 | 14 | 13 | 16 | 0 | 18 |
| F | 10 | 6 | 17 | 15 | 18 | 0 |

The shortest distance is between B and C, and therefore, in the first step, the algorithm creates a cluster BC. Now there are five clusters: A, BC, D, E, and F. To compute the distance between BC and the other clusters, we can adopt the single linkage approach or the complete linkage, as discussed above. For instance, using the single linkage approach, the shortest distance between BC and A is 5 (the distance between B and A); conversely, using complete linkage, we would measure a distance

between BC and A equal to 7 (the distance between C and A). Using this logic, and applying the single linkage criterion, the new distance matrix is given in [Table 2.4](#).

Table 2.4 The distance matrix after the first iteration of agglomerative hierarchical clustering, using single-linkage

| | A | BC | D | E | F |
|----|----|----|----|----|----|
| A | 0 | 5 | 4 | 8 | 10 |
| BC | 5 | 0 | 9 | 13 | 6 |
| D | 4 | 9 | 0 | 16 | 15 |
| E | 8 | 13 | 16 | 0 | 18 |
| F | 10 | 6 | 15 | 18 | 0 |

The shortest distance is between A and D, hence we create the cluster AD. Now we have four clusters: AD, BC, E, and F. We obtained a new four-by-four distance matrix.

| | AD | BC | E | F |
|----|----|----|----|----|
| AD | 0 | 5 | 8 | 10 |
| BC | 5 | 0 | 13 | 6 |
| E | 8 | 13 | 0 | 18 |
| F | 10 | 6 | 18 | 0 |

Carrying on with the reasoning, one could show that the next step is to merge AD and BC, forming the cluster ADBC.

| | ADBC | E | F |
|------|------|----|----------|
| ADBC | 0 | 8 | 6 |
| E | 8 | 0 | 18 |
| F | 6 | 18 | 0 |

In the following step F is merged to ADBC; finally, E is merged to ADBCF, and the algorithm stops.

| | ADBCF | E |
|-------|----------|----------|
| ADBCF | 0 | 8 |
| E | 8 | 0 |

This process produces a hierarchy of clusters that is represented using a dendrogram. The dendrogram for the example above is depicted in [Figure 2.7](#).

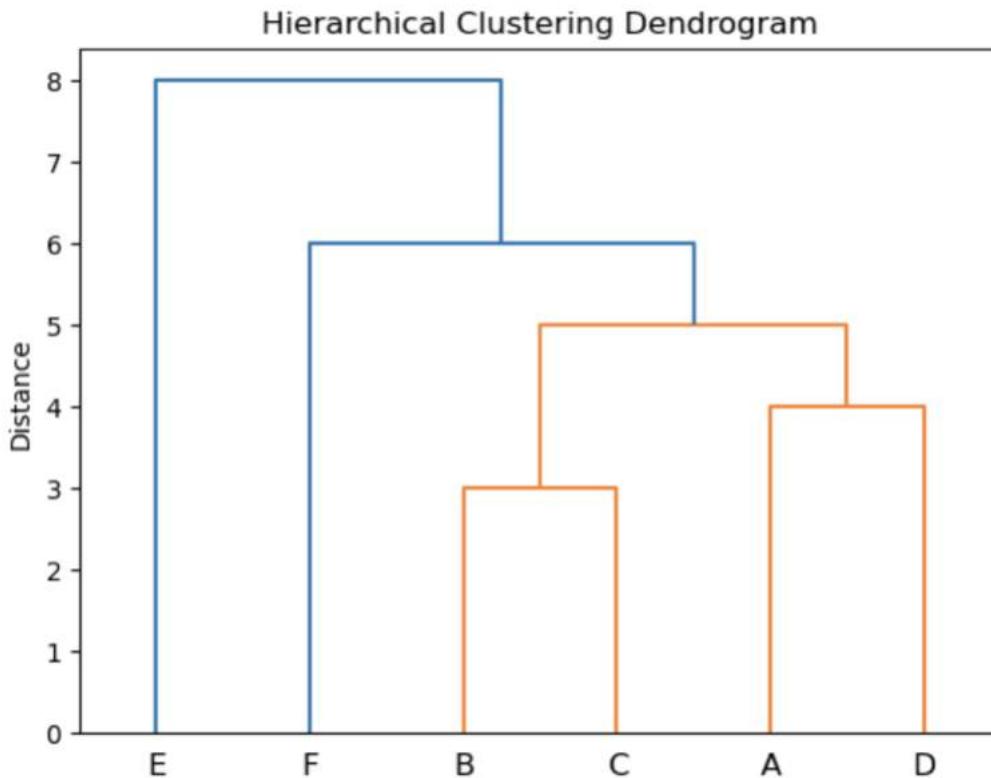


Figure 2.7 Dendrogram for the hierarchical clustering of six fictional objects

We can now look at a more realistic example. Namely, we apply hierarchical clustering to segment the stock and bond data already presented in the k -means clustering example. The resulting dendrogram is plotted in [Figure 2.8](#).

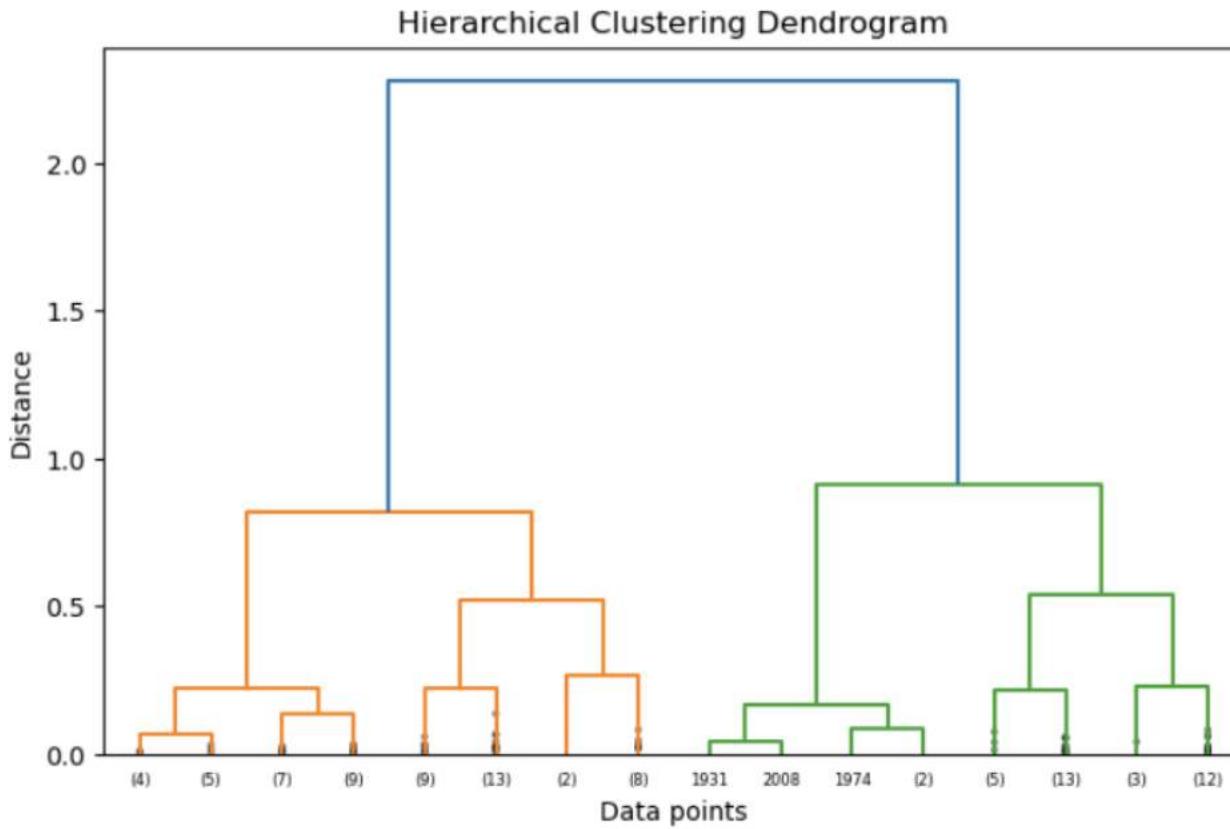


Figure 2.8 Dendrogram for stock and bond data

We specify a cutoff so that only three levels from the initial split are shown to avoid proliferation of tiny clusters culminating in data points at the bottom. The value in parentheses shows the total number of points that are within that cluster, whereas the value without parentheses corresponds to the individual data for a specific year. The distance is calculated using a slightly more sophisticated method known as Ward

linkage, which minimizes the variance of the clusters being grouped.⁵ By applying this method, a dendrogram is constructed, which is shown in Fig. 2.8 above. The y -axis shows the distance between two clusters or data points. For example, the node of the orange line and the blue line shows the distance between clusters (4), (5), (7), (9) and (9), (13), (2), (8) is about 0.75. To select the optimal number of clusters, we examine the heights of each vertical line before a split occurs, which shows how much the distance drops as an additional cluster is introduced. In this case, if splitting the data into two clusters provides a large drop in distance relative to others, it would suggest that a division of cluster is preferred. But if the line is small, it would suggest the incremental benefit in terms of reduced distances is not worthwhile. Like a scree plot, a dendrogram will not be able to show a uniquely correct optimal number of clusters, rather the optimal number will be a matter of interpretation.

⁵ [sklearn.cluster.AgglomerativeClustering — scikit-learn 1.4.0 documentation](#)

2.3 Density-based Clustering

Another approach to clustering is the family of techniques including “DBSCAN” (density-based spatial clustering of applications with noise) and “SNN” (shared nearest neighbors) that are based on densities of points. Here, a region in feature space constitutes a cluster if the density of points exceeds a pre-specified threshold. DBSCAN distinguishes between three types of data points:

1. An observation is a *core point* if at least a pre-specified number of other points are within a threshold distance of it.
2. An observation is a *border point* if it is within the threshold distance from a core point, but it has fewer than the pre-specified number of other points close to it.
3. An observation is considered a *noise point* if it is neither a core nor a border point.

Each core point constitutes a cluster, and the border points are allocated to their nearest core point cluster; noise points are ignored and remain unclustered. The threshold and number of points required to assign an observation as core are two hyperparameters to be tuned.

Although more complex – in terms of both the intuition and the optimization method – density-based clustering has two important advantages. First, it can handle non-spherical distributions of points in feature space, including the situations presented in Figure 2.5. Second, it is considerably more robust with respect to outliers, and indeed observations identified as noise points are automatically excluded from the clusters

Appendix 2.A Different Distance Measures

Manhattan Distance

For two dimensions, the Manhattan distance between two points P and Q , sometimes known as the L^1 -norm, would be calculated as:

$$d_M = |x_{1Q} - x_{1P}| + |x_{2Q} - x_{2P}| \quad (2A.1)$$

Extending this to m dimensions (i.e., m features):

$$d_M = \sum_{j=1}^m |x_{jQ} - x_{jP}| \quad (2A.2)$$

The Euclidean distance is the direct route (the indicated solid line in Figure 2.1), whereas the Manhattan measure gives an approximation to the distance between two buildings that might be required when driving a car (the sum of the two dashed lines in the figure).

Minkowski Distance

More generally, we can nest both the Euclidean and Manhattan distances in a broader framework known as the Minkowski distance measure:

$$d = \left[\sum_{j=1}^m |x_{jQ} - x_{jP}|^L \right]^{1/L} \quad (2A.3)$$

If $L = 1$, the Manhattan measure is calculated; if $L = 2$, it is the Euclidean; finally, if $L = \infty$, it is the *Chebyshev distance* – this is the maximum of the absolute distances along each dimension. In the context of Figure 2.1, this would be the horizontal distance, $|x_{1P} - x_{1Q}|$, which is bigger than the corresponding vertical distance $|x_{2P} - x_{2Q}|$. The Euclidean and Manhattan distance metrics are by far the most common, and hence they will be used in the subsequent examples.

Cosine Similarity

The cosine similarity is the cosine of angle between two vectors, P and Q . By definition,

$$S_c(P, Q) = \cos \theta = \mathbf{P} \cdot \mathbf{Q} / \| \mathbf{P} \| \| \mathbf{Q} \| \quad (2A.4)$$

Expressed in scalar form,

$$Sc(P, Q) = \frac{\sum_{j=1}^m x_{jQ} x_{jP}}{\sqrt{\sum_{j=1}^m x_{jQ}^2 \sum_{j=1}^m x_{jP}^2}} \quad (2A.5)$$

which is bounded to lie in the $[-1, 1]$ interval with all notations as above. In machine learning this measure is used to analyze the similarities between documents.

The cosine distance is defined as the complement of S_c :

$$d = 1 - S_c(P, Q), \text{ or}$$

$$d = 1 - \frac{\sum_{j=1}^m x_{jQ} x_{jP}}{\sqrt{\sum_{j=1}^m x_{jQ}^2 \sum_{j=1}^m x_{jP}^2}}, \quad (2A.6)$$

which is bounded to lie in the [0,2] interval.

Mahalanobis Distance

The Mahalanobis distance is the distance between a data point and a distribution. It is the multivariate equivalent of the Euclidian distance. It can be also viewed as a multivariate version of the Z-score. The Mahalanobis distance is defined as:

$$\mathbf{D}^2 = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \quad (2A.7)$$

where \mathbf{x} is the vector of observations, $\boldsymbol{\mu}$ is the vector of means, and $\boldsymbol{\Sigma}$ is the variance covariance matrix. This distance measure is used for cluster analysis and classification as well as for detecting outliers.

Appendix 2.B Silhouette Method Example

The silhouette method is applied here to the stock and bond data discussed earlier in this chapter.

Figure 2B.1 produces silhouette plots for $K = 2$ up to 9 starting from the top left to the bottom right plot. In each of these plots, the horizontal axis represents the silhouette score and each of the colored areas represent the silhouette scores within a cluster. For example, in the case of two clusters, cluster 1 and cluster 2's constituent scores are plotted in the top left chart. The vertical dashed lines are the average silhouette scores for the value of K , which are 0.57 for two clusters, 0.47 for three clusters, and 0.48 for four clusters, with lower numbers for the remainder. Ideally, the sizes of each silhouette would be the same for a given value of K , which is not the case here for $K \geq 3$. We would also like to see every silhouette having a right-hand tip bigger than the average score line, which holds for all values of K except 9 . When K becomes large, the thicknesses of the silhouettes become increasingly variable, with some thin lines, indicating small clusters, and some negative values, indicating incorrectly clustered points. Overall, we would conclude that $K = 2$ or 3 would be ideal here.

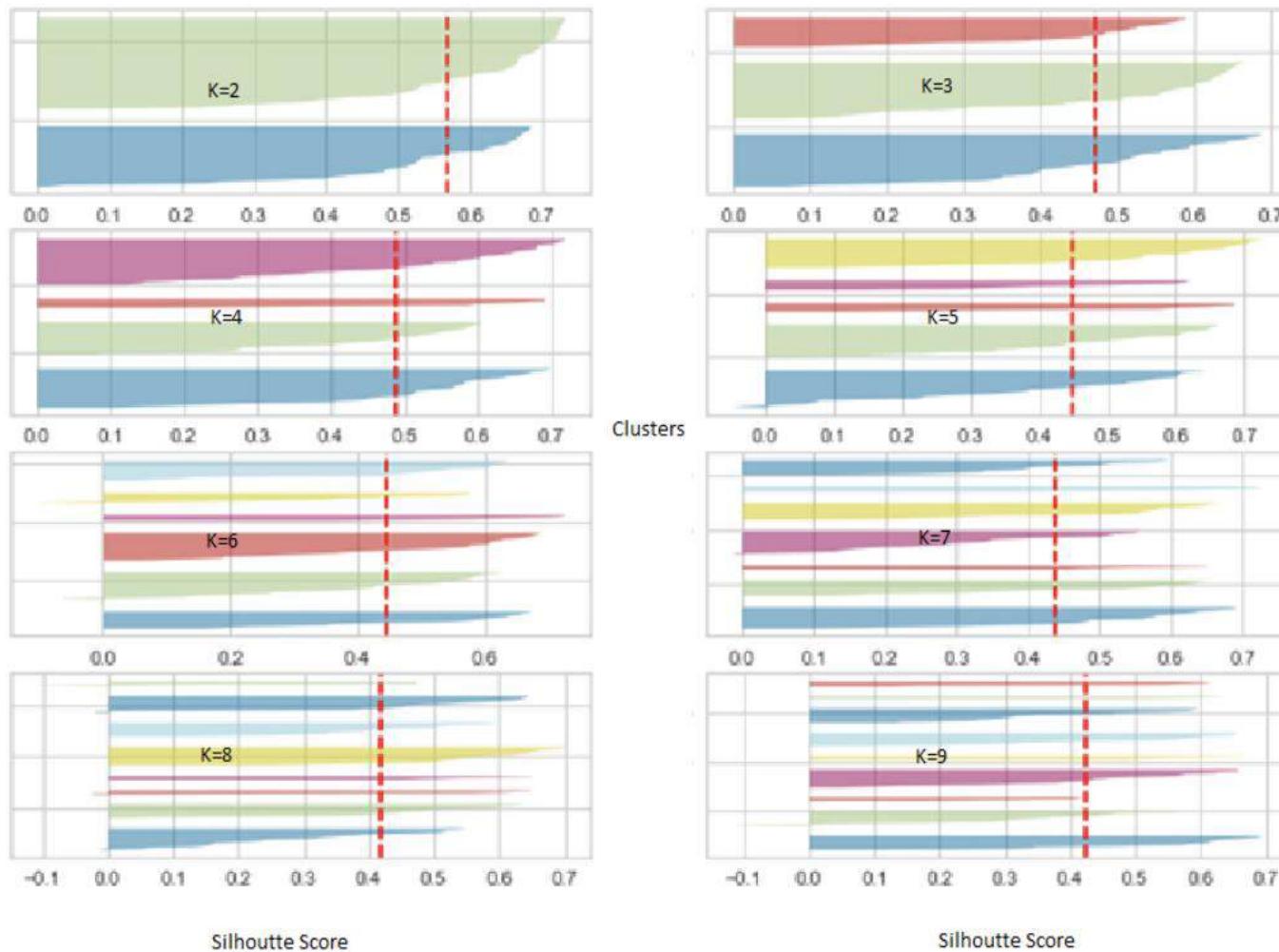


Figure 2B.1 Silhouette plots for the yields and stocks data.

Appendix 2.C K-Means Clustering Example

A retail bank is interested in examining the characteristics of its customers who take a loan to buy a car. It identifies two features (client age, and loan amount in US dollars) for six customers. The data are presented in the following table, in both raw and standardized forms. Use this data to form two clusters using the k -means algorithm, iterating twice with initial centroid positions – assume that these were randomly selected – of (0.5,0.5) and (1,1) in standardized feature space.

| Data point number | Age | Loan amount | Standardized age (SA) | Standardized loan amount (SLA) |
|-------------------|-----|-------------|-----------------------|--------------------------------|
| 1 | 23 | 15000 | -0.96 | -0.75 |
| 2 | 19 | 8000 | -1.19 | -1.48 |
| 3 | 46 | 22000 | 0.38 | -0.02 |
| 4 | 55 | 25000 | 0.90 | 0.30 |
| 5 | 61 | 35000 | 1.25 | 1.34 |
| 6 | 33 | 28000 | -0.38 | 0.61 |

The first stage is to work out the distances from each point individually to each of the centroids. Then a point is allocated to the centroid to which it is closest. We work with

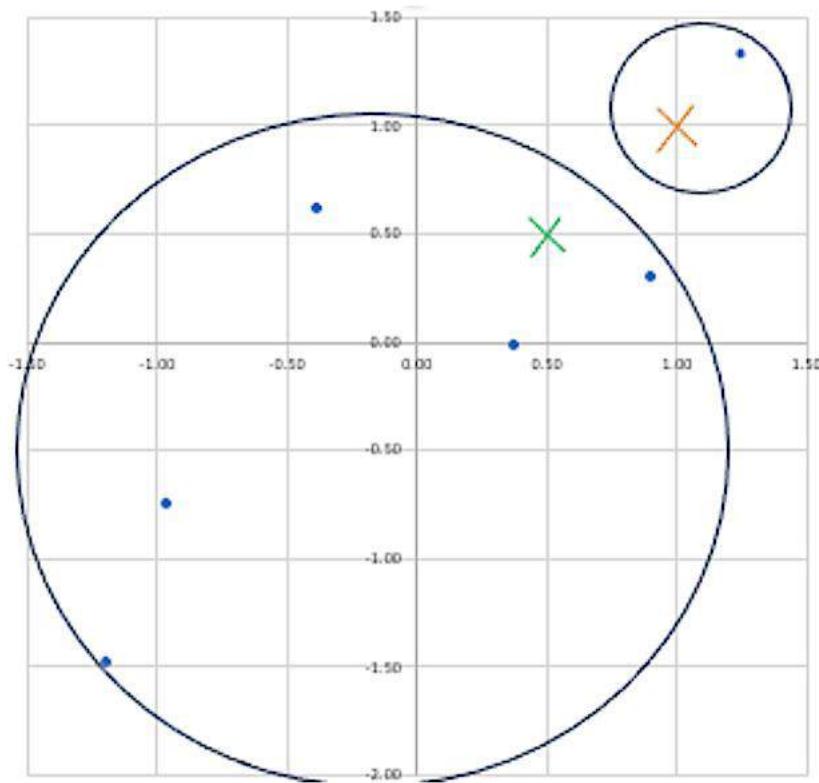
the standardized features and assume that the Euclidean distance measure is used. The results for the first clustering stage are given in the Table 2.C.1.

The calculations are straightforward – for instance, the distance from data point 1 to the (0.5,0.5) centroid is:

and the distance from data point 1 to the (1,1) centroid is:

Because the distance is lower to (0.5,0.5) than (1,1), the point is allocated to the former. Repeating these calculations for the other five data points, we find that all except point 5 are allocated to (0.5,0.5).

The initial centroid positions (denoted by crosses) and clusters formed are shown in the following figure:



Having completed the allocations to first stage clusters, we then reposition the centroids at the centers of those clusters by computing the average values of each co-ordinate over the data points allocated to that cluster. Because there is only one point allocated to (1,1), the position of this centroid will shift to the co-ordinates of

that data point, namely $(1.25, 1.34)$. We take an average of each of the features across the other five data points to get $(-0.25, -0.27)$.

Next, we repeat the process by determining which of the two new centroids each point is closest to. The results are given in Table 2.C.2. Point 4 now switches to the other cluster, but the other four points are still allocated to the same cluster as in the previous stage.

We then recalculate the positions of the centroids, which will differ due to the reallocation of Point 4. The new positions are: $(-0.54, -0.41)$ and $(1.08, 0.82)$. It turns out that when we calculate the distances of the points to each of these new centroids, none of them switch clusters and therefore that is the end of the process and we have identified the optimal clusters positions. The final allocations and centroid positions are shown in the following figure:

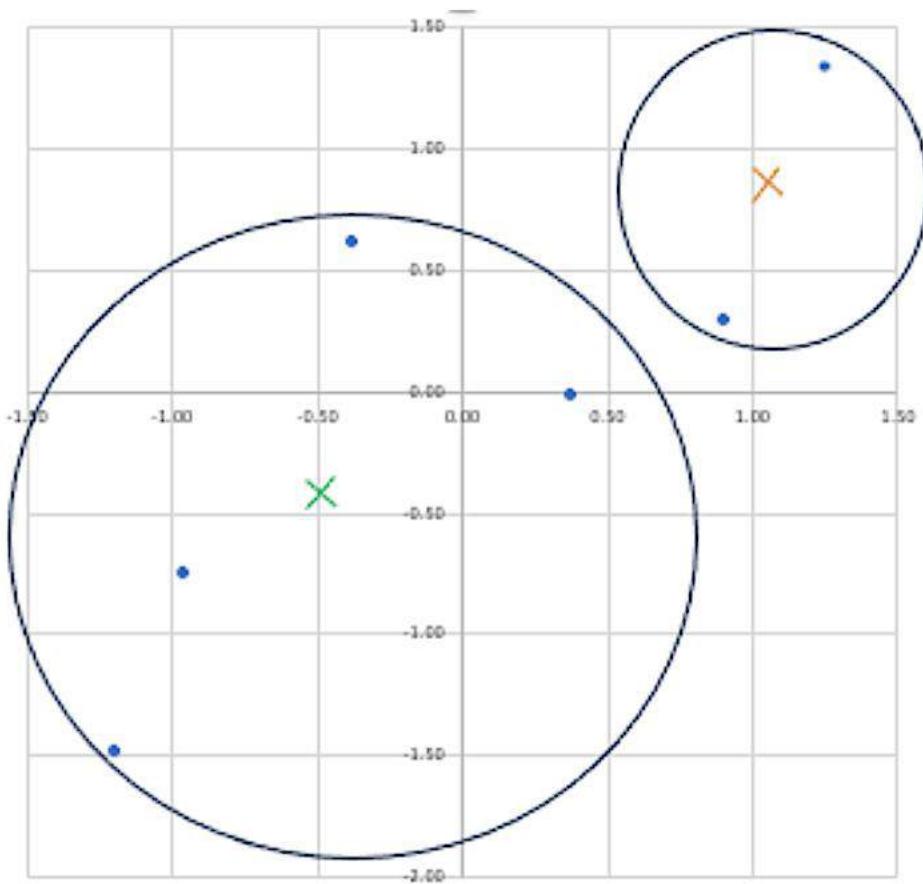


Table 2.C.1 First stage distance calculations and allocations to clusters

| Data point number | Age | Loan amount | Standardized age (SA) | Standardized loan amount (SLA) | Distance to (0.5,0.5) centroid | Distance to (1,1) centroid | Allocation 1st stage |
|-------------------|-----|-------------|-----------------------|--------------------------------|--------------------------------|----------------------------|----------------------|
| 1 | 23 | 15000 | -0.96 | -0.75 | 1.92 | 2.63 | (0.5,0.5) |
| 2 | 19 | 8000 | -1.19 | -1.48 | 2.6 | 3.31 | (0.5,0.5) |
| 3 | 46 | 22000 | 0.38 | -0.02 | 0.53 | 1.19 | (0.5,0.5) |
| 4 | 55 | 25000 | 0.9 | 0.3 | 0.45 | 0.71 | (0.5,0.5) |
| 5 | 61 | 35000 | 1.25 | 1.34 | 1.13 | 0.42 | (1,1) |
| 6 | 33 | 28000 | -0.38 | 0.61 | 0.88 | 1.43 | (0.5,0.5) |

Table 2.C.2 Second stage distance calculations and allocations to clusters

| Data point number | Age | Loan amount | Standardized age (SA) | Standardized loan amount (SLA) | Distance to (-0.25, -0.27) centroid | Distance to (1.25, 1.34) centroid | Allocation 2nd stage |
|-------------------|-----|-------------|-----------------------|--------------------------------|-------------------------------------|-----------------------------------|----------------------|
| 1 | 23 | 15000 | -0.96 | -0.75 | 0.86 | 3.04 | (-0.25, -0.27) |
| 2 | 19 | 8000 | -1.19 | -1.48 | 1.53 | 3.73 | (-0.25, -0.27) |

| Data point number | Age | Loan amount | Standardized age (SA) | Standardized loan amount (SLA) | Distance to (-0.25, -0.27) centroid | Distance to (1.25, 1.34) centroid | Allocation 2nd stage |
|-------------------|-----|-------------|-----------------------|--------------------------------|-------------------------------------|-----------------------------------|----------------------|
| 3 | 46 | 22000 | 0.38 | -0.02 | 0.68 | 1.61 | (-0.25, -0.27) |
| 4 | 55 | 25000 | 0.9 | 0.3 | 1.28 | 1.1 | (1.25, 1.34) |
| 5 | 61 | 35000 | 1.25 | 1.34 | 2.2 | 0 | (1.25, 1.34) |
| 6 | 33 | 28000 | -0.38 | 0.61 | 0.89 | 1.78 | (-0.25, -0.27) |

Table 2.C.3 Third stage distance calculations and allocations to clusters

| Data point number | Age | Loan amount | Standardized age (SA) | Standardized loan amount (SLA) | Distance to (-0.54, -0.41) centroid | Distance to (1.08, 0.82) centroid | Allocation 3rd stage |
|-------------------|-----|-------------|-----------------------|--------------------------------|-------------------------------------|-----------------------------------|----------------------|
| 1 | 23 | 15000 | -0.96 | -0.75 | 0.54 | 2.57 | (-0.54, -0.41) |
| 2 | 19 | 8000 | -1.19 | -1.48 | 1.25 | 3.23 | (-0.54, -0.41) |

| Data point number | Age | Loan amount | Standardized age (SA) | Standardized loan amount (SLA) | Distance to (-0.54, -0.41) centroid | Distance to (1.08,0.82) centroid | Allocation 3rd stage |
|-------------------|-----|-------------|-----------------------|--------------------------------|-------------------------------------|----------------------------------|----------------------|
| 3 | 46 | 22000 | 0.38 | -0.02 | 1.00 | 1.09 | (-0.54, -0.41) |
| 4 | 55 | 25000 | 0.9 | 0.3 | 1.60 | 0.55 | (1.08,0.82) |
| 5 | 61 | 35000 | 1.25 | 1.34 | 2.50 | 0.55 | (1.08,0.82) |
| 6 | 33 | 28000 | -0.38 | 0.61 | 0.89 | 1.47 | (-0.54, -0.41) |

Unsupervised Learning: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

2.1 How would you choose the number of clusters when using unsupervised learning?

The more clusters are used when fitting an unsupervised model, the better the fit of the algorithm to the data, but as the number of clusters increases, the usefulness of the model will start to diminish.

Determining the most appropriate number of clusters for a particular dataset could involve constructing a “scree plot,” which charts the inertia (sum of squared distances of each point to its centroid) against the number of clusters. We would then search for the number of clusters beyond which the inertia only declines very slowly. Silhouette scores, which

compare the distance of each point (a) to points in its own cluster and (b) to points in the closest other cluster, can also be used.

2.2

A. Explain the steps in using the k -means clustering algorithm.

1. *Specify the number of centroids, K and choose a distance measure (e.g., the Euclidean or Manhattan distance).*
2. *Scale the features using either standardization or normalization.*
3. *Select K points at random from the training data to be the centroids*
4. *Allocate each data point to its nearest centroid.*
5. *Given the points allocated to each centroid, re-calculate the appropriate location of the centroids.*
6. *If the positions of the centroids have changed from those in the previous iteration, then repeat step 4. If the positions of the centroids have not changed (and the clusters are not changed), then stop.*

B. In practice, the k -means clustering algorithm is often carried out with several different initial values for the centroids. How would you choose between clusters that result from different initial choices for the centroids?

You could select the centroids where the total inertia was the lowest, as this would represent the choice of centroid positions that best fitted the feature data.

C. How do you use the scree plot for choosing the number of k -means clusters?

A scree plot has the number of clusters on the x-axis and the within-cluster sum of squares on the y-axis, which is a measure of how closely the points allocated to each cluster fit their respective centroids. We look for an 'elbow' in the plot, where its gradient changes from steep to almost flat and that is the optimal K .

2.3 What are the two types of hierarchical clustering? What are the advantages of hierarchical clustering?

Hierarchical clustering starts with all points in one cluster and then sequentially splits them into separate clusters until an optimal allocation is reached (divisive hierarchical clustering) or starts with each data point in its own cluster and sequentially combines them until an optimal allocation is reached (agglomerative hierarchical clustering).

The advantages of hierarchical clustering are first, that it does not require pre-defining the number of clusters and second, it can uncover hierarchical relationships within the data, which can reveal nested clusters within larger groups. Third, the dendograms produced are straightforward to interpret.

2.4 State whether the following are true or false and explain your answers.

A. *K*-means with Euclidean distance can only be used when the clusters are approximately spherical.

True. Because K-means is based on linear Euclidean distances, it runs into problems when the cluster is not approximately spherically shaped. When the Manhattan distance measure is used with K-means, the clusters are approximately rhombus-shaped. This can result in poorly defined clusters, or some points even being allocated to the wrong cluster.

B. The within cluster sum of squares will never rise when the number of clusters is increased in a *K*-means application.

True. The situation is analogous to what happens to the residual sum of squares when more features are added to a linear regression model. When additional clusters are added (i.e., the value of K is increased), the fit of the model to the data cannot get worse. Therefore, the within-cluster sum of squares must fall as the new cluster will capture one or more of the data points better than the cluster(s) to which it(they) was(were) previously allocated.

2.5 What is a dendrogram and how would we interpret one?

A dendrogram is a pictorial representation of the steps in a hierarchical clustering application, which shows how the clusters are split or combined. Each bifurcation (for divisive clustering) or combination (for agglomerative clustering) of the lines shows a cluster being formed or removed, respectively. The heights of the vertical lines show the impact of the marginal cluster on the distance of the points affected by it to their nearest cluster. If a particular vertical line is

long, this would suggest that the additional cluster has a considerable effect on the model fit and therefore is worth incorporating.

2.6 Suppose that we have the following data on three features for each of three banks, A, B, and C:

| Features | Bank A | Bank B | Bank C |
|----------------------------------|--------|--------|--------|
| Number of customers (millions) | 1.2 | 6.0 | 0.5 |
| Total size of loan book (USD bn) | 5 | 25 | 7 |
| Number of branches | 80 | 400 | 50 |

Determine the centroid of a cluster comprising banks A, B, and C using the raw (unscaled) data.

The centroid is simply the average of the feature values across the three banks. So, if we define the co-ordinate space (i , j , k) as the three-dimensional point, with the raw data in their original units as presented in the question table, the centroid would be given by

which is $(2.567, 12.333, 176.667)$.

Chapter 3: Supervised Learning for Numerical Data- Part 1, Econometric Techniques

Learning Objectives

This chapter covers the core models used for supervised learning that arise from econometrics. Broken out in two main parts, the chapter first examines linear regression models that constitute the foundation upon which more sophisticated approaches are built. It then provides an overview of the types of models used in the context of classification problems including logistic regression and linear discriminant analysis.

Supervised learning techniques originating from computer science and more commonly associated with machine learning will be covered in Chapter 4.

After completing this chapter, you should be able to:

- Identify uses and limitations of the following models:
 - Single and multi-variable linear regression
 - Single and multi-variable non-linear regression
- Interpret the results of the following regression analyses:
 - Single and multi-variable regression

- Single and multi-variable non-linear regression
- Identify problems that may occur with linear regression models and possible remedies for them.
- Describe how logistic regression can be applied to classification problems.
- Describe the use of linear discriminant analysis for classification problems.

This chapter covers the core models used for supervised learning that arise from econometrics. Techniques originating from computer science and more commonly associated with machine learning will be covered in the following chapter. Although some of the material discussed in this chapter will already be familiar to readers if they have studied a course in statistics or econometrics, where feasible the models will be cast in the language and notation of machine learning in order to make the transition to machine learning techniques in Chapter 4 as seamless as possible.

The chapter has two main parts: the first examines linear regression models that constitute the foundation upon which more sophisticated approaches are built. Linear regression models are used for situations where we want to model a continuous variable as a function of one or more features, and they are used primarily for prediction. The second part of chapter examines the classes of models used in the context of classification problems including logistic regression and linear discriminant analysis.

3.1.1 Simple Linear Regression

We begin by outlining the most basic of specifications, namely the simple linear regression model, sometimes known as bivariate regression because there are only two variables: the output or target variable, y , and the feature or input variable, x .

$$y_i = \beta_0 + \beta_1 x_i + u_i \quad (3.1)$$

The model postulates that y varies because of changes in x . Here, y is a linear function of x and an unobservable error term, u with mean zero and constant variance; x and y are observable variables (y is the target and x the feature in machine learning parlance), whereas β_0 and β_1 are parameters to be estimated. Using econometric terminology, β_0 is the intercept parameter, and it is interpreted as the value that y would take if x were zero, and β_1 measures the impact on y of a unit change in x . In machine learning, β_0 is known as the bias, and β_1 is the weight. The index i for each variable, or feature, denotes the observation number ($i = 1, \dots, N$ where N is the total number of data points, or instances available for each variable).

The simple linear regression model is used in situations where it is only of interest to examine the effects of one feature on the target variable, such as evaluating the market risk level of a stock using the capital asset pricing model (CAPM) or estimating an optimal hedge ratio.

There are three main methods for estimating the parameters of a regression model:

1. Least squares

2. Maximum likelihood

3. The method of moments

The first two of these three estimation techniques are discussed in detail in Chapter 7, but a specific case of the first technique from this list, known as *ordinary least squares* (OLS), is the most straightforward approach and hence it is most used for linear regression models.

The model in equation (1) is known as a linear regression model because it embodies a linear relationship that can be represented by a straight line. Being slightly more specific, the model is both *linear in the parameters* (the equation is a linear function of β_0 and β_1) and *linear in the variables* (linear with respect to y and x). To use OLS, the model must be linear in the parameters, although it does not necessarily have to be linear in the features. We will examine some examples of models that are non-linear in the features in the following sub-section.

3.1.1 Simple Linear Regression

We begin by outlining the most basic of specifications, namely the simple linear regression model, sometimes known as bivariate regression because there are only two variables: the output or target variable, y , and the feature or input variable, x .

$$y_i = \beta_0 + \beta_1 x_i + u_i \quad (3.1)$$

The model postulates that y varies because of changes in x . Here, y is a linear function of x and an unobservable error term, u with mean zero and constant variance; x and y are observable variables (y is the target and x the feature in machine learning parlance), whereas β_0 and β_1 are parameters to be estimated. Using econometric terminology, β_0 is the intercept parameter, and it is interpreted as the value that y would take if x were zero, and β_1 measures the impact on y of a unit change in x . In machine learning, β_0 is known as the bias, and β_1 is the weight. The index i for each variable, or feature, denotes the observation number ($i = 1, \dots, N$ where N is the total number of data points, or instances available for each variable).

The simple linear regression model is used in situations where it is only of interest to examine the effects of one feature on the target variable, such as evaluating the market risk level of a stock using the capital asset pricing model (CAPM) or estimating an optimal hedge ratio.

There are three main methods for estimating the parameters of a regression model:

1. Least squares
2. Maximum likelihood
3. The method of moments

The first two of these three estimation techniques are discussed in detail in Chapter 7, but a specific case of the first technique from this list, known as *ordinary least squares* (OLS), is the most straightforward approach and hence it is most used for linear regression models.

The model in equation (3.1) is known as a linear regression model because it embodies a linear relationship that can be represented by a straight line. Being slightly more specific, the model is both *linear in the parameters* (the equation is a linear function of β_0 and β_1) and *linear in the variables* (linear with respect to y and x). To use OLS, the model must be linear in the parameters, although it does not necessarily have to be linear in the features. We will examine some examples of models that are non-linear in the features in the following sub-section.

3.1.2 Multiple Linear Regression

In most practical contexts, having a model with a single feature is not sufficiently flexible to capture all the variability in the target variable, and we can build a much better model by including more than one feature. Such a model is known as a multiple linear regression. To incorporate multiple features in the model, we simply extend the model in equation 3.1 to include m features:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \cdots + \beta_m x_{mi} + u_i \quad (3.2)$$

In the multiple linear regression model, there will be $m + 1$ parameters ($m \geq 1$) to estimate: one for the intercept, β_0 , and one for each of the m slope parameters. Once we have data on y and x_1, \dots, x_m , again, OLS can be used to estimate the parameters. In the multiple linear regression model, each parameter measures the partial effect of the attached variable after controlling for the effects of all the other features included in the regression.

Even within this straightforward framework, we can nonetheless incorporate a wide range of specifications. For instance, it is common to apply a logarithmic transformation to some or all the feature variables and/or to the output variable. Such a transformation would imply a different interpretation of the parameter estimates but OLS could still be used as the model would remain linear in the parameters.

Alternatively, we could incorporate interaction terms (i.e., features multiplied together) or power terms of the features. For instance, a model including an interaction term is:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{1i}x_{2i} + u_i \quad (3.3)$$

Here, y depends not only on the levels of x_1 and x_2 , but also on how they work together. Hence β_3 will capture any complementarity between them, where changes in both variables are required to have an impact on y rather than in isolation. A commonly employed example is how the amounts of water and fertilizer affect crop

yields, because an abundance of one and none of the other will not lead to high yields. Hence the amount of one of them influences the effectiveness of the other, implying a need to model them jointly.

And a model including power terms is:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{2i}^2 + u_i \quad (3.4)$$

Here, only a squared term on x_2 is included in the model, and it is common to stop there, allowing for a quadratic relationship between x_2 and y . But in principle it would be feasible to include cubed terms, fourth-order powers, and so on. However, when adding further terms, one must be mindful not to overfit the data.¹

To use OLS for model estimation, the output variable y must be continuous, but the features could be continuous or discrete. Examples of continuous variables include returns and yields, whereas a particularly relevant case of discrete variables are categorical variables (which are encoded using dummy variables) as outlined in Chapter 1.

It would be instructive at this point to examine an example to see how regression models work and importantly, how the parameter estimates are interpreted. A simple linear regression and a quadratic regression model are discussed in [Box 3.1](#).

Box 3.1: The effect of experience and having a degree on wage rates

Suppose that we have the data given in [**Table 3.1**](#), which show the hourly salaries in US dollars of 14 notional employees at a US retail bank alongside the number of years of experience that each has, the square of the number of years of experience, and a dummy variable capturing whether they have a college degree.

Table 3.1: Salary, experience, and degree dummy variable for notional sample of bank employees

| Experience | Experience ² | Degree | Salary |
|------------|-------------------------|--------|--------|
| 1 | 1 | 0 | 15.46 |
| 3 | 9 | 1 | 22.40 |
| 7 | 49 | 1 | 27.47 |
| 12 | 144 | 1 | 34.31 |
| 9 | 81 | 0 | 33.08 |
| 3 | 9 | 0 | 18.70 |
| 20 | 400 | 1 | 35.06 |
| 22 | 484 | 0 | 35.78 |
| 0 | 0 | 1 | 14.81 |
| 4 | 16 | 1 | 14.84 |
| 6 | 36 | 0 | 25.78 |

| Experience | Experience^2 | Degree | Salary |
|------------|--------------|--------|--------|
| 8 | 64 | 0 | 28.17 |
| 4 | 16 | 1 | 26.36 |
| 2 | 4 | 0 | 11.12 |

In this example we believe that salary should be driven by experience. So, to begin with, we run a simple linear regression of salary (y) on experience (x_1), and we would obtain the following regression results:

$$\widehat{\text{salary}}_i = 16.88 + 1.06 \text{ experience}_i \quad (3.5)$$

We use a hat (^) above the output variable to denote the fitted equation from the regression line. Here, the intercept estimate is 16.88, meaning that someone just joining the bank with no experience could expect to earn \$16.88 per hour on average, and each additional year of experience would lead wages to increase by an average of \$1.06 per hour (the slope estimate, β_1). A scatter plot of experience against salary is given in [Figure 3.1](#) together with the fitted regression line.

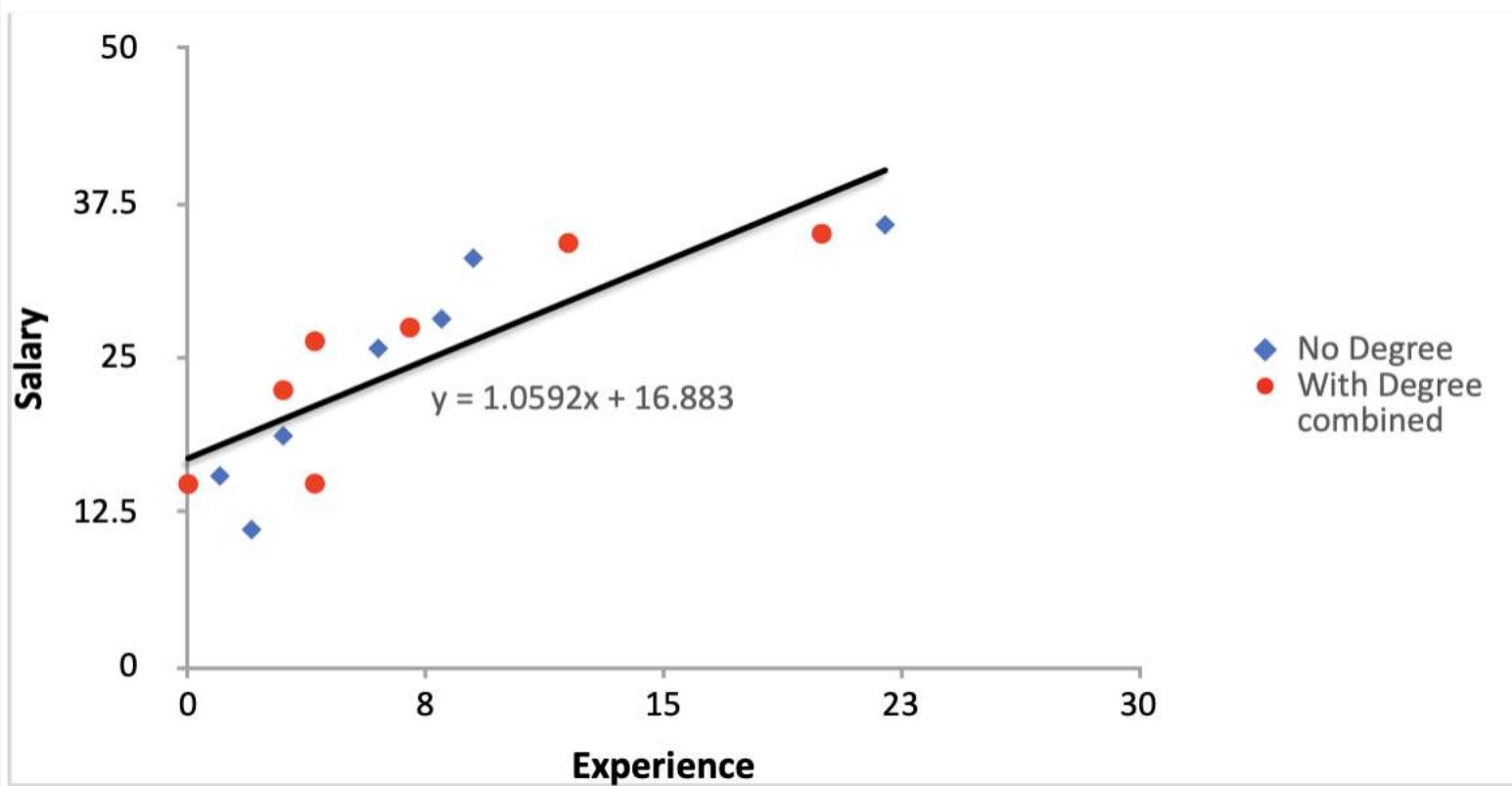


Figure 3.1: Experience versus salary and fitted straight line

The plot suggests that the line fits the data well, although it seems that additional years of experience lead wages to increase significantly when experience is low but the impact tails off after a while. It appears from the diagram that each additional year of experience leads to a lower incremental increase in wages. We cannot capture

this with a linear model, because it embodies a fixed gradient of the fitted line and therefore a fixed relationship between y and x whatever the value of x .

To capture this potential non-linearity in the relationship between wages and experience, we can fit a quadratic model, where the square of experience is included in the model as a second feature. The fitted model is now:

$$\widehat{\text{salary}}_i = 11.82 + 2.77 \text{ experience}_i - 0.08 \text{ experience}_i^2 \quad (3.6)$$

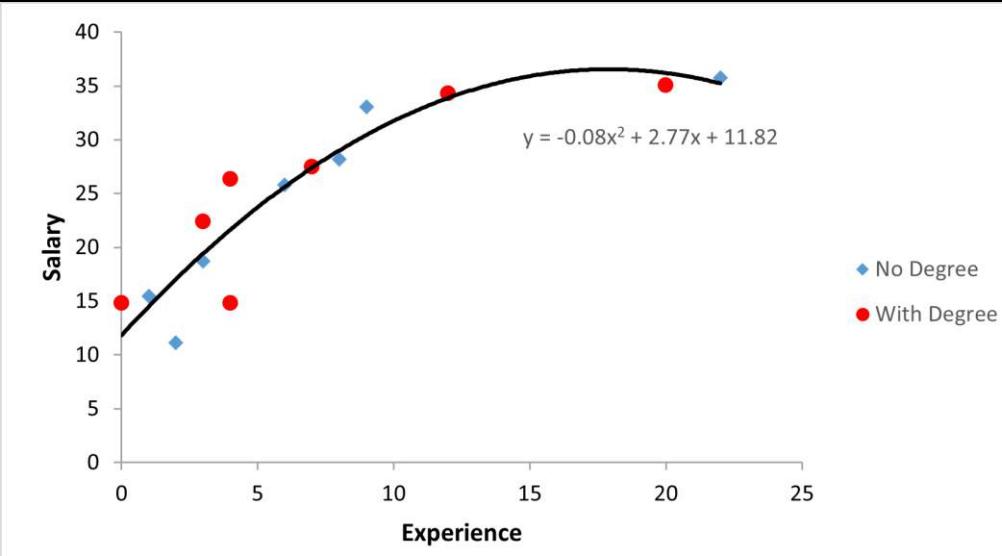


Figure 3.2: Experience versus salary and fitted quadratic equation

Note that when the square of experience is added, the other parameter estimates change compared to their values when only the level of experience was included in the model. Now that we have two features relating to experience, determining the relationship between experience and salary is slightly trickier. But to do this, we differentiate the expression with respect to experience:²

$$\frac{\partial \widehat{\text{salary}}_i}{\partial \text{experience}_i} = 2.27 - 2 \times 0.08 \text{ experience}_i \quad (3.7)$$

It is clear from equation (3.7) that when the square of experience is included, the relationship between salary and experience is no longer constant with respect to the latter. Setting the expression in (3.7) to zero and rearranging it to make experience the subject of the formula will show the value of the feature that maximizes expected salary, which is 29.6 years. The fitted quadratic relationship between experience and salary is plotted in [Figure 3.2](#), where the fit to the data is now noticeably better.

The regression model now includes two variables: experience and its squared value, but there is data on a third feature in [Table 3.1](#) that could affect an employee's wage rate, namely whether they have a degree-level qualification. This is captured by a binary dummy variable taking the value 1 if the person has a degree and zero if they do not. The fitted model adding this feature too is:

$$\widehat{\text{salary}}_i = 11.38 + 2.76 \text{ experience}_i - 0.08 \text{ experience}_i^2 + 0.92 \text{ degree}_i \quad (3.8)$$

The parameter estimate on the degree dummy is 0.92, which is interpreted as meaning that an employee with a degree could expect to earn an additional 92 cents per hour on average compared with someone having identical experience but no degree.

If we had more than one dummy variable in the model to capture other qualitative information, we would interpret the associated parameter estimates in the same way. Dummy variables can also be used in time-series regressions to capture seasonal behavior in the output such as day-of-the-week effects or diurnal patterns.

¹ Please see chapter 7 for a discussion on over- and under-fitting.

² This equation is the result of taking the first derivative of salary equation. The derivative of a constant is zero, whereas the derivative of a constant*x is the constant itself, and the derivative of a constant *x² is 2*constant*x.

3.1.3 Potential Problems With Regressions

Although linear regression models are easy to estimate and straightforward to interpret, several aspects can go wrong. In such cases, although we can usually still estimate values for the parameters, they might no longer be optimal or reliable. This section will now proceed to discuss several common problems with regression models,

known as *model misspecifications*, identifying why these issues might occur, how we can detect them, and how they might be avoided or mitigated.

Problem 1: Wrong Features or Wrong “Functional Form”

In the example above, we assumed that the most appropriate features to model the variation in wage rates for bank employees were based on their experience and degree-level qualifications. But it might be that the model did not include the most relevant features, and broadly there are three ways that the model could be wrong:

1. *The model omits some relevant features.* This would occur if the true relationship describing the output includes some extra features that the researcher has not included in the model – for example, because of a lack of data or a lack of awareness of their relevance. This can be a serious potential misspecification that could lead the parameter estimates to be biased and not to become more accurate as the sample size increases.
2. *The model includes some irrelevant features.* This is less serious than the first misspecification, but can result in “inefficiency,” where the parameters are not estimated precisely. A further consequence is that it is likely the model will find it hard to generalize from the specific training sample to the test sample.
3. *The model includes the correct features, but they are incorporated in the wrong way.* This is known as an *incorrect functional form*. It could occur, for instance, if the true relationship

between the features and the output is non-linear but a linear regression model is used.

These three problems are all more challenging to resolve in practice than they appear because the researcher never knows the true relationship between the variables. This is where a strong theoretical knowledge of the problem at hand and the wider context can be valuable in guiding the model development, rather than a purely data-driven approach.

Problem 2: Multicollinearity

Multicollinearity occurs when the features are highly related to one another. We can draw a distinction between two degrees of multicollinearity: perfect and near. *Perfect multicollinearity* occurs when two or more of the features have an exactly linear relationship that holds for every data point – for example, if $x_{2i} = 10x_{3i}$ for all values of i . In such cases, there is insufficient information to be able to estimate the parameters on both x_{2i} and x_{3i} , and hence the only solution is to remove one of these perfectly correlated variables from the model.

A slightly different situation is where two or more of the variables are closely, but not perfectly, correlated with one another – for instance, if the correlation between x_{2i} and x_{3i} is 0.9. This would be known as *near multicollinearity*, and in such circumstances, the estimation technique would find it hard to disentangle the separate influences of each variable. A common consequence is that the parameter estimates become highly unstable, changing wildly when a feature is added or

removed from the model. There are various empirical approaches to dealing with near multicollinearity. These include the removal of one or more of the highly correlated variables or turning them into a ratio or difference rather than including them individually. Another way forward is to use *regularization*, as described in Chapter 7.

Problem 3: Outliers

Although there is not a widely accepted formal definition of an *outlier*, in broad terms it refers to an anomalous data point that lies a long way from the others. Outliers can have a considerable effect on the estimated parameters. The least squares technique used to estimate the parameters in a regression model takes the sum of squares of the distances from the data points to the fitted line, and the process of squaring these distances means that points that are a considerable distance from the others will exert a disproportionate effect on the estimates (see Chapter 7 for further details on estimation methods).

The situation that occurs if there is an outlier in the data is illustrated in [Figure 3.3](#). In the left-hand diagram, there are ten data points plotted in a scatter diagram, one of which is an outlier. The line of best fit is drawn onto the diagram, and the parameter estimates are:

The right-hand diagram shows the result when the outlier is removed from the dataset and the parameters are estimated using the remaining nine points. The estimates are now:

which are substantively different.

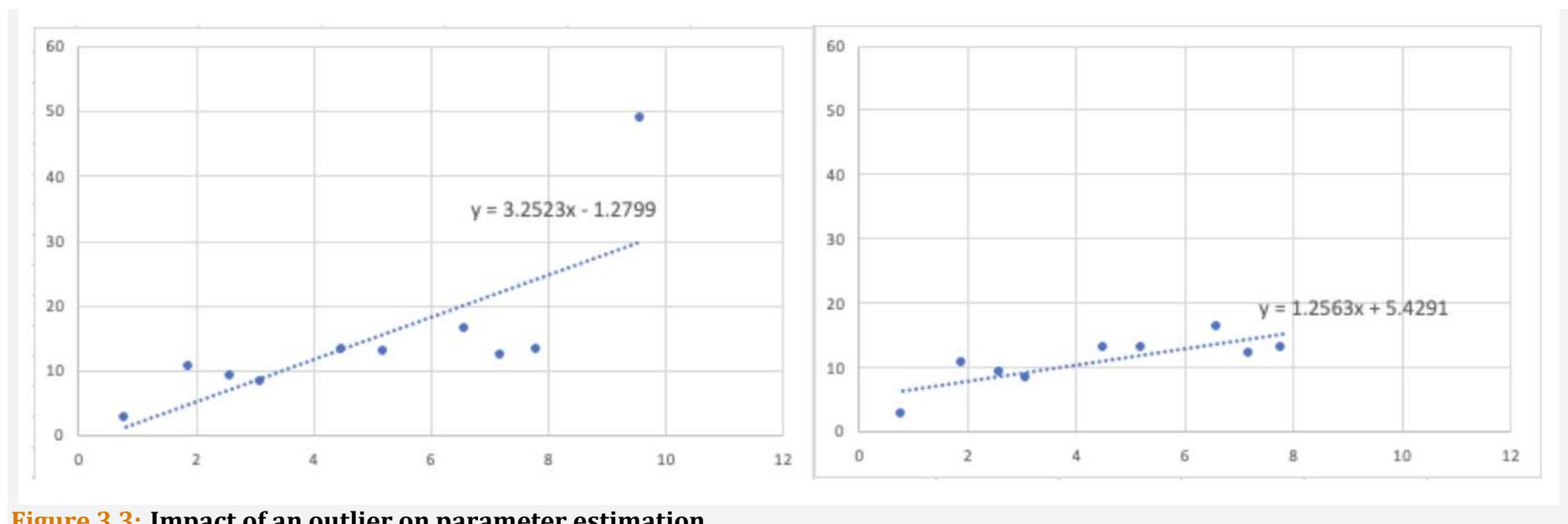


Figure 3.3: Impact of an outlier on parameter estimation

Outliers can be detected examining a plot of the residuals (the difference between the actual data points and the corresponding values fitted from the regression line) and noting any points that lie further from the line than others. Note that if both the input and output values are further from the other datapoints but the point nonetheless lies near the regression line, this would not be classified as an outlier.

A more sophisticated approach to outlier detection is to calculate Cook's distance, which measures the influence of each individual data point on the parameter estimates. This is achieved by removing each data point separately from the

regression and determining the difference in model fit for all the remaining points. If a particular data point is not very influential for parameter estimation (hence it is not an outlier), the model fit will not change much when it is removed from the sample and Cook's distance will be small.

Problem 4: Heteroskedasticity

For OLS to estimate the parameters optimally, a number of assumptions are required regarding the unobservable error term (as in, for example, equation (3.2)). Among these is an assumption that the variance of this error term is constant and finite, which is known as the *homoskedasticity* assumption. If the assumption does not hold, and the variance is not constant, this is known as *heteroskedasticity*. It occurs frequently in time-series of stock and bond returns and is usually also present in the residuals from models of these series.

Heteroskedasticity can lead to several issues with regression estimation, most notably that it becomes *inefficient* and that it is hard to accurately evaluate the empirical importance of each feature for determining the output.

As for outlier detection, a residual plot can sometimes be useful in detecting heteroskedasticity, where we would be looking for whether the spread of the residuals around their mean (usually of zero) is constant or systematically changing. It is common to plot the residuals on the y -axis against the fitted values from the model on the x -axis, and [Figure 3.4](#) shows a stylized example of what heteroskedasticity might look like in such a diagram. Alternatively, there are various formal statistical

tests for heteroskedasticity. A popular such test is the Goldfeld-Quandt test, which splits the sample into two parts and statistically compares the residual variances between the two.

An alternative is White's test, which involves obtaining the residuals, u_i , from a regression such as (3.2) and conducting a second ("auxiliary") regression of the squared residuals (u_i^2) on features (such as x_{1i} and x_{2i}), the squares of features (such as x_{1i}^2 and x_{2i}^2) and the interactions between features (such as $x_{1i}x_{2i}$). If there is no heteroskedasticity, the parameter estimates from this auxiliary regression will not be statistically significant.

Heteroscedasticity Example

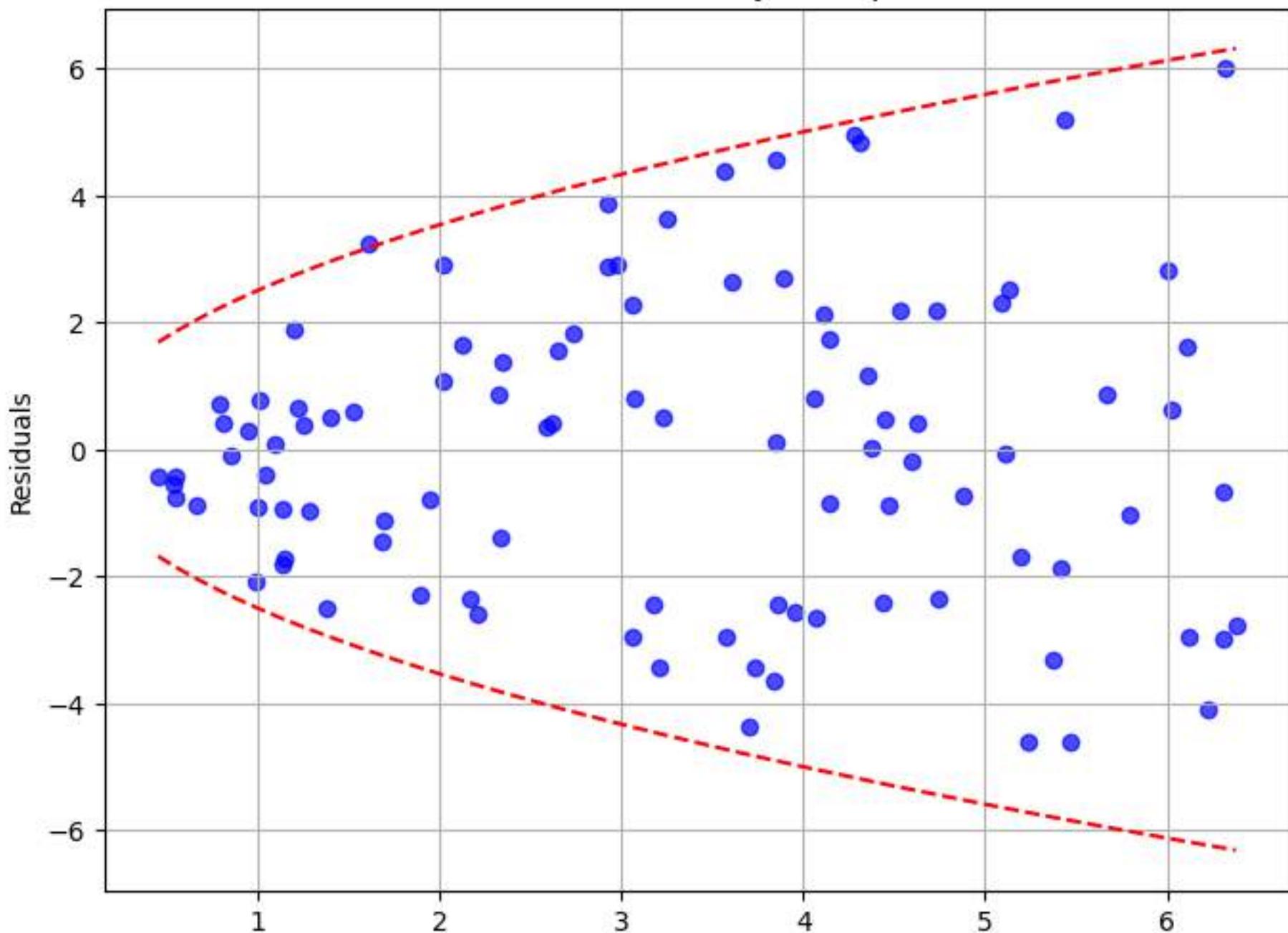


Figure 3.4 Plot of fitted values against residuals to demonstrate heteroskedasticity

Several remedies exist for heteroskedasticity. One approach is to weight the observations to account for the changing error variance, using a technique known as weighted least squares (WLS) instead of OLS. Alternatively, making a logarithmic transformation of the variables and using these in place of the raw variables in the regression model can also help to resolve the issue.

We finish this section by noting that there are further problems that can occur with regression models, such as if there is measurement error in the features. But to address these issues requires more sophisticated approaches that we do not have the space to cover here.

3.1.4 Stepwise Regression Procedures

As discussed above, the presence of non-informative or redundant features in a linear regression model can add uncertainty to the predictions and reduce the effectiveness of the model, especially in the presence of highly correlated features. This introduces a need for methods to remove non informative predictors. *Stepwise regression*, like the LASSO regularization technique described in Chapter 7, is a method for *feature selection*. It belongs to the category of *wrapper* methods, which add or remove predictors to a regression with the aim of finding the combination that maximizes the model performance. The inclusion or exclusion of a feature is based on a criterion that measures the predictive accuracy of alternative sets of predictors. A popular approach

is to choose the model that minimizes the Akaike Information Criterion (AIC), which is a measure of prediction errors adjusted to account for the number of features in the model (we cover it in more detail in Chapter 8). Unlike R-squared, the AIC penalizes large models and therefore it can either increase or decrease when an additional feature is added to the model.

There are various stepwise procedures, but the most straightforward are the **unidirectional forward and backward stepwise selection methods**. The *forward stepwise selection* method starts with a model with no features and includes additional features into the model one-at-the-time starting from those that produce the largest drop in the AIC. The procedure stops when the addition of any new predictor fails to decrease the AIC. Conversely, the *backward stepwise selection* method begins with the full model and removes the predictors one-by-one starting with the least important, until any further elimination fails to decrease the AIC.

Table 3.2 An illustration of the different steps involved in forward and backward stepwise selection methods for a regression of house prices on their ages and ages' powers up to the fifth.

| | Forward Selection | Backward Selection |
|--------|---|--|
| Step 0 | No predictors (AIC: 3202.69) | All predictors: (AIC: 3122.82) |
| Step 1 | Age (AIC: 3193.36) * Age ⁵ (AIC: 3202.2) Age ² (AIC: 3203.33) Age ⁴ (AIC: 3203.63) Age ³ (AIC: 3204.66) | All – Age ⁴ (AIC: 3120.91) * All – Age ³ (AIC: 3120.98) All – Age ⁵ (AIC: 3121.20) All – Age ² (AIC: 3123.96) |

| | Forward Selection | Backward Selection |
|----------------|--|--|
| Step 2 | Age + Age ³ (AIC: 3123.20) * Age + Age ² (AIC: 3125.5) Age + Age ⁴ (AIC: 3126.69) Age + Age ⁵ (AIC: 3132.87) | All – Age ⁴ – Age ⁵ (AIC: 3131.17) All – Age ⁴ – Age ³ (AIC: 3153.06) All – Age ⁴ – Age ² (AIC: 3169.77) No variables to be removed |
| Step 3 | Age + Age ³ + Age ⁵ (AIC: 3124.50) Age + Age ³ + Age ⁴ (AIC: 3124.77) Age + Age ³ + Age ² (AIC: 3125.19) No variable to be included | |
| Selected Model | Age + Age ³ | All – Age ⁴ |

Table 3.2 compares the forward and backward selection methods for the regression of house prices on their ages, the square, the cube, the fourth and fifth power of their ages. When the forward procedure is used, the Age level is the first variable to be selected, as its inclusion to a model with no predictors decreases the AIC from 3202.69 to 3193.36. In the second step, age cubed is selected, as it decreases the AIC from 3193.36 to 3123.20 when added to a model that only contains the age level. Finally, in step 3 no variable is added, as including any further variable will produce an increase rather than a decrease in the AIC. The backward procedure is similar in spirit, but proceeds by elimination rather than by addition of variables. As it emerges from the example above, the two procedures will not necessarily select the same model. In fact, more generally, there is no guarantee that either of the two methods will select the optimal model, as only a subset of the 2^m models (where m is

the number of predictors) is considered. Although in principle it is possible to estimate all the possible models and compare them, this is very impractical when the set of candidate predictors is large. For instance, with 20 features, one would have to estimate $2^{20} = 1,048,576$ models! Therefore, when the set of candidate predictors is large, both forward and backward stepwise regression offer a valid alternative. Both procedures have their pros and cons and the choice between the two depends on the problem at hand. Backward selection tends to be more computationally efficient when the set of candidate predictors is large. However, as the full model is the first to be estimated, the number of predictors is required to be strictly smaller than the number of observations. In contrast, forward selection can also be applied when the number of predictors is larger than the number of observations.

3.2 Classification Problems

In finance, there are many instances where a model's output (dependent variable) is categorical and where the output for each observation can only be one of a small number of categories. In the machine learning jargon, the problem of predicting a qualitative outcome is defined to be a *classification problem* and assigning an observation to one class rather than another is referred to as *classifying* the observation. A specific case of categorical data is where the output is binary – that is, it only has two outcomes. Some examples are:

- Will an individual default on a mortgage?

- Is a transaction fraudulent?
- Does a person have a private pension plan?
- Will an option expire in the money?

In such cases, we would be interested in modeling the probability of one of the outcomes occurring (the probability of a prospective borrower defaulting, the probability of a transaction being fraudulent, etc.). One outcome (referred to as the positive outcome) is assigned a value of one, and the other (referred to as the negative outcome) is assigned a value of zero. A standard linear model would be inappropriate to apply in such cases because there would be nothing in the model's design to ensure that the estimated probabilities lie between zero and one, and we could obtain nonsensical predictions.

3.2.1 Logistic Regression

Instead of a linear regression model, a different specification is used, known as a logistic regression or a *logit model*. This specification uses a cumulative logistic function transformation, resulting in the output being bounded between zero and one. The logistic function $F(y)$ has a sigmoid shape as in [Figure 3.5](#).

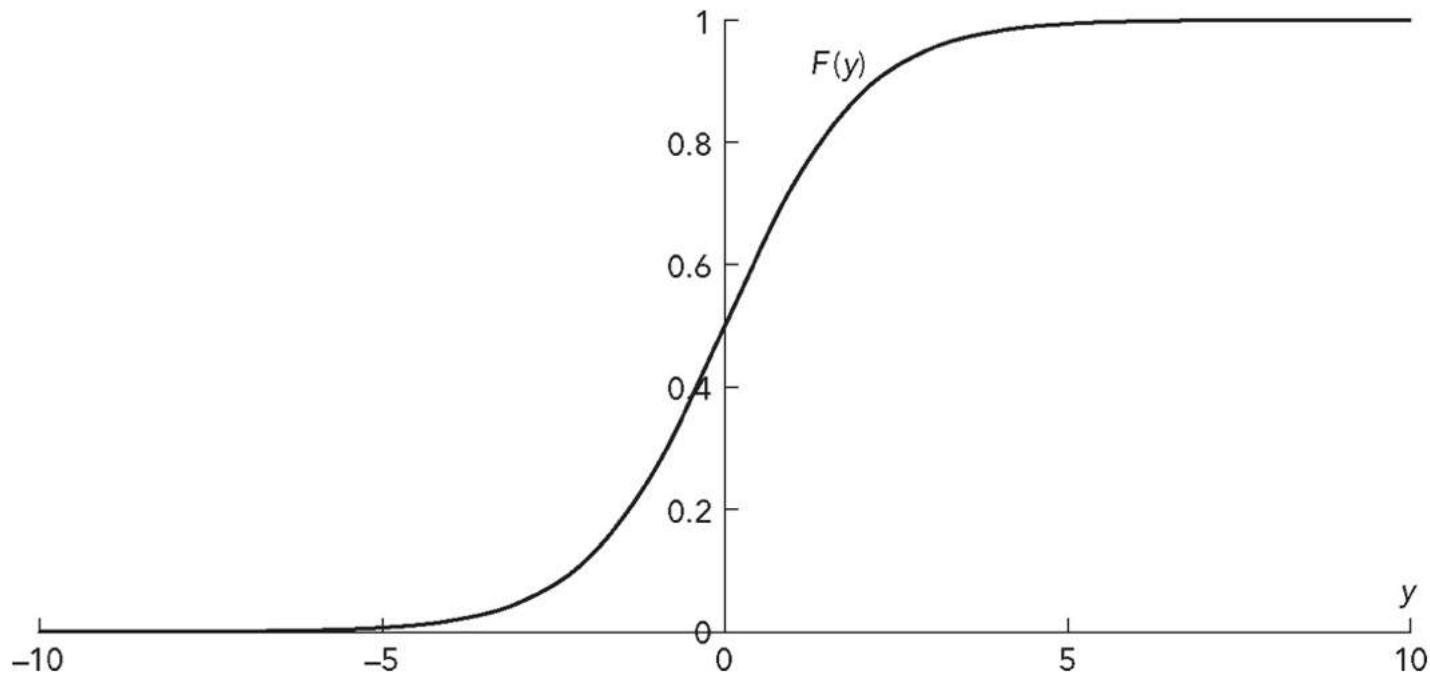


Figure 3.5 Cumulative distribution function for the logistic distribution of a random variable y with a mean of zero
The logistic function is written:³

$$F(y_i) = \frac{1}{1 + e^{-y_i}} \quad (3.9)$$

When there are m features, the functional form y_i is estimated as:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_m x_{mi} + u_i \quad (3.10)$$

The probability that $y_i = 1$ is given by:

$$P_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_m x_{mi} + u_i)}} \quad (3.11)$$

and the probability that $y_i = 0$ is $(1 - P_i)$. An example of a logistic regression model fitted to loan data is given in Box 3.2.

Box 3.2: Logistic Regression Example

To illustrate how logistic regression works, a sample of the data from the LendingClub database is employed.⁴ LendingClub was a peer-to-peer retail lender. The dependent variable is either 0 or 1 for the terminal state of the loan being 0 (fully paid off) or 1 (deemed irrecoverable). Table 3.4 shows the results from only 500 observations.

The parameter estimates from a logit model cannot be interpreted in the usual fashion due to the presence of the logistic transformation, which is nonlinear. Nevertheless, their signs and levels of statistical significance can still be examined.

Borrowers with longer loan terms and those paying higher interest rates have a significantly higher probability of default, whereas those with a mortgage have a significantly lower probability of default. The total sum borrowed, installment,

employment history, whether they own their home, their annual income, and any previous delinquencies or bankruptcies do not significantly affect the probability that borrowers will default on their loans given the other variables.

Table 3.3 Parameter estimates from a logistic regression to model loan outcomes.

| Parameter | Definition | Estimate | Standard error |
|--------------------|--|------------|----------------|
| Bias term | The intercept | -5.3041*** | 1.051 |
| Amount | Total sum borrowed | -0.0001 | 0.000 |
| Term | Length of the loan (months) | 0.0768** | 0.034 |
| Interest rate | APR charged (%) | 0.1147** | 0.045 |
| Installment | Monthly installment | 0.0025 | 0.004 |
| Employment history | Length of borrower's employment history (years) | 0.0428 | 0.059 |
| Homeowner | 1 = owns home; 0 otherwise | 0.1149 | 0.409 |
| Mortgage | 1 = has a mortgage; 0 no mortgage | -0.9410** | 0.435 |
| Income | Annual income (USD) | -0.0001 | 0.000 |
| Delinquent | Number of times borrower has been more than a month behind with payments in the past two years | 0.0985 | 0.113 |

| Parameter | Definition | Estimate | Standard error |
|--------------|--|----------|----------------|
| Bankruptcies | Number of publicly recorded bankruptcies | -0.1825 | 0.361 |

Notes: ** and *** denote significance at the 5% and 1% levels, respectively⁵. The dependent variable is 1 for loans that were charged off (irrecoverable) and zero for paid off loans.

³ The exponential function $\exp(x)$ is defined as e^x , where e is 2.7182..., the base of natural logarithm.

⁴ See <https://www.kaggle.com/datasets/wordsforthewise/lending-club>.

⁵ A $x\%$ significance level indicates that there is only $x\%$ probability that the null hypothesis ($\beta_i = 0$) is rejected when it's true. Here the regression coefficients denoted by * are statistically significant.

3.2.2 Other Types of Limited Dependent Variable Models

Discrete Choice Models

The scenarios examined so far in this section are all related to models where the output variable is binary. There will, however, also be other situations where we wish to model categorical data with more than two categories, which is called a discrete choice model. The classic example is where a commuter chooses between different modes of transport (e.g., car, bicycle, bus), and we are interested in modelling the probability that a particular individual will travel by each mode, which would be the output. Relevant features could include the person's age, level of fitness, distance to travel, start time, income, etc. As for the binary output case discussed in the previous sub-section, it would not be appropriate to use a linear regression model, and instead

we use an extension of the logit regressions described above, known as *multinomial logit* models.

We estimate models for all categories that are present in the data but one, which will serve as baseline category. In the above case, for example, we could model the probability that a commuter will choose to travel by car, and the probability of travelling by bicycle. Then the probability of travelling by bus is simply one minus the sum of the two calculated probabilities.

Examples of problems in finance where a discrete choice framework would be appropriate include:

- Customers choosing between different banks.
- A borrower paying off the loan, defaulting, or extending a loan at maturity.
- A firm choosing which of several competing exchanges to list their shares on.

Ordinal Variables

All the examples presented in the previous subsection related to modelling nominal data – that is, outputs where there is no natural ordering of the categories. A further class of problems relates to categorical data where an output could be drawn from one of several categories but where the categories have an implicit ordering – i.e., *ordinal data*. Examples of relevant problems where the data are ordinal include:

- The credit ratings categories assigned to bonds by ratings agencies (AAA, AA, BBB, etc.).

- The degree of risk aversion of retail investors, which is typically measured using integers on a 1-10 scale.
- Modelling salary information that is only available in bands rather than actual values (e.g., low/medium/high).

For modelling ordinal variables where there are more than two outcomes, *ordered logit* models would be used. The estimation principles are the same as for the binary case, but the values of cutoff parameters between categories must also be estimated. In some circumstances, the values of the output variable in a model that we observe are not sampled randomly from the population, resulting in a biased sample. Appendix 3.A briefly discusses a procedure that can be used when there is selection bias in the sample.

3.3 Linear Discriminant Analysis

The logistic regression discussed above works very well for binary classification problems. On the contrary, when there are multiple, well-separated classes, the estimates from a logistic regression turn out to be very unstable. In this case, an alternative to logistic regression is offered by linear discriminant analysis (LDA).

Linear discriminant analysis assumes that the joint distribution of features is multivariate normal with a common variance-covariance matrix, but with different mean vectors. Like logit, the idea is to assign each instance to the class with the

highest conditional probability. A discriminant function is calculated for each of the classes. It is the probability that a new data point belongs to that class. New data points are classified based on which class has the highest probability. It is possible to show that the *discriminant functions* $\hat{\delta}_j(x)$ are linear functions of x for each class j , for $j = 1, \dots, g$.

$$\hat{\delta}_j(x) = x^T \hat{\Sigma}^{-1} \hat{\mu}_j - \frac{1}{2} \hat{\mu}_j^T \hat{\Sigma}^{-1} \hat{\mu}_j + \log(\hat{\pi}_j)$$

(3.12)

where x is the feature vector, $\hat{\mu}_j$ is the vector containing the means of the predictors of each of the g classes, estimated using the training sample data, $\hat{\Sigma}^{-1}$ is the inverse of the data covariance matrix, $\hat{\pi}_j$ is the prior probability of class j , and T denotes the transpose operator. The prior probabilities could be assumed to be equal for all classes or estimated using the frequency of class j in the training data. A new data point is assigned to the class for which $\hat{\delta}_j(x)$ is the largest. Interestingly, LDA has been proven to work well in practice even when the assumptions are not met. An example of how this method works is presented in [Box 3.3](#), and a more general approach to LDA, known as Fisher Discriminant Analysis, is discussed in Appendix 3.B.

Box 3.3: Linear Discriminant Analysis

As an example, consider a notional sample of ten borrowers to be classified as default (1) or no default (0). [Table 3.5](#) reports the loan balance, income, and a dummy variable for whether they have defaulted. Because LDA, like PCA, requires data scaling, the data provided in the table have already been standardized (see Chapter 1). In this case, there are only two classes (default and non-default).

Table 3.5: Balance, income, and default status for a group of borrowers

| Default | Balance | Income |
|---------|---------|--------|
| 0 | -0.55 | -0.75 |
| 0 | -0.23 | 0.19 |
| 0 | -0.76 | 0.53 |
| 0 | -0.08 | -0.73 |
| 0 | -0.72 | 1.32 |
| 0 | -0.79 | 2.00 |
| 0 | -0.80 | -0.34 |
| 1 | 0.46 | -0.71 |
| 1 | 1.95 | -0.96 |
| 1 | 1.51 | -0.54 |

(The default flag is equal to one if the borrower defaulted, and zero otherwise)

The first stage is to calculate the class means for each predictor from the data by averaging the observations that belong to each class. For example, among non-defaults (default = 0), the average (standardized) balance is -0.56 and the average (standardized) income is 0.32 . We write these averages in column vectors as:

$$\hat{\mu}_0 = \begin{pmatrix} -0.56 \\ 0.32 \end{pmatrix}, \quad \hat{\mu}_1 = \begin{pmatrix} 1.31 \\ -0.74 \end{pmatrix}$$

The covariance matrix can be estimated from the data as⁶:

$$\hat{\Sigma} = \begin{pmatrix} 1 & -0.58 \\ -0.58 & 1 \end{pmatrix}$$

where the variance of both predictors is equal to 1 as the predictors have been standardized. The inverse of this matrix is⁷:

$$\widehat{\Sigma}^{-1} = \begin{pmatrix} 1.52 & 0.88 \\ 0.88 & 1.52 \end{pmatrix}$$

Finally, we can set the prior probabilities⁸ to 0.7 and 0.3, respectively, based on the class frequencies of no default (7/10) and default (3/10) in the training sample. Suppose now that we had to classify an applicant with a standardized balance equal to -1.42 and a standardized income equal to -0.20 . This means that we can write

$$\mathbf{x}^T = (-1.42 \quad -0.20)$$

We could use equation (3.12) to compute the discriminant score of each class. First, for the non-default class we would have:

$$\delta_0 = (-1.42 \quad -0.20) \begin{pmatrix} 1.52 & 0.88 \\ 0.88 & 1.52 \end{pmatrix} \begin{pmatrix} -0.56 \\ 0.32 \end{pmatrix} - 0.5(-0.56 \quad 0.32) \begin{pmatrix} 1.52 & 0.88 \\ 0.88 & 1.52 \end{pmatrix} \begin{pmatrix} -0.56 \\ 0.32 \end{pmatrix} + \log(0.7) = 0.30$$

Then, for the default case, the score is:

$$\delta_1 = (-1.42 \quad -0.20) \begin{pmatrix} 1.52 & 0.88 \\ 0.88 & 1.52 \end{pmatrix} \begin{pmatrix} 1.31 \\ -0.74 \end{pmatrix} - 0.5(1.31 \quad -0.74) \begin{pmatrix} 1.52 & 0.88 \\ 0.88 & 1.52 \end{pmatrix} \begin{pmatrix} 1.31 \\ -0.74 \end{pmatrix} + \log(0.3) = -3.95$$

The score is 0.30 for no default and -3.95 for default. Because the score for the no default class is larger than for the default class, this new applicant is classified as "no default."

$$cov(x_1, x_2) = \frac{\sum_i^N (x_1 - \bar{x}_1)(x_2 - \bar{x}_2)}{N-1}$$

⁶ Covariance between two features x_1 and x_2 is defined as

$$: \begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ is } \frac{1}{(ad-bc)} \begin{pmatrix} d & -c \\ -b & a \end{pmatrix}$$

⁷ The inverse of a 2×2 matrix

⁸ Prior probability is the assumed probability before considering any new information.

Appendix 3.A The Heckman Two-stage Procedure

The values of the output variable in a model that we observe may not be sampled randomly from the population, which leads to a biased sample. For example, willingness to respond to surveys is often correlated with the variables we are trying to measure. So, if we asked bank customers to complete a survey indicating their satisfaction with the level of service they have received, those who are the least happy may be those most likely to complete the survey. If we were then interested in

modelling the factors that affect customer satisfaction levels using these survey results, our parameter estimates would be biased.

Another situation where such an issue would occur is in the context of modelling the value of share repurchases. Most listed firms do not make share repurchases, and therefore the output variable would have some problematic characteristics: no observations would be negative, and the bulk of the observations would have a value of zero, with the remainder having a distribution a long way from zero.

The Heckman approach deals with these situations by separating them into two stages:

1. Model the probability that a bank customer will complete a survey or firm will choose to make share repurchases using a binary logit function.
2. Model the determinants of customer satisfaction among those who have elected to complete the survey or model the size of the repurchases among firms that have chosen to make them.

Appendix 3.B Fisher Discriminant Analysis

There is an alternative and more general approach to Linear Discriminant Analysis (LDA), known as Fisher discriminant analysis. The idea is to find a vector on which to project the data such that the maximum between-group variance of the projection relative to within-group variance is obtained. In this respect, LDA can be related to

dimensionality reduction techniques: we find a projection of the data on a lower dimensional space that is optimal for classification of the data. Once this vector has been found, new data to be classified are projected onto this vector and assigned to the class whose mean they are closer to. In other words, we aim to find a way to separate the data into g distinct classes such that the distance between the means of the different classes is maximized while the variation within each class is minimized. Technically, we find the vector b that maximizes the signal-to-noise ratio, which is given by the ratio of the between- and within-group variances:

$$\frac{b^T \mathbf{B} b}{b^T \mathbf{W} b} \quad (3B.1)$$

Where \mathbf{B} is the between-group covariance matrix and \mathbf{w} is the within-group covariance matrix.

To make it more concrete, consider a simple case where only two classes are available. In this case, solving the maximization problem gives:

$$b = \widehat{\Sigma}^{-1} (\widehat{\mu}_2 - \widehat{\mu}_1) \quad (3B.2)$$

where $\widehat{\Sigma}$, $\widehat{\mu}_2$ and $\widehat{\mu}_1$ are the covariance matrix, and the class means, respectively. The discriminant vector is perpendicular to b . Therefore, the discriminant score of a new data instance x is obtained by projecting x onto the vector b :

$$x^T \hat{\Sigma}^{-1} (\hat{\mu}_2 - \hat{\mu}_1) \quad (3B.3)$$

We classify x as belonging to class 2 if $x^T \hat{\Sigma}^{-1} (\hat{\mu}_2 - \hat{\mu}_1) > c$, where c is a threshold to be decided by the researcher. If we are ready to assume that the two classes display approximately the same distribution, then the optimal threshold is:

$$c = \frac{1}{2} \hat{\mu}_2^T \hat{\Sigma}^{-1} \hat{\mu}_2 - \frac{1}{2} \hat{\mu}_1^T \hat{\Sigma}^{-1} \hat{\mu}_1. \quad (3B.4)$$

In this case, the Fisher formulation of the problem is equivalent to the probabilistic formulation that was presented earlier for LDA with equal prior probabilities.

Appendix 3.C Linear Discriminant Analysis Example

A bank is planning to run a tailored marketing campaign to promote a new savings product among its current clients. The marketing campaign will consist of one-to-one phone calls. To save on the cost of the campaign, the bank wants to know which clients are most likely to subscribe to the product if approached with a marketing campaign. To this end, they have gathered data from a previous campaign, which are reported in the table below:

| Subscribed | Balance (Dollars) | Age |
|------------|-------------------|-----|
| yes | 2343 | 59 |
| yes | 45 | 56 |
| yes | 1270 | 41 |
| yes | 381 | 37 |
| yes | 40 | 35 |
| yes | 22 | 31 |
| no | 390 | 29 |
| no | 6 | 53 |
| no | 71 | 58 |
| no | 0 | 33 |

where "Subscribed" is the outcome of the campaign, "Balance" is the outstanding balance of the client at the bank in US dollars, and "Age" is the age of the client in years. Perform the following tasks:

A. Standardize the features.

B. Estimate the within-group means for the clients who subscribed (class 1) and those who did not subscribe (class 2).

- C. Compute the data covariance matrix.
- D. How would you classify a client who is 51 years old and has an outstanding balance of 10,635 dollars? (hint: use the features' means and standard deviations computed in part a. to standardize the new data first).
- A. To standardize the features, we first must compute their sample mean and standard deviation. The sample mean of the first feature is $\overline{\text{balance}} = 456.8$ and its standard deviation is 769. The sample mean of the second feature is $\overline{\text{age}} = 43.2$ and its standard deviation is 11.99. The standardized features are obtained from subtracting the mean and dividing by the standard deviation. For instance, for age, the first entry will become:

$$\frac{59 - 43.2}{11.99} = 1.32$$

and so on. The table with the standardized features is below.

| Subscribed | Balance | Age |
|-------------------|----------------|------------|
| yes | 2.45 | 1.32 |
| yes | -0.54 | 1.07 |
| yes | 1.06 | -0.18 |

| Subscribed | Balance | Age |
|-------------------|----------------|------------|
| yes | -0.10 | -0.52 |
| yes | -0.54 | -0.68 |
| yes | -0.57 | -1.02 |
| No | -0.09 | -1.18 |
| No | -0.59 | 0.82 |
| No | -0.50 | 1.23 |
| No | -0.59 | -0.85 |

B. The within-group means are simply the means of the features for each class. For class 1, the vector of the features' means would be:

$$\mu_1 = \begin{pmatrix} 0.29 \\ 0.00 \end{pmatrix}$$

where the first entry is obtained
as:

$$\bar{x}_{11} = \frac{2.45 - 0.54 + 1.06 - 0.10 - 0.54 - 0.57}{6} = 0.29.$$

and the second entry is:

$$\bar{x}_{21} = \frac{1.32 + 1.07 - 0.18 - 0.52 - 0.68 - 1.02}{6} = 0.00.$$

For class 2, the vector of the features' means would be:

$$\mu_2 = \begin{pmatrix} -0.44 \\ 0.00 \end{pmatrix}$$

where the first entry is obtained as:

$$\bar{x}_{12} \frac{-0.09 - 0.59 - 0.50 - 0.59}{4} = -0.44.$$

and the second entry is:

$$\bar{x}_{22} \frac{-1.18 + 0.82 + 1.23 - 0.85}{4} = 0.00.$$

c. As the two features have been standardized, their variances are both equal to one. Therefore, we only need to compute the covariance between the two features, which is given by:

$$\bar{x}_{22} \frac{-1.18 + 0.82 + 1.23 - 0.85}{4} = 0.00.$$

Hence, the covariance matrix is:

$$\begin{pmatrix} 1 & 0.33 \\ 0.33 & 1 \end{pmatrix}$$

d. The only further piece of information that we need to determine is the prior probabilities of each class. In the sample, 6/10 of the clients subscribed to the product and 4/10 did not. Therefore, we assign 0.6 as a prior to class 1 and 0.4 to class 2. The inverse of the covariance matrix can be easily obtained by hand or using, for example, the function MINVERSE in Excel. The standardized features for the new data point are:

$$age_{sd} = \frac{51 - 43}{12} = 0.65$$

And,

$$\text{balance}_{sd} = \frac{10,635 - 457}{769} = 13.24$$

Therefore, we get:

$$\delta_1 = (13.24 - 0.65) \begin{pmatrix} 1.12 & -0.37 \\ -0.37 & 1.12 \end{pmatrix} \begin{pmatrix} 0.30 \\ 0.00 \end{pmatrix} - 0.5 (0.30 - 0.00) \begin{pmatrix} 1.12 & -0.37 \\ -0.37 & 1.12 \end{pmatrix} \begin{pmatrix} 0.30 \\ 0.00 \end{pmatrix} +$$

$$+ \log(0.6) = 3.76$$

$$\delta_2 = (13.24 - 0.65) \begin{pmatrix} 1.12 & -0.37 \\ -0.37 & 1.12 \end{pmatrix} \begin{pmatrix} -0.44 \\ 0.00 \end{pmatrix} - 0.5 (-0.44 - 0.02) \begin{pmatrix} 1.12 & -0.37 \\ -0.37 & 1.12 \end{pmatrix} \begin{pmatrix} -0.44 \\ 0.02 \end{pmatrix} +$$

$$+ \log(0.4) = -7.51$$

As the largest discriminant score is that of class 1, the new client is classified as a subscriber. In fact, from the large level of standardized balance 13.24, it is quite clear that the new client should be a subscriber.

Supervised Learning for Numerical Data- Part 1, Econometric Techniques: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

3.1

A. What is an outlier in the context of regression?

An outlier is a data point that demonstrably does not fit with the pattern of the others so that in a regression context, the fitted and actual values would be a long way apart, leading to a residual of larger magnitude than the others.

B. How can outliers be detected?

There are various methods available to detect outliers. A good first step is to examine the residuals from the proposed model to see whether any are significantly larger in absolute value than the others. More formally, a measure known as Cook's distance can be calculated for each point. This evaluates how much each parameter would change if a given data point were excluded from the sample. Large values of Cook's distance indicate a point that would be more likely considered an outlier.

3.2 What is a stepwise regression and how does it work?

Stepwise regression is a technique for feature selection. Beginning with a list of candidate features that could be included in the model, the analyst selects an approach: either beginning with a model containing no features (forwards

selection) or containing all the features (backwards selection). With forwards selection, the analyst adds the feature that would have the most additional explanatory power until a further addition does not decrease the AIC. With backwards selection, all features are initially included in the model, and they are removed one-by-one starting with the feature having the least explanatory power until removing a further variable fails to decrease the AIC.

3.3 Explain why linear regression cannot be used when the dependent variable in a regression model can only take the values 0 or 1.

3.4 Explain the main assumptions underlying LDA.

The standard approach to linear discriminant analysis assumes that the data arise from g multivariate normal distributions (where g is the number of output levels) with different mean vectors but common covariance matrix.

3.5 Suppose that we have the following results for a model estimated using ordinary least squares for the relationship between a firm's return on assets (ROA) and its $SIZE$, measured using market capitalization, for a group of small cap stocks:

$$\widehat{ROA}_i = 2.6 + 4.22SIZE_i - 1.32SIZE_i^2$$

where ROA is measured in percent and $SIZE$ is measured in hundreds of millions of US dollars.

A. According to this model, is the relationship between $SIZE$ and ROA linear?

No, because the equation includes a squared term in $SIZE$, the relationship between $SIZE$ and ROA is non-linear, therefore the relationship between the two variables will depend on the value of $SIZE$.

B. Comparing two firms with market caps of \$100m and \$110m, what would be the difference in their expected ROA ?

There is more than one way to calculate the answer to this question, but arguably the most straightforward is to plug the values 1 and 1.1 into the fitted equation (note that market cap is measured in hundreds of millions of dollars, not in millions of dollars, hence \$100m and \$110m need to be divided by 100). Then,

according to the model, the *ROA* for a firm of *SIZE* \$100m would be 5.56%, and the *ROA* for a firm of *SIZE* \$110m would be 5.70%, which is 0.14% higher.

C. What would be the optimal size of firm for an investor to choose if they wanted a firm with maximal *ROA*, and what *ROA* would this generate? Hint:

$$\frac{\partial \widehat{ROA}_i}{\partial SIZE} = 4.22 - 2.64SIZE_i$$

We need to differentiate the fitted line with respect to *SIZE*:

$$\frac{\partial \widehat{ROA}_i}{\partial SIZE_i} = 4.22 - 2.64SIZE_i$$

Then we would set this derivative to zero, and rearrange for *SIZE* to give the value of *SIZE* that maximizes *ROA*:

$$4.22 - 2.64SIZE_i = 0 \Rightarrow SIZE_i = 1.60$$

Therefore, a firm with a market cap of \$160m would have the highest possible value of *ROA*. To calculate the latter, we simply plug 1.6 into the original fitted equation given in the question:

$$\widehat{ROA}_i = 2.6 + 4.22 \times 1.6 - 1.32 \times 1.6^2 = 6.03\%.$$

6.03% is the highest possible value of *ROA* that a firm of any size could expect to achieve according to the model estimates.

Chapter 4: Supervised Learning – Part 2: Machine Learning Techniques

Learning Objectives

This chapter continues the discussion of models for supervised learning with a focus on machine learning techniques grounded in computer science. It provides an overview of techniques applied in classification and prediction problems, including decision trees, K-nearest neighbors, and support vector machines. An overview of neural networks — a modeling method used in machine learning to replicate how the human brain processes data to perform functions like time-series prediction and natural language processing — is also included. The chapter concludes with the presentation of autoencoders.

After completing this chapter, you should be able to:

- Differentiate between the two types of decision trees and illustrate how each is constructed and interpreted.
- Explain how pruning and ensemble techniques can be used to enhance the performance of decision trees.
- Apply the K-nearest Neighbors method for classification.

- Illustrate how support vector machines are used to classify data.
- Describe how neural networks are constructed and discuss associated challenges.
- Discuss advanced neural network structures.
- Describe how autoencoders are used for dimensionality reduction and differentiate between autoencoders and PCA.

This chapter continues the presentation of models for supervised learning, now focusing specifically on machine learning techniques that originated predominantly in computer science rather than econometrics. We begin with a discussion of decision trees, a highly interpretable technique used primarily for classification applications. The chapter then moves on to consider K nearest neighbors, another popular, intuitive, and straightforward technique for both classification and prediction. Support vector machines, which are used for a wide variety of classification problems, are considered next. We then examine neural network models, which are a broad class of models motivated by how the human brain performs computations. Neural networks are, perhaps, the most important machine learning techniques and they have been applied in a range of financial domains including time-series prediction and natural language processing. The chapter concludes with the presentation of autoencoders

4.1 Decision Trees

A decision tree is a supervised machine-learning technique that examines input features sequentially and is so-called because, pictorially, the model can be represented as a tree. At each node is a question, which branches an observation to another *node* or a *leaf* (a terminal node). Although decision trees are particularly popular for classification problems, they can also be employed to estimate the value of a continuous variable and so are sometimes known as classification and regression trees (CARTs). CARTs are popular due to their interpretability, and for this reason, they are sometimes known as “white-box models,” in contrast to other techniques such as neural networks where the fitted model is very difficult to interpret. However, they typically perform less well than “black-box” techniques including neural networks in terms of predictive accuracy. To improve their performance, trees are often combined using ensemble techniques such as *random forests*, *bagging* and *boosting*, as we shall see later in this chapter.

Figure 4.1 shows an illustrative simple decision tree that has already been constructed for assessing the creditworthiness of borrowers. The internal nodes represent attribute-value tests, the edges indicate how to proceed in case of different results from the tests and the circles are the leaves of the tree that contain the labels (or the prediction, in the case of regression trees). In the example below, featuring a classification tree, there are only two attributes – credit score and income, and four labels – default probability equal to two, three, four, or five percent. An instance to be classified is first subject to the test at the topmost node (the *root*), which in this case is whether the credit score is larger than 700. The result of this test decides the route that we use to proceed down the tree: if the score is smaller than 700, we test

whether the income is larger than \$80,000 per year; otherwise, we test whether the income is larger than \$120,000 per year. In the example, the label is then assigned based on the results of this second test as the prevailing class that reaches each leaf.

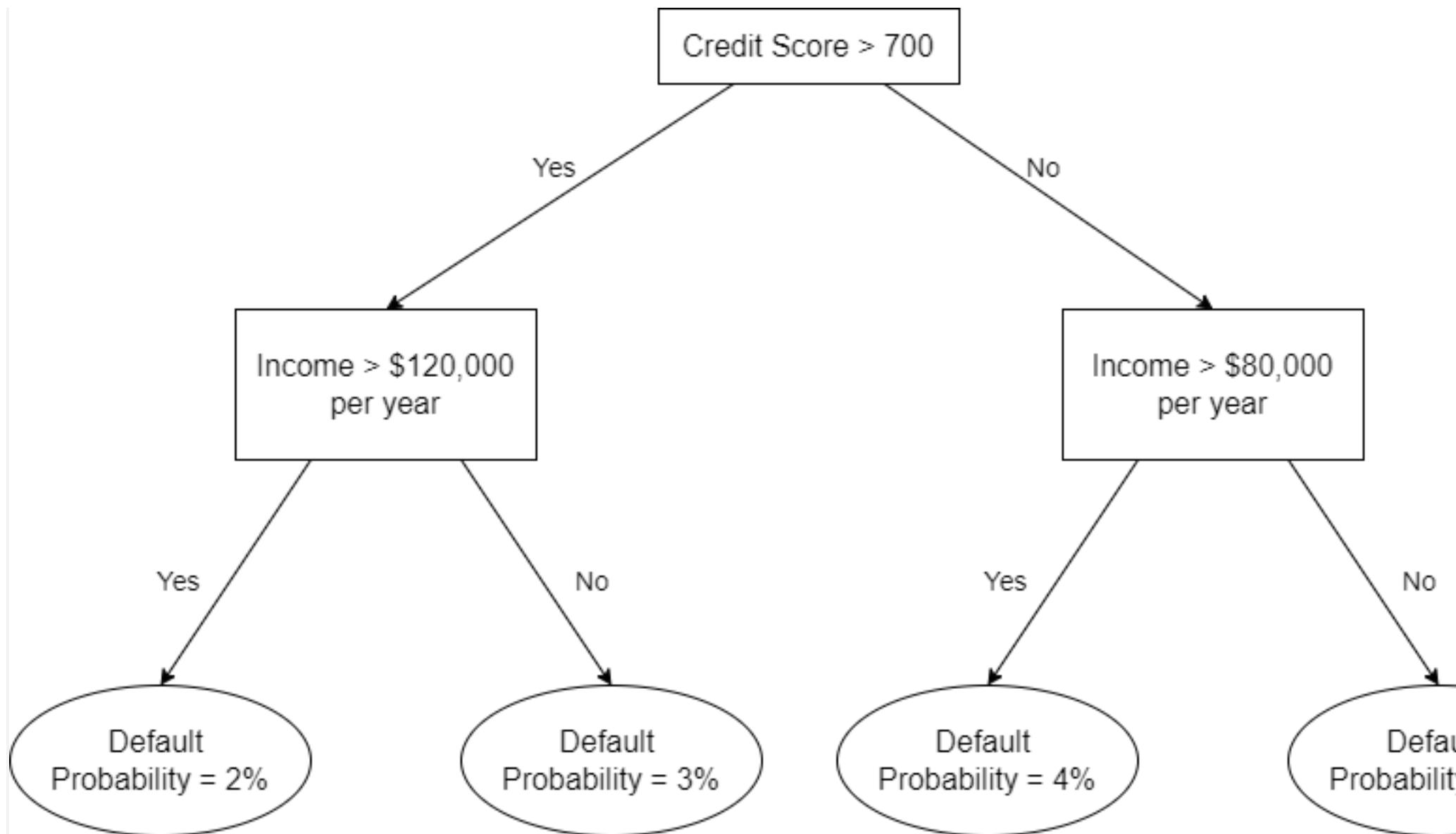


Figure 4.1 Illustration of a simple decision tree for assessing creditworthiness.

4.1.1 Regression Trees

As discussed above, decision trees can be applied both when the target is a continuous variable and when it is a categorical (qualitative) one. In the first case, we call them regression trees. The goal is to split the feature space into regions such that we minimize the residual sum of squares (RSS) given by:

$$\text{RSS} = \sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \quad (4.1)$$

where y_i is an observation in the training set, \hat{y}_{R_j} is the average outcome of the observations in region j , and J is the total number of regions.

Unfortunately, it is computationally infeasible to check all possible partitions of the feature space to find the one that minimizes the quantity above. Therefore, we employ a top-down recursive binary splitting approach. In this approach, we start with all the observations in one region and search for the split that produces the maximum reduction of the RSS; then, for each of the two regions obtained in this way, we look for a further best split, and we proceed recursively until a given stopping criterion is reached.

For instance, suppose that we wish to predict the house price per unit area (e.g., in thousands of dollars per square foot) using two features: the age of the house and the distance to the closest metro station. We have collected a training sample of 414

observations¹, depicted in [Figure 4.2](#). A regression tree is a set of rules that tells us how we can optimally segment the training sample into regions of the feature space. If we estimate a tree for this dataset, we find that the first step is to split the sample between the houses that are less than 826.83 meters from the closest metro station, as depicted in [Figure 4.2](#). Next, we further split the subsample of houses that are less than 826.83 meters from the closest metro station between those that are less than 11.7 years old and those that are more than 11.7 years old. Region 1 (R1) containing the houses that are less than 826.83 meters from the closest metro station and less than 11.7 years old represents a first leaf of the tree. The prediction of the price per unit area for the houses in R1 is 52.25, which is the average price of houses belonging to that region of the feature space. We continue splitting the training sample into non-overlapping regions until further splits fail to improve the prediction accuracy or some stopping criterion is reached. The full tree is depicted in [Figure 4.3](#).

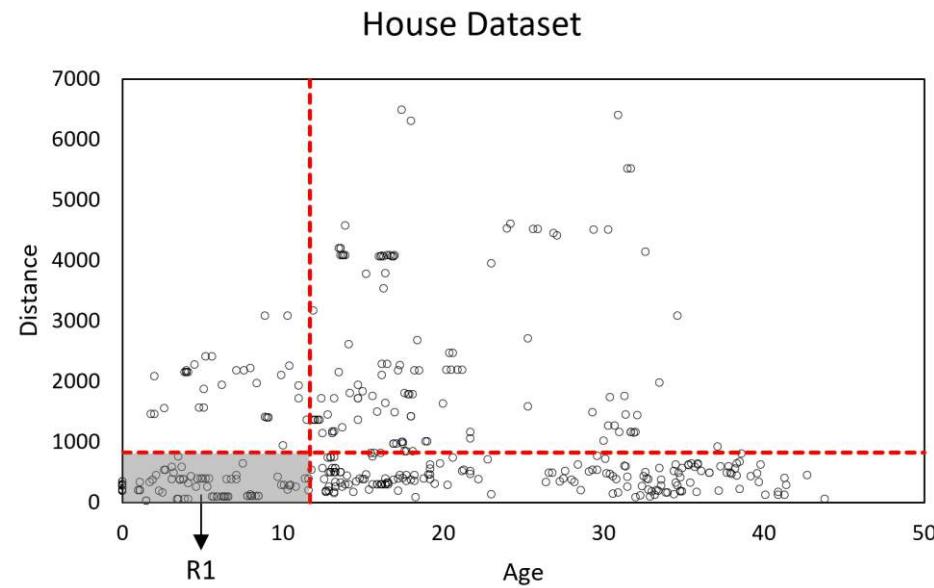


Figure 4.2 Distance from the closest metro station (vertical axis) and age (horizontal axis) for a sample of 414 houses.

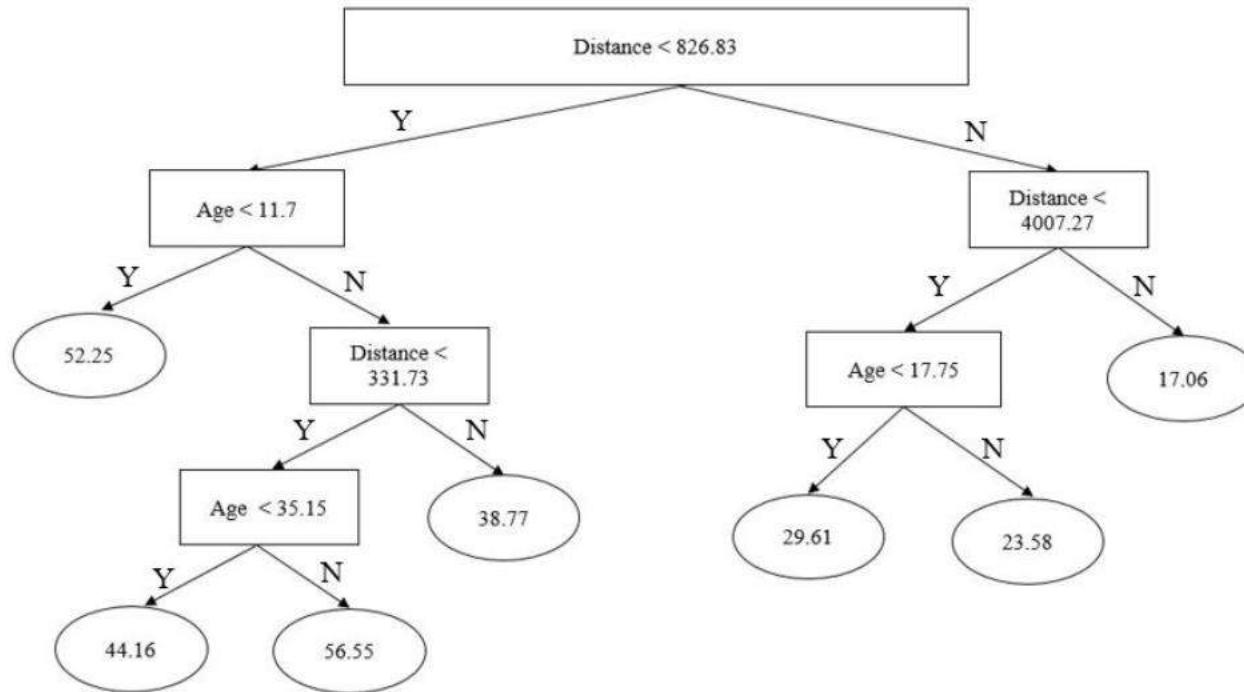


Figure 4.3 Completed regression tree for the prediction of house price per unit of area. The squares denote decision nodes and the circles denote leaf nodes.

¹ Yeh, I-Cheng. (2018), *Real Estate Valuation*, UCI Machine Learning Repository, <https://doi.org/10.24432/C5J30W>.

4.1.2 Classification Trees

Classification trees follow a very similar logic to regression trees except that the outcome variable is categorical in nature. The objective is to split the data into groups that are as *pure* as possible (that is, they contain the largest proportion of one class as possible). However, in classification problems, the RSS cannot be used as a criterion to determine the splits and we need to find measures of purity or, alternatively impurity in classification. Two measures are generally considered: *entropy* and the *Gini coefficient*.

Entropy is a measure of disorder in a system². It is defined as:

$$\text{Entropy} = - \sum_{j=1}^J p_j \log_2(p_j) \quad (4.2)$$

where J is the total number of possible outcomes and p_j is the probability of the j^{th} outcome for $j = 1, \dots, J$. Note that the formula includes the logarithm to base two rather than the more common natural logarithm.³ The Gini coefficient is a measure of the impurity of a node and can be calculated as:

$$Gini = 1 - \sum_{j=1}^J p_j^2 \quad (4.3)$$

A small value of the Gini index indicates that a node mostly contains instances from the same class⁴. Gini and entropy usually lead to very similar decision trees. We will use Gini in the example that follows.

² In information theory, entropy is a measure of the average information content of a message.

³ Changing the base of the logarithm multiples all entropy measures by the same constant and does not affect the results. For base 2 logarithms, entropy can be interpreted as the expected number of bits needed to store the discrete random variable with probability mass function p_1, \dots, p_J .

⁴ That corresponds to the case of $J = 1$ and $p_1 = 1$. Thus, the entropy is zero.

4.1.3 Classification Decision Tree Example

Suppose that a risk manager at an equity income fund is concerned that firms held within the portfolio will stop paying dividends next year and so wishes to build a model to predict whether a firm i will pay ($y_i = 1$) or will not pay ($y_i = 0$) a dividend. A model to classify this output is based on the following variables: whether the firm's earnings have dropped in the previous year, whether it is a large-cap stock, the percentage of retail investors constituting its shareholder base, and whether it is in the tech sector. The features listed in [Table 4.1](#) concern a notional sample of 20 firms that did and did not pay dividends in the past year. For the output variable and the binary feature variables, the value being equal to one indicates that: the firm did pay a dividend, its earnings did drop, it is a large-cap stock, and it is in the tech sector. On the other hand, a value of zero indicates that the firm did not pay a dividend last year, its earnings did not drop, it is not a large-cap stock, or it is not in the tech sector. Therefore, examining the first row of [Table 4.1](#), this firm paid a dividend, its earnings did not drop, it is a large-capitalization stock, 40% of its shareholders are retail investors, and it is in the technology sector.

The percentage of retail investors is a continuous variable that could take any real value from zero to 100. In a decision tree, we need to select a threshold value for each continuous variable that maximizes the information gain at a particular node. Note that the optimal threshold will vary depending on the node at which the split occurs.

Ideally, a particular question will provide a perfect split between categories – i.e., each terminal node will be a pure set. For instance, if it had been the case that no technology stocks paid a dividend, this would be highly beneficial information and the node containing technology stocks would be pure (that is, it will only contain non dividend paying stocks). On the other hand, the worst possible scenario would be where exactly half of the technology stocks paid a dividend, and the other half did not, in which case having information only on whether a company was a tech stock or not would be much less useful.

Table 4.1 Data for dividend payment decision tree example

| Data point | Dividend | Earnings_drop | Large_cap | Retail_investor | Tech |
|------------|----------|---------------|-----------|-----------------|------|
| 1 | 1 | 0 | 1 | 40 | 1 |
| 2 | 1 | 1 | 1 | 30 | 0 |
| 3 | 1 | 1 | 1 | 20 | 0 |
| 4 | 0 | 0 | 0 | 80 | 1 |
| 5 | 1 | 0 | 1 | 20 | 0 |
| 6 | 0 | 1 | 0 | 30 | 1 |
| 7 | 0 | 1 | 0 | 40 | 0 |
| 8 | 1 | 0 | 1 | 60 | 0 |

| Data point | Dividend | Earnings_drop | Large_cap | Retail_investor | Tech |
|------------|----------|---------------|-----------|-----------------|------|
| 9 | 1 | 1 | 1 | 20 | 1 |
| 10 | 0 | 1 | 1 | 40 | 0 |
| 11 | 0 | 0 | 0 | 20 | 1 |
| 12 | 0 | 0 | 1 | 70 | 0 |
| 13 | 1 | 1 | 0 | 30 | 1 |
| 14 | 1 | 0 | 1 | 70 | 0 |
| 15 | 1 | 0 | 1 | 50 | 1 |
| 16 | 1 | 0 | 1 | 60 | 1 |
| 17 | 1 | 1 | 1 | 30 | 0 |
| 18 | 0 | 1 | 0 | 30 | 1 |
| 19 | 0 | 0 | 0 | 40 | 0 |
| 20 | 1 | 1 | 1 | 50 | 0 |

Looking at the output variable (Dividend), 12 out of 20 firms in the sample paid a dividend (and therefore 8 out of 20 did not). Although it is possible to construct the tree using entropy, in this example we will use the Gini coefficient as the calculations

are slightly simpler. We measure the Gini coefficient before we know anything about the features as:

This provides a base level with which we can compare the fall in the Gini coefficient as the tree grows. A fall in the Gini coefficient (or entropy) represents an information gain. The first stage is to select the feature (from among Earnings_drop, Large_cap, Retail_investor, Tech) that will go at the root node. This choice is made by selecting the one that would cause the Gini coefficient to fall the most, which is Large_cap. The calculations underpinning this choice are as follows.

First, examining the earnings drop variable, among firms with an earnings drop (= 1), six paid dividends and four did not. This means the Gini coefficient for firms with an earnings drop is:

Similarly, among firms with no earnings drop, six paid dividends and four did not. Hence, again, the Gini coefficient is 0.480. We next calculate the average Gini coefficient for splitting according to this feature, which is calculated from weighting the coefficients for firms with an earnings drop and for those without an earnings drop according to the proportion of firms in each of those two categories, which is:

We can calculate the information gain from this feature as the base Gini (0.480) minus the entropy from splitting according to this feature (0.480):

In this case, there is no information gain from splitting the sample according to whether the firm experienced an earnings drop. That makes sense because this feature provides no useful information for classifying firms as to whether they will pay a dividend, because the proportion of firms paying a dividend is identical (6/10) both for firms that experienced an earnings drop and those that did not.

This process to calculate the information gain is repeated for the other three features. In the case of whether the firm is a large capitalization stock, of the 13 firms that are, 11 paid dividends and two did not. The Gini calculation is therefore:

Among firms that are not large capitalization (for which the value of this feature is 0), one paid a dividend and six did not, leading to the following Gini coefficient:

This leads to a weighted Gini coefficient for the Large_cap feature of:

and an information gain of:

Repeating the above steps for the Tech dummy variable leads to a weighted Gini coefficient of 0.477 and an information gain of 0.003⁵.

When retail investors are considered, because the variable is continuous, it is necessary to use an iterative procedure to determine the threshold that maximizes the information gain. It turns out that the information gain is less than 0.225 for all possible values of the threshold. Overall, the information gain is therefore maximized when the Large_cap feature is used as the root node. Once this is done, the tree branches out separately for the large-cap firms (13 firms) and for those that are not (7 firms).

At subsequent nodes, features are chosen in a similar way to maximize information gain. Note that the tree does not need to be symmetrical. For example, a different feature can be selected next for the branch comprising small-cap firms compared with the large-cap ones. A possible final tree is shown in [Figure 4.4](#). For large-cap stocks the most important next feature is the proportion of retail investors. It turns out that a threshold of 35% (or equivalently any threshold between 30% and 40%) is optimal for this feature at this node of the tree.

The tree is completed when either a leaf is reached that is a pure set or all the features have already been used so that the data cannot be split further. Creating a perfect classification is impossible in the dividend payment example, so although some branches end in a pure set, others do not. Alternatively, to reduce the size of the tree, stopping criteria can also be used.

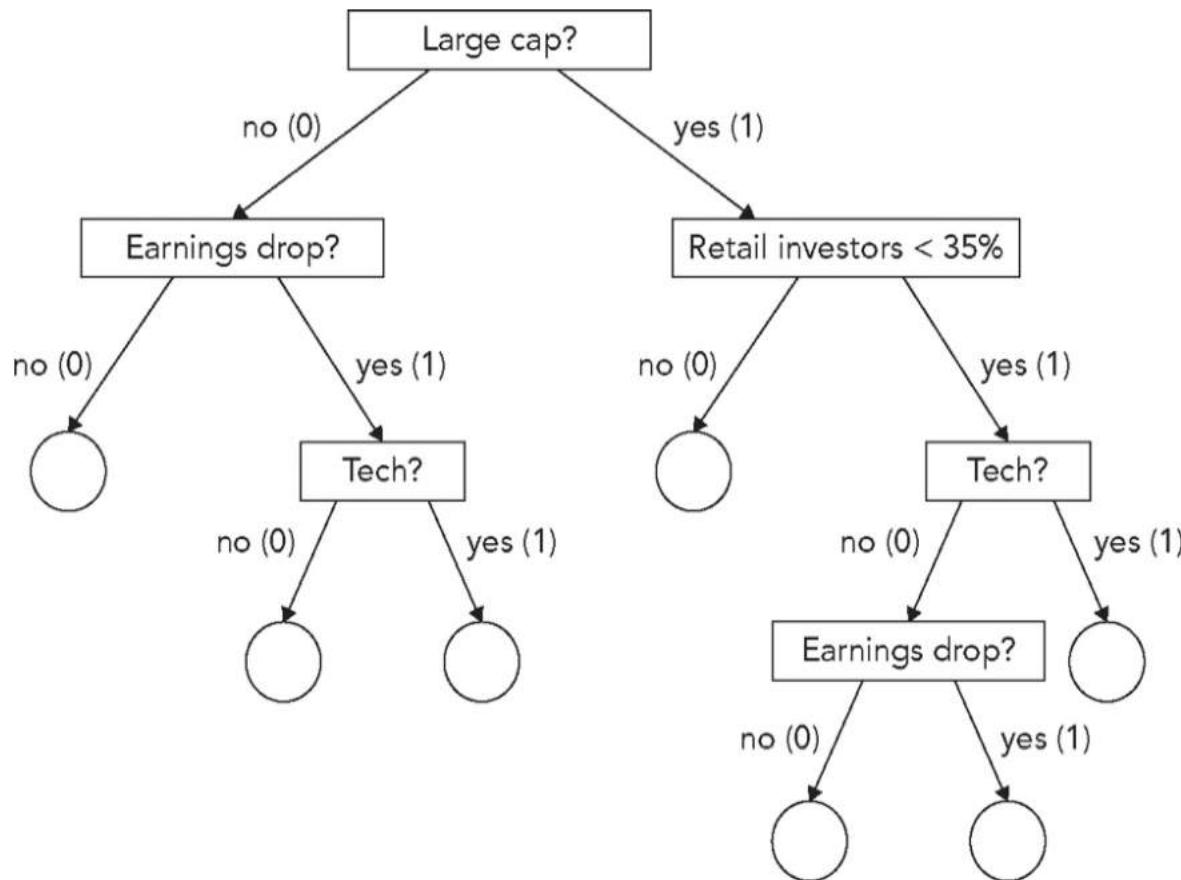


Figure 4.4 Completed decision tree for dividend payment classification. The squares denote decision nodes and the circles denote leaf nodes.

⁵ The base entropy is $-0.6(\log_2 0.6) - 0.4(\log_2 0.4)$, or 0.29, which is the same for the Earnings Drop variable. The Large Cap variable's entropy gain is 0.11, and the Tech feature's entropy gain is 0.

4.1.4 Pruning

Small trees offer several advantages over large trees: interpretability, fewer irrelevant features and, especially, avoidance of overfitting. As well as employing a separate testing sub-sample, overfitting can be prevented by using stopping rules specified *a priori* (pre- or online pruning) or by pruning the tree after it has been grown (post pruning). An example of stopping rules is when a certain number of branches is reached, no further splitting is allowed. Another example is the termination of the splitting of a node if the number of observations under that node is smaller than a certain number.

Whereas pre-pruning prevents a tree from growing too much, post pruning consists of growing the tree fully and then identifying ‘weak links’ *ex post*. In other words, it consists of replacing some subtrees with leaves whose label is the class of most of the instances that reach the subtree in the original classifier (model). There are several pruning algorithms that can be distinguished between top-down and bottom-up approaches depending on whether they start at the leaves or at the root of the tree.

One of the simplest forms of pruning is *reduced error pruning*, which is a bottom-up approach. Starting at the bottom, this algorithm replaces a node with its most popular class any time that the resulting pruned tree does not perform worse than the original tree in the validation sample. Another method generally used to prune regression trees is *cost complexity pruning*. It consists of adding a penalty term to the RSS such that a trade-off between accuracy over the training sample and the number of terminal nodes is established. The extent of the trade-off is determined by a tuning parameter α , which is chosen with cross-validation (see the discussion in Chapter 8).

4.1.5 Ensemble Techniques

As discussed above, trees tend to deliver a worse performance compared to other classifiers. An effective way to improve their accuracy is to construct ensembles of trees, which means to use a range of techniques to combine several trees in a single meta-model. This has two objectives: first, through the “wisdom of crowds” and a result somewhat like the law of large numbers, model fit can be improved by making many predictions and averaging them; second, the techniques can build in protection against overfitting. Although ensembles could involve

combining any types of machine-learning models, the approach is straightforwardly explained using decision trees as an illustration. Three ensemble techniques are briefly discussed here: *bootstrap aggregation*, *random forests*, and *boosting*.

Bootstrap Aggregation

As the name suggests, bootstrap aggregation, or bagging as it is sometimes called, involves bootstrapping⁶ (sampling with replacement) from among the training sample to create multiple decision trees. The resulting predictions or classifications are aggregated to construct a new prediction or classification. A basic bagging algorithm for a decision tree involves the following steps:

1. Sample a subset of the complete training set. For example, if the training set consists of 100,000 observations, sample 10,000.
2. Construct a decision tree in the usual fashion.
3. Repeat steps 1 and 2 many times, sampling with replacement, so that an observation in one subsample can also be in another subsample.
4. If the problem is a regression, average across the forecasts of the several trees to obtain the final prediction. If the problem is a classification, record the class predicted by each of the trees and take a *majority vote*: the class predicted by most of the trees is the overall prediction.

Because the data are sampled with replacement, some observations will not appear at all. The observations that were not selected (called *out-of-bag* data) will not have been used for estimation in that replication and can be used to evaluate model performance.

Pasting is an approach identical to bagging except that sampling takes place without replacement (so that each datapoint can only be drawn at most once in any replication). In pasting with 100,000 items in the training set and sub-samples of 10,000, there would be a total of 10 sub-samples.

Random Forests

Aggregating several forecasts works particularly well when the different learners exhibit low correlation. Random forests provide an improvement over bagging by reducing correlation across the trees. To achieve this, each time

that a tree is split, only a random subset of all the features is considered⁷. Although it may appear counterintuitive to purposefully exclude some features as each tree taken alone may result in a suboptimal result, it is a very strong predictor. The logic is that, if, for instance, a feature is a very strong predictor whereas the rest only have modest predictive power, all the resulting trees will have this feature at the top and they are likely to yield very similar forecasts. In contrast, by forcing some trees to deliberately ignore this strong predictor, the other features are given a chance, and the resulting forecasts are less correlated.

Boosting

Like bagging, boosting entails combining the forecasts from many decision trees. However, while in bagging each tree is grown independently from the others, in boosting each tree is grown while exploiting the information from the prediction errors of the previously grown trees. The two main varieties of boosting are *gradient boosting* and *adaptive boosting* (so-called *AdaBoost*).

Gradient boosting constructs a new model on the residuals of the previous one, which then become the target—in other words, the labels in the training set are replaced with the residuals from the previous iteration. AdaBoost involves training a model with equal weights on all observations and then sequentially increasing the weight on misclassified outputs to incentivize the classifier to focus more on those cases.

⁶ Bootstrapping is the technique of sampling data from the same data set and replacing the drawn samples in the data set before sampling again. In this process, a particular data point can be sampled multiple times.

⁷ The number of features chosen is usually approximately equal to the square root of the total number of features available.

4.2 K Nearest Neighbors

K nearest neighbors (KNN) is a simple, intuitive, supervised machine-learning model that can be used for either classification or predicting the value of a target variable. To predict the outcome (or the class) for an observation not in the training set, we search for the K observations in the training set that are closest to it using one of the distance measures introduced in Chapter 2. Our prediction is the mean of the nearest neighbors' outcomes. If the problem is a classification one, the instance to be classified is assigned to the class to which most of the nearest neighbors belong (an approach that is known as majority voting).

KNN is sometimes termed a *lazy learner* because it does not learn the relationships in the dataset in the way that other approaches do. In other words, it does not actually build a model; instead, every time KNN encounters a new instance, it compares it to all the existing instances to make a prediction.

The steps involved in a typical KNN implementation are as follows:

1. Select a value of K and a distance measure, usually either the Euclidean or Manhattan measure.
2. Among the points in the training sample, identify the K points in feature space that are closest to the point in feature space for which a prediction is to be made according to the chosen measure.
3. a) If it is a prediction problem, compute the mean of the outcomes for the K neighbors that have been identified, or
 b) If it is a classification problem, assign the instance to the class to which most of the nearest neighbors belong.

To understand how the method works, let us consider again the notional sample of borrowers that we had introduced in Chapter 3. **Table 4.2** reports the two features, balance and income, which have been already standardized, and whether the borrower has defaulted (default equal to one) or not (default equal to zero). Note that, like the k-means algorithm, it is crucial that the features are standardized as the distances among the data points depend dramatically on the features' scales. If the data is not scaled, large differences in the features' scales would give more importance to features with the largest scale when the distances are computed.

Table 4.2 Balance, income, and a dummy variable equal to one if they default for a notional sample of borrowers

| N | Default | Balance | Income | Euclidean Distance |
|---|---------|---------|--------|--------------------|
| 1 | 0 | -0.55 | -0.75 | 1.46 |
| 2 | 0 | -0.23 | 0.19 | 1.38 |
| 3 | 0 | -0.76 | 0.53 | 2.01 |
| 4 | 0 | -0.08 | -0.73 | 0.99 |
| 5 | 0 | -0.72 | 1.32 | 2.52 |

| N | Default | Balance | Income | Euclidean Distance |
|----------|----------------|----------------|---------------|---------------------------|
| 6 | 0 | -0.79 | 2 | 3.11 |
| 7 | 0 | -0.8 | -0.34 | 1.72 |
| 8 | 1 | 0.46 | -0.71 | 0.45 |
| 9 | 1 | 1.95 | -0.96 | 1.11 |
| 10 | 1 | 1.51 | -0.54 | 0.61 |

Assume that we must classify a new instance, which is a borrower with a standardized balance of 0.90 and a standardized income of -0.61. Suppose also that we select K equal to four and we use the Euclidean distance to find the nearest neighbors. Henceforth, we shall compute the Euclidean distance between the new, unclassified instance and each of the instances in the training sample using the formula introduced in Chapter 2. For instance, the distance between the instance to be classified and the first instance in the training sample would be:

$$d_E = \sqrt{(-0.55 - 0.90)^2 + (-0.75 + 0.61)^2} = 1.46$$

The last column in the table reports the Euclidean distance in the feature space between the instance to be classified and those in the training sample. It is easy to see that the four closest instances are observations 8, 10, 4, and 9. Out of those, three of them are defaulted borrowers. Therefore, the new instance is classified as likely to default.

A crucial choice in the context of KNN is, of course, the value of K . Small values of K tend to overfit the data whereas large values of K may underfit. A common choice is to set K approximately equal to the square root of N , the total size of the training sample. So, if $N = 5,000$ points, then set $K = 71$. Another approach is to use cross-validation to tune K and choose the value that minimizes the error over the validation sample (see the discussion in Chapter 8).

KNN is simple and yet tends to yield quite accurate forecasts. However, its main disadvantage is that it is computationally intensive as the distances between one instance and all the others must be computed before KNN

can identify the nearest neighbors. Another drawback of this method is that it performs poorly when there are a few irrelevant or noisy features, as those can drive similar instances apart in the feature space.

4.3 Support Vector Machines

Support vector machines (SVMs) are a class of supervised machine-learning models that are particularly well suited to classification problems when there are large numbers of features. To understand how they work, we will start with a sample of linearly separable data points that belong to one of two classes (labelled as -1 or $+1$).⁸ If there were only two features, the data could be easily represented in the cartesian plane, such as in [Figure 4.5](#).

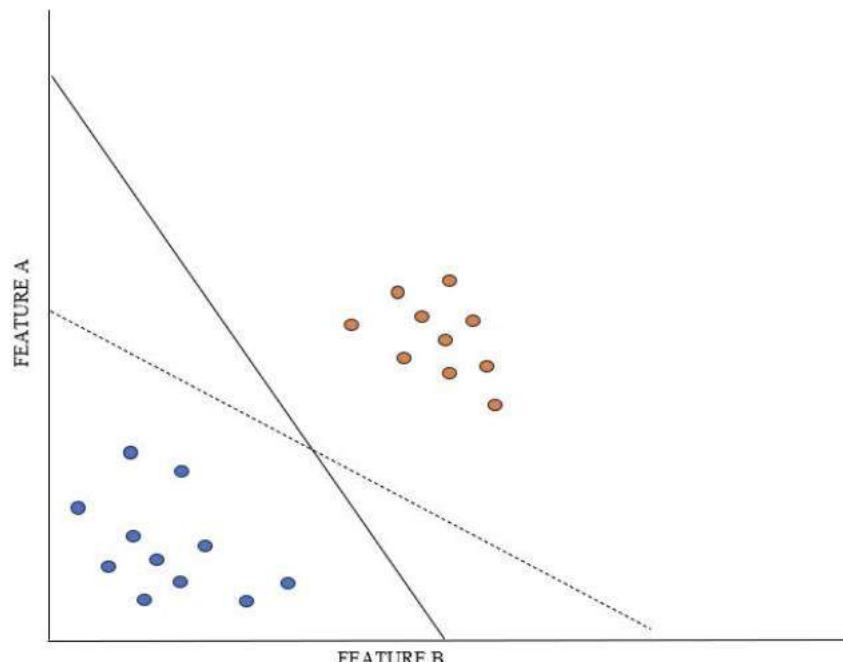


Figure 4.5 An example of linearly separable data on a cartesian plane.

Our objective is to identify the position of a line that would best separate the two groups (technically, the *classification boundary*), enabling us to predict for an additional data point not in this sample whether the outcome should be -1 or $+1$. The blue and orange points can be perfectly separated using either the dotted or the solid line. More generally, there is an infinite number of linear boundaries that perfectly classify the data. Therefore, we need a metric that helps us to identify which of the boundaries is the most appropriate. SVM uses a metric called *margin*. Broadly, the margin is the sum of the distances between the classification boundary and the closest instance in the training data for each of the two classes. Given a classification boundary, it is possible to construct two lines that are parallel to it and that touch the training data of opposite classes on either side, and that have no data points between them. This is represented in [**Figure 4.6**](#). The training data points on these two lines are called *support vectors* and the distance between the two lines is the margin; the two lines are sometimes referred to as *margin constraints*. The optimal classification boundary, also known as the *maximum margin classifier*, is such that it is equidistant from each support vector and the margin is maximized. Again, this is illustrated in [**Figure 4.6**](#).

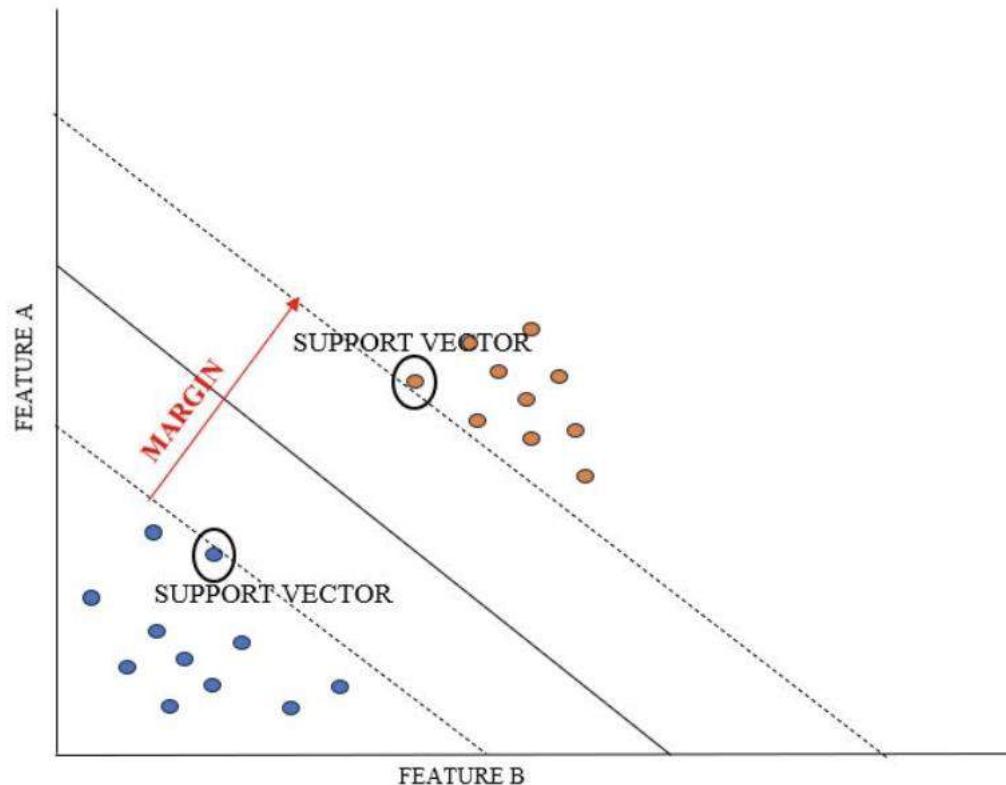


Figure 4.6 A graphical illustration of SVMs.

The maximum margin classifier creates a decision value $D(x)$ that classifies a new instance such that if $D(x) > 0$ we would predict the instance to belong to “class 1” (coded +1); otherwise, we would predict the instance to belong to “class 2” (coded -1). Now that we have the intuition of how the optimal classification boundary is found, we can discuss how the optimization problem is solved in practice – all the details are presented in Appendix 4.A.

⁸ Two sets of data points are linearly separable, if there exists at least one line or hyperplane that can separate them.

4.3.1 SVM Example

The concepts above are best illustrated with an example. A retail bank has previously made car loan decisions manually on a case-by-case basis but is interested in developing a machine-learning model that would replicate the process. It has information on the customers' incomes, savings, and whether the loans were granted for a balanced sample of 12 notional prior applicants as in [Table 4.3](#).

Table 4.3 Data for SVM car loan example

| Applicant number | Monthly income (USD 000s) | Total savings (USD 000s) | Loan granted? (yes = +1; no = -1) |
|------------------|---------------------------|--------------------------|-----------------------------------|
| 1 | 2.5 | 5.0 | -1 |
| 2 | 1.8 | 0.5 | -1 |
| 3 | 4.1 | 1.6 | -1 |
| 4 | 0.8 | 2.0 | -1 |
| 5 | 6.2 | 4.0 | -1 |
| 6 | 3.8 | 6.2 | -1 |
| 7 | 2.1 | 9.0 | +1 |
| 8 | 4.6 | 10.0 | +1 |
| 9 | 1.8 | 13.0 | +1 |
| 10 | 5.2 | 8.0 | +1 |
| 11 | 10.5 | 3.0 | +1 |

| Applicant number | Monthly income (USD 000s) | Total savings (USD 000s) | Loan granted? (yes = +1; no = -1) |
|------------------|---------------------------|--------------------------|-----------------------------------|
| 12 | 7.4 | 8.5 | +1 |

Loans granted are coded as +1 and loans not granted coded as -1. The solution of the optimization problem above (which is defined in Appendix 4.A) leads to estimation of $w_0 = -12.24$, $w_1 = 0.90$, and $w_2 = 1.26$. Therefore, the maximum margin classifier is:

$$-12.24 + 0.90 \text{ (Monthly income)} + 1.26 \text{ (Total savings)} = 0$$

and the margin constraints are:

$$-12.24 + 0.90 \text{ (Monthly income)} + 1.26 \text{ (Total savings)} = +1$$

$$-12.24 + 0.90 \text{ (Monthly income)} + 1.26 \text{ (Total savings)} = -1$$

The fitted decision boundary, the support vectors, and the data points are plotted in [Figure 4.7](#). If we had to classify a new borrower with monthly income equal to 5.7 and total savings equal to 3.5 USD thousands, we would obtain:

$$D(x) = -12.24 + 0.90(5.7) + 1.26(3.5) = -2.7.$$

As this value is below zero, we would classify the loan as not granted (i.e., we would label it "-1").

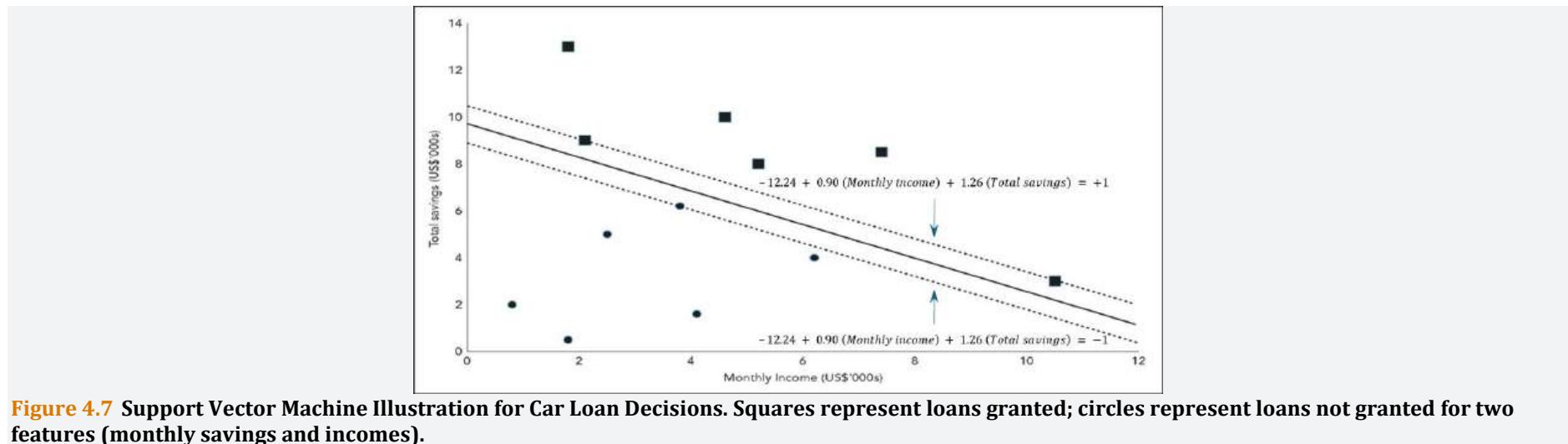


Figure 4.7 Support Vector Machine Illustration for Car Loan Decisions. Squares represent loans granted; circles represent loans not granted for two features (monthly savings and incomes).

4.3.2 SVM Extensions

Although the preceding illustration was simplified by having only two features, the principles and the optimization framework would be the same for any number of features. But instead of identifying a line in the center with the biggest margin, with more features it would be a hyperplane having several dimensions one less than the number of features.

The two groups of points (loans granted and loans not granted) had a clear degree of separation in the example above, but a more likely practical scenario would be that there is some overlap between the two. For instance, some borrowers with low wealth might have low incomes with very stable jobs who are unlikely to default and therefore should be granted a loan, and *vice versa*. In such cases, where the data are not *linearly separable*, the approach described above could not be used and requires some modification. A more flexible method would be to use *soft margins*, which introduces a penalty term into the optimization for incorrect classifications. An equation for the modified optimization problem is given in Appendix 4.A. Further extensions are available to allow for the path being nonlinear.

4.4 Neural Networks

Artificial neural networks (ANNs) are a class of machine-learning approaches loosely modeled on how the brain performs computation. By far the most common type of ANN is a feedforward network with backpropagation, sometimes known as a *multi-layer perceptron*. The basic unit of a multi-layer perceptron is the *neuron*, a unit that holds information. The neurons are arranged in layers and a multi-layer perceptron consists of several layers of neurons. [Figure 4.8](#) depicts a simple multi-layer perceptron with three layers: the input layer, a hidden layer, and the output layer. The input layer has three neurons, represented by circles, each of them containing one of three features, x_1 , x_2 , and x_3 . Each neuron in the input layer relates to each neuron in the hidden layer (there are three of

$$w_{ji}^{(h)}$$

them in this case). Every connection carries a *weight*, $w_{ji}^{(h)}$, where the notation means that we are connecting neuron j in layer h with the neuron i in the next layer. Each of the inputs is multiplied by the weight associated with each link and passed to the hidden layer; each neuron in the hidden layer receives as input the weighted sum of the features and transforms it through an *activation function*, $f(\cdot)$:

$$H_1 = f\left(w_{11}^{(1)}x_1 + w_{21}^{(1)}x_2 + w_{31}^{(1)}x_3 + b_1\right) \quad (4.4a)$$

$$H_2 = f\left(w_{12}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{32}^{(1)}x_3 + b_2\right) \quad (4.4b)$$

$$H_3 = f\left(w_{13}^{(1)}x_1 + w_{23}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_3\right) \quad (4.4c)$$

The term b_1 is called a *bias* and it is often added to the weighted sum of the features. It is a constant, like the intercept in a standard linear regression.

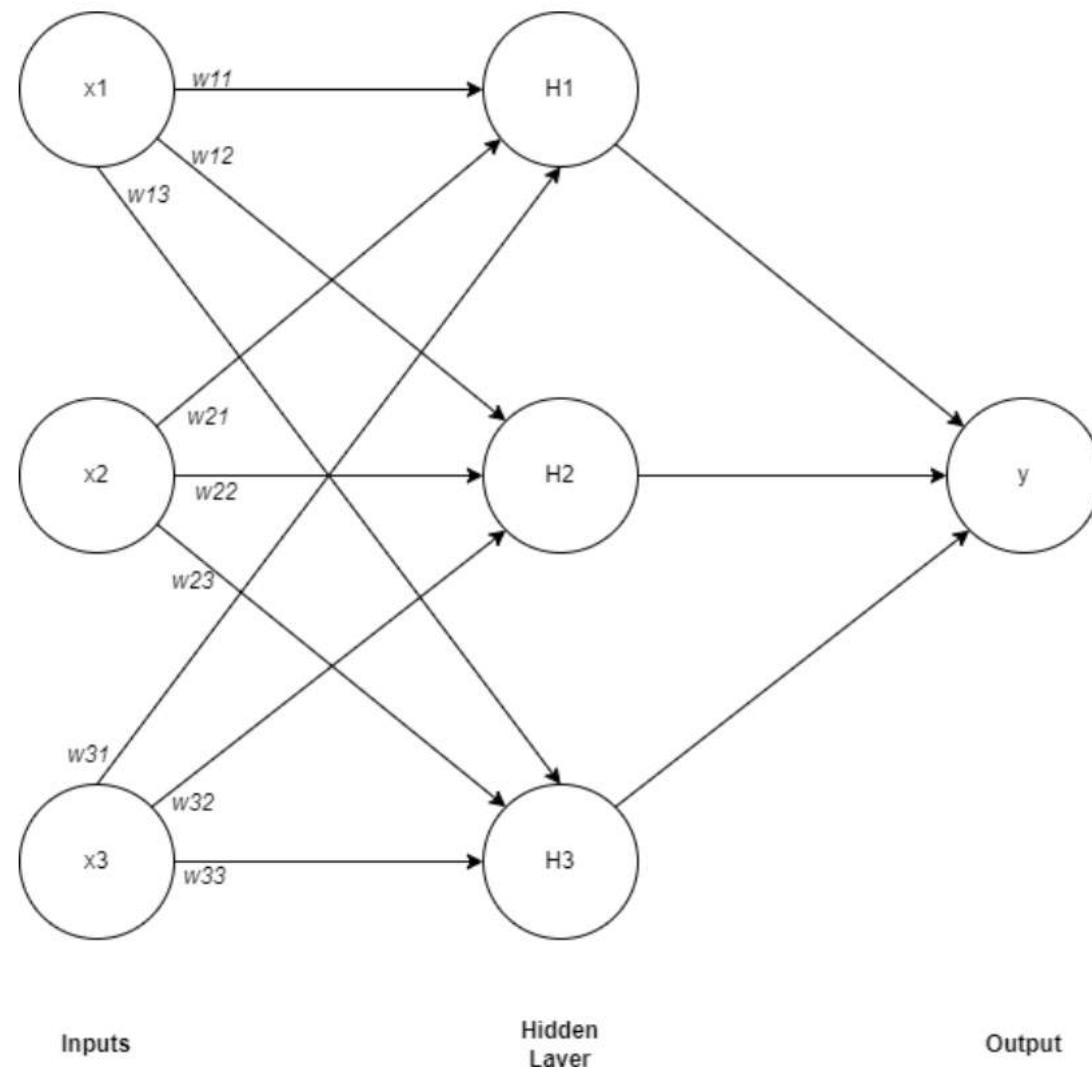


Figure 4.8 A pictorial representation of a neural network with three inputs (features) and three hidden units in a single hidden layer and with one output (target).

The activation function introduces nonlinearity into the relationship between the inputs and output. Without it, the outputs from the model would merely be linear combinations of the hidden layer(s), which would, in turn, be linear combinations of the inputs. Such a structure would be, in essence, a linear regression and not of interest because the purpose of a neural network is to discover complex nonlinear relationships.

The neurons in the hidden layer become inputs for the next layer (in this case, the output layer); they are

$$w_{11}^{(2)}, w_{21}^{(2)} \text{ and } w_{31}^{(2)}$$

multiplied by the weights and again transformed by means of an activation function. The result is the prediction of the output. The process of propagating the attributes from the input layer to the output is called feeding forward. Notably, although in this case the output layer only contains a single neuron, this does not have to be the case; for example, in a multiclass classification problem each class can be assigned its own output neuron.

A multi-layer perceptron can also contain more than one hidden layer. We shall discuss the choice between shallow (only one or two hidden layers) and deep (several hidden layers) neural networks later in this chapter.

Deep learning refers to machine learning methods utilizing multiple neural network layers to extract the nonlinear relationships embedded in the data being modeled. The deep learning methods are widely used in natural language processing, generative artificial intelligence, image processing and other areas.

4.4.1 The Choice of Activation Function

There are several activation functional forms in common usage. The logistic (sigmoid) function we encountered in connection with logistic regression is a popular choice. As discussed in Chapter 3, the logistic function takes z as an input and returns $f(z)$:

$$f(z) = \frac{1}{1 + e^{-z}} \quad (4.5)$$

This function outputs a value between zero and one. Other examples of activation functions are the softmax function, the rectified linear unit (ReLU), the leaky ReLU, and the hyperbolic tangent.

The softmax function is a more generalized version of the logistic activation function that is used in multiclass classification problems. For $z_i = z_1, z_2, \dots, z_k$, where K is the number of classes, it delivers:

$$f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (4.6)$$

Softmax applies the standard exponential function to each element z_i of the vector z of inputs and normalizes the result by dividing by the sum of all these exponentials. The sum of all $f(z_i)$ is 1.0. This function exponentially magnifies the importance of the largest members of the input values.

In contrast to the sigmoid and the softmax functions, the ReLU activation function is unconstrained from above. The ReLU activation function takes z as input and returns:

$$f(z) = \max(z, 0) \quad (4.7a)$$

When the input is negative, it returns zero and when the input is positive it returns the value itself. The leaky ReLU activation function is a variation of the ReLU function which allows for small negative numbers:

$$f(z) = \begin{cases} 0.01z, & z < 0 \\ z, & z \geq 0 \end{cases} \quad (4.7b)$$

Finally, the hyperbolic tangent activation function:

$$f(z) = \frac{e^{2z} - 1}{e^{2z} + 1} \quad (4.8)$$

has a similar shape to the logistic function but it produces values between -1 and $+1$ rather than between zero and one. [Figure 4.9](#) depicts the shapes of the logistic (top left), the ReLU (top right), and the hyperbolic tangent (bottom left) functions for different values of the input z .

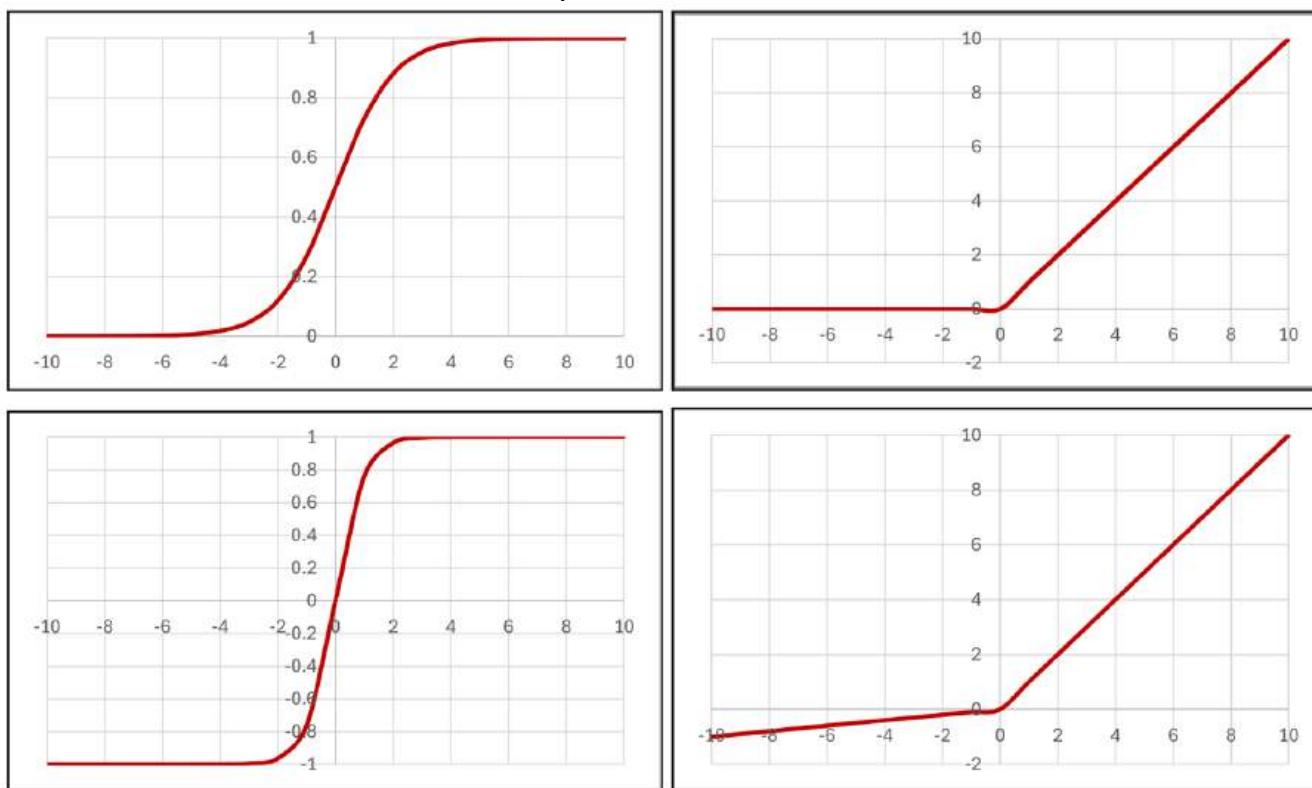


Figure 4.9 The logistic (top left), the ReLU (top right), the hyperbolic tangent (bottom left), and the leaky ReLU (bottom right) activation functions.

Notably, each layer can employ a different activation function (while all the neurons in the same layer apply the same activation function). However, it is common to use the same activation function for all the hidden layers (when there is more than one) and a different function for the output layer. For the multi-layer perceptron (and convolutional neural networks, which will be discussed later) a common choice is to use the ReLU function for hidden layers. When a recurrent neural network (which we will also discuss in Chapter 10) is employed, popular choices of activation function for the hidden layers are instead the logistic and the hyperbolic function. The activation function of the output layer tends to depend on the problem at hand; the logistic function is a common choice for binary classification problems whereas the softmax function is typically employed for multiclass classification problems.

4.4.2 A Numerical Example

The following (simplified) numerical example illustrates the functioning of a basic neural network. The structure of the network is depicted in [Figure 4.10](#). There are three layers: an input layer, a hidden layer, and an output layer. The input layer contains four neurons, as many as the features. Assume that for one instance, the values of the features are $x_1 = 0.6$, $x_2 = -0.4$, $x_3 = 0.3$, and $x_4 = -0.2$. The output of the neural network is used to perform binary classification, i.e., to determine whether the observation belongs to the '0' class or '1' class. In the hidden layer there are only two neurons.⁹

Estimation of the weights will be discussed below, but for now assume that they have already been determined and are as follows:

$$\begin{array}{ll}
w_{11}^{(1)} = 0.15 & w_{12}^{(1)} = -0.2 \\
w_{21}^{(1)} = -0.4 & w_{22}^{(1)} = 0.3 \\
w_{31}^{(1)} = -0.3 & w_{32}^{(1)} = -0.6 \\
w_{41}^{(1)} = 0.2 & w_{42}^{(1)} = -0.2
\end{array}$$

$w_{01}^{(1)} = 0.7$ and $w_{02}^{(1)} = 0.3$. The activation function for the hidden layer is ReLU.
the biases are $w_{01}^{(1)} = 0.7$ and $w_{02}^{(1)} = 0.3$. The activation function for the hidden layer is ReLU.
The value of z_1 is obtained via multiplying the weights by the features, and summing their values (and the bias):

$$\begin{aligned}
z_1 &= w_{11}^{(1)}x_1 + w_{21}^{(1)}x_2 + w_{31}^{(1)}x_3 + w_{01}^{(1)} \\
&= 0.15(0.6) - 0.4(-0.4) - 0.3(0.3) + 0.2(-0.2) + 0.7 \\
&= 0.82
\end{aligned}$$

Then the chosen activation function is applied to obtain H_1 :

$$H_1 = \max(z_1, 0) = \max(0.82, 0) = 0.82.$$

Similarly, z_2 is calculated as:

$$z_2 = -0.2(0.6) + 0.3(-0.4) - 0.6(0.3) - 0.2(-0.2) + 0.3 = -0.08$$

which is transformed into:

$$H_2 = \max(-0.08, 0) = 0.$$

In the output layer there is one neuron (assume that the problem is a binary classification) and the activation

$w_{11}^{(2)} = 0.4$ and $w_{21}^{(2)} = 0.2$; the bias is $w_0^{(2)} = 0.3$.
function is logistic. The weights are

The value of the output neuron (which is our prediction) is obtained by applying the logistic activation function to the weighted sum of H_1 and H_2 :

$$\hat{y} = \frac{1}{1 + e^{-(0.3+0.82(0.4)+0(0.2))}} = 0.65.$$

This can be interpreted as the probability that the value of y is 1 for that instance. Because $0.65 > 0.5$, the observation is assigned to the 1 class.

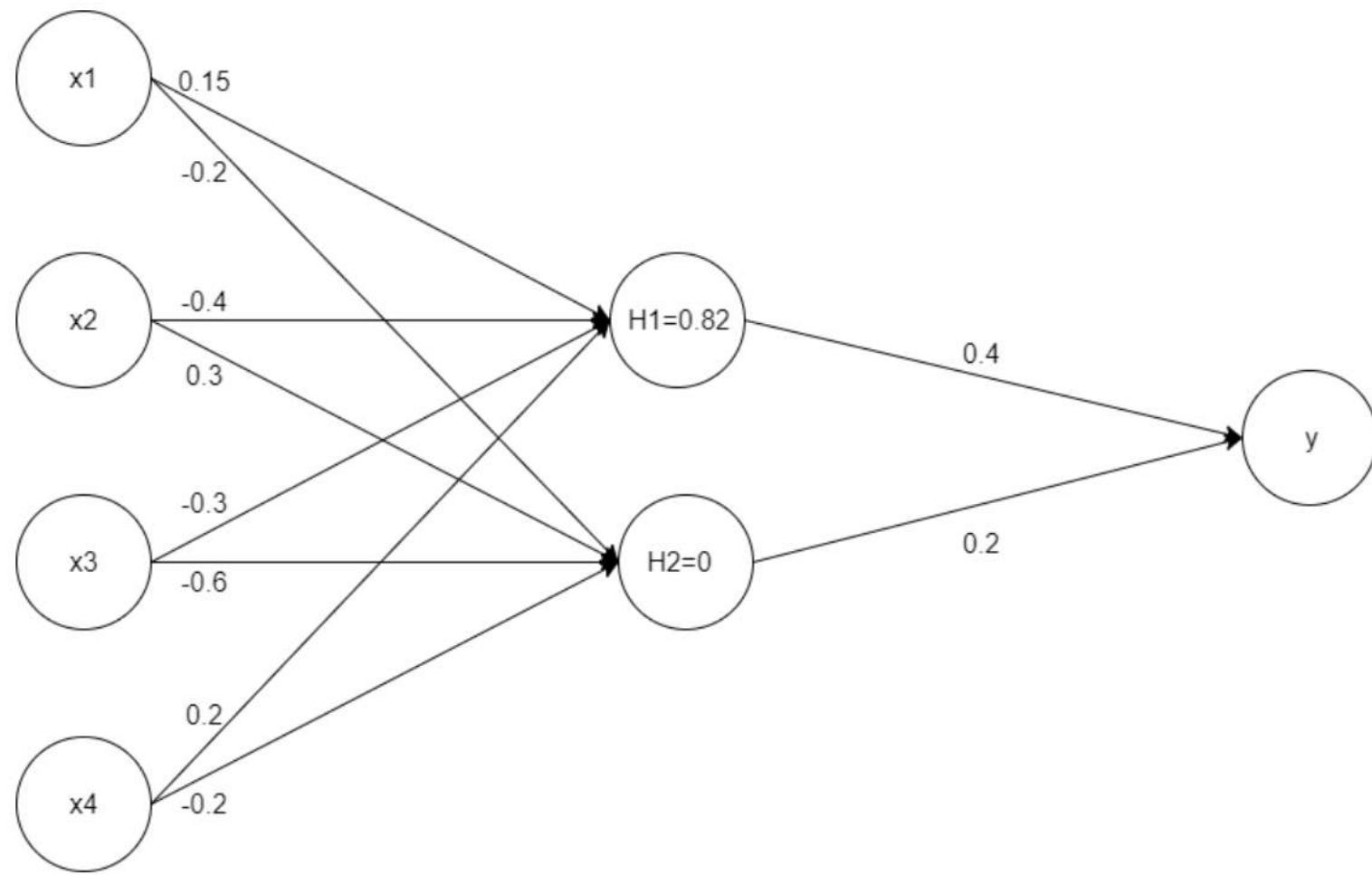


Figure 4.10 A fictional multi-layer perception with three layers.

• Note that there is no need for the number of neurons in the hidden layer(s) to be equal to the number of neurons in the input layer, it could be more or fewer.

4.4.3 Backpropagation

The sets of weights that link the neurons are parameters. These were given in the previous example, but they must be estimated. The strategy is to find weights such that some measure of classification or prediction error, a loss function, typically residual sum of squares or mean squared error (MSE), is minimized.¹⁰ The procedure is recursive. At the beginning, the weights are assigned random values (typically within the range $[-0.1, 0.1]$). Then, a first training example is forward propagated to the network's output. Then the weights are updated using the error between the calculated and actual values of the labels. After updating the weights, a new example is introduced, and the weights are updated again. When the last training example is introduced, one *epoch* (i.e., iteration) is completed. In general, the successful training of a network requires many epochs (typically, tens of them for a simple architecture but it could be in the thousands depending on the complexity of the network and the convergence rate of the optimizer). Therefore, neural networks are usually regarded as a computationally intensive technique. Further details of the backpropagation method are given in Chapter 7.

¹⁰ See chapter 7 for further details on these metrics

4.4.4 Architectural Issues

Neural networks are an incredibly flexible tool. In fact, mathematicians have proven that, with the right number of hidden neurons, and under some assumptions, neural networks are able to approximate any function with arbitrary precision. This result is known as the *universal approximation theorem*. However, in practice, neural networks are very prone to overfitting the training data and this problem is worse with large networks.

Usually, one can start with one or two hidden layers, and add more layers after assessing the performance of the model. Adding more layers is useful for modeling more complex phenomenon. This leaves us with the question of how to determine the number of hidden layers and the number of neurons within each layer to use. One solution could be to experiment with different sizes of network, compute their accuracy over the test sample, and pick the network structure that minimizes the error rates. However, neural networks are highly computationally intensive. Consider a network with 100 features, one hidden layer with 100 neurons, and an output layer with only one neuron. This is not an unlikely case in real life applications and entails the estimation of $100 \times 100 + 100 =$

10,100 weights! Besides, many epochs are generally needed to reach convergence, which makes this approach highly impractical.

An alternative technique to find an appropriate size of network proceeds as follows. We start with a small number of hidden layers. At the end of each epoch, the algorithm computes the value of the loss function over the training set. This value is likely to keep decreasing when the number of epochs increases, until a point is reached where the performance fails to improve. This can happen either because the network lacks the necessary flexibility to make correct predictions, or because a local minimum has been found. When this is observed, additional layers are added to the network and the training is resumed. If this allows a reduction in the value of the loss function, the newly added neurons are retained; if not, the smaller model is preferred.

The number of neurons within each layer is dictated by the sizes of the features set and target(s). It was common practice to structure the network such that the number of neurons decreased from one layer to the next as the network approaches the output layer. This “pyramid” style structure has been largely superseded by a more uniform structure with an equal number of neurons in the hidden layers coupled with regularization techniques to ensure that the model is not overfitting. Still, depending on the model, it may be useful to have a larger number of neurons in the first layer than in the other layers.

Systematic approaches for selecting hyperparameters such as the number of hidden layers are discussed in Chapter 7.

4.4.5 Overfitting

As stated above, neural networks’ greatest advantage, their flexibility, is also their Achilles’ heel. Especially when the model is large and insufficient training examples have been provided, neural networks tend to learn random artifacts of the training data. This implies that they will fail to generalize well to unseen test instances. An extreme form of overfitting is *memorization*, which results in an almost perfect fit to the training data, and which is not uncommon with neural networks. Signs of overfitting are:

- The same model obtains very different predictions depending on the sample it is trained with.

- The gap between the prediction error over the training and the test samples is very large.

Apart from avoiding parameter proliferation, there are a few techniques that can be used to reduce overfitting. These include:

- **Penalty based regularization.** This is like the discussion in Chapter 7 and involves imposing a penalty over the loss function, such as $\lambda \sum_{i=0}^d w_i^2$, where d is the number of neurons.
- **Dropout.** This is an ensemble technique explicitly designed for neural networks. It consists of creating alternative networks by selectively dropping a few neurons each time. The forecasts from these networks are then aggregated to create the final prediction.
- **Early stopping.** The optimization is stopped before converging to the optimal solution on the training data. A portion of the data is held out and used to determine the optimal stopping point. The training is stopped when the error in the hold-out sample starts to rise.

4.4.6 Advanced Neural Network Structures

Before closing our discussion of neural networks, we should briefly mention some advanced neural network architectures.

Convolutional Neural Networks

Convolutional neural networks (CNN) are a specialized form of neural networks where the neurons in one layer are only connected to a subset of the neurons in the next layer. They are designed to work with inputs that have a grid structure and where adjacent points in the grid exhibit dependencies. The most obvious application of CNNs is with 2-dimensional images, but they can also be employed for textual, voice or time series data. CNNs are ideal in such cases because the number of network weights to be trained is drastically reduced, which results in faster training of the model. CNNs take their name from the fact that they contain at least one *convolutional layer*. The most common type of convolutional layer is the 2D or planar convolutional layer. A 2D convolution applies an $n \times n$ kernel matrix W (where n is generally smaller than m) over a $m \times m$ input grid (typically, this is called the

image regardless of whether or not it contains pixels) to obtain a new filtered image that has a smaller size than the original image. The latter is called *feature map*. The kernel matrix contains weights that should be learned during the training process, but in this example, assume that this has already been done. To illustrate how it works, one could assume that the input is a 4×4 image that we represent with the matrix X :

This can be filtered using a 3×3 kernel:

The feature map is obtained by “sliding” the kernel over the image starting from the top left corner to move the kernel through all the positions where it fits entirely within the boundaries of the image. Each position corresponds to a single cell in the feature map, the value of which is calculated by multiplying together the kernel value and the underlying image for each of the cells in the kernel, and then adding all these numbers together.

In practice, for the image and kernel above, we will start by placing the kernel over the area in red below (top left corner):

$$X = \begin{pmatrix} \boxed{\begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 2 & 1 & 0 \end{matrix}} & 2 \\ 0 & 2 & 2 & 1 \end{pmatrix}.$$

We can obtain the first element of the feature map as:

Then, we slide the kernel to the area in blue:

$$X = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 2 & 1 & 0 & 1 \\ 0 & 2 & 2 & 1 \end{pmatrix}$$

and obtain:

Then, we slide the kernel to the area in green:

$$X = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 2 & 1 & 0 & 1 \\ 0 & 2 & 2 & 1 \end{pmatrix}$$

and obtain:

It is now intuitive to make the move to final position and obtain the feature map:

$$F = \begin{pmatrix} 3 & 4 \\ 2 & 2 \end{pmatrix}.$$

The area in red (blue, green) is termed a *receptive field*. It is the region in the input space that influences a cell in the feature map.

A non-linear layer, where a non-linear activation function is applied to the feature map, can also be added after the convolutional layer. It is also common to have a pooling layer. The pooling layer replaces the output of the previous layer at certain locations with summary statistics. For instance, it would be possible to summarize F by taking the average or the maximum of the values in the cells. CNNs are parsimonious in terms of parameters as the same weights are applied to all the receptive fields. Therefore, CNNs are useful to process images, which typically involve millions of pixels.

Recurrent Neural Networks

Recurrent neural networks (RNNs) differ from a standard multi-layer perceptron as the former models employ a temporal sequence to preserve the order in which the observations occur. In other words, a RNN is designed to have some memory. RNNs are often employed in time series applications, and they are at the heart of large language models; therefore, they will be discussed in more detail in Chapter 10.

4.5 Autoencoders

Autoencoders are a class of artificial neural network models (ANNs) used for unsupervised learning. They are feedforward specifications, but the outputs are the same features as the inputs, and hence there are no labels. Unlike K -means clustering, autoencoders are primarily used for dimensionality reduction and so are best thought of as non-linear extensions of PCA. Autoencoders can provide a compact representation of the feature data and are particularly useful for high-dimensional systems. It should be noted that although PCA is commonly discussed in machine learning contexts, in fact there is no learning involved because it is merely a decomposition with a unique mathematical solution. Autoencoders, on the other hand, are trained and learn the relationships present in the data through model estimation. The advantage of autoencoders over PCA is the use of non-linear activation functions which provide the universal approximation property in high dimensional space.

[**Figure 4.11**](#) provides a pictorial representation of an autoencoder. Beginning on the left-hand side, each circle represents a feature ($m = 5$ in this illustration). The features are then put through the *encoder*, which is a function, to arrive at the values in the hidden layer (also known as latent variables). In the second part of the autoencoder, the values in the hidden layer are converted back to the feature values through the *decoder*. Each line connecting

the circles represents a weight or parameter to be estimated. The optimization objective is to reconstruct the original features as accurately as possible. The weights between the input layer and the hidden layer encode the information from the features, and the weights between the hidden layer and the output layer decode the information.

In most applications, there are fewer units in the hidden layer than there are features, which is the case in [Figure 4.11](#) where there is a single hidden layer containing three neurons. This is known as a *constricted* or *bottleneck* hidden layer and will lead to dimensionality reduction.

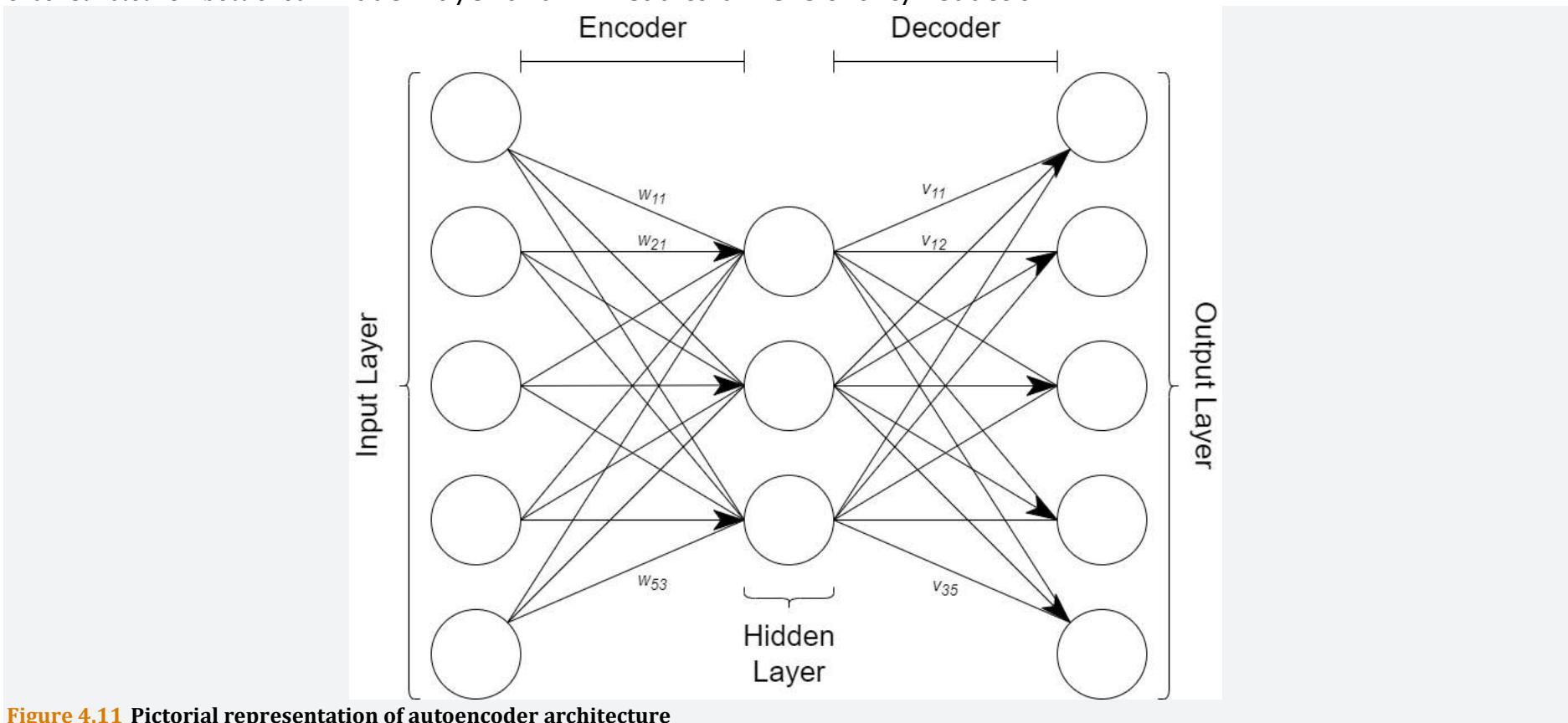


Figure 4.11 Pictorial representation of autoencoder architecture

The hidden layers are simply calculated as a weighted sum of the inputs (features), and the outputs (reconstructed features) are a weighted sum of the values at the hidden layers. Let the inputs be denoted by x_{ij} , for $j = 1, \dots, m$ and $i = 1, \dots, N$, for each feature j and datapoint i . If $m = 5$ as in [Figure 4.11](#), and using w_{ij} to denote the weights from the input to hidden layer and w_1 to denote a bias (constant term), the values at the hidden layer could be calculated as follows for each data point:

$$H_{1i} = w_1 + w_{11}x_{1i} + w_{21}x_{2i} + w_{31}x_{3i} + w_{41}x_{4i} + w_{51}x_{5i} \quad (4.9)$$

There are similar expressions for the other two hidden units. The values at the hidden layer, H_{1i}, H_{2i}, \dots are the reduced dimension representation of the data, sometimes known as the *code*.

If we let \hat{x}_{ij} denote the fitted output (i.e., the reconstituted features) from the model this would be calculated as:

$$\hat{x}_{1i} = v_1 + v_{11}H_{1i} + v_{21}H_{2i} + v_{31}H_{3i} \quad (4.10)$$

The weights are chosen by minimizing a loss function, L , akin to the residual sum of squares (RSS) in a linear regression, which is the difference between the actual values of the features and their reconstituted values, \hat{x}_{ij} :

$$L = \sum_{j=1}^m \sum_{i=1}^N (x_{ij} - \hat{x}_{ij})^2 \quad (4.11)$$

An alternative measure called the mean squared error (MSE) which is the mean of the squared residuals is also used as the loss function:

$$L = \frac{\sum_{j=1}^m \sum_{i=1}^N (x_{ij} - \hat{x}_{ij})^2}{mN} \quad (4.12)$$

L will be a positive number to reflect that the feature inputs will not be reconstructed precisely after the encoding and decoding processes, but this is the price paid to obtain a more parsimonious representation. Autoencoders can

be used, for example, to store images such as photos without requiring much space, but when they are reconstructed, some definition will inevitably be lost.

As outlined until this point, the autoencoder model is linear in the weights, and thus it will perform a function comparable to PCA. More specifically, in such a linear model, if there is only one hidden layer with K nodes, if both the encoder and decoder are linear, and if the inputs are suitably normalized, then the encoder hidden nodes will be the first K principal components.

However, it is more common to use a non-linear autoencoder by applying an *activation function* to the weighted sums in the hidden layer(s). The activation functions introduce nonlinearity into the relationship between the inputs and output. Without them, the outputs from the model would merely be linear combinations of the hidden layer(s), which would, in turn, be linear combinations of the inputs.

There are several activation functional forms in common usage. The logistic (sigmoid) function we encounter in connection with logistic regression (see Chapter 3) is a popular choice. The value at one of the hidden neurons would now be:

$$H_{1i} = \phi(w_1 + w_{11}x_{1i} + w_{21}x_{2i} + w_{31}x_{3i} + w_{41}x_{4i} + w_{51}x_{5i}) \quad (4.13)$$

where ϕ is the activation function.

By capturing any non-linear relationships between the features that are present in the data, activation functions can also allow the number of nodes in the hidden layer to be further reduced so that the representation is even more compact than if a purely linear specification was used.

The number of hidden neurons is usually lower than m so that dimensionality is reduced. If the number of hidden neurons was exactly m , it would be possible to trivially reconstruct the exact features, but this would be pointless because no reduction in dimensionality would have been achieved. The number of hidden neurons is set to a value greater than m in the context of a *sparse* autoencoder. Hence, even though the number of hidden units is large, many of the weights (e.g., 80% of them) are set to zero, so that the effective number of weights is much lower and commensurate with a smaller number of hidden units.

It is also possible to add additional hidden layers to form a *deep autoencoder*, which has more scope to capture more sophisticated non-linear patterns between the features. In such models, the design is usually (although not necessarily) symmetrical about the bottleneck center hidden layer. So, for instance, if we have $m = 10$ features,

we might have a center layer with three neurons and two intermediate hidden layers (one on the encoder side and one on the decoder side), each with five neurons.

We evaluate autoencoders by calculating the loss, L , as described above, on the final fitted model, where it is called the reconstruction error.

4.5.1 Autoencoder Example

We examine an autoencoder application to dimensionality reduction in modeling the daily changes in Treasury yield series examined previously in Chapter 1. There are eight Treasury series in the dataset used in Chapter 1. We have seen that the fewer principal components are used, the more the dimensionality is reduced but the greater extent to which information is lost. The leaky ReLU activation function is used in the autoencoder model.

To determine whether the autoencoders performed better than PCA for modeling the changes in yield, we compare the former's loss metrics (MSE) with those from an application of PCA to the same dataset in [Table 4.4](#). The autoencoder approach has performed similarly to PCA with MSEs of the two methods close to each other.

Table 4.4 Loss Measures (MSE) for PCA and Autoencoders for Modeling the Changes in Treasury Yields

| Number of components for PCA | MSE for PCA | Number of hidden units for autoencoder | MSE for autoencoder |
|------------------------------|-------------|--|---------------------|
| 1 | 0.18208 | 1 | 0.18270 |
| 2 | 0.04965 | 2 | 0.05027 |
| 3 | 0.01708 | 3 | 0.01740 |
| 4 | 0.00848 | 4 | 0.00854 |
| 5 | 0.00528 | 5 | 0.00534 |
| 6 | 0.00295 | 6 | 0.00299 |

| Number of components for PCA | MSE for PCA | Number of hidden units for autoencoder | MSE for autoencoder |
|---------------------------------|----------------|---|------------------------|
| 7 | 0.00107 | 7 | 0.00113 |
| 8 | 0.00000 | 8 | 0.00000 |

Appendix 4.A Technical Details of How SVM Boundaries Are Determined

In this Appendix, we consider the two-dimensional case with features x_1 and x_2 . Let the upper margin constraint be defined as:

$$w_0 + w_1 x_{1i}^{(u+)} + w_2 x_{2i}^{(u+)} = 1 \quad (4A.1a)$$

while the lower one is:

$$w_0 + w_1 x_{1j}^{(l-)} + w_2 x_{2j}^{(l-)} = -1 \quad (4A.1b)$$

where w_0 , w_1 , and w_2 are the weights to be estimated, the $l-$, $u+$ superscripts denote the data points corresponding to the lower and upper margins, respectively, and x_1 , x_2 denote features 1 and 2, respectively.

Subtracting the second of these two equations from the first provides us with an equation for the margin width to be maximized:

$$w_1(x_{1i}^{(u+)} - x_{1j}^{(l-)}) + w_2(x_{2i}^{(u+)} - x_{2j}^{(l-)}) = 2 \quad (4A.2)$$

Note that $(x_{1j}^{(l-)}, x_{2j}^{(l-)})$ and $(x_{1i}^{(u+)}, x_{2i}^{(u+)})$ are two specific instances of features.

We also need to apply two constraints to ensure that all the points lie on or outside of the estimated margins:

$$w_0 + w_1 x_{1i}^+ + w_2 x_{2i}^+ \geq 1, i = 1, \dots, N_1 \quad (4A.3a)$$

and:

$$w_0 + w_1 x_{1j}^- + w_2 x_{2j}^- \leq -1, j = 1, \dots, N_2 \quad (4A.3b)$$

The $-$, $+$ superscripts denote the data points corresponding to class 1 and class 2, respectively and N_1 and N_2 are the total number of observations in each of classes 1 and 2, respectively.

Using the index j to now denote all the observations regardless of the class to which they belong, we could combine the constraints as:

$$y_i(w_0 + w_1 x_{1j} + w_2 x_{2j}) \geq 1, j = 1, \dots, N \quad (4A.4)$$

where x_{1j} and x_{2j} combine all the positive and negative outcomes, y_i is $+1$ if the observation belongs to class 1 and -1 if the observation belongs to class 2, and $N = N_1 + N_2$.

$$(x_{1j}^{(l-)}, x_{2j}^{(l-)}) \text{ and } (x_{1i}^{(u+)}, x_{2i}^{(u+)})$$

By considering the relationship of through w_1 , and w_2 , one can show that

$$|w| = \sqrt{w_1^2 + w_2^2}$$

the margin width is given by $2/|w|$, where

is the Euclidean norm of the weights.

Maximizing the margin width is equivalent to minimizing $|w|$. However, rather than working with $|w|$, it is often

easier to minimize $\frac{1}{2} |w|^2$ because its derivative is just w . Therefore, we can write the optimization problem as:

$$\min_{w_0, w_1, w_2} \frac{1}{2} |w|^2 \text{ subject to } y_j(w_0 + w_1 x_{1j} + w_2 x_{2j}) \geq 1, j = 1, \dots, N. \quad (4A.5)$$

To allow for overlapping classes that are not linearly separable, we employ a *hinge (max) function*, which sets the penalty to zero for correct classifications and to the distance between the point and the decision boundary for incorrect classifications. We can write the optimization problem as minimizing the function below:

$$L(w) = \frac{1}{N} \sum_{j=1}^N \max(1 - y_j(w_0 + w_1 x_{1j} + w_2 x_{2j}), 0) + \frac{\lambda}{2} |w|^2 \quad (4A.6)$$

The function $\max(\cdot)$ is zero for all correct classifications regardless how far the point is from the margin, but it will be equal to the distance between the boundary and the point for incorrect classifications. The regularization here is somewhat like a ridge regression, with the hyperparameter λ controlling the relative weights on the margin width versus incorrect classifications.

Appendix 4.B Pricing Options Using Neural Networks

Introduction

Ever since Black, Sholes and Merton developed their option pricing model, there have been many refinements and extensions to price options by incorporating dividends and volatility smile and to price such diverse instruments as exotic options, and bond options. These models, whether they are analytical or numerical, rely on various assumptions regarding the underlying asset returns (Hull, 2017). Hence, the prices derived from models may not

always reflect market realities due to the underlying assumptions. Moreover, in some cases, a theoretical model that accurately prices the options may not be available. Therefore, methods that are not dependent on assumptions about options markets are very attractive to market participants for trading or risk management purposes.

Machine learning technology is particularly useful for developing alternative pricing models which do not rely on a priori assumptions on underlying asset price distribution. An example of an early AI based option model is Hutchinson, Lo, and Piaggio's artificial neural network (ANN) model. They used two years of option prices to build an ANN model and found that the model can be used to price and hedge options out of sample.

Option Prices

The Black Scholes model assumes that stock returns are normally distributed, and stock prices are log normally distributed. It also assumes that there are no transaction costs, the assets are traded continuously, short selling is allowed, and that there are no arbitrage opportunities in the market. The Black-Scholes value for a European call option on a non-dividend paying stock is:

$$C = S N(d_1) - K e^{-rT} N(d_2)$$

where,

S = Stock Price

K = Strike Price

r = risk free rate

T = time to expiration

σ = volatility of stock returns

$$d_1 = (\ln(S/K) + (r + \sigma^2/2)T)/\sigma T^{1/2}$$

$$d_2 = d_1 - \sigma T^{1/2}$$

$N(d_1)$ and $N(d_2)$ are cumulative probability distribution functions for a standardized normal distribution.

Neural Network Model for Option Prices

In this example we train an ANN to calculate the price of a European call option on a stock. The goal is to build an ANN model that mimics the Black Scholes option pricing formula.

In an artificial neural network, there are multiple layers of neurons. The input layer consists of the features of the model. In the present case they are the inputs to the Black Scholes formula. The final layer consists of the label or labels being calculated using the ANN. In this example there is a single output – the call price. In each of the layers between the input and output layers, there are several artificial neurons or nodes. Each node has a value and an activation function associated with it. The neurons in each layer are connected to those of the previous layers. Each of these connections have weights associated with them. The logistic, ReLu, and tanh functions are some of the activation functions typically used in ANNs.

The activation function at each node relates its value to the values of the nodes in the previous layer. The objective in building the ANN model is to determine the parameters of the ANN such that the predictions of the ANN are as close as possible to the actual training values. The parameters are determined by minimizing the cost or loss function, which is typically the mean squared error or other similar measure. The process of fitting an ANN involves multiple forward and backward passes through the network and the loss function is minimized using gradient descent algorithm or similar algorithms.

Data

The data used for training and testing the ANN model is generated synthetically using Black Scholes equation. We use the following data to simulate the stock price, strike price, time to expiration, risk-free rate, and volatility. This data is then split into training and test data, and scaled using scikit-learn library's StandardScaler function.

Table 1 Values of different parameters used in the ANN models

| Item | Min | Max |
|-----------------|-----|-----|
| Stock price (S) | 30 | 100 |

| Item | Min | Max |
|-------------------------|-------|-------|
| Strike price (K) | 0.5 S | 1.5 S |
| Time to expiration (t) | .25 | 1.0 |
| Risk-free rate (r) | 0.01 | 0.1 |
| Volatility (σ) | 0.05 | 0.5 |

The generated option price data is divided into training and test data sets. We use a sample size of 20000, of which the first 15000 are used for training the model and the remaining data is used for testing the trained model. The features data is scaled using the mean and standard deviation of each feature in the training data before fitting the ANN model.

ANN Model for Pricing Call Options

The objective is to predict call price given S , K , T , r , and σ . We use the scikit-learn library's *MLPRegressor* to develop an ANN model for this purpose. The logistic function is used as the activation function in this model. The Adam (Adaptive Moment Estimation) solver in scikit-learn is used for estimating the model parameters.

The *random_state* variable is set at 42 in this analysis. Fifteen configurations of the ANN model are created with different neurons in each layer as shown in the table below.

The cost function used in *MLPRegressor* is:

Where W_i are penalty terms used for regularization of the model. Here α is parameter that controls the penalty terms. In the present analysis, we set α to zero. The cost function is plotted for the models in the figure below. It is seen that the model is converging to theoretical values as we increase the number of neurons in the NN models. The performance of the models on the test and training and test data set are shown in the table below. It can be seen from the results that as the number of neurons are increased in a one-layer configuration, the MSEs decrease

rapidly. In addition, adding more layers results in further reduction in MSEs. The MSEs of a 16x16x16 and 32x32 network configurations are lower than that of a single layer of 64 neurons. In practice, a grid search is usually conducted to determine the best configuration and values of other hyperparameters for developing a neural network model.

Table 2 MSEs of the training and test data sets

| # of Layers | # of Neurons in each layer | MSE Training | MSE Test |
|-------------|-------------------------------|--------------|----------|
| 1 | 4 | 0.3577 | 0.3576 |
| 1 | 8 | 0.1731 | 0.1678 |
| 1 | 16 | 0.1399 | 0.1374 |
| 1 | 32 | 0.0727 | 0.0720 |
| 1 | 64 | 0.0273 | 0.0269 |
| 2 | 4 | 0.1741 | 0.1714 |
| 2 | 8 | 0.0711 | 0.0705 |
| 2 | 16 | 0.0593 | 0.0557 |
| 2 | 32 | 0.0081 | 0.0083 |
| 2 | 64 | 0.0074 | 0.0077 |
| 3 | 4 | 0.1562 | 0.1550 |
| 3 | 8 | 0.0668 | 0.0664 |
| 3 | 16 | 0.0165 | 0.0169 |

| # of Layers | # of Neurons in each layer | MSE Training | MSE Test |
|-------------|-------------------------------|--------------|----------|
| 3 | 32 | 0.0123 | 0.0120 |
| 3 | 64 | 0.0063 | 0.0060 |

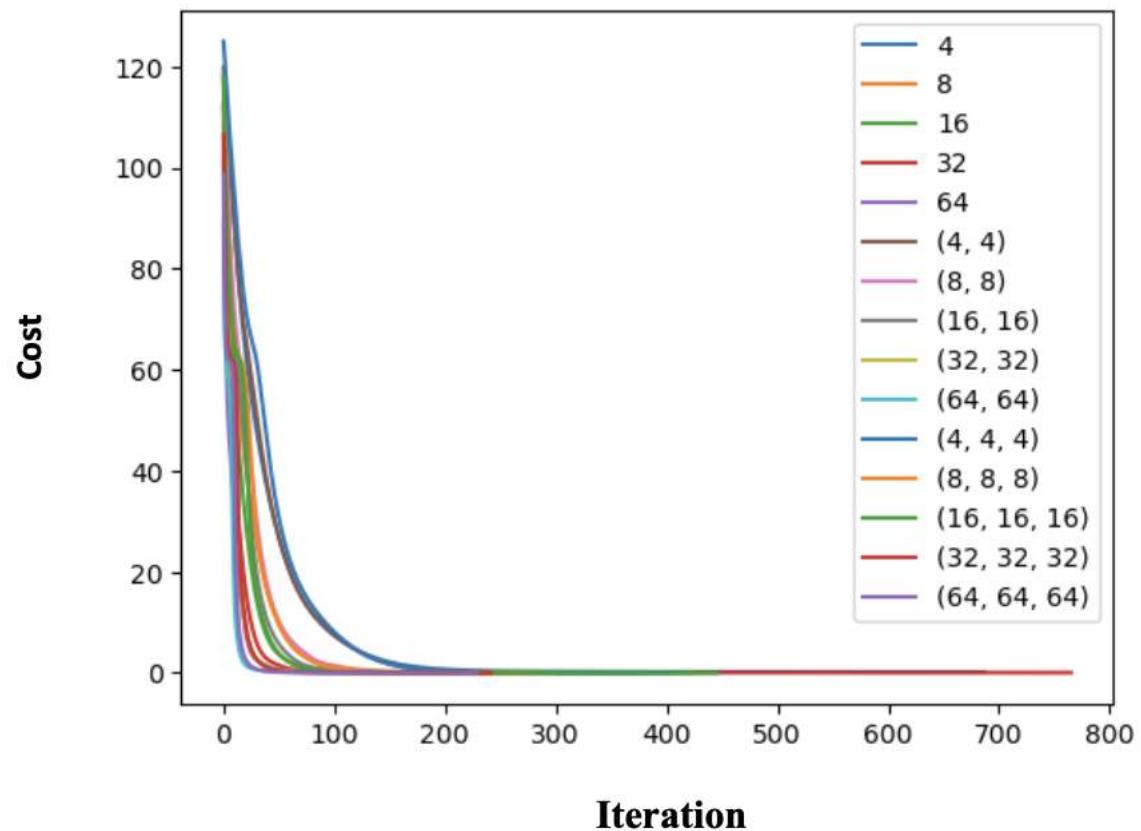


Figure 1 ANN Cost function vs. Iteration

Concluding Remarks

Using a machine learning model to predict plain vanilla stock options for which closed form solutions exist may appear to be an unnecessary exercise. However, machine learning techniques certainly have applications for pricing difficult to value options for which theoretical models are either not available or are computationally intensive to use. Such models can be trained using historical trading prices for derivatives which are difficult to value using analytical models. In addition, the machine learning based models may be able to capture real life market conditions such as market inefficiencies and regime shifts more accurately than conventional models. While models based on theoretical assumptions are useful for understanding the various factors influencing option prices, the machine learning based models are useful for pricing exotics and for incorporating real market conditions in pricing options.

Supervised Learning – Part 2: Machine Learning Techniques: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

4.1 What are the main differences between regression and classification decision trees?

The main difference between classification and regression trees is type of the outcome variable. When the outcome variable is continuous, we refer to regression trees, and when it is categorical, we refer to classification trees. In both cases, we use a top-down recursive binary split to grow the tree. However, for regression trees the splits are decided to minimize the RSS; conversely, the splits in classification trees are decided to produce the largest drop in either entropy or the Gini coefficient.

4.2 What are the differences between bagging, boosting, and random forests?

Bagging, boosting, and random forests are all ensemble techniques that are based on obtaining a final prediction as the average between the forecasts of many trees. When the problem is a classification, a majority vote is generally used to determine the class to which an unseen instance is classified. Bagging relies on the construction of many sub-samples by randomly extracting observations from the training sample, with replacement. Trees are fitted on each of the sub-samples and the forecast is obtained as an average (or as the most popular class) across the predictions of those trees. Random forests are like bagging, but the correlation among the trees is reduced by using a random subset of $p < m$ features at each split of a tree. Finally, boosting is a sequential procedure where each new tree that is grown uses the information from the residuals of the previously grown trees. Two popular boosting algorithms are gradient boosting and adaptive boosting.

4.3 In the context of decision trees, what is pruning?

Pruning is a technique to reduce the size of the tree to avoid overfitting and enhance interpretability. Pre- or online pruning are conducted while the tree is grown by imposing stopping criteria (e.g., minimum number of observations in a node, maximum number of branches, etc.). Post pruning is conducted after the tree has been grown by replacing subtrees with leaves when the substitution does not decrease the predictive accuracy of the learner.

4.4 What are the main advantages and disadvantages of decision trees compared with other supervised machine learning techniques?

The main advantage of decision trees is their interpretability and the fact that they closely resemble the human decision-making process. Because of this, they are sometimes called ‘white box’ models. Their main disadvantage is that they are often less accurate than black box models such as neural networks. To enhance their performance, ensemble techniques are often used; however, this comes at the expense of interpretability.

4.5 How would K nearest neighbors proceed to classify a new instance given a training sample of 10,000 observations?

The first step involves choosing a measure of distance (e.g., Euclidean or Manhattan distance) and K, the number of nearest neighbors to be considered. For instance, we could set K equal to \sqrt{N} , which in this case is $\sqrt{10000}=100$. Then, the distances between the instance to be classified and each of the instances in the training sample are computed and the K nearest neighbors are identified. A majority vote is used to assign the unclassified instance to the class of most of the neighbors.

4.6 In the context of support vector machines, what is the maximum margin classifier?

The maximum margin classifier is the optimal decision boundary. It is the line (hyperplane) that is equidistant from the margin constraints and maximizes the margin (the distance between the margin constraints).

4.7 In the context of artificial neural networks, what is an activation function?

An activation function for the neuron. Popular choices for activation functions are the logistic (sigmoid) function, the softmax function, the ReLU, and the hyperbolic function.

4.8 Describe some potential ways to address overfitting in a neural network.

Overfitting is a common problem with neural networks, as their great flexibility is also their primary disadvantage. In addition to avoiding parameter proliferation, ways to address overfitting include penalty-based regularization, dropout, and early stopping. The first approach involves imposing a penalty on the loss function. The second relies on the generation of alternative networks by selectively dropping a few neurons. The final forecast is obtained by aggregating the prediction from the various networks. Finally, early stopping entails stopping the training before the convergence is achieved over the training sample. The stopping point is decided by looking at the error rate over a hold-out sample.

4.9 What is the primary purpose of Autoencoders?

Autoencoders are used principally for dimensionality reduction – in other words, for representing the most important characteristics of a dataset using a smaller number of (transformed) features.

4.10

A. How do autoencoders achieve dimensionality reduction?

Dimensionality reduction is achieved by constructing a network with fewer nodes in the hidden layer than in the input and output layers. For example, if there are $m = 11$ features in the dataset, there might be five nodes in the hidden layer that can approximate these features.

B. How do autoencoders differ from PCA in the way that they reduce the dimensionality of a dataset?

PCA reduces the dimensionality of a dataset by constructing a set of orthogonal (linearly independent) components that are linear combinations of the original features. Although there will be m principal components if there are m features, only the first K of those, which are ordered to explain most of the variation within the features, are retained. Autoencoders use a neural network specification, usually with a non-linear activation function, which will identify a non-linear combination of the original features that is able to closely approximate them.

4.11 Consider the neural network depicted below, estimated to model the probability of default of a credit card holder.

A. Describe the structure of the neural network.

This is a multi-layer perceptron with three layers: an input layer, a hidden layer, and an output layer. The input layer contains three neurons (equal to the number of features), the hidden layer contains two neurons, and the output layer contains one neuron.

B. Using the ReLU function at the hidden layer and the logistic function at the output layer, make a prediction for the default (=1) of a credit card holder who is 35 years old, has an income of \$100,000, and an outstanding amount of \$20,000. Assume that all the biases are equal to zero.

We shall first compute the value assigned to each of the two neurons in the hidden layer. Assuming a ReLU transfer function, we obtain:

$$H_1 = \max(35 \times 0.25 + 100,000 \times 0.2 - 20,000 \times 0.8, 0) = \max(4,008.75, 0) = 4,008.75$$

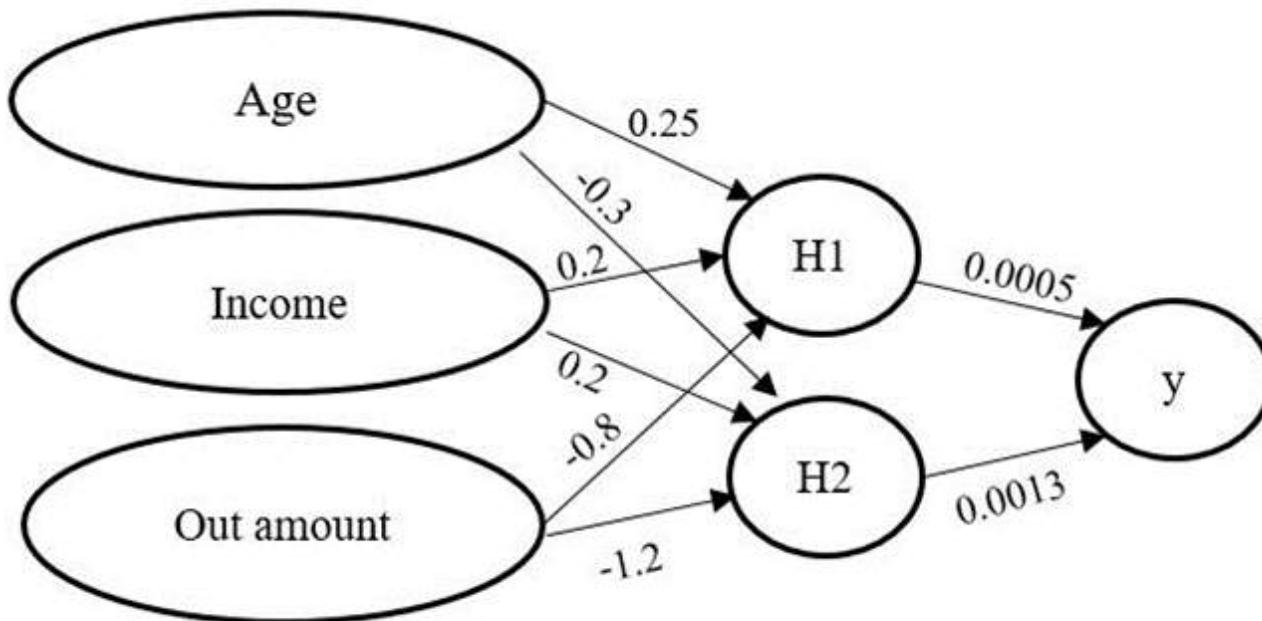
$$H_2 = \max(35 \times (-0.3) + 100,000 \times 0.2 - 20,000 \times 1.2, 0) = 0$$

Then applying the logistic function to the weighted sum of the inputs of the hidden layers, we get:

$$\hat{y} = \frac{1}{1 + e^{-0.0005 \times 4,008.75}} = 0.88.$$

Because the predicted probability is high, the risk that credit card holder defaults is quite high.

- C. Why is the logistic function adequate for the output layer in this context?



View Answer

This is a classification problem. The logistic function is appropriate because its output is between 0 and 1. We can interpret the output as the predicted probability of the outcome coded as 1. In this case, it is the probability of the borrower defaulting.

Chapter 5: Semi-supervised Learning

Learning Objectives

Semi-supervised learning is typically applied when data sets contain a mix of labeled and unlabeled data. This chapter presents the assumptions that must be satisfied for semi-supervised learning to be effective as well as illustrations of self-training and co-training, both popular methods of semi-supervised learning.

After completing this chapter, you should be able to:

- Explain how semi-supervised learning differs from unsupervised and supervised learning.
- Discuss the assumptions required for effective semi-supervised learning.
- Compare and contrast self-training and co-training methods of semi-supervised learning.

5.1 Introduction to Semi-supervised Learning

The previous chapters have contrasted unsupervised and supervised learning, explaining the different techniques employed for analysis in each case. In summary, unsupervised learning is used when the data has no labels, whereas supervised learning takes place when the data is labeled. But suppose that only some of the data is labelled – what could be done then? Two possible approaches are:

- Remove the labels and treat the whole dataset as if it is unlabeled, using appropriate techniques for unsupervised learning.
- Drop the observations that are not labeled, and then use supervised learning on the remainder.

Neither of these options would be ideal because they both imply throwing information away. A third option could be to try to identify labels for the unlabeled data, but the labels could either be impossible to obtain or very costly to collect.

As an example, suppose that a bank asks its customers to write brief comments on the service they have received after each significant transaction. The bank would like to classify the customers as “happy” or “unhappy” and then correlate these classes with customer-specific information such as their age, income, and educational attainment level, as well as characteristics related to the transaction such as the nature of the business (e.g., loan application, account deposit, etc.), amount of money involved, branch used, etc. This body of data would begin as unlabeled, because the comments would not have been classified, and creating labels would require a human to read through them individually and make a classification. Doing so could be infeasible if the dataset was large, and in other scenarios it might require subject matter expertise (e.g., providing a rating of company’s environmental performance based on a range of qualitative indicators). But in either scenario, it should be possible at least to determine appropriate labels for a small subset of, say, 30 customers.

This would create a “hybrid” dataset with some labeled and some unlabeled observations. We could still employ one of the two approaches described above, but there is also a third category of structure to consider, namely *semi-supervised learning*, sometimes also known as *weak supervision*. This is a combination of the two techniques: semi-supervised learning is a generic term used to describe the situation where a

dataset contains both labeled and unlabeled observations. It has achieved widespread applicability in areas as diverse as anomaly detection, image recognition, and website classification.

5.2.1 Semi-supervised Learning Assumptions

Invariably, semi-supervised learning is used in the context of classification rather than prediction scenarios. The technique makes use of the parallels between classification and clustering, and for it to work well several assumptions about the nature of the data need to hold:

- The “*clustering assumption*” – the unlabeled data fall naturally into separable clusters (locally dense regions in feature space).
- The “*smoothness assumption*” or “*continuity assumption*” – there is a smooth and continuous boundary separating the classes that can be used for deciding the classes of unlabeled instances.
- The “*manifold assumption*” –the observed datapoints in the high-dimensional feature space are often concentrated along lower dimensional substructures that are topological manifolds. A topological manifold is a topological space that locally resembles the Euclidian space \mathbb{R}^n .¹ A way to understand the manifold assumption is to think about a sphere (a three-dimensional object)

where all the datapoints are concentrated on the surface (a two-dimensional object). The surface of a sphere is a two-dimensional manifold embedded in a three-dimensional space. The manifold assumption states that the input space is composed of multiple manifolds on which all the datapoints lie and all the datapoints in the same manifold belong to the same class.

These assumptions are discussed further in Appendix 5.A. The assumptions imply that the clusters in the unlabeled data map naturally onto the classifications in the labeled data. An example situation would be where a bank was interested in developing a model to predict defaults among mortgage borrowers and had a dataset with two subsets: one with labels (where default had occurred or not occurred), and another without any labels where there is only information relating to the mortgage (e.g., loan amount, collateral, loan term, borrower characteristics, etc.) but the bank does not know whether those customers defaulted or not.

The bank might build a classification model (e.g., using logit or SVM) on the labeled part of the dataset, and a clustering model for the unlabeled part of the dataset (e.g., using k-means). Semi-supervised learning would work best if the clusters that formed in the unlabeled data naturally captured the same characteristics as the classification of the labeled data (e.g., two clusters formed: one with low income, high borrowing, low collateral and the other with the reverse of those characteristics). If the clusters and classifications separate the features in very different ways, the additional benefit from employing the unlabeled data to bolster the labeled data is much diminished.

This link between classification and clustering provides a foundation for how semi-supervised learning works: specifically, the assumption is that if a set of instances are clustered closely together, they would likely share the same label if they were labeled. On the other hand, points far apart in feature space are less similar and therefore less likely to share a label.

¹ Technically speaking, an n-dimensional topological manifold is a topological space such that each point has a neighbourhood that is homeomorphic to an open subset of n-dimensional Euclidean space. For instance, lines and circles are one-dimensional manifolds, planes and spheres are two dimensional manifolds, etc.

5.2.2 Semi-supervised Learning Techniques

There are several practical approaches to combining the labeled and unlabeled data, but the highest-level categorization of the available techniques is as transductive or inductive. **Transductive** methods do not aim to build a generalizable model and are therefore sometimes considered to arise from a “closed world view”. In this case, because there is no model, the objective is solely to identify labels for the unlabeled data already observed. All instances need to be specified at the time of conducting the analysis, and no new instances can be incorporated into the study and classified at a later stage, so there is no separate test data. One transductive technique is label propagation, which is a graphical technique that assigns labels to unlabeled instances based on how close they are to labeled data points using a metric such as the Euclidean distance applied to the features.

Inductive methods, on the other hand, involve building a model that links the features to the labels, and that can then be applied to other instances. Common inductive methods include self-training and co-training, each of which are described in the following subsections

5.2.3 Self-Training

Among all the inductive methods, *self-training* is perhaps the most popular, owing to its intuitiveness and simplicity. It is sometimes referred to as a heuristic technique, because it employs unlabeled data from a supervised perspective, using methods and models from the latter, rather than using both labeled and unlabeled data together in learning. Self-training is one of the family of “wrapper” methods, and works as follows:

1. Generate a classification model using any preferred technique (e.g., K nearest neighbors or logistic regression) applied to the labeled part of the data.
2. Apply the model generated in step 1 to all the unlabeled data and generate predicted labels for each instance in the unlabeled part of the data.
3. Select the single instance for which the model’s predicted label has the highest probability of being correct based on the probabilities output from logistic regression, neural networks etc.

4. Apply the predicted label to the instance selected at stage 3 and shift that datapoint from the unlabeled to the labeled portion of the dataset.
5. Return to stage 1 with the labeled set having now been enlarged by one observation and the unlabeled set reduced by one.
6. Repeat stages 1 to 5 until all the unlabeled data have been labeled, then stop and that would be the final classification model.

The labels applied to the previously unlabeled data are known as *pseudo-labels*. This approach is intuitive and can work with many supervised training models. However, it has a couple of disadvantages.

- It is very computationally intensive because the model is retrained as many times as there are instances in the unlabeled data. If computational resources are constrained, this problem can be mitigated by selecting the best predicted k observations at stage 3 (where k is a positive integer) and shifting all k observations, along with their predicted labels, in stage 4. For instance, if $k = 10$, this will reduce by tenfold the number of rounds of training required.
- Retraining the model after the addition of each individual datapoint can result in severe overfitting. Overfitting can be guarded against by a process known as co-training, discussed in the next sub-section.

The process of self-training is demonstrated visually in [**Figure 5.1**](#).

Features

Target

| Observation | Feature 1 | Feature 2 | Feature 3 | Feature 4 | Target |
|-------------|-----------|-----------|-----------|-----------|--------------------------------------|
| 1 | X(1,1) | X(1,2) | X(1,3) | X(1,4) | Y(1) Y(2) Y(3) Y(4) Y(5) |
| 2 | X(2,1) | X(2,2) | X(2,3) | X(2,4) | |
| 3 | X(3,1) | X(3,2) | X(3,3) | X(3,4) | |
| 4 | X(4,1) | X(4,2) | X(4,3) | X(4,4) | |
| 5 | X(5,1) | X(5,2) | X(5,3) | X(5,4) | |
| 6 | X(6,1) | X(6,2) | X(6,3) | X(6,4) | |
| 7 | X(7,1) | X(7,2) | X(7,3) | X(7,4) | |
| 8 | X(8,1) | X(8,2) | X(8,3) | X(8,4) | |
| 9 | X(9,1) | X(9,2) | X(9,3) | X(9,4) | |
| 10 | X(10,1) | X(10,2) | X(10,3) | X(10,4) | |

Labeled

Fit r
w
lab
d

Unlabeled

Figure 5.1 Illustration of Self-Training

Box 5.1: An Example of Self-Training

To illustrate how self-training works, let's revisit the hypothetical sample of borrowers discussed in Chapters 3 and 4. Suppose we have four additional unlabeled data points (Borrowers 11 to 14) where the borrowers' default status is unknown, as shown in [Table 5.1](#). In this example, we employ logistic regression for classifying the unlabeled borrowers in two classes: Default and No Default.

Table 5.1 Initial Data for Borrowers (Standardized)

| Borrower # | Default | Balance | Income |
|------------|---------|---------|--------|
| 1 | 0 | -0.55 | -0.75 |
| 2 | 0 | -0.23 | 0.19 |
| 3 | 0 | -0.76 | 0.53 |
| 4 | 0 | -0.08 | -0.73 |
| 5 | 0 | -0.72 | 1.32 |
| 6 | 0 | -0.79 | 2 |
| 7 | 0 | -0.8 | -0.34 |
| 8 | 1 | 0.46 | -0.71 |
| 9 | 1 | 1.95 | -0.96 |

| Borrower # | Default | Balance | Income |
|------------|---------|---------|--------|
| 10 | 1 | 1.51 | -0.54 |
| 11 | ? | -1.95 | 0.83 |
| 12 | ? | -0.03 | 0.05 |
| 13 | ? | 1.99 | -1.04 |
| 14 | ? | -0.01 | -0.02 |

Step 1: Initially, we fit the logistic model using only the labeled data (Borrowers 1 to 10).

$$P(i) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \text{Feature } A_i + \beta_2 \text{Feature } B_i)}} \quad (5.1)$$

Step 2: Utilizing this trained model, we predict outcomes, the probability of default, and the probability of non-default for the unlabeled data. The predicted probabilities are shown in [Table 5.2](#). Based on the results, the 13th borrower has the “clearest split” between the two categories, i.e., the biggest difference in the predicted probabilities.

Table 5.2 Predicted Probabilities for Unlabeled observations after First regression

| Borrower # | Probability of non-default | Probability of default | Predicted outcome |
|------------|----------------------------|--------------------------|-------------------|
| 11 | 1.0000 | 8.0819×10^{-29} | 0 |
| 12 | 0.9934 | 6.6148×10^{-3} | 0 |

| Borrower # | Probability of non-default | Probability of default | Predicted outcome |
|------------|----------------------------|------------------------|-------------------|
| 13 | 0.0000 | 1.0000 | 1 |
| 14 | 0.9896 | 0.0104 | 0 |

Step 3: We include the 13th observation in the labeled sample and label it as a defaulted loan (1). Subsequently, we repeat the process using the updated data to fit the logistic regression model and predict the remaining unlabeled data. The predicted probabilities in this step are shown in [Table 5.3](#). This time the 11th borrower has the biggest difference in the predicted probabilities. Hence, we include this observation into the labeled sample and label it as a non-defaulted loan (0).

Table 5.3 Predicted Probabilities for Unlabeled observations after Second Regression

| Borrower # | Probability of non-default | Probability of default | Predicted outcome |
|------------|----------------------------|--------------------------|-------------------|
| 11 | 1.0000 | 2.1989×10^{-28} | 0 |
| 12 | 0.9909 | 9.0841×10^{-3} | 0 |
| 14 | 0.9860 | 0.0140 | 0 |

After including the 11th observation into the labeled sample, we run the logistic regression again to predict the remaining two unlabeled observations. It can be seen in [Table 5.4](#) that the 12th borrower has a bigger difference in the predicted probabilities than the 14th borrower. Accordingly, the corresponding observation is moved to the labeled sample.

Table 5.4 Predicted Probabilities for Unlabeled observations after Third Regression

| Borrower # | Probability of non-default | Probability of default | Predicted outcome |
|------------|----------------------------|-------------------------|-------------------|
| 12 | 0.9957 | 4.3539×10^{-3} | 0 |
| 14 | 0.9929 | 7.1020×10^{-3} | 0 |

Finally, we run the logistic regression again using these 13 labeled observations and use this model to predict default class for the 14th borrower. It is classified as a non-defaulted loan. The fully labeled data is shown in [Table 5.5](#)

Table 5.5 Fully Labeled Data after the Fourth Regression

| Borrower # | Default | Balance | Income |
|------------|---------|---------|--------|
| 1 | 0 | -0.55 | -0.75 |
| 2 | 0 | -0.23 | 0.19 |
| 3 | 0 | -0.76 | 0.53 |
| 4 | 0 | -0.08 | -0.73 |
| 5 | 0 | -0.72 | 1.32 |
| 6 | 0 | -0.79 | 2 |
| 7 | 0 | -0.8 | -0.34 |
| 8 | 1 | 0.46 | -0.71 |
| 9 | 1 | 1.95 | -0.96 |

| Borrower # | Default | Balance | Income |
|------------|---------|---------|--------|
| 10 | 1 | 1.51 | -0.54 |
| 11 | 0 | -1.95 | 0.83 |
| 12 | 0 | -0.03 | 0.05 |
| 13 | 1 | 1.99 | -1.04 |
| 14 | 0 | -0.01 | -0.02 |

5.2.4 Co-Training

Co-training is another useful method that can be applied when we have two different “views” of an example. For instance, in building a credit risk model, a borrower’s financial information (income, assets, liabilities etc.) and non-financial information (marital status, tenure of employment, education level etc.) are considered. In this case, the financial and non-financial features can be considered as two different “views” of a borrower. Both are important descriptions of the borrower and provide complementary information. Co-training can utilize both “views” to build two classifiers that teach each other on unlabeled data.

Let us divide the feature set x into two disjoint subsets² x_A and x_B representing two different views of the dataset. Co-training assumes that either x_A and x_B are individually sufficient for learning if we have enough labeled data, and thus classifiers can be built for each of them. A wrapper method like the one described in section 5.2.3 can be used to augment the labeled dataset step by step. The detailed steps are described as follows:

1. Split feature set x into two disjoint subsets, x_A and x_B , corresponding to two different views A and B, both for the labeled and unlabeled data.
2. Generate classification models (Model A and Model B) for the two feature sets of the labeled data.
3. Apply the models generated in step 2 to the two unlabeled subsets of data and generate predicted labels for each instance in the unlabeled subsets.
4. Select the predicted observation from the unlabeled subset for each model with the highest probability score (e.g., logistic regression, neural networks with softmax layer, etc.).
5. Assign the predicted labels to the instances selected at stage 4 and shift those data points from the unlabeled to the labeled sets. (The key difference with co-training compared with self-training is that the data points move to the labeled dataset of the other feature subset.) So, the best predicted instance from unlabeled subset A moves to the labeled subset B and vice-versa.

6. Return to stage 2 with the labeled sets having now been enlarged by one observation each and the unlabeled sets reduced by one each.
7. Repeat stages 2 to 6 until all the unlabeled data have been labeled, then stop and those would be the final classification models, one for each of the two disjoint sets.
8. Estimate a single supervised model that reunites the two subsets A and B now that all instances have been labeled.

Because the co-training technique uses different subsets of features to build two different models to augment the training set, it reduces the risk of overfitting. Co-training is sometimes referred to as a disagreement-based method, because it exploits differences in the predictions based on the two subsets of features to improve the training classifications of both as they learn from one other. Co-training is demonstrated visually in [**Figure 5.2**](#).

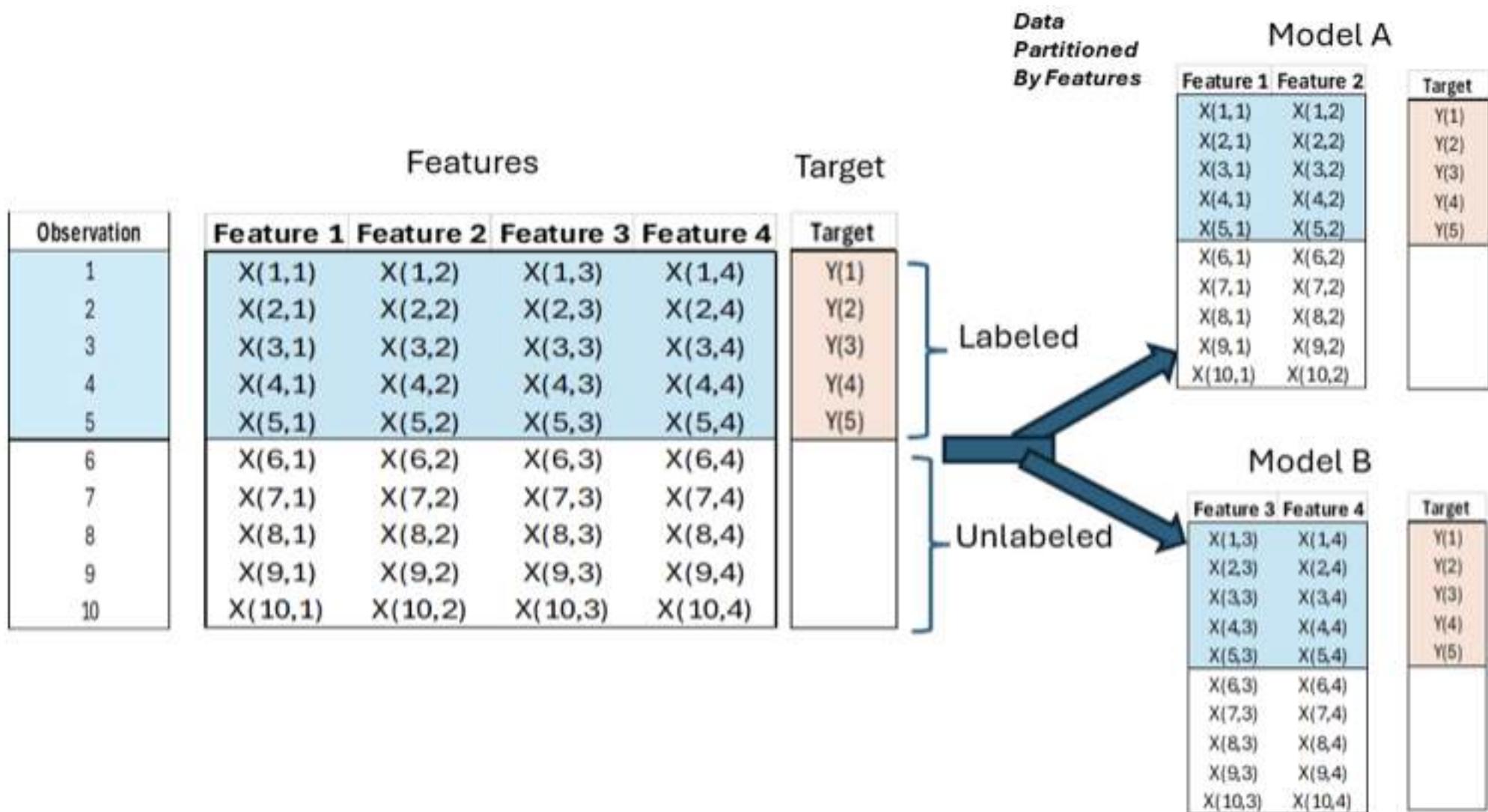


Figure 5.2 Illustration of Co-Training

Box 5.2: An Example of Co-Training

To demonstrate the co-training process, we use the same data used in the self-training example³. Once again logistic regression is used as the classification model. Initially, we separate the dataset into labeled and unlabeled samples. Subsequently, we split the features into two distinct groups, denoted as "Group A" and "Group B." The split data is depicted in [Table 5.6](#).

Table 5.6 Data split into two groups based on features

| Group A | | | Group B | | |
|------------|---------|-------------------|------------|---------|------------------|
| Borrower # | Default | Feature A balance | Borrower # | Default | Feature B income |
| 1 | 0 | -0.55 | 1 | 0 | -0.75 |
| 2 | 0 | -0.23 | 2 | 0 | 0.19 |
| 3 | 0 | -0.76 | 3 | 0 | 0.53 |
| 4 | 0 | -0.08 | 4 | 0 | -0.73 |
| 5 | 0 | -0.72 | 5 | 0 | 1.32 |
| 6 | 0 | -0.79 | 6 | 0 | 2 |
| 7 | 0 | -0.8 | 7 | 0 | -0.34 |
| 8 | 1 | 0.46 | 8 | 1 | -0.71 |

| Group A | | | Group B | | |
|------------|---------|-------------------|------------|---------|------------------|
| Borrower # | Default | Feature A balance | Borrower # | Default | Feature B income |
| 9 | 1 | 1.95 | 9 | 1 | -0.96 |
| 10 | 1 | 1.51 | 10 | 1 | -0.54 |
| 11 | ? | -1.95 | 11 | ? | 0.83 |
| 12 | ? | -0.03 | 12 | ? | 0.05 |
| 13 | ? | 1.99 | 13 | ? | -1.04 |
| 14 | ? | -0.01 | 14 | ? | -0.02 |

Two logistic regression models (A and B) are trained independently using labeled data, focusing on each feature group separately.

$$P(i) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \text{Feature } A_i)}} \quad (5.2a)$$

$$P(i) = \frac{1}{1 + e^{-(\beta_0 + \beta_2 \text{Feature } B_i)}} \quad (5.2b)$$

These models are then employed to predict the labels of the unlabeled observations 11 through 14. The predicted probabilities for the unlabeled observations are shown in [Table 5.7](#).

Table 5.7 Predicted Probabilities for Unlabeled Observations after First Regression

Group A

| Borrower # | Probability of non-default | Probability of default | Predicted outcome |
|------------|----------------------------|--------------------------|-------------------|
| 11 | 1.0000 | 1.3211×10^{-30} | 0 |
| 12 | 0.9991 | 9.1305×10^{-4} | 0 |
| 13 | 0.0000 | 1.0000 | 1 |
| 14 | 0.9983 | 0.0174 | 0 |

Group B

| Borrower # | Probability of Non-default | Probability of default | Predicted outcome |
|------------|----------------------------|-------------------------|-------------------|
| 11 | 0.9989 | 1.1419×10^{-3} | 0 |
| 12 | 0.9635 | 0.0365 | 0 |
| 13 | 0.1655 | 0.8346 | 1 |
| 14 | 0.9507 | 0.0493 | 0 |

Based on the predictions from the two logistic regression models, the 13th borrower has the biggest difference in the probabilities in model A, whereas the 11th borrower has the biggest difference in the probabilities in model B. Consequently, the 13th observation is moved to group B and labeled as having defaulted, whereas the 11th observation is moved to group A with a no-default label.

Utilizing the updated labeled data, we repeat the process of fitting logistic regression models and predicting the labels of the remaining unlabeled observations 12 and 14. The predicted probabilities in this run are shown in [Table 5.8](#).

Table 5.8 Predicted Probabilities for Unlabeled Observations after Second Regression

Group A

| Borrower # | Probability of non-default | Probability of default | Predicted outcome |
|------------|----------------------------|-------------------------|-------------------|
| 12 | 0.9993 | 7.3523×10^{-3} | 0 |
| 14 | 0.9986 | 0.0143 | 0 |

Group B

| Borrower # | Probability of non-default | Probability of default | Predicted outcome |
|------------|----------------------------|------------------------|-------------------|
| 12 | 0.8944 | 0.1057 | 0 |
| 14 | 0.8784 | 0.1216 | 0 |

Borrower #12 is classified consistently across both models as having the biggest difference in the predicted probabilities. So, this observation will be incorporated in both Group A and Group B. This leaves the 14th observation as the only unlabeled observation. Finally, we re-run the two logistic models to classify the 14th borrower. The predicted probabilities are shown in [Table 5.9](#), which clearly shows that borrower falls in the non-default class.

Table 5.9 Predicted Probabilities for Unlabeled Observations after Second Regression

Group A

| Borrower # | Probability of non-default | Probability of default | Predicted outcome |
|-------------------|-----------------------------------|-------------------------------|--------------------------|
| 14 | 0.9997 | 2.9574×10^{-4} | 0 |

Group B

| Borrower # | Probability of non-default | Probability of default | Predicted outcome |
|-------------------|-----------------------------------|-------------------------------|--------------------------|
| 14 | 0.9738 | 0.0262 | 0 |

The fully labeled data is shown in [Table 5.10](#).

Table 5.10 Fully Labeled Data

| Borrower # | Default | Balance | Income |
|-------------------|----------------|----------------|---------------|
| 1 | 0 | -0.55 | -0.75 |
| 2 | 0 | -0.23 | 0.19 |
| 3 | 0 | -0.76 | 0.53 |
| 4 | 0 | -0.08 | -0.73 |
| 5 | 0 | -0.72 | 1.32 |
| 6 | 0 | -0.79 | 2 |
| 7 | 0 | -0.8 | -0.34 |
| 8 | 1 | 0.46 | -0.71 |
| 9 | 1 | 1.95 | -0.96 |

| Borrower # | Default | Balance | Income |
|------------|---------|---------|--------|
| 10 | 1 | 1.51 | -0.54 |
| 11 | 0 | -1.95 | 0.83 |
| 12 | 0 | -0.03 | 0.05 |
| 13 | 1 | 1.99 | -1.04 |
| 14 | 0 | -0.01 | -0.02 |

² Two sets are disjoint if there is no element common between them.

³ Please see Box 5.1

5.2.5 Unsupervised Pre-processing

Unsupervised pre-processing involves working with the unlabeled data portion first before dealing with the labeled data in a subsequent stage. Within this family of approaches, there are at least three possibilities:

- **Feature extraction** – this involves employing techniques such as principal components analysis or autoencoders, discussed in previous chapters, to reduce the dimensionality of the unlabeled data and to represent it more efficiently.
- “*Cluster-then-label*” – as the name suggests, the combined unlabeled + labeled datasets are subjected to a clustering algorithm such as k -means, and then

the resulting clusters are used to train a classifier model. If most of the labeled instances with a given label appear in the same cluster, then that label is assigned to all the unlabeled points in the same cluster. This is another example of pseudo-labeling. Alternatively, a supervised learning model can be built on the clusters rather than the individual labeled data points.

- Pre-training – here, the unlabeled data are formed into clusters that are useful to develop preliminary decision boundaries prior to applying supervised learning.

Appendix 5.A Semi-supervised Learning – Assumptions

The success of semi-supervised learning relies on the following assumptions about the data: (i) clustering assumption, (ii) smoothness or continuity assumption, and (iii) manifold assumption. These assumptions are discussed below.

Figure 5.A.1 provides a pictorial representation of the first two assumptions. In panel (a) on the left, the data belong to two classes, but fail to form clusters, and thus the clustering assumption is violated. The data points are dispersed in the feature space and do not form dense regions that can be separated. In panel (b) on the right, the data form clusters, but these are partly overlapping and cannot be separated using a line, and thus the smoothness assumption is violated.

Figure 5.A.2 shows the difference between a manifold and a non-manifold. The cubic box can be unfolded on a 2-dimensional space, and therefore each of its faces is a two-dimensional manifold. However, we will not be able to unfold the two cubes linked only by one edge represented in panel (b) on the right in the same way as we did for the cube in panel (a) on the left. In fact, the figure in panel (b) is a non-manifold structure. Notably, if each face of the cube in panel (a) contained datapoints that belong to a single class, the manifold assumption would be satisfied. However, if one face contained datapoints belonging to different classes, the assumption would be violated as the datapoints in a manifold would not belong to the same class.

(a) Failure of the clustering assumption

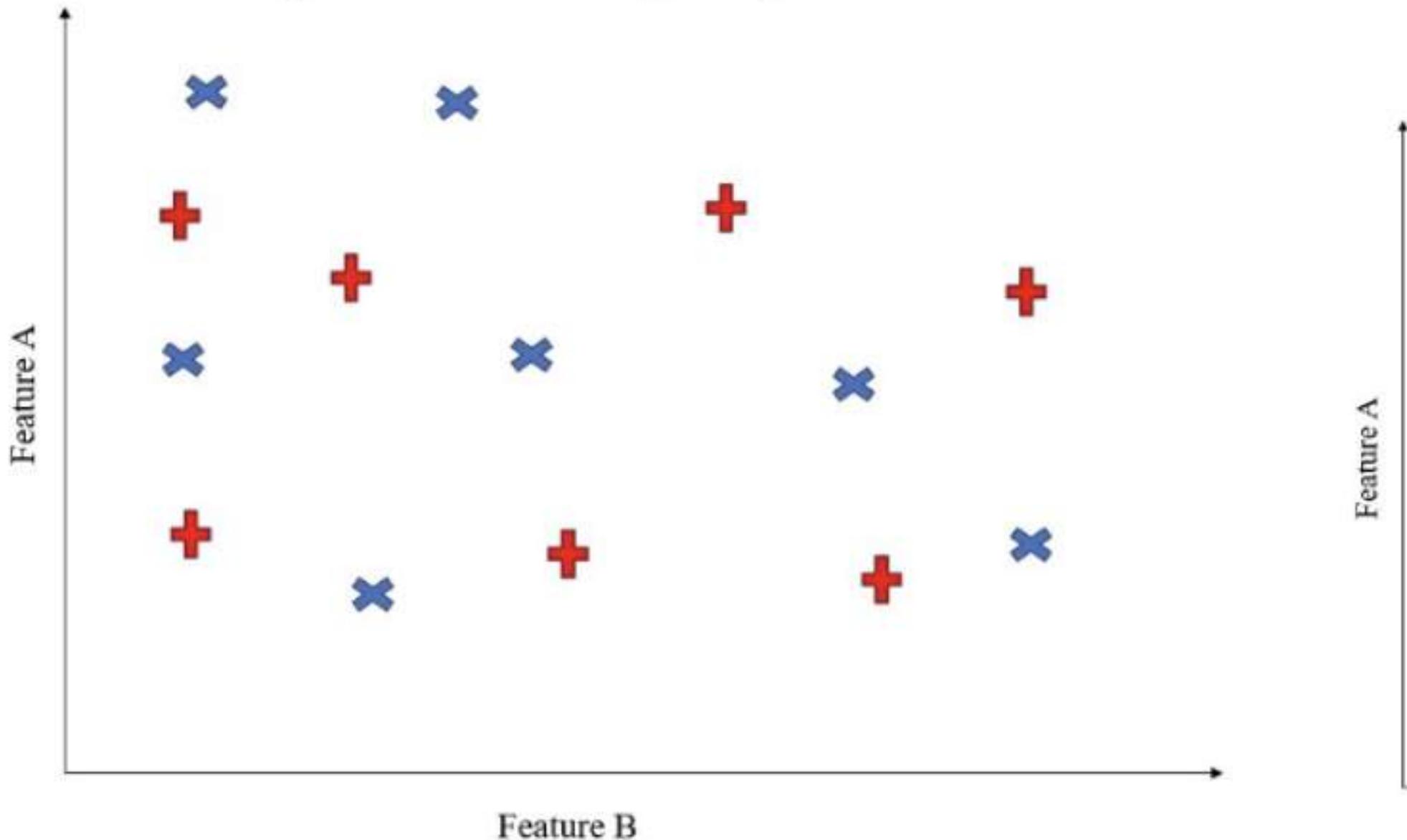


Figure 5.A.1 Violations of clustering and smoothness assumptions.

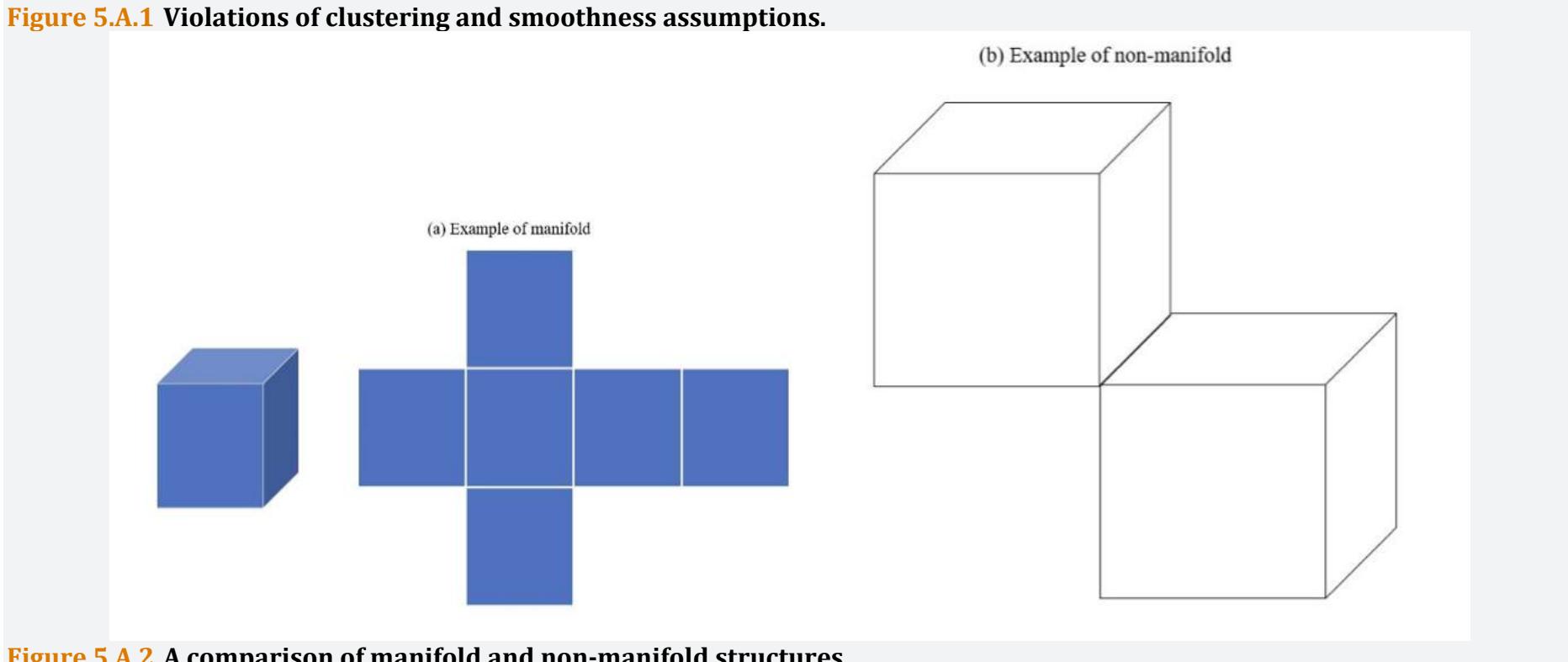


Figure 5.A.2 A comparison of manifold and non-manifold structures.

Semi-Supervised Learning: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

5.1

- A. When should we consider using semi-supervised machine learning?

Semi-supervised learning is most valuable when the dataset contains both labeled and unlabeled instances, but where the labeled proportion is relatively small and where it is infeasible or excessively costly to manually add labels to the currently unlabeled instances.

B. How does self-training differ from co-training in semi-supervised machine learning?

Self-training and co-training are both methods for applying pseudo-labels to the unlabeled part of the data in semi-supervised machine learning. With self-training, once the labeled data have been classified using a supervised machine learning technique, the unlabeled data are assigned labels one at a time and added to the labeled portion of the data. Co-training adopts similar principles, but it involves splitting the features of the labeled and unlabeled data into two separate sub-datasets. Then, two separate classification models are built, and the most confident outcomes generated for each group of features become labeled instances for the other group of features.

Chapter 6: Reinforcement Learning

Learning Objectives

This chapter introduces reinforcement learning, a machine learning technique that applies a trial-and-error feedback loop to train models to optimize actions that maximize a defined long-term or cumulative reward. The “output” from reinforcement learning applications is a recommended action based on defined parameters rather than a prediction, classification, or cluster produced in unsupervised or supervised learning applications.

After completing this chapter, you should be able to:

- Explain key principles and frameworks behind reinforcement learning.
- Compare and contrast exploration, exploitation, and ϵ -greedy strategies.
- Describe reinforcement learning in the context of the Multi Armed Bandit (MAB) problem.

- Explain Markov decision processes.
- Differentiate between the Monte Carlo and Temporal Difference methods.
- Describe how neural networks can be used in reinforcement learning.

6.1 The Principles of Reinforcement Learning

Reinforcement learning is concerned with developing a policy for a series of decisions to maximize a long-term reward. In reinforcement learning, the learner is presented with *feedback* on the quality of the reward in a process analogous to trial-and-error. The technique is advantageous when decisions need to be made repeatedly so that the algorithm can learn based on the rewards or sanctions received in previous rounds. Unlike both unsupervised and supervised learning, the “output” from reinforcement learning applications is a recommended action given the circumstances, rather than a prediction, classification, or cluster.

A typical example of a situation where reinforcement learning can be applied would be a video game in which players can hone their strategies based on whether they won or lost (and by how much) in prior turns. A further frequently employed example is that of teaching a dog to do tricks. Games and dog training represent good examples of reinforcement learning because each player or dog behaves somewhat differently, and therefore a fixed set of universal instructions covering all steps in the process cannot be developed. For these examples, it is easy to define “success” (Did we win the game? Did the dog perform the tricks?) but hard to specify *a priori* what is the appropriate action in every situation. Machines using reinforcement learning emulate the way that people and animals learn from experience to improve their future performance at the task.

The technique has had some very successful applications. For example, reinforcement learning can produce algorithms that play board games such as Chess and Go better than the most skilled humans. The algorithm learns by playing against itself many times and using a systematic trial-and-error approach. Other recent applications include controlling movements in robots, self-driving cars, traffic light control, and inventory management. There are also many potential uses of reinforcement learning in finance, including for technical trading rules, determining how to split large-volume trades to sell quickly while minimizing the adverse price effect, and determining how much of a position to hedge using derivatives.

A disadvantage of reinforcement learning algorithms is that they tend to require larger amounts of training data than other machine-learning approaches. By construction, a machine learning application using such a technique will initially make many errors and perform poorly, but it should improve considerably with practice.

6.2 The Multi-arm Bandit Problem

Let us start with a simple and commonly used reinforcement learning problem known as the multi-arm bandit (MAB). The MAB problem involves a gambler (agent) who can choose to play one of several different slot machines. The gambler believes that the machines have different probabilities of winning, but is unsure which machine is more generous than the others. The gambler wants to maximize the total payout from a fixed number of rounds. The problem is straightforward because:

- In each trial, the gambler picks only one machine, and the outcome (reward) is either that they win (i.e., they get paid one of several possible payout levels), or they lose. Therefore, each machine has its own probability distribution of rewards, and each round has only one step (i.e., one action, one state, and one reward).
- There is no other gambler/agent involved, i.e., it is a single agent framework.
- The gambler's action in the current round does not affect the states in subsequent rounds. Therefore, the current action only affects the current reward, not future rewards.

6.2.1 Terminology in MABs

The terminology used in reinforcement learning is quite specific to these techniques and therefore it is important to clarify it before we set up the model. Such models are defined in terms of states, actions, and rewards. The states define the environment (the slot machines in each round), an action is the decision taken (decision on which slot machine to play), and rewards are the goal of the problem (payout from the chosen slot machine). The

aim is to choose the decision that maximizes the value of total subsequent rewards that are earned, possibly applying a discount rate to the rewards. We introduce the following terms commonly used when describing a reinforcement learning problem, along with the corresponding meaning in the MAB problem for better understanding:

- *The agent (The gambler)* – the person or algorithm making the decision. Usually, there is a single agent, although in some models it is possible that there is more than one agent, in which case the agents could be working together or in competition. In the MAB problem, the gambler is the agent of the game.
- *Actions, A (Decision on which slot machine to play)* – these are the possible choices that the agent can select from at each timestep. In MAB, the gambler is free to choose which slot machine to play.
- *State, S (The slot machines)* – the circumstances, or a description of the environment, in which the decision is being made at each timestep. Because it is assumed that our actions do not change the slot machines in any way and we are always playing only those machines, there is only one single state for our slot machines, and it does not change.
- *Reward, R (The payout from the slot machine)* – the feedback that the agent receives based on its previous action. Note that this could be either positive or negative (a reward or a sanction/penalty, respectively).
- *Expected Future Rewards, G (The expected payout from the slot machine in the future)* – the expected value of future rewards. The objective is usually to maximize G . In our MAB case, the objective is to maximize the payout we get in the future, whenever we choose a slot machine.
- *Policy, Π (The plan of action on which slot machine to play)* – this is the plan of action that the agent takes based on observing the current state. The policy maps the states to actions that will maximize the reward. Because there is only one single state in MAB problems, we only need to consider the policy in this state.
- **Value function, $V(\cdot)$ (Measures how good a state is)** – This is also known as the state-value function. It relates the expected reward to a given state. It measures how good a state is. Because there is only one state in MABs, it is not relevant in this case, but it can be very useful for other problems with many states to guide the agent on policy improvement.
- **Action-value function, $Q(\cdot)$ (Measures how good an action is, given a certain state, like our slot machines)** – this relates the expected reward to the actions and the state. It measures how good an action is, given a certain state. In the MAB context, it measures how rewarding it is if we choose a certain slot machine to

play. This is very important and useful for us to compare the different actions and use them to optimize our policy.

The relationship between the agent and the environment is depicted in [Figure 6.1](#), and [Figure 6.2](#) shows the linkages between the state, policy, agent, and action.

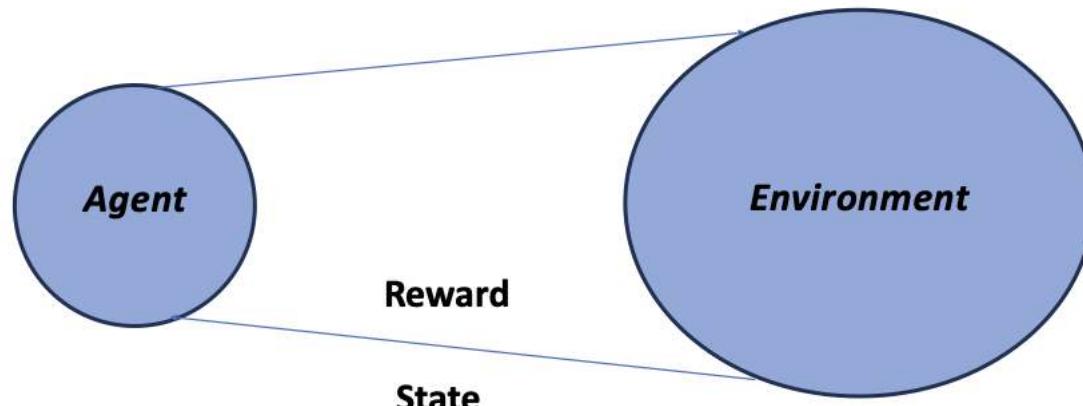


Figure 6.1 The Agent and the Environment

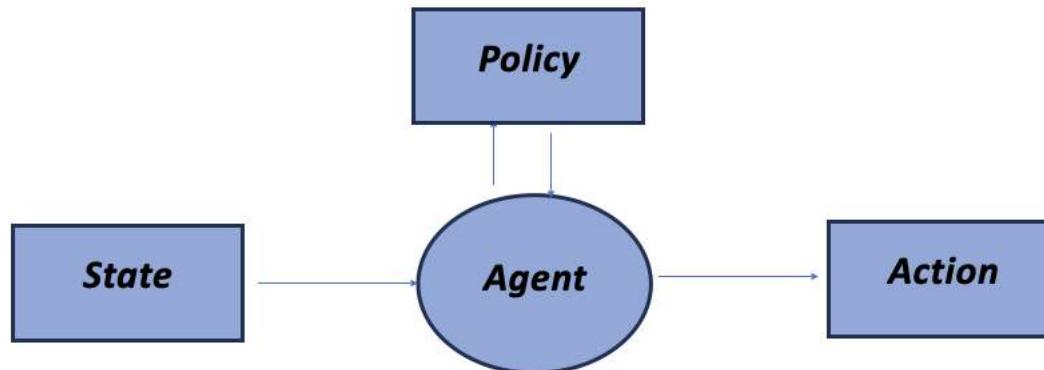


Figure 6.2 State, Policy, and Action

The capital letters S and A are used to denote the set of states and the set of actions in general, whereas their lower-case counterparts denote specific states or actions. Time subscripts are generally suppressed unless they are specifically required for clarity, such as when describing the transition from the state in one period to the next.

6.2.2 Strategy in MAB

After defining the terminologies used in MAB, as well as in a more general reinforcement learning context, it is time to think about a strategy to find a good policy that maximizes the total payout from a fixed number of rounds. A very intuitive way of playing is to always choose the best actions identified so far, which is called a greedy strategy.

1. Greedy strategy (exploitation)

The greedy strategy is a simple strategy in which the agent always chooses the actions with the best rewards seen so far. In the MAB problem, it means that we always choose the slot machine that gives us the best payout, in a “greedy” way. This strategy focuses on the idea of exploitation of the information gained from the agent’s experience so far. It may appear like a good strategy, but it has problems. If we find one slot machine that seems to pay out well and stick with it, it may produce a suboptimal result because we did not experiment with other slot machines, which may be better.¹ So, we should explore other possible actions. An extreme case would be random selection.

2. Random strategy (exploration)

The random strategy is an intuitive strategy where we randomly select a slot machine to play. Whereas the greedy strategy only chooses the action with the best payout up to that point, the random strategy is useful for exploring other possible actions. But it does not exploit the knowledge gained from past rewards to make more informed decisions over time.

We can see from the preceding two strategies that neither exploitation nor exploration alone are promising in the MAB problem. To address the shortcomings of a pure exploitation or pure exploration strategy, a combination of the two, called ϵ -greedy, was developed.

3. ϵ -greedy strategy (exploitation & exploration)

The ε -greedy strategy combines greedy exploitation and random exploration. Here ε is a hyperparameter (with $0 < \varepsilon < 1$) that determines whether a random selection is made to explore, or a greedy selection is made to exploit. Usually, a random number between 0 and 1 is drawn. If that number is below ε , we explore by choosing a random slot machine, otherwise we choose the machine that had the best payoff up to that point. This helps us to continue exploiting the slot machines that have provided the best payouts up to that point, while still exploring other options.

Usually, a small value such as $\varepsilon = 0.05$ or 0.1 is used so that the agent mostly relies on existing accumulated knowledge, experimenting with new strategies relatively infrequently. Although the ε -greedy strategy does not use random selection, and it is more adaptive in the face of time-varying reward structures, it might still select an obviously suboptimal action in many random trials.

A refinement to the ε -greedy strategy is to allow ε to vary systematically throughout the exercise, so that it is initially larger, allowing a lot of experimentation while the amount of accumulated knowledge about the relationships between actions and rewards is low. Then the hyperparameter is gradually reduced as more information becomes known and the benefit of additional exploration is diminished because the algorithm has already learned more about the best strategy. A popular approach is to use a decay factor, β , and set $\varepsilon = \beta^{t-1}$, where t is the trial number and with $0 < \beta < 1$.

To illustrate how the algorithm works in practice, let us assume that there are three different slot machines (call them A, B, and C) and that the game can be played many times. The payoff from the i th machine is normally distributed with mean μ_i and a standard deviation of one. The μ_i are not known in advance. We assume that:

$$\mu_A = 0.8 \quad \mu_B = 0.5 \quad \mu_C = 0.7$$

and adopt a decay factor $\beta = 0.85$ for ε . In the first simulation, ε is equal to 1, as we know nothing about the slot machines, and we must explore. We choose machine A and receive a payoff equal to 1.2. At this point, the value of ε is 0.85. Therefore, we draw a random number between 0 and 1: if the number is below 0.85, we decide to explore; otherwise, we choose the most profitable machine so far. Suppose that by chance we draw 0.99, then we return to slot machine A. If the payoff is now 0.8, we will update our expected reward for slot machine A by averaging over the two outcomes: the expected reward for machine A ($Q(A)$) is now $(1.2 + 0.8)/2 = 1$. [Table 6.1](#) shows how the first 10 trials would work.

Table 6.1 The outcome of 10 trials in a multi-arm bandit problem.

| Trial | ϵ | Random number (0,1) | Decision | Choice | Payoff | Reward |
|-------|------------|---------------------|----------|--------|--------|--|
| 1 | 1 | 0.7 | Explore | A | 1.2 | $Q(A) = 1.2$ $Q(B) = 0$ $Q(C) = 0$ |
| 2 | 0.85 | 0.99 | Exploit | A | 0.8 | $Q(A) = 1$ $Q(B) = 0$ $Q(C) = 0$ |
| 3 | 0.72 | 0.65 | Explore | B | 1.7 | $Q(A) = 1$ $Q(B) = 1.7$ $Q(C) = 0$ |
| 4 | 0.61 | 0.5 | Explore | C | 0.6 | $Q(A) = 1$ $Q(B) = 1.7$ $Q(C) = 0.6$ |
| 5 | 0.52 | 0.7 | Exploit | B | -0.7 | $Q(A) = 1$ $Q(B) = 0.5$ $Q(C) = 0.6$ |
| 6 | 0.44 | 0.4 | Explore | C | 1 | $Q(A) = 1$ $Q(B) = 0.5$ |

| Trial | ϵ | Random number (0,1) | Decision | Choice | Payoff | Reward |
|-------|------------|---------------------|----------|--------|--------|---|
| | | | | | | $Q(C) = 0.8$ |
| 7 | 0.38 | 0.8 | Exploit | A | -0.2 | $Q(A) = 0.6$ $Q(B) = 0.5$ $Q(C) = 0.8$ |
| 8 | 0.32 | 0.55 | Exploit | C | 0.5 | $Q(A) = 0.6$ $Q(B) = 0.5$ $Q(C) = 0.7$ |
| 9 | 0.27 | 0.25 | Explore | A | 1.2 | $Q(A) = 0.75$ $Q(B) = 0.5$ $Q(C) = 0.7$ |
| 10 | 0.23 | 0.45 | Exploit | A | 1 | $Q(A) = 0.8$ $Q(B) = 0.5$ $Q(C) = 0.7$ |

Although many other trials would be generally needed, if we had to stop the process at this stage the best strategy identified would be to play slot machine A because it has the highest average payout after 10 trials. It is easy to see that as the number of trials increases, progressively less exploration is conducted thanks to the decay factor. The parameter ϵ is close to 0 after about 57 trials, which implies that any new random number drawn will be greater than ϵ and exploration stops. We just play the machine with the highest expected reward.

In general, the new value of the action-value function, Q_k , for the k^{th} slot machine after it was chosen for n number of trials, is:

$$Q_k^n = \frac{1}{n} \sum_{j=1}^n R_j$$
(6.1)

Here, R_j is the reward for the j^{th} trial. It can be shown with simple algebra that each time slot machine k is chosen, its new Q -value will be a weighted average of its old Q -value and the new reward, with weights of $\frac{n-1}{n}$ and $\frac{1}{n}$ ¹, respectively:

$$Q_k^n = \frac{1}{n} \sum_{j=1}^n R_j = \frac{1}{n} \sum_{j=1}^{n-1} R_j + \frac{1}{n} R_n = \frac{n-1}{n} Q_k^{n-1} + \frac{1}{n} R_n = Q_k^{n-1} + \frac{1}{n} (R_n - Q_k^{n-1})$$
(6.2)

The MAB problem is clearly much simpler than the setup for most potential applications of reinforcement learning because there is only one state here and the slot machines do not change. It is possible for some reinforcement learning problems to have a changing environment with many states. One classic example of this is the Markov decision process.

¹ It is assumed here that the slot machines continue to payout indefinitely.

6.3 Markov Decision Processes

Markov decision processes (MDPs) are simple settings for environment dynamics. In this case the environment changes based on the actions of the agent. MDPs are processes that have no memory, which means that only the current state is relevant for determining the most appropriate current action and not any of the previous states.

MDPs are useful for modeling decision making in cases where the agent is not fully in control of the evolution of the states. The use of MDPs establishes a straightforward framework where there are m states, denoted s , each of which will occur with a given probability, and there is also a fixed probability of being in a particular state s_{t+1} at time $t + 1$ given that the state at time t was s_t , written mathematically as:

$$\text{Prob}(S_{t+1} = s_{t+1} \mid S_t = s_t)$$

The assumption that each state follows a Markov process greatly simplifies the analysis because such processes have no memory, as described earlier. We can express the Markov property as:

$$\text{Prob}(S_{t+1} = s_{t+1} \mid S_t = s_t, S_{t-1} = s_{t-1}, \dots, S_1 = s_1) = \text{Prob}(S_{t+1} = s_{t+1} \mid S_t = s_t) \quad (6.3)$$

That is, the future state at $t + 1$ is only dependent on the current state at t , and independent of past states at $t - 1, t - 2$, etc. We can then specify a transition probability matrix,² P , that shows the probabilities of moving from any state in one period to any other state in the next period. The probabilities of being in each state n timesteps into the future, given an initial state, are then simply given by the elements of P^n , i.e., the transition matrix P multiplied by itself n times. The Markov assumption therefore provides a simple way for the algorithm to determine how likely each future state is given the current state.

The agent will select an action at time t , $A_t, = a$, based on observing state $S_t = s$, and receives a reward, R_{t+1} in the next period at $t + 1$ as a function of the state $S_{t+1} = s'$ in that period and the action in the previous period:

$$R_{t+1} = f(s', a) \quad (6.4)$$

The agent's goal will be to select the policy that maximizes this expected return aggregated over all future time periods (noting that the current and previous returns are 'sunk costs' and hence are not included in the objective function). Denoting the discount factor by γ , with $0 < \gamma < 1$, we can define the goal at time t , G_t , as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \quad (6.5)$$

Here, the rewards, R , are discounted in the usual fashion and so are worth less the further they are received in the future, and they will depend on both the actions and the states in subsequent time periods. In other words, G_t is

the present value of future rewards. This summation could be finite, ending at time T , say, or it could be infinite. We could also rewrite the equation for G_t as:

$$G_t = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \dots) \quad (6.6)$$

The term in parentheses in this equation is simply G_{t+1} and so the expression for G_t could be further redrafted as a recursion:

$$G_t = R_{t+1} + \gamma G_{t+1} \quad (6.7)$$

Given the definition of G_t , we can define the state-value function V , which measures how good a certain state s , is following a certain policy Π :

$$V_\pi(s) = E_\pi[G_t | s] \quad (6.8)$$

We can also define the action-value function Q , which measures how good a certain action a is, in a state s , following a certain policy Π , as its expected return:

$$Q_\pi(s, a) = E_\pi[G_t | s, a] \quad (6.9)$$

Clearly, the agent's objective will be to choose the optimal policy Π^* , that maximizes Q , which we could write as:

$$\pi^* = \operatorname{argmax}_\pi Q_\pi(s, a) \quad (6.10)$$

Where argmax is the set of values of Π for which Q_Π is maximized.

² Please see Appendix 6.A for additional discussion on Markov transition probability matrices.

6.4 Approaches to Reinforcement Learning

Reinforcement learning algorithms can be classified as model-based and model-free algorithms. When there are a limited number of states, with well-defined actions and the transition probabilities are well defined, a dynamic programming technique can be used to obtain a solution. Typically, however, we only have partial information about the model. In such cases, the algorithms search for an optimal solution to maximize the reward. There are two different approaches to find the policy that maximizes Q. They are:

- a. Value-based approaches that work on maximizing the reward by determining the best action in each state. Several algorithms such as the Temporal Difference Method, Q-learning, SARSA, and Deep Q-learning belong to this category.
- b. Policy-based approaches that find the optimal policy to map a state into an action. The policy gradient approach is a commonly used policy-based algorithm.

We will focus on two commonly used value-based approaches in the next section.

6.5.1 The Bellman Equations

The Bellman equations provide a framework for evaluating policies. The equations can be written either in terms of values, V , or in terms of action-values, Q . They establish an updating mechanism, or in other words a recursive algorithm that sets the current value or action-value function for a given policy equal to the reward from the action a at time t , plus the discounted value function of future rewards at time $t + 1$. The equations make it possible to pinpoint actions that will lead to greater future values of R for each given state. The Bellman optimality equation for the value function V can be stated as:

$$V_t^*(s) = \max E[R_{t+1} + \gamma V_{t+1}(s')] \quad (6.11)$$

where s and s' are the states at t and $t + 1$ respectively.

The optimality equation for Q , the action-value function is:

$$Q_t^*(s, a) = \max E[R_{t+1} + \gamma V_{t+1}^*(s')] \quad (6.12a)$$

or, because $V_t^*(s) = \max Q_t^*(s, a)$

$$Q_t^*(s, a) = \max E[R_{t+1} + \gamma \max Q_{t+1}^*(s', a')] \quad (6.12b)$$

Here, a and a' are the actions taken at t and $t + 1$. Solving the Bellman equations gives the optimal policy, . If all the rewards and transition probabilities are known, then dynamic programming can be used to determine . But they are unlikely to be known in practice, in which case an iterative technique is required. There are two common ways to solve reinforcement learning problems iteratively: the Monte Carlo and Temporal Difference methods.

6.5.2 The Monte Carlo (MC) Method

In the Monte Carlo method, we conduct trials (termed *episodes*) repeatedly, assuming random initialization for each state and estimating the average reward for each state over all episodes for each policy . Through repeated trials, the algorithm develops an estimate of the expected value of acting A in state S . Usually, we perform trials until we meet a convergence criterion which depends on the problem domain or the environment. The resulting action-value function, $Q(s, a)$, is the value of taking action A in state S . In the Monte Carlo method, we directly work with $Q(s, a)$, the action-value function.

Suppose that the algorithm acts a in state s and the total subsequent discounted rewards prove to be G . Under the Monte Carlo method, $Q(s, a)$ is updated like equation (6.2). Instead of equally weighting the trials as in equation (6.2), we use exponential moving average for the trials and update as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[G_t - Q(s, a)] \quad (6.13)$$

where α is the smoothing factor that is chosen after some experimentation. The quantity α controls how much Q is updated at each iteration when a new reward value R is observed.

The Monte Carlo method for reinforcement learning has two main disadvantages. First, it can only be used for processes that have a finite horizon (in other words, the rewards do not continue into the indefinite future).

Second, convergence to the true reward for each state could be infeasibly slow, particularly for long episodes with many states.

6.5.3 The Temporal Difference (TD) Method

This technique begins by assuming that the agent is only aware of the states and possible actions that could be taken; the likelihood of each state and the transition probabilities are initially unknown. The objective is to estimate $V_t(S)$, but unlike the Monte Carlo method, there is no need to wait until the end of the episode before assigning the rewards. Instead, the temporal difference method focuses on the reward following a transition from the current state to the next state only. Consequently, the temporal difference method can be used for very long episodes or those with infinite lifetimes.

The temporal difference method combines the immediate reward and the discounted value of the next state. Suppose that at time t the algorithm acts a in state s and a reward of R is earned. It then moves to s' , where the value is $V_{t+1}(s')$. Under the temporal difference method, the value function is updated towards its target $R_{t+1} + \gamma V_{t+1}(s')$ as:

$$V_t(s) \leftarrow V_t(s) + \alpha[R_{t+1} + \gamma V_{t+1}(s') - V_t(s)] \quad (6.14)$$

The updating process is controlled by the hyperparameter α . Instead of updating the value function, we can work with Q , the action-value function, and update it as:

$$Q_t(s, a) \leftarrow Q_t(s, a) + \alpha[R_{t+1} + \gamma \max Q_{t+1}^*(s', a') - Q_t(s, a)] \quad (6.15)$$

This is called Q -learning.

6.5.4 Illustrative Example

To provide a simple example of reinforcement learning, suppose that there are four states and three actions, and that the current $Q(S, A)$ values are as indicated in [Table 6.2](#). Suppose that on the next trial, Action 3 is taken in State 4 and the total subsequent reward, G , is 1.0. If $\alpha = 0.05$, using the updating equation for the Monte Carlo method (equation (6.13)) would lead to $Q(4,3)$ being updated from 0.8 to:

$$0.8 + 0.05(1.0 - 0.8) = 0.81$$

Suppose we are going from State 4 in the current trial to State 3 in the next trial after taking Action 3. Suppose further that a reward, R , of 0.2 is earned between the two decisions. Assuming that γ is 0.9 and V is 1.0 in state 3, γV , the discounted value, is 0.9. Using the temporal difference method, if $\alpha = 0.05$, this would lead to $Q(4,3)$ being updated using the updating equation for temporal difference method (equation 6.15) from 0.8 to:

$$0.8 + 0.05(0.2 + 0.9 - 0.8) = 0.815$$

(A more detailed example is presented in Appendix 6.B.)

Table 6.2 Current Q values

| | State 1 | State 2 | State 3 | State 4 |
|----------|----------------|----------------|----------------|----------------|
| Action 1 | 0.1 | 0.2 | 0.4 | 0.2 |
| Action 2 | 0.8 | 0.3 | 0.5 | 0.1 |
| Action 3 | 0.3 | 0.7 | 0.9 | 0.8 |

6.5.5 Curse of Dimensionality and Neural Network Approximation

For the methods discussed so far in the chapter, we are still able to define the value function and action-value function as lookup tables, where the values are stored in a table corresponding to their states and actions. This is possible for simple reinforcement learning problems with a small number of actions and states. But it becomes impractical when the number of actions and states becomes very large, even infinite sometimes in problems such as self-driving cars, which would require an infinitely large table to store all possible actions and states. This also leads to an exponentially growing computational cost.

In many real-world applications, there are simply too many states, making it impractical to tabulate them all, so a different approach is required. In such circumstances, the task of the model will be to learn the value of each action as a function of the current state by averaging over many (but not all) possible future rewards arising after the action taken at this stage. Instead of calculating and then using the possibly infinitely large lookup table, we can estimate the Q value function for the given state and action, with a significantly smaller number of parameters we need to store compared to a tabular approach. Neural networks can be used to estimate the complete table from the observations that are available. This approach is called *deep reinforcement learning*. In this approach, the Q value function is estimated by training a neural network. The gradient descent method is used to estimate the weights of the network by minimizing a loss function which is the sum of the squared difference between the targeted and neural network's estimate of the Q values. Once the network is trained, it can be used to

generate Q values for any input set of state variables. This replaces the need to generate a very large lookup table for the Q values.

6.6 Chapter Summary

In this chapter, we started from a simple one-state reinforcement learning example using MAB, and introduced and formalized the basic concepts in a reinforcement learning problem setting with a discussion of how to optimize the policy with the ϵ -selection strategy. Then, we tackled harder problems such as MDPs with multiple states, introduced a dynamic programming solution if the exact model for state transition is known, and other sampling solutions such as Monte Carlo and Temporal Difference if we do not know the state transition dynamics. Finally, we discussed real-world level reinforcement learning problems with a very large, even infinite, number of states, where we have too many states for storage and calculations, which require using a neural network for approximating the value function or the action-value function.

Appendix 6.A Markov Transition Probabilities

Let us consider a simple scenario where there are three states, A, B and C, in a Markov decision process. The transition probability is the probability of moving from one state at time t to another state at the next period, $t + 1$. Typically, a transition probability matrix is used to summarize the probabilities. In this case the probability matrix is:

$$\Pi(t, t+1) = \begin{pmatrix} p_{aa} & p_{ab} & p_{ac} \\ p_{ba} & p_{bb} & p_{bc} \\ p_{ca} & p_{cb} & p_{cc} \end{pmatrix} \quad (6.A.1)$$

In the above matrix, the first element of the first row, p_{aa} , is the probability of remaining in state A from time say t to $t + 1$. The next element, p_{ab} , is the probability of moving from state A at t to state B at $t + 1$. It is followed by p_{ac} , which is the probability of moving from state A at t to state C at $t + 1$. The one-period movements between states are illustrated in Figure 6A.1.

Using this matrix, we can calculate the probabilities of moving from one state to another state in two time periods, e.g., $t = 0$ to $t = 2$. This requires the multiplication of the transition probability matrix by itself once:

$$\Pi(0,2) = \Pi(0,1)\Pi(0,1) \quad (6.A.2)$$

In general, the n step transition probabilities can be calculated as:

$$\Pi(0,n) = \Pi(0,1)\Pi(0,1) \dots \dots \quad \Pi(0,1) = \Pi^n(0,1) \quad (6.A.3)$$

As an example, consider the transition probability matrix below:

$$\Pi = \begin{pmatrix} 0.9 & 0.09 & 0.01 \\ 0.09 & 0.8 & 0.11 \\ 0.01 & 0.11 & 0.88 \end{pmatrix}$$

The transition probability matrix for moving from $t=0$ to $t=2$, i.e., in two-time steps is:

$$\Pi^2 = \begin{pmatrix} 0.82 & 0.15 & 0.03 \\ 0.15 & 0.66 & 0.19 \\ 0.03 & 0.19 & 0.79 \end{pmatrix}$$

Similarly, the transition probability matrix for moving from $t=0$ to $t=3$, i.e., in three-time steps is:

$$\Pi^3 = \begin{pmatrix} 0.75 & 0.20 & 0.05 \\ 0.20 & 0.56 & 0.24 \\ 0.05 & 0.24 & 0.71 \end{pmatrix}$$

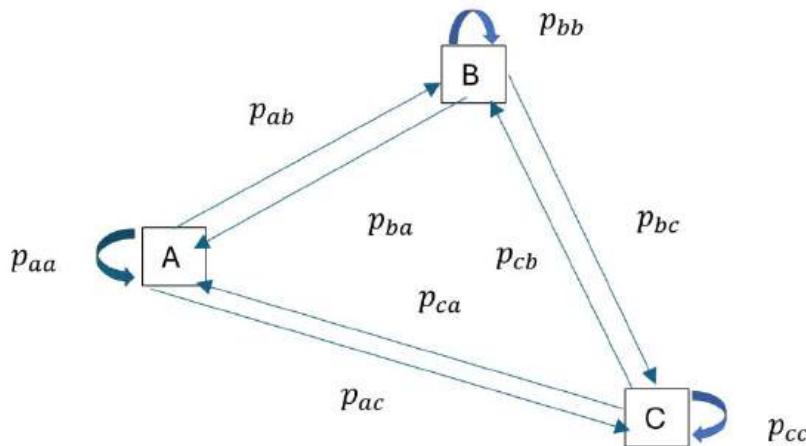


Figure 6.A.1 Markov Transition Probabilities

Appendix 6.B Detailed Reinforcement Learning Example - The Game of Nim

To illustrate how Monte Carlo and Temporal Difference methods work, let's revisit the game of Nim example introduced in Module 1, Section 1.3 - Simple Reinforcement Learning. For simplicity, we will be using a total of 6 coins instead of 20 coins. These 6 coins are placed on a table and each player can remove either 1, 2, or 3 coins each time. The winner is the player who removes the last coin.

We begin without any strategy and employ reinforcement learning to develop an optimal strategy. We assume that the opponent (Player 2) behaves randomly rather than optimally. To setup the framework, we employ the ϵ -Greedy policy with initial ϵ of 1 and the decay factor of 0.9995. Additionally, we choose the learning rate (α) as 0.05, and a discount factor (γ) as 1.

Monte Carlo Method

First, we initialize the Q-Table by setting all Q-values to 0.

Table 6.B.1: Initial Q-Table

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 |

First Episode:

Step 1:

State: 6 coins; Action: Player 1 picks 3 coins.

State: 3 coins; Action: Player 2 picks 2 coins.

Step 2:

State: 1 coin; Action: Player 1 picks 1 coin.

Player 1 wins.

After completing this episode and Player 1 received a reward of 1, we update the Q-Table using the following formulae:

$$Q^{\text{New}}(S, A) = Q^{\text{old}}(S, A) + \alpha [G - Q^{\text{old}}(S, A)]$$

where:

S and A are the current state and action respectively.

γ is the discount factor, which equals 1.

R_t is the reward received at time t .

and,

$G = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$, is the total expected discounted return.

In this episode, players only receive reward at the end of the episode. At $t = 2$, Player 1 receives a reward of $R_2 = 1$. Since $R_1 = 0$ and assumed to be 1, $G = R_1 + \gamma R_2 = 1.0$.

Therefore, after the first step,

$$\begin{aligned} Q^{\text{New}}(6, 3) &= Q^{\text{old}}(6, 3) + \alpha [G - Q^{\text{old}}(6, 3)] \\ &= 0 + 0.05 \times (1 - 0) \\ &= 0.05 \end{aligned}$$

After the second step,

$$\begin{aligned} Q^{\text{New}}(1, 1) &= Q^{\text{old}}(1, 1) + \alpha [G - Q^{\text{old}}(1, 1)] \\ &= 0 + 0.05 \times (1 - 0) \\ &= 0.05 \end{aligned}$$

Table 6.B.2: Q-Table after First Episode

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|---|---|------|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0.05 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0.05 |

Second Episode:

Step 1:

State: 6 coins; Action: Player 1 picks 1 coin.

State: 5 coins; Action: Player 2 picks 1 coin.

Step 2:

State: 4 coins; Action: Player 1 picks 3 coins.

State: 1 coin; Action: Player 2 picks 1 coin.

Player 2 wins

After completing this episode, Player 1 received a reward of $G = -1$. We can update the Q-Table as shown below:

After the first step,

$$\begin{aligned}
Q^{\text{New}}(6, 1) &= Q^{\text{old}}(6, 1) + \alpha [G - Q^{\text{old}}(6, 1)] \\
&= 0 + 0.05 \times (-1 - 0) \\
&= -0.05
\end{aligned}$$

After the second step,

$$\begin{aligned}
Q^{\text{New}}(4, 3) &= Q^{\text{old}}(4, 3) + \alpha [G - Q^{\text{Old}}(4, 3)] \\
&= 0 + 0.05 \times (-1 - 0) \\
&= -0.05
\end{aligned}$$

Table 6.B.3: Q-Table after Second Episode

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|-------|---|-------|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0.05 | 0 | 0 | 0 | 0 | -0.05 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | -0.05 | 0 | 0.05 |

Third Episode:

Step 1:

State: 6 coins; Action: Player 1 picked 3 coins.

State: 3 coins; Action: Player 2 picked 2 coins.

Step 2:

State: 1 coin; Action: Player 1 picked 1 coin.

Player 1 wins

After completing this episode, Player 1 received a reward of G=1. The Q-table is updated as shown below:

After the first step,

$$\begin{aligned}Q^{\text{New}}(6, 3) &= Q^{\text{old}}(6, 3) + \alpha [G - Q^{\text{old}}(6, 3)] \\&= 0.05 + 0.05 \times (1 - 0.05) \\&= 0.0975\end{aligned}$$

After the second step,

$$\begin{aligned}Q^{\text{New}}(1, 1) &= Q^{\text{old}}(1, 1) + \alpha [G - Q^{\text{old}}(1, 1)] \\&= 0.05 + 0.05 \times (1 - 0.05) \\&= 0.0975\end{aligned}$$

Table 6.B.4: Q-Table after Third Episode

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|---|---|-------|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0.0975 | 0 | 0 | 0 | 0 | -0.05 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|-------|---|--------|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 3 | 0 | 0 | 0 | -0.05 | 0 | 0.0975 |

We repeat this process many times. After 1,000,000 episodes, the Q-values converge to the following table:

Table 6.B.5: Q-Table after 1,000,000 Episodes

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|----|--------|--------|---|-------|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | -1 | -0.071 | 0.380 | 0 | 0.774 |
| 2 | 0 | 1 | -1 | -0.110 | 0 | 1 |
| 3 | 0 | 0 | 1 | -0.858 | 0 | 0.020 |

The program eventually learns the correct strategy which is:

| Number of Coins | Optimal Strategy |
|-----------------|------------------|
| 6 | Pick 2 coins |
| 5 | Not applicable |
| 4 | Pick 1 coin |
| 3 | Pick 3 coins |
| 2 | Pick 2 coins |
| 1 | Pick 1 coin |

Temporal Difference Method

Let's re-work the example using temporal difference method. Unlike the Monte Carlo method, where Q-values are updated only after an episode is completed and the reward (G) is obtained, in temporal difference method, we update the Q-values within each episode by considering one time-step ahead of the state we are in. We update the Q-values using the following formula:

$$Q^{\text{New}}(S, A) = Q^{\text{old}}(S, A) + \alpha [R_{t+1} + \gamma V(S') - Q^{\text{old}}(S, A)]$$

Where:

S and A are our current State and Action.

S' is the next state we move to after taking action A in state S .

γ is the discount factor, which is assumed to be 1 here.

R_{t+1} is the reward received in the next period. R equals 1 if Player 1 wins and receives the reward and -1 if Player 1 loses and Player 2 receives the reward.

$V(S) = \max(Q(S, A))$ is the state value function that measures how good a state is.

Note that in our example, R_{t+1} will always equal 0 unless we are at the last step, because players only receive rewards at the end and not at intermediary steps within an episode.

First, we initialize the Q-Table by setting all Q-values equal to 0, as in the Monte Carlo method.

Table 6.B.6: Initial Q-Table

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 |

First Episode:

Step 1:

State: 6 coins; Action: Player 1 picks 3 coins.

State: 3 coins; Action: Player 2 picks 2 coins.

Player 1 will move from $S = 6$ to $S' = 1$.

Step 2:

State: 1 coin; Action: Player 1 picks 1 coin.

Player 1 wins.

The following Q-values can be updated for Player 1:

After the first step, Player 1 moved to $S' = 1$. From [Table 6.B.6](#),

$$V(S') = V(1) = \max(Q(1, A)) = 0$$

$$Q^{\text{New}}(6, 3) = Q^{\text{old}}(6, 3) + \alpha [R_{t+1} + \gamma V(1) - Q^{\text{old}}(6, 3)] = 0 + 0.05 \times (0 + (1 \times 0) - 0) = 0$$

After the second step, no coins are left, and Player 1 won and received the reward, $R_{t+1} = 1$, $V(0) = 0$.

$$Q^{\text{New}}(1, 1) = Q^{\text{old}}(1, 1) + \alpha [R_{t+1} + \gamma V(0) - Q^{\text{old}}(1, 1)] = 0 + 0.05 \times (1 + (1 \times 0) - 0) = 0.05$$

Table 6.B.7: Q-Table after First Episode

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0.05 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 |

We subsequently update $V(1)$ with the maximum value of 0.05 from the Q-Table for State 1.

Second Episode:

Step 1:

State: 6 coins; Action: Player 1 picks 1 coin.

State: 5 coins; Action: Player 2 picks 1 coin.

Player 1 will move from $S = 6$ to $S' = 4$

Step 2:

State: 4 coins; Action: Player 1 picks 3 coins.

State: 1 coin; Action: Player 2 picks 1 coin.

Player 2 wins.

We update the following Q-values for Player 1:

After the first step, Player 1 moved to $S' = 4$. From [Table 6.B.7](#),

$$V(S') = V(4) = \max(Q(4, A)) = 0$$

$$Q^{\text{New}}(6, 1) = Q^{\text{old}}(6, 1) + \alpha [R_{t+1} + \gamma V(4) - Q^{\text{old}}(6, 1)] = 0 + 0.05 \times (0 + (1 \times 0) - 0) = 0$$

After the second step, no coins are left, and Player 1 lost and did not receive any reward, $R_{t+1} = -1$, $V(0) = 0$.

$$Q^{\text{New}}(4, 3) = Q^{\text{old}}(4, 3) + \alpha [R_{t+1} + \gamma V(0) - Q^{\text{old}}(4, 3)] = 0 + 0.05 \times (-1 + (1 \times 0) - 0) = -0.05$$

[Table 6.B.8: Q-Table after Second Episode](#)

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|-------|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0.05 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | -0.05 | 0 | 0 |

Based on [Table 6.B.8](#), the value of $V(1)$ and $V(4)$ remains 0.05 and 0 after the second episode.

Third Episode:

Step 1:

State: 6 coins; Action: Player 1 picks 3 coins.

State: 3 coins; Action: Player 2 picks 2 coins.

Player 1 will move from $S = 6$ to $S' = 1$.

Step 2:

State: 1 coin; Action: Player 1 picks 1 coin.

Player 1 wins.

The following Q-values are updated:

After the first step, Player 1 moved to $S' = 1$.

From [**Table 6.B.8**](#),

$$V(S') = V(1) = \max(Q(1, A)) = Q(1, 1) = 0.05$$

$$Q^{\text{New}}(6, 3) = Q^{\text{old}}(6, 3) + \alpha [R_{t+1} + \gamma V(1) - Q^{\text{old}}(6, 3)] = 0 + 0.05 \times (0 + (1 \times 0.05) - 0) = 0.0025$$

After the second step, no coins are left, and Player 1 received the reward, $R_{t+1} = 1$, $V(0) = 0$.

$$\begin{aligned} Q^{\text{New}}(1, 1) &= Q^{\text{old}}(1, 1) + \alpha [R_{t+1} + \gamma V(0) - Q^{\text{old}}(1, 1)] = 0.05 + 0.05 \times (1 + (1 \times 0) - 0.05) \\ &= 0.0975 \end{aligned}$$

Table 6.B.9: Q-Table after Third Episode

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|---|---|-------|---|--------|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0.0975 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | -0.05 | 0 | 0.0025 |

We update $V(1)$ and $V(6)$ to 0.0975 and 0.0025 based on the maximum value from the Q-Table for State 1 and 6.

After 1,000,000 episodes, the Q-values converge to the following table:

Table 6.B.10: Q-Table after 1,000,000 Episodes

| Coins picked up | State (Number of coins left) | | | | | |
|-----------------|------------------------------|----|--------|--------|---|-------|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | -1 | -0.105 | 0.383 | 0 | 0.845 |
| 2 | 0 | 1 | -1 | 0.058 | 0 | 1 |
| 3 | 0 | 0 | 1 | -0.842 | 0 | 0.011 |

The program eventually learns the correct strategy, which is listed below. In this case, it is identical to the optimal strategy determined using the Monte Carlo method.

| Number of Coins | Optimal Strategy |
|-----------------|------------------|
| 6 | Pick 2 coins |
| 5 | Not applicable |
| 4 | Pick 1 coin |
| 3 | Pick 3 coins |
| 2 | Pick 2 coins |
| 1 | Pick 1 coin |

Reinforcement Learning: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

6.1 State whether the following statements are true or false and explain your answers.

- A. Reinforcement machine learning algorithms could never outperform a human in competitive games such as Chess.

False, although reinforcement learning algorithms mimic the way that humans learn to perform tasks effectively by trial and error, the ability of computers to be able to simulate future states and evaluate the effectiveness of different possible actions across those states means that reinforcement learning can outperform human decision-makers. In fact, there are many situations where reinforcement learners have outperformed even the most skilled humans – for instance, at games such as Chess and Go, and the speed of improvement of reinforcement learners is greater than that of humans so the performance gap is likely to further widen in the future.

- B. The Bellman equations specify the links between the states and the agent's actions.

False. The Bellman equations define the value of a policy – in other words, taking a particular action in a particular state. The equations specify that the value of taking a particular action in a particular state is given by the expected sum of the next period reward plus the discounted value of the next state reached after that action.

- C. The Monte Carlo method involves simulating the entire episode before updating the Q function whereas temporal difference updates Q function one timestep at a time.

True. This is precisely the difference between the two approaches to solving reinforcement learning problems. The Monte Carlo method updates strategies using the total future rewards of one episode. Temporal difference method looks only one decision ahead when updating strategies.

- D. Reinforcement learning is underpinned by the assumption that the agent's goal is to maximize their total (discounted) future rewards.

True. To formulate a policy – in other words, what action the agent should take in each state – we need to assume about the agent's goal, and this is usually that they want to maximize their return over the longer term, i.e., over the entire exercise. Future rewards are usually discounted in the same way as cashflows because they are less attractive the further ahead in the future that they are received.

- E. The ε -greedy strategy involves both exploration and exploitation whereas the random strategy only exploits.

False. The ϵ -greedy strategy involves both exploration and exploitation phases. The random strategy makes a random selection each time ignoring the performance of selections made up to that point.

- F. In a Markov process, the actions taken in the current state are dependent on the current state and previous states.

False. Markov decision processes (MDPs) are simple settings for environment dynamics. In this case the environment changes based on the actions of the agent. MDPs are processes that have no memory, which means that only the current state is relevant for determining the most appropriate current action and not any of the previous states.

6.2 What are the three different strategies that can be used in MAB? How do they differ?

Greedy strategy, Random strategy and ϵ -greedy strategy. The greedy strategy is a simple strategy in which the agent always chooses the actions with the best rewards seen so far. So, it tends to stick with one action. The random strategy is to randomly select a slot machine to play. So, it tends to change actions. The ϵ -greedy strategy combines greedy exploitation and random exploration. Here ϵ is a hyperparameter (with $0 < \epsilon < 1$) that determines whether a random selection is made to explore, or a greedy selection is made to exploit.

6.3 Consider a multi-arm bandit problem with β equal to 0.99 and five slot machines.

- A. What would be the value of ϵ at the 89th trial assuming an exponential decay factor is used?

The value of ϵ at the 89th trial can be computed using the formula .

- B. Assume that random number of 0.8 has been extracted at the 89th trial. Should you explore or exploit?

The value of ϵ calculated above is smaller than 0.8. Therefore, we should exploit the machine that has been the most successful so far.

- C. At trial 150, you explore the third machine for the 50th time; the current value of the expected reward for this machine is 1.8. You obtain a payoff equal to 1. What is the updated expected reward for the machine?

Using equation (6.2),

$$Q_3^n = Q_3^{n-1} + \frac{1}{n}(R_n - Q_3^{n-1})$$

Here, $n = 50$, $Q_3^{n-1} = 1.8$, and $R_n = 1$ therefore:

$$Q_3^{n-1} = 1.8 + \frac{1}{50}(1 - 1.8) = 1.784.$$

Chapter 7: Supervised Learning - Model Estimation

Learning Objectives

This chapter builds on concepts learned in Chapter 3, with a focus on the estimation of linear regression models using ordinary least squares and maximum likelihood methods, the estimation of model parameters when data with nonlinear characteristics is used, and the optimization of model parameters using gradient descent method. Initial insight on the predictive value of models and techniques for improving model output is also covered, including over-and-under-fitting, bias-variance tradeoff, and methods for adjusting models with highly correlated features.

After completing this chapter, you should be able to:

- Compare and contrast the Ordinary Least Squares (OLS) and Maximum Likelihood methods.
- Explain how gradient descent method is used to optimize parameter estimates.
- Explain how backpropagation is used to determine the weights in neural networks.
- Discuss the differences between underfitting and overfitting and potential remedies for each.
- Describe the tradeoff between bias and variance.
- Explain the use of regularization techniques to simplify models.
- Describe cross-validation and its uses.
- Describe the accuracy-interpretability tradeoff.
- Describe how grid search and bootstrapping can be used to optimize hyperparameter estimation.

7.1.1 Ordinary Least Squares

Ordinary least squares (OLS) is the most straightforward of the available estimation methods and is an analytical approach used to estimate the parameters in linear regression models of the type discussed in Chapter 3. To see how it works, let us return to the example used in that chapter of the relationship between bank worker years of experience and salary. As a reminder, the 14 data points are repeated in [Table 7.1](#).

Table 7.1: Salary and experience for a notional sample of bank employees

| Observation, i | Experience, x_i | Salary, y_i |
|------------------|-------------------|---------------|
| 1 | 1 | 15.46 |
| 2 | 3 | 22.40 |
| 3 | 7 | 27.47 |

| Observation, i | Experience, x_i | Salary, y_i |
|------------------|-------------------|---------------|
| 4 | 12 | 34.31 |
| 5 | 9 | 33.08 |
| 6 | 3 | 18.70 |
| 7 | 20 | 35.06 |
| 8 | 22 | 35.78 |
| 9 | 0 | 14.81 |
| 10 | 4 | 14.84 |
| 11 | 6 | 25.78 |
| 12 | 8 | 28.17 |
| 13 | 4 | 26.36 |
| 14 | 2 | 11.12 |

In this case, suppose that we use a simple linear regression so that there is one feature, x_i (experience) and one label or target, y_i (salary). We would like to identify the intercept and slope parameters in the model that best describe the relationship between the variables. This line of best fit was shown in Figure 3.1, but we will now give more detail on where it came from. OLS finds this line by forming the residual sum of squares as a function of the parameters and minimizing it. We can write the regression equation as:

$$y_i = b_0 + b_1 x_i + u_i \quad (7.1)$$

Here, x_i and y_i are the variables, and we have 14 observations for each; b_0 and b_1 are the intercept and slope parameters, respectively, to be estimated; u_i is a random error term, which is assumed to have a zero mean and

constant variance, also known as a disturbance term. The error term allows for the fact that it is very unlikely there would be a perfect linear relationship between the two variables. We define the residual, \hat{u}_i , as the difference between the actual value of the series, y_i , and the fitted value from the model for that data point i , \hat{y}_i :

$$\hat{u}_i = y_i - \hat{y}_i$$

Then we can construct the residual sum of squares, RSS , which is obtained by taking each of the residuals, squaring them, and then adding them over all N data points:

$$RSS = \sum_{i=1}^N \hat{u}_i^2 = \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (7.2)$$

Note that we square the residuals first rather than simply summing them, otherwise the positive and negative residuals (corresponding to points above and below the fitted line, respectively) would cancel each other out, and squaring makes them all positive.

The mean squared error (MSE) is another commonly used metric. It is the average of the residual sum of squares:

$$MSE = \frac{1}{N} \sum_{i=1}^N \hat{u}_i^2 = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (7.3)$$

We want to find the values of the parameters that minimize this RSS , and so we substitute \hat{y}_i with the fitted equation for the line, $\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i$, where the hats above the parameters denote the estimated values:

$$RSS = \sum_{i=1}^N (y_i - \hat{y}_i)^2 = \sum_{i=1}^N (y_i - \hat{b}_0 - \hat{b}_1 x_i)^2 \quad (7.4)$$

We partially differentiate this expression for the *RSS* separately with respect to each of the parameters, set the derivatives to zero and then rearrange the formulae to obtain expressions for the optimal intercept and slope estimates. The resulting formulae are, for the intercept:

$$\hat{b}_0 = \bar{y} - \hat{b}_1 \bar{x} \quad (7.5)$$

And for the slope:

$$\hat{b}_1 = \frac{\sum_{i=1}^N y_i x_i - N \bar{y} \bar{x}}{\sum_{i=1}^N x_i^2 - N \bar{x}^2} \quad (7.6)$$

Where \bar{x} and \bar{y} denote the mean of the observations on x and y , respectively. If we plugged the 14 observations for the experience and salary data of [Table 7.1](#) into these formulae, we would end up with the estimates given in Chapter 3, which were $\hat{b}_0 = 16.88$ and $\hat{b}_1 = 1.06$. Any other regression line having a different intercept or slope coefficient would fit the data less well and would lead to a higher *RSS*, as shown for the impact of changing the slope in [Figure 7.1](#). The same results will be obtained if we use *MSE* instead of *RSS*.

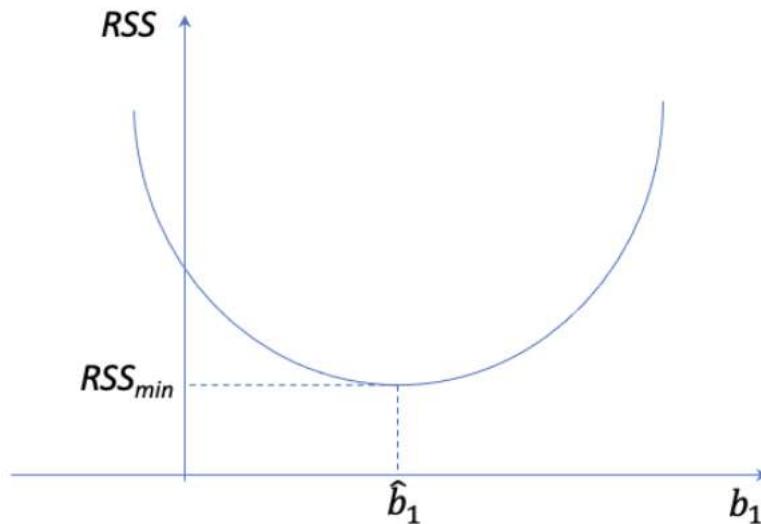


Figure 7.1 How RSS changes as the parameter moves away from its optimal value

OLS minimizes the sum of the squares of the distances from all the points together to the fitted line. [Figure 7.2](#) shows the fitted line, the actual data point, the fitted value, and the residual for just a single data point. In this case we plot the data point # 5 in [Table 7.1](#), but we could have done this for any of the points in the table. We can see here that the actual value of the salary is \$33.08 per hour, but the model would have predicted a value of \$26.42 (calculated as: $16.88 + 9(1.06)$). Thus, for this data point, the value of the residual is $33.08 - 26.42 = 6.66$.

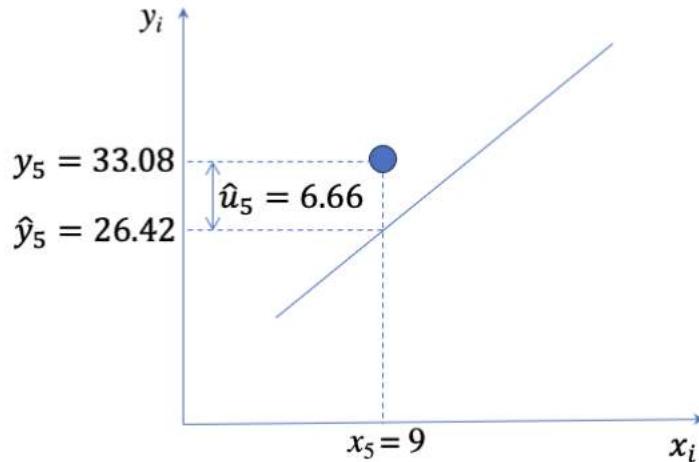


Figure 7.2 The residual, actual and fitted values for data point number 5

Once the parameters are estimated, we can similarly calculate the fitted values and residuals for all the data points, as shown in [Table 7.2](#).

Table 7.2: Salary and experience for a notional sample of bank employees with calculated fitted values and residuals

| Observation, i | Experience, x_i | Salary, y_i | Fitted value, \hat{y}_i | Residual, \hat{u}_i |
|------------------|-------------------|---------------|---------------------------|-----------------------|
| 1 | 1 | 15.46 | 17.94 | -2.48 |
| 2 | 3 | 22.40 | 20.06 | 2.34 |
| 3 | 7 | 27.47 | 24.30 | 3.18 |
| 4 | 12 | 34.31 | 29.59 | 4.71 |
| 5 | 9 | 33.08 | 26.42 | 6.66 |
| 6 | 3 | 18.70 | 20.06 | -1.36 |

| Observation, i | Experience, x_i | Salary, y_i | Fitted value, \hat{y}_i | Residual, \hat{u}_i |
|------------------------------------|-------------------------------------|---------------------------------|---|---|
| 7 | 20 | 35.06 | 38.07 | -3.01 |
| 8 | 22 | 35.78 | 40.19 | -4.40 |
| 9 | 0 | 14.81 | 16.88 | -2.07 |
| 10 | 4 | 14.84 | 21.12 | -6.28 |
| 11 | 6 | 25.78 | 23.24 | 2.54 |
| 12 | 8 | 28.17 | 25.36 | 2.81 |
| 13 | 4 | 26.36 | 21.12 | 5.24 |
| 14 | 2 | 11.12 | 19.00 | -7.88 |

As [Table 7.2](#) shows, although OLS will minimize the collective distances from the data points to the line, none of the residuals is very close to zero, and some points are well above the line (positive residual), while others are well below it (negative residual). If we believe the noise level is low, then we may suspect that the model fit is not very good, which might motivate us to consider alternative or additional predictors such as the square of the experience term discussed in Chapter 3.

A similar approach would be used in the multiple linear regression context where we have more than one explanatory variable. If there were two explanatory variables (i.e., three parameters to estimate: an intercept and two slopes), we could derive a set of three equations like the two above. But in the case of more explanatory variables than two, although the formulae can still be derived, they become increasingly long and unwieldy.

Therefore, it is more common to use a matrix notation for the model and the estimation formula, which will extend naturally and straightforwardly to however many explanatory variables there are.

7.1.2 Nonlinear Least Squares

OLS can be used in situations where the underlying model is linear in the parameters, but this does not apply to many machine learning models, such as neural networks. In these cases, a more flexible approach is needed. Nonlinear least squares (NLS) is an approach that can be used when the model is nonlinear, and it works using the same principles as OLS – i.e., by minimizing the residual sum of squares. This would still be written as:

$$RSS = \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (7.7)$$

But in this case, $\hat{y} = f(x_1, x_2, x_3, \dots, x_m; w)$ where f can be any nonlinear function of the m explanatory variables or features, which are denoted by x_i , and the corresponding parameters are denoted by w_i (also known as weights in the case of neural networks). Similarly, mean squared error (MSE) can also be calculated. Because the relationship between the features and the output could in principle take any form, it is often not possible to derive a set of closed form solutions to this minimization problem. Therefore, NLS usually uses a numerical approach to finding the optimal parameter estimates, and it proceeds with the following steps:

1. Begin with a set of initial values for the parameters – these could either be randomly generated or ‘best preliminary guesses.’
2. Evaluate the objective function (*RSS or MSE*).
3. Modify the parameter estimates and re-evaluate the objective function.
4. If the improvement in the objective function is below a pre-specified threshold, then stop looking and report the current parameter values as the chosen ones. If not, return to step 2.

The third of the above steps is the crucial aspect, and usually, a *gradient descent algorithm* is employed, which is discussed in a following sub-section after a preliminary discussion of hill climbing¹.

¹ Genetic algorithms are one of the alternative approaches for machine learning model parameter optimization. They are discussed in Appendix 7.A.

7.1.3 Hill Climbing

A simple form of optimizer for estimating the parameters of a nonlinear model is *hill climbing*. This involves starting with initial guesses of each parameter and then making small changes in both directions to each parameter one-at-a-time. The aim is to maximize the value of an objective function (for instance, increasing the value of a likelihood function or increasing the negative of the RSS) until no further improvement in its value is observed. Hill climbing is very straightforward because it does not require the calculation of derivatives, and therefore it can be applied to non-differentiable functions. It is also simple to implement and for this reason it is sometimes termed a “heuristic optimizer,” but it has several disadvantages:

- Of all the optimization techniques available, hill climbing is the most susceptible to getting stuck in local optima.
- Convergence to the optimal solution can be very slow.
- Only one parameter can be adjusted at a time, meaning that it is easy for the algorithm to miss optimal parameter combinations, particularly for complex and highly interconnected models.

Given these limitations of hill climbing, an approach based on gradient descent instead usually forms the workhorse of parameter optimization for machine learning models.

7.1.4 The Gradient Descent Method

A popular numerical procedure for parameter estimation is known as *gradient descent*, which is illustrated in [Figure 7.3](#). In this method, the objective function, for example the residual sum of squares, is minimized. Suppose that all the parameters to be estimated are stacked into a single vector W and the objective function in this case is known as the loss function and is denoted as $L(W)$. At each iteration, the algorithm chooses the path of steepest descent (“slope”), which is the one that will minimize the value of the loss function the most. The method works

similarly when the maximum likelihood method is used, in which case the negative of the log-likelihood function is minimized.

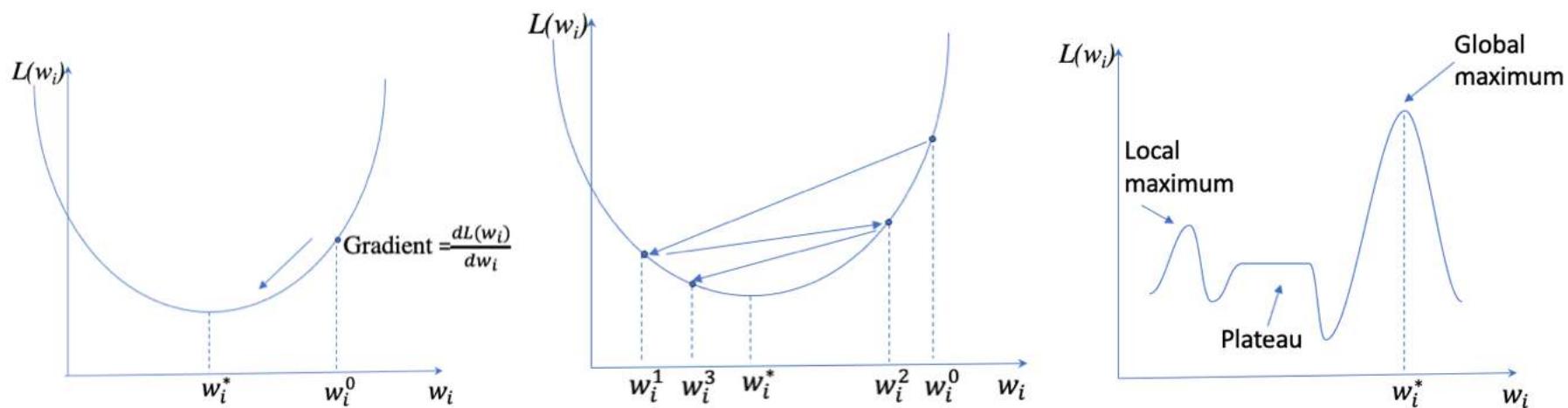


Figure 7.3 Gradient descent. The left panel demonstrates how gradient descent works; the center panel shows how an overly high learning rate leads to overshooting; and the right panel shows the loss function containing a local optimum (in this case, local maximum) and plateau.

Figure 7.3 demonstrates how the technique works for a single parameter, w_i . Starting with an initial guess w_i^0 , the aim is to move towards the optimal value, w_i^* , which occurs where the loss function is minimized. The gradient of the function is given by its partial derivative with respect to the parameter, evaluated at the initial guess:

$$\frac{\partial \mathcal{L}(w_i)}{\partial w_i} \Big|_{w_i=w_i^0}$$

The guess of the weight is then updated to a revised value w_i^1 , according to the formula:

$$w_i^1 = w_i^0 - \eta \frac{\partial \mathcal{L} (w_i)}{\partial w_i} \quad (7.8)$$

Here, $\frac{\partial \mathcal{L} (w_i)}{\partial w_i}$ is the partial derivative of the loss function with respect to the weight w_i and η is called the learning rate, usually in the $(0,1)$ range.

We can also write this partial derivative of the loss function with respect to the weight w_i as:

$$\frac{\partial \mathcal{L} (w_i)}{\partial w_i} = 2 \sum_{i=1}^N (y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial w_i} \quad (7.9)$$

If mean squared error is selected as the loss function, the partial derivative of the loss function with respect to the weight w_i is:

$$\frac{\partial \mathcal{L} (w_i)}{\partial w_i} = \frac{2}{N} \sum_{i=1}^N (y_i - \hat{y}_i) \frac{\partial \hat{y}_{in}}{\partial w_i} \quad (7.10)$$

Determining the gradient of the loss function with respect to the weight involves calculating the partial derivative of the output with respect to the weight.

Note that adjustment to weight w_i takes place in the opposite direction to the gradient (or "slope"), and hence the negative sign in the equation 7.8 above for updating the weight. When the gradient is positive, as it is in the illustrative example of the curve in the left panel of [Figure 7.3](#), the estimate of the weight w_i will be reduced at the next step. At the same time, if the gradient had been negative, the current value of w_i would be too small, and hence it would be increased at the next step. As described earlier, the process of calculating the gradient and updating the weights continues until the value of the loss function can no longer be improved, indicating convergence has been achieved. To avoid the iteration going into infinite cycles, a maximum number of iterations

is assigned to the algorithm. In machine learning parlance, each iteration, comprising calculating the loss function and gradient then adjusting the weights using the whole training data sample, is known as an *epoch*.

We usually use the entire training data sample and minimize the loss function with respect to all of it, which is known as *batch* gradient descent. A slightly less common alternative approach is to apply gradient descent to each data point at a time, individually selected at random from the training set. This is known as *stochastic* gradient descent, or applying it to subsets of the training data, known as *mini-batch gradient descent*. The benefit of stochastic gradient descent is that it does not require the entire database to be loaded into memory simultaneously, which reduces the required computational resources compared with the batch approach. However, because updating the weights will take place after the algorithm sees each new data point, the convergence on the optimal values will require more iterations and will be less smooth.

The parameter η is a hyperparameter that defines how much adjustment to weights takes place. This hyperparameter must be chosen judiciously: if it is too small, each iteration will yield only modest improvements in the loss function and will result in a slow movement towards the optimum, requiring many iterations in total. On the other hand, if η is too large, there is a danger of overshooting and an erratic path towards the optimal w_i^* . The algorithm can overshoot the minimum, oscillate around it, or may not converge, as illustrated in the center panel of [Figure 7.3](#).

For batch gradient descent, η is fixed *a priori*, but for stochastic gradient descent, the learning rate can be made a diminishing function of the number of epochs that have already occurred so that the weight updating slows down as the algorithm comes closer to the optimum. In other words, we could employ dynamic learning, which entails starting with a larger η to get close to the optimal solution faster, but to then reduce η as the learning proceeds to avoid overshooting. Technically, this means that η is subject to a decay function. Popular choices are the exponential decay:

$$\eta_t = \eta_0 e^{-kt} \quad (7.11)$$

or the inverse decay:

$$\eta_t = \frac{\eta_0}{1 + kt} \quad (7.12)$$

where the parameter k controls the rate of the decay and t is an epoch. Gradient descent is a neat technique that usually works well, but it can run into problems. One such issue occurs when the function does not have a single, global optimum but rather a series of local optima or an extended plateau away from the true optimum, as illustrated in the right panel of [Figure 7.3](#). In such circumstances, the optimizer can get stuck at the local optimum or plateau and never reach the optimal solution w_i^* .

7.1.5 Illustration of the Gradient Descent Method

Although we know that the OLS estimator can be derived analytically as shown above, it might be useful for pedagogical purposes to illustrate how gradient descent would work if we had used it to derive the estimates for the regression of salary on experience. To keep complexity at a minimum and focus on illustrating how the optimization works, assume that we already know $b_0 = 16.88$ and hence we only need to optimize over b_1 . The function that we are trying to minimize is:

$$\text{MSE}(b_1) = \frac{1}{N} \sum_{i=1}^N (y_i - 16.88 - b_1 x_i)^2 \quad (7.13)$$

We also know from above that the gradient of this function is given by:

$$\frac{\partial \text{MSE}(b_1)}{\partial b_1} = \frac{2}{N} \sum_{i=1}^N (y_i - \hat{y}_l) \frac{\partial \hat{y}_i}{\partial w_i} = -\frac{2}{N} \sum_{i=1}^N (y_i - \hat{y}_l) x_i \quad (7.14)$$

Suppose that we start from an initial value of 0.8 and we use a learning rate equal to 0.005. [Table 7.3](#) illustrates the iterations that lead the algorithm to converge to the value of b_1 for which the MSE is minimized. Using the initial value, we compute the gradient, which is equal to -48.66 . Given the learning rate of 0.005, the change that we need to apply to b_1 is $-0.005 \times -48.66 = 0.243$ (as discussed above, we move in the opposite direction to the gradient). Therefore, the new value of the parameter is 1.043. We continue in this fashion for a number of iterations as shown in the table below.

Table 7.3: Value of b_1 in successive iterations when the learning rate is 0.005.

| Iteration | b_1 | Gradient | Change in b_1 | New b_1 |
|-----------|-------|----------|-----------------|-----------|
| 0 | 0.800 | -48.664 | 0.243 | 1.043 |
| 1 | 1.043 | -3.024 | 0.015 | 1.058 |
| 2 | 1.058 | -0.188 | 0.001 | 1.059 |
| 3 | 1.059 | -0.012 | 0.000 | 1.059 |
| 4 | 1.059 | -0.001 | 0.000 | 1.059 |
| 5 | 1.059 | 0.000 | 0.000 | 1.059 |

After five iterations, the algorithm converges to 1.059. If we calculate the gradient for this parameter value, it is approximately zero as we have reached the minimum point, and no further adjustments are needed.

7.1.6 Backpropagation

Although the above discussion illustrated the technique by focusing on one parameter, w_i , in practice, the gradients will be calculated for all the weights at each iteration. Determining the optimal weights in a neural network model is particularly challenging because, even with a single hidden layer, the output is a function of a function. It is like running a logistic regression on the output from another logistic regression. Therefore, to use a technique such as gradient descent would require repeated use of the chain rule of differentiation. As outlined in Chapter 4, a technique known as *backpropagation* is used along with gradient descent to determine the weights in neural network models.

The backpropagation algorithm involves starting on the right-hand side of the neural network (i.e., beginning with the output layer) and then successively working backward through the layers to update the weights estimates at

each iteration. This begins by calculating the errors (actual – fitted values) for each target data point, then these errors are “assigned” to each of the weights in the layer before it. Gradient descent can then be applied to the weights to calculate improved values. The output layer error in the target is determined via a feedforward of the feature values with the updated weights, and the process continues again. The derivatives are computed starting from the output layer and moving backward, with an application of the chain rule (hence, the name backpropagation). The algorithm stops when convergence is achieved – that is, when updating the weights no longer reduces the cost function (or reduces it by a trivially small amount). The key to backpropagation is to consider each layer separately rather than trying to do all the computation in a single step because breaking it down in this way greatly simplifies the mathematics.

In summary, the steps to implement the backpropagation approach to learning the optimal ANN weights are:

1. Generate initial guesses (usually at random) for all the weights, including the biases.
2. Given these weights, feedforward the values of the inputs to calculate the values at each neuron and then finally the value of the outputs. This is done separately for each of the N data points in the training sample.
3. Once the fitted values of the outputs are determined, the error, which is the difference between the network output and the actual value, can be calculated for each observation.
 - a. If this is the first iteration, proceed to step 4.
 - b. If the residual sum of squares is below a particular threshold or has not improved much since the previous iteration, or the number of iterations has reached a pre-specified maximum value, stop and fix the weights at their current values. Otherwise, proceed to step 4.
4. During the backward pass, the gradient descent method is used to calculate improved values of the weights. In this process the error is propagated through the network, and the weights are updated to minimize the loss function. Return to step 2 and run through a further iteration.²

A solution is obtained after several iterations (epochs). The required number of iterations depends on the problem at hand and criteria used to determine convergence of the solution as described in step 3(b).

² The gradient of the loss function is calculated for each datapoint by working backward using the chain rule of partial differentiation.

7.1.7 Computational Issues

In the previous section we discussed how to find the weights such that some cost function is minimized. Clearly, we hope to find the global minimum of such a function. However, a risk exists that the algorithm will fail to find a global minimum but will instead get trapped in a local optimum (see the right-hand diagram in Figure 7.3 for an illustration of this).

Sometimes a *momentum term* is added to the optimizer, which increases the learning rate if the previous change in the weights and the current change are in the same direction but reduces it if they are in the opposite directions. The benefit of this approach is that it speeds up convergence but at the same time reduces the probability of overshooting the optimal parameter values. Incorporating momentum involves a modification of the updating scheme discussed above to:

$$w_i^{\text{new}} = w_i^{\text{old}} - \eta \frac{\partial E}{\partial w_i^{\text{old}}} + \mu |w_i^{\text{old}} - w_i^{\text{older}}| \quad (7.15)$$

where w_i^{older} is the weight before w_i^{old} and μ is the momentum rate, which can be chosen between 0 and 1. The parameter μ controls how much of the previous weight change we will keep in the next iteration. This works by ‘overshooting’ the target, which helps prevent the algorithm from getting stuck in local minima.

Generally, the problem of local minima is less critical in networks with many hidden layers (deep networks). However, deep networks are plagued by another computational issue: the so-called *vanishing gradient*. Loosely, the problem can be understood as follows. Backpropagation is an application of the chain rule, which entails the multiplication of several derivatives (as many as the layers in the network). When the derivatives are numbers between 0 and 1 (which is always the case for the logistic function), their product tends to become very small very quickly. An opposite problem is an *exploding gradient*, where the product of the derivatives becomes larger and larger. When one of these problems emerges, the only way to find the optimum is by using an extremely large number of small updates, which of course makes the learning very slow.

Vanishing or exploding gradients are by far the most relevant problems for deep network structures such as convolutional and recurrent neural networks (see the discussion below). Solutions to these issues include:

- *An appropriate choice of activation function.* In recent years, the use of the logistic function (sigmoid) as an activation function for hidden layers has been abandoned in favor, for instance, of the ReLU function that is less prone to the vanishing gradient problem.
- *Batch normalization.* This consists of adding ‘normalization layers’ between the hidden layers. The normalization works in a fashion like what we discussed in Chapter 1, where the features were normalized by subtracting the mean and dividing by the standard deviation. Here, it is the new inputs originating from the hidden layers that are normalized.
- *Specific network architectures.* Some network architectures have been developed to be resistant to the vanishing or exploding gradient problem. An example is the long short-term memory (LSTM) network (see the discussion in Chapter 10).

7.2 Maximum Likelihood

Maximum likelihood (ML) is another estimation technique that can be used for nonlinear models as an alternative to NLS. ML can also be used for linear models, in which case it will give identical estimates for the intercept and slope parameters as OLS if the random errors are independently normally distributed with zero mean and constant variance. To a certain degree, ML is a more flexible framework that enables it to be used to estimate the parameters for a wide range of specifications if the distribution of the data generation mechanism is known. For instance, it is used for numerous models from the GARCH (generalized autoregressive conditionally heteroskedastic) family that are commonly used for forecasting volatility. More relevant here, ML is also used to estimate logit and probit, which, like GARCH models, are nonlinear.

Recall from Chapter 3 that for a logit model, we have two possible outcomes: $y_i = 0$ or $y_i = 1$, which might, for example, correspond to a customer not being granted a loan and being granted a loan, respectively, or did not default on a mortgage payment and did default, etc. The probability that $y_i = 1$ is given by:

$$P_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \dots + \beta_m x_{mi} + u_i)}} = F(y_i) \quad (7.16)$$

and the probability that $y_i = 0$ is $1 - P_i$. The function F denotes the logistic function, and this is just a short-hand way of writing the middle of this equation. We could also call P_i the likelihood that $y_i = 1$, and therefore the likelihood that $y_i = 0$ is $1 - P_i$.

The probability distribution for y_i given the data on the explanatory variables can be written:

$$L(y_i) = F(y_i)^{y_i} (1 - F(y_i))^{1-y_i} \quad (7.17)$$

Here, P_i is the likelihood that $y_i = 1$ for just one observation i , but because ML works by selecting the parameters that maximize the chances of the training data occurring, we want the joint likelihood of all N data points taken together, not just a single observation. This joint likelihood, denoted as L , will be obtained by multiplying all the individual probability distributions together:

$$L = L(y_1)L(y_2)\dots L(y_N) = \prod_{i=1}^N L(y_i) = \prod_{i=1}^N F(y_i)^{y_i} (1 - F(y_i))^{(1-y_i)} \quad (7.18)$$

Notice that the \prod notation is used here to denote that the functions are multiplied because the joint probability of all the N data points is the product of the $F(y)$ across the positive outcomes ($= 1$) in the training set multiplied by the product of the $(1 - F(y))$ across the negative outcomes ($= 0$) in the training set, as long as they are independent. If y_i is 1, the i th function reduces to $F(y_i)$; if it is zero, the i th function reduces to $(1 - F(y_i))$.

It is easier to maximize the log-likelihood function, $\log(L)$, than to maximize the likelihood function because the logarithmic transformation turns a multiplication into a sum. The log-likelihood is obtained by taking the natural logarithm of the above expression:

$$\log(L) = \sum_{i=1}^N [y_i \log(F(y_i)) + (1 - y_i) \log(1 - F(y_i))] \quad (7.19)$$

This can also be written:

$$\log(L) = \sum_{i=1:y_i=1}^N \log(F(y_i)) + \sum_{i=1:y_i=0}^N \log(1 - F(y_i)) \quad (7.20)$$

Once the parameters that maximize this expression have been estimated, predictions can be constructed from the model by setting a threshold, Z , estimating the value of P_i using the equation above, and then specifying the category that observation i is predicted to belong to as follows:

$$\hat{y}_i = \begin{cases} 1 & \text{if } P_i \geq Z \\ 0 & \text{if } P_i < Z \end{cases} \quad (7.21)$$

If the costs of being wrong are the same for the two categories (i.e., if incorrectly classifying a value of y as one when it should be zero is just as bad as classifying y as zero when it should be one), we might set $Z = 0.5$. But in other cases, a different threshold is more useful (see also the discussion in Chapter 8). Consider classifying loans according to the probability that they will default ($y_i = 1$) and the probability that there will be a full payback ($y_i = 0$). In this case, we might set Z equal to a low value such as 0.05 for decision making. This is because the cost of predicting that a loan will pay back when it defaults (i.e., the cost of making a bad loan) is much greater than the cost of predicting that a loan will default when it turns out to be fine (i.e., the profit foregone because the loan was not made).

7.3 Overfitting, Underfitting, and Bias-variance Trade Off

Overfitting was only briefly discussed in Chapter 1. We return to the issue here and discuss it in more detail. Ideally, we would always specify the “correct” model containing the “correct” variables, which would lead us to estimate the most appropriate number of parameters given the features and the outputs. However, we will never know the true process generating the data, and we will only have a sample of data upon which to select an appropriate model and estimate the parameters. Consequently, it is an empirical choice as to the size of model we estimate, which leads to the possibility that the model contains too many (overfitting) or too few (underfitting) parameters. Each of these problems and their implications are discussed below.

7.3.1 Overfitting

Overfitting is a situation in which a model is chosen that is “too large” or excessively parameterized. A simple example is when a high-dimensional polynomial is used to fit a data set that is roughly quadratic. The most obvious sign of an overfitted model is that it performs considerably worse on new data. When building a model, we use a training data set and a validation data set. The training set is used to estimate the model parameters, and the validation set is used to evaluate the model’s performance on a separate data set. An overfitted model captures excessive random noise in the training set rather than just the relevant signal. Overfitting gives a false impression of an excellent specification because the RSS on the training set will be very low (possibly close to zero). However, when applied to other data not in the training set, the model’s performance will likely be poor, and the model will not be able to generalize well.

Overfitting is usually a more severe issue with machine learning than with conventional econometric models due to the larger number of parameters in the former. For instance, a standard linear regression model generally has a relatively small number of parameters. By contrast, it is not uncommon for neural networks to have several thousand parameters.

7.3.2 Underfitting

Underfitting is the opposite problem to overfitting and occurs when relevant patterns in the data remain uncaptured by the model. For instance, we might expect the relationship between the performance of hedge funds and their size (measured by assets under management) to be quadratic. Funds that are too small would have insufficient access to resources with costs thinly spread, and funds that are too big may struggle to implement their strategies in a timely fashion without causing adverse price movements in the market. A linear model would not be able to capture this phenomenon and would estimate a monotonic relationship between performance and size, and so would be underfitted. A more appropriate specification would allow for a nonlinear relationship between fund size and performance.

Failure to include relevant interaction terms, as described earlier in Chapter 3, would be a further example of underfitting. It is clear from these examples that underfitting is more likely in conventional models than in some machine-learning approaches where only a minimal assumption (such as smoothness) on the signal is imposed.

However, it is also possible for machine-learning approaches, as well as econometric models, to underfit the data. This can happen either when the number or quality of inputs is insufficient, or if steps taken to prevent overfitting are excessively stringent. In such cases, the model fit to the training data will be poor, and there will be characteristics of the output variable that remain uncaptured by the model postulated. This will also likely lead to biased estimates of the parameters on the variables that are included in the model.

7.3.3 Bias-variance Trade Off

The choice of the “size” of the machine-learning model, which will determine whether the data are overfitted, underfitted, or appropriately fitted, involves what is termed a bias-variance tradeoff. If the model is underfitted, the omission of relevant factors or interactions will lead to biased predictions but with low variance. On the other hand, if the model is overfitted, there will be low bias but a high variance in predictions. This is illustrated in [Figure 7.4](#).

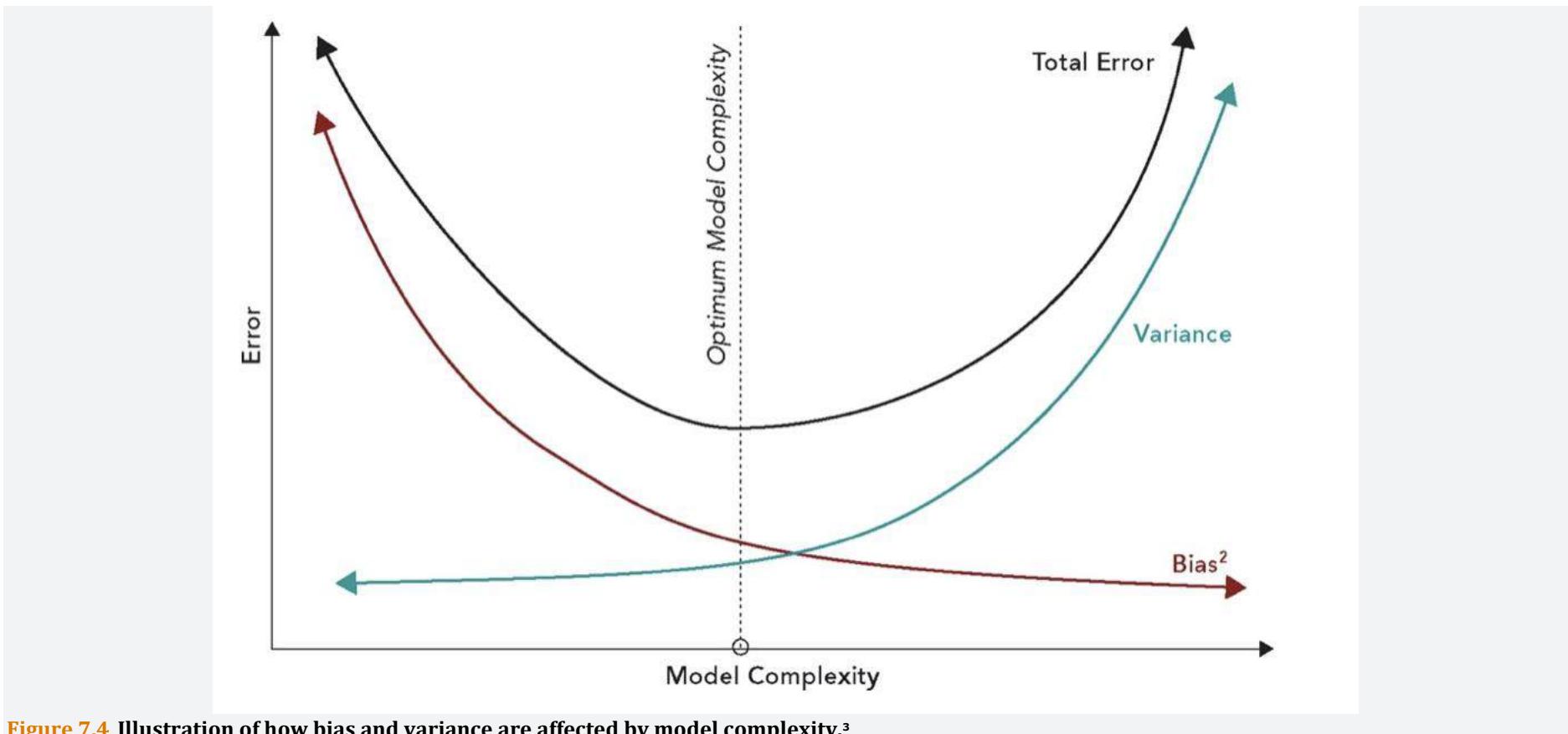


Figure 7.4 Illustration of how bias and variance are affected by model complexity.³

An example can help to understand the problem of overfitting discussed above and how an analyst can strike a balance between that and the opposite problem, that is, *underfitting*. Consider an analyst assigned to the task of predicting the price at which a house will sell using the age of the house as a predictor. A simple approach would be to estimate a linear regression of the house prices on their ages:

$$\widehat{\text{House Price}} = \hat{\beta}_0 + \hat{\beta}_1 \text{Age} \quad (7.22)$$

The regression line fitted using a training sample of 201 data points is depicted in [Figure 7.5](#) (Panel A, blue line). Visibly, a linear regression is insufficient to fully capture the relationship between house prices and their ages. In fact, both very new and very old houses (with a historic value) seem to be more expensive, while the linear regression forces the predicted prices to be strictly decreasing in age. Fortunately, the linear regression model can accommodate polynomials of higher degree, as the linearity requirement concerns the parameters, not the explanatory variables (see Chapter 3). Therefore, the analyst could include a quadratic term to the regression above and estimate:

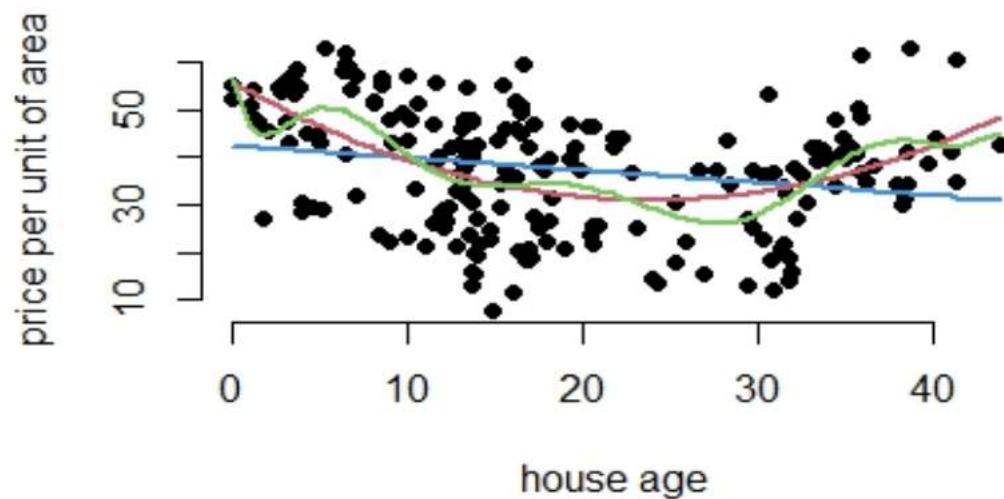
$$\widehat{\text{House Price}} = \hat{\beta}_0 + \hat{\beta}_1 \text{Age} + \hat{\beta}_2 \text{Age}^2 \quad (7.23)$$

The red line in [Figure 7.5](#), Panel A is the estimated quadratic regression. This looks like a much more accurate representation of the data. Finally, the green line in [Figure 7.5](#) (Panel A) is the result of fitting a polynomial of the ninth degree to the data. This highly complex function can capture some highly nonlinear patterns in the data. However, it also captures some data-specific noise (e.g., see the curved shape between 20 and 30 years of age) and it does not generalize well when we deploy the estimated model to predict unseen data, as shown in [Figure 7.5](#) (Panel B).⁴

A highly complex model minimizes the MSE in the training sample but may fail to do so when deployed on new data. In the example illustrated in [Figure 7.5](#), the MSE over the training sample is 157.67 for the linear regression, 130.34 for the quadratic regression, and 121.73 for the ninth order polynomial regression. However, when we use the trained model to predict observations that were not part of the training set, the MSE is 156.37 for the linear regression, 137.08 for the quadratic regression, and 139.82 for the ninth order polynomial regression. Therefore, in this example, the quadratic model strikes the right balance between fitting the training data accurately while also generalizing well to new data.

Panel A

House prices



Panel B

House prices (Test)

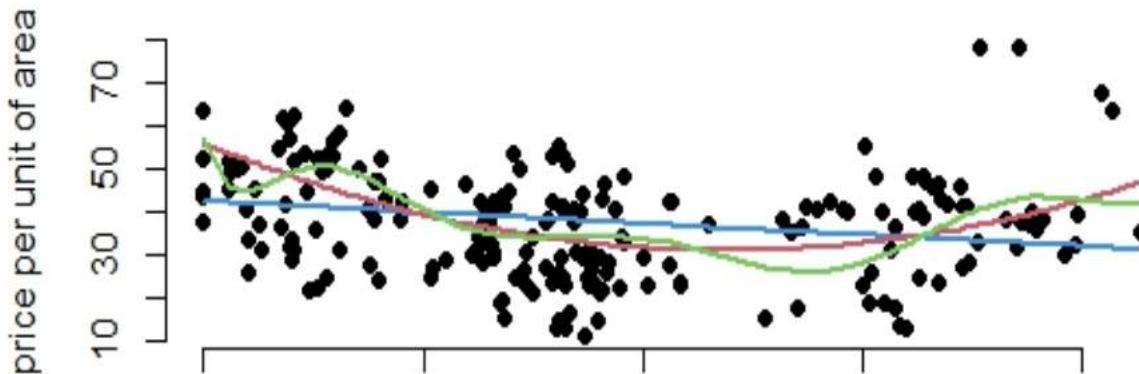


Figure 7.5 A scatter plot of house prices (y-axis) and house ages (x-axis) for a training and a test sample of 201 observations each.

Notes: The data have been fitted using a linear regression (blue line), a quadratic regression (red line), and a ninth-degree polynomial (green line). Panel A shows the training sample that has been used to estimate the parameters of the linear, quadratic, and polynomial regressions. Panel B plots the estimated regression lines against the data belonging to the test sample.

Figure 7.6 illustrates another example of how underfitting and overfitting can manifest themselves. Here, a single feature is plotted on the x-axis and an output variable plotted on the y-axis. The left panel shows a linear regression fit to the data, which is clearly insufficient to describe the series and will give rise to predictions that are highly biased. The center panel shows the result of applying a high-order polynomial fit. This line contours perfectly with the training set but is evidently overfit. The right panel shows a quadratic polynomial, which has a better balance between over- and underfitting.

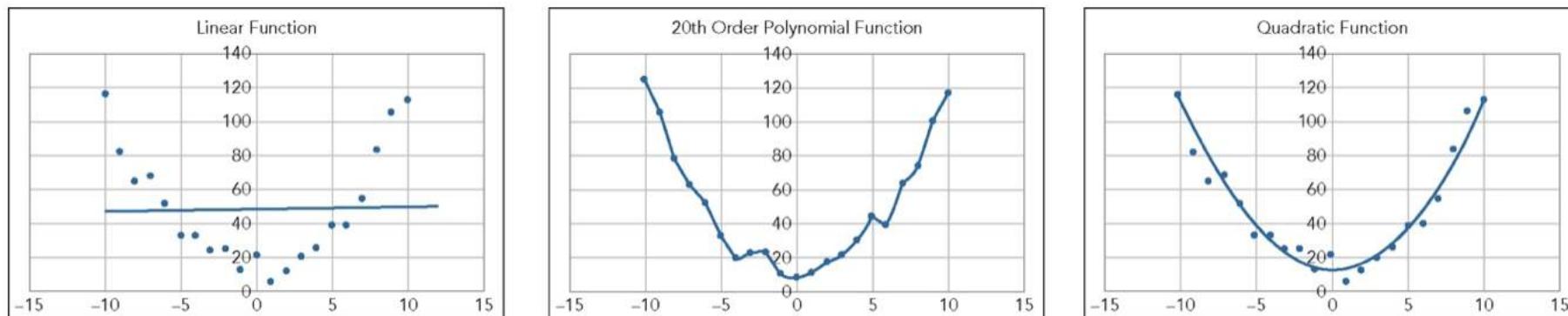


Figure 7.6 Three line fits to a set of points. The left panel shows a linear regression, the center panel shows a 20th order polynomial, and the right-hand panel shows a quadratic equation.

Overfitting is a particularly likely and severe problem with neural networks. The very basic neural network in the example presented in Figure 4.10 in Chapter 4 has 10 parameters. But often, there are several hidden layers and many more nodes per layer. This leads to an enormous number of parameters and the likelihood that there will be overfitting unless specific steps are taken to avoid it.

One approach to limit the chances of overfitting is to carry out calculations for the validation data set at the same time as the training data set. As the algorithm steps down the multi-dimensional valley, the objective function will

improve for both data sets, but at some stage, further steps down the valley will start to worsen the value of the objective function for the validation set while still improving it for the training set. This is the point at which the gradient descent algorithm should be stopped because further steps down the valley will lead to overfitting the training data and therefore poor generalization and poor predictions for the test sample.

³ Reprinted with permission from Scott Fortmann-Roe

⁴ The choice to split the sample into two portions with an equal number of observations is for ease of exposition. As discussed above, in machine learning applications, the sample is generally split into three sets with the training sample typically being larger.

7.3.4 Prediction Accuracy Versus Interpretability

There is another often unrecognized trade-off between a model's prediction accuracy and its interpretability (or lack of thereof). Machine learning models are often highly complex and heavily parametrized, so that they have often been accused of being "black boxes."⁵ More flexible models often deliver more accurate predictions (with the caveats discussed above concerning overfitting), as they can generate a wider range of shapes for the function that maps the features to the outcome. Therefore, they can fit the highly complex and nonlinear patterns in real-world data. However, these models often lack interpretability.

The linear regression model is often inadequate to model the complex nature of real-world relationships between the predictors and the target variable. Yet, its popularity among financial economists remains unchallenged, thanks to its ability to deliver an easy-to-understand relationship between the predictors and outcome that resonates with financial theory. Generally, less flexible but more interpretable models are preferred when the goal is to investigate causal relationships. In contrast, more flexible models tend to be the obvious choice when the goal is to make accurate predictions.

⁵ Unlike some other machine learning models, decision tree models are highly interpretable.

7.4 Regularization

The stepwise selection methods discussed in Chapter 3 add or remove predictors to a regression with the aim of finding the combination that maximizes the model performance. An alternative is to fit a model on all m features but using a regularization technique that shrinks the regression coefficients towards zero. Regularization can be used for standard linear regression models such as those discussed above and for many other machine-learning models discussed in subsequent chapters. It is usually a good practice to normalize or standardize the data beforehand using one of the methods discussed in Chapter 1.

The two most common regularization techniques are ridge regression and least absolute shrinkage and selection operator (LASSO). Both work by adding a penalty term to the objective function that is being minimized. The penalty term is the sum of the squares of the coefficients in ridge regression and the sum of the absolute values of the coefficients in LASSO. Regularization can simplify models, making them easier to interpret, and reduce the likelihood of overfitting to the training sample.

7.4.1 Ridge Regression

Suppose that we have a dataset with N observations on each of m features in addition to a single output variable y and, for simplicity, assume that we are estimating a standard linear regression model with hats above parameters denoting their estimated values. The relevant objective function (referred to as a loss function) in ridge regression is:

$$L = \frac{1}{N} \sum_{i=1}^N \left(\hat{y}_i - \hat{\beta}_0 - \hat{\beta}_1 x_{1i} - \hat{\beta}_2 x_{2i} - \cdots - \hat{\beta}_m x_{mi} \right)^2 + \lambda \sum_{j=1}^m \hat{\beta}_j^2 \quad (7.24)$$

The first sum in this expression is the usual regression objective function (i.e., the residual sum of squares), and the second is the shrinkage term that introduces a penalty for large-slope parameter values (of either sign). The parameter λ controls the relative weight given to the shrinkage versus model fit, and some experimentation is necessary to find the best value in any given situation. Parameters that are used to determine the model but are

not part of a model are referred to as *hyperparameters*. In this case, λ is a hyperparameter and $\hat{\beta}$ s are model parameters.

7.4.2 LASSO

LASSO is a similar idea to ridge regression, but the penalty takes an absolute value form rather than a square:

$$L = \frac{1}{N} \sum_{i=1}^N \left(\hat{y}_i - \hat{\beta}_0 - \hat{\beta}_1 x_{1i} - \hat{\beta}_2 x_{2i} - \cdots - \hat{\beta}_m x_{mi} \right)^2 + \lambda \sum_{j=1}^m |\hat{\beta}_j| \quad (7.25)$$

Whereas there is an analytic approach to determining the values of the β s for ridge regression, a numerical procedure must be used to determine these parameters for LASSO because the absolute value function is not everywhere differentiable.

Ridge regression and LASSO are sometimes known, respectively, as *L2* and *L1* regularization due to the order of the penalty terms in these methods. There are key differences between them:

- Ridge regression (*L2*) tends to reduce the magnitude of the β parameters, making them closer to, but not equal to zero.
- This simplifies the model and avoids situations in which for two correlated variables, a large positive coefficient is assigned to one and a large negative coefficient is assigned to the other. LASSO (*L1*) is different in that it sets some of the less important β estimates to zero.
- The choice of one approach rather than the other depends on the situation and on whether the objective is to reduce extreme parameter estimates or remove some terms from the model altogether.
- LASSO is sometimes referred to as a *feature selection* technique because *de facto* it removes the less important features by setting their coefficients equal to zero. As the value of λ is increased, more features are removed.

Ridge regression and LASSO can be used with logistic regression. Maximizing the likelihood is equivalent to minimizing its negative. Therefore, to apply a regularization, we add λ times the sum of the squares of the

parameters or λ times the sum of the absolute values of the parameters to the negative of the expression for the log-likelihood. Then the objective would be to find the values of the parameters that jointly minimize this composite of the negative of the log-likelihood and the sum of the absolute values of the parameters.

7.4.3 Elastic Net

A third possible regularization tool is a hybrid of the two above, where the loss function contains both squared and absolute-value functions of the parameters:

$$L = \frac{1}{N} \sum_{i=1}^N \left(\hat{y}_i - \hat{\beta}_0 - \hat{\beta}_1 x_{1i} - \hat{\beta}_2 x_{2i} - \cdots - \hat{\beta}_m x_{mi} \right)^2 + \lambda_1 \sum_{j=1}^m \hat{\beta}_j^2 + \lambda_2 \sum_{j=1}^m |\hat{\beta}_j| \quad (7.26)$$

By appropriately selecting the two hyperparameters (λ_1 and λ_2), it is sometimes possible to obtain the benefits of both ridge regression and LASSO: reducing the magnitudes of some parameters and removing some unimportant ones entirely.

7.4.4 Regularization Example

Suppose that we were interested in running a regression of a time-series of stock index returns on a set of Treasury yields for different maturities using the data from the PCA example in Chapter 1 as features. As mentioned, the features are usually rescaled before using ridge or LASSO. But in this case, the magnitudes are similar and so, to keep the example simple, we will skip the rescaling step.

The results presented in [Table 7.4](#) show that an ordinary least squares (OLS) regression provides β parameters that are quite large in magnitude, with some β s having a sign that is opposite to what is expected. This indicates

that the features in this model are highly correlated. The ridge regressions reduce the magnitude of the parameters, with the higher value of λ shrinking them more. Some of the β s changed signs in going from OLS to Ridge regression. LASSO, on the other hand, reduces some coefficient values to zero. When $\lambda = 0.1$, only one coefficient (plus the intercept) is non-zero with a negative sign, indicating an inverse relationship between stock returns and treasury yields.

Conducting a regularized regression effectively requires selecting the hyperparameter carefully. Often, this involves choosing a value of λ that produces a model that is easy to interpret while still producing accurate forecasts. The data can be split into a training set, validation set, and test set as explained in Chapter 1. The training set is used to determine the coefficients for a particular value of λ . The validation set is used to determine how well the model generalizes to new data, and the test set is used to provide a measure of the accuracy of the chosen model. Sometimes, the simpler models produced using regularization generalize better than the original OLS linear regression model.

Table 7.4 OLS, Ridge, and LASSO Regression Estimates. An illustration of ridge regression and LASSO applied to a regression containing highly correlated features, with two different hyperparameter values.

| Feature | OLS | Ridge, $\lambda = 0.1$ | Ridge, $\lambda = 0.5$ | LASSO, $\lambda = 0.01$ | LASSO, $\lambda = 0.1$ |
|-----------|--------|---------------------------|---------------------------|----------------------------|---------------------------|
| Intercept | 5.17 | 2.67 | 2.46 | 2.61 | 2.39 |
| USTB1M | -23.22 | -6.55 | -2.00 | -1.13 | 0 |
| USTB3M | 50.64 | 10.00 | 2.45 | 1.35 | 0 |
| USTB6M | -37.64 | -3.82 | -0.51 | 0 | 0 |
| USTB1Y | 11.00 | 0.70 | 0.40 | 0 | 0 |
| USTB5Y | -5.55 | -1.75 | -1.41 | -1.22 | -0.71 |
| USTB10Y | 9.13 | 0.57 | -0.11 | 0 | 0 |
| USTB20Y | -5.88 | -0.08 | 0.36 | 0.14 | 0 |

7.5 Cross-validation and Grid Searches

The methods presented so far have concentrated on training a model using the data set, validating it, and then testing it. This section presents more advanced techniques for developing machine learning models which use the model fitting methods discussed so far repeatedly in a systematic manner as they search for models with better fit or to identify optimal values of parameters that will result in better models. The motivation for these techniques comes from the necessity of making efficient use of available data and to identify optimal values for hyperparameters such as the regularization term in regressions or the number of layers in a neural network, so computational costs can be optimized.

7.5.1 Cross-Validation

So far, we have discussed fitting a model, after partitioning the available data set into training, validation, and test samples. Ideally, the available dataset will be sufficient to allow for reasonably sized training, validation, and test samples. But sometimes, that will not be the case. For example, suppose that we only have a whole dataset of 100 instances. To split this into three sub-samples in the conventional way might entail a training sample of 70 instances and just 15 for validation and 15 for testing. In such circumstances, cross-validation is a technique that can be deployed to use the data more efficiently. It involves combining the training and validation data into a single sample, with only the test data held back. This means that there is effectively not a separate validation sample, only a combined sample, which we now call the training sample. Then, this training data are split into equally sized sub-samples, with the estimation being performed repeatedly and one of the sub-samples being left out each time.

The technique known as k -fold cross-validation splits the total data available, N , into k samples, and it is common to choose $k = 5$ or 10 . Suppose for illustration that $k = 5$. Then the training data would be partitioned into five equally sized, randomly selected sub-samples, each comprising 20% of that data. If we define the sub-samples k_i ($i = 1,2,3,4,5$), the first estimation would use samples k_1 to k_4 with k_5 left out. Next, the estimation

would be repeated with sub-samples k_1 to k_3 and k_5 , with k_4 left out, and so on. This situation is illustrated in [Figure 7.7](#) with the training folds in green and the validation folds in blue. In this case, $k = 5$, there will be a 5×5 framework. At the end, there will be $k = 5$ “validation samples” (one for each iteration) that can be averaged to determine the model’s performance.

| | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
|-------------|------------|------------|------------|------------|------------|
| Iteration 1 | Training | Training | Training | Training | Validation |
| Iteration 2 | Training | Training | Training | Validation | Training |
| Iteration 3 | Training | Training | Validation | Training | Training |
| Iteration 4 | Training | Validation | Training | Training | Training |
| Iteration 5 | Validation | Training | Training | Training | Training |

Figure 7.7 Five-fold cross-validation

A larger value of k will imply an increased training sample size, which might be valuable if the overall number of observations is low. The limit as k increases would be $k = N$, which would correspond to having as many folds as the total number of data points in the training set. This situation is known as N -fold cross-validation, *jack-knifing*, or *leave-one-out cross-validation* (LOOCV).

Cross-validation represents a very resourceful use of the data, but it has at least two disadvantages:

- Using LOOCV will increase the size of the matrix in [Figure 7.7](#) to $N \times N$, which maximizes the sizes of the training and validation samples but will be computationally expensive as the data are trained at each of the N iterations. (Remark: For linear models, LOOCV is computationally inexpensive because of the updating formula, i.e., the Sherman–Morrison–Woodbury formula.) For nonlinear models, cross-validation exercises always increase the computational costs, which grow with k .
- For unordered data, the points allocated to each fold will usually be selected randomly. This implies that k -fold cross-validation cannot be used when the data have a natural ordering. For example, if they

are time-series observations, where there is a need to preserve the order and where the validation sample would usually comprise points that are chronologically after the training data. In this case, the appropriate framework would be to use a rolling window.

7.5.2 Stratified Cross-validation

When the overall sample is very small, there is a heightened risk that one or more of the training or validation sub-samples will, purely by chance, comprise a set of datapoints that are atypical compared with the other sub-samples. Some classes or types of data will be overrepresented in the training sample and underrepresented in the validation sample and *vice versa* for the other classes or types of data. The use of cross-validation will mitigate this issue to some extent, but an extreme case of it could cause even more severe problems. Specifically, if the output data are categorical but unbalanced between categories, it might be the case that there are no instances of one or more categories in one or more of the sub-samples. For instance, suppose that we have $N = 100$ sample observations on whether a car loan borrower defaulted (=1) or repaid their loan (=0), with only ten customers who defaulted. If we use cross-validation with $k = 10$, it is likely to be the case that for some of the ten iterations, there will be no defaulting customers in the validation sample, which will lead to a distorted evaluation of the models applied to the validation sample in those cases.

A potential solution to this problem is to use stratified k -fold cross-validation. In that case, instead of drawing k samples (without replacement) to comprise the validation data, the positive and negative outcomes would be sampled separately in proportion to their presence in the overall sample. So if, for example, $k = 10$ and we have $N = 100$ with 90 non-defaults and 10 defaults, we would select nine non-defaults and one default at random from the overall sample, so that each of the 10 folds would contain the same 9:1 ratio of the two classifications. This approach guarantees that each class is represented to an equal degree in all training and validation samples.

An alternative way to deal with imbalanced classes is to generate further artificial instances of the minority (underrepresented) class, using what is known as SMOTE (Synthetic Minority Over-sampling Technique). Or to use an asymmetric loss function that puts more weight on any incorrect predictions for this

class. However, these approaches are more complex than those described above and hence are not considered further.

7.5.3 Bootstrapping

A further variant on k -fold cross-validation is to use a bootstrap. Bootstrapping is a simulation technique where new data distributions are created by sampling with replacement from the original data. In this context, it would involve, for each iteration, drawing a sample of size N (the combined size of the training and validation sample) with replacement. It is highly likely that this sample will contain some instances more than once from the original sample and some instances will not appear at all – typically, around a third of the original data will not be sampled in each iteration⁶.

Those instances not appearing in the bootstrapped training sample (called *out-of-bootstrap* data) then comprise the validation sample for that iteration. A large number of iterations would be performed (10,000 or more if computational resources permit) and the results averaged over the iterations as for k -fold cross-validation.

Cross-validation and bootstrapping represent more efficient ways to deal with the data than an arbitrary separation between training and validation sets, because, effectively, every observation appears in both the training and validation samples for different folds. Cross-validation is also straightforward to implement, but a disadvantage is that it might be computationally expensive if the model is complex, or the number of folds is large, or the total sample size is large. If there are k folds and h different possible values for the hyperparameter to consider, this will involve estimating kh separate models each for a sample of size N ($1 - 1/k$). That could be computationally infeasible. Bootstrapping with a large number of iterations will be even more computationally demanding, although recent advances in computing have made this less onerous than previously.

⁶ Technically, the expected fraction of data points not appearing in the bootstrapped training sample is $1/e \approx 37\%$, where e is the exponential number (2.72...) if the sample is large.

7.5.4 Grid Searches

The purpose of cross-validation might be to determine the optimal value of a hyperparameter. To do this, the researcher might use a grid search procedure, which involves selecting a set of possible parameter values. To illustrate, suppose that the model under study involves specifying one hyperparameter, λ . This might, for example, be the hyperparameter that controls the strength of the penalty term in a Lasso regularization. Assume that the researcher determines that a range of 0 to 100 is plausible to investigate, with a step size of 1. Using 5-fold cross-validation to determine the most appropriate value of λ could be achieved using the following steps:

1. Separate the composite training sample into five randomly assigned sub-samples.
2. Set $\lambda = 0$.
3. Perform the following operations:
 - a. Combine four of the sub-samples and estimate the model under study on that composite.
 - b. Using the remaining subsample, calculate a performance measure, such as the percentage of correct classifications or the mean squared error of the predictions.
4. Repeat steps 3a and 3b for the other four combinations of the sub-samples.
5. Calculate the average of the performance measure across the five validation folds.
6. Add one to λ , and if $\lambda \leq 100$, repeat steps 3 to 5, otherwise proceed to the next step.
7. There will now be 101 performance statistics: one for each value of λ (0,100). Select the optimal value of λ (call this λ^*) corresponding to the best value of the performance statistic.
8. Perform one final estimation of the model, this time using the entire training sample with the hyperparameter set to λ^* .

Constructing a grid search framework has some drawbacks. A first issue is that the researcher may have no idea of even the scale of the hyperparameter, so a power scale might need to be used for λ , such as 10^{-1} , 10^0 , 10^1 , $10^{10}, \dots$ It would be possible to search over a coarse grid, including relatively few points over a wide range, but that could leave the best hyperparameter value from the grid search still a long way from the optimal value. Even getting close to the latter using a more refined grid could impose a vast computational burden, but it is becoming

less of a concern with the recent advances in computing technologies. A second problem is that searching over too many grid points is another manifestation of overfitting and could lead to weaker test sample performance. An alternative to grid search for hyperparameter selection would be to use random draws. This could reduce the computational time significantly and seems to work surprisingly well compared with more structured approaches. But if the researcher is unlucky, it could be that none of the randomly selected hyperparameter values come close to the optimum.

Appendix 7.A Genetic Algorithms

Another family of alternative approaches for machine learning model parameter optimization is based on *genetic algorithms* (GAs). These techniques, sometimes known as *evolutionary algorithms*, apply thinking from evolutionary biology that capture the fundamental aspects of how populations of animals evolve over the very long term according to Darwinian principles to be more resilient to hazards in their environment. Hence, they are loosely analogous to the genetic processes of reproduction and “survival of the fittest”. GAs have found widespread applicability for the estimation of parameters or fine tuning hyperparameters in decision trees, support vector machines, and neural networks. They can also aid feature selection.

GAs treat each individual parameter as a chromosome, and combinations of parameters are people, with all possible parameter combinations considered to be the human population. The parameters are optimized over many generations. The process is initialized by establishing a first-generation population of randomly assigned parameter combinations. Then, at each generation:

1. We define a “fitness” measure (known in GA as a *fitness function*) in terms of how good the optimization is for that combination of parameters – this could be, for instance the RSS (minimizing) or the value of a likelihood function (maximizing).
2. Each individual person (set of parameters) is entered into a “mating pool” and two people (sets of parameters) are selected at random to combine, but with the selection process such that individuals (sets of parameters) having high levels of fitness (better model fit) are more likely to be selected.

3. The genetic information (parameter values) is combined between these two individuals to create offspring (i.e., new parameter combinations). This is termed “crossover” or “recombination”.
4. Random changes (known as “mutations”) are also made to the parameter values at the time of combination to create more diversity in the population (estimates). A hyperparameter controls the extent of mutation.

This process continues until either a pre-specified number of generations has been reached, or the improvement in the fitness of the fittest individual (i.e., the best parameter combination so far) falls below a threshold and the solution has converged.

Genetic algorithms constitute a very unstructured approach to parameter optimization. They do not require gradient calculations, and so they are particularly useful for very complex models where solution using gradient-based methods would be challenging. GAs are more robust with respect to local optima than more conventional techniques because they have the potential to search over a wider range of the parameter space, although they can still suffer from this problem when “long-term fitness” is dominated by “short-term fitness” and parameter combinations that maximize the latter are selected over those that might have maximized the former. GAs also do not require estimates of the derivatives of the loss function with respect to the parameters, which means that they can be employed in a variety of contexts and used with complex model structures.

In practice, encoding parameter values into strings (chromosomes) efficiently could pose challenges. If there does not exist an efficient encoding scheme, the length of the chromosomes could be large, and it would make GA to be an infeasible choice of optimization algorithm. GAs can demand considerable computational resources because the required number of iterations could be large, and for each iteration the fitness level must be calculated for the whole population of parameter combinations.

Supervised Learning - Model Estimation: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

7.1 Define the term, “bias-variance trade-off.”

The trade-off between choosing a model that is not complex enough to capture the true nature of the relationship between the outcome and the features (underfitting) and choosing a model that is too complex and will model the random noise in the training sample (overfitting) is generally known as the bias-variance trade-off.

7.2 Explain how the gradient descent algorithm works.

The gradient descent algorithm is designed to minimize an objective function. By first setting the starting values for the weights, the algorithm computes the initial value of the gradient of the function. Then the values of the weights are updated by going one small step in the direction of the gradient. Such a series of computations is repeated until convergence (i.e., the magnitude of the gradient is smaller than a prespecified threshold, or the change in the objective function's value is within a prespecified threshold) or the maximum number of iterations is reached.

7.3

- A. Why does ordinary least squares estimation (OLS) minimize the sum of the squared residuals rather than the sum of the residuals?

Some residuals will be positive (actual value above the fitted line) and others will be negative (actual value below the line), and if we sum these residuals, they will cancel out. Squaring the residuals ensures that the positive and negative values do not cancel out. In fact, the sum of the residuals is always zero for the optimal choice of parameter values.

- B. When is it not possible to use OLS for parameter estimation?

OLS cannot be applied in cases where the target is a binary variable instead of a continuous variable.

- 7.4 Given that it is applicable to a wider range of model classes, why do we not typically use maximum likelihood estimation (MLE) to estimate the parameters of linear regression models?

Maximum likelihood estimation (MLE) is a more general and flexible method than OLS. But MLE is less tractable – in other words, it is inherently more complex to understand, and parameter estimation is also more complex. Both methods give the same optimal coefficients in linear regression models, but OLS is a simpler method. Hence, it is usually preferred to use OLS when the model structure allows it.

- 7.5 What is the role of the learning rate η in determining the weights in a neural network?

η is a hyperparameter that regulates the speed of adjustment of the weights. If η is too small, the learning can take considerable time; on the other hand, if η is too large, the algorithm may fail to achieve convergence. A common solution is to start with a large η but then let it decay as the learning process progresses. Popular choices are to subject the learning rate to exponential or inverse decay.

- 7.6 What is the vanishing gradient problem?

A vanishing gradient is a problem that tends to occur especially in deep networks, when many hidden layers are present. It is a consequence of the application of the chain rule for derivatives when the gradients are small numbers (between 0 and 1). When a long chain of small

numbers is multiplied together, the gradient quickly becomes very close to zero (it vanishes). An opposite problem is an exploding gradient, which is the result of multiplying large numbers.

7.7

A. Explain the benefit of regularization for regression models and how it works.

Regularization is useful for several different types of models—both linear regression and machine learning—where there are several highly correlated features that make coefficient determination difficult. For instance, where coefficient estimates are offsetting one another to some extent, and are unstable in the face of minor changes in the specification. Regularization works by adding a penalty term to the loss function (e.g., the residual sum of squares) that penalizes the model for including large parameter values of either sign. Depending on the nature of the penalty term (see part b of this question), some parameters are either shrunk toward zero or set to zero. This process will make the fitted model more parsimonious, which will usually improve its performance when applied to a validation or test data set.

B. How do LASSO and ridge regression differ?

LASSO and ridge regression are identical in spirit and approach; the only difference is the penalty term. In LASSO, this takes the form of the sum of the absolute values of the coefficients (the so-called L1 measure), while in ridge regressions it is the sum of the squared coefficients (the L2 measure). LASSO can set coefficients to zero, whereas ridge regression will simply push their values towards zero.

7.8 What is “overfitting” in the context of a machine learning model, and how could it be detected?

Overfitting is a situation where a model fits not only to the signal in the training set data, but also to the noise. In such circumstances, although the training sample fit might be very good, the estimated model will not generalize well to the validation or test sets and so the test sample predictions would be poor.

Chapter 8: Supervised Learning - Model Performance Evaluation

Learning Objectives

Previous chapters mentioned the concept of “model evaluation,” specifically in the context of cross-validation and the fine-tuning of hyperparameters. This chapter formalizes these concepts by introducing metrics that can be used to evaluate the individual performance of a model or for comparison across models. A distinction is made

between the measures used to evaluate the performance of a model when the output is a continuous variable or a discrete variable.

After completing this chapter, you should be able to:

- Discuss metrics used to evaluate the performance of a model when the outcome variable is continuous.
- Evaluate the performance of a classification model using a confusion matrix and related metrics.
- Explain the relationship between true and false positive rates and how this trade off can be illustrated using the receiver operating curve (ROC).

8.1 Model Evaluation When the Output Is Continuous

When the output is a continuous variable (e.g., a return or yield forecast), the most common measure of the predictive ability of a model is the *mean squared error* (MSE), sometimes referred to as the *mean squared forecast error* (MSFE). This is obtained by computing the average of the squares of the difference between the observed values and the predicted ones:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (8.1)$$

where y_i is the true value of the outcome for observation i and \hat{y}_i is its model prediction and N is the number of observations in the test sample, i.e., the number of predictions. Another measure of predictive ability is root mean square error (RMSE), which is the square root of MSE:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} \quad (8.2)$$

Because the forecast errors are squared, the MSE scales with the square of the units of the data, whereas the RMSE scales with the units of the data and so is easier to interpret.

Nothing prevents us from computing the MSE over the training sample. However, we are generally more interested in computing performance measures over the validation sample (to tune the model's parameters) or over the test sample (to evaluate the model's performance over unseen instances) than on the training sample. These would also be known as out-of-sample forecasts. Often, models overfit to the training sample and examining a model's accuracy in predicting observations that it has already seen and used to determine the parameters is not a true test of its performance. As this chapter is about model performance, we will generally assume that we are computing the measures of forecast accuracy over the test sample, which comprises observations not used in determining the parameter estimates or tuning the hyperparameters.

It is apparent that MSE measures how close on average the predictions are to the actual data and the square is taken to remove the sign from negative distances, which occur when the prediction overestimates the target value. Taking the squares ensures that the positive and negative forecast errors do not cancel out. The closer the predictions are to the target, the more accurate the model is. Therefore, among a set of different predictive models, the most accurate is the one with the lowest MSE in the test sample.

To understand the mechanics of MSE computation, we reconsider the regression model estimated in Chapter 7 to predict the salary of a sample of notional bank employees based on their experience:

$$\widehat{\text{salary}}_t = 16.88 + 1.06 \times \text{experience}_t \quad (8.3)$$

We now evaluate the performance of this model using the test sample reported in [Table 8.1](#). Note that this is a test sample of observations not used to estimate the model parameters. First, we obtain the difference between the predictions and the observed values, reported in the fifth column of the table. Then, we square those differences to obtain the values in the sixth column. A MSE equal to 43.1 is obtained as the sum of the observations in the sixth column divided by ten (the number of observations in the test sample). Sometimes the RMSE (in this case equal to $\sqrt{43.1} = 6.57$) is reported instead of the MSE as the former is on the same scale as the prediction.

Table 8.1: Illustration of calculation of MSE for a test sample of bank employees

| Observation, <i>i</i> | Experience, <i>x_i</i> | Actual salary, <i>y_i</i> | Predicted salary <i>ŷ_i</i> | <i>y_i–ŷ_i</i> | (<i>y_i–ŷ_i</i>)² |
|----------------------------------|---|--|---|--|--|
| 1 | 4 | 20.15 | 21.12 | -0.97 | 0.94 |
| 2 | 6 | 21.48 | 23.24 | -1.76 | 3.10 |
| 3 | 9 | 31.88 | 26.42 | 5.46 | 29.81 |
| 4 | 10 | 32.88 | 27.48 | 5.40 | 29.16 |
| 5 | 21 | 49.92 | 39.14 | 10.78 | 116.21 |
| 6 | 25 | 57.46 | 43.38 | 14.08 | 198.25 |
| 7 | 3 | 21.13 | 20.06 | 1.07 | 1.14 |
| 8 | 1 | 12.18 | 17.94 | -5.76 | 33.18 |
| 9 | 2 | 19.47 | 19.00 | 0.47 | 0.22 |
| 10 | 7 | 28.66 | 24.30 | 4.36 | 19.01 |

Although it remains the most widely used measure of performance in practice (also because of its connection with the least squares estimation technique, as discussed in Chapter 7), MSE has some drawbacks. The most notable one is that, because of taking the squares of the distances between actual data and the predictions, MSE overweights large deviations from the observed values. This implies that, when MSE is used to pick the best model, one that is generally very accurate but displaying a few occasional large deviations from the observed value might be less preferable (i.e., have a higher MSE) than a model that is in general less accurate but where no prediction largely under- or overestimates the true value. In other words, MSE is highly sensitive to outliers in the test sample.

To overcome this limitation, we could use the mean absolute error (MAE), which is like the MSE but instead of taking the squares of the distances between the predictions and the actual data, it uses the absolute value:

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| \quad (8.4)$$

RMSE and MAE are both unnormalized measures. If we consider the salary example above, this means that if we underestimate the salary by \$2,000, this will contribute to the RMSE and MAE in the same way irrespective of the level of salary that we are predicting. In other words, predicting a salary of \$148,000 when it is \$150,000 is treated symmetrically to predicting a salary of \$10,000 when it is \$12,000. However, in some practical situations, the first error might be considered much less important than the second one, as it represents about 1% of the total salary, whereas in the second situation the error is more than 10% of the salary.

A measure that considers the relative importance of errors relative to the scale of the variable being predicted is the mean absolute percentage error (MAPE). This can be computed as:

$$MAPE = 100 \times \frac{1}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (8.5)$$

where the multiplication by 100 is to express the metric as a percentage of the actual value. Similarly, the MSE or RMSE can be expressed as a percentage of the actual value by dividing the squared forecast errors by the corresponding actual values. [Table 8.2](#) extends the calculations in [Table 8.1](#) to the case of MAE and MAPE.

Table 8.2: Illustration of calculation of MAE and MAPE for a test sample of bank employees

| Observation, <i>i</i> | Experience, <i>x_i</i> | Actual salary, <i>y_i</i> | Predicted salary <i>ŷ_i</i> | <i>y_i - ŷ_i</i> | <i> y_i - ŷ_i </i> | | |
|--------------------------|-------------------------------------|--|--|--------------------------------------|--|-------|------|
| 1 | 4 | 20.15 | 21.12 | -0.97 | 0.97 | -0.05 | 0.05 |
| 2 | 6 | 21.48 | 23.24 | -1.76 | 1.76 | -0.08 | 0.08 |
| 3 | 9 | 31.88 | 26.42 | 5.46 | 5.46 | 0.17 | 0.17 |
| 4 | 10 | 32.88 | 27.48 | 5.40 | 5.40 | 0.16 | 0.16 |

| Observation, i | Experience, x_i | Actual salary, y_i | Predicted salary \hat{y}_i | $y_i - \hat{y}_i$ | $ y_i - \hat{y}_i $ | | |
|--|---|--|--|-------------------|---------------------|-------|------|
| 5 | 21 | 49.92 | 39.14 | 10.78 | 10.78 | 0.22 | 0.22 |
| 6 | 25 | 57.46 | 43.38 | 14.08 | 14.08 | 0.25 | 0.25 |
| 7 | 3 | 21.13 | 20.06 | 1.07 | 1.07 | 0.05 | 0.05 |
| 8 | 1 | 12.18 | 17.94 | -5.76 | 5.76 | -0.47 | 0.47 |
| 9 | 2 | 19.47 | 19.00 | 0.47 | 0.47 | 0.02 | 0.02 |
| 10 | 7 | 28.66 | 24.30 | 4.36 | 4.36 | 0.15 | 0.15 |

The MAE is computed by averaging the absolute differences reported in the sixth column. The calculations yield a MAE of 5.01. The MAPE is computed by taking the averages of the values in the last column, which have been obtained by dividing the differences by the actual salary value and taking the absolute value of this ratio. The calculations yield a value of 0.16 that is multiplied by 100 to obtain a MAPE of 16%. Of all the forecast accuracy measures introduced so far in this chapter, MAPE is the most intuitive. The figure of 16% can be interpreted as implying that the average forecast error is 16% of the actual salary.

When used to evaluate alternative models, MSE, MAE and MAPE can all yield different model rankings. The choice of the measure that is most appropriate ultimately depends on the problem at hand. For instance, a risk manager is likely to evaluate the expected loss in units of dollars. On the contrary, an index manager is interested in controlling the deviation of a portfolio's percentage return (either + or -) from the index it is tracking.

8.1.1 An Example of Continuous Variable Model Performance Comparison

The performance metrics discussed above can be used to compare alternative models. Therefore, in this example, we demonstrate how they can be used to compare two different models employed to predict the house price per unit area: a simple linear regression (Model A) and a tree-based regression (Model B)¹. The sample consists of 414 observations, equally split between training and test sample. The features are house age, the distance from the closest metropolitan train station, the number of convenience stores in the area, and the longitude and latitude coordinates. The performance metrics calculated by estimating the model using the training sample and using it for predicting the output values in the test sample are reported in [Table 8.3](#). The best performing model according to each metric is denoted with an asterisk.

Table 8.3 Performance measures for two alternative models for house price prediction

| | MSE | MAE | MAPE |
|---------|--------|-------|--------|
| Model A | 94.85 | 6.56 | 18.25 |
| Model B | 77.35* | 5.41* | 14.45* |

Notably, a non-linear, tree-based regression is more accurate to predict the house price per unit area according to all performance measures in this case because the values in the second row are all smaller than the corresponding ones in the first.

¹ Please refer to section 4.1 for a discussion of this model.

8.2 Model Evaluation: Classification

Classification models generally yield two types of predictions: a continuous value, which could be a score or a probability, and a discrete value, which is the predicted class. The continuous value is typically transformed to a discrete one (the class to which an instance is assigned) using a threshold Z . For instance, suppose that we had to predict whether a firm will pay a dividend the following year (positive outcome) or not (negative outcome). We could use the logistic regression model introduced in Chapter 3 to obtain predicted probabilities of dividend payment, \hat{y} , between 0 and 1. We then predict a positive outcome if $\hat{y} \geq Z$ and a negative outcome if $\hat{y} < Z$.

Although an obvious and popular choice for the threshold Z is 0.5, it does not need to be, and a different value might be more appropriate as discussed below.

Although metrics such as the MSE discussed above could be used to evaluate the continuous prediction (of the probability), we are often interested in evaluating the discrete prediction of the outcome derived from the probability. Similarly, it is often the case that continuous predictions are distilled into discrete values for use in a decision rule. For instance, a hedge fund analyst might have used a neural network model to make time-series forecasts of future returns but wishes to turn these into an automated trading rule. The approach would be the same as for predicted probabilities: establish a threshold and generate a binary dummy variable indicating whether the prediction is above or below the threshold. In this case, the threshold might be zero so that a buy signal (1) is generated if the return is predicted to be positive and a sell signal (0) if the forecast is negative. To evaluate discrete (class) predictions, different performance measures are necessary.

When the outcome is a class, a common way to evaluate the model is through calculations based on a *confusion matrix*, which is a simple cross tabulation of the observed and the predicted classes. The main diagonal (from top left to bottom right) elements of the matrix denote cases where the correct class has been predicted, and off-diagonal elements illustrate all the possible cases of misclassification. It is easier to illustrate how the confusion matrix is constructed with reference to the case in which the outcome is binary. In this case, the confusion matrix is a 2×2 . For example, suppose that we constructed a model to calculate the probability that a firm will pay a dividend in the following year or not based on a sample of 1,000 firms, of which 600 did pay and 400 did not. We could then set up a confusion matrix such as the following:

| | | Prediction | |
|----------------|----------------------|----------------------------|-------------------------------|
| | | <i>Firm will not pay</i> | <i>Firm will pay dividend</i> |
| Outcome | <i>No dividend</i> | 279 (27.9%) – <i>TN</i> | 121 (12.1%) – <i>FP</i> |
| | <i>Pays dividend</i> | 168 (16.8%) – <i>FN</i> | 432 (43.2%) – <i>TP</i> |

The confusion matrix would have the same structure however many features were involved in the model—whatever the sample size and whatever the model—so long as the outcome variable was binary. We identify the four elements of the table as follows:

1. True negative: The model predicted a negative outcome, and it was indeed negative. (TN)
2. False positive: The model predicted a positive outcome, but it was negative. (FP)
3. False negative: The model predicted a negative outcome, but it was positive. (FN)
4. True positive: The model predicted a positive outcome, and it was indeed positive. (TP)

Based on these four elements, it is possible to compute a simple performance metric, accuracy (with calculations using the dividend example numbers in the confusion matrix above and expressed as percentages):

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} = \frac{432 + 279}{432 + 279 + 121 + 168} \times 100\% \\ &= 71.1\% \end{aligned} \tag{8.6}$$

or, alternatively

$$\begin{aligned}
Error rate &= 1 - \frac{TP + TN}{TP + TN + FP + FN} = \frac{FP + FN}{TP + TN + FP + FN} \\
&= \frac{121 + 168}{432 + 279 + 121 + 168} \times 100\% = 28.9\%
\end{aligned} \tag{8.7}$$

Accuracy, the sum of the diagonal elements of the matrix divided by the sum of all its elements, is straightforward to interpret as it simply reflects the agreement between the predicted and observed classes or equivalently, the percentage of all predictions that were correct. In this case, 71.1% of the dividend predictions were correct and 28.9% were incorrect. Similarly, the error rate is simply one minus accuracy, which is the proportion of instances that were misclassified.

Accuracy and error rate are very intuitive classification performance measures, but both metrics ignore the type of error that has been made. In other words, they do not distinguish between type I and type II errors, where the former happens when an outcome is predicted to be true when the outcome is false (false positive), and the latter when an outcome is predicted to be false when the outcome is true (false negative). In practical situations, the cost of committing each type of error is typically not the same. For instance, for a bank extending credit, misclassifying a borrower as solvent (who then goes on to default, losing the bank a lot of money) is typically more costly than the opposite (losing out on the profit from having made an additional loan to a borrower who would not have defaulted).

Additionally, error rate and accuracy are problematic as performance measures when the classes are largely unbalanced. For instance, suppose again that we want to classify a pool of borrowers as solvent (positive outcome) or insolvent (negative outcome). If 98% of the borrowers were solvent, a model that classifies 100% of the borrowers as solvent would only have a 2% error rate; however, this model would be practically useless.

To overcome the limitations discussed above, two other metrics are available, *precision* and *recall*:

$$Precision = \frac{TP}{TP + FP} = \frac{432}{432 + 121} = 78.1\% \quad (8.8)$$

$$Recall = \frac{TP}{TP + FN} = \frac{432}{432 + 168} = 72\% \quad (8.9)$$

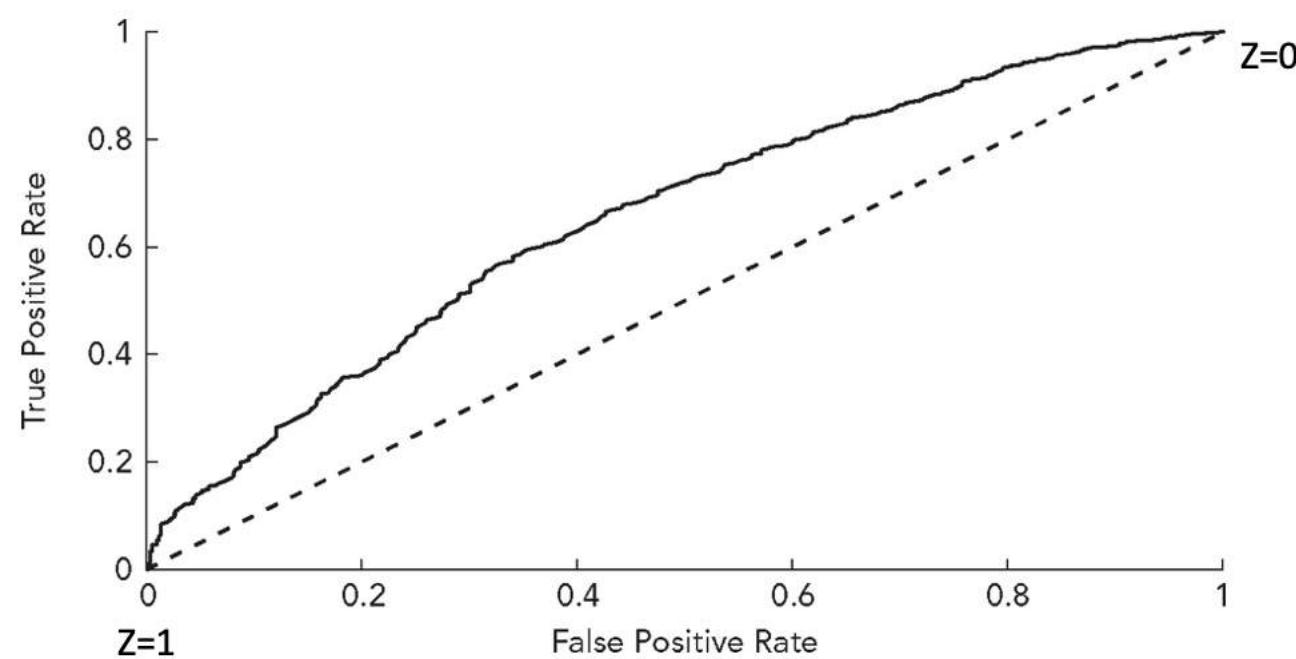
Precision is the number of correctly classified positives among all the instances that have been classified as positive. In other words, precision is the estimate of the probability that the model is right when labelling an outcome as true. Recall (also known as *sensitivity*) is the true positive rate, that is, the number of correctly classified positives over the total number of positives. It is also possible to compute the true negative rate (also known as *specificity*), which is the proportion of negative outcomes that were correctly predicted as negative. Either precision or recall can be more useful depending on the context. For instance, a bank predicting whether a borrower is solvent (positive outcome) might be more interested in precision. Conversely, an analyst predicting whether a company would pay a dividend might be more interested in recall. A further measure, known as the *F1 score*, combines precision and recall (technically, it takes the “harmonic mean” of the two) into a single measure:

$$F1 \text{ score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (8.10)$$

The *F1* score will be bounded between 0 and 1, and it will be closer to 1 if the model has both high precision and high recall, but a low score can arise from either poor precision, poor recall, or both. Like most other performance measures, *F1* is sensitive to severe class imbalances.

More generally, there is a trade off between the true and false-positive rates when setting the decision threshold, Z . As the true positive rate increases, the false-positive rate also increases. For instance, taking the example above, we can identify more dividend paying firms at the cost of also misclassifying more non-dividend paying firms as dividend paying ones. In other words, we can increase recall at the cost of decreasing precision, and vice versa.

A way to illustrate this tradeoff is the receiver operating curve (ROC), which plots the true and false-positive rates against different choices of the threshold Z , as illustrated in [Figure 8.1](#). Considering again the example of dividend paying firms, if we had chosen $Z = 0$, we would have classified all the firms as dividend paying. This implies that all the positives are accurately classified but all the negatives are misclassified so both the true and false-positive rates are 100%. At the other extreme, if $Z = 1$, all the firms are classified as non-dividend paying. Therefore, both the true and false-positive rates are 0%. For values of Z between 0 and 1, the true positive rate increases when the false-positive rate increases.



[Figure 8.1](#) A sample receiver operating curve

The greater the area under the ROC curve (referred to as *area under curve* or AUC), the better the predictions from the model. A completely accurate set of predictions gives an AUC of 1. A value of AUC equal to 0.5 corresponds to the dashed line and indicates that the model has no predictive ability. An AUC value less than 0.5 indicates that the model performs worse than randomly guessing.

The formulae for the performance metrics described above implicitly assumed only two possible outcomes (e.g., 0 or 1). Such cases are more intuitive and make for more straightforward examples. However, the formulae can be extended naturally to situations where there are several classes, such as when credit ratings are being predicted. For instance, in the multi-class case, the *Precision* measure would be calculated by summing all the true positive classifications and dividing by the sum of all the true positive and all the true negative classifications. Likewise, the other performance evaluation metrics would be generalized in a similar fashion.

8.2.1 An Example of Classification Model Performance Comparison

Suppose that we had to build a model to classify loans in terms of whether they turn out to default or repay. Several different models can be used to this end, and we want to test whether a single hidden layer feedforward neural network (see Chapter 4) outperforms a simple logistic regression (see Chapter 3). The neural network contains ten units in the hidden layer and a logistic activation function.

The confusion matrices for both models are given in [Table 8.4](#). The first panel presents the matrix for a logistic regression estimation on the training sample; the second panel presents the matrix for the logistic regression predictions on the test sample; the third panel presents the matrix for the neural network estimated on the training sample; and the fourth panel presents the matrix for the neural network predictions on the test sample. Interpreting or evaluating a neural network model is harder than for more conventional econometric models. It is possible to examine the fitted weights, looking for very strong or weak connections or where estimates are offsetting (one large positive and another large negative), which would be indicative of overfitting. However, in the spirit of machine learning, the focus is on how useful the specification is in classification using a validation sample.

Given that the same data and features have been employed for both the logistic regression and neural network, the results from the models can be compared in [Table 8.5](#). For simplicity, a threshold of 0.5 is employed, so that for any predicted probability of default greater than or equal to 0.5, the fitted value is of a default, whereas if the probability is less than 0.5, the fitted value is of no default.

Table 8.4 Confusion matrices for predicting defaults on personal loans

| Logistic regression training sample | | | |
|-------------------------------------|------------|------------|---------|
| | | Prediction | |
| | | No default | Default |
| Outcome | No default | 400 | 11 |
| | Default | 68 | 21 |
| Logistic regression test sample | | | |
| | | Prediction | |
| | | No default | default |
| Outcome | No default | 104 | 10 |
| | Default | 38 | 15 |
| Neural network training sample | | | |
| | | Prediction | |
| | | No default | default |
| Outcome | No default | 406 | 5 |
| | Default | 74 | 15 |
| Neural network test sample | | | |
| | | Prediction | |

| Logistic regression training sample | | | |
|-------------------------------------|------------|------------|---------|
| | | No default | default |
| Outcome | No default | 97 | 17 |
| | Default | 33 | 20 |

Table 8.5 Comparison of logistic regression and neural network performance for a sample of loans from the LendingClub

| | Training sample (500 data points) | | Test sample (167 data points) | |
|-----------|-----------------------------------|----------------|-------------------------------|----------------|
| Measure | Logistic regression | Neural network | Logistic regression | Neural network |
| Accuracy | 0.842 | 0.842 | 0.713 | 0.701 |
| Precision | 0.656 | 0.750 | 0.600 | 0.541 |
| Recall | 0.236 | 0.169 | 0.283 | 0.377 |

The performance summary measures show that, as expected, the fit of the model is somewhat weaker on the test data than on the training data. This result could be interpreted as slight overfitting, and it might be worth removing some of the least empirically relevant features or applying a regularization to the fitted models.

The confusion matrices shown in [Table 8.4](#) show that the classifications from the two models are more divergent than the summary measures suggested. The logistic regression predicts more defaults for the training sample, whereas the neural network predicts more defaults for the test sample. Hence the logistic regression has a higher true positive rate but a lower true negative rate for the training data, whereas the situation is the other way around for the test data. (Note that “positive event” is “default” because both models predict default.)

When comparing the logistic regression and neural network approaches, there is very little to choose between them. On the training sample, their accuracies are identical, and although the neural network performs better in terms of its precision, its recall is weaker. But when applied to the test sample, the logistic regression does better on accuracy and precision grounds, but worse on recall. Overall, these contradictory indicators illustrate the importance of fitting the evaluation metric to the problem at hand.

Supervised Learning - Model Performance Evaluation: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

8.1 Explain in words the definitions of (a) MSE; (b) MAE; (c) MAPE.

MSE, MAE, and MAPE are three different performance measures that can be used to rank models when the output variable is continuous. MSE is simply the average of the squared differences between the observed values of the target variable and the model predictions. The square is taken to eliminate the sign of the difference, as under- and over-estimating the output are penalized symmetrically. MAE is the average of the absolute value of the differences between the observed values of the target variable and their model predictions. Finally, MAPE is a relative measure of performance that divides each difference between the actual observation and the prediction by the value of the actual observation before taking the absolute value. This is a percentage measure, and it is hence multiplied by 100.

8.2 Specify the meanings of the terms true positive, true negative, false positive, false negative.

Consider a classification problem where there are only two possible outcomes, 1 (positive) and 0 (negative). A true positive is an instance that is correctly classified as positive. A false positive is an instance that is classified as positive, but it is in fact negative. A true negative is an instance that is correctly classified as negative. A false negative is an instance that is classified as negative but is in fact positive.

8.3 Explain in words the definitions of (a) accuracy and (b) error rate and discuss how the two are interrelated.

Accuracy is the percentage of correctly classified instances, and it is computed as the sum of true positives and true negatives over the total number of instances in the sample. The error rate is the percentage of incorrectly classified instances, and it is simply obtained as one minus the accuracy measure.

8.4 Discuss the main limitations of accuracy and error rate as measures of model performance.

Accuracy and error rate do not account for the difference between type I and type II errors, which can be relevant depending on the problem at hand. In addition, they fail to convey meaningful information when the classes are largely imbalanced. For example, if 98% of the instances were positive, classifying all instances as positive would produce an error rate of just 2%, but the model would hardly be useful.

8.5 Explain in words the definitions of (a) precision and (b) recall.

Precision is the estimate of the probability that a model will correctly classify an outcome as true. It is computed as the number of true positives over the number of all instances classified as positive. Recall is the proportion of correctly classified positives among all positive instances.

8.6 Explain what ROC and AUC stand for and how they could be used in making lending decisions.

ROC stands for receiver operating curve and AUC is the area under the curve. The ROC plots the true positive rate on the y-axis against the false-positive rate on the x-axis and the points on the curve emerge from varying the decision threshold. The ROC curve shows the tradeoff between the true positive rate and false-positive rate when selecting the decision threshold.

The AUC shows pictorially how effective the model has been in separating the data points into clusters, with a higher AUC implying a better model fit, and so the AUC can be used to compare between models. An AUC of 1 would indicate a perfect fit, whereas a value of 0.5 would correspond with an entirely random set of predictions and therefore a model with no predictive ability.

One possible application of the ROC and AUC would be in the context of comparing models to determine whether a loan application should be rejected or accepted. A better model would be one with a higher AUC of the test set.

8.7 A risk manager is evaluating the performance of two separate default prediction models for a sample of corporate loans in particular town. The models predict that the borrower will default or not default in the following year, which is then compared with the realized outcome as summarized in the following tables:

Model 1

| | | Predicted result | |
|---------------|------------|------------------|---------|
| | | No default | Default |
| Actual result | No default | 592 | 4 |
| | Default | 63 | 2 |

Model 2

| | | Predicted result | |
|---------------|------------|------------------|---------|
| | | No default | Default |
| Actual result | No default | 498 | 98 |
| | Default | 5 | 60 |

- A. Using the data in the tables above, calculate the true positive and true negative rates as well as the precision and accuracy of each model.

The findings classified as true or false positives or negatives are:

Model 1

| | | Predicted result |
|--|--|------------------|
| | | |

| | | | |
|-----------------------------|-------------------|-------------------|----------------|
| | | <i>No default</i> | <i>Default</i> |
| <i>Actual result</i> | <i>No default</i> | 592 TN | 4 FP |
| | <i>Default</i> | 63 FN | 2 TP |

Model 2

| | | <i>Predicted result</i> | |
|-----------------------------|-------------------|--------------------------------|----------------|
| | | <i>No default</i> | <i>Default</i> |
| <i>Actual result</i> | <i>No default</i> | 498 TN | 98 FP |
| | <i>Default</i> | 5 FN | 60 TP |

The calculations are:

Model 1:

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} = \frac{2 + 592}{2 + 592 + 4 + 63} \\ &= 89.9\% \end{aligned}$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{2}{2 + 4} = 33.3\%$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{2}{2 + 63} = 3.1\%$$

$$\begin{aligned} \text{Error rate} &= 1 - \frac{TP + TN}{TP + TN + FP + FN} \\ &= 1 - \frac{2 + 592}{2 + 592 + 4 + 63} = 10.1\% \end{aligned}$$

Model 2:

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} = \frac{60 + 498}{60 + 498 + 98 + 5} \\ &= 84.4\% \end{aligned}$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{60}{60 + 98} = 38.0\%$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{60}{60 + 5} = 92.3\%$$

$$\begin{aligned} \text{Error rate} &= 1 - \frac{TP + TN}{TP + TN + FP + FN} \\ &= 1 - \frac{60 + 498}{60 + 498 + 98 + 5} = 15.6\% \end{aligned}$$

B. Comment on the differences between the models.

Model 1 has a higher accuracy (lower error rate), but Model 2 has better precision and recall. Model 1 would probably not be preferred because it misses so many defaults. This shows that accuracy can be a misleading statistic when a data set is imbalanced (i.e., many more positive outcomes than negative outcomes, or vice versa).

Chapter 9: Natural Language Processing

Learning Objectives

Natural language processing, sometimes also known as content analysis, text mining or computational linguistics, is one of the most exciting and fast-developing applications of machine learning. NLP works with data with an unstructured, free text format to understand and analyze human language, both written and spoken. The U.S. Securities Exchange Commission was an early adopter of NLP in its effort to detect accounting fraud.

This chapter provides a comprehensive overview of NLP models, including the preparation of textual information for use in NLP models, the construction of NLP models, a comparison of non-machine learning approaches to NLP models, and how NLP model fit can be evaluated.

After completing this chapter, you should be able to:

- Discuss applications of natural language processing (NLP).
- Describe the pre-processing steps for NLP.
- Discuss the bag of words (BoW) and n-grams approaches.
- Explain how the Naïve Bayes classifier is used to categorize documents.
- Illustrate how term frequency-inverse document frequency (TF-IDF) can be used to determine the appropriate weighting to assign to words in a document.
- Describe and contrast different approaches to sentiment analysis.

Natural language processing, sometimes also known as *content analysis*, *text mining* or *computational linguistics*, is an aspect of machine learning concerned with understanding and analyzing human language, both written and spoken. Most global information has an unstructured, free text format, making NLP one of the most exciting and fast-developing

applications of machine learning. It has found numerous uses in finance. An early example was when the US Securities and Exchange Commission used the tool to detect accounting fraud. Other uses include:

- *Recognition of specific words to determine the purpose of a message.* For example, a financial institution might use an automated process initially to ask callers to the main helpline to say, in a few words, what is the purpose of their call. Then, depending on the keywords identified, an automated process would direct the person making the call to the most appropriate operator or department without the need for a person to triage the calls.
- *Categorization of a particular piece of text.* For example, a set of newswire statements could be classified depending on the news they represent using “semantic analysis”: corporate, government, human interest, environmental, social, education, or by the most relevant country they apply to.
- *Determining the sentiment of a statement.* Corporations use this application of NLP to assess from social media comments how the market is reacting to a new product or advertisement, and academic studies have examined how asset prices are affected by statements depending on whether they represent positive or negative news.
- *Assessing the readability of a document.* Organizations often commit to using plain English so that their websites and materials are as accessible as possible to their customers. The “fog index” measures the mean number of words in each sentence and the proportion of “complex” words (with more than two syllables). Other approaches also measure whether the passive voice or technical terminology has been used in communications.
- *Evaluating the similarity of two documents.* Document similarity can be used to assess the extent to which a news article or corporate announcement constitutes new information, or to determine whether the tone of successive accounting disclosures has changed.

More generally, NLP lies behind the operation of many everyday computational tools such as internet search engines and e-mail spam detection mechanisms. The main benefits of NLP over the manual reading of documents by a human are the vastly superior speed with which the former can complete the task, with no scope to miss any aspects (provided they have been built into the design). There is also a guarantee that all documents will be considered identically with no scope for biases or inconsistencies.

NLP is more challenging than modeling purely numerical data because textual data are often noisy with errors and ambiguities, particularly when dealing with information from social media or informal settings as opposed to formal communications from a firm or central bank. The same words often have several meanings depending on

the context or even the tone of voice with which they are spoken. For instance, it is hard for a machine to identify sarcasm or to decipher abbreviations that are common in everyday usage or text messages (e.g., 'u r gr8', or 'lol'). Such issues might mean that NLP fails to fully capture the essence of a particular document, although its power and flexibility are improving rapidly alongside advances in the computational capacity for storing and processing such information.

The most used NLP data sources include:

- Corporate disclosures such as mandatory filings (financial reports such as 10Q or 10K, equity or bond issue prospectuses).
- Social media posts such as generalist messages on platforms such as X (formerly known as Twitter) or Reddit and more focused financial sources such as Stocktwits or Seeking Alpha.
- News wires, newspapers and magazines including the Wall Street Journal and the Dow Jones newswire.

The documents might be available for download as a collection of pdfs using an Application Programming Interface (API) or by 'web scraping,' which uses code in a language such as Python to find and download the documents in a batch. The collection of all the documents that have been retrieved and are available for analysis by NLP is known as a *corpus*.

The remainder of this chapter is organized as follows. Section 9.1 explains how textual information is prepared for analysis in various pre-processing steps. Section 9.2 explains how NLP models are constructed, discussing how to construct vectors of word occurrences, and Section 9.3 presents a non-machine learning approach to NLP based on pre-existing dictionaries, focusing on the term frequency-inverse document frequency approach. Section 9.4 discusses machine-learning techniques for NLP. The chapter concludes with a brief discussion on how the fit of such models can be evaluated.

9.1 Data Pre-processing

Prior to processing textual information statistically, the first stage is to convert and store the data in machine-readable raw text files if they are not already in that format – for example, by extracting the text from the PDFs; or, if the files have been web-scraped, removing any Hypertext Markup Language (HTML) code or other non-textual information. For instance, HTML code will contain tags such as “

,” which indicates a paragraph. These have no content value and can be dropped. Sometimes, rather than complete removal, symbols such as emojis can be converted into a numerical representation in a process of *text encoding* and retained for later analysis. Various software tools¹ exist that can identify emojis and replace them with their written definitions or numerical values.

The initial quality of the data will depend upon its source. Newswires or central bank communications will usually be drafted in a formal and error free language, but social media or bulletin board messages will likely contain numerous spelling errors, colloquialisms, and abbreviations. It will be necessary to edit the most common repeated instances of these to retain as much useful information as possible – for example, using a spell-checker, to guard against an artificial proliferation of separate words that were intended to be the same, e.g., ‘their’ and ‘thier.’ On the other hand, spell-checkers often change already correct words (such as the names of people or places) that they do not recognize or replace misspelled words wrongly. More straightforwardly, it is worth editing the documents to expand out contractions (don’t, won’t, etc.) so that negations are correctly identified.

Once the data have been initially cleaned, the next stage involves several pre-processing steps to ensure that the text is as amenable to accurate analysis as possible. The steps are:

1. “*Tokenize*” the passage. This means separating the document into sentences at each period, question mark or exclamation mark (sentence tokenization). Then, each sentence is split into words, usually removing any punctuation, spacing, special symbols, and so forth (word tokenization). Any letters or words in capitals would all be modified to lower case.
2. “*Stop word*” removal. Stop words are those that have no informational value but that are included in sentences to make them flow and so that they are easier to follow, such as *has, a, the, also*, and so on. Precisely what constitutes a stop word will vary from one application to another depending on the purpose, because particular words may have no value in one context but convey useful information in another.

3. Replace words with their stems. This process is sometimes known as stemming, where words are replaced with their core (or ‘root’) forms to be treated equally in subsequent analysis. An example is that *disappointing* and *disappointed* would both be replaced with *disappoint*.
4. Replace words with their lemmas. This procedure is sometimes known as lemmatization, where words such as *good*, *better*, *best* are all replaced with *good*. This step is undertaken so that words expressing a very similar perspective on the topic are considered equally. For instance, the words in italics here all convey a positive sentiment and separating them would cause unnecessary complexity.
5. Feature extraction. This step involves turning the text processed using the previous steps into numeric vectors that can be analyzed by machine learning models and is sometimes also known as *text representation*. It is an important and involved part of the process, so it is discussed in detail in a separate sub-section below.

These pre-processing steps are summarized in [Figure 9.1](#).

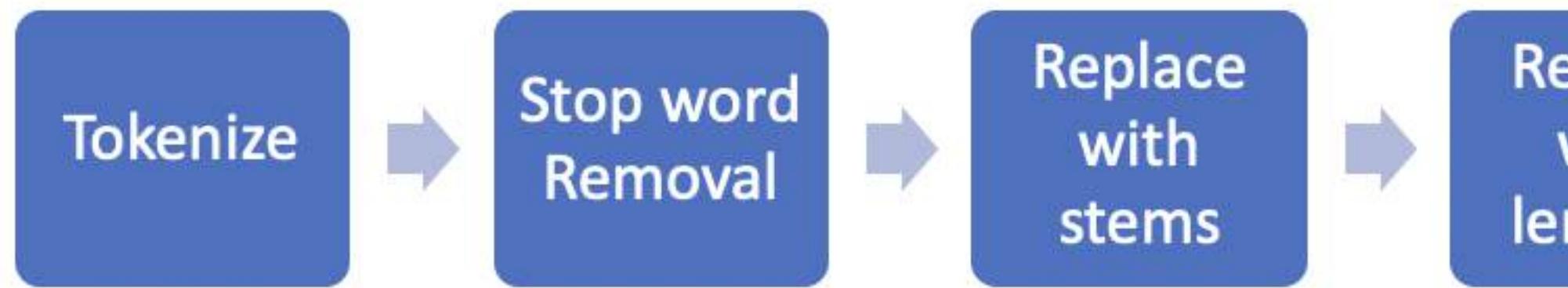


Figure 9.1 Preprocessing Steps

A few further notes are in order concerning the steps above, noting their limitations and disadvantages. Although the removal of punctuation simplifies the task at hand, we should note that in many cases, by doing so, valuable context will be lost. For instance, the phrase “Is the CEO performing well?” is written as a question and therefore, arguably, has little useful information content. Yet a textual analysis without the punctuation would conclude that

the phrase expressed a positive view owing to the presence of the terms “performing” and “well” in the same sentence without a negation word. Relatedly, an exclamation mark at the end of a sentence may denote a phrase that is written in humor or sarcasm and therefore might have less information value than a similar sentence ended with a period.

Lowercasing can lead to inaccuracies or misclassifications of words because an uppercase first letter is usually used to denote a proper noun (the name of a person, place, or firm) – for example, comparing *Reading* and *reading*. Meaning can be lost through lowercasing in other ways as well, as for example in going from *Polish* to *polish*. The errors caused by this can be addressed by labelling each word according to its class (common noun, proper noun, verb, adjective, etc.) prior to lowercasing, which is also known as part-of-speech (POS) tagging. In this case *Reading* would be classified as a proper noun and *reading* would be treated as a verb. Stemming and lemmatization are used so that similar words are treated the same as one another to simplify the analysis. However, again the simplification can lead to loss of nuance and degrees of positivity or negativity, such as the terms *good* and *outstanding*, which might be lemmatized to be the same.

It might be the case that words are so common across documents that they have little value in classification. For instance, the term “like” might convey positive sentiment but has more than one meaning and will be ubiquitous. Rather than seeking out such words individually for removal, the researcher can rank words by their frequency of occurrence across all documents and then drop, say, the top 5%.

Once these steps have been undertaken, the remaining text segment can be subject to examination. Most straightforward NLP tasks treat the processed text as a “bag of words,” which means that the ordering of the words and any linkages between them are ignored to simplify the task. It is also necessary to examine the context in which a word is used to understand the intended meaning. Techniques like n-grams (see section 9.3.2) are used for this purpose.

¹ For example, the *emoji* module is available in the Python programming language for preprocessing emojis.

9.2 NLP Models

How text documents are analyzed will depend on the task at hand. Broadly, we can separate the various techniques into two categories: heuristic approaches that primarily involve a direct analysis of the vector representations of the documents, and machine-learning approaches that use similar techniques to those discussed in previous chapters (e.g., naïve Bayes classifiers, support vector machines or neural networks). These two categories are each discussed in the following sections.

9.2.1 Feature Extraction

Whether heuristics or ML techniques are adopted, the first stage is to transform the text into numerical values. The most common approach involves treating each sentence or document as a bag of words (BoW), where each word in a document is considered distinctly and equally. We begin with a *vocabulary*, which is a collection of all words to look for, and then we examine whether each word from the vocabulary occurs in the document. The simplest technique is the binary BoW, where we assign a value of one if the word appears in the document and zero otherwise, storing the result in a vector. The standard BoW (sometimes referred to as a count BoW) instead counts how many times each word appears and records these non-negative integers in the vector.

Introducing some notation, let W be a vector containing all the words in the vocabulary, which is of length (sometimes referred to as *cardinality*) $|w|$.²

One possibility is that the vocabulary is imposed exogenously – that is, it is created *a priori* and then applied to the document; alternatively, we can combine every distinct word in the corpus of all documents under study and this long list will comprise the vocabulary.

For instance, suppose that we had constructed a small vocabulary of 15 words to classify predictions about market movements from bulletin boards that contains the following words (for simplicity at this stage ignoring all the pre-processing steps outlined above): $W = ['high', 'low', 'buy', 'sell', 'stock', 'bond', 'profit', 'loss', 'gain', 'volatility', 'up', 'flat', 'down', 'rise', 'fall']$. Suppose further that we have two documents available:

$d_1 = ['I expect the stock market to rise, creating profit opportunities, offsetting yesterday's fall, and reaching a high of 1550 by lunchtime with a high of 1560 through the whole day']$

$d_2 = ['We anticipate that the bond market will experience volatility but will finish flat or down']$

We start off with the vocabulary W and create vectors of the same length for each document, where each element of the vector represents the number of times the word is mentioned in that document. In the document d_1 , 'high' occurs two times, whereas 'stock', 'profit', 'rise' and 'fall' occur once. So, the vector corresponding to d_1 is:

$$v_1 = [2 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1] \quad (9.1)$$

Similarly, the vector corresponding to d_2 is:

$$v_2 = [0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0] \quad (9.2)$$

It is evident that the vectors corresponding to each document are of the same length (i.e., they have the same feature space) irrespective of the number of words in the original document, which makes them amenable to quantitative analysis in the same way as we require there to be the same number of observations on each variable in a regression model. It is also clear from this illustration that the document word vectors will be *sparse* – that is, they contain a lot of zeros. This will make statistical modelling slow and inefficient.

The vector representations of the documents can be collected into a matrix known as the *document term matrix* of dimensions $|W| \times D$, where D is the total number of documents in the corpus.

² The cardinality of set of numbers is defined as the number of items in the set. For example, the cardinality of set $A = \{1,2,3,4\}$ is 4.

9.2.2 Vector Normalization

If individual words are used repeatedly in the same document, that can cause problems with the analysis akin to the issues with outliers in econometrics. For instance, consider a bank that is trying to classify its customer feedback as either positive or negative and comes across the review, 'Their service is just awful. Bad people, bad products, bad branch location, bad rates.' This respondent's repeated use of the word 'bad' could skew the analysis. To circumvent this issue, it is common to normalize the word vectors prior to analysis. Using what's known as the L_2 norm³ would involve creating vectors of length one via dividing each element by the square root of the sum of the squares of all the elements. In the case of v_1 above, the new vector would be:

$$\begin{aligned}
 v_{1,norm} &= \frac{v_1}{\|v_1\|_2} = \frac{[2 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1]}{\sqrt{2^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 1^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 1^2}} \\
 &= \frac{[2 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1]}{2.83} = [0.71 \ 0 \ 0 \ 0 \ 0.35 \ 0 \ 0.35 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0.35 \ 0.35]
 \end{aligned} \tag{9.3}$$

This process has scaled all the elements in v_1 so that they lie between 0 and 1 and no individual words would dominate even if they were used repeatedly in a document. A similar scaling would be used separately for each document so that they are all of unit length.

$$\sqrt{\sum x_i^2}$$

³ The L₂ norm of a vector of length n is given by $\|X\|_2 = \sqrt{\sum x_i^2}$, i = 1, ..., n where x_i are the elements of the vector.

9.3 Dictionary Comparison Approaches

Dictionary-based approaches are by far the most used ways to analyze text in finance research. The dictionary is a pre-defined list of words that share a common characteristic. Typically, the objective is to use NLP to assess the sentiment expressed in a document by providing separate counts of the number of positive words and the number of negative words it contains, each divided by the total number of words. Then the proportion of negative words is subtracted from the proportion of positive words to obtain a net sentiment score. Neutral words can be placed into a third category or simply ignored, along with stop words and any others having little relevant information content for measuring the document's tone.

Box 9.1 presents an illustrative example of a short newswire announcement, which is used to demonstrate the dictionary approach.

Box 9.1: Sample Newswire Report

"Firm XYZ has just reported a year-on-year rise in earnings before tax of just 0.1%, disappointing investors, despite total sales growth in double-digits. The company also highlighted that the previous safety worries with the new release had been resolved, which should underpin future growth. An analyst expressed relief, suggesting that there had been concerns that the accidents would have led to a decline in the firm's share of this competitive market."

A dictionary of sentiment words that have already been classified under the positive and negative headings could be employed. Informally applying this approach to the sample above, we would classify the words as positive and negative, respectively, as in [Table 9.1](#). Because there are a total of five positive words and only four negatives, we might conclude that the sentiment of this feed is slightly positive.

However, the example highlights some of the challenges of text mining because many of the negative words occur in counterfactual sentences explaining that things are better than feared (e.g., "would have led to a decline"). These sorts of issues indicate that the research design requires meticulous construction, particularly where the sentence structure is formal or complex.

Table 9.1 Positive and negative word stems for sample newswire text

| <i>Positive word stems</i> | <i>Negative word stems</i> |
|----------------------------|----------------------------|
| Rise | Disappoint |
| Grow (occurs twice) | Worry |
| Resolve | Concern |
| Relief | Decline |

Dictionary approaches usually rely on existing lists of words because constructing a new dictionary from scratch would be very time-consuming and would require examining many sample documents to be assured that most relevant words were included. Several dictionaries are available, such as Diction, which is a software package that classifies over 10,000 words under 35 separate categories.

More specifically in finance, Loughran and MacDonald (2011)⁴ developed comprehensive dictionaries of words classified as positive, negative, uncertain, litigious, strong modal, and weak modal, based on an extensive examination of 10-K filings. Research has suggested that the accuracy of these dictionaries is around 75% in terms of classifying Thomson Reuters news articles.

⁴ Loughran, T., & McDonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of Finance*, 66(1), 35-65.

9.3.1 Advantages and Disadvantages of the Dictionary Approach

The primary advantages of the dictionary approach are its transparency and its ease and speed of implementation. Dictionaries also facilitate comparisons across different corpuses if the same word lists are used in each case. However, it also entails several key limitations:

- The methods are only as good as the word lists on which they are built: if the lists are incomplete or include ambiguities or errors, classification accuracy could suffer.
- The dictionaries were designed for the analysis of formal documents, and they are likely to perform much less well when used for the analysis of bulletin board posts or Tweets using colloquial language such as, "this market is going to the moon" or "dog stock."
- Dictionaries are only available for a narrow range of subjects. If the researcher did not want to assess a document's tone but something else (for instance, determining whether a message post relates to the market as a whole or a particular stock, or conducting a broader topic analysis), the dictionaries are unlikely to exist *ex ante*. Creating new dictionaries for other applications is likely to be infeasibly time-consuming, in which case using a machine learning technique would be preferable.
- Negations (especially where they occur away from their object in a sentence) and ambiguous words (such as 'call' or 'put') can often confound dictionary applications.

9.3.2 n-grams

Despite its intuitive appeal and simplicity, a major limitation of the basic BoW approach is that it treats each word independently, irrespective of where they occur in the document, and simply counts the number of times they occur. However, there are sometimes pairs or groups of words with a specific meaning when placed together that need to be considered rather than individually (e.g., *red herring* or *San Diego*).

Moreover, negation words are frequently used in sentences to reverse the meaning of a word, for example, 'not good,' 'nothing useful,' neither up nor down.' Negation words can turn an otherwise positive sentence into a negative one, or, less cleanly, turn a negative sentence into a slightly positive one (e.g., 'not bad,' 'hardly boring'). Ignoring these negation words could severely compromise the accuracy of any analysis of a document.

A technique for dealing with these issues is to use '*n*-grams' where *n* (which is an integer equal to one or more) contiguous words in a document are treated as if they were a single word. A single word is sometimes known as a 'uni-gram,' an *n*-gram of two words is a bi-gram, and so on. When using bi-grams, pairs of words are analyzed together. For example, if the document is ['credit spreads narrowed today'], the bi-grams would be ['credit spreads,' 'spreads narrowed', 'narrowed today']. When the BoW approach is applied in the context of *n*-grams, it is known as the Bag of *n*-grams (BoN).

9.3.3 Term Frequency-Inverse Document Frequency

Documents will typically comprise a set of words that are commonly used plus others that are rarer. Distinguishing between the two can be a useful way to analyze the information more fully than basic word counts. For example, if our corpus comprised a set of sell-side analyst recommendations, words such as 'increasing,' and 'growth' are likely to feature frequently in many of the documents. But other terms such as 'phenomenal,' and 'spectacular' probably occur less often and therefore when they do, we should assign a higher priority to the sentiment they convey. Calculating the term frequency-inverse document frequency (*TF-IDF*) achieves this by assigning higher weights to words that occur frequently in a particular document but rarely in the corpus. On the other hand, if a

word occurs very commonly in many documents, whether it appears in document j is of little value in helping us to classify the document.

We define the term frequency, $TF_{i,j}$, as the ratio of the number of times a particular term i appears in document j , w_{ij} , to the total number of terms in document j , $|v_j|$, counting each occurrence separately. So, if a given word appears three times in a particular document, then three is added to the numerator for that word:

$$TF_{i,j} = \frac{w_{i,j}}{|v_j|} \quad (9.4)$$

where $i = 1, \dots, |W|$, and there are D documents in total, $j = 1, \dots, D$. We normalize the frequency of occurrence of a word in the document by the total number of words in the document, because a particular word will have more importance in a short document than a long one.

The inverse document frequency for term i , IDF_i , is the natural logarithm of the ratio of the total number of documents in the corpus, D , to the number of those documents containing term i .

$$IDF_i = \ln\left(\frac{D}{\sum_j d_{i,j}}\right) \quad (9.5)$$

where $d_{i,j}$ is a dummy variable taking the value 1 if document j contains word i and zero otherwise. We can then determine an appropriate weight to apply to each term i in document j , $w_{i,j}$:

$$(TF - IDF)_{i,j} = TF_{i,j} \times IDF_i \quad (9.6)$$

Although IDF for any word does not vary by document, TF varies from document to document because it is based on how many times a word appears in a particular document. A straightforward example of TF-IDF implementation is given in [Box 9.2](#).

Box 9.2: TF-IDF Example

Suppose that we are interested in identifying the most relevant words for characterizing a set of messages written by retail investors on a message board and we have the following four posts.

d_1 = "this disappointing stock let me down"

d_2 = "disappointing earnings, down for this stock"

d_3 = "earnings down for this month"

d_4 = "Fed hikes rates this month"

We first calculate the inverse document frequencies for each word as the log of the [number of documents in the corpus, 4, divided by the total number of documents containing that word]. Note that these do not vary by document, so we calculate the total number of times each word appears across the entire corpus. For instance, "disappointing" occurs in two documents out of four, so the IDF for this word is $\ln(4/2) = 0.69$, whereas for "down," which occurs in three documents, it is $\ln(4/3) = 0.29$.

| Word | Documents containing word | IDF | Word | Documents containing word | IDF | Word | Documents containing word | IDF |
|---------------|---------------------------|------|-------|---------------------------|------|-------|---------------------------|------|
| disappointing | 2 | 0.69 | for | 2 | 0.69 | month | 2 | 0.69 |
| down | 3 | 0.29 | hikes | 1 | 1.39 | rates | 1 | 1.39 |
| earnings | 2 | 0.69 | let | 1 | 1.39 | stock | 2 | 0.69 |
| fed | 1 | 1.39 | me | 1 | 1.39 | this | 4 | 0.00 |

The term frequencies are the number of times each word appears in each document divided by the number of words in that document, N .

| Document, j | | Word 1 | Word 2 | Word 3 | Word 4 | Word 5 | Word 6 | N |
|------------------|------------|-----------|---------------|-----------|-----------|-----------|-----------|-----|
| d_1 | word | this | disappointing | stock | let | me | down | 6 |
| | $TF_{i,1}$ | 0.17 | 0.17 | 0.17 | 0.17 | 0.17 | 0.17 | |

| Document, <i>j</i> | | Word 1 | Word 2 | Word 3 | Word 4 | Word 5 | Word 6 | <i>N</i> |
|-------------------------------|-----------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-----------------|
| | $(TFIDF)_{i,j}$ | 0.00 | 0.11 | 0.11 | 0.22 | 0.22 | 0.05 | |
| d_2 | word | disappointing | earnings | down | for | this | stock | 6 |
| | $TF_{i,2}$ | 0.17 | 0.17 | 0.17 | 0.17 | 0.17 | 0.17 | |
| | $(TFIDF)_{i,j}$ | 0.11 | 0.08 | 0.05 | 0.11 | 0.00 | 0.11 | |
| d_3 | word | earnings | down | for | this | month | | 5 |
| | $TF_{i,3}$ | 0.20 | 0.20 | 0.20 | 0.20 | 0.20 | | |
| | $(TFIDF)_{i,j}$ | 0.14 | 0.06 | 0.14 | 0.00 | 0.14 | | |
| d_4 | word | fed | hikes | rates | this | month | | 5 |
| | $TF_{i,4}$ | 0.20 | 0.20 | 0.20 | 0.20 | 0.20 | | |
| | $(TFIDF)_{i,j}$ | 0.28 | 0.28 | 0.28 | 0.00 | 0.14 | | |

Although no single word appears more than once in any of the four documents, the *TF-IDF* scores for each word show considerable variation because the total number of times each word is used varies considerably. This means that a particular word (e.g., “down”) will have a different TF for documents of different lengths. Also note that because “this” appears in all four documents, it has no discriminatory power across documents and therefore has a *TF-IDF* value of zero; likewise, “down” appears in three of four of the documents and so it has a low *TF-IDF* value, and it would not be remarkable if we found it in another document. On the other hand, words appearing just once in shorter documents such as ‘fed’ and ‘hikes’ have the highest *TF-IDF*.

It is worth being aware that there are several common variations on the *TF-IDF* formulae, and different versions are sometimes implemented in software libraries to avoid confusion. For instance, sometimes the frequencies in the TF formula are replaced with a binary dummy for whether the word is in the document or not (irrespective of

how many times it is used), or $w_{i,j}$ in the TF formula is replaced by $\ln(1 + w_{i,j})$. It is also possible to make a more sophisticated adjustment to TF to ensure that longer documents do not come to dominate the determination of the most important words in the corpus overall.

9.4 Machine Learning Approaches

The use of dictionaries to classify documents does not involve any learning. An alternative approach would be to use a sample of documents that have already been classified by a human. For instance, suppose that these were announcements as in box 9.2, labeled as positive, neutral, or negative. Then an algorithm would be used to learn from these documents the words most strongly associated with positive sentiment and the words most strongly associated with negative sentiment. The models developed in this way can then be used to classify other (unlabeled) documents that the machine has not seen.

Such an approach can be designed in a bespoke fashion, tailored to each specific application. But labeling a sufficiently large sample of announcements that the algorithm can learn from could be extremely time-consuming. Hence this collection of techniques is less widely implemented in finance than the dictionary approach described above. Such techniques are becoming more popular, now aided by recent advances in computing and advanced machine learning algorithms such as deep learning.

Once the documents are prepared and then the word frequency vectors constructed as above and labeled, the same machine learning classification techniques as described in Chapter 4 can be employed, including the naïve Bayes classifier, support vectors machines, logistic regressions, or neural networks. Naïve Bayes is sometimes known as a *generative classifier* because it can create new classifications for unclassified documents, whereas other approaches such as logistic regressions are *discriminative classifiers*. The latter can select the features (words in NLP applications) that provide the sharpest characterization between the categories, whereas naïve Bayes assumes that each feature (word) has an equal role in determining the outcome.

Although many of the supervised learning techniques for classification discussed in Chapter 4 were presented in the context of a binary choice, they can usually be extended to cover the situation where there are more than two possible classes (e.g., positive, negative, or neutral sentiment). In the case of logistic regression, this can be

extended to a multinomial logit model, and support vector machines can be adapted so that the data are separated into K clusters by $K-1$ hyperplanes. All the data organization and pre-processing steps would proceed as above and then the ML technique can be applied as discussed previously.

9.4.1 The Naïve Bayes Classifier

Suppose a retail bank is concerned that its customer service department is providing a poor experience, and it wants to investigate this via feedback forms that clients complete on its website. The bank has a small sample of eight such completed forms, and an employee has read each of them, classifying the feedback as either “good,” “bad,” or “indifferent.” The employee has also created a word bank (vocabulary) of four words that they believe will provide a useful characterization of the service level. For instance, the words might be “slow,” “inefficient,” “helpful,” and “great.” The employee has created a column vector for each customer’s feedback, based on whether each of the four words appears in their statement. This information is given in [Table 9.2](#) with the label in the final row.

Table 9.2: Data for customer service naïve Bayes example

| | Customer number | | | | | | | |
|--------|-----------------|-----|------|------|-------------|------|-------------|-----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Word 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| Word 2 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| Word 3 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| Word 4 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| Label | Good | Bad | Good | Good | Indifferent | Good | Indifferent | Bad |

Suppose also that we have an unlabeled statement, and we wish to use NLP to classify it. This new statement includes words 1 and 2 but not words 3 or 4, and we want to determine how this statement should be classified. The naïve Bayes classifier is useful for such classifications.

The naïve Bayes approach is based on a straightforward application of Bayes rule, which calculates the probability of an event conditional upon the value of another variable related to that event. It is termed “naïve” because it assumes the words in each document are independent of one another (or, more generally, that the features are independent of one another in a machine learning application). It is popular due to its simplicity and solid classification accuracy in practice.

We define C as a set of classes, $C = \{c_1, c_2, \dots, c_K\}$ and suppose that we have a corpus from which we have labeled (i.e., classified) by hand a subset N of the documents, $D = \{d_1, d_2, \dots, d_N\}$ as each belonging to one of the classes. We can write Bayes rule as:

$$P(c | d) = \frac{P(d | c)P(c)}{P(d)} \quad (9.7)$$

Here $P(c/d)$ is called the posterior probability that a document belongs to class c , $P(c)$ is known as the prior probability, or the unconditional probability of each class, and $P(d|c)$ is the conditional probability or likelihood of the document. $P(d)$ is the predictor prior probability, or the probability of the predictor variables, in this case the documents.

The classification problem is to choose the most likely of the K classes that a specific unlabeled document (i.e., one that had not already been classified) belongs to. We do this by calculating the probability of the document belonging to each class (the class probabilities) and then assigning the document to the class with the highest probability, which in technical terms is known as the “maximum *a posteriori* class.”

For convenience, the denominator of the equation 9.7, $P(d)$, can be eliminated because it does not change by class and hence it effectively cancels across classes. We can therefore write an equation for the selected class, \hat{c} :⁵

$$\hat{c} = \operatorname{argmax}_{c \in C} P(d | c) P(c) \quad (9.8)$$

We can express the document d equivalently using its words (w_1, \dots, w_N), and rewrite the preceding equation:

$$\hat{c} = \operatorname{argmax}_{c \in C} P(w_1, w_2, \dots, w_N | c) P(c) \quad (9.9)$$

The expression $(w_1, w_2, \dots, w_N | c)$ is the conditional joint probability that all these words would have appeared in that class. Given the independence assumption that underpins the naïve Bayes approach, we can write this joint probability as a product of the marginal probabilities and thus obtain:

$$\hat{c} = \operatorname{argmax}_{c \in C} P(w_1 | c) P(w_2 | c) \dots P(w_N | c) P(c) \quad (9.10)$$

$P(c)$ can be calculated easily for each class c_i as the number of documents labeled as belonging to that class divided by the total number of documents:

$$P(c_i) = \frac{n(c_i)}{N} \quad (9.11)$$

⁵ The argmax function gives the argument for determining the maximum value of a function.

9.4.2. Naïve Bayes Example

We now return to the example introduced earlier before introducing the Naïve Bayes classifier.

The features are the four words, while the responses or outputs are the three classes for the quality of the bank's customer service, which we can annotate as c_1 = "good"; c_2 = "bad"; and c_3 = "indifferent". The first stage is to calculate the unconditional probabilities of each class; there are a total of eight customers, with four of them rating the service as good; two as bad; and two as indifferent. Therefore, the unconditional probabilities are:

$$P(c_1) = \frac{4}{8} = 0.5; P(c_2) = \frac{2}{8} = 0.25; P(c_3) = \frac{2}{8} = 0.2525$$

Next, we want to calculate the conditional probabilities of "words given class." It will make it more straightforward to calculate these if we construct another table (Table 9.3) that puts each word in a row and the classes in columns (rather than each individual customer as above).

Table 9.3 Count of the number of customers using each word given their classification.

| | Class | | |
|-----------------|----------|---------|-----------------|
| Word | 1 (Good) | 2 (Bad) | 3 (Indifferent) |
| 1 ($w_1 = 1$) | 0/4 | 2/2 | 1/2 |
| 2 ($w_2 = 1$) | 1/4 | 2/2 | 2/2 |
| 3 ($w_3 = 0$) | 2/4 | 1/2 | 1/2 |
| 4 ($w_4 = 0$) | 0/4 | 1/2 | 1/2 |

Focusing first on the reviews classed as good, we need to find out how many statements from class c_1 have used word 1, which is zero (nobody who was classified as having rated the service as good, included word 1 in their statement). Next, how many people from class 1 used word 2? The answer is one person (out of four whose statements were classed as good). How many people from class 1 did not use word 3? Two people out of four. How many people from class 1 did not use word 4? None (everyone from class 1 used word 4). We can express this information using the probability notation for the four words as follows:

$$P(w_1 = 1 \mid c_1) = \frac{0}{4} = 0$$

$$P(w_2 = 1 \mid c_1) = \frac{1}{4} = 0.25$$

$$P(w_3 = 0 \mid c_1) = \frac{2}{4} = 0.5$$

$$P(w_4 = 0 \mid c_1) = \frac{0}{4} = 0$$

Note that we are interested in $P(w_3 = 0 \mid c_1)$ rather than $P(w_3 = 1 \mid c_1)$ because the scenario given in the question states that word 3 is *not* present in the document to be classified.

Now we can calculate $P(w_1 \mid c_1)P(w_2 \mid c_1)\dots P(w_n \mid c_1)P(c_1) = 0 \times 0.25 \times 0.5 \times 0 \times 0.5 = 0$.

We then repeat the process for the two respondents in class 2 (bad review). How many people from class 2 used word 1? Both; how many people from class 2 used word 2? Both. How many people from class 2 did not use word 3? One of them. How many people from class 2 did not use word 4? One of them. We can therefore write out the conditional probabilities for class 2 as we did for class 1 above:

$$P(w_1 = 1 \mid c_2) = \frac{2}{2} = 1$$

$$P(w_2 = 1 \mid c_2) = \frac{2}{2} = 1$$

$$P(w_3 = 0 \mid c_2) = \frac{1}{2} = 0.5$$

$$P(w_4 = 0 \mid c_2) = \frac{1}{2} = 0.5$$

$$P(w_1 \mid c_2)P(w_2 \mid c_2) \dots P(w_n \mid c_2)P(c_2) = 1 \times 1 \times 0.5 \times 0.5 \times 0.25 = 0.0625$$

Repeating the same steps for class 3, the conditional probabilities would be:

$$P(w_1 = 1 \mid c_3) = \frac{1}{2} = 0.5$$

$$P(w_2 = 1 \mid c_3) = \frac{2}{2} = 1$$

$$P(w_3 = 0 \mid c_3) = \frac{1}{2} = 0.5$$

$$P(w_4 = 0 \mid c_3) = \frac{1}{2} = 0.5$$

$$P(w_1 \mid c_3)P(w_2 \mid c_3)\dots P(w_n \mid c_3)P(c_3) = 0.5 \times 1 \times 0.5 \times 0.5 \times 0.25 = 0.03125$$

Note that the three class likelihoods (conditional probabilities) will not sum to one because we have removed the denominator in the Bayes rule formula of equation (9.7) above. We can turn these numbers into probabilities by normalizing them (i.e., dividing by their sum):

$$P(c_1 \mid w_1 = 1, w_2 = 1, w_3 = 0, w_4 = 0) = \frac{0}{(0 + 0.0625 + 0.03125)} = 0$$

$$P(c_2 \mid w_1 = 1, w_2 = 1, w_3 = 0, w_4 = 0) = \frac{0.0625}{(0 + 0.0625 + 0.03125)} = 0.67$$

$$P(c_3 \mid w_1 = 1, w_2 = 1, w_3 = 0, w_4 = 0) = \frac{0.03125}{(0 + 0.0625 + 0.03125)} = 0.33$$

Therefore, we would classify this unlabeled statement as most likely belonging to group 2 ("bad") because it has the highest posterior probability.

We should note two potential issues with the above approach:

1. The probabilities of a specific word appearing in a particular document could be small, and for long and diverse documents we could end up multiplying together several extremely small numbers. This could cause an 'underflow' problem for the computer, so instead it is common to take the natural logarithm of the probabilities. Because $\ln(AB) = \ln(A) + \ln(B)$, the transformation will mean that rather than multiplying raw probabilities we are summing the logs of probabilities, which will be computationally more manageable. Taking logs will not change the result because the class with the highest probability will be the same as the class with the highest log probability.
2. A further, and often fatal, issue arises when a specific word does not appear in the labeled sample for a particular category, because then the probability of that category having the word will be calculated at exactly zero. This will render the probability of the category to be zero for that document irrespective of

whether it contains many other words that are usually associated with that category. For instance, in the customer service example above, the probability that the unlabeled review would be classified as ‘good’ is zero, which seems rather extreme. Given the data here, we would believe that the probability that this customer has a good view of customer service is low but not zero. More generally, even with a larger training sample and a substantial vocabulary, if one client uses a positive word in an otherwise negative review (e.g., being sarcastic), and if there were no reviews labeled negative in the training set that contained that word, the probability of that review being negative would be set to zero. The probability of the review being negative would still be zero even if it contained many other words that were usually associated with negative reviews.

To avoid the second of these situations, an adjustment, known as (additive) *smoothing*, is sometimes made to the dataset. This adds a positive integer, λ , to all the counts in the conditional probability matrix, which will ensure that they are all non-zero and therefore all the probabilities will be non-zero. *Laplace smoothing* is most used, which sets $\lambda = 1$. We can see the effect of this by reconsidering the customer service example above and adding one to the count of each outcome from Table 9.2 to create [Table 9.4](#).

Table 9.4: Customer data with one dummy observation added for each outcome for each class

| | Customer number | | | | | | | | | | | | | |
|--------|-----------------|-----|------|------|-------------|------|-------------|-----|------|------|-----|-----|-------------|-------------|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Word 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| Word 2 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| Word 3 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| Word 4 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| | Good | Bad | Good | Good | Indifferent | Good | Indifferent | Bad | Good | Good | Bad | Bad | Indifferent | Indifferent |

Effectively, we have added an additional (dummy or “pseudo”) respondent for each outcome and each class. We can then recalculate the probabilities for each class given the words, as we did in [Table 9.3](#) above, and this revised matrix is given in [Table 9.5](#).

Table 9.5 Modified count of the number of customers using each word given their classification after Laplace smoothing

| Word | Class | | |
|--------|----------|---------|-----------------|
| | 1 (Good) | 2 (Bad) | 3 (Indifferent) |
| 1 (=1) | 1/6 | 3/4 | 2/4 |
| 2 (=1) | 2/6 | 3/4 | 3/4 |
| 3 (=0) | 3/6 | 2/4 | 2/4 |
| 4 (=0) | 1/6 | 2/4 | 2/4 |

We also need to recalculate the unconditional probabilities given the enlarged sample including the dummy observations:

$$P(c_1) = \frac{6}{14}$$

$$P(c_2) = \frac{4}{14}$$

$$P(c_3) = \frac{4}{14}$$

Then we can calculate:

$$P(w_1 \mid c_1)P(w_2 \mid c_1)\dots P(w_n \mid c_1)P(c_1) = \frac{1}{6} \times \frac{2}{6} \times \frac{3}{6} \times \frac{1}{6} \times \frac{6}{14} = 0.0020$$

$$P(w_1 \mid c_2)P(w_2 \mid c_2)\dots P(w_n \mid c_2)P(c_2) = \frac{3}{4} \times \frac{3}{4} \times \frac{2}{4} \times \frac{2}{4} \times \frac{4}{14} = 0.0402$$

$$P(w_1 \mid c_3)P(w_2 \mid c_3)\dots P(w_n \mid c_3)P(c_3) = \frac{2}{4} \times \frac{3}{4} \times \frac{2}{4} \times \frac{2}{4} \times \frac{4}{14} = 0.0286$$

As above, these numbers can be transformed into probabilities by normalizing them (i.e., dividing by their sum):

$$P(c_1 \mid w_1, w_2, w_3, w_4) = \frac{0.0020}{(0.0020 + 0.0402 + 0.0286)} = 0.029;$$

$$P(c_2 \mid w_1, w_2, w_3, w_4) = \frac{0.0402}{(0.0020 + 0.0402 + 0.0286)} = 0.583;$$

$$P(c_3 \mid w_1, w_2, w_3, w_4) = \frac{0.0286}{(0.0020 + 0.0402 + 0.0286)} = 0.388$$

We can see that the probability that the unlabeled review is good has increased from zero to 2.9%, with the probability of the review being bad falling while the probability of it being indifferent has gone up slightly.

Smoothing is a shrinkage technique that works in a similar way to the Lasso and ridge regression techniques discussed in Chapter 7, and the result is that the empirically estimated probabilities will be pushed towards a uniform distribution and the larger the value of λ , the stronger would be the extent of smoothing. On the other hand, as the sample size increases (more customers surveyed and labeled), the impact of smoothing is reduced. Smoothing will solve the problem of zero probabilities, but adding one to each count is arbitrary.

Another issue is that there will likely be words appearing in the unlabeled (test) dataset that were not in the labeled (training) set, and therefore about which the model will have no information. Such words would be removed from the test document because they cannot help the algorithm to select an optimal classification for that document.

9.4.3 Word Meanings

The BoW approach, even when used with n -grams, simply counts the occurrences of words or groups of words, not imputing any meaning to them. This simplifies the analysis but also represents a severe limitation.

A technique known as *word embedding* attempts to capture a word's meaning by presuming that if two words have similar meanings, they will appear in similar contexts across documents. To represent this numerically, we need to identify not only which words from the vocabulary appear in each document, but also to pay attention to their positions. A solution is to set up a separate vector for each word in the vocabulary and then to use one-hot encoding (0-1 binary variables) for whether the word occurs in a particular position or not. The resulting matrix will be vast in dimension and extremely sparse.

Word2Vec is an algorithm developed by Google for doing such word embeddings, and it reduces the dimensionality using neural networks. For each word in the vocabulary, Word2Vec produces an ordered list with scores of similar words. This algorithm will be discussed further in Chapter 10.

9.5 NLP Evaluation

The most appropriate way to evaluate an NLP model will depend on the nature of the problem that was specified in the first place. As for other machine learning applications, the dataset can be separated into training, validation and testing sub-samples, with evaluation conducted on the latter. If the documents are labeled, this provides a "right answer" and in such cases the machine classifications can be compared with the labels using a confusion matrix and standard measures such as accuracy, precision, and recall as discussed in detail in Chapter 8.

Simple NLP methods like BoW are not useful for determining the context in which a word is used to understand the intended meaning. Advanced techniques like n-grams and Skip-Grams (see Chapter 10) need to be employed for determining the context in which a word is used.

However, in many relevant scenarios the documents will not have a label. For instance, if we were interested in determining how customers were reacting to a new financial product based on the social media posts they were

making, there would be no labels and it would probably be infeasible to go through each message manually to classify it as “favorable” or “unfavorable.” In such cases, the models can be evaluated in the same way as unsupervised approaches, again discussed in Chapter 8.

When evaluating NLP applications, it is also essential to consider the speed of the algorithm and how much data is required to train it. As we discussed, textual databases can be vast, and sparse word vectors are hard to handle efficiently, which together may make training more sophisticated models such as neural networks very slow.

Appendix 9.A Naïve Bayes Application Problem

You work as a data scientist at a sell-side analyst firm. The firm gives away for free the narratives about the companies it provides buy/hold/sell recommendations for, making money by selling the recommendations themselves. The CFO is concerned that it would be possible for third parties to predict the recommendations from an NLP analysis of the narratives. You investigate this by examining a random sample of ten known recommendations and a set of five keywords from their accompanying narratives, observing the information in the following table, where one denotes that the word is present in that narrative and zero denotes where it is not. To simplify the analysis, you ignore the hold recommendations and focus on buy and sell.

| Label | Buy | Buy | Buy | Buy | Buy | Sell | Sell | Sell | Sell | Sell |
|--------|-----|-----|-----|-----|-----|------|------|------|------|------|
| Word 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| Word 2 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| Word 3 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| Word 4 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| Word 5 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |

- A. Suppose that you decide to make predictions for another set of recommendations based on an analysis of the above table, which can then be compared with the actual recommendations. The first of these new instances includes each of the first three words (i.e., words 1 to 3) in its narrative but not words 4 or 5. By making use of the naïve Bayes classifier, identify whether the most likely recommendation is a buy or a sell.
- B. If an examination of additional documents from the sample reveals other words not incorporated in your analysis, how would you treat them?
- C. Explain why “smoothing” is often required in the context of naïve Bayes applications and explain how the technique works.

A. We first calculate the unconditional (i.e., prior) probabilities, which is simply a count of the number of each outcome divided by the total number of outcomes. Because there are five buys and five sells out of ten recommendations, the unconditional probabilities are both 0.5. Defining a buy as c_1 and a sell as c_2 , we can write:

$$P(c_1) = \frac{5}{10} = 0.5; P(c_2) = \frac{5}{10} = 0.5$$

Next, we calculate the conditional probabilities of the form $P(\text{words} | \text{classification})$ as in the following matrix:

| Word, i | $P(c_1 w_i)$ | $P(c_2 w_i)$ |
|-----------------------------|----------------------------------|----------------------------------|
| 1 | 4/5 | 1/5 |
| 2 | 5/5 | 2/5 |
| 3 | 2/5 | 3/5 |
| 4 | 1/5 | 3/5 |
| 5 | 2/5 | 4/5 |

The question states that the new instance contains words 1-3 but not words 4 or 5, so we can calculate the conditional probabilities that the recommendation would be a buy or a sell respectively as:

$$P(w_1 = 1, w_2 = 1, w_3 = 1, w_4 = 0, w_5 = 0 \mid c_1) P(c_1) = \frac{4}{5} \times \frac{5}{5} \times \frac{2}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{1}{2} = 0.0768$$

$$P(w_1 = 1, w_2 = 1, w_3 = 1, w_4 = 0, w_5 = 0 \mid c_2) P(c_2) = \frac{1}{5} \times \frac{2}{5} \times \frac{3}{5} \times \frac{2}{5} \times \frac{1}{5} \times \frac{1}{2} = 0.00192$$

We could then normalize these probabilities so that they sum to one.

$$P(c_1) = \frac{0.0768}{(0.0768 + 0.00192)} = 0.98$$

$$P(c_2) = \frac{0.00192}{(0.0768 + 0.00192)} = 0.02$$

Evidently, it is much more likely, given the words the narrative contains, that the recommendation for this new instance is a buy rather than a sell.

B. If there are words in the unlabeled documents that were not in the labeled sample, they cannot be used in the analysis, because we cannot obtain any information from them that will help us to select the most appropriate classifications for unlabeled documents. Unless we were able to extend the labeled sample to include documents containing those words, we would simply have to ignore them.

C. An issue with the Naïve Bayes method for classification is known as the “zero probability problem”. This arises when a specific word does not appear in a document for a particular classification in the labeled sample, which would render the probability of that classification as zero for the new unlabeled data point. For example, suppose that word 4 had not appeared in any of the narratives labeled as buy recommendations. Then if a new instance

included word 4, then the conditional probability that it is a buy recommendation would be calculated as zero. Even if it is unlikely that this narrative corresponds to a buy recommendation, it would be rather draconian to set the probability to exactly zero. Laplace is one of a family of smoothing algorithms that tackles this issue by adding a positive integer to all the counts in the conditional probability matrix, which has the effect of increasing otherwise zero conditional probabilities.

Natural Language Processing: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

9.1 Suppose that an analyst wants to determine the sentiment embodied in the prospectuses of fifty firms that are undertaking an initial public offering in terms of how bullish they are about the company's prospects. Briefly explain the basic steps involved in doing that via a heuristic approach.

A sentiment analysis of each of a collection of reports could be conducted in the following steps, assuming that the document texts are available in electronic form:

1. Establish a dictionary—this would comprise three lists of words: positive, negative, and neutral.
2. Pre-process the text—removing punctuation, symbols, and stop words; modify all words to lower case.
3. Replace words with their stems and lemmas.
4. Calculate the proportion of positive words, the proportion of negative words, and the proportion of neutral words.

9.2 Explain how a bag of words model is used in natural language processing.

The bag of words model treats each term in a document identically, irrespective of the position of the words and we make a count of how frequently each word is used, with these numbers placed into a vector for each document. The vector can then be analyzed, either by comparing it with pre-defined lists of words or using a machine-learning approach. The word lists are known as dictionaries – for example, to capture whether a particular document embodies a positive or negative sentiment. The machine-learning approach would use a labeled sub-set of the documents and treat the words as features with the aim of classifying the remaining unlabeled documents accordingly.

9.3 State and explain two assumptions that underpin the use of the naïve Bayes classifier technique for document grouping.

The naïve Bayes classifier technique for document grouping requires two crucial assumptions:

- *Each word in a document is independent of all other words in the document. This independence assumption is required so that the joint probability can be expressed as a product of all the marginal probabilities, e.g., for two events A and B, $P(A \text{ and } B) = P(A) \times P(B)$.*
- *Each word is given equal weighting in constructing the outcome. We could of course remove any stop words prior to analysis, but once that is done, each word is assumed to have equal information value.*

9.4 Explain why negation words can cause problems when classifying documents and suggest an approach to dealing with them.

Negation words such as “not”, “isn’t”, have a dramatic effect on a sentence, often completely reversing its meaning. For instance, compare “the bank’s governance processes are adequate” with “the bank’s governance processes are not adequate.” If we use the bag of words approach, any negation words will be decoupled from the other words in the document so that the meaning of the sentence will probably be incorrectly interpreted as the reverse of what its author intended. An approach to dealing with this issue is to use “n-grams” ($n > 1$), where words are considered as blocks rather than individually. For example, within the bag of words technique we could analyze bigrams, which are pairs of words, to ensure that a pair included both the negation word and the word it proceeds, such as “not sufficient,” “not good,” etc.

9.5

A. Why do some words have zero TF-IDF values?

A word will have a zero score if it appears in every document. In such cases, the word will have no power to discriminate between documents and so it has no value and would be assigned no weight.

B. Why does the same word have a higher TF-IDF score in one document than another?

The inverse document frequency (IDF) does not vary by document, but when this is multiplied by the term frequency within the document, this can lead the same word to have different TF-IDF values across documents because of the differing lengths of the documents, which lead to different term frequency values. In the very short document examples here, each word was only used at most once in each document, but in any practical scenario, some words are likely to be used several times in particular documents, which would lead the TF-IDF for a given word to change further between one document and another.

Chapter 10: Generative Artificial Intelligence

Learning Objectives

The well-known ChatGPT is based on a generative AI technology known as a transformer, a specific type of a large language model (LLM). The rapid adoption of ChatGPT has created confusion regarding the distinctions between GenAI and LLMs. This chapter provides an understanding of the relationship between GenAI, LLMs, and the technologies and algorithms that underlie them.

After completing this chapter, you should be able to:

- Describe and distinguish between different generative artificial intelligence technologies.
- Describe the role of LLMs in GenAI.
- Explain how embeddings are used to represent word vectors.
- Differentiate between the two Word2Vec architectures.
- Differentiate between recurrent neural networks (RNNs) and transformers for capturing the relationships between words in a sentence.
- Describe the basic structure of LLMs at a conceptual level.
- Discuss prompt engineering and temperature in the context of LLMs.
- Describe applications of GenAI and LLMs.

10.1 Introduction

Technological innovation in generative artificial intelligence (GenAI) has spurred a great deal of interest over the past several years, accentuated by OpenAI's release of ChatGPT in November 2022. According to data collected by Statista, by the end of 2022, ChatGPT had made history by becoming the fastest technology to reach one million users in five days.¹ In comparison, it took Facebook almost a year to reach this milestone (although arguably, far fewer people had access to the internet or were comfortable trying new web-based technologies back in 2004).

ChatGPT² is a type of chatbot³ based on a GenAI technology known as a *transformer*, which is a specific type of a *large language model* (LLM). Due to the rapid adoption of this technology, there has been some confusion regarding the distinctions between GenAI and LLMs. Therefore, this chapter begins with an outline of the relationship between the types of GenAI technology and the algorithms that underlie them. We then revisit the word embeddings that were discussed in Chapter 9 and extend the framework to a more flexible representation in the *Word2Vec* model. The two forms of Word2Vec architecture – namely, the *continuous bag of words* (CBOW) and *skip-gram*, are presented. We then discuss the *Recurrent Neural Network* (RNN) model. This is followed by a discussion of the more advanced technologies of transformers and LLMs which, through their use of the concept of “attention,” can develop semantic as well as syntactic knowledge and represented a significant step forward in NLP. We conclude the chapter with an overview of the applications of GenAI and LLMs.

¹ <https://www.statista.com/statistics/1360613/adoption-rate-of-major-iot-tech/>

² GPT stands for Generative Pre-trained Transformer.

³ A chatbot is a software that mimics human conversation either via text or voice.

10.2 A Simple Taxonomy for Generative AI

To better understand the different types of GenAI, we present a simplified taxonomy in a grid format below (see [Table 10.1](#)). This describes GenAI by modality of content generated (e.g., text, video, etc.) and by the type of model used (e.g., transformer, recurrent neural network, etc.). Shaded cells indicate that a particular type of model could, or is, used to generate a certain type (or mode) of content.

The five major content modalities and their associated underlying models are:

- ***Text***: These are models that will take a prompt and generate textual output. The underlying technology is usually a type of *transformer* which is discussed in section 10.4.
- ***Image***: These are models that will take a prompt and create an image as the output. A popular image generator, developed by OpenAI, is DALL-E. DALL-E can take a text prompt and then create an image. For example, the image in [Figure 10.1](#) was created using DALL-E.⁴



Figure 10.1: DALL-E generated image showing the financial markets, and its participants, as a giant roller coaster

- **Audio:** These are models that will generate audio including speech, sound effects, and even music.
- **Video:** These are models that will create videos based on a prompt or can be used to modify existing videos.
- **Multimodal:** The models take textual prompts as their input and then generate text, images, audio, or video, respectively. However, significant effort and resources are being deployed to develop *multimodal* GenAI solutions. These technologies use multiple modalities and formats of data as input and then can generate more robust (and realistic) outputs in a multimodal format.

The next layer in the taxonomy of Gen AI is the modeling algorithms that support the different output modalities.

Table 10.1 Simple Taxonomy for Generative AI

| | Content Output Modality | | | | |
|--------------------------------------|-------------------------|--------------------------------|---|--|--|
| Underlying Model | Text | Image | Audio | Video | Multi-modal |
| Large Language Models (LLMs) | ■ | | | | ■ |
| Transformers | ■ | ■ | | ■ | ■ |
| Stable Diffusion | | ■ | | | |
| Variational Auto-Encoder (VAE) | | ■ | | | |
| Generative Adversarial Network (GAN) | | ■ | | ■ | |
| Recurrent Neural Network (RNN) | | | ■ | ■ | |
| Examples | GPT, BERT | DALL-E, StyleGAN, VQ-VAE | Jukebox, WaveNet, Tacotron TTS | Deepfakes, Synthetic Media Tools | Gemini Imagebind GPT4-V Med-Palm2 Kosmos-1 Blip-2 |

- **Large Language Models (LLMs):** LLMs are models developed for natural language processing (NLP) and are built using vast neural network models with billions of parameters. Instead of utilizing human-created dictionaries, however, LLMs develop rules entirely from the enormous corpus of text that they are trained on and are then able to apply this “universal knowledge” across a wide range of tasks. LLMs are built mainly on the transformer architecture and have given rise to the growth of easy-to-use text creation GenAI platforms.
- **Transformers:** Transformers are a type of Deep Learning models⁵ used for NLP. Popular examples of transformers are BERT – bidirectional encoder representations from transformers – and GPT – generative pretrained transformers.
- **Stable Diffusion:** Another powerful image generating technology is the Stable Diffusion model.⁶ It is trained on a database of images and then uses probabilistic functions to reconstruct an image based on a textual prompt.
- **Variational Autoencoders (VAE):** Another technology for image (and video) modalities is variational autoencoders (VAE). VAEs are a type of neural network with multiple hidden layers – and therefore more specifically would be classified as a type of Deep Learning (DL) algorithm or Deep Neural Network (DNN). A VAE is trained on a set of images that is compressed (this is the “encoding” layer) and then reconstructed in such a way where the loss of information in the first layer is replaced using a probabilistic function to create a new image (the “decoding” layer). A VAE is particularly useful for *modifying* existing images due to its ability to fill in missing information by probabilistic inference.
- **Generative Adversarial Networks (GANs):** GANs are used to create images, audio, and video. GANs employ two neural networks that both compete against and work with one another. The network is the generator which creates simulated data. This simulated data is combined with real data – which may be from a curated database (a training set of sorts) or just pulled from the internet – and then fed into another neural network called the discriminator. The discriminator’s job is to figure out which data items are real, and which are fake. It then assigns a probability score that is sent back to the generator, which then uses this updated information to create better generated data (i.e., images, etc.). This process continues iteratively until the generated data is virtually indistinguishable from the real data.
- **Recurrent Neural Network (RNN):** Lastly, in the technology level of the Gen AI taxonomy is the recurrent neural network (RNN). In a feed-forward neural network, which we saw in Chapter 4, the input features

flow from the input to the output layers generating outputs. However, in a basic RNN, the inputs from the current time step as well as its output from the prior time step are fed back to it. In finance, these types of models are being embraced for their ability to represent time series data. RNNs can be used to create speech or other audio outputs.

⁴ The prompt used was “create an image that personifies the ‘ups and downs’ of financial markets as if it were investors sitting on a roller coaster.”

⁵ Deep learning refers to neural network-based machine learning models with multiple hidden layers to extract the nonlinear relationships embedded in the data being modeled. Neural network models were discussed in Chapter 4.

⁶ The stable diffusion model was developed by Stability AI and released in 2022.

10.3 Word Embedding, Word2Vec, and RNNs

We saw in Chapter 9 how vectors can be created to represent the words in a document. As a reminder, the *binary bag of words* sets up a separate vector, V_i , of length $|W|$ of the entire vocabulary for each document i (where $i = 1, \dots, N$). The length $|W|$ could be of the order of 100,000 words for a real application. Each element in the vector will take the value 1 if the word appears in that document and 0 otherwise. A slight variant on this is the *count bag of words*, or *frequency bag of words*, in which the vector includes counts of the number of times that each word appears in the document. These bag of words approaches to analyzing text have three major limitations:

1. The techniques have nothing to say about the interpretation of a particular word, and therefore an analyst using the bag of words will not be able to identify when two words have a similar meaning, such as “amazing” and “fantastic.”
2. This process creates vast vectors that are extremely sparse, containing mostly zeros. This is extremely inefficient, and makes the information very difficult to store, let alone analyze.
3. Each word is treated independently of all other words in a document so that the *context* in which they occur is lost. This causes issues when a specific word has more than one meaning (e.g., a *call* option and a phone *call*).

Word2Vec addresses the first two of these issues through a technique known as *word embedding*. RNNs are an alternative to Word2Vec, and they are useful for identifying the dependencies between words. Word2Vec is discussed next followed by RNNs.

10.3.1 Word2Vec

Word2Vec is a model for NLP that was created in 2013 by a Google team led by Tomas Mikolov⁷ as an alternative to the conventional BoW method for the vector representation of words. With Word2Vec, each word in the vocabulary has its own vector that is constructed by examining all the surrounding words in each document where that word occurs. The model can be used to determine the other words that are most like a particular word by determining the degree of “co-occurrence” of a group of words, i.e., how likely they are to appear together in a sentence.

Compared to the standard BoW approach discussed in Chapter 9, Word2Vec permits a compressed representation by using an *embedding*, which employs a neural network to effectively reduce the dimensionality by creating a dense vector representation of words for storage and analysis.

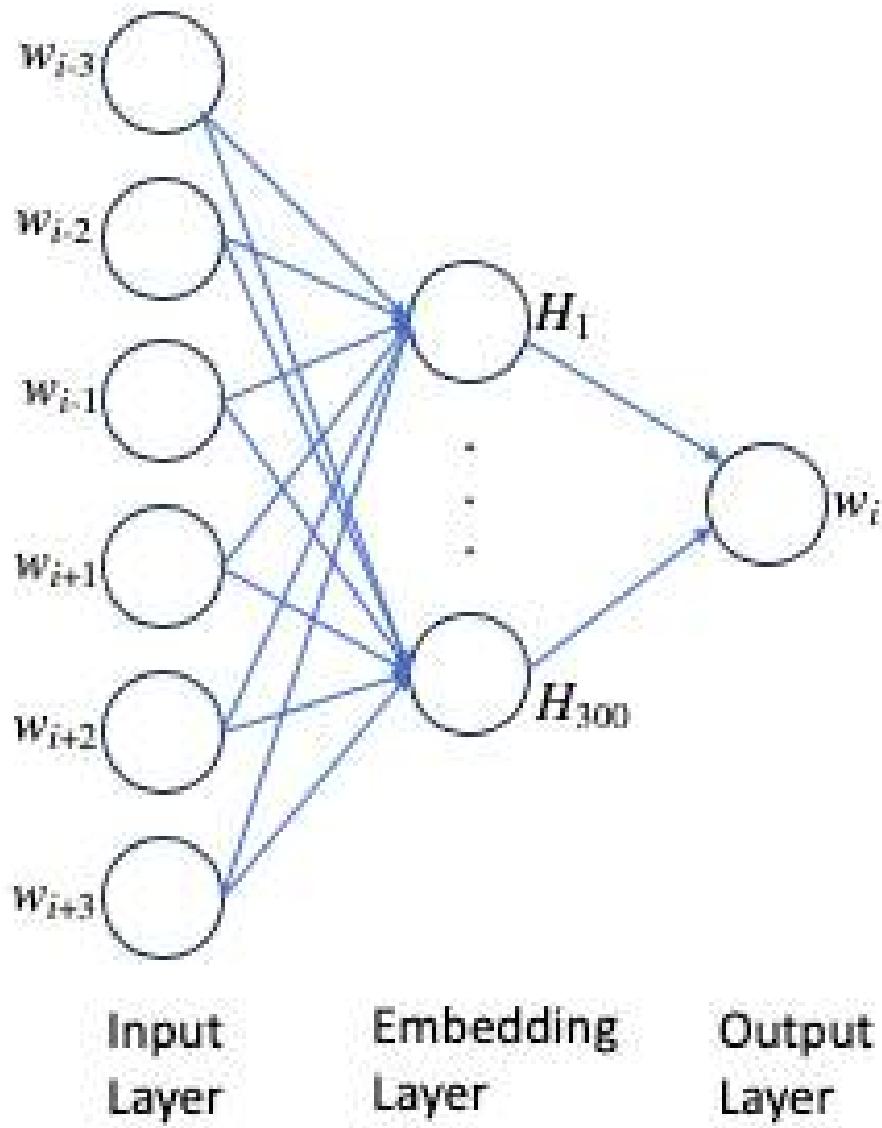
There are two architectures through which Word2Vec can be implemented: the *continuous bag of words* (CBoW) and the *continuous skip-gram*, both of which roll iteratively through the words in all documents in the corpus:

1. The CBoW is a “fill in the blank” technique that proposes a probability-ordered list of words that would fill in the gap between a few words before and after a missing word. If we use a window of length c on each side of a target word, w_i , we will use the context words $w_{i-c}, w_{i-c+1}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+c-1}, w_{i+c}$. We thus have $2c$ context words (c before and c after the target word), and the goal is to select the target word with maximum probability, i.e., the most likely target words given the surrounding words, which we could write as $P(w_i | w_{i-c}, \dots, w_{i+c})$.
2. The skip-gram, on the other hand, is like a “word association” technique that uses a particular word to predict the words that surround it within a certain number of places before and after that word. Here, we would try to predict the context words $w_{i-c}, w_{i-c+1}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+c-1}, w_{i+c}$, based on knowledge of only the word w_i . Now the goal is to maximize the joint probability of all the context words given one input word: $P(w_{i-c}, \dots, w_{i+c} | w_i)$.

Word2Vec can capture semantic similarity and uses a neural network that is somewhat like an autoencoder (see Chapter 2) with a single hidden layer, known as an embedding layer, as represented as in [Figure 10.2](#). This is illustrated assuming a context of three words before ($w_{i-1}, w_{i-2}, w_{i-3}$) and after ($w_{i+1}, w_{i+2}, w_{i+3}$) the target word. For CBOW, there is one output (target word) and six inputs, whereas for skip-gram there is one input and six outputs.

Typically, the embedding layer includes 300 neurons. This implies that we can measure how close every word is to every other word using a weight space, of dimension $|W| \times 300$, from the input layer to the hidden layer, rather than $|W| \times |W|$ for the conventional BoW representation, which implies a considerable reduction in dimensionality. The input layer comprises a set of one hot encoded vectors that take the value 1 in the position reflecting the index of the word and 0 everywhere else, while the output layer contains the probabilities associated with the predictions for the target word (for CBoW) or for the context words (for skip-gram). Estimation of the weights usually takes place via gradient descent method like the one employed for training feedforward neural networks.

CBOW



Skip-gram

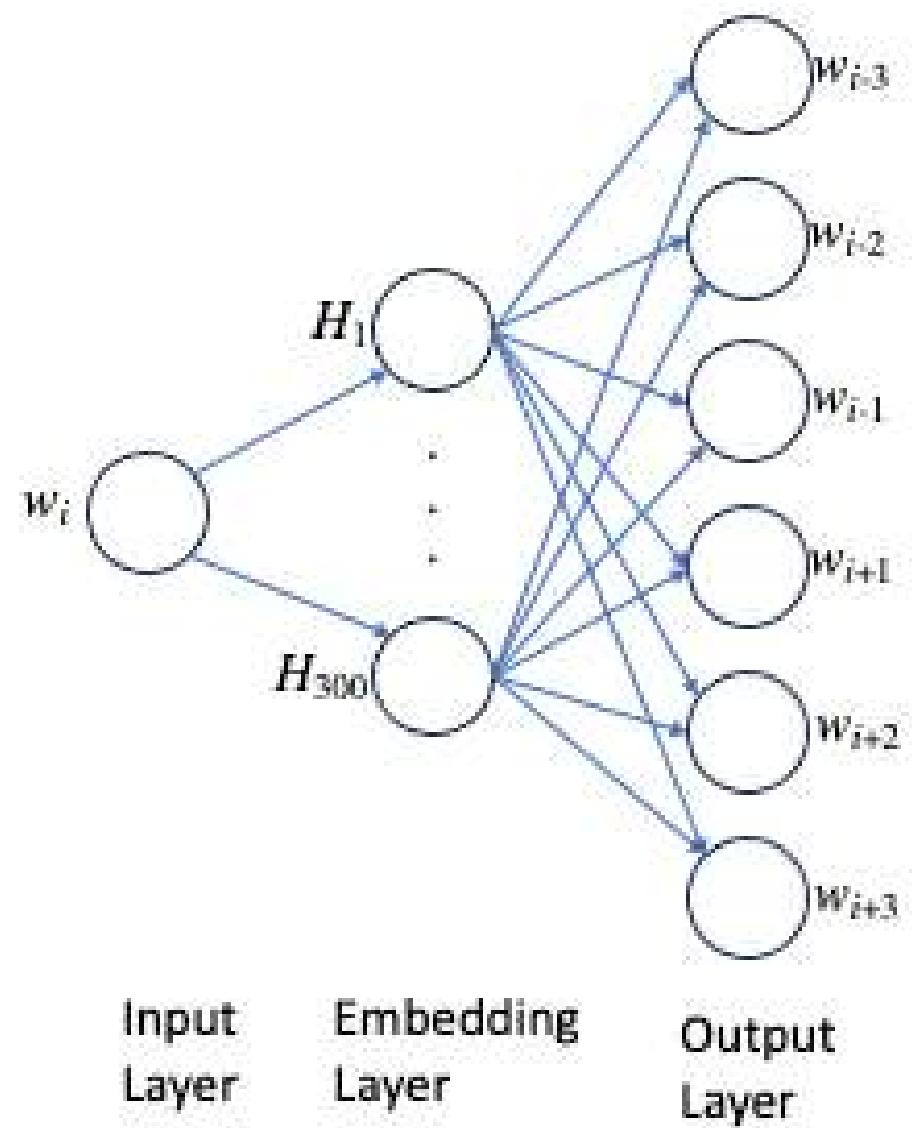


Figure 10.2: The two architectures available for the Word2Vec vector representations

To find word similarities, a measure such as the cosine similarity⁸ of two words in a word vector space is typically used. Cosine similarity measures the “distance” between words where similar words will be close together and dissimilar words will be far apart in the vector space. The model learns the relationships between words by looking for co-occurrences across sentences. For example, if the word is “risk,” it is likely that the words “loss” and “management” would be close to it, but “university” would probably be much further away. In addition to analyzing how close the words are to each other, the embeddings can also reveal other relationships between words. One can perform quasi-mathematical operations with the word vectors, which capture word associations in different dimensions of word vector space. For example, the embeddings can be used to identify analogous words. It is possible to perform following operation:

$$\text{vector("X")} = \text{vector("USA")} - \text{vector("New York")} + \text{vector("London")}$$

and find the word (“UK”) that is closest to the resulting vector using the cosine similarity measure. This example demonstrates that embeddings have geographical, and other relationships embedded in them.

Word embeddings are useful because they can capture “pseudo-meanings” through similarities measured by distances between vector representations. However, although word embedding techniques such as Word2Vec represented a major step forward in natural language processing, they are nonetheless limited by their inability to capture sentence-level contexts (where the same word has an entirely different meaning depending on the context in which it is used in a sentence). Word embeddings also ignore more distant words that fall outside of CBoW or skip-gram range, and do not consider word order within the range. Word2Vec’s usefulness is also constrained by its requirement that the context window length is fixed and must be specified *a priori* by the analyst.

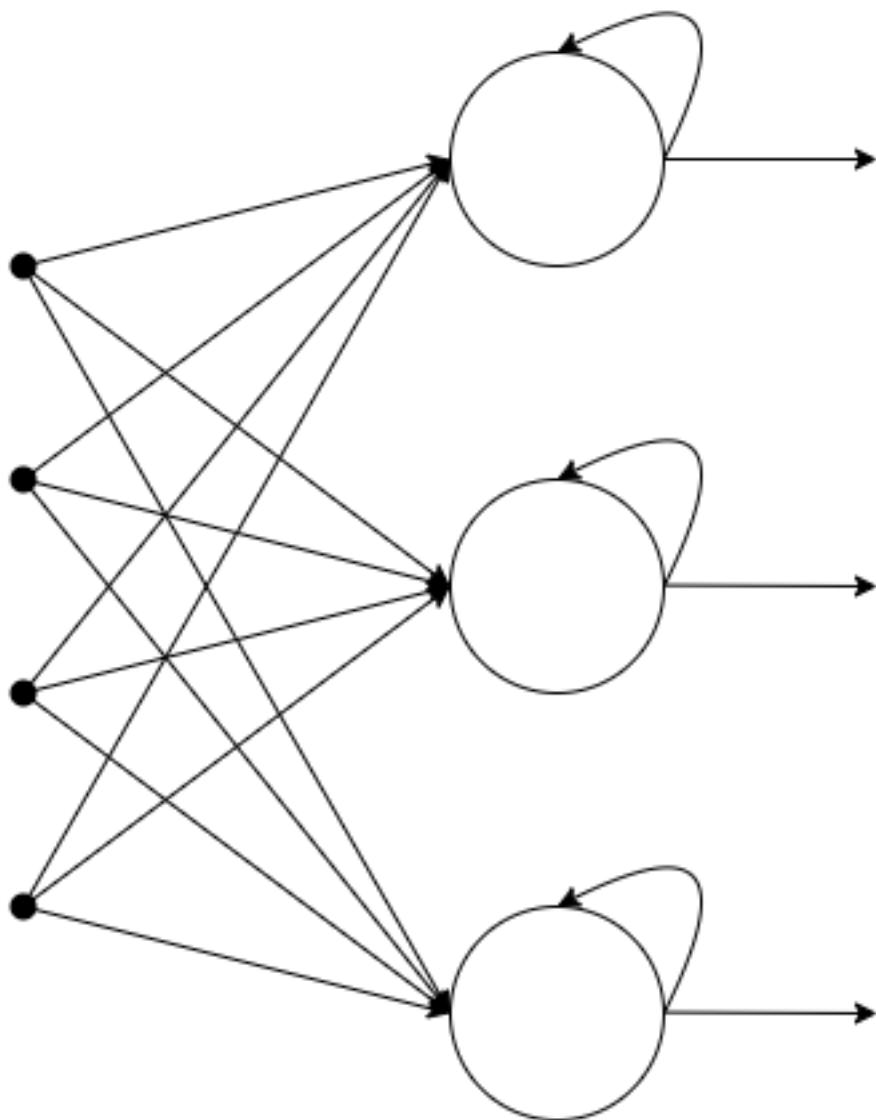
⁷ Mikolov, T., Chen, K., Corrado, G., and Dean, J., “Efficient Estimation of Word Representations in Vector Space,” 1301.3781.pdf (arxiv.org)

⁸ Cosine similarity is discussed in Appendix 2A.

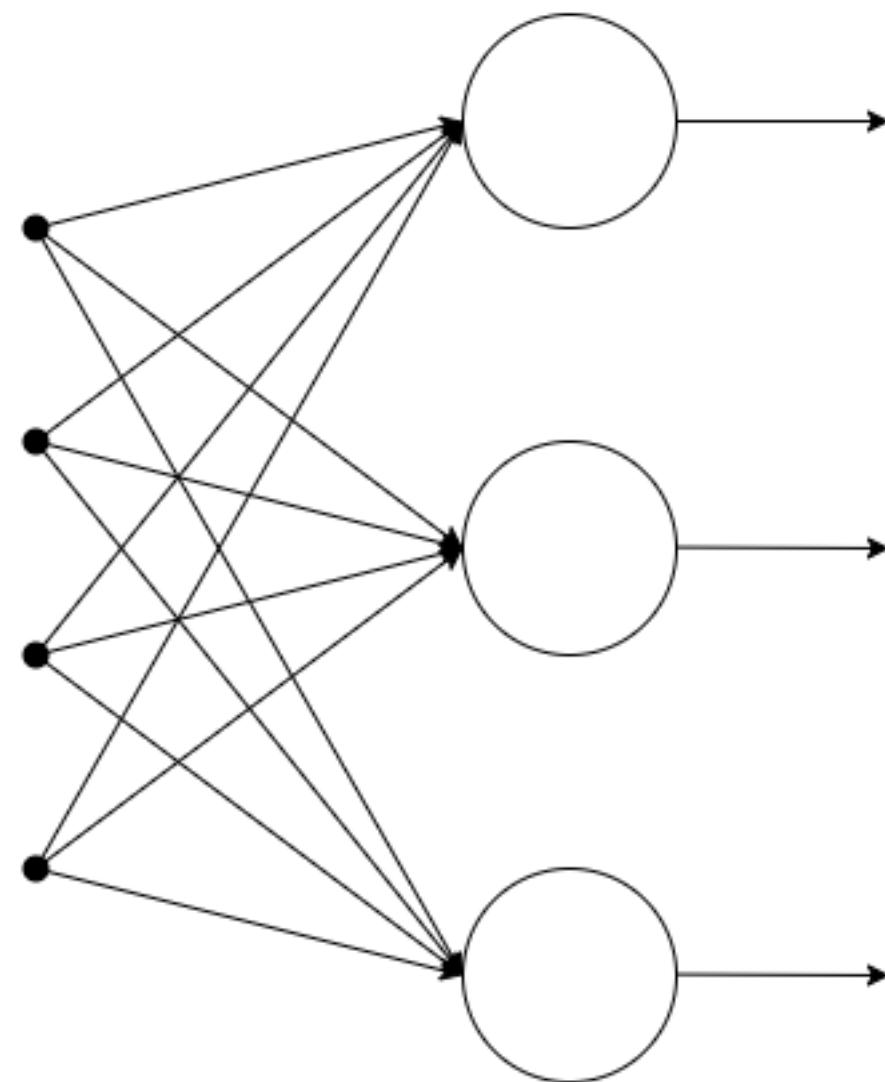
10.3.2 Recurrent Neural Networks (RNNs)

An alternative architecture that has been used to capture the relationship between distant words is a recurrent neural network (RNN). In a basic version of RNN, a memory (or, hidden) cell, is present. The hidden cell's value at time t is a function of the current inputs, as well as its value at time step $t - 1$. At any time step, the RNN combines the inputs from the current time step as well as the values stored in memory to generate the output. [**Figure 10.3**](#) shows a basic RNN and a feed-forward neural network (FNN).

Recurrent Neural Network Structure



Recurrent Neural Network



Feed-Forward Neural Network

Figure 10.3 Recurrent and Feed-Forward Neural Networks

RNNs are trained like feed-forward neural networks, by unrolling the network in the time domain. In other words, the same RNN cell can be thought of as separate units in a feed-forward network with the inputs moving through the cells in each time step. Similar to the training of FNNs, backpropagation is used to train the RNNs with the cost function being evaluated only at the final time step.

RNNs do not treat words in a sentence as if they are independent of one another. Rather, RNNs are designed to handle ordered (sequential) data and they operate iteratively (i.e., in a loop) by combining the current inputs with the values stored in the hidden units to generate output for a time period. The output values are stored in the hidden units, which then become part of the output calculations in the next time period. For this reason, RNNs are sometimes known as autoregressive neural networks. They can capture the order of words in a sentence and, potentially, they can also handle dependencies between one word and another word any number of places away in a sentence. RNNs use data sequentially, which means that they process each observation one-by-one. Whereas Word2Vec requires that each vector be of the same length, RNNs can handle variable length vectors.

However, generic RNN models face the issue of *vanishing gradient*, discussed in Chapter 7, limiting their ability to learn long-range dependencies. This limitation means that the further away a word is in a sentence, the lower its impact on the current value. The gradients can also explode if activation functions with derivatives that can increase in value from step to step, which is known as the *exploding gradient* problem.

With RNNs, therefore, words a long way from the current one can have a very small connection with the current word, and hence only a weak link with it in the model. Yet it might be the case that a word a long way from the current one has the most important association with it. For instance, consider the sentence, “the analyst stopped using MGARCH because it is too hard to estimate.” Here, the most important word connection with “estimate” is “MGARCH,” which is seven places away. Thus, RNNs are not ideal for capturing long-range dependence between words. A particular class of RNNs, known as long short-term memory (LSTM) models, was developed to mitigate this issue by using gates to regulate information flows. However, an entirely different family of models known as transformers can capture long-range dependencies in a more natural way – these are discussed in Section 10.4.

10.4 Transformers and LLMs

Transformers were the next major development in models for natural language processing.⁹ They are a class of feedforward neural network models based on the concept of “attention.” The original transformers consisted of an *encoder* and a *decoder*. The encoder is the unit that processes text, converts it to vectors and “understands” the text. The decoder is the unit that generates text based on the input or prompt it receives. This type of structure with both an encoder and a decoder is called a sequence-to-sequence structure. The transformer architecture has evolved from its beginnings with some later transformers specializing only in encoding and others focusing only on the decoding part. The basic structure of a transformer is shown in [Figure 10.4](#).

Transformers attempt to create a semantic understanding of a sentence by learning the relationships between all the words that it contains, which allows them to capture long-range dependencies in sentence structures. Each sentence is encoded into a set of vectors with one for each word. Consider again the sentence, “the analyst stopped using MGARCH because it is too hard to estimate.” To analyze it correctly, it is essential to link the word “it” with “MGARCH” rather than “the analyst.” The transformer does this by calculating, for each word, a similarity score between a *query vector*, which captures the information that the model seeks, and a *key vector*, which captures the usefulness of each word for the query. The similarity scores are then standardized and used as weights for each word. Here, the similarity score for “it” would be highest for “MGARCH.”

The *attention* mechanism is a key feature of transformers and other deep learning models. It is applied to each set of words repeatedly to encapsulate different relationships, resulting in it emphasizing some parts of the text being processed that are determined to be important. In this way, the model calculates the joint probability distributions across sequences of words, encapsulating the word combinations that are most likely to occur.

Transformers have proven much better at most NLP tasks than RNNs. They are deep learning neural networks that have many layers. They can make use of parallel processing capabilities and can be trained much faster than RNNs. Consequently, they have rendered Word2Vec as well as RNNs outdated. The structure of transformers enables them to construct “universal word representations” – i.e., it makes them better at applying their training to a variety of tasks.

Additionally, transformers make use of “contextualized embeddings,” which means that words have different vector embeddings when they are used in different contexts, allowing the models to differentiate successfully between instances of a specific word that may have more than one meaning (e.g., “set,” “speech,” and “rose”).

Transformers operate on whole sequences of text input simultaneously. This makes them better at capturing long-range dependencies between words and allows them to have more efficient parameter estimation when compared to RNNs. On the downside, transformers are computationally more intensive than RNNs. Also, when compared with other types of neural networks, transformers are more difficult to interpret due to their complex, multi-layered structure. Transformers' relative opaqueness can make the identification of the source(s) of any issues, errors, or problems challenging.

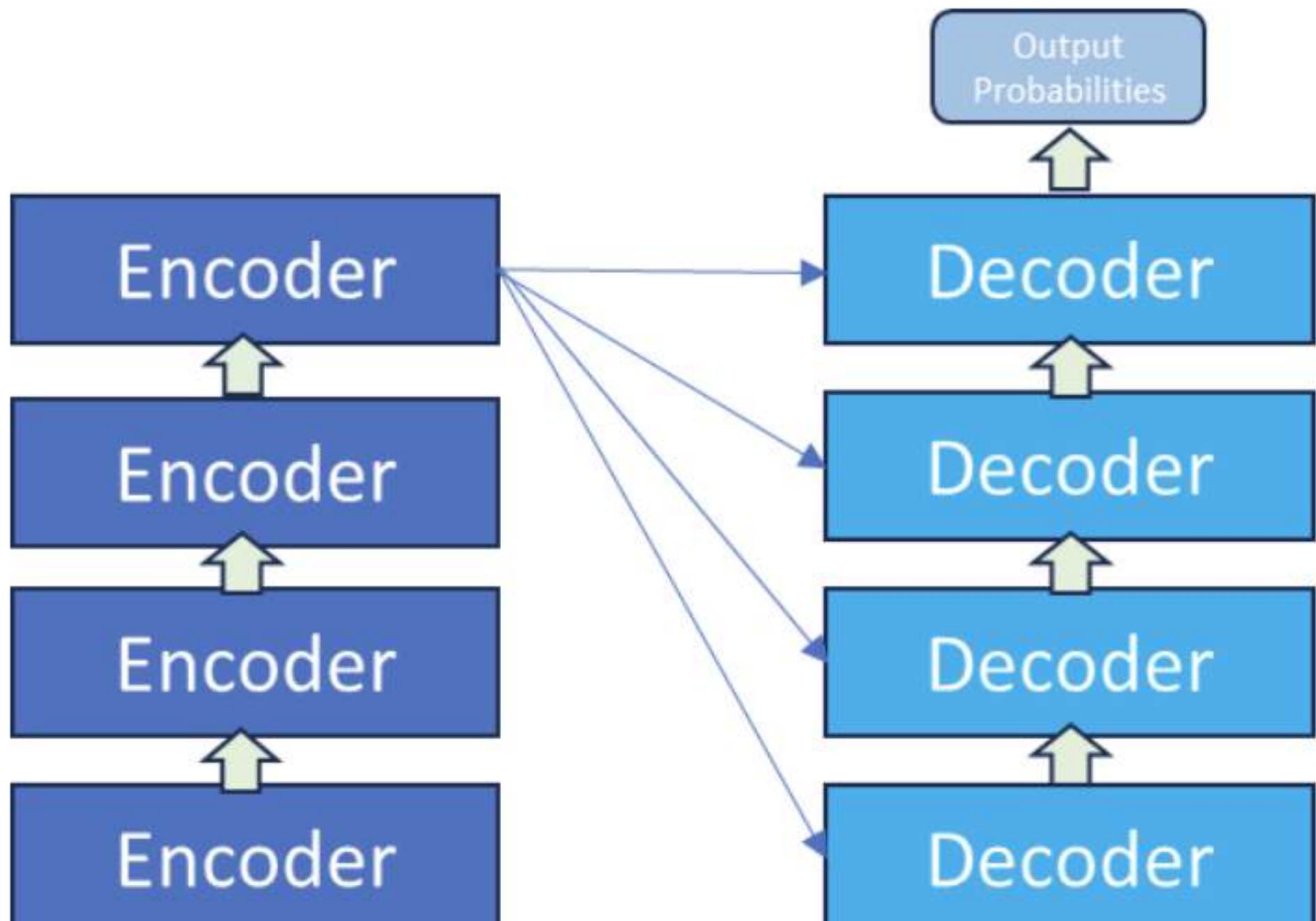


Figure 10.4 Basic Architecture of a Transformer

⁹ Vaswani, A., Shazeer, N., Parmar, N., Uszkorei, J., Jones. L., Gomez, A., Kaiser, L.:., Polosukhin, I., "Attention is all you need", <https://doi.org/10.48550/arXiv.1706.03762>

10.4.1 Large Language Models

Large language models (LLMs) are predominantly based on transformers that are pre-trained on a range of vast textual datasets. At the heart of LLMs is a large neural network with billions of parameters that is trained on a very large corpus of text. The weights of the parameters are determined using gradient descent or other methods similar to the process used for training a basic neural network.

The LLMs can be either autoencoding or autoregressive, or a combination of both. An autoencoding LLM is trained to encode sentences by masking some of the input and predict the masked words using the other parts of the input. An autoregressive LLM, on the other hand, predicts the next word or part of a word; this makes them useful for generating text. A combination autoencoding/autoregressive LLM is more versatile than the other two types, as they have both the encoder and decoder components. Although BERT is an example of an autoencoding LLM, the GPT family of LLMs are examples of autoregressive LLMs. BART is an example of a combination LLM model. They are discussed in Exhibit 10.1.

Training an LLM

The training of an LLM starts with cleaning the training data (corpus) which includes filtering, removal of duplicates, noisy data and punctuation marks as well as resolving or removing ambiguous data. The data also should be balanced, which entails adjusting the class distributions to have a fair representation and avoid any biases. The data is then separated into small parts or tokens to enable efficient processing. This is followed by encoding to store the positional information of sequences of text. The model is then pre-trained on the textual data usually without any labels. Through pre-training, the LLM understands the relationship between the words in the corpus it is trained on.

Once pre-training is completed, the models may require fine-tuning to transfer the knowledge gained during pre-training to conduct specific tasks. To accomplish this, the model may be fed labeled data relevant to the specific task being handled.¹⁰ The model is then modified in a process called “alignment” to conform with human preferences,¹¹ beliefs, and principles. During this process, feedback from users may be used to improve the model. The decoding module employs different strategies to generate output based on the input text. Finally, the model is optimized for production use by turning off some lower-level neural network layers, dropping unimportant weights, and reducing the precision of weights. The training process of LLMs is depicted in [Figure 10.5](#).¹²

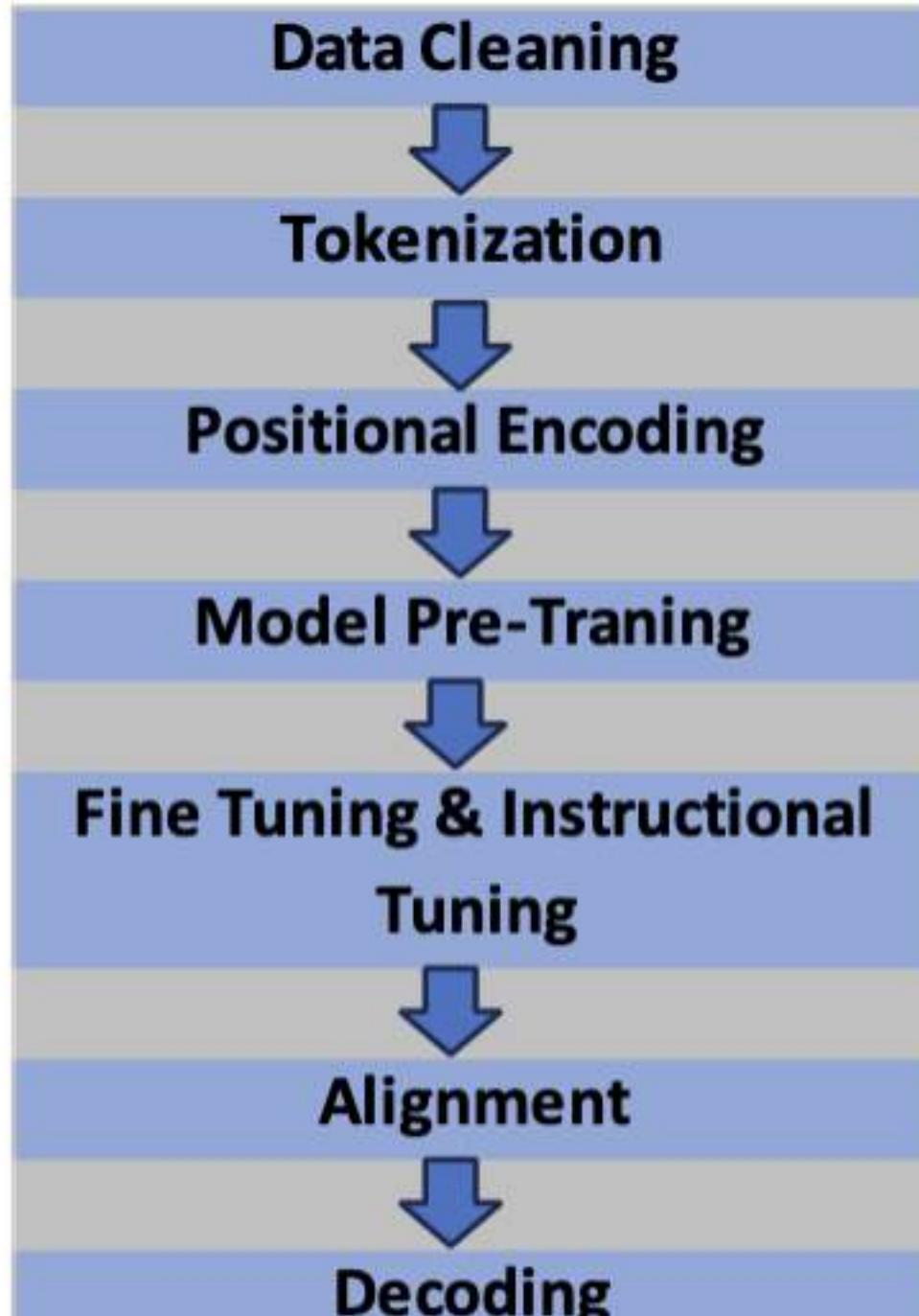


Figure 10.5 The Training Process of LLMs

BOX 10.1: EARLY LLMs

BERT

BERT (bi-directional encoder representation from transformer) is a class of LLMs that was introduced by Google in 2018. BERT is an example of an autoencoding LLM. It comprises about 100 million parameters and was trained on a corpus of books and Wikipedia pages. Since then, several variations have been developed that have slightly different structures or that are tailored to specific types of applications. For instance, RoBERTa (Robustly Optimized BERT pre-training Approach) allows for different parts of a sentence to be hidden in each training cycle, which ensures that the model will be more robust to diversity in the training text. RoBERTa also uses a larger training dataset with more training cycles than BERT, and it has also been trained with web crawled data from many languages.

Another variant is FinBERT, which is a model that was further trained on financial documents from Reuters including corporate statements and analyst reports. This fine-tuned the transformer for optimal use in financial applications such as sentiment analysis.

BERT and its descendants include only an encoder side, which converts the words into embeddings and a *positional encoder* that captures the locations of words within sentences; there is no decoder as there would be in an autoencoder model, for instance. These encoders are bi-directional, which means that to determine the meaning of a sentence, they can examine words appearing both before and after a given word.

A set of hidden layers in a transformer model employ self-attention to capture the relationships between words. Several self-attention “heads” are used in parallel, with each capturing the linkages between different sets of words in the sentence. Because BERT includes only an encoder, it is not generative (i.e., it cannot create new instances), but it can be used for applications such as classification.

The first stage of training LLMs such as BERT involves training the model on a large body of unlabeled text. The second stage involves two aspects:

- “Masking” words randomly within sentences and getting the model to predict the hidden words; usually, 15% of words are masked. This is known as self-supervision as the hidden words are known and therefore act as a supervised technique.

- Next sentence prediction (NSP), where sentences are extracted in pairs from documents and the model is tasked to predict an entire sentence from the preceding one rather than just a single word.

BART

BART (Bidirectional and Auto-Regressive Transformer) was developed in 2019 by a team of researchers from Facebook AI.¹³ It is another class of transformer model incorporating both encoder and decoder layers. Incorporating a decoder makes it able to generate output text rather than merely being able to analyze inputs and encode them in vectors. This means that BART can be used for a wider range of tasks such as document summarization, language translation, and creative writing, although it requires greater computational resources than encoder-only models.

BART is trained by deliberately distorting blocks of text (e.g., by shuffling, removing, or replacing words) and getting the algorithm to learn to generate the original forms. The autoregressive decoder allows it to generate sentences one word at a time while remembering the previous words in those sentences. BART has been found to have provided superior results in many standardized test applications, such as machine translation, document summarization, and responding to questions, in comparison with alternative models. The original BART model is open source, but several pre-trained versions are now available in commercial packages that can be fine-tuned for specific applications.

¹⁰ This process is called low-shot task transfer.

¹¹ This process is called Reinforcement Learning through Human Feedback (RLHF).

¹² Minaee, S., Mikolov, M., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., Gao, J., "Large Language Models – A Survey", <https://doi.org/10.48550/arXiv.2402.06196>

¹³ Meta Platforms Inc. was formerly known as Facebook Inc.

10.4.2 Cloud-based LLMs

Because the corpus is huge for LLMs, and the estimated weights matrix is in the millions, billions or even trillions, training is only performed once. It would be infeasible to train separately on this volume of data for each task. Most LLMs are so large that it is also infeasible to store them locally. Hence, they are known as cloud based LLMs.

The best known LLM is Chat-GPT (also known as GPT-3.5), which is one of a family of generative pre-trained transformers (GPTs) developed by OpenAI. The first generation of this family was GPT-1, which, like BERT, was a model of approximately 100 million parameters that was pre-trained using a large (unlabeled) corpus of unpublished books. Then, at a second stage, the model was fine-tuned for specific applications using smaller, labeled datasets. These applications could be, for example, sentiment analysis, responding to questions, summarizing a document, or classification.

Before we continue, it is useful to mention two terms here: *low-shot task transfer* and *no-shot fine-tuning*. The first term refers to fine-tuning a model that has been already trained with a large data set using only a small set of new examples. The second term is a fine-tuning technique wherein we simply rely on the prior training of the model without any fine-tuning with new examples.

Each generation of GPTs has become better at “low-shot task transfers,” implying that the number of labeled examples required to fine-tune the model for a specific application has declined. GPT-2 was released in 2019 with two versions – the base model with around 124 million parameters and GPT-2-Large with 774 million parameters. GPT-2 removed the need for the second stage fine-tuning, so that “no-shot fine-tuning” became possible, meaning that the model can adapt to new situations without any supervised examples at all. It was trained on a much larger and more varied corpus of textual material than GPT-1, including millions of documents created from scraped webpages. GPT-2 contained around 1.5 billion parameters.

GPT-3 was the next generation of the model, released in 2020 and comprised close to 175 billion parameters. It was trained on a much larger body of web text together with books and the English Wikipedia pages. This further enhanced the model’s ability to adapt to different kinds of tasks, and gave it *generative capacity*, whereby it could create new instances of text from human instructions and examples contained within the training set. For instance, it could create synthetic news articles that were comparable to those generated by human writers. This is a form of generalization, which is different from simply memorizing sentences and repeating them verbatim when tasked to create material on a specific topic. More recent LLMs are presented in [**Box 10.2**](#).

BOX 10.2: RECENT LLMS

Chat-GPT (GPT 3.5) is a freely available generative LLM released in 2022 with close to 175 billion parameters, which improved GPT-3 by making the output more useful and less likely to provide offensive material. This enhancement compared with GPT-3 was achieved by showing the model labeled examples of the required output, instructing the model to create additional instances, then allowing the user to provide ratings for the generated outputs. This is

known as reinforcement learning from human feedback (RLHF). Through this process, the LLM was fine-tuned to be able to follow instructions more closely, providing informative responses to queries while refusing to perform tasks deemed inappropriate.

GPT-4 has recently been introduced, which at the time of writing (January 2024) is the latest and most developed member of the GPT-family. As well as further increasing the size of the training corpus and the number of parameters in the model to an estimated 1.76 trillion, the specification includes significant new capability to be able to accept images as an input rather than purely text. As well as the publicly available data used to train previous GPT generations, it also employed proprietary data under license, such as Google Search results. Most details of the training process and architecture have not been published.

Bard a rival LLM by Google, was introduced early in 2023. It is a fine-tuned version of their LaMDA (Language Model for Dialog Applications) model. Bard was trained using "Infiniset," a vast corpus including Wikipedia pages, books, and scraped webpages, although the specific details are proprietary. Like the most recent version of GPT, Bard also now has the capability to analyze images – for instance, being able to identify the breed of a dog from a photo of it.

Gemini, a multi-modal LLM that is an updated version of Bard, was released by Google at the end of 2023. It is a successor to LaMDA and PaLM2 models. Gemini was trained on text, video, and audio. According to Google, it is the first LLM to outperform human experts in massive multi-task language understanding (MMLU). Other recent models released include Meta's Llama, Google's Gemma, and Databrick's DBRX.

10.4.3 Chatbots

A chatbot is a conversational interface that interacts with a human user. It can be either web-based or, through automated speech recognition and text-to-speech synthesis, phone-based. Chatbots represent an important application of NLP, and they have been increasingly used by organizations to reduce costs and improve response times to queries. For instance, banks often use chatbots as automated customer service representatives that can deal with queries and either provide the required information directly or put the customer in contact with the most appropriate department in the bank.

The most basic forms of chatbots simply provide responses to standard lists of frequently asked questions (FAQs). They have no real conversational ability and are of limited use in customer service, extracting and quoting information that is probably already available on an organization's website. More complex chatbots are "flow-based," meaning that they can have an interactive, multi-step conversation with a customer by asking more detailed follow-up questions in response to a customer question or comment. For instance, if a customer states that they are unhappy with the service they have received, the chatbot might ask whether it relates to a mortgage or savings product, or whether they transacted in a branch or on-line.

The quality of chatbots is being improved significantly using LLMs, which can give a deeper and more accurate understanding of customer requests than earlier NLP models, therefore providing more informed answers. They can also draw on a wider range of material than simply what is on the organization's website. We are fast approaching the point where chatbot responses will be almost indistinguishable from those given by a human customer service agent.

10.4.4 Using LLMs – Prompt Engineering and Temperature

Although building an LLM is challenging, using such models poses its own challenges. This is where *prompt engineering*, the art and science of designing prompts for effective use of LLMs, comes in. Prompt engineering is an iterative process for getting the best results for a given task from an LLM. The typical guidelines for drafting good prompts are like those for writing any good letter, memo, or article. They include:

- Prompts should be direct and concise.
- Prompts should provide the context to the model by providing reference material and/or output from other tools.
- Prompts should split tasks into smaller and more manageable tasks for better responses.
- Prompts should provide sufficient information to obtain the best results.
- Prompts should be flexible for use in different models.

- Prompts should allow the model time to find good results (e.g., asking that the analysis be finished before answering, asking the model to see if anything is missing in its answers, etc.).

During the prompt design process, users may supply the model with a few relevant examples, so that the model can generate more context-specific responses. This technique is known as *few-shot learning*. It is also good practice to ask for output in a structured format so that it can be easily used elsewhere. Prompts with different personas, or styles may be used to test how the model responds to a query asked in different ways. The model developers can use the output from these tests to fine-tune the model to avoid undesirable responses.

Users can also control how an LLM behaves by choosing a hyperparameter called *temperature*. This parameter is used to change the shape of the probability distribution of tokens being predicted by the model. In OpenAI, temperature ranges from 0 to 2. If temperature is set below one, the probability distribution is narrowed such that only the most likely tokens are selected. If it is above one, the probability distribution is widened and flattened, resulting in outlying tokens also being selected. If the temperature is one, then the probability distribution is not altered. The lower the temperature parameter is, the more predictable the output is. Other LLMs have similar functionality.

There are other parameters that users can adjust to control the behavior of LLMs. For example, in OpenAI, users can choose the model version, the maximum number of tokens generated, the penalty levels for the tokens that have already appeared, and the frequency of their usage in the generated text. Users can select similar parameters in other LLMs to control their responses.

10.5 Applications of Generative AI and LLMs

Generative AI and LLMs are one of the most recent and exciting developments in AI. Users in academia and industry are currently experimenting with these tools. It is not feasible to present all such applications in this section. An overview of the current applications of these technologies is presented below:

- *Textual Processing:* This is the area that has seen the most usage so far. There are many applications of GenAI and LLMs in this area. Examples are:

- Analysts can use Generative AI to assist with writing reports as well as for generating summaries of documents. This is particularly useful in legal analysis and litigation support.
- GenAI-based tools can be used to monitor news feed, customer opinions, and social media for posts on specific companies and to help identify any potential risks.
- LLMs can be used for analyzing incoming market information, economic data, and news flows as well as for predicting short-term movements in prices.
- SEC filings and other reports by companies can be analyzed quickly using GenAI based tools for detecting any emerging risks. Compliance monitoring is another area where LLMs can be effective.
- LLMs can be deployed for translating text written in foreign languages.
- GenAI and LLMs are useful for answering students' questions and developing tests.
- LLMs can play an important role in credit underwriting. They are useful for processing both structured and unstructured data and generating reports for credit analysts. LLMs will be useful in conducting an in-depth examination of all available materials on borrowers, as well as in reducing the time taken to arrive at credit decisions.

Popular GenAI tools like Gemini, ChatGPT, and Llama have been used successfully in textual processing. Bloomberg GPT and FinGPT are examples of LLMs developed for analyzing financial news flows and for generating trading signals. Several investment firms like AQR, BlackRock, and Bridgewater are using LLMs for analyzing trends in financial news.

- *Code generation:* Developers can use LLMs to generate and debug code for their projects, which could reduce software development costs. LLMs can also be useful for generating documentation. Examples of LLMs with code generation capabilities include ChatGPT, GitHub CoPilot and CodeLlama.
- *Chatbots and virtual assistants:* Chatbots can be developed for applications such as online customer support, providing information to risk professionals on existing and upcoming regulations, helping medical professionals analyze symptoms, assisting investors and investment advisors in identifying potential investments for their portfolios, etc. Most LLM providers have tools for developing custom chatbots and

virtual assistants. For example, OpenAI provides functionality for building chatbots and virtual assistants using the GPT family of LLMs.

- *Fraud and anomaly detection:* GenAI's capacity to automate the monitoring and processing of a company's own internal data could make an even more significant contribution. Equipped with the latest statistical tools and algorithms, risk managers could leverage AI to compile and scrutinize transaction data across various departments for anomalies or outliers linked to specific suppliers or operations. Mastercard's *Decision Intelligence Pro* is an example of a dedicated GenAI tool for fraud detection.
- *Cyber security:* LLMs can be used to constantly monitor emails and other data streams for potential cyber threats, cross-referencing them with an institution's system profile to pinpoint specific vulnerabilities such as malware. After identifying a threat, the system could automatically alert risk managers and other relevant individuals in real time. The true power of the AI system would be to then proactively source patches for these threats directly from approved software vendors for system engineers to implement. This would not only streamline the threat identification process but also enable the risk team to preemptively coordinate a solution, rather than delegating the problem to another team. Examples are Cisco Systems' custom LLM to detect malware. and Crowdstrike's Charlotte AI.
- *Claims processing* is another area where LLMs can be useful for processing documents. GenAI applications can be used for developing applications that can automate the interviews with customers that are filing claims.
- *Image generation and image processing:* GenAI and LLMs can be used to generate images from text inputs. This can be very useful in providing rapid prototype development of images as a starting point for artistic projects such as marketing materials, book covers, and illustrations to be used in slide decks. DALL-E-2 is an example of such systems. Image recognition systems are quite useful for assessing security threats as well as for medical image processing.
- *Text, speech, and video conversion:* Multimodal LLMs will be useful for converting text to speech and for generating video from text. They are also useful for transcribing text from audio. Examples are OpenAI's Whisper, which converts audio to text and text-to-speech (TTS) that convert text to audio. Gemini and OpenAI's Sora create realistic video from text instructions.
- *Autonomous driving vehicles:* This technology has been under development since the 1970s. While there are currently no systems that offer completely autonomous driving, semi-autonomous driving systems have been deployed by various companies, notably Tesla. Artificial intelligence is a crucial element of

autonomous driving technology. Although there is considerable interest in this technology, there are some concerns about it. It is expected that the technology will be more robust in the future, resulting in increased adoption.

Although text generation capabilities have been used to produce good first drafts, others like images and videos are still at an experimental stage, with their usage currently limited to generating logos, mockups, and basic videos. As these technologies are adopted and used widely, the quality of the text generation function is likely to converge to near final draft quality by the end of this decade. Similarly, code generation can also progress from first drafts that need to be modified by experienced coders before use to production level code quality in a few years. A recent forecast on the evolution of GenAI and LLMs in different modalities is shown in Figure 10.6.

| | Pre-2020 | 2020 | 2022 | 2023 ? | 2025 |
|--------|--|-----------------------------------|--------------------------------|--|--|
| Text | Spam detection Translation Basic Q&A | Basic copywriting First drafts | Longer form Second drafts | Vertical fine-tuning gets good (scientific papers, etc.) | Final better human-like output |
| Code | 1-line auto-complete | Multi-line generation | Longer form Better accuracy | More languages More verticals | Text generation (dramatic improvements) |
| Images | | | Art Logos Photography | Mock-ups (product design, architecture, etc.) | Final (product design, architecture, etc.) |

Figure 10.6 The Evolution of Generative AI¹⁵

¹⁵ Generative AI: A Creative New World | Sequoia Capital

10.6 Chapter Summary

GenAI and LLM technologies are evolving, with ramifications far beyond the technology sector. There is tremendous scope for applying these technologies in various sectors of the economy. Although it is still too early to make a prediction, it seems fair to say that they will be playing a big role in the future as they are adopted by businesses and others to improve processes. This is likely to result in productivity gains and fundamental changes in how workers perform their work and the nature of their tasks.

Appendix 10.A Operations With Word Embeddings

Word embeddings are vector representation of words used in natural language processing applications. They provide a dense representation of words, as opposed to the sparse representation provided by other approaches like one-hot encoding. Synonyms and words that belong to a particular category such as geography have embeddings that are close to each other.

The purpose of this exhibit to illustrate how embeddings can be used to perform quasi-mathematical operations. We use GloVe embedding vectors¹ and calculate cosine similarity or similarity score, which is the dot product of two vectors, for word pairs. If a pair of words are close to each other, the similarity score would be close to 1.0. If they are not close to each other, the score would be closer to 0 than 1.0. For example, China and Japan have a similarity score of 0.84. [Table 10.A.1](#) shows the similarity scores for other word pairs that are similar.

Table 10.A.1 Similarity Scores for Similar Words

| Word 1 | Word 2 | Similarity Score | Category |
|--------|--------|------------------|------------|
| China | Japan | 0.84 | Countries |
| King | Queen | 0.78 | Royalty |
| Gold | Silver | 0.95 | Metals |
| Europe | Asia | 0.83 | Continents |

We can also expect that dissimilar word pairs to have low similarity scores. For example, Africa and Tokyo belong to different categories: continent and city. The similarity score for this word pair is 0.40. [Table 10.A.2](#) shows the similarity scores for other word pairs that are dissimilar.

Table 10.A.2 Similarity Scores for Dissimilar Words

| Word 1 | Word 2 | Similarity Score | Category |
|-----------|--------|------------------|-----------------|
| Australia | Tokyo | 0.40 | Country, City |
| King | Man | 0.53 | Royalty, Gender |
| Gold | Dollar | 0.50 | Metal, Currency |
| Banana | Rose | 0.30 | Fruit, Flower |

Another example of operations that can be performed using word embeddings is the testing of word analogies. A word analogy is of the form 'a is to b' as 'c is to d'. For example, consider the word pairs Europe – Spain and Asia – China. Europe and Asia are continents, while Spain and China are countries. If we subtract Spain from Europe, and add China back, the resulting embedding vector is close to Asia in embedding space. [Table 10.A.3](#) shows the similarity scores for other examples of word analogies². The GloVe embeddings of the words in this analogy are presented in [Figure 10.A.1](#). The embeddings were plotted after they were reduced to 2 dimensions using the t-SNE technique³. It can be seen from this figure that Europe and Asia are close to each other, similarly Spain and China are also close to each other in the embedding space.

Table 10.A.3 Word Analogies

| Operation | Closest Word | Similarity Score |
|-------------------------------|---------------------|-------------------------|
| America - Washington + London | UK | 0.77 |
| King-Man+ Woman | Queen | 0.86 |
| Gold-Dollar + Pound | Silver | 0.80 |
| Europe-Spain + China | Asia | 0.79 |

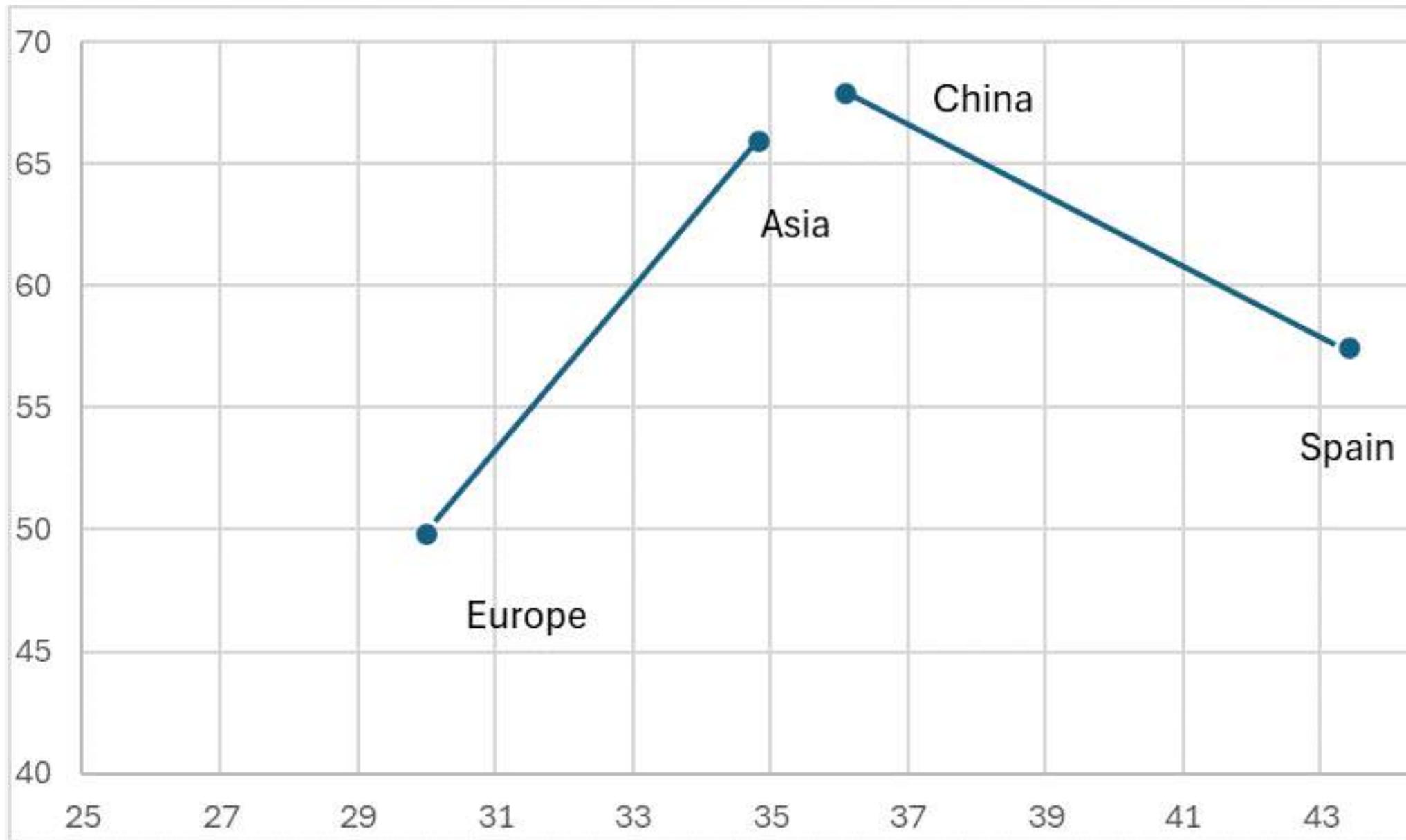


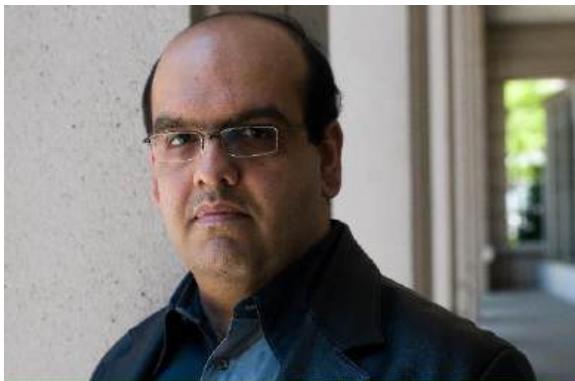
Figure 10.A.1 Word Pairs in Embedding Space After Projecting to 2 Dimensions

¹ GloVe: Global Vectors for Word Representation (stanford.edu)

² Please Jurafsky, D., and Martin, J.H., Speech and Language Processing, 6.pdf (stanford.edu) for a more in-depth discussion on this topic.

³ TSNE — scikit-learn 1.5.1 documentation

Appendix 10.B Rama Cont on Generative AI



Rama Cont, PhD, is Professor of Mathematical Finance at the University of Oxford, and Head of the Oxford Mathematical and Computational Finance Group.

He has authored several books and more than 100 research papers on mathematical modelling in finance, quantitative risk management, liquidity risk modeling, algorithmic finance, and applications of machine learning in finance.

Cont has worked for numerous financial institutions, CCPs, exchanges and regulatory bodies on a range of topics including pricing models for derivative securities, high-frequency market making algorithms, liquidity stress testing and risk management systems for CCPs.

He received the Louis Bachelier Prize from the French Academy of Sciences in 2010 and the Royal Society Award for Excellence in Interdisciplinary Research in 2017 for his work on systemic risk modelling.

Generative Artificial Intelligence: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

10.1 State whether each of the following statements is true or false and explain your reasoning.

- A. Word2Vec can generate blocks of text, such as article summaries and responses to questions.

View Answer

False. A large language model such as GPT could generate long pieces of text, but Word2Vec was developed to represent text as numerical vectors and to be able to analyze that text – for example, looking for similar words or groups of words.

- B. Recurrent neural networks use a static embedding that cannot allow for word ordering in a sentence.

View Answer

False. Recurrent neural networks operate in a dynamic fashion, iterating through sentences so that they can capture the ordering of words through their “autoregressive” structure.

- C. Transformer models can be trained to understand different contexts that the same word is used in.

View Answer

True. This is one of the benefits of such models. The “attention heads” in the model can link together words in different parts of a sentence and so they can capture different ways that the same word will be used in other contexts.

- D. Cloud-based large language models can be accessed remotely or alternatively the training corpuses can be downloaded and stored locally.

View Answer

False. Cloud-based large language models only exist in the cloud. The training databases and weights matrices are huge – far too big to be downloaded and stored locally. These details are also usually proprietary to the model developer.

- E. BART includes both encoder and decoder layers.

View Answer

True. BART includes both a bidirectional encoder of the sort incorporated into BERT and an auto-regressive decoder like the one incorporated into the GPT family.

- F. Large language models can only analyze textual input and not images.

View Answer

False. This was true until recently, but both Gemini by Google and the latest version of GPT (GPT-4) can handle images as inputs, which the model can analyze.

10.2 Explain the differences between the continuous bag of words and skip-gram model architectures.

View Answer

The continuous bag of words is tasked to predict a masked word based on a few context words before and after that, whereas the skip-gram tries to predict the context words around a particular word.

10.3 What are the disadvantages of the traditional count bag of words approach to representing documents as vectors of numbers and how does Word2Vec overcome these?

View Answer

The traditional bag of words (BoW) model treats each word as independent from all other words in a document – the only aspects it captures are whether a particular word appears in a document (and how many times if the count version is used). This limitation implies that the BoW cannot impute any meaning to words and nor can it propose synonyms for a particular word. The word embedding is also very inefficient because the vectors created will each be of the same length as the entire vocabulary but very sparse (containing predominantly zeros). Word2Vec uses a variable length embedding that can reduce the dimensionality by creating a dense representation and encodes the position of words within a document so that their meaning can be captured.

10.4 What is the key difference between Recurrent Neural Networks (RNN) and Feedforward Neural Networks (FNN)?

View Answer

At any time step, the RNN combines the inputs from the current time step as well as the values stored in memory to generate the output. Refer to Figure 10.3.

10.5 Describe the training process of LLMs.

View Answer

The training starts with cleaning the training data (corpus) which includes filtering, removal of duplicates, noisy data and punctuation marks as well as resolving or removing ambiguous data. Then, it should be separated into small parts or tokens to enable efficient processing. This is followed by encoding to store the positional information of sequences of text. The model is then pre-trained on the textual data usually without any labels. Then, the model is finetuned to transfer the knowledge gained during pre-training to conduct specific tasks. Then it is modified by the process called alignment to conform with human preferences. Then generate outcome by decoding the model. Finally, the model is optimized for production use by turning off some lower-level neural network layers, dropping unimportant weights, and reducing the precision of weights.

Refer to Figure 10.8.

10.6

- A. What are the guidelines for drafting good prompts?

[View Answer](#)

Prompts should allow the model time to find good results (e.g., asking that the analysis be finished before answering, asking the model to see if anything is missing in its answers, etc.).

B. How to control the behavior of LLMs?

[View Answer](#)

Users can also control how an LLM behaves by setting a parameter called temperature. This parameter is used to change the shape of the probability distribution of tokens being predicted by the model. If temperature is below one, the probability distribution is narrowed such that only the most likely tokens are selected. If it is above one, the probability distribution is widened and flattened, resulting in outlying tokens also being selected. If the temperature is one, then the probability distribution is not altered.

Module 3: Risk and Risk Factors

Learning Objectives

This module provides a comprehensive overview of the primary risks associated with AI development and deployment. It discusses the numerous challenges associated with the creation of a “fair” algorithm, highlighting the different sources of bias that might affect algorithmic fairness. It also addresses the twin problems of explainability and interpretability, and other noteworthy risks, including risk to human autonomy,

risk of AI-driven manipulation, reputational risk, existential risk, and global risks and challenges.

After completing this module, you should be able to:

- Describe and differentiate between the concepts of individual and group fairness.
- Describe various measures of group fairness.
- Discuss trade-offs associated with different concepts and measures of fairness.
- Describe sources of algorithmic bias and unfairness.
- Describe explainability, interpretability, and transparency.
- Describe techniques for making AI algorithms more explainable.
- Discuss risks posed by AI to human autonomy, safety, and wellbeing.
- Describe sources of AI-related reputational risk and strategies for mitigating those risks.
- Discuss global challenges and risks associated with AI.

1.0 Introduction

Technologies pose risks. Toasters burn, planes crash, and diagnostic tools fail. Similarly, AI¹ technologies are not immune to failure: We are only now beginning to understand the various risks associated with the use of AI technologies. The evolving nature of these risks requires us to adopt a dynamic and informed approach both to ensure beneficial use and to mitigate unintended consequences. In considering the risk implications of AI, it's important to recognize the following:

- AI is hugely pervasive in both range and variety of use. This poses challenges to risk prediction, as risks can take on different forms depending on the use case.
- AI technologies also have increasingly large capabilities, which enables us to use them in increasingly high-stakes settings. Doing so implies that any technological shortcomings can have significant consequences for people. For example, an automatically generated outcome of a loan application can profoundly affect the applicant. Similarly, a clinical diagnosis generated by a medical algorithm can have a long-lasting impact on the patient's life.
- AI development is rapid, which often poses challenges for impact prediction, risk measurement, and risk mitigation. A further consequence of this rapid development is the difficulty of establishing robust governance mechanisms that effectively address risks. This is especially a challenge for regulatory responses, which are often slow to develop and difficult to change once established. A slowness to regulate, on the other hand, may allow for

potentially harmful technologies to persist, and can lead to the normalization and entrenchment of harmful practices.

It is important that those developing and using AI technologies recognize and manage the risks. As will become clear, risks from AI are often interconnected: Lack of transparency can affect fairness and safety, algorithmic bias can lead to reputational risks, and so on.

¹ As with other parts of this course, this module focuses largely on machine learning (ML) systems, a subset of artificial intelligence (AI), so the terms AI, ML, and AI/ML may be used interchangeably.

2.0 Algorithmic Bias and Fairness

In this chapter, we delve into the complex and increasingly relevant topic of algorithmic fairness. AI systems are being deployed across myriad important domains, executing tasks that can have profound effects on people's lives. These include medical diagnostics, bail decisions, policing, loan approvals, college admission, and recruitment. Although algorithms are often hailed for providing more consistent and objective decision making than humans, they, too, can exhibit bias and create unfair results. By now, there are countless accounts of algorithmic bias and discrimination, some with severe consequences for the people subject to algorithmic decision-making.²

Although consistent decision making is desirable for many reasons, the downside of such consistency is that bias, if it occurs, becomes systematic. As a result, it is extremely important to ensure that algorithms do not exhibit such unwanted biases.

This has been recognized by regulators across the globe, who increasingly demand that AI systems be fair and non-discriminatory.

² See for example: West, S. M., Whittaker, M., & Crawford, K. (2019). Discriminating systems. *AI Now*, 1-33; Ledford, H. (2019). Millions affected by racial bias in health-care algorithm. *Nature*, 574(31), 2; Buolamwini, J., & Gebru, T. (2018, January). Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency* (pp. 77-91). PMLR.

2.1 What is Bias?

What does it mean for an algorithm to be biased?

The term “bias” is used differently across different disciplines and contexts, and it is useful to keep this in mind when referring to “algorithmic bias.” Data scientists and statisticians use bias to refer to a systemic error in the measurement or prediction process, which in turn leads to a discrepancy between the ground truth and the measurement or prediction. Within psychology and cognitive science, bias is a systemic error in judgment, or a systemic deviation from rationality. For example, a user putting more trust into a computational system than would rationally be justified is said to suffer from automation bias. When used within the context of algorithms, bias typically not only includes systemic errors of various kinds, but also the resulting unfair outcome for certain groups of people. Here, bias has become a value-laden concept that explicitly refers to the effects of algorithms on humans. We use a more neutral definition of bias.³

Definition: algorithmic bias is a systemic deviation of an algorithm's output, performance, or impact relative to some norm, aim, standard, or baseline.

We can make an additional distinction between explicit and implicit bias. In the former case, the algorithm's developer intentionally projects its own biases into the algorithm, leading to biased outputs. In the latter case, bias creeps in undetected through the algorithm's design or via the data used to train the algorithm. In those cases, bias is implicit and unintended.

³ See also Fazelpour, S., and Danks, D. (2021). Algorithmic bias: Senses, sources, solutions. *Philosophy Compass*, 16(8), e12760. <https://doi.org/10.1111/phc3.12760>

2.2.1 Individual Fairness

In college admissions, the prevailing belief is that admissions should be based on merit, with only the best students receiving offers. We often consider grades as a good indicator of merit, and so the natural consequence would be to offer places only to applicants with top grades. This is an application of the individual fairness doctrine. Because the relevant metric for college admission is scholarly achievement and nothing else, we avoid direct discrimination on the basis of protected characteristics, such as race, gender, or religion. If one applicant has higher grades than another, she is selected, no matter what other characteristics she might possess. Here, treating like cases alike means treating applicants that are alike in all relevant aspects (i.e., scholarly achievement) alike.

However, we quickly encounter limits to this seemingly straightforward principle of individual fairness. Not all students start out equally. Some might have had to overcome significant hurdles to achieve certain grades. If we truly are interested in merit, should we take this fact into account when making decisions? And if so, how should we weigh this fact against other relevant criteria? Is it fair to reject an applicant with top grades but admit a student with slightly lower grades whom we know to have displayed extraordinary resilience, motivation, and self-discipline to overcome socioeconomic hurdles or physical difficulties? Moreover, how would we determine what those hurdles were and translate them into numbers that allow us to compare applicants?

These questions have no easy answer. Different individuals will hold different views as to whether and how socioeconomic factors should influence college admissions. This is a limitation of the treating-like-cases-alike doctrine. Intuitive as it may be, it requires us first to agree on what counts as “alike,” which is often contentious. This issue becomes especially pronounced in the case of algorithmic design, where the designers need to be explicit about the relevant factors that feature in the decision-making process. These may, furthermore, vary with context.

2.2.2 Group Fairness

As opposed to individual fairness, group fairness does not consider the treatment of individuals, but instead looks at the statistical differences between groups. For

example, we may ask whether there are statistical differences between the admission rates of male and female college applicants. This would be a matter of group fairness, as we are interested in whether there are certain groups that are disadvantaged by the algorithm and that share a protected characteristic.

There are many different notions of group fairness, and which one is appropriate depends heavily on the context in which the algorithm is deployed. Broadly, these notions fall into two categories. The first category focuses on performance equality and compares algorithmic performance across different demographics. The second category focuses on output distributions without regard to the correctness or accuracy of those outputs. The main concern here is how a given outcome (e.g., being admitted to college) is distributed across different demographics.

Demographic Parity

One of the most important fairness notions with regard to this second category is demographic parity. Demographic parity requires that the distribution of predictions is identical across subpopulations. In the case of college admission, this means that the admission rate is the same across groups. In the case of lending, it means that the proportion of granted loans is the same across groups. Mathematically, demographic parity can be expressed as:

$$P(\hat{Y} = 1 \mid A = 0) = P(\hat{Y} = 1 \mid A = 1)$$

With

- $\hat{Y} \in \{0,1\}$, a binary classifier with predicted outcomes 0 (negative) or 1 (positive)
- $A \in \{0,1\}$, presence or absence of an attribute A, indicating group membership
- $P(\hat{Y} = 1 | A = 1)$, the probability of the algorithm making a positive prediction ($\hat{Y} = 1$) conditional on the membership to a particular group ($A = 1$)

As can be immediately seen, demographic parity is entirely independent of the true value (y). In other words, it does not consider if the predictions are *correct*, only if the predictions are equally *distributed*. In some cases, this can be useful, such as when we don't have direct access to the true value of the prediction, the target variable. In practice, this is often the case. In the case of credit scoring, for example, there is a significant time lapse between the time of decision making (grant or deny a loan) and the time the actual outcome (default or not default) is observed. In fraud detection, the algorithm is trained on historical data, but fraud is rare and fraud patterns evolve over time. Often, direct access to "true" fraud cases is limited, especially when it involves new types of fraud.

Demographic parity ensures that the outcomes are equally distributed, and this is often a desirable property. However, because it does not consider the *quality* of predictions, using demographic parity as a fairness measure can have drawbacks. For

example, demographic parity can be satisfied even if *all predictions about a particular subpopulation are wrong*. In the college admission case, the algorithm might perform worse on a particular subpopulation and thus label many unqualified candidates as qualified. Demographic parity would still be satisfied if the proportion of positive predictions remains the same across subpopulations. Other times, using demographic parity is simply inappropriate for a given task, such as when base rates significantly differ between groups, and those differences are relevant to the decision at hand.⁵ In a medical context, for example, demographic parity might not be the best or only fairness measure because the primary concern is the quality of prediction. Similarly, in credit risk assessment, it might be unreasonable to ignore certain relevant factors, such as financial behaviors and credit histories, that are crucial for assessing individual risk but that might be strongly correlated with protected attributes, such as age.

Confusion Matrix

As opposed to demographic parity, fairness measures that aim to achieve performance equality across subpopulations require us to examine the true value of the target variable.

To better understand the various measures of performance equality it is useful to consider what is known as a *confusion matrix* ([Figure 3.1](#)). Note that the confusion matrix is presented in a different order in Module 2. Both formats are acceptable. Please make sure to read the labels carefully before performing any calculations.

| | | Actual Values | |
|------------------|----------|----------------|----------------|
| | | Positive | Negative |
| Predicted Values | Positive | True Positive | False Positive |
| | Negative | False Negative | True Negative |

Figure 3.1: A confusion matrix helps illustrate the meanings of true positives, true negatives, false positives, and false negatives.

The confusion matrix illustrates how algorithmic predictions can be divided into four categories. In the case of fraud detection, for example, the confusion matrix labels transactions that were correctly identified as fraudulent (TP), transactions that were correctly identified as non-fraudulent (TN), transactions that were erroneously identified as fraudulent (FP), and transactions that were erroneously identified as non-fraudulent (FN).

The first important concept we can illustrate is algorithmic accuracy. Accuracy denotes the fraction of correct predictions, or:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

It is tempting to consider accuracy a good indicator for the quality of a classification algorithm, but this isn't always the case. Consider the following example: There are 100 job applicants, 10 of whom are suitable, 90 of whom are not. Imagine now the algorithm correctly classifies the 90 applicants as unsuitable (TN), but also classifies the 10 suitable candidates as unsuitable (FN). In this case the algorithm would have an accuracy of 90% even though it would only provide the user with a single output: unsuitable. Especially in cases of unbalanced data, accuracy is not a good measure.

2.2.2 Group Fairness (continued)

Predictive Rate Parity

We saw earlier that demographic parity often is insufficient to ensure fair results because it does not consider the quality of predictions. In some contexts, such as medical diagnostics or credit lending, the cost of false positives is high, and thus accuracy of the predictions for the positive or negative class is important. In other words, we want to maximize the positive predictive value (PPV) or precision:

$$PPV(precision) = \frac{TP}{TP + FP}$$

PPV is an important concept for algorithmic performance evaluation. It denotes the proportion of correctly predicted positive outcomes.⁶ In the case of fraud detection, for example, it indicates how many transactions that were marked as fraudulent were indeed fraudulent. In the case of credit scoring, it indicates the proportion of applicants predicted to default that actually default. A high PPV in this case means that the algorithm can effectively identify individuals who are likely to default, whereas a low PPV indicates that there are many individuals who would have repaid their loans but were erroneously labeled as high-risk.

We can now define another fairness measure, predictive rate parity, which is satisfied when PPV is equal across subpopulations. In other words, predictive rate parity demands that among those predicted as positive by the model, the proportion of individuals correctly identified as positive is the same across groups.

Its formal definition for two groups is:

$$P(Y = 1 | \hat{Y} = 1, A = 0) = P(Y = 1 | \hat{Y} = 1, A = 1)$$

Where Y is the target variable (e.g., whether an individual will default on a loan), \hat{Y} the predictor (e.g., whether the individual is predicted to default), and A a protected characteristic (e.g., whether the individual belongs to a particular social group). This definition can easily be extended to multiple groups.

Predictive rate parity has an intuitive appeal as a fairness measure: We often want to avoid stark differences in false positives (or negatives) across subpopulations. For example, predictive rate parity ensures that the proportion of accepted college applicants who succeed at college is the same across groups. This would prevent a situation where the academic potential of applicants is consistently overestimated in one group and underestimated in another.

A downside of predictive rate parity is that it does not account for differences in base rates across groups. If base rates differ significantly, then achieving predictive rate parity becomes extremely challenging and can have adverse effects on algorithmic performance.

Equal Opportunity

In some cases, it is important to ensure that positive cases are recognized as such, as in the case of medical diagnostics, where failing to diagnose a disease can have life-changing consequences. A measure that fits this task is sensitivity, also known as true positive rate or recall. It quantifies the proportion of true positives that are correctly identified or predicted as such.

The formula for sensitivity is:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Whereas PPV measured the proportion of correctly predicted positives to all *positive predictions*, sensitivity measures the proportion of correctly predicted positives to all *actual positives*.⁷ As a performance measure, sensitivity is useful in situations where it is important not to miss out on a positive. Maximizing sensitivity requires us to minimize false negatives.

If sensitivity is the same across groups, we have achieved equal opportunity, which is the last important fairness measure we discuss here. Equal opportunity ensures that individuals of each group have an equal chance of being correctly identified as positive. In medical diagnostics, for example, it ensures that patients who belong to different demographic groups and who have a given disease are equally likely to be diagnosed correctly.

The mathematical formula for equal opportunity for two groups is:

$$P(\hat{Y} = 1 \mid A = 0, Y = 1) = P(\hat{Y} = 1 \mid A = 1, Y = 1)$$

This definition can easily be extended to multiple groups. In the case of equal opportunity, the equality constraint is only applied to the true positive rate, so that each group has the same opportunity of being granted a positive outcome. A more-restrictive version of equal opportunity is equalized odds, where we not only require equality of sensitivity across groups, but also equality of specificity (see footnote).

Equal opportunity is a useful fairness measure that ensures, say, that the same proportion of qualified candidates is admitted to college. However, just as with

predictive rate parity, equal opportunity requires access to the true value of the target and can be difficult to achieve when there are stark differences in base rates across groups. In those cases, it is difficult to achieve fairness without compromising on other aspects of the model's performance.

Impossibility and Trade-offs

We saw above that different notions of fairness highlight different requirements for fair decision making. Each of them has some drawbacks, so could we not simply combine them and demand that *all* fairness measures hold to ensure the decision is fair? Unfortunately, a mathematical theorem demonstrates that doing so is impossible. Specifically, when base rates between populations are different, which is almost always the case, then it is impossible to satisfy demographic parity, predictive rate parity, and equal opportunity simultaneously.⁸ The very fact that we cannot achieve fairness on all dimensions demonstrates how important it is to deliberate on the kind of group fairness one wants to achieve when designing an algorithm, because there are always trade-offs to consider.

These trade-offs not only apply to fairness measures, but to algorithmic performance optimization more generally. For example, a good fraud-detection algorithm ideally would have *high* sensitivity and *low* false-positive rate.⁹ That is, we want the system to be effective at correctly identifying fraudulent transactions while minimizing the likelihood of mistakenly flagging a legitimate transaction as fraudulent. To maximize sensitivity, then, we could simply block all transactions (predict them as positive) and

achieve maximal sensitivity: Because there are no negatives, there are - by definition - no *false* negatives and therefore *Sensitivity* = 1. However, the false-positive rate would shoot up in this case. On the other hand, to minimize the false-positive rate, we might go to the other extreme and label all transactions as non-fraudulent. Again, because there would be no positives, there would also not be any *false* positives and therefore the false-positive rate drops to 0. However, because there are now many false negatives, sensitivity would massively decrease. The example demonstrates that balancing different performance measures is by no means a trivial task. Instead, it forces us to consider the trade-offs that are involved and pick the balance appropriate to the task at hand.

Table 1: Group Fairness Measures and the Questions they Answer

| | |
|--|--|
| Accuracy | "Of all cases, how many did we correctly identify as either positive or negative?" |
| Positive predictive value/Precision | "Of all positively predicted cases, how many were actually positive?" |
| Sensitivity | "Of all actual positive cases, how many did we correctly identify as such?" |
| Demographic parity | "Is the likelihood of a positive prediction the same across groups?" |
| Predictive rate parity | "Is the likelihood of a true positive prediction the same across groups?" |

| | |
|--------------------------|---|
| Equal opportunity | "Is the likelihood of being positive when predicted to be positive the same across groups?" |
|--------------------------|---|

⁵ Base rate refers to the general prevalence of something within a population. For example, a disease or medical condition may have different levels of prevalence amongst men and women, old and young, etc.

⁶ The related notion of negative predicted value (NPV) is calculated by focusing on negative predictions instead of positive ones.

⁷ A related notion is *specificity*, which captures the proportion of actual *negatives* that are identified as such. The formula is

$$\text{Specificity} = \frac{TN}{TN+FP}.$$

⁸ See also Kleinberg, Jon, Sendhil Mullainathan, and Manish Raghavan. "Inherent trade-offs in the fair determination of risk scores." arXiv preprint arXiv:1609.05807 (2016). and Chouldechova, Alexandra. "Fair prediction with disparate impact: A study of bias in recidivism prediction instruments." Big data 5.2 (2017): 153-163.

⁹ As the name suggests, the false positive rate is the proportion of false positives out of all actual negative cases:

$$FPR = \frac{FP}{FP+TN}.$$

2.3 Sources of Unfairness

In the previous section, we encountered different notions of fairness that allow us to examine the fairness properties of algorithmic outputs. We did not, however, address *why* algorithms might lead to unfair outcomes in the first place. Because there are numerous sources of unfairness and it is not always straightforward to identify them, we approach the topic in a more-or-less chronological order that leads us through the AI development process. At each step, we highlight different issues that might arise. In particular, we discuss algorithmic bias arising in problem specification and feature selection, data collection and data composition, model development, and model deployment.

2.3.1 Problem Specification and Feature Selection

Technology carries instrumental value: It helps us solve a particular problem by carrying out a particular task. To build useful technology, however, we first need to specify what this problem is and how it could be solved. During problem specification, purpose and design of the algorithm are explicitly stated. Unsurprisingly, how a problem is specified can have large effects on both algorithmic performance and algorithmic fairness. Consider the following example:

Hiring algorithm: Your company wants to improve and speed up the recruitment process by using a hiring algorithm. You choose to try HireMe, an algorithm that scans applicant CVs and outputs a shortlist of the best applicants. HireMe is trained on data provided by current employees of your firm.

For HireMe to work as intended, we first need to specify what is a good applicant. Doing so requires expertise to determine what is important: An editor is better placed to recognize a promising author than a scientist, whereas the scientist is better placed to determine which PhD student should receive a scholarship. Problem specification always requires us to have some degree of familiarity with the domain and needs in question. In other words, domain expertise is a necessary (though not sufficient) condition for good problem specification.¹⁰ In the case of HireMe, let us assume your company is looking for someone who is productive, has a strong work ethic, is a good team player, and is likely to remain in the company for a substantial time.

As any hiring manager will know, the choice of job criteria will affect who will apply for the job and who will get hired. Within the field of econometrics, this phenomenon is also called endogeneity. Technically speaking, endogeneity refers to situations in which the error term in a model is correlated with the predictor variable. In simpler terms, it refers to situations in which there are hidden or overlooked correlations that are not captured by the model in question. In our hiring case, there are selection effects in the applicant pool, because the job specification will affect who ends up applying for a particular position.¹¹

Once we have decided on the relevant criteria, they need to be translated into measurable and quantifiable features. In some cases, this translation is straightforward. Most of the time, however, it requires us to use proxies. These proxies we judge to be strongly correlated with the criterion in question. For example, we cannot easily measure productivity, but we can measure how many sales a given sales agent has made within one year of hiring. Using proxies always risks the introduction of bias as they reduce complex concepts to a much less-complex, measurable construct. In the case of HireMe, using total sales as an indicator for productivity might miss that female employees are more likely to work part-time, and thus have a lower total sales rate. Finally, some things are harder to quantify than others. Being a team player, for example, can be harder to quantify than productivity. Yet, simply skipping this feature may lead to predictions that can hurt candidates if, for example, teamwork was an area where women generally outperformed men.

Careful! It is tempting to think that because supervised ML algorithms learn on past data, we don't need to be explicit about the specific criteria we are looking for in a job applicant (such as productivity or teamwork). We could simply provide our algorithm with previous applicants' CVs and a simple binary target variable that indicates whether they were hired for a given role. This is another way of specifying the problem. Now, a "good" candidate is by definition a candidate that resembles previous hires. This method bears its own shortcomings, as discussed in the next section. A prominent example of the difficulties associated with this method is Amazon's attempt to create an unbiased recruiting tool. In the end, Amazon gave up the project because it could not prevent its algorithm from systematically downgrading CVs from women applicants, even if they were identical to those of their male counterparts in all relevant aspects.¹²

Finally, when choosing which features to incorporate into your algorithm, sometimes less is more. A well-known criminal risk-assessment algorithm takes as input 137 distinct features to make predictions about the likelihood of recidivism. Researchers showed that nearly equivalent predictions could be made by only using two features.¹³

Fairness Through Unawareness?

Intuitively, we don't want protected characteristics to influence algorithmic decision making in most contexts. For example, most people will consider it inadmissible for an algorithm to explicitly take into account the ethnicity of a criminal defendant when determining whether the defendant should receive bail. The demand that protected characteristics play no role in the decision-making process is equally embedded in most legal doctrines.¹⁴ As a result, these frameworks currently prohibit the explicit use

of protected characteristics by an algorithm. Does removing the protected characteristic from the input features solve the problem of discrimination? To address this question, consider the following example:

FTA prediction: JusticeGuard is a fictional algorithm that helps judges decide whether to grant bail. It outputs the likelihood that a defendant awaiting trial will fail to appear (FTA) in court.

Imagine now that there is some correlation between belonging to a protected group and FTA. Should we use group membership to make predictions? Most legal doctrines agree that protected characteristics should not be used as inputs. In fact, making decisions that are explicitly based on the use of protected characteristics can lead to charges of direct discrimination (EU) or disparate treatment (US). If we do not use a protected characteristic as input, does that mean our algorithm becomes unbiased?

To illustrate the effect of removing protected characteristics from inputs, let us consider the following example.¹⁵ Imagine we estimate the following, linear relationship between FTA (Y), having a prior conviction (X^{prior}) and being a member of a protected group (X^{prot}):

$$Y = \beta_0 + \beta_1 \cdot X^{prior} + \beta_2 \cdot X^{prot} + \epsilon_0 \quad (1)$$

where β_0 is a constant and ϵ_0 an error term.¹⁶ For illustration purposes, we assume that there is a correlation between belonging to the protected group and FTA. We also assume that there is a correlation between belonging to the protected group and having a prior offense. [Figure 3.2](#) illustrates a concrete example of what this relationship could look like.

If we consider all our variables to take on binary values 0 or 1 and run an OLS regression, we obtain the following statistical relationship for our predictions:

$$\hat{Y} = 0.034 + 0.543 \cdot X^{prior} + 0.332 \cdot X^{prot} + \epsilon_0 \quad (2)$$

This means that our algorithm predicts a 54% increase in risk if an individual has committed a prior offense, and a 33% increase if they belong to a protected group! As already mentioned, we don't want our algorithm to make predictions on the basis of group membership. What happens when we simply take out the protected characteristic?

In this case, Equation 2 would simplify to:

$$Y = \gamma_0 + \gamma_1 \cdot X^{prior} + \xi_0 \quad (3)$$

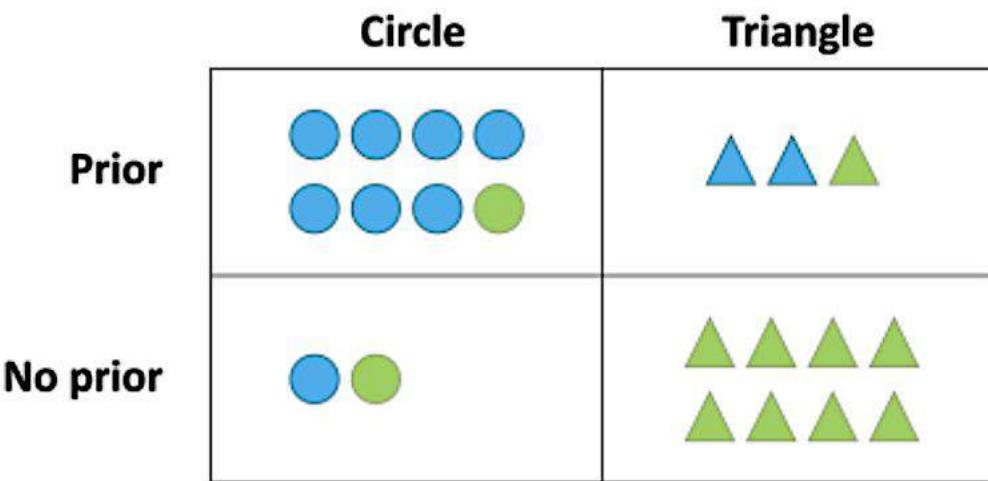


Figure 3.2: Illustration of the relationship between appearing in court (green) and failure to appear in court (blue). Circles represent the protected group with triangles representing the remainder. Each shape (circle or triangle) represents 5 individuals.

Notably, γ_0 is not identical to β_1 . We already mentioned that X^{prior} and X^{prot} are correlated. This means that we can express the protected variable in terms of the correlated variable X^{prior}

$$X^{prot} = \alpha_0 + \alpha_{corr} \cdot X^{prior} + v_0, \quad (4)$$

where α_{corr} quantifies the correlation between the protected variable and having a prior. When we plug equation 4 into our original statistical estimate equation 1, we see that the coefficients have changed:

$$Y = (\beta_0 + \beta_1 \alpha_{corr}) + (\beta_1 + \beta_2 \alpha_{corr}) \cdot X^{prior} \quad (5)$$

The coefficient of X^{prior} is contaminated (in the statistical sense) due to the correlation between X^{prior} and X^{prot} ! Plugging in the numbers confirms this. For our prediction we get:

$$\hat{Y} = 0.10 + 0.72 \cdot X^{prior} + \xi_0, \quad (6)$$

Instead of the previous 54% we calculated, having a prior now increases the predicted probability of FTA by 72%! In other words, now that we have removed the protected variable from our model, the model counts having a prior more heavily for risk predictions.

What this shows us is that simply removing the protected variable from our input features does not mean that the protected variable won't play any role anymore. Instead, correlated variables are weighted more strongly in the predictions.

Could we then just simply remove all correlated variables? We could, but much of the time, these variables are highly relevant for prediction, and the algorithm would perform much worse if we removed them.

Another solution could be to include the protected variable when we model our relationship, as we did in equation 3, but then not use it in the prediction step (equation 2), and instead substitute it with an average. This would only slightly decrease accuracy but ensure that (a) people who are alike in all relevant respects but have different protected characteristics are treated the same, and (b) correlated variables are not contaminated by the protected variables. However, as previously mentioned, care needs to be taken to ensure legality.

- ¹⁰ This also means that some care needs to be taken when algorithms are designed and developed by third parties who are not familiar with the particular domain of application.
- ¹¹ For a more detailed discussion of endogeneity, see e.g. Wooldridge (2009). Introductory Econometrics: A Modern Approach (Fourth ed.). Australia: South-Western. p. 88
- ¹² <https://www.reuters.com/article/idUSKCN1MK0AG>
- ¹³ Dressel J, Farid H. The accuracy, fairness, and limits of predicting recidivism. *Sci Adv.* 2018 Jan 17;4(1)
- ¹⁴ This clearly does not hold for *all* decisions. Especially in medical contexts, considering protected characteristics is not only permissible, but often necessary.
- ¹⁵ The following discussion is based on results by Yang, Crystal S., and Will Dobbie. "Equal protection under algorithms: A new statistical and legal framework." *Mich. L. Rev.* 119 (2020): 291.
- ¹⁶ We set up the problem so that the output is binary. Normally, the natural model choice would have been a logistic regression. We chose to use a linear model, however, because it provides a more intuitive and clear illustration of the problem. Furthermore, in practice many risk assessment algorithms use linear models.

2.3.2 Data Collection and Data Composition

Supervised and unsupervised learning rely on large amounts of data. These algorithms make evidence-based decisions, and as with any evidence-based decision, the decision can only be as good as the evidence it is based on.

Historical Bias and Feedback Loops

Supervised learning algorithms rely on historical data. Even if this data is representative, it can reflect historical biases. For example, if a hiring algorithm is trained on past hiring data, and women were historically less likely to be selected for certain roles than men, this algorithm will not only repeat the same patterns but exacerbate them. This is an example of a feedback loop.

Data Collection

One of the biggest challenges when developing an AI algorithm is obtaining a training set that adequately reflects the actual statistical distribution of features within the target population. One of the most common forms of data bias is representation bias. Here, the collected sample is not representative of the target population and can result in problematic model specifications. This has been an issue in many settings, including the training of facial-recognition algorithms, medical testing, and others where the sample population was dominated by one ethnic group (typically White/European). In those cases, a supervised AI algorithm will perform better on the dominant group and perform less well on underrepresented groups. Performance can be particularly low for groups that are at the intersection of different, underrepresented subgroups.¹⁷

Another form of bias that can occur during the process of data collection is measurement bias, when measurement methods are not consistent across the sample. If a doctor spends more time meeting with and evaluating wealthy patients, resulting data on the likelihood of early-stage disease discovery may be biased in favor of that group.

Data Composition

Even if measurement and sampling methods have been carefully monitored, algorithms can end up biased as a result of data composition. Given the statistical nature of both supervised and unsupervised learning algorithms, these algorithms

deliver the *statistically* ideal output. For example, consider a medical training set of patients, 3% of whom are pregnant women who would require different treatment. It is likely that the algorithm would significantly underperform for this group because there are many fewer data points.

As a result, it is often important that datasets are balanced. This means that relevant minority subgroups occupy a similar proportion of the space as majority groups. This ensures that the algorithm recognizes the minority groups as statistically relevant.

Notably, this is different from having proportional representation. Proportional representation would mean that, if 10% of the population are triangles and 90% are circles, then the dataset would equally contain 10% triangles and 90% circles.

Instead, if we want the dataset to be balanced, we would increase the proportion of triangles in the dataset. As becomes clear, balancing a dataset can mean that the dataset becomes less representative of the actual population. A balanced dataset for a hiring algorithm, for example, could mean that male and female applicants are represented equally (50%) within the dataset, even if only 20% of job applicants historically were female.

Bootstrapping is a process by which datapoints are “resampled” from an existing dataset and added back into the dataset. There are three methods through which bootstrap resampling can be used to balance a dataset: over-sampling datapoints from the minority group; under-sampling datapoints from the majority group; or a combination of these.

It is also possible to create (generate) synthetic data points or to remove existing datapoints to adjust data composition as well. Each of these methods has its own challenges. It is, for example, often challenging to create new, synthetic data without compromising an algorithm's ability to generalize. Taking out datapoints, on the other hand, only works if datasets are large enough.

Balancing datasets can be challenging, especially in domains where the relevant cases are rare. In fraud detection, for example, it might be the case that less than 1% of transactions are fraudulent. In addition to the strategies mentioned above, the issue of imbalanced data can also be addressed through model design. Ensemble methods, for example, use multiple models. Here, one could ensure that the entirety of the minority class data from the training set is used to train each model, whereas the majority class data is randomly sampled in each case. Alternatively, one could design a cost function that has a higher penalty for misclassification of minority class cases than for misclassification of majority class cases. There are several other methods available that combine both algorithmic and resampling methods.¹⁸

Trying to decide how to balance data presupposes that we know *with respect to what* we ought to balance the data. But which are the salient social groups to which we ought to pay attention? One obvious answer is that we need to pay particular attention to groups of people who share a protected characteristic. However, there are many other ways to carve up the space. In the medical context, for example, it can make a difference whether a patient is pregnant, overweight, has diabetes, is under medication, and so on. There is no one-fits-all solution to the problem of data

composition, and it will be informed by legal considerations as well as the ethical and practical concerns of those involved with the algorithm's construction.

¹⁷ Buolamwini, Joy, and Timnit Gebru. "Gender shades: Intersectional accuracy disparities in commercial gender classification." *Conference on fairness, accountability and transparency*. PMLR, 2018.

¹⁸ See for example <https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html> for a good overview over different ways to handle imbalanced datasets.

2.3.3 Model Development

Some of the preprocessing techniques can introduce bias. For example, a popular preprocessing technique for natural language processing (NLP) algorithms is to use word embeddings. These embeddings transform words into high-dimensional vectors. These vectors capture the meaning of a word through the way they relate to other word-vectors. For example, you can query these word embeddings to give you relationships: man is to woman as king is to ...? Queen! It has been shown, however, that these word embeddings may contain stereotypes. If you instead query: man is to computer programmer as woman is to...? The computer may reply, "homemaker."¹⁹ As a result, NLP algorithms that are based on word embeddings that contain stereotypes may incorporate them.

The choice of objective function introduces value judgments into the development of algorithms. What the algorithm optimizes for can make an ethically significant difference. Is the algorithm trying to minimize prediction errors (focus on performance), or does it focus on how errors are distributed (focus on error

distribution)? The choice will heavily depend on context, as the following example shows.

Cancer detection: Consider two cancer-detection algorithms. Algorithm A takes as input an image of a skin discoloration or mole and outputs a risk score that predicts whether the mole is cancerous. Algorithm B takes as input a brain scan and outputs a diagnosis as to whether the patient has brain cancer.

For Algorithm A, it seems reasonable to err on the side of caution and minimize the number of times cancer goes undetected and moles are erroneously characterized as harmless. In the case of Algorithm B, on the other hand, treatment can be highly invasive and might require major brain surgery. In this case, there are additional considerations, such as risks that are introduced through surgery.

Another source of bias can enter in the testing stage. It is currently common practice to test algorithms against publicly available benchmark datasets. These datasets themselves may not be representative of the intended use population and are not necessarily a good indicator of a model's quality.

¹⁹ <https://www.microsoft.com/en-us/research/publication/quantifying-reducing-stereotypes-word-embeddings>

2.3.4 Model Deployment

There are numerous sources of harm that might arise during the deployment of algorithms. For example, algorithms that are used within different contexts than what

they were designed for can lead to a loss of performance or other challenges. This applies both to different contexts as well as to the use of historical data that might be outdated. To give just one example: In the context of law enforcement, an algorithm to predict recidivism may not be adequate when trying to determine sentence length.

Related, if the values of the user do not align with the values that were built into the algorithm, this might lead to harm. Going back to our example of a “good employee”: often, companies will not develop their algorithms in-house, but instead use algorithms developed by third parties. If their understanding of a good employee differs from the developers’ understanding, the algorithm will not deliver the desired results.

Ensuring safe and fair algorithms will require human oversight of the data-collection process, the development of the algorithm, and its deployment. As algorithms are deployed in more consequential domains, regular auditing will become important.

3.0 Explainability, Interpretability, and Transparency

As AI systems are increasingly deployed to perform tasks that are of consequence to people’s lives, the twin concepts of explainability and interpretability emerge as a central concern when evaluating risks associated with AI systems. Explainability refers to the capacity to explain in understandable terms how an AI system makes decisions or predictions, often after the fact (hence referred to as “ex-post

explainability” in the literature). Interpretability, on the other hand, refers to the degree to which a human can comprehend and predict the model’s behavior with built-in mechanisms to understand inherently how the inputs in the model affect the outputs (hence referred to as “inherently interpretable models” in the literature). For risk professionals, understanding these concepts is essential for ensuring transparency, accountability, and trust in AI systems. In environments in which AI-aided decision making is of significant consequence, the ability to both explain and interpret algorithmic decision making becomes a cornerstone for risk mitigation as it helps us align AI systems to ethical standards, organizational values, and regulatory requirements.

3.1 The Black-Box Problem

One of the most notorious challenges in the field of AI is the so-called black-box problem. At the core of this challenge is the difficulty of understanding how certain algorithms make decisions. Here, there are vast differences among models. Some, such as decision-trees, can be interpreted so the algorithmic reasoning becomes clear to humans, but others are much more opaque. For example, as neural networks become more intricate with numerous layers and large amounts of data, understanding the decision-making process becomes an arduous if not impossible task.

In some domains of application, the black-box problem can pose significant challenges. When decisions made by AI systems have substantial impacts on individuals or groups of individuals, the inability to explain algorithmic decisions can lead to accountability gaps or the erosion of trust among users and stakeholders. If a healthcare algorithm makes a recommendation for medical treatment, for example, doctors and patients alike need to be able to understand the basis of this recommendation.

The black-box challenge can be divided into a technical and a philosophical part. The philosophical part consists of determining what counts as a good explanation. When we ask people for explanations, we often intuitively know whether an explanation is good or bad. However, finding a satisfying definition of the concept of explanation is an entirely different story. The technical part consists of finding ways to obtain good explanations.

3.2 Opaqueness

Very often, the issue in question is not so much that an algorithm itself is opaque, but that modes of deployment used to implement the algorithm are opaque. To see various forms of opaqueness, consider the following example:

Recommendation algorithm: An algorithm extracts personal data from social media posts, clicks, buys, and visits to websites to extrapolate a profile associated with a given

individual. Based on this profile, the algorithm makes recommendations as to what advertisement is shown to the individual.

A first form of opaqueness is secrecy, in which an individual subject to algorithmic profiling is not aware of having been subject to algorithmic decision making in the first place. An even more radical form of ignorance is when individuals are unaware of the very existence of the algorithm in question. For example, many people are not aware of various types of consumer scores, such as the Individual Health Risk Score.²⁰

A second form of opaqueness is confidentiality. Confidentiality refers to the opaqueness of the algorithmic decision-making process itself. That is, affected parties may be aware that they are subject to algorithmic decision making, but they do not have access to the workings of the algorithms or to the reasons for why a certain outcome was reached. This has been a main concern in the recent discourse on the use of algorithmic decision making, but as Selbst and Barocas (2018) point out, this type of opacity is not unique to algorithmic decision making: "It is an objection to being subject to a decision where the basis of decision-making remains secret, which is a situation that can easily occur without algorithms or machine learning."²¹

There are sometimes valid reasons for why the inner workings of an algorithm are not disclosed. Security is one of them: If spam filter service providers were to make their code public, circumventing these filters would become much easier. Releasing too much detail about an algorithm therefore isn't always desirable, as it might increase risks of fraud, scams, or cyber-attacks. A second reason for not making the workings of algorithms public regards proprietary interests and trade secrets. Keeping their algorithm secret helps companies retain their competitive advantages. For companies

using third-party algorithms, confidentiality can become a big challenge if it prevents them from assessing the impacts of the algorithm on their own customer base. In fact, companies are recommended to ensure some access to the algorithm in question to enable them to perform their own risk assessments and provide accountability.

Finally, a third hurdle is that interrogating an algorithm often requires specialized knowledge. Even if subjects or users have access to source codes and training data, they may not be able to make sense of them, unless they have received specialized training. This type of opacity results from a lack of preexisting knowledge.

²⁰ Dixon and Gellman. The Scoring of America: How Secret Consumer Scores Threaten Your Privacy and Your Future 84 (2014).

²¹ Selbst, Andrew D., and Solon Barocas. "The intuitive appeal of explainable machines." *Fordham L. Rev.* 87 (2018): 1085, p.1092.

3.3 Explainable AI (XAI)

Explainable AI aims to make AI systems more understandable. AI systems are becoming increasingly complex and therefore increasingly inscrutable for human interlocutors. Some techniques to make algorithmic decision making more explainable include the following:

Feature importance scores are a prominent technique used in XAI. It ranks the importance of input features for outcomes. For example, consider the case of a credit-scoring AI system. Feature importance scores can reveal which factors (e.g., income, credit history, etc.) have the most influence on the credit score. Feature importance scores can not only help to scrutinize how a model makes decisions, but can help

identify potential biases or errors in the model. Other techniques similarly analyze the impact of input features on the model's output. Shapley values, for example, calculate the contribution of each input feature to the model's output by calculating the marginal effect of including each feature and doing so across all possible feature combinations.²² LUCID (Locating Unfairness through Canonical Inverse Design), on the other hand, works backward from a particular algorithmic output (e.g., "loan granted") to create a distribution over the input space conditional on the particular output, thereby revealing a model's internal logic.²³

Surrogate models are what their name suggests: simpler, more interpretable models that approximate the behavior of a complex AI system. LIME (local interpretable model-agnostic explanations), for example, approximates a complex model's prediction locally with an interpretable model (e.g., linear regression or decision-tree).²⁴

There are several challenges associated with XAI. The first is that an oversimplification of a complex decision-making process can distort our understanding of the decision to the point of misinterpretation. Second, several XAI techniques are computationally intensive, which makes developing and deploying them more costly. Finally, the above techniques are all ex-post techniques, which means that they are used to understand the models after they have been trained. Wouldn't it be better to avoid building complex models in the first place and stick to interpretable models? The answer is yes and no: Often (though not always) complex models achieve higher accuracy than simpler, explainable models. In these cases, we are necessarily faced

with an accuracy-interpretability trade-off that requires weighing both factors in the context of the algorithm's application.

²² Owen, A. and Prieur, C. On shapley value for measuring importance of dependent inputs. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):986–1002, 2017

²³ Algabe, Andres, et al. "LUCID-GAN: Conditional Generative Models to Locate Unfairness." *World Conference on Explainable Artificial Intelligence*. Cham: Springer Nature Switzerland, 2023.

²⁴ Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, 1135–1144.

4.0 Autonomy and Manipulation

AI systems increasingly inform or even dictate choices in various consequential domains, such as medical diagnostics, policing, or finance. This raises questions about the impact AI systems have and will have on the human ability to make meaningful choices. Human autonomy is the ability to hold beliefs, make, and execute decisions of our own.

4.1 Autonomy

Understanding how AI systems might affect autonomy requires us to understand what autonomy is. There are (at least) two dimensions to the way we use and value autonomy.²⁵ The first dimension refers to our ability to have values, hold beliefs, and

make decisions that are in some important sense our own and not the product of distorting external factors, such as manipulation. The second dimension refers to our ability to execute these decisions and requires us to have the freedom and the opportunity to do so.

A first misconception to address is that by delegating tasks to AI systems, we automatically lose our autonomy. This certainly isn't the case by default: We routinely outsource tasks to other people and entities, but we are not less autonomous by doing so. As long as users and subjects have adequate control over this outsourcing, as long as there is meaningful human oversight, we remain autonomous in these respects.²⁶

²⁵ Prunkl, Carina. "Human autonomy in the age of artificial intelligence." *Nature Machine Intelligence* 4.2 (2022): 99-101.

²⁶ Prunkl, Carina. "Is there a trade-off between human autonomy and the 'autonomy' of AI systems?" *Conference on Philosophy and Theory of Artificial Intelligence*. Cham: Springer International Publishing, 2021.

4.2 Manipulation

AI systems shape our online experience. Search algorithms determine what search engine entries show up first, and recommendation algorithms determine what advertisement we see, or which social media posts show up on our timelines. Simultaneously, these algorithms draw on large amounts of information about us and our past (online) behavior, allowing them to make predictions about future actions and influence our behavior.

A wake-up call for many was the now infamous Facebook Cambridge Analytica scandal. Cambridge Analytica collected data from numerous Facebook users who had completed personality tests and extracted information about (alleged) correlations. Based on these correlations, Cambridge Analytica claimed to be able to infer a Facebook user's personality profile from their online behavior, allowing them in turn to send psychologically tailored political advertising to influence voter behavior. While it is questionable whether this strategy had any significant effects, the incident powerfully demonstrated how big data could be used by political interests to manipulate voter behavior.

Another, more clear-cut example of online manipulation is the Facebook emotional contagion experiment. In this experiment, researchers wanted to test whether emotional contagion—an empathetic reaction to another person's emotions—could take place online. Without the knowledge of the users, the researchers changed users' timelines to display predominantly positive or negative posts of their peers. As a result, the users' posts themselves became slightly more positive or negative, respectively. The experiment caused a public outcry because users did not consent to participate in it. In any case, it demonstrates the power of recommendation algorithms to shape our experience and, in this case, to manipulate us.

The above examples not only powerfully demonstrate the need for robust data privacy and security measures, but also underscore the potential of AI to manipulate public opinion. The potential for misuse of AI systems must be balanced against the benefits of AI in personalization and targeting. To do that, adequate human oversight

throughout the development process as well as some degree of regulatory oversight may be necessary. Current efforts to prevent manipulation through AI systems are mostly located in the policy domain, where regulators strive to address the issue. The European Union's AI Act, for example, addresses the risk of manipulation by imposing transparency obligations on certain systems and by prohibiting systems that use subliminal techniques on people. However, the ambiguous notion of manipulation and the difficulty of operationalizing it into a quantifiable, measurable quantity continue to hamper progress in developing and enforcing mitigation techniques.

5.0 Safety and Well-Being

As AI continues to integrate into various aspects of our lives, understanding the risks it poses to safety and well-being, especially in human-AI interactions, becomes important for risk professionals. This section delves into some of these risks, focusing on AI in critical systems, the phenomenon of automation bias and overreliance, and the dynamics of human-machine interactions, specifically in mental health support systems.

Critical systems, such as transportation, healthcare, and public safety, increasingly rely on AI for efficiency and improved decision-making. However, this integration comes with significant risks. One of the most prominent examples of a critical system failure was the crash of an Uber driverless vehicle that fatally injured a pedestrian in

Tempe, Arizona. The crash was the result of various shortcomings: (a) the vehicle's systems failed to recognize the pedestrian in time, (b) some of the systems' safety mechanisms were deactivated, and (c) the human safety driver was distracted and did not react quickly enough, a result of automation bias. The incident underscores the risk of over-dependence on (often immature) AI systems in critical scenarios, but also highlights the need for robust testing and validation processes. In these cases, there needs to be comprehensive testing protocols that assess the systems' performances in real-world scenarios.

Additionally, automation bias is a real problem. It refers to the tendency of humans to over-rely on automated systems, leading to complacency and reduced vigilance. This in turn can lead to errors of oversight, which is especially problematic in critical situations. Another risk of automation bias is skill atrophy. Prolonged reliance on automation can lead to a decay in essential skills, as operators move from being active decision makers to becoming passive observers. There are some ways we can mitigate automation bias, however. The first is to inform users and operators of the risks of automation bias, and to ensure they understand the limitations of AI. The second step is to implement design mechanisms that require periodic human input and verification. This keeps operators and users engaged and alert. For example, Uber could have helped the driver keep her eyes on the road by using eye tracking technology in combination with alerts to ensure meaningful human control of the AI technology.

A related topic is that of human-machine interactions. To understand the risks of integrating AI into our everyday lives, we need a thorough understanding of the dynamics of human-machine interactions. Consider as an example AI-driven mental health support systems, such as chatbots or virtual therapists.²⁷ Although they offer patients accessibility and anonymity, we need to ask ourselves about the adequacy of these systems in handling complex emotional and psychological issues. It has been shown that AI systems can help patients with seemingly empathetic responses to their worries, better than the average human. However, if patients are told that the responses were AI-generated, positive effects typically dwindle. This is because AI systems have no empathy, which is a crucial part of mental health support.

There is a golden rule for AI development, which is: Just because we can does not mean we should. This rule serves as a reminder that we need to reflect carefully on which issues lend themselves to automation and which issues should stay in human hands until we have found ways to overcome existing inadequacies or challenges. A further golden rule for AI design is to involve diverse stakeholders, which include domain experts, the wider public, ethicists, and policy makers for AI applications in critical situations. This ideally ensures that the technology is ethical, safe, and appropriate. Finally, AI in critical systems requires both developers and users to be transparent and accountable. There needs to be clear lines of responsibility in case of failures or adverse outcomes. Such failures, finally, can be avoided by ensuring AI systems are regularly audited and updated.

²⁷ D'Alfonso, Simon. "AI in mental health." *Current opinion in psychology* 36 (2020): 112-117. Nie, Jingping, et al. "Conversational ai therapist for daily function screening in home environments." *Proceedings of the 1st ACM International Workshop on Intelligent Acoustic Systems and Applications*. 2022.

6.0 Reputational Risks

As companies begin to deploy more and more AI systems, they simultaneously expose themselves to novel risks and controversies that are associated with their use. When the use or the output of algorithms goes against prevalent norms and values, companies face reputational risks. We already discussed the Cambridge Analytica scandal in the context of manipulation, but what we haven't mentioned so far was the reputational damage Facebook faced in the wake of the scandal, both in brand value and public trust.

6.1 Causes for AI-related Reputational Damage

It has been shown that the three main causes for AI-related reputational damage are (a) privacy breaches, (b) algorithmic bias, and (c) lack of explainability.²⁸ As mentioned previously, the Cambridge Analytica scandal serves as a cautionary tale about mishandling user data. Incidents involving algorithmic bias and unfair algorithms can equally have large repercussions for those responsible. In 2021, for example, it became public that a fraud-detection algorithm deployed by the Dutch

government to detect fraudulent child benefits claims had falsely denied and reclaimed child benefits of thousands of Dutch citizens with dire consequences. In the wake of this scandal, the Dutch prime minister resigned. Finally, lack of transparency and explainability can equally lead to reputational losses. This relates to the phenomenon that customers are often not informed about their being subject to algorithmic decision making or about the reasons behind that decision making. Customers might reasonably be unhappy if they are automatically denied a loan but have no way to find out why.

²⁸ Holweg, Matthias, Rupert Younger, and Yuni Wen. "The reputational risks of AI." California Management Review Insights (2022).

6.2 Types of AI-related Criticism Companies Face

We can divide criticism of companies broadly into two categories: competence and value alignment. Competence is criticized when there is a perception that the company lacks the relevant expertise and capacities to ensure the algorithms they deploy are safe and fair, and the data they collect or use is stored safely. Technical failings of algorithms, unfairness, or lack of explainability often fall into this category. Value alignment is questioned when it becomes clear that the company has been fully aware of the risks and nevertheless went ahead. In the emotional contagion experiment discussed previously, the lead scientist was a researcher at Facebook. The company therefore was fully informed of the experiment, and the fact that users were not asked for consent was a violation of user trust and expectations. Value alignment can also be challenged if there is an obvious lack of due diligence to prevent incidents

that harm stakeholders. Often, the distinction between criticisms of competence and value alignment is not clear-cut. A data breach can be the result of incompetence, but it can also be the result of a company not investing enough resources and therefore not valuing the privacy of their stakeholders. In these cases, the perception may be both one of incompetence and of misaligned company values.

6.3 Management and Mitigation Strategies

Continuous monitoring and evaluation: The first and obvious strategy to prevent reputational loss from AI is to identify the risks and address them proactively. This can be achieved by **putting in place organizational governance mechanisms** that ensure risks are regularly monitored and evaluated. There are a number of measures that can be taken to address risks from algorithmic bias and explainability. They can be used to ensure a company lives up to the high standards of its customer base and the wider stakeholder community. They should include at least the following:

- Ensuring privacy and security: Data governance policies should comply with regulations such as GDPR and include secure data storage, handling, and processing practices, along with transparent data usage policies.
- Ensuring algorithmic fairness: Algorithms should be trained on diverse datasets and regularly audited for potential discriminatory patterns.

- Promoting transparency: Be as transparent as possible to your customer base. This includes ensuring that algorithms remain explainable using XAI techniques, but also ensuring that customers are aware of the use of algorithms. This builds both trust and accountability.

Demonstrate the will to change: Once an incident has happened, it is important to limit reputational damage. This is best achieved by being as transparent as possible about what has gone wrong and why. In cases of incompetence, companies can demonstrate that they are working to fix the issue. In cases of value misalignment, the problem may be rooted more deeply in governance structures or company culture.²⁹ In these cases, stakeholders need to be reassured that the underlying problems will be fixed via organizational means. Again, transparency is key to rebuilding trust.

Ethical AI frameworks: Ethical guidelines can help prevent incidents by providing both guidance and an overview of the risks. A word of warning: Many AI frameworks are too vague to be helpful in cases of ambiguity. Developing ethical AI frameworks in collaboration with experts may help ensure they provide the guidance needed to prevent and address risks from AI.

Stakeholder engagement and communication: Regularly engage with stakeholders, customers, employees, and regulators about AI initiatives and their impacts. Clear communication prevents misunderstandings and builds trust.

Risk analysis and crisis-management plan: Companies can perform a risk analysis, the basis of which can inform the development of a crisis-management plan specifically tailored to

AI-related incidents. This can include communication strategies as well as potential steps to rectify the issue.

²⁹ "Holweg, Matthias, Rupert Younger, and Yuni Wen. "The reputational risks of AI." California Management Review Insights (2022).

7.0 Existential Risks

Existential risks from AI relate to scenarios in which the advent of advanced AI could pose severe or even catastrophic threats to human existence. A central theme in discussions on existential risks from AI is the notion of superintelligence, which refers to an AI system that operates beyond human-level intelligence. Such a superintelligent AI system could, in theory, act in ways that threaten human society or existence. The concern here is not just about malevolent AI, but equally about well-intentioned AI systems whose objective misaligns with human values or priorities.

There are several reasons for one to be concerned about AI existential risk. The first is that AI development follows a rapid pace. By now, large language models can easily fool us into believing that a human has written a text. Soon, some scholars predict, we will be reaching a point of "singularity," where AI surpasses human intelligence, which in turn might lead to unpredictable outcomes. Second, in the case of a superintelligent AI, we should be concerned about AI alignment, which refers to the issue of ensuring that AI systems' means and goals are aligned with human values. An often-cited example is that of a superintelligent AI system that is

programmed to make paperclips³⁰ and wipes out humanity because this would help it to make as many paperclips as possible. Finally, several scholars have drawn attention to the possibility of an AI arms race, which describes the risks emerging from different stakeholders creating powerful AI systems without adequate safety measures. This could lead to catastrophic, if not existential, global conflicts.

Just as there are people advocating for existential risk research, there are others who believe the above concerns are overstated. A key argument that is often made refers to the current state of AI development. It is true that AI has surpassed us in some tasks, such as chess or large-scale calculations. Yet, current AI is “narrow” in that it is good only at one task. Despite impressive advances, many scientists say, we are still far away from superintelligent or even general human-level AI systems. Another argument puts trust in our regulatory and governance mechanisms: As AI systems become more capable and awareness of their risks grows, we will surely be able to develop the right governance frameworks to contain those risks. Finally, some scholars draw parallels between historical events and the initial reactions to those events, which often included hype and unfounded fears.³¹ With the introduction of the railway, for example, some thought that humans would melt at speeds faster than 50 miles per hour. Generally speaking, practical and ethical considerations have historically guided the responsible integration of technology into society.

³⁰ Bostrom, Superintelligence

³¹ <https://www.afr.com/technology/what-happens-when-the-god-ai-arrives-20230524-p5dasr>

8.0 Global Challenges and Risks

AI poses several challenges and risks that could manifest at the global level. These include economic risks associated with the potential displacement of jobs by AI, as well as the exacerbation of existing socio-economic inequalities brought on by a widening gap between those who have the skills to work alongside AI systems and those who lack them. In addition to inequality experienced at the individual level (based on skill attainment), there is also the risk of a widening gap between nations (in terms of economic growth and political influence) based on their relative ability to contribute to or take advantage of advancements in AI.

The ability to leverage AI to mislead or influence public opinion through misinformation is an ongoing concern for governments worldwide, as it can affect democratic elections or lead to a loss of trust in institutions or the reliability of information itself.

Finally, the growth in the data economy and the widespread collection of data for the purpose of training AI models has caused data privacy to be a top concern globally

8.1 Economic Risks from AI

One of the most immediate economic concerns is the displacement of jobs by AI. An initial study from 2017 by two Oxford-based researchers made a splash in the media,

claiming that 47% of total US employment is at risk from computerization over the next decade or two.³² Immediate responses were both alarmist and skeptical, with subsequent studies often producing much lower numbers. With the release of OpenAI's ChatGPT, however, the discussion about job displacement has been reignited due to the perceived potential of large language models (LLMs) to affect work. A study by Accenture estimates that on average 40% of all working hours could be affected by LLMs. The World Economic Forum has released updated predictions (2023) that 83 million jobs will be destroyed and 69 million created, which leaves a net job loss of 14 million worldwide—2% of total employment. Goldman Sachs (2023) predicted that within the US and Europe, two-thirds of jobs are exposed to some degree to AI automation, and a quarter of them could be automated.

When looking at past technological breakthroughs, such as the ones that led to the industrial revolution, automation traditionally affected low-skill job sectors. This may not be the case with AI, as high-skill jobs, such as accounting equally seem to be at stake. Nevertheless, the gap between those who have the skills to work alongside AI systems and those without them could widen, further exacerbating existing socio-economic inequalities. For risk professionals, this underscores the need for strategies that address workforce re-skilling, and policies that mitigate the unequal distribution of AI's economic benefits and burdens.

Predictions about job loss should be taken with care, however. It is difficult to predict how quickly AI will advance, and how quickly it will be ready to take over tasks. It is also too early to say how much (expert) human supervision will be required in

practice and by law for many tasks. And it is difficult to forecast the potential for new job creation resulting from automation.

³² Frey, Carl Benedikt, and Michael A. Osborne. "The future of employment: How susceptible are jobs to computerisation?." *Technological forecasting and social change* 114 (2017): 254-280.

8.2 Global Inequality

Currently, most large AI firms are based in the US, and technological adoption of AI systems varies widely between countries. At this point, there is a risk that AI advancement will exacerbate global inequalities with nations that are at the center of AI development leaping ahead in terms of economic growth and political influence, leaving behind less technologically developed countries. This presents ethical considerations as well as risks of triggering economic and geopolitical tensions

8.3 Misinformation Campaigns

AI technologies enable sophisticated misinformation campaigns. Ranging from bots that create misleading, harmful social media posts to AI generated imagery and videos, AI has the potential to mislead and influence public opinion. This poses a serious risk to political institutions and democratic elections. Managing risks from misinformation campaigns requires strict governance measures that ensure

information authenticity and reliability, as well as the deployment of AI detection tools to expose AI generated content

8.4 Privacy and Surveillance

Many AI models require large amounts of data for training. As a result, there is a strong motivation for companies to collect such data. Risks to privacy have therefore exploded alongside the growth in the data economy

9.0 Conclusion

The topics presented in this module underscore the profound impact AI technologies can have on individuals, companies, governments, and society as a whole, and the importance of managing these risks effectively. In the next module, we build on the risks we've examined, and explore how ethical principles and governance can guide the development and deployment of AI technologies in a way that promotes trust, safety, and fairness

Risk and Risk Factors: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

1. What is algorithmic bias?

Algorithmic bias is a systematic deviation of an algorithm's output, performance, or impact relative to some norm, aim, standard, or baseline.

2. Name three group fairness measures and provide either their informal or formal definition.

- Demographic parity: the distributions of predictions is identical across subgroups;
- Predictive rate parity: the proportion of individuals correctly identified as positive is equal across subgroups; ; also possible to express in terms of precision = $TP/(TP+FP)$
- Equal opportunity: the proportion of positive individuals that are correctly identified as positive; ; also possible to express in terms of sensitivity = $TP/(TP+FN)$

3. Name some possible sources of algorithmic bias.

- Historical bias
- Data collection (representation bias; measurement bias)
- Data composition (imbalanced datasets; statistical minorities)
- Model development (preprocessing techniques; choice of objective function)

- Deployment (context)

4. What is the difference between explainability and interpretability?

Explainability refers to the capacity to explain in understandable terms how an AI system makes decisions or predictions, often after the fact (hence referred to as "ex-post explainability" in the literature). Interpretability, on the other hand, refers to the degree to which a human can comprehend and predict the model's behavior with built-in mechanisms to understand inherently how the inputs in the model affect the outputs (hence referred to as "inherently interpretable models" in the literature).

5. Name three distinct ways in which the mode of deployment of algorithms can be opaque.

- Secrecy: individuals are not aware of the existence of the algorithm
- Confidentiality: individuals lack access to the workings of the algorithm
- Specialized knowledge: understanding the algorithm would require specialized training

6. What are surrogate models, why do we use them, and what are the challenges associated with them?

Surrogate models are simple(r) models that approximate the behavior of a complex model. They are used to enhance interpretability of an AI system. Challenges are oversimplification, potential computational costs, and the fact that such models are ex post techniques.

7. What is automation bias and how could it be mitigated?

Automation bias is the tendency of humans to over-rely on automated systems. Mitigation techniques include informing users about the bias itself as well as to implement design mechanisms that require periodic human input and verification.

8. True or False: Regarding bias, if a sample is representative of the population, algorithms will always perform equally well or equally bad across all subgroups.

False

9. How does group fairness differ from individual fairness?

As opposed to individual fairness, group fairness does not consider the treatment of individuals, but instead looks at the statistical differences between groups. For example, we may ask whether there are statistical differences between the admission rates of male and female college applicants. This would be a matter of group fairness, as we are interested in whether there are certain groups that are disadvantaged by the algorithm and that share a protected characteristic.

10. Identify and describe two common XAI techniques aimed at making algorithmic decision making more explainable.

- *Feature importance scores* is a prominent technique used in XAI. It ranks the importance of input features for outcomes. For example, consider the case of a credit-scoring AI system. Feature importance scores can reveal which factors (e.g. income, credit history, etc.) have the most influence on the credit score. Feature importance scores can not only help to scrutinize how a model makes decisions but can help identify potential biases or errors in the model.

- *Surrogate models* are what their name suggests: simpler, more interpretable models that approximate the behavior of a complex AI system. LIME (local interpretable model-agnostic explanations), for example, approximates a complex model's prediction locally with an interpretable model (e.g. linear regression or decision-tree).

11. What are some challenges associated with XAI?

- An oversimplification of a complex decision-making process can distort our understanding of the decision to the point of misinterpretation.
- XAI techniques may be computationally intensive, which makes developing and deploying them more costly.
- XAI techniques are typically ex-post techniques, which means that they are used to understand the models after they have been trained.

12. What are some common causes for AI-related reputational damage?

- privacy breaches
- algorithmic bias
- lack of explainability

13. What are examples of economic concerns associated with AI?

- The potential for displacement of jobs by AI

- The potential for a widening gap between those with the skills to work alongside AI systems and those without them, further exacerbating existing socio-economic inequalities

Module 4: Responsible & Ethical AI

Learning Objectives

This module builds on the risks examined in Module 3 and explores how ethical principles and governance can guide the development and deployment of AI technologies in a way that promotes trust, safety, and fairness. It also presents various ethical frameworks that can be applied to AI, the governance challenges associated with AI, and current global governance initiatives around AI.

After completing this module, you should be able to:

- Discuss potential benefits of implementing a practical ethics framework.
- Compare and contrast consequentialism, deontology, and virtue ethics.
- Discuss the principles of nonmaleficence, beneficence, justice, autonomy, and explainability.
- Discuss sources of and strategies to address algorithmic bias and unfairness.
- Describe important ethical principles related to privacy.
- Discuss the current regulatory landscape and governance challenges associated with AI.

1.0 Introduction

Responsible AI refers to the ethical and responsible development, deployment, and use of artificial intelligence systems. It aims to ensure that AI benefits society holistically while mitigating potential risks. Responsible AI is about aligning AI systems in a variety of ways, including ethical standards, industry standards, regulation, legislation, global principles, and more to ensure that they operate in a fair, transparent, and accountable manner.

Ethical frameworks help us decide how to design AI, and how to deploy it in a way that minimizes risks, harms, and wrongs. Ethical frameworks shape AI by helping us define principles to abide by, identify unacceptable biases, assess what kind of transparency is desirable, identify stakeholders and their interests, promote accountability, and make justifiable decisions about the design and implementation of AI. Different ethical frameworks help us make justifiable decisions in pluralistic societies.

2.0 Practical Ethics

Practical ethics is the application of ethical theory, and the development of good practices to solve real-world moral dilemmas. The goal of practical ethics is to provide concrete guidance for moral decision making and problem solving. Some key aspects of practical ethics include:

- **Applied focus.** Practical ethics aims to address actual ethical dilemmas that arise in domains like medicine, law, politics, business, and now AI. It can offer recommendations, not just theories.
- **Multidisciplinary.** Practical ethics draws on moral philosophy, social sciences like psychology and sociology, domain expertise, law, computer science, and other fields to inform the analyses of complex issues.
- **Context-sensitive.** Practical ethics emphasizes that situational nuances matter in moral decision making.
- **Pluralistic approach.** Different ethical frameworks like consequentialism, deontology, and virtue ethics can provide insights for evaluating issues. Practical ethics may blend these approaches.

Typical decisions about moral dilemmas involve making a choice in which someone stands to lose out like, for example, when the task is to distribute scarce resources, or design an algorithm that selects job candidates. Practical ethics aims to make decisions in a way that can be justifiable to the whole society.

In short, practical ethics aims to offer actionable advice for ethical issues by applying philosophical theories, principles, and methods in context, using an interdisciplinary lens, and focusing on practical guidance over theoretical debates.

2.1 Why Might a Firm Consider a Practical Ethics Framework?

Proponents of practical ethics would argue that it can provide a foundation for decision making that may help firms when facing difficult challenges. Furthermore, they would assert that it can also make good business sense by providing benefits to a company, such as:

- **Building trust and reputation.** Customers, employees, providers, partners, and the public may be more likely to trust and support companies that are known for adhering to an ethical standard. This, in turn, can improve brand image.¹
- **Avoiding scandals.** Unethical behavior, like fraud, can damage a company's reputation and bottom line. Practical ethics can help avoid major scandals.²
- **Attracting and retaining talent.** There are workers and individuals entering the job market that are increasingly concerned about the responsible and ethical practices of the companies with which they interact. Therefore, having a practical ethics framework may help in recruiting and retaining talent.³
- **Strengthening culture.** Ethics programs can nurture teamwork, accountability, and integrity in the workplace culture, which in turn can boost morale and productivity.
- **Supporting risk management.** Having an ethical framework undergirding policies and procedures can help companies reduce their risk of safety failures, data breaches, and regulatory non-compliance.⁴
- **Encouraging and guiding innovation.** An ethical framework can help guide innovation in a responsible direction that considers impacts on people and society.
- **Promoting long-term thinking.** Managing in an environment in which there are ethical considerations can help promote a long-term view.

In summary, proponents of practical ethics would argue that they are good for business. Regardless, ethical frameworks underlay many of the responsible AI approaches that are advanced by regulatory, quasi-regulatory, legislative bodies, consulting firms, and businesses around the world. As such, it is helpful to understand the major ethical frameworks that have developed over the years.

- ¹ Brien, A. Professional Ethics and The Culture of Trust. *Journal of Business Ethics* 17, 391–409 (1998). <https://doi.org/10.1023/A:1005766631092>; Brenkert, G. Trust, Business, and Business Ethics. *Business Ethics Quarterly*, Volume 8, Issue 2, April 1998, pp. 195 – 203. Why Digital Trust Truly Matters, 2022, McKinsey Report, <https://www.mckinsey.com/capabilities/quantumblack/our-insights/why-digital-trust-truly-matters>
- ² Why Digital Trust Truly Matters, 2022, McKinsey Report, <https://www.mckinsey.com/capabilities/quantumblack/our-insights/why-digital-trust-truly-matters>; Carson, T.L. Self-Interest and Business Ethics: Some Lessons of the Recent Corporate Scandals. *Journal of Business Ethics* 43, 389–394 (2003). <https://doi.org/10.1023/A:1023013128621>; Sen, A. (1993). Does Business Ethics Make Economic Sense?. In: Minas, P.M. (eds) The Ethics of Business in a Global Economy. Issues in Business Ethics, vol 4. Springer, Dordrecht. https://doi.org/10.1007/978-94-015-8165-3_6
- ³ Fernández-Aráoz, C. A Strong Purpose Can Make Your Company a Magnet for Talent. 2023. *Harvard Business Review*. <https://hbr.org/2023/11/a-strong-purpose-can-make-your-company-a-magnet-for-talent>
- ⁴ Why Digital Trust Truly Matters, 2022, McKinsey Report, <https://www.mckinsey.com/capabilities/quantumblack/our-insights/why-digital-trust-truly-matters>; Doyle, E.M., Hughes, J.F. & Glaister, K.W. Linking Ethics and Risk Management in Taxation: Evidence from an Exploratory Study in Ireland and the UK. *J Bus Ethics* 86, 177–198 (2009). <https://doi.org/10.1007/s10551-008-9842-9>; Young, P. Ethics and Risk Management: Building a Framework. *Risk Manag* 6, 23–34 (2004). <https://doi.org/10.1057/palgrave.rm.8240187>

3.0 Ethical Frameworks

Three common ethical frameworks worth considering in the context of AI include the following: **Consequentialism**, which is an ethical theory that judges the morality of an action based on the consequences of that action; **Deontology**, which is an ethical framework that judges the morality of actions based on adherence to ethical duties and rules rather than focusing on consequences; and **Virtue Ethics**, which emphasizes virtuous character traits and living a good life, rather than rules or consequences.

3.1 Consequentialism

Consequentialism is an ethical theory that judges the morality of an action based on the consequences of that action.

At its core, consequentialism focuses on the outcomes or results of an action to determine whether it is right or wrong. The most common form is *utilitarianism*, which aims to maximize overall utility. Utility is often defined in terms of pleasure, happiness, or the satisfaction of desires. Under utilitarianism, the morally correct action in any situation is the one that produces the greatest net utility for all affected.

For example, designing or implementing a particular AI system would be considered morally good under utilitarianism if it increases the overall pleasure across society more than other actions one could take. Actions are not intrinsically moral, but derive their moral value solely from their results. Utilitarianism is forward-looking, circumstantially relative, and focused on end consequences. Two of the most famous proponents of this approach were Jeremy Bentham and John Stuart Mill.⁵

A key advantage of consequentialism is that it provides a single, quantifiable metric for determining moral value. One shortcoming, however, is that the choice of metric is highly subjective, and quantification of value can be challenging. Utilitarian calculations aim to be impartial, objective, and amenable to scientific measurement of utility. However, consequences are often unpredictable, and it is unclear where the calculation should stop (i.e., should we consider only immediate consequences, or also the consequences of the consequences, and the consequences of those, etc.?).

Critics of utilitarianism and other forms of consequentialism argue that always maximizing utility can lead to actions that many consider immoral, like severely violating individual rights for the greater good. Utilitarianism struggles with situations in which utility is maximized by something most would consider unethical. The classic example is one in which a doctor has the chance to kill an unfriendly patient to use his organs and save five other patients. In the realm of AI, someone could justify implementing an AI system that violates rights with the justification of it being more efficient and therefore saving resources.

In response, some consequentialists grant moral weight to following general moral *rules* (as opposed to acts) that tend to maximize utility. *Rule consequentialists* judge acts by whether they adhere to utility-maximizing rules, not only by their case-specific outcomes. This workaround addresses some issues with utilitarianism by ensuring rules against murder, lying, and the like are upheld even when breaking them may increase utility in isolated cases. The doctor should not kill her patient because that would undermine trust in the medical system. We should not

implement AI systems that violate rights because, in the long run, that would create more problems than it solves. Some theorists, however, believe that rule consequentialism collapses into deontology.

⁵ Driver, Julia, "The History of Utilitarianism", *The Stanford Encyclopedia of Philosophy* (Winter 2022 Edition), Edward N. Zalta & Uri Nodelman (eds.),.

3.2 Deontology

Deontology is an ethical framework that judges the morality of actions based on adherence to ethical duties and rules rather than focusing on consequences.

One of deontology's most important proponents is Immanuel Kant.⁶ According to Kant, individuals have a moral obligation to act in a way that is universally applicable and treats others as ends in themselves, rather than only as means to an end. In other words, we shouldn't treat people like things. People are ends in themselves, in that they have their own values and goals as a part of being autonomous.

The *categorical imperative*, a fundamental principle of Kantian ethics, asserts that one should act only according to maxims that could be willed as a universal law without contradiction, such as, "It's wrong to lie." Kant emphasizes the importance of moral principles, rationality, and a sense of duty in guiding ethical decision making, irrespective of the consequences.

Morality, in Kant's view, is grounded in reason and the intrinsic value of individuals, providing a principled foundation for ethical behavior. Kant is famous for not having placed moral weight on consequences. In his view, lying is always wrong, irrespective of potential consequences.

Most contemporary deontologists care about consequences and grant them moral weight. For a deontologist, however, consequences are not the *only* moral consideration worth taking into account and, for most deontologists, there will be red lines that should not be crossed even when it seems like the consequences might be beneficial overall.

Rights, rules, and ethical principles are all deontological in nature. They provide ethical guidance of what to do and what not to do that goes beyond consequences.

⁶ Alexander, Larry and Michael Moore, "Deontological Ethics", *The Stanford Encyclopedia of Philosophy* (Winter 2021 Edition), Edward N. Zalta (ed.)

3.3 Virtue Ethics

Virtue ethics emphasizes virtuous character traits and living a good life, rather than rules or consequences. It has roots in ancient Greek philosophers like Aristotle, who taught that happiness comes from living a life guided by virtues.⁷

The key question in virtue ethics is, "What kind of person should I be?" Rather than focusing on universal duties or maximizing utility, virtue ethicists ask what character traits we should cultivate to live well. These virtues include wisdom, courage, humanity, justice, temperance, and generosity. Acting virtuously means exercising practical wisdom to moderate our emotions, appetites, and behavior appropriately in each situation.

Virtue ethicists believe we should aspire to ideals of human excellence. Virtues are nurtured through practice, habit, and modeling virtuous exemplars. One way to approach ethical dilemmas from a virtue ethics point of view is to ask what a virtuous agent would do in a particular situation; the agent can be thought of in the abstract or as a concrete example (e.g., What would Jesus, Mohammed, Solomon, Gandhi, etc. do in this circumstance?). Virtue ethics sees morality as a matter of character built over a lifetime, not just discrete acts.

Critics argue virtue ethics lacks clear guidance for moral decisions compared to duty-based or consequentialist approaches. Because different virtues can conflict, how to weigh them is unclear. Virtue ethicists counter that practical wisdom helps navigate hard cases, and that they are in no disadvantage with respect to other theories; moral duties can also conflict, and consequences are not always comparable. Virtue ethics also integrates well with common morality, given that most people seem to learn about morality through habituation in the context of socialization (e.g., parents teaching us over and over how to behave kindly toward others).

Modern developments in virtue ethics expand its scope beyond individual character. For organizations and societies, virtues might include justice, accountability, environmental stewardship, and responsible innovation. Virtue ethics is seeing renewed interest across disciplines like moral psychology and business ethics. In the context of AI, some scholars have suggested that to build AI that is ethical, we must build it in a virtue ethics way, which would imply it learning from experience and habit, like children do. Otherwise, morality is so complex that we might never be able to code it in a top-down approach.

As mentioned before, consequentialism, deontology, and virtue ethics are not mutually exclusive, and in the context of practical ethics, they complement one another. The best kind of moral decision is one that accords with all three theories, that is, an act that maximizes good consequences, respects rights, complies with ethical principles, and embodies virtues.

⁷ Kraut, Richard, "Aristotle's Ethics", *The Stanford Encyclopedia of Philosophy* (Fall 2022 Edition), Edward N. Zalta & Uri Nodelman (eds.)

4.0 What Can AI Ethics Learn From Medical Ethics?



Play Video

In considering a constructive path for AI ethics, it may be helpful to turn to another realm within practical ethics with a more extensive history: medical ethics.

Ethical concerns have a long history in the field of medicine, given its direct involvement in matters of life and death. The Hippocratic Oath, thought to have been first written in Greece between the fifth and third centuries BC, emphasizes the importance of physicians doing no harm to patients. Since that time, there have been various attempts at formalizing a code of medical ethics and exploring issues related to medical ethics.⁸

The American Medical Association adopted its first code of ethics in 1847, drawing on earlier work done in the UK. There was an acceleration of, and an increased attention to issues surrounding medical ethics in the 20th century, which saw the creation of several important documents, including the Nuremberg Code (1947), the Declaration of Geneva (1948), the Declaration of Helsinki (1964), and the Belmont Report (1978). Medical ethics became more fully evolved in the 1970s, driven by factors including the increased concentration of medical care in hospitals and other depersonalized settings, the rising cost of medical care and increased role of government in health insurance funding, the development of "patients' rights" as an outgrowth of broader efforts around civil rights, public outrage at medical scandals such as the Tuskegee Syphilis Experiment, and rapid advances in technology.

Technological advances posed new ethical challenges for doctors that needed solutions. The advent of the mechanical ventilator, for instance, prompted a reconsideration of the concept of death and led to the development of ethics surrounding organ transplantation. Physicians were now confronted with the dilemma of warm, heart-beating bodies with non-functioning brains, who presented an opportunity for organ procurement for transplantation. Whether to take the organs of these bodies is a moral question, not a medical one.

Practical needs, therefore, were an impetus behind the establishment of ethical frameworks, emphasizing that the responsibility of resolving ethical dilemmas should not rest solely on healthcare professionals, whose expertise lies in maintaining health rather than navigating ethical complexities.

In a manner not wholly dissimilar to that of the medical field in the 1970s, AI and other digital technology companies have found themselves to be central figures in significant controversies in recent years. As people have become concerned about the potential impact of digital practices on their lives, demand for ethical standards has grown.

Furthermore, with rapid advances in technology related to the collection, analysis, and utilization of personal data, along with the design of new applications, platforms, and tools such as autonomous cars, novel ethical dilemmas have arisen. Engineers, programmers, data analysts, risk managers, CIOs, CEOs, and Boards find themselves faced with new challenges that their training and experience may not fully equip them to face.

⁸ *Formula Comitis Archiatrorum* written in the 5th century is thought to be the first known code of medical ethics. Islamic (e.g., Ishaq ibn Ali-Ruhawi in the 9th century), Jewish (e.g., Moses ben Maimon in the 12th century), and Christian (e.g., Thomas Aquinas in the 13th century) scholars have also been concerned with and made contributions to the field of medical ethics.

5.0 Principles of AI Ethics

Although most AI ethics codes contain long lists of principles, the following principles of nonmaleficence, beneficence, justice, autonomy, and explainability are both relevant and common.

5.1 Nonmaleficence

The principle of nonmaleficence asserts an obligation to avoid harming others or inflicting injuries. Part of what it means to avoid harming others is a prohibition on imposing risks of harm that are not justified or that outweigh potential benefits. In other words, not only should you not go around hurting others (e.g., subjecting them to algorithms that can harm them), but you should also not impose unnecessary or unjustified risk on others (e.g., subject them to untested algorithms that could be harmful).

Nonmaleficence does not prohibit all types of harm unconditionally. Some level of risk is permissible if it enables benefits that justify that risk, or if no alternatives are available. For instance, medical procedures (and clinical trials) inherently incur some risk but may still be justified by their necessity and benefits.

Importantly, it matters who is making the decision, and who will bear the brunt of the harm if things go badly. It is more ethically acceptable to impose risks on people who stand to benefit from whatever the proposed action is. For example, very risky clinical research may be morally acceptable if the research subjects suffer from a sufficiently serious ailment and stand to benefit from the research if it goes well. The same risky research may

very well be considered morally unacceptable on healthy research subjects, or on research subjects who have an ailment that does not stand to be cured by the research. An analogous situation in AI would be considering it unacceptable for people who are at no risk of harm to impose algorithmic risks on people who do not stand to gain from those risks.

5.2 Beneficence

The principle of beneficence refers to the moral obligation to act for the benefit of others. Beneficence requires taking positive steps to help others, rather than simply refraining from harm. It moves beyond nonmaleficence, which tells us not to injure others, and commands us to advance the welfare and legitimate interests of people in need. Beneficence could include acts like donating to charity, volunteering, and providing resources or assistance to improve people's lives.

A common misunderstanding is that beneficence is solely an outcome-focused, consequentialist concept. However, duty-based deontological frameworks include beneficence as an obligation we must fulfill above and beyond what may maximize utility.

There are limits to the duty of beneficence, however. No one individual can alleviate all suffering in the world, so reasonable constraints apply. Considerations like scarce resources, competing obligations, reasonableness, and demandingness (i.e., there's only so much ethics can demand of individuals) should factor into determining the extent of our duty of beneficence. Additionally, the recipient's right to autonomy may preclude unwanted "benefits" that disrespect personal agency and choice. People have a right to decide what is best for them, and with some exceptions (e.g., public health worries, or worries about whether a person is autonomous), that right usually trumps unwanted offers of beneficence. Nonetheless, within these bounds, actively pursuing the welfare and legitimate interests of others remains a key deontological duty.

In the context of AI, one way to think about beneficence is a duty that AI systems benefit humanity in some way. At a minimum, an AI system should offer solutions to problems, and be designed to improve the lives of those who interact with it.

5.3 Justice

Justice refers to the moral obligation to act in accordance with principles of fairness, equality, impartiality, and proportionality. In ethics, justice requires giving each person his or her proper due while upholding duties toward fairness and equality.

There are different concepts of justice. Procedural justice demands fair processes and impartiality. Distributive justice focuses on equitable allocation of benefits and burdens in society. Restorative justice aims to repair harms through reconciling victims and offenders. Interactional justice concerns respect and fairness between individuals. Social justice refers to just institutions in society that provide for basic rights and needs.

Justice is concerned with ensuring human rights are respected, resources are distributed equitably, opportunities are available to all, the law is applied impartially, and no one is discriminated against unfairly. Violations of justice may lead to human rights abuses, discrimination, corruption, inequality, and exploitation of vulnerable groups.

However, there are debates around what constitutes a just distribution of goods or a fair process. Different principles of justice - like egalitarianism, utilitarianism, meritocracy, or need-based allocation - can conflict. There are also disagreements around what goods justice should be concerned with distributing, like resources, opportunities, power, or welfare.

Despite these debates, there is broad agreement that justice is a vital moral principle and remains a cornerstone of ethics. To enjoy legitimacy, moral decisions must be justifiable to all and align with what is fair. Another point on which there is broad agreement is that, as a matter of justice, people should not be discriminated against for characteristics that are morally irrelevant (e.g., race).

5.4 Autonomy

Autonomy refers to the capacity of people to make their own informed, un-coerced decisions about their lives and actions. As an ethical principle, autonomy commands respecting and supporting others' abilities to determine their own course in life.

In cases in which autonomy may be constrained because a person lacks the capacity to make rational decisions (i.e., children, unconscious patients, and patients who lack certain crucial cognitive abilities), a surrogate decision maker, such as a family member or a legal guardian, may need to act in the individual's best interests.

Autonomy has roots in humanistic and existentialist traditions. It depends on capacities for self-awareness, independent decision making, critical reflection, and personal freedom. Infringing on someone's autonomy contravenes her right to direct her own life.

In healthcare, respect for autonomy is crucial. Patients have a right to voluntary informed consent and refusal regarding their treatment. The doctor's duty is to inform the patient appropriately, and it is up to the patient to decide what, if any treatment to pursue. Coercion, deception, manipulation, and undue influence all undermine autonomy.

An ethical AI system respects people's autonomy by not using coercive or manipulative tactics to get people to act in a particular way. Technology should help people further their own life goals, as opposed to trying to further the goals of third parties (e.g., companies, governments, etc.).

5.5 Explainability

The ethical principle of explainability (sometimes called *explicability*) has gained significant attention in the context of AI. It refers to the idea that AI systems, especially those with decision-making capabilities, should provide transparent and understandable explanations for their actions or decisions.

One reason explainability is thought to be important is for the purposes of accountability. Decisions made by AIs (particularly in areas like healthcare, finance, and criminal justice) can have a profound impact on individuals' lives. Ensuring that AI systems can explain their decisions is essential for being able to hold accountable the companies and people that design and implement them. It can also help identify and rectify errors, biases, or unfair practices.

Explainability is also thought to further trust. Trust is a fundamental component of the adoption and acceptance of AI technologies. If users or stakeholders cannot understand how a system reaches its conclusions, they are less likely to trust it. Explainability fosters trust by making AI systems more transparent and predictable.

Without the ability to explain why an AI system made a particular decision, it becomes harder to ensure that it adheres to ethical guidelines and respects individual rights.

There are various levels and types of explainability in AI:

Local explainability focuses on explaining the decisions of a specific AI model on a single instance or prediction. Local explanations provide insights into why a particular decision was made for a particular case.

Global explainability looks at an AI model's overall behavior and decision-making processes. It provides a more comprehensive understanding of how the model operates across various inputs.

Model-specific explainability refers to the fact that some AI models have specific explainability techniques tailored to their architecture. For example, decision trees have intuitive rules for explaining their decisions, whereas deep neural networks may require different methods. In contrast, *model-agnostic methods* are designed to work with any AI model, making them more versatile. They don't rely on the specific architecture or algorithms used in the model. There is considerable debate about what exactly counts as an explanation and to whom an explanation is owed. The former partially depends on the latter because the explanations intended for experts will likely differ from the kinds of explanations that are intended for regulators or ordinary citizens.

What counts as a good explanation will likely vary depending on the kind of AI, but one popular approach is to develop counterfactual explanations. Consider a case in which an algorithm decides whether to grant loans. A counterfactual explanation might involve presenting a hypothetical scenario that contrasts with the actual decision. For instance, suppose the AI denies a loan to an individual based on certain criteria, such as having too little money in the bank, or earning too low a salary. A counterfactual explanation could be constructed by presenting an alternative scenario, stating the conditions under which the loan would have been approved (e.g., having \$10,000 more in the bank, or earning \$1,000 more per month as a salary). This counterfactual scenario helps the individuals understand the specific factors that led to the denial and provides actionable insights, such as increasing their salary or their bank savings. Counterfactual explanations contribute to transparency and help users comprehend the influence of different variables on AI decisions.

6.0 Bias, Discrimination, and Fairness

Bias within the realm of AI refers to a systemic deviation in the output or impact of an algorithm compared to a desired norm or standard. Essentially, an algorithm is considered biased when it deviates from its intended function. Suppose an algorithm is designed to identify the best job candidates for a position as an executive. If the algorithm tends to recommend for or against candidates based, not on their qualifications, but on an unrelated feature such as race or sex, then it is biased because it is deviating from its intended function.

A well-designed AI should align with its stated purpose, optimizing performance according to established standards. The aim is not to achieve an entirely "objective" algorithm, as every algorithm inherently reflects values embedded in its design. These values are shaped by the perspective that certain aspects are deemed valuable or important, as the algorithm strives to excel based on specific metrics. For instance, an algorithm assessing loan eligibility may prioritize a person's bank account balance, considering it relevant to optimizing loan repayment.

Not all biases are inherently problematic from an ethical standpoint. Justifiable biases can form part of a well-designed AI. Conversely, not all AIs that are statistically or legally unbiased are necessarily ethically acceptable. Even statistically unbiased algorithms can inflict unwarranted harm, such as implementing a service that charges exorbitant fees to everyone. The ethical concern when it comes to AI bias arises when biases result in unfairness, disadvantaging individuals for unjustifiable reasons in comparison to others.

6.1 Problematic Biases

Problematic biases in algorithms often occur unintentionally. Due to their complexity, algorithms can inadvertently incorporate ethically problematic biases. Four primary sources contribute to biases:

- Problem specification
- Data
- Modeling, validation, and design

- Deployment

This overview of biases is not exhaustive, and various categorizations exist that consider possible interactions and the multifaceted nature of biases in algorithms.

6.1.1 Biases in Problem Specification

An algorithm may exhibit bias from its inception if the goals it is designed to achieve contain inherent problems. Operationalizing complex goals is a nuanced task, and often, the selected target variables may fail to capture real-world objectives accurately. For instance, if a bank aims to eliminate all risks and lends only to individuals that it is certain will repay, it may inadvertently end up catering exclusively to affluent individuals. In such cases, modifying the target goals to accommodate a reasonable level of risk may be necessary. Another example is an algorithm that was meant to identify patients who are sicker to assist health professionals with triage, but that was using health care expenditure as a proxy, thereby favoring not the sickest patients, but the richest ones who tend to spend more on healthcare.

6.1.2 Biases in Data

If they rely on historical data, ML algorithms may tend to perpetuate biases from the past; this propensity is commonly known as *historical bias*. In addition to the use of historical data possibly perpetuating bias related to features such as sex and race, it can more broadly lead to inaccurate or malfunctioning algorithms, especially when lab data does not align with real-world trends.

Consider a theoretical dataset containing all relevant data for all the loans that have ever been made. That data would likely show that successful loans (i.e., loans that have been repaid in the agreed-upon time frame) have mostly been given to men, as women have been excluded from active participation in the banking system until relatively recently. If an algorithm used that full historical data set as input, it would likely favor men, even if there is no valid reason for such a preference. Likewise, a dataset that spanned the pre-pandemic years 2010-2019 that was used to help predict office vacancy rates and commuter rail volume for 2025-2030 may lead to inaccurate predictions because the patterns that existed in 2010-2019 may be significantly different from those in 2025-2030.

A different but related data challenge is that historical data rarely show counterfactual outcomes. This problem is called the *selective labels problem*. For example, a company probably doesn't track the career progression of those it didn't hire; therefore, it will never know whether it indeed hired the best candidate. A bank has data on the people to whom it gave loans, but it doesn't have data on the people to whom it denied loans. The people who were denied loans might've become even better clients than those to whom it gave loans, but because it doesn't have that data, it will continue to select people who are like those to whom it has granted loans in the past.

Another related but distinct kind of bias stemming from data is *sampling bias*. It is a bias that is well known in science. Sampling bias arises when the data sample is not random. If the data sampled are not random, the trends shown by the population under study may not generalize to another population. Let's suppose that most of our data comes from young men. And let's suppose that an AI finds that a particular drug at a certain dose is effective in treating pain. Even if that correlation were not spurious, it may not generalize to other groups such as women or the elderly.

6.1.3 Biases in Modeling, Validation, and Algorithm Design

Even when the problem specification is sound and the data unbiased, biases can emerge during the modeling, validation, and design phases of algorithms. Choices related to optimization functions, the application of different regression models, consideration of subgroups, and how information is presented can all introduce biases. For example, a search engine designed to help in selecting financial products may inadvertently favor popular items, perpetuating a cycle where popularity leads to increased exposure, irrespective of product quality. These biases can affect the overall fairness and effectiveness of algorithms, highlighting the importance of careful choices throughout the development process.

6.1.4 Biases in Deployment

Even when all other aspects are meticulously addressed, biases can still emerge when an unbiased algorithm is deployed in the real world. Consider an algorithm designed to assess the risk of a person defaulting on a loan. Suppose the algorithm is still undergoing testing, and its limitations are well-known, prompting a cautionary advisory against relying solely on it for decision making. However, in practice, some bank employees may defer entirely to the algorithm's suggestions. Some studies indicate that when human beings receive a suggestion from a computer, they often opt to defer to the automated system. There are a few hypotheses as to why people tend

to defer to automatized systems. It might be because it's convenient and time saving. It might also be because automatized systems appear to be more "objective" and people know their own fallibilities and that might create self-doubt. Perhaps deferring to an algorithm shields people from responsibility. At best, responsibility seems shared, whereas going against the recommendation of an algorithm might expose people who make mistakes to harsher judgments and blame. This tendency to do as we are told by an algorithm creates unintended incentive effects, where individuals might relinquish responsibility, allowing them to attribute blame to the algorithm in case of any issues. The implementation of algorithms can thus inadvertently shape behavior and decision-making processes in unanticipated ways.

6.2 When Does Bias Count As Discrimination?

Whether bias results in discrimination in a legal sense may vary depending on jurisdiction. In general, algorithmic bias is likely to lead to discrimination when it results in disfavoring people based on their race, sex, ethnicity, age, or any other classification protected by law. Such disadvantages typically violate legal protections, and designers, developers, deployers, supervisors of automated systems, and risk managers have an interest in taking proactive and continuous measures to protect against it.

6.3 Fairness

Fairness entails the absence of bias or preference toward an individual or group based on irrelevant characteristics, such as their race. An algorithm is considered fair when it does not exhibit problematic biases. There are two primary types of fairness: group and individual fairness. Group fairness involves statistical criteria, where, for example, in loan distribution, statistical parity would require the demographics of approved individuals mirror the overall population (if 51% of the population is female, 51% of loan recipients should be women). On the other hand, individual fairness emphasizes treating similar individuals similarly, even though defining similarity can pose challenges.

A significant challenge to ensuring fairness in AI algorithms relates to the mathematical impossibility of automating fairness when base rates are unequal. As cited previously, when base rates between populations are different, which is almost always the case, then it is impossible to satisfy demographic parity, predictive rate

parity, and equal opportunity simultaneously.⁹ For instance, consider a scenario in which a majority of individuals who engage in certain criminal activities are men, and an AI is evaluating the risk of a specific man and woman committing a crime. If the AI assigns a higher risk score to the man, he may argue that he is being treated unfairly. Conversely, if the AI assigns a similar score to the woman, she may argue unfair treatment, given that women statistically exhibit a lower likelihood of committing those crimes. Automating fairness becomes feasible only when base rates are equal, which is seldom the case in reality. Fairness can ultimately be considered a moral or ethical judgment, not a mathematical one, and it can involve making imperfect compromises and trade-offs that might need to change in response to changing circumstances.

Fairness is not only about outcome, but about procedure. Procedural fairness provides reassurances, not only that a fair outcome will be sought, but that it will be sought through impartial and just processes. Take the justice system as an analogy. Procedural fairness involves having the right structures in place to have rule of law.

Outcome fairness involves making sure guilty people receive an appropriate punishment and innocent people go free. Sometimes there are mistakes (e.g., innocent people can end up in jail and guilty people can be set free), but when there is a fair process in place, those mistakes can be justifiable (e.g., the evidence suggests guilt or innocence) and there are ways to right some wrongs (e.g., if new exculpatory evidence emerges, convicted individuals can be granted new trials and possibly be released and seek redress for an unfair outcome). In the context of AI ethics, the challenge is to create corporate structures that can carry out both procedural and outcome fairness. For instance, having an ethics committee that can weigh consequentialist, deontological, and virtue ethics considerations to develop and implement best practices can help achieve both outcome and procedural fairness.

⁹ See also Kleinberg, Jon, Sendhil Mullainathan, and Manish Raghavan. "Inherent trade-offs in the fair determination of risk scores." arXiv preprint arXiv:1609.05807 (2016). and Chouldechova, Alexandra. "Fair prediction with disparate impact: A study of bias in recidivism prediction instruments." Big data 5.2 (2017): 153-163.

6.4 Avoiding Problematic Biases and Unfairness

There is no such thing as an "objective" algorithm. Given that there is a mathematical impossibility to satisfy all definitions of fairness simultaneously, the task is not to avoid biases in general, but to avoid problematic biases. For example, it may be that the AI in self-driving cars must be tweaked to make it more sensitive to some people. If the algorithm is tweaked to be especially sensitive to identifying (and avoiding) children, its accuracy when it comes to identifying adults may be diminished.

What is most important is to be aware of trade-offs and to make decisions that are justifiable to the population at large, the stockholders, the stakeholders, regulators, and those who lose out. Consequentialist considerations to be considered in the self-driving car example would include calculating the potential risk of accidents using different versions of the AI. Deontological considerations would include taking care that an AI doesn't disfavor people within protected categories (e.g., it could be that the self-driving car AI is better at identifying men because they tend to be larger), or include safety minimums below which we would not be willing to make compromises. Virtue ethics considerations would include putting in place processes for ethical decision making that would result in responsible professionals and a responsible company.

Among the tools that can help companies avoid problematic algorithmic biases are the following:

Technological solutions. Some toolkits are being developed to assess the amount of fairness in a system.¹⁰ Companies can create their own internal "auditing" systems to identify potential biases that their AIs display. Companies acquiring AIs from third parties can demand bias reports from providers (or, possibly even better, from independent assessors) to be aware of how the AIs they are using behave.

Although some technological tools can be helpful in identifying and avoiding bias, technology rarely solves bias problems on its own. This is partly because of the difficulty in automating fairness. If fairness cannot be automated, then algorithmic auditing for fairness cannot be automated either, and AI cannot solve the fairness problems that it creates.

Data quality. Ensuring that data is diverse, updated, accurate, representative, and free from past discriminatory tendencies goes a long way toward avoiding biases, but again, is not a panacea on its own. In some cases, it is possible to use synthetic data (fabricated datasets). The advantages of synthetic data include that it doesn't entail privacy risks, given that the data does not come from real data subjects; synthetic data can also be more easily rid of problematic biases by design. The disadvantage is that sometimes synthetic data isn't as precise as real data (especially when it comes to visual data), and that it might differ from real data in ways that are not obvious.

Auditing. Algorithmic auditing is likely the best way to identify and correct biases. There are private companies that offer this service.¹¹ Auditing will include using technological tools and statistical analyses, but also a fresh and diverse look at your systems.

Ethical committees or similar structures. Instituting forums in which possible problematic biases can be discussed, and in which decisions about trade-offs can be made considering consequentialist, deontological, and virtue ethics considerations can help ensure procedural fairness and can contribute to outcome fairness.

Training for board and C-suite. Given the wide range of risks to which a firm might be exposed if AI models are designed and implemented in a manner that is not responsible or ethical, firm leadership should be educated as to the risks associated with AI and the importance of ethical/responsible AI practices to help mitigate those risks.

¹⁰ Aequitas, for example, lets users test models regarding different bias metrics for different groups. AI Fairness 360 (AIF360) is another tool that creates a benchmark against which algorithms can be tested.

¹¹ O’Neil Risk Consulting & Algorithmic Auditing, and Ethicas, among others.

7.0 Privacy and Cybersecurity

In the context of AI, someone has privacy with respect to some person or institution and in reference to some personal data point if that person or institution has no access to that personal data point. In other words, we have privacy to the extent that others don’t have access to our personal information. Privacy is important because, among other things, it protects us from possible abuses of power. The more someone knows about you, the easier it is for them to interfere with your life.

Data security is doing what is necessary to prevent unauthorized access to data. Data security includes protecting data from attacks such as ransomware, which can encrypt or destroy data, as well as from theft, and from attacks that can corrupt data. It covers the physical security of hardware and storage devices, administrative and access controls, organizational policies and procedures (including employee training), and software applications.

7.1 Why is Privacy an Ethical Issue?

Privacy is an ethical issue because the lack of it can lead to wrongs, harms, and risks for individuals, institutions, and society at large. In ethics, wrongs are sometimes distinguished from harms. Wrongs can sometimes lead to harm, but they are immoral even when they don’t lead to harm. For example, cheating on your spouse wrongs them, even if they never find out about it and no tangible harm comes from it. Similarly, violating people’s privacy is wrong because it violates their moral and legal rights to privacy, and because it amounts to treating people instrumentally (as things, as a means to *your* objectives, and not as autonomous human beings).

Privacy losses can harm citizens in a variety of ways. Individuals can suffer discrimination, blackmail, exposure and public shaming, identity theft, and the like. Privacy losses are also a potential liability to institutions. Every

personal data point is a potential lawsuit, a potential fine. Finally, privacy losses can result in harm to society. The extent of personal data collection, for example, has made it relatively easy for anyone (including foreign adversaries) to learn of sensitive information about military personnel or politicians and blackmail them, which can endanger national security and democracy in various ways.

The more personal data is collected, the longer it is stored, and the more it is analyzed and shared, the higher the risk of harm down the line. Personal data suffers from the very dangerous combination of being cheap to mine, very valuable, very sensitive and prone to being abused, and very hard to keep safe.

In cyberspace, defenders are at a disadvantage with respect to attackers. Whereas the attacker can choose the moment and method of attack, defenders must always protect themselves against every type of attack. If there's an attacker with enough resources and motivation, it's a matter of time before they get to the data they want. Arguably, the imposition of a risk amounts to a wrong (e.g., someone trying to kill you imposes an unjustifiable risk on you, and therefore a wrong, even if they fail).

Companies that collect more personal data than is needed are creating their own risk. One way to think about it is that personal data is a toxic, albeit potentially highly valuable asset. It might be cheap, easy to mine, and profitable, but it also exposes the company to risk of hacks, leaks, lawsuits, and more. Personal data can also be expensive to manage; given its sensitivity, it needs expensive infrastructure as well as legal teams to ensure compliance. Companies, therefore, have an incentive to consider the risks and rewards related to the collection, storage, and use of personal data when they design products and services.

7.2 Principles and Good Practices

Once again, making decisions about what personal data, if any, to collect, how to keep it safe, how to use it, and how and when to delete it will include a combination of technical and practical capabilities as well as ethical reflection. Consequentialist considerations will include: What are the best- and worst-case scenarios? How can we minimize the risk that the worst happens? Just how sensitive is the data? How confident are we that we can keep it safe? Deontological considerations will include taking care to respect the moral and legal right to privacy, as well as following ethical principles like data minimization (see below). Virtue ethics considerations will include asking ourselves what would a responsible company do given the circumstances.

The following are the most important ethical principles related to privacy and cybersecurity to ensure best practices.

Right to privacy. The right to privacy is generally considered a moral right. That is, for ethical reasons, we have a claim against others that, other things being equal, they do not access our personal data, whether the law in a particular country or historical moment recognizes that right or not. The right is also enshrined in constitutions around the world (thereby making it a legal right as well, in many countries), and in the Universal Declaration of Human Rights.

Data minimization. Data minimization is the most effective way to protect privacy: collecting only the data that is necessary to fulfil a specific purpose. Given that one is imposing a risk on data subjects when one collects their data, personal data should only be collected when the benefit to the data subjects outweighs the possible disadvantages.

Right to be forgotten. The right to be forgotten is a person's right to have private information about them removed from Internet searches and other directories (thereby making it less accessible), when the data is somehow inadequate (e.g., incorrect), irrelevant or no longer relevant, or excessive in relation to the purposes for which it was processed.

Control over data. Giving people control over their data is another measure that can minimize potential abuses of data. This would involve making it easy for individuals to deny or withdraw consent for data collection and processing, to access their personal data (and the inferences that are made about them), transfer their personal data, and delete their personal data.

Contextual integrity. The use of personal data should adhere to contextual norms of privacy. When people give up their data in a particular context, they have certain expectations about how that data will be used. When we transfer the data to a different context, privacy norms are violated. For example, if a person gives personal data to a bank to carry out a transaction, that data should not be sold to a marketing company or used for another purpose. If a patient gives their data to their doctor for the purposes of receiving a diagnosis and treatment, that data should not end up in the hands of a data broker.

Data deletion. Personal data should be deleted as soon as it is not necessary. Routine data deletion is a way to protect individuals, and it is also a way to keep data accurate (personal data tends to change quite quickly; people change tastes, jobs, houses, cars, etc.). Personal data should not be collected with the intention of being kept forever. Having an expiry date is an element of good data-security practices.

Data security. Data security is part of complying with due diligence. It's good practice to use all technical tools available to keep data safe, from strong encryption (and strong passwords) to thorough anonymization of data and use of cryptographic methods such as differential privacy. If an organization cannot keep data safe, it puts itself at risk by collecting personal data in the first place

8.0 Governance Challenges

AI can be difficult and challenging to govern. Some of the difficulties typically considered include power asymmetries, institutional opaqueness, algorithmic opaqueness, lack of AI ethics structures, lack of national regulation, lack of international regulation, unpredictability of how these systems might change or how people might interact with them, generative AI not being truth-tracking, and worries about privacy and copyright.

8.1 Power Asymmetries

Many of the tech companies that are at the cutting-edge of AI are often more powerful—in terms of wealth, influence, and expertise—than some national governments and regulatory agencies. These asymmetries can lead to challenges in terms of legislative and regulatory bodies effectively responding to potential risks posed by AI and its uses.

8.2 Institutional Opaqueness

Most AI systems have been developed by private companies that may not be subject to the same transparency requirements as public institutions or universities. As a result, the public, academics, journalists, policymakers, and regulatory agencies may have little detailed information about the practices that went into designing and training large language models, for example, from the datasets used to details about whether and how the systems were tested and tuned for safety.¹²

¹² One of the promises that resulted from the UK's AI Summit in November 2023 was the commitment on the part of a few companies—including OpenAI, Google DeepMind, and Anthropic—to give the UK government early access to their systems to be tested for safety. The details of this

agreement are unknown. It is unclear whether the government will receive the source code, or what kind of access will be involved. Furthermore, the commitment is not legally binding

8.3 Algorithmic Opaqueness

Even with greater knowledge regarding the companies building AI, there is still a challenge related to the opaqueness of these systems. Neural networks are sometimes called "black boxes" because often not even computer scientists can be sure of exactly how the model is doing what it's doing. One major reason for this lack of transparency is that AI systems like large language models are trained using a method called backpropagation, which adjusts the weights of the neural network so as to minimize the error between the model's output and the desired output. Although this method can be effective at improving the model's performance, it does not provide any insight into how the model arrived at its decisions.

The use of proxies is a common technique used by AI systems to simplify the training process by representing complex or difficult-to-measure objectives with a simpler, easier-to-measure metric. For example, an AI system designed to generate news articles might use word count as a proxy for article quality, rather than trying to measure the quality of the content directly. By using proxies, AI systems can make progress toward a goal without needing to optimize directly for the goal itself, which can be a challenging and computationally expensive task. However, this approach can also introduce potential issues, such as optimizing for a goal that is not truly aligned with the system's overall objective, and if outside observers — be it academics or regulatory bodies — don't know what proxies the model is using, it is difficult to govern that model. Additionally, the sheer size and complexity of the models, with millions or even billions of parameters, make it difficult to trace back the processes behind the model's outputs. Different methods to audit the outputs of the systems are being developed to try to get around the difficulty of looking into the "black box."

8.4 Lack of AI Ethics Structures

If we compare AI ethics with medical ethics, the lack of structure seems evident. Medical ethics is supported by bioethicists, ethical codes, and ethics committees, among others. Every doctor must take a bioethics class and is a

licensed professional. Every hospital adheres to international ethics codes. Every clinical research is overseen by an ethics committee; there is nothing similar when it comes to AI ethics.

Even though AI ethics is gradually becoming mainstream, a computer scientist can still go through an education without ever taking a course in AI ethics or being a licensed or certified professional. Although boards are increasingly worried about AI risks, it is still rare to see AI ethicists as board members.

8.5 Lack of National and International Regulation

At the time of writing, there are still not national laws regulating AI (with the exception of China). There are likewise no specialized agencies overseeing AI, and no government-mandated auditing of large language models or other kinds of AI. Just as regulatory agencies have arisen for other kinds of regulation (e.g., the FDA for food and drugs), the future may bring regulatory agencies that have the specialization needed to understand and govern AI.

AI is an international technology, from its sources of data and talent to its implementation and use. It is therefore not unreasonable to expect some movement toward international regulation or agreed best practices.

8.6 Unpredictability Issues

One of the characteristics of some kinds of AI like neural networks is that they present emergent behavior and properties that were not explicitly coded into the system. These systems can therefore surprise human beings--laypeople and experts alike--not only in their capabilities, but also in the kind of mistakes that they make. AIs can also be surprising in the way people interact with them. It can be difficult to predict the uses and misuses to which they can be subject. One potential option to minimize the risk of unpredictability is to subject AIs to randomized controlled trials (RCTs) to test for their safety. Another option, complementary to RCTs, is to audit AIs periodically for safety and accuracy.

8.7 Lack of Truth-Tracking Abilities

One kind of AI that has become most popular, large language models (LLMs), are not based on an understanding of truth or a knowledge of the world. Rather, they make statistical inferences and probabilistic “guesses” to construct responses. Given the input and their training, they are designed to give plausible responses. But plausible responses are not necessarily truthful. Even when responses are false, they can still appear plausible and convincing. This can create risks including the creation of plausible misinformation, physical safety risks (e.g., LLMs giving incorrect medical advice), and libel and other speech-related risks. Speech is a contentious area of governance, but speech created by a machine that is in turn created by a company with data that is unclear whether it was acquired lawfully can make for a governance nightmare.

8.8 Privacy and Copyright

When companies scrape data off the internet, or collect data from their users, questions related to whether they have a claim to that data arise. There is the worry that the privacy of data subjects has been violated by collecting the personal data of millions of unsuspecting internet users. It is unclear, for example, whether LLMs can comply with Europe's General Data Protection Regulation, as European citizens are supposed to have a right to ask companies what data they have on them, to modify that data, and delete that data. It is far from clear that the companies that develop and sell access to these LLMs can comply with such data requests. Finally, there is the concern that copyright has been violated with LLMs ingesting material like books. At the time of writing there are various lawsuits in process related to these matters.

9.0 Regulatory Landscape

There is broad consensus that AI should be regulated, and the passage of AI-related laws in countries around the world bears this out. There is not, however, a commonly held view of what form such regulation should take. The remainder of this module takes a high-level look at the regulatory approaches and initiatives of three

governmental players likely to have an impact on how AI models are developed and deployed worldwide: Europe, the U.S. and China.

9.1 The Relationship Between Ethics and the Law

Ethics is considered a complement to law and necessary to ground, inform, and shape law. Societies tend to regulate behavior according to what they deem morally acceptable. Ethics helps one distinguish between just and unjust laws. Laws, however, are narrow in scope; they typically establish minimal requirements of behavior for social institutions to function well. Ethics goes beyond that—it identifies moral issues, reflects on the kind of society we want to live in based on ideas of what a good life looks like, and makes recommendations accordingly. Ethics, therefore, can be considered more ambitious than the law.

Think of the kind of people with whom you like to surround yourself. Would you be content with them being law-abiding citizens, or would you also like to be around people who are honest, reliable, and loyal? It is not against the law to abandon your friends when they need you the most, but it is certainly immoral. Laws allow us to have orderly interactions with one another within a framework of basic fairness. Ethics allows us to strive toward ways of life that will be most conducive to our own and others' wellbeing.

Even though AI ethics is gaining importance, there are some who feel that laws are also needed to govern AI. Some laws, like Europe's General Data Protection Regulation, were not designed to legislate AI, but are relevant for the design and implementation of AI.

9.2 Europe

9.2.1 GDPR – General Data Protection Regulation

The European GDPR was implemented in May 2018. It is designed to regulate *personal* data. According to the GDPR, personal data is any information concerning an identified or identifiable person. Under the GDPR, data subjects have a right to receive concise and transparent information about their data; access their personal data upon request; request erasure of their personal data; and object to processing of personal data from marketing or other purposes unrelated to the service being offered. The latter objection, however, does not apply when there is

a “legitimate interest” to carry out the service. What counts as a legitimate interest is the subject of much debate. Recent and upcoming rulings are gradually clarifying the matter.

The law also states that data controllers are under a legal obligation to notify within 72 hours the supervisory authority of any data breaches.

Part of what was significant about the GDPR is that it applies not only if the data controller (any organization collecting personal data) or processor (any organization processing data on behalf of a data controller) is located inside Europe, but also if it’s located outside the European Economic Area (EEA), if it offers a service to data subjects within the EEA. That *extraterritorial jurisdiction* has made the GDPR hugely effective across the world. It has made some international corporations improve their standards everywhere, because it is too complicated to have one system for European residents and a different one for the rest of the world. It also looks bad to have better standards for some users than others. The GDPR has also inspired some countries to come up with their own privacy legislation.

9.2.2 DMA – Digital Markets Act

The Digital Markets Act (DMA) is a landmark piece of legislation passed by the European Union in 2022 that aims to create a more fair and competitive digital marketspace. The DMA targets large online platforms with "gatekeeper" status, which are defined as companies that hold a dominant position in a market and have the ability to distort competition. **Key provisions of the DMA include:**

- **Ban on self-preferencing:** Gatekeepers are prohibited from favoring their own services or products over those of their competitors. This includes practices such as ranking their own products higher in search results or making it difficult for users to switch to rivals.
- **Open access to data:** Gatekeepers must provide access to their data to third-party developers and businesses, allowing them to create innovative services that compete with the gatekeepers' offerings.
- **Transparency obligations:** Gatekeepers must be transparent about their algorithms and practices, allowing users and regulators to understand how they operate and identify potential anti-competitive behavior.
- **No tying and bundling:** Gatekeepers cannot require users to purchase additional services or features that they don't want or need to access their core services.

- **Interoperability of messaging services:** Gatekeepers must ensure that their messaging services are interoperable with other messaging services, allowing users to communicate seamlessly across platforms.

9.2.3 DSA – Digital Services Act

The Digital Services Act (DSA) is a regulation in EU law that aims to update the Electronic Commerce Directive 2000 regarding illegal content, transparent advertising, and disinformation. It was adopted by the European Parliament and the Council of the European Union in October 2022 and came into force in 2023.

The DSA applies to online intermediaries and platforms, including marketplaces, social networks, content-sharing platforms, app stores, and online travel and accommodation platforms. It sets out obligations for these platforms to:

- **Prevent the dissemination of illegal content:** Platforms must proactively identify and remove illegal content, such as hate speech, child sexual abuse material, and counterfeit goods. They must also have clear and effective reporting mechanisms for users to flag illegal content.
- **Be more transparent about their content-moderation practices:** Platforms must publish clear information about their policies and procedures for identifying and removing illegal content. They must also provide users with access to their content-moderation data, allowing them to see how their content has been handled.
- **Address the issue of disinformation:** Platforms must take measures to prevent the spread of disinformation, such as providing clear labels on political advertising and promoting fact-checking initiatives.
- **Protect users from algorithmic bias:** Platforms must ensure that their algorithms are not biased against certain groups of users. They must also be transparent about how their algorithms work and how they are used to personalize user experiences.
- **Allow users to opt out of receiving personalized content:** Very large online platforms are required to allow their users to opt out of receiving personalized content — which typically relies on tracking and profiling user activity — when viewing content recommendations.

There are 19 companies currently beholden to the DSA's rules, including TikTok, YouTube, Instagram, and X.

9.2.4 The EU AI Act

On March 13, 2024, the EU Parliament approved the Artificial Intelligence Act, commonly known as the EU AI Act, with an aim to establish a common regulatory and legal framework for AI. On May 21, 2024, the Council of the European Union approved the AI Act, which is the final stage in the legislative process.

The AI Act has been designed to ensure proportionate risk mitigation over a range of AI functions. For instance, a company offering an AI service to screen job applicants would have to take steps to prevent their systems from unduly hurting individuals' access to opportunities. The regulation also imposes a legally binding requirement to notify people when they are interacting with a chatbot, biometric systems, or emotion recognition. Companies will also need to label deepfakes and content generated by AI, as well as design systems to make AI-generated media detectable.

Organizations like banks and insurance companies that offer essential services, and companies that deploy AI classified as high-risk, are obligated to do an impact assessment on how AI will affect people's rights. Providers of high-risk AI must also keep thorough records of the datasets used, programming and training methodologies, and measures taken for oversight.

The following systems are expected to be prohibited with just six months for companies to ensure compliance:

- Biometric categorization systems that use sensitive characteristics (e.g. political, religious, philosophical beliefs, sexual orientation, race);
- Untargeted scraping of facial images from the internet or CCTV footage to create facial recognition databases;
- Emotion recognition in the workplace and educational institutions;
- Social scoring based on social behavior or personal characteristics;
- AI systems that manipulate human behavior to circumvent their free will;
- AI used to exploit the vulnerabilities of people (due to their age, disability, social or economic situation).

Non-compliance can lead to substantial fines, ranging from €35 million or 7% of global turnover to €7.5 million or 1.5% of turnover, depending on the offense and company size.

The AI Act established the European AI Office that will be in charge of compliance, implementation, and enforcement. It is the first body in the world to enforce binding rules on AI. Much like the GDPR, this law likely will set new global standards. The AI Act will be effective on August 1, 2024. A ban on AI systems with unacceptable risk will be in place starting from February 2, 2025, and a majority of the AI Act's provisions will be enforced from August 2, 2026.

9.3 The United States

9.3.1 Privacy

The United States does not have a federal privacy law. However, it does have some laws that are relevant for privacy.

Health Insurance Portability and Accountability Act (HIPAA)

HIPAA was enacted in 1996 and protects the privacy and security of health information. It applies to health care providers, health plans, and other organizations that use or store electronic health information (EHI). HIPAA requires these organizations to implement safeguards to protect EHI from unauthorized access, use, disclosure, alteration, or destruction.

State-level Privacy Regulations

The United States has a patchwork of state privacy regulations that govern how businesses can collect, use, and share personal information about consumers. These regulations vary in scope and enforcement, but they are all designed to protect consumer privacy. See Appendix for a sampling of key U.S. state-level privacy regulations.

9.3.2 Cybersecurity

On May 12, 2021, President Biden issued an Executive Order (EO) on Improving the Nation's Cybersecurity. The EO is a comprehensive and ambitious plan to strengthen the cybersecurity of the United States against evolving threats.

Key provisions of the EO:

- Require the federal government to adopt and implement a zero-trust architecture for all federal networks and systems.
- Increase the security of software supply chains and promote the use of open-source software.
- Enhance the cybersecurity of critical infrastructure, such as energy, transportation, and healthcare.
- Expand public-private partnerships to improve cybersecurity.
- Invest in education and training for cybersecurity professionals.

9.3.3 AI

The United States does not have any federal laws related to AI, however both the White House and several federal agencies have been actively working on the development of guidelines for the development, deployment and use of AI systems.

Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence

On Oct. 30, 2023, President Biden issued an EO on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence. The order is a comprehensive plan to address the national security, economic, and ethical challenges posed by AI.

Key provisions of the EO:

- Promote the development and use of trustworthy and reliable AI that is aligned with U.S. values.
- Address the risks of AI-enabled harms, such as biases, algorithmic decision-making, and cybersecurity threats.
- Enhance international cooperation on AI governance.
- Invest in research and development to advance AI safety, security, and reliability.
- Accelerate the hiring of AI professionals.
- Require that developers of powerful AI systems share their safety test results and other critical information with the U.S. government.

National Institute of Standards and Technology (NIST)'s AI Risk Management Framework (AIRMF)

The NIST developed the AI RMF in 2023. This voluntary framework serves as a guide for organizations designing, developing, deploying, or using AI systems. Its overarching goal is to promote the responsible and trustworthy development of AI while mitigating potential harms.

The AI RMF focuses on four key functions: Govern, map, measure, and manage. Through these functions, organizations can build governance structures; identify and map AI risks; measure and assess those risks; and implement appropriate mitigation strategies. The framework emphasizes flexibility and adaptability, catering to organizations of various sizes and across different sectors. Its non-prescriptive approach allows customization based on specific AI use cases and risk profiles.

AI Safety and Security Board

Established by the U.S. Department of Homeland Security on April 26, 2024, the Artificial Intelligence Safety and Security Board (AISSB) advises the Secretary, the critical infrastructure community, other private sector stakeholders, and the broader public on the safe, secure, and responsible development and deployment of AI technology in our nation's critical infrastructure. The AI Board will develop recommendations to help critical infrastructure stakeholders, such as transportation service providers, pipeline and power grid operators, and internet service providers, more responsibly leverage AI technologies. It will also develop recommendations to prevent and prepare for AI-related disruptions to critical services that impact national or economic security, public health, or safety.

9.4 China

China has been rapidly developing and implementing regulations related to AI, covering technology, cybersecurity, privacy, and intellectual property. One significant regulation is the Provisional Administrative Measures of Generative Artificial Intelligence Services (Generative AI Measures), which were published by the Cyberspace Administration of China (CAC) and have taken effect since Aug. 15, 2023. These measures apply to the use of generative AI technology to provide services for generating text, pictures, sounds, videos, and other content within the territory of China. They impose various obligations on generative AI service providers, including the prohibition of generating illegal content, taking measures to prevent the generation of discriminatory content, and not infringing on others' rights, including privacy rights and personal information rights.

China has enacted several comprehensive laws aimed at protecting personal information, like the Personal Information Protection Law (PIPL) and the Internet Information Service Algorithmic Recommendation Management Provisions. These regulations mandate data minimization, user consent, and transparency in algorithm decision-making.

Appendix – Key U.S. State-Level Privacy Regulations

California Consumer Privacy Act (CCPA)

The CCPA is the most comprehensive state privacy law in the United States. It was enacted in 2018 and applies to businesses that do business in California and that collect or control the personal information of over 50 million Californians. The CCPA gives consumers the following rights:

- The right to know what personal information is being collected about them.
- The right to access and delete their personal information.
- The right to opt out of the sale of their personal information.
- The right to non-discrimination.

The CCPA also requires businesses to provide clear and conspicuous notice of their privacy practices and to obtain consumers' consent before collecting certain types of personal information.

Virginia Consumer Data Protection Act (CDPA)

The CDPA is modeled after the CCPA and was signed in 2021. It applies to businesses that do business in Virginia and that control or process the personal information of at least 100,000 Virginia consumers or 25,000 Virginia consumers and earn revenue of at least \$25 million from the sale of personal information. The CDPA gives consumers many of the same rights as the CCPA, including the right to know, the right to access, the right to delete, the right to opt out of the sale of personal information, and the right to non-discrimination. The CDPA also includes additional protections for sensitive data, such as genetic and biometric data.

Colorado Privacy Act (CPA)

The CPA was enacted in 2021 and applies to businesses that do business in Colorado and that control or process the personal information of at least 100,000 Colorado consumers. The CPA gives consumers the right to know, the right to access, the right to delete, the right to opt out of the sale of personal information, and the right to opt out of the use of their personal information for targeted advertising. The CPA also includes provisions for data minimization, security, and transparency.

Illinois has two major privacy laws that govern how businesses can collect, use, and share personal information about consumers:

Biometric Information Privacy Act (BIPA)

The BIPA was enacted in 2008 and applies to businesses that collect, use, or disclose biometric identifiers or biometric information of Illinois residents. Biometric identifiers are biological characteristics of an individual, such as fingerprints, facial patterns, or voiceprints. Biometric information is information derived from these identifiers, such as a fingerprint template or a stored facial image.

The BIPA gives Illinois residents the following rights with respect to their biometric information:

- **Notification:** Businesses must provide individuals with clear and conspicuous notice before collecting their biometric identifiers or biometric information. This notice must explain the purpose of collecting the information, how it will be used, and how it will be protected.
- **Access:** Individuals have the right to access their biometric information and to obtain a copy of it. They also have the right to request that businesses correct any inaccuracies in their biometric information.
- **Deletion:** Individuals have the right to request that businesses delete their biometric information. Businesses are only permitted to retain biometric information for as long as necessary to fulfill the purpose for which it was collected.
- **Notice of Breach:** Businesses must notify individuals and the Illinois Attorney General within 45 days of a breach of their biometric information. This notice must describe the breach in detail, including the types of biometric information that were breached, the number of individuals affected, and the steps that the business is taking to mitigate the risk of harm.
- **Civil Remedies:** Individuals who have suffered harm as a result of a violation of the BIPA can sue for damages, including actual damages, statutory damages of \$1,000 per violation (and \$5,000 if the violation is intentional or reckless), or both.

Personal Information Protection Act (PIPA)

The PIPA was enacted in 2005 and applies to businesses that own or license, or maintain or store but do not own or license, records that contain personal information concerning an Illinois resident. Personal information is defined broadly to include any information that can be used to identify an individual, such as name, address, phone number, email address, social security number, driver's license number, or credit card number.

The PIPA requires businesses to:

- **Implement and maintain reasonable security measures to protect personal information from unauthorized access, acquisition, destruction, use, modification, or disclosure.**
- **Notify individuals and the Illinois Attorney General within 24 hours of a breach of personal information that could result in identity theft or fraud.**
- **Provide individuals with a notice of privacy practices that describes how the business collects, uses, and shares personal information.**
- **Provide individuals with access to their personal information and the opportunity to correct any inaccuracies in it.**
- **Allow individuals to opt out of the sale of their personal information.**

The PIPA also includes provisions for data minimization, transparency, and enforcement.

Responsible & Ethical AI: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

1. Consequentialism, deontology, and virtue ethics are mutually exclusive.

A. True

B. False

False. In pluralistic societies it is beneficial to take into account different ethical theories.

2. What are five principles common to many AI ethics codes?

- Nonmaleficence
- Beneficence
- Justice

- Autonomy
- Explainability

3. What is the difference between consequentialism, deontology, and virtue ethics?

Consequentialism is an ethical theory that judges the morality of an action based on the consequences of that action. Deontology is an ethical framework that judges the morality of actions based on adherence to ethical duties and rules rather than focusing on consequences. Virtue ethics emphasizes virtuous character traits and living a good life, rather than rules or consequences. It has roots in ancient Greek philosophers like Aristotle, who taught that happiness comes from living a life guided by virtues.

4. What are some best practices surrounding privacy?

- Data minimization
- Implementing the right to be forgotten
- Giving control to data subjects
- Respecting contextual integrity
- Routine personal data deletion
- Having strong data security
- Following the provisions of GDPR.

5. What is practical ethics?

The application of ethical theory and the development of good practices to solve real-world moral dilemmas.

6. What are some reasons why a company might consider a practical ethics framework?

- Building trust and reputation
- Avoiding scandals
- Attracting and retaining talent
- Strengthening culture
- Supporting risk management

- Encouraging and guiding innovation
- Promoting long-term thinking.

7. Fairness can be automated.

A. True

B. False

False. Because it's mathematically impossible to satisfy operationalizations of fairness when base rates are unequal.

8. When does bias count as discrimination?

Whether bias results in discrimination in a legal sense may vary depending on jurisdiction. In general, algorithmic bias is likely to lead to discrimination when it results in disfavoring people based on their race, sex, ethnicity, age, or any other classification protected by law. Such disadvantages typically violate legal protections, and designers, developers, deployers, supervisors of automated systems, and risk managers have an interest in taking proactive and continuous measures to protect against it.

9. What are some factors that can make AI difficult or challenging to govern?

- Power asymmetries
- Institutional and algorithmic opaqueness
- Lack of ethics structures
- Lack of AI ethics structures
- Lack of regulation
- Unpredictability
- Lack of truth-tracking abilities
- Privacy and copyright

10. What are some actions a company can take to avoid problematic biases?

- Use technological solutions

- Ensure data quality
- Audit algorithms
- Be inclusive
- Establish ethics structures

11. What are four common sources of algorithmic bias?

- Biases in problem specification
- Biases in data
- Biases in modeling, validation and algorithmic design
- Biases in deployment

12. Differentiate between nonmaleficence and beneficence.

The principle of nonmaleficence asserts an obligation to avoid harming others or inflicting injuries. The principle of beneficence refers to the moral obligation to act for the benefit of others. Beneficence requires taking positive steps to help others, rather than simply refraining from harm. It moves beyond nonmaleficence, which tells us not to injure others, and commands us to advance the welfare and legitimate interests of people in need.

13. Describe explainability in the context of AI.

The ethical principle of explainability (sometimes called explicability) has gained significant attention in the context of AI. It refers to the idea that AI systems, especially those with decision-making capabilities, should provide transparent and understandable explanations for their actions or decisions.

14. Discuss fairness in the context of an AI algorithm.

Fairness entails the absence of bias or preference toward an individual or group based on irrelevant characteristics, such as their race. An algorithm is considered fair when it does not exhibit problematic biases. There are two primary types of fairness: group and individual fairness. Group fairness involves statistical criteria, where, for example, in loan distribution, statistical parity would require the demographics of approved individuals mirror the overall population (if 51% of the population is female, 51% of loan recipients should be women). On the other hand, individual fairness emphasizes treating similar individuals similarly, even though defining similarity can pose challenges.

15. Discuss autonomy in the context of AI.

Autonomy refers to the capacity of people to make their own informed, un-coerced decisions about their lives and actions. As an ethical principle, autonomy commands respecting and supporting others' abilities to determine their own course in life. An ethical AI system respects people's autonomy by not using coercive or manipulative tactics to get people to act in a particular way.

Module 5: Data and AI Model Governance

Learning Objectives

This module discusses data and model governance and provides a starting point to establish a firm-specific model validation framework across the entire AI/ML model life cycle — from model development through performance monitoring and decommissioning. The principles presented apply to a wide range of industries, but the primary focus is on the financial sector, and the quantitative risk models (QRMs) heavily relied upon and subject to formal regulatory oversight. The opacity of AI/ML models is also discussed, along with the need for proper governance of the data used to train these models.

After completing this module, you should be able to:

- Describe elements of a data governance framework.
- Describe elements of a model governance framework.
- Describe steps in the model development and testing process.
- Discuss model validation and its importance.
- Discuss policies and procedures related to model governance.
- Describe factors to be considered when registering AI/ML applications in a model inventory.
- Discuss roles and responsibilities associated with model risk management.
- Describe how the model review framework differs for AI/ML models.

- Describe the steps involved in model implementation and adaptation.
- Discuss potential sources of misinterpretation of model results.

1.0 Introduction

This module discusses data and model governance and provides a starting point to establish a firm-specific model validation framework for artificial intelligence and machine learning (AI/ML). In what follows, the primary focus is on the financial sector, and specifically quantitative risk models (QRMs), due to the established formal regulatory framework around QRMs.¹ Nevertheless, the overarching principles presented should prove valuable and applicable to a wide range of industries.

Sound governance of models is essential for effective and prudent risk management, regardless of the types of models employed. Without it, a range of outcomes and decisions may occur that can prove detrimental. Therefore, model governance should exist across the entire model lifecycle – from model development through performance monitoring and ultimately decommissioning.

Compared with more traditional models, sound governance of AI/ML models may be even more important as it helps ensure that technological advances do not overshadow the need for fundamental and necessary challenges to applicability, theory, benchmarking, and outcomes analysis. The opacity of AI/ML models makes the proper governance of the data used to train these models particularly relevant, and this topic is explored in the next section.

To make this module more accessible to a general audience, the basic principles of model governance are studied together with some relevant issues concerning AI/ML techniques.

¹ 15-01.pdf (upenn.edu)

The Fed - Supervisory Letter SR 11-7 on guidance on Model Risk Management — April 4, 2011 (federalreserve.gov)

2.0 Data Governance

The Data Governance Institute defines data governance as “a system of decision rights and accountabilities for information-related processes, executed according to agreed-upon models which describe who can take what actions with what information, and when, under what circumstances, using what methods.”² Broadly speaking, it encompasses the people, processes, and technologies required to manage and protect data assets.

² <https://datagovernance.com/>

2.1 Data Strategy: Developing Vision, and Setting Goals and Priorities

Data procurement and use requires a clear and approved data-governance policy that is strategic and outlines operational practices. This should include a level of authorization for non-publicly available personal information, including the specification of whether the data is for single or recurring use, and the source of the information.

The use of alternative data may require additional controls and testing due to concerns about privacy, data quality and accuracy, and other potential risks, as well as the general nature of unbounded data streams. Data applicability warrants an evaluation and approval process. Data treatment and relevant controls (e.g., procedures for handling missing data, trend monitoring, the use of proxies, etc.) should also be well specified, documented, and approved by the data governance authority, such as a data governance committee. A data governance committee is typically responsible for a range of approved data practices and policies, including data collection, reconciliations, treatment, access, monitoring, and management of data stewards. Complete, relevant, transparent, and accurate data should be the primary objective.

2.2 Data Quality: Accuracy, Consistency, Integrity

Model risk management requirements around data quality do not change with AI/ML algorithms. However, the use of alternative data sources, while increasingly common with AI/ML implementations, warrants additional scrutiny.

Verifying the accuracy, consistency, and integrity of alternative data sources may be quite difficult, as the data provider(s) may change field definitions in the historic time series or after deployment. Even well-established data sources are subject to this risk.

For example, during the 2020 pandemic, the U.S. Bureau of Labor Statistics stated in its footnotes that its published unemployment rate information was biased relative to historic measures because of an inability to conduct the survey on which the data was based properly. Such anomalies may occur with any data stream, but new data sources introduce additional risks.

The pandemic also demonstrated that data sources can suddenly disappear, have reporting biases, or exhibit previously unimagined shocks. Therefore, a key part of model monitoring is input data monitoring to alert model owners of flawed or otherwise problematic inputs. As in the case of the biases in the pandemic unemployment rate data, the data flaws may not be apparent from the data series alone, and outlier detection may be insufficient to spot anomalies, particularly when definitional changes are only disclosed in the footnotes of the data report.

2.4 Data Classification: Structure and Confidentiality

Data will be either quantitative or qualitative, such that it is either measured or determined numerically, or determined non-numerically, as with characteristics such as gender and ethnicity. Furthermore, these may be observed, projected, or based purely on judgment. What is critical for good governance is to distinguish the classification of data being used, and ensure access and use is permissible. When data is either confidential or personally identifiable, controls must be present, operational, and effective

2.5 Metadata Management: Collection, Documentation, and Management, and its Relation to Origin, Structure, Definitions, and Relationships to Other Data

Metadata gives basic information about data, including file type, time of creation, data author, data source, file size, and other relevant information. There are several distinct types of metadata - including descriptive, structural, administrative, reference, and statistical metadata – that all provide unique information about the data used for modeling.

A robust metadata management strategy should aim to ensure data is high-quality, consistent, and accurate across various systems. Data documentation, data mapping, data dictionary, data definitions, data process flows, data relationships to other data, and data structures are essential for robust metadata management. The use of a comprehensive metadata-management strategy should enable better-informed business decisions, which is an important objective of any data governance initiative.

2.6 Data Protection, Security, and Compliance: Regulatory Aspects and Overview

Data security is often used interchangeably with data protection and data integrity. There are several regulatory requirements in safeguarding and protecting data. The strategic security process, data protection plan, and procedural steps identified to safeguard the availability of data, access to data, and data privacy are all part of data security. A solid data-protection strategy should be in place for safeguarding important information from corruption, malicious or accidental damage, compromise, or loss. The importance of data protection increases with the amount of data created and stored. A data retention policy should also be in place and adhered to.

2.7 Data Access: Permissions and Secure and Effective Data Sharing

Ensuring that data is readily accessible and shareable to those who have the requisite permission and need to access it is the most important aspect of data access. Data security, data protection, data permissions, and data access controls are all part of the data-governance process.

Multiple regulations exist requiring organizations to implement appropriate security measures, including encryption, to safeguard data. By encrypting data, organizations can ensure its confidentiality and integrity, thereby helping to maintain compliance. The following are US-based examples:

- The **Safeguards Rule** requires financial institutions under FTC jurisdiction to have measures in place to keep customer information secure.
- The **Gramm-Leach-Bliley Act (GLBA)** requires financial institutions to provide customers with information about the institutions' privacy practices and about their opt-out rights, and to implement security safeguards for customer information.
- The **Privacy Act of 1974** governs how federal agencies can collect and use data about individuals in its system of records. The act prohibits agencies from disclosing personal information without written consent from the individual, subject to limited exceptions including the Census Bureau for statistical purposes.

2.8 Compliance: Ensuring Compliance With Legal and Regulatory Requirements When Handling Data

Although there are many rules within current regulations, the majority can be boiled down to three basic principles: obtaining consent, minimizing the amount of data you hold, and ensuring the rights of data subjects.

2.9 Roles and Responsibilities

A company's board of directors plays a crucial role in overseeing a firm's data governance framework, and ensuring that the framework aligns with the organization's strategic objectives, risk management policies, and compliance requirements. Among other responsibilities, the board provides approval of the overall data governance framework and policies. The policies should include clear guidelines on data quality, privacy, security, and compliance with relevant regulations. The data governance framework serves as the foundation for how data is collected, stored, used, and shared within the organization.

The board is also responsible for oversight of compliance and risk management, ensuring that the organization's data-governance practices comply with legal, regulatory, and ethical standards. This includes overseeing compliance with data-protection laws (e.g., GDPR), industry standards, and internal policies. The board also assesses and manages risks related to data breaches, data quality issues, and misuse of data. The board further ensures that the data-governance framework is regularly reviewed and updated to adapt to changing business needs, technologies, and regulatory requirements.

Ultimately, the board's involvement in data governance is critical to ensure that data is treated as a strategic asset, managed responsibly, and used to enhance decision-making, operational efficiency, and compliance. By providing oversight and setting the tone at the top, the board can help foster a strong data-governance culture that supports the organization's long-term success.

3.0 Model Governance



Play Video

Before delving into the intricacies of risk model validation, let us first explore how risk (or uncertainty) can be conceptualized and how this understanding can inform the design of QRMs. It is important to note that the concept of risk contains subjective elements in its perception and modeling. Quantitative risk modeling encompasses three main elements:

1. **Quantity of interest:** A numerical object whose future value, referring to a specific point in time (the risk horizon), is uncertain. Examples include the value of a portfolio in ten business days, the revenue projection for a business over the next five years, or the expected number of new clients by the end of the current calendar year.
2. **Potential future scenarios:** These scenarios represent possible values of the quantity of interest. They depict potential future outcomes, such as the value of a portfolio in ten business days conditioned on a specific investment decision, the revenue projection for a business over the next five years after implementing changes to the business model, or the expected number of new clients by the end of the current calendar year following the latest advertising campaign. To facilitate quantitative analysis, each potential scenario is assigned a weight (probability), indicating its relative importance compared to other scenarios.
3. **Risk measure:** This summarizes the essential information derived from analyzing the potential future scenarios. Examples include evaluating the value of a portfolio in ten business days using value at risk (VaR) or expected shortfall (ES), assessing the revenue projection for a business over the next five years corresponding to the scenario with the highest weight, or determining the average number of new clients by the end of the current calendar year across all scenarios.

Even the most basic statistical risk measures can be useful within the context of QRMs. Often these statistical measures are then brought back in to quantity of interest (e.g., estimation of volatility which is then used as an input or risk factor of the risk model). Risk measures are frequently presented and monitored via dashboards or discussed periodically at stakeholder meetings and are crucial for evaluating potential losses, model performance, making informed decisions, managing risks effectively, and complying with regulatory requirements.

In summary, risk models offer a structured approach to envision the future through scenario analysis. However, the effectiveness of this approach depends on the quality of each element. Some important considerations include:

- **Completeness of scenario sets.** It is challenging to anticipate every potential future scenario, especially regarding rare events. Historical data may not fully reflect these events, and capturing them through expert judgment can be difficult.
- **Feedback effects.** The presence of feedback can complicate matters as scenarios and subsequent decisions may influence the behavior of other market participants. As a result, scenario sets and their weights may need ongoing updates.
- **Communication of results.** Reports on QRMs should provide a summary of the main assumptions used, ensuring transparency and avoiding complacency, and should reflect perceived risk based on perceived uncertainty and exposure.

The Federal Reserve's Supervisory Guide on Model Risk Management⁶ (SR 11-7) from 2011 established some regulatory standards concerning model governance. Although general in nature, these standards are relevant to AI/ML models. In 2021 the Office of the Comptroller of the Currency (OCC) published a Handbook on Model Risk Management⁷, which includes some AI/ML related perspectives.

⁶ <https://www.federalreserve.gov/supervisionreg/srletters/sr1107a1.pdf>

⁷ <https://www.occ.treas.gov/publications-and-resources/publications/comptrollers-handbook/files/model-risk-management/pub-ch-model-risk.pdf>

3.1 Model Development and Testing

The model-development process can vary depending on the organization, the specific type of model being developed, and the intended purpose of the model. Institutions may decide to develop their models in-house, rely on vendor models, or a mix of the two approaches. A general overview of the typical steps involved in the model development process is presented below.

- **Define objectives and scope.** Clearly define the objectives and scope of the model. Determine the specific problem or risk being addressed, the applicable products, the required data, desired outcomes, and the target audience for the model's outputs.
- **Data collection and preprocessing.** Gather relevant data needed for model development. This may involve identifying and retrieving appropriate data sources, cleaning and transforming the data, handling missing values or outliers, and ensuring data quality and consistency. Whether the models are AI/ML or traditional, document any data excluded along with the rationale. With AI/ML models, automation may

lead to exclusion of data that may contain explanatory power and, therefore, should be examined by a human before proceeding.

- **Exploratory data analysis.** Conduct exploratory data analysis to gain insights into the data, identify patterns or relationships, and understand any limitations or biases present. This is a key step that helps inform the subsequent modeling approach and feature selection. Graphical packages for visualization, summary statistics, identification of missing data, potential data leakage, and so on need to be considered up front. Certain models require data scaling and normalization, so any data transformations should also be tested and documented.
- **Feature engineering.** Select and engineer the appropriate features or variables that will be used as inputs for the model. This could involve transforming or combining variables, creating new derived features, or performing dimensionality-reduction techniques.

Sometimes the algorithm will identify key features that a human may not have detected. In this case, the features recommended by the algorithm should be carefully inspected rather than pre-imposing features on the model. The human user can then review the feature importance ranking, and use his domain knowledge to decide which features should be included. In addition to experience, there are several quantitative tools to support feature selection (e.g., correlation, ranking criteria, and dimensionality reduction).⁸

- **Model selection.** Choose the most suitable modeling technique based on the objectives, data characteristics, and available resources. This may involve selecting from various statistical models (e.g., linear regression, Monte Carlo simulation, etc.) or other techniques like time series analysis or machine learning algorithms. If AI/ML models are used for econometric or time-series modeling, always run a traditional model alongside to benchmark results and ensure the expected parameter signs, weights, and other key model components are as expected. Be sure to test a range of outcomes to determine the boundaries/limits of the model and document any assumptions.

Appropriate model performance metrics should also be defined at this point, whether this is root mean square error (RMSE), mean absolute error (MAE), or some other. Backtesting procedures and frequency must also be defined and all associated assumptions should be documented.

- **Model training / model calibration.** Train the selected model or calibrate the parameters using the preprocessed data. This typically involves optimizing the model's parameters or hyperparameters to best fit the data and minimize errors or loss functions. Special emphasis should also be put on training

with data leakage (using future or concurrent data that wouldn't have been observable at the point of the training data). Use accepted techniques such as cross-validation.

Regarding model development, SR11-7 offers the following guidance:

"A sound development process includes: a clear statement of purpose to ensure that the model is developed in line with its intended use; sound design, theory, and logic underlying the model; robust model methodologies and processing components; rigorous assessment of data quality and relevance; and appropriate documentation. An integral part of model development is testing, in which the various components of a model and its overall functioning are evaluated to show the model is performing as intended; to demonstrate that it is accurate, robust, and stable; and to evaluate its limitations and assumptions. Importantly, organizations should ensure that the development of the more judgmental and qualitative aspects of their models is also sound."

⁸ Guyon, I., and Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3, 1157-1182.

3.1.1 Who is Responsible for Testing QRMs?

A vital question pertains to the individuals involved in testing QRMs. Naturally, programmers bear the responsibility of debugging and resolving known bugs, many of which may have already been identified during the development phase. Testers, who possess expertise and qualifications in software testing, are entrusted with the task of discovering any remaining significant bugs. "Good testers think technically, creatively, critically, and practically," and choose their testing tool "based on the software under examination."⁹ Given the need for quantitative skills, it is common to "rely partially on external testers to offer a fresh perspective and ensure comprehensive testing."¹⁰ However, it is crucial to avoid exploiting end users of a QRM for testing purposes, as their frustration with buggy software might impede their ability to provide useful bug reports. In many companies, though, the software developer must also bear responsibility for developing use and test cases; these would not always be different people. Good code design and documentation are key.

Testing should commence as early as possible, as identifying bugs early usually facilitates easier resolutions.

Traditionally, testing follows a chronological order encompassing specific stages:

- **Unit tests** validate that each piece of the code performs as designed.

- **Component tests** verify the functionality of individual code sections (e.g., the QRM's core can be tested independently).
- **Integration tests** verify the interfaces between components (e.g., between the QRM's core and its input processing components).
- **System tests** ensure that the completely integrated modeling system meets its requirements. This process may be repeated on a higher level as a system integration test.
- **Performance tests** assess the speed, responsiveness, and stability of the model or application under various conditions.
- **Regression tests** verify that the system continues to function after modifications to some of its components.
- **Acceptance tests** determine whether the model or application meets the agreed-upon requirements and is ready for deployment.

Overall, testing is not a continuous activity but is conducted on demand. However, due to changes in the QRM, its implementation, or testing requirements, the demand for testing will persist. In addition, because it is unlikely a model will anticipate every action a user will take, it is important to release in beta versions or parallel to production for some time to allow any additional and obvious bugs to surface.

⁹ Lessons Learned in Software Testing: A Context-Driven Approach 1st Edition by Cem Kaner (Author), James Bach (Author), Bret Pettichord (Author)

¹⁰ In SR 11-7, a guiding principle is "effective challenge" where both the users of the model as well as external validators provide critical and objective feedback on the model

3.1.2 How are “White-Box” and “Black-Box” Tests Conducted?

Tests could be categorized as white-box testing or black-box testing. White-box testing involves testers having access to internal data structures, algorithms, and the actual code. It may include line-by-line proofreading of the code. On the other hand, black-box testing treats the software as a closed box, without any knowledge of its internal implementation (partial knowledge is referred to as grey box testing). Employing both approaches can be beneficial, as white-box testing is considered more effective, whereas black-box testing reduces the likelihood of bias. Note that the types of tests that can be conducted may depend on whether the model was developed in-house or externally. Many proprietary vendor models are black box.

In general, tests need to be appropriately configured, and their usefulness hinges on the adequacy and sufficiency of these configurations. Testers should not limit their configurations solely to artificial cases, as this may render their testing irrelevant in practice. Sometimes, the most valuable test configurations emerge from real-world situations. For instance, if the model implementation has reached a prototypical state where parameters and input data can be fed into it, establishing a preliminary process that automatically generates test results from available data is recommended. The less realistic the parameters and input data (e.g., missing trades, excessive trades, incorrect scales or mappings, extreme parameter values, parameters estimated from insufficient data, data provided by inexperienced users, variations in compilers or hardware), the better. Documenting the experience gained from these tests is invaluable.

3.1.3 What is the Use Test?

The validation tools discussed earlier focus on assessing a QRM from an external perspective. In contrast, the **use** test examines the QRM within its context, considering human interaction, actual usage, acceptance of the model, interpretation of results, and the application of those results. The use test is closely intertwined with user testing, but its implications go beyond that. It serves as an ongoing validation tool, which may not be initiated until the QRM has been in use for a considerable period. The use test is qualitative in nature, and it is difficult to follow a schematic treatment. It represents a validation ideal rather than a specific tool. In essence, the use test evaluates adherence to a foundational principle.

Unlike other quantitatively oriented validation activities, the use test revolves around people rather than numbers. It focuses on recording and conveying credible stories of acceptance, interpretation, and application of the model, whether they are stories of success or failure. Technical and process-related challenges usually take a back seat. The results of QRMs are present during decision-making processes, but the crucial questions are whether they are utilized appropriately, whether they prove useful, and if not, why. The use test should explore factors such as acceptance, trust, comprehensibility of results, and communication of feedback to modelers.

The results of the use test are typically presented to senior management rather than documented as technical reports or spreadsheets. Consequently, it is often easier to communicate the results of a use test, minimizing the

risk of it being regarded as an arduous requirement. Furthermore, the use test can contribute to improving the culture of risk modeling and risk management.

Overall, testing QRMs requires a collaborative effort between programmers and dedicated testers, early initiation of testing, a combination of white-box and black-box testing approaches, appropriate test configurations, and a focus on user-oriented testing. Additionally, the use test offers valuable insights into the acceptance, interpretation, and application of the model, contributing to an improved risk modeling and risk management culture.

3.2.1 Model Validation: What is Model Validation and When Should it be Performed?

Model validation is a critical component in identifying model risk, involving qualitative and quantitative aspects. Model validation follows a lifecycle starting with the identification of the model. Once a model is identified, it is inventoried and scheduled for an initial baseline validation, which occurs prior to implementation and usage. Once the model is in production, routine periodic validations occur to ensure that the model continues to perform as expected. These can include annual reviews and more in-depth periodic baseline revalidations. A change-based validation will be triggered if the model owner makes a material change. Ultimately, the model may be retired, in which case it should be stored in a retired model inventory and then decommissioned. Back testing and performance monitoring occur throughout the model's production usage. The model validation effort culminates in a validation report and a rating of whether the model has passed validation. Findings or issues that need to be addressed by the model developers and/or owners may also result from the validation.

The frequency and intensity of validation activities should be determined based on the risk ranking of the model. For example, a bank's high-risk models may have a periodic baseline revalidation every two years, whereas lower-risk models may be on a three- or four-year frequency, and so on.

Model validation generally includes but is not limited to the following activities:

- Assessment of appropriateness of data sources and data analysis (use of proxies, whether the time series lengths are sufficient to draw conclusions on and so on)
- Assessment of any model assumptions and parameters
- Identification of model limitations, strengths, and weaknesses
- Identification of the approved products and users
- Review of appropriateness of implementation platform
- Review of usage of expert judgment and/or management overlays
- Review of the model conceptual soundness as well as assessment of any benchmarks and best practices
- Analysis and appropriateness of ongoing performance-monitoring plans including frequency, key risk indicators, and escalation plans in the event of breaches
- Appropriateness and adequacy of key risk indicators
- Review the quality of model documentation
- Review the documentation to ensure that backup plans exist for both internal and external models
- Review controls around model implementation and usage
- Review model implementation and access controls
- Review outcomes analysis, including the appropriateness of the dependent variable and how the model supports the business objectives
- Review of stress and scenario tests of model boundaries
- Review of parameter stability and so forth

Backtesting needs to be performed at an appropriate cadence with the model usage. For example, a VaR model requires daily monitoring along with frequent backtesting to detect potential issues early on; a credit risk model may undergo testing monthly; and an asset-liability management model may be tested quarterly.

It is useful to differentiate between initial validation—conducted during or after the design and implementation of the model—and ongoing validation, performed periodically or on demand throughout the model's lifespan.

Initial validation serves three key goals:

1. Ensuring that the model's operational feasibility has been checked, and that the model can run as intended without technical malfunctions
2. Ensuring that the model is properly documented, adheres to firm-wide standards (including but not limited to model development standards, documentation standards, implementation standards, model monitoring standards, and third-party governance standards), and includes executive summaries for essential documents
3. Ensuring that model users receive sufficient training to interpret and utilize the model's results effectively

Initial validation requires a preproduction phase to run the model on its intended schedule. Proper documentation is essential for validating activities, especially when engaging with external parties such as auditors or regulators. The results of initial validation form the basis for the model's official approval (sign-off), emphasizing the importance of conducting thorough initial validation. Assumptions need to be justified and limitations and weaknesses of the model must be documented.

The primary objective of **ongoing or periodic validation** is to observe whether the model remains aligned with its intended purpose, the assumptions remain valid, the data are still appropriate, and performance monitoring indicates that the model continues to perform as expected. In addition, model methodology should be reassessed periodically to ensure that it is still in line with best practices and reflects the real world. The real world consistently challenges model assumptions, requiring ongoing validation to assess whether the original assumptions remain valid. Ongoing validation involves an iterative process between the modeling and validation cycles, adapting to changes in the model and repeating successful validation activities when necessary. Ongoing validation could also be supported using benchmark or challenger models.¹¹ Like initial validation, ongoing validation must be well documented. It is often convenient to consolidate validation results within a single document to support a comprehensive view. Regular validation, following a fixed agenda with on-demand activities, is typically prescribed by regulatory authorities.

¹¹ An example of a challenger model for measuring market risk is described in a case study in Module 6 of this course.

3.2.2 Should There Be Limits on Model Use?

Every quantitative risk model is based on underlying assumptions that should be clearly stated in the documentation and communicated to model users. However, it is important to identify model limitations and weaknesses and to establish limits on the usage of models to prevent misuse. This can include product applicability, model access restrictions, and so forth. Rigorous assessment of model usage through the model inventory and model landscape processes can help identify instances in which a model is being used beyond its intended domain. For example, if a bank has approved a model for pricing standard interest rate swaptions, it should not be used for a Bermudan swaption without additional validation and approval.

Imposing portfolio size limits can also be valuable, particularly when new markets or instruments lack sufficient data or experience within the institution. Limits on model use should be mentioned in the model inventory (see next section). This is to ensure that the model risk ranking does not drift above the understood ranking (for example, what is initially classified as a low-risk model due to small portfolio size may become a high-risk model if the portfolio grows significantly).

The Fed's SR 11-7 established regulatory guidance for model risk management. Nevertheless, each institution needs to adapt these general guidelines to its specific situation. Activities in this domain depend on the specific models, the size and structure of the financial institution, the established governance framework, regulatory requirements, budget constraints, and the preferences of the individuals involved. Three general guidelines can support management, modelers, and validators in establishing a risk modeling culture:

Awareness: Be aware of the limitations and assumptions of risk modeling. Understand your company's history with QRMs, the risks they entail, and the validation processes in place. Stay informed about market practices. Recognize that the world is constantly changing.

Transparency: Transparently communicate the assumptions, limitations, and documentation of QRMs. Provide detailed documentation with executive summaries. Document the decision-making process during model development and all validation activities, including unsuccessful attempts. Engage in open communication with end users.

Experience: Learn from past modeling endeavors and apply relevant lessons. Emphasize proper project management and develop prototypes early on. Collect data and continuously improve quantitative skills. Establish

and maintain libraries of reusable code. Seek input from other modelers and consider external experts for validation activities.

By adhering to these guidelines, organizations can foster a modeling culture that views validation as an opportunity for insight and value creation, rather than a mere regulatory obligation.

3.3.1 Model Governance Policies

Model risk frameworks rely on written policies. The model risk function should provide clear definitions and guidelines addressing key issues such as:

- Defining what constitutes a model and where the distinction lies between an end-user tool or non-model (e.g., a "simple spreadsheet") and a model. A model is often described as a simplified representation or abstraction of reality, designed to understand, analyze, or predict aspects of the real world. Whether a calculation constitutes a model can be a gray area, but the presence of uncertainty may indicate that the calculation rises to the level of being a model. This makes consistent adherence to a robust model-risk framework even more important.
- Defining model risk and agreeing on a definition for internal use. The definitions should align with regulatory requirements.
- Identification and specification of a model's risk tier as this drives the detail and frequency of model validation activities.
- Establishing a definition for model validation and setting expectations for the validation process.
- Specifying the information that model owners should provide to enable model-risk assessment.
- Documenting the institution's model-risk appetite and outlining procedures to address unacceptable levels of model risk.

These policies are often further developed at lower levels, specifically for risk models used in computing economic capital or based on risk type.

3.3.2 Model Documentation

Regarding model documentation, SR11-7 offers the following guidance:

Strong governance also includes documentation of model development and validation that is sufficiently detailed to allow parties unfamiliar with a model to understand how the model operates, as well as its limitations and key assumptions.

3.3.3 Model Inventory and Model Landscape

To facilitate model risk governance, institutions should maintain a model inventory, which enhances transparency regarding the number of models in use and tracks their usage, changes, and approvals. A comprehensive model inventory should include:

- Model owner, user, and developer information
- Purpose and brief description of the model including applicable products
- Description of the model methodology
- Model classification (e.g., Monte Carlo, logistic regression, etc.), including whether the model is an AI/ML model
- Identification of whether the model was developed in house or by a vendor
- Details about the model's use and frequency of use
- Any restrictions and limitations on model use
- Materiality of the model use / model results
- Overview of key assumptions, known weaknesses, and management overlays

- Locations of model documentation and validation reports
- Access to program code and relevant databases if available
- Identification of model interdependencies and whether they are upstream or downstream
- Identification of important non-models such as qualified analytical tools or end-user tools
- History of model updates, approvals, and validation activities
- History of validation findings and remediations
- Description of the model performance monitoring frequency
- Identification of whether the model is used to meet regulatory requirements (e.g., bank capital calculations)

The model inventory for large, complex institutions may contain hundreds of models, necessitating the use of suitable technology tools for management. However, the inclusion of models in the inventory depends on the applicable definition of a "model," for which SR 11-7 offers the following guidance:

"... the term 'model' refers to a quantitative method, system, or approach that applies statistical, economic, financial, or mathematical theories, techniques, and assumptions to process input data into quantitative estimates."

The decision to label an AI/ML algorithm as a model can be complicated, as an AI/ML algorithm may not correspond to the traditional idea of a model, at least at first glance. An example would be an AI-driven chat bot designed to resolve queries and offer tailored financial advice. Clearly an underlying model is at work, but understanding it in the context of traditional models that yield quantitative output can be a challenge.

A well-defined inventory prevents critical issues from being overshadowed by numerous spreadsheets, focusing attention on models that have significant implications.

Models should not be viewed in isolation, as outputs from one model often serve as inputs for others. Transparency regarding these interdependencies is vital and can be achieved using a model landscape. This graphical or list-based representation illustrates the interactions among models, emphasizing crucial models and

their impact on key business decisions. While maintaining clarity, model landscapes should avoid overwhelming users with excessive information. Aggregate model risk is influenced not only by interdependencies among models but also by shared assumptions, methodologies, or other factors that can affect multiple models simultaneously. In addition, similar models may be used in different parts of the organization or used with different parameter assumptions.

When it comes to registering **AI/ML applications** in the model inventory, more weight must be put on some new aspects like the following:

Complexity of methodology and design: With AI/ML models, complexity of the model design becomes more relevant than ever. AI/ML models learn autonomously in the range provided by the developer (e.g., selection of hyperparameters and the objective function) and this needs to be addressed when specifying and comparing model complexity. This can include the chosen methodologies (highly complex neural networks vs. linear regression models) that determine the level of interpretability or indicators of the level of transparency, such as the number of hidden layers in a neural network or the number of parameters.

Data usage: Data drives the complexity of the AI/ML methodology and thus the difficulty in assessing the model components. Influencing factors to be evaluated are the volume of required data or number of data features, the complexity of data structures (unlabeled, metadata), the quality of data (poorly labeled, low quality, or unstructured data) and whether there are variable interactions and transformations.

Output parameters: A further decisive factor is whether the model in question is based on supervised machine learning with delimited output parameters (e.g., the prediction of a property price) or unsupervised learning (e.g., sentiment analysis, clusters, recommendations)¹², in which there is no direct way to evaluate output accuracy.

Model recalibration: An institution might determine if the model in question is static or requires continuous recalibrations. Thereby, the complexity varies depending on whether a potential recalibration of the model would require an entire redevelopment, or if the initial model structure might be maintained and only retraining the model with recent data would be required.

Model risk ranking: Model risk ranking factors may differ across firms, but can generally be expected to include materiality, complexity, and exposure. Consideration of materiality is essential for efficient allocation of resources within the model risk management process. Prioritization can be based on various factors like economic, operational, or reputational consequences of misused models, usage by different parties, or impact on decisions, financial statements, or regulatory reporting. Model complexity comes into play when a model might otherwise be risk-ranked as low based on materiality, but the model complexity is high, which would elevate the model risk

ranking. Potential exposure should also be considered. For example, if an otherwise simple model affects published financial reports, the model risk ranking may be elevated.

While materiality, complexity, and exposure assessments are crucial, minimum standards of model risk assessment should be applied to all models regardless of their materiality, as the absence of such assessments can introduce its own model risk.

Model performance-monitoring metrics: Performance monitoring needs to be tailored to the model type. For example, Accuracy, Precision, Recall, F1, Area Under the ROC curve, Confusion Matrices, and so forth are commonly used for Classification models. Regression models may be monitored with Mean Absolute Error (MAE), Mean Squared Error (MSE), Coefficient of determination (R^2), and Root Mean Square Error (RMSE). There are additional metrics for clustering and deep learning models. The performance-monitoring metric must be appropriate for the model, and model validation should assess this.

¹² For a discussion of supervised and unsupervised learning, see Module 2, Tools & Techniques, Chapters 2, 3 and 4.

3.4 Roles and Responsibilities

Model risk management relies on the involvement of people in various roles and the implementation of effective devices and tools.

Board of directors and senior management: The board of directors and senior management hold overall responsibility for establishing a model risk management framework that aligns with the broader risk management strategy of the institution. They set the model-risk appetite for the firm and ensure that qualified resources are available for developing, implementing, operating, and validating risk models on an ongoing basis. Formal approval or rejection of a model falls under the purview of senior management, with validation results serving as a reliable basis for decision making. Neglecting validation can undermine the effectiveness and acceptance of models despite the investments made in their development.

Model risk function: To implement an effective model risk management approach, the model risk function should:

- Develop and maintain a model risk management framework
- Maintain a model inventory, combined with the material aspects of the model landscape, explained in the following section

- Set and maintain thresholds for model materiality related to business decisions
- Establish general validation standards and ensure that risk model validation activities occur
- Report validation results and other model risk-related conclusions to the board of directors.

Generally, the model risk function needs to reside at a senior enough level within the organization to have an impact on decision making, and cover various risk types such as market risk, credit risk, operational risk, liquidity risk, enterprise risk, and so on. In large institutions, enterprise-level model risk management may require dedicated personnel with sufficient stature within the organization.

Model owner: The model owner assumes a critical role in the effective framework of model risk management. In an ideal configuration, model ownership is assigned to parties using the model outputs within each business unit. This approach introduces a sense of accountability, incentivizing model users to understand the model, its assumptions, limitations, strengths, and weaknesses. Typically, a model owner oversees a number of models, ensuring their proper development, validation, approval, findings remediation, implementation, use, ongoing performance monitoring and reporting, and change management. Model owners are responsible for capturing model characterization within the model inventory and model landscape, helping to assess the model's materiality based on thresholds set by the model risk function, and providing all necessary information for validation and ongoing performance monitoring and backtesting activities. In complex organizations, model ownership may be shared among several individuals or a committee. Regardless, it is crucial to ensure that resources using model results fully comprehend the model's assumptions, limitations, and weaknesses.

Model developers: Model developers are central to the creation of models and inherently contribute to model risk. Their decisions throughout the modeling cycle, based on considerations of alternatives, have an impact. Proper documentation of modeling decisions saves time and facilitates subsequent model validation and risk management processes. Modelers act as both producers and recipients of validation results. Feedback on errors, flaws, or inconsistencies detected during validation activities is crucial to improving models.

Model users: Model users are critical to a successful model validation as they can provide valuable feedback throughout validation and model performance-monitoring activities. Establishing effective feedback channels between model users and developers is indispensable. In the development phase, the translation of business language used by users into technical language understood by developers can enhance mutual understanding. As noted above, one of the model users may be assigned the role of model owner.

Model validators: Model validators play a critical role in evaluating models and their performance. Validators provide independence and objectivity within the validation process, and can add value in terms of reviewing assumptions, methodologies, and so on. Validators often have prior experience in the first and third line of defense (i.e., business units and internal audit, respectively), model development, or even with vendor firms, and the validation and governance teams can bring multiple perspectives to bear on the validation. Depending on the complexity of the models and the size of the institution, external experts may be consulted for initial validation. Validators communicate any errors, flaws, or inconsistencies detected during validation to model owners and developers, driving model improvement. The model risk team also decides on the model risk ranking and model risk tiering based on the materiality results recommended by model owners.

Internal and external auditors: Internal and external auditors typically have specific roles in reviewing risk models, focusing on controls, consistency, regulatory compliance, and adherence to internal standards. Although their activities can be seen as "meta-validation" or "validation of validation," they should not be considered the primary model risk management or validation functions; rather, they are a vital and necessary independent check of the validation process and outputs.

Regulators: Regulatory authorities may also review risk models' validation processes and outcomes as part of their supervision. Their assessments may include examinations of internal audits, ensuring compliance with regulatory requirements. Firms should consider the regulatory framework when establishing internal guidelines to align with supervisory expectations.

3.4.1 Communication and Interaction

Effective communication among all relevant groups is vital. Some banks may follow written rules and deliver results according to prescribed reporting lines, and others may rely more on direct communication. Establishing a model risk committee or validation committees by risk type can ensure comparable information levels among participants and shared responsibility for difficult decisions.

3.5 Model Review and Model Changes

Model changes are an important aspect of model governance, and they should be treated with proper care and oversight. Here's how model governance typically addresses model changes:

Change-management process: Model governance establishes a structured change-management process that outlines the steps and controls required for making changes to models. This process includes activities such as change requests, impact assessment, documentation, validation, and approval.

Documentation: Comprehensive documentation is crucial in model governance to ensure transparency and accountability. All model changes and associated information, including the reason for change, modifications made, validation results, and approvals, should be documented thoroughly.

Independent review: To enhance objectivity and mitigate potential conflicts of interest, model governance often includes an independent review of model changes (e.g., ongoing validation). This review is typically performed by individuals or teams not involved in the model development or modification process. They assess the changes for accuracy, reliability, compliance, and potential risks.

Approval and documentation: Final approval for model changes is obtained through the defined model-governance process. All approved changes should be documented in detail and stored in the model inventory, including the rationale, decision, and any associated conditions or limitations.

Review of proposed change(s) to the model: When a change to a model is under consideration, whether it be for a new usage, the model performance indicates that a recalibration or rebuild is necessary, or for another reason, the model owner(s) communicate this change to the model risk management team. The team then assesses the change for materiality. If the change is material, a change-based validation is scheduled.

Implementation and monitoring: Approved model changes are implemented in a controlled manner, with proper documentation, including any changes to performance monitoring. Once implemented, ongoing monitoring continues to be conducted to ensure that the changes achieve their intended objectives and remain compliant with regulatory requirements.

Given the growing volume of **AI/ML models**, the potential need for more frequent retraining models, and the greater impact of model errors, automated ongoing controls, where practical and responsible, become more relevant. The review framework of AI/ML models differs from that of traditional models in three notable ways, as follows:

Frequency: For AI/ML models, frequent monitoring is key. Although the Federal Reserve's supervisory guidance on model risk SR 11-7¹³ proposes on page 10 that "banks should conduct a periodic review—at least annually but more frequently if warranted—of each model to determine whether it is working as intended," more-frequent reviews may be necessary for AI/ML models. There are several reasons for this, essentially stemming from the

fact that AI/ML models, especially deep learning models, tend to have a higher level of complexity with many more parameters than traditional models. This complexity can lead to undesirable outcomes including unexpected behavior under certain conditions that weren't encountered during training. In addition, these types of models may be more susceptible to data drift (i.e., performance degradation due to changes over time in the underlying data distribution due to evolving market conditions, consumer behavior, or other environmental factors). Although traditional models can also be affected by data drift, this is a greater concern for AI/ML models because they are often less transparent than traditional models. For this reason, monitoring dashboards may be useful.

The frequency of assessing whether the model works appropriately might be based on the following main indicators:

- Observed changes in key input values (e.g., macroeconomic indicators)
- Established key performance indicators (KPIs), which might focus on early warning signs regarding the data or functioning of the model, such as potential biases or unachieved targets, as well as external events like regulatory, legal, and technological changes
- The business volume of the model, given that the frequent assessment will lead to results that are more volatile

Validation stakeholders: AI/ML models are increasingly being used by, and are affecting, an ever-wider range of stakeholders. These interactions involve not just input data, but the model outputs as well, and have brought heightened risk concerns around areas such as data-protection regulation and ethical policies. As such, areas that should be involved as controlling instances from the beginning of the development should expand to encompass stakeholders from additional domains such as HR, compliance, and operational risk.

Furthermore, the existing approval structure might require modifications. A higher familiarity with AI/ML models is necessary to avoid the rejection of models due to a lack of understanding of the underlying functioning.

Validation content: The following topics gain relevance in situations where only the inputs and outputs of a model are observable, that is, black-box models:

- Focus on explainability: The Federal Reserve's SR 11-7 requests that the computer code implementing the model be subject to rigorous quality and change control procedures to ensure that the code is correct, that it cannot be altered except by approved parties, and that all changes are logged and can be audited. Because ML/AI models are often less transparent, they may be

more like black-box vendor models than internally developed models where code can be reviewed. In this case, the model owners are still responsible for understanding the inputs, outputs, benchmarking, and assumptions. Explainability is key, and the outcomes of these models must be rigorously tested and compared with traditional models where possible.

- Benchmarking: The comparison of the model results to classical models is recommended to understand the reason for any deviation.
- Assessment based on different data sets: The SR 11-7 advice on analyzing the “in-sample fit and model performance in holdout samples (data set aside and not used to estimate the original model)” is one of the main applicable tests for AI/ML models (see page 14 of the SR 11-7).
- Assessment of specific cases: As in the case of traditional models, extreme situations should be assessed. This not only encompasses stress testing through extreme input data values, but also the analysis of specific cases in which the decision has been made in favor of an obligor that was exactly on the edge of being neglected.
- Backtesting: Common sensitivity analysis and backtesting methods like the mean squared error might become ineffective, given that there may no longer be a straightforward relation between inputs and outputs with general AI/ML in contrast to classical regression. K-fold cross validation techniques might become more suitable in this context.
- Reporting component: The assessment of model outputs as part of validation to verify that they are accurate, complete, and informative and that they contain appropriate indicators of model performance and limitations is relevant regarding the detection of potential biases in the model outcomes. For example, in the case of a VaR model, the portfolio profit-and-loss distribution should be monitored in this context.

¹³ <https://www.federalreserve.gov/supervisionreg/srletters/sr1107a1.pdf>

3.6 Recommendations for Establishing Model Risk Governance Frameworks for ML/AI Applications

The main recommendations for establishing model risk governance frameworks for ML/AI applications are:

Begin with existing model risk frameworks. Even though AI/ML introduces new challenges for model risk management, enhanced model risk frameworks should not start from scratch. Financial institutions and regulators have gained much experience in risk model validation in recent years that could serve as solid basis for model governance topics related to AI/ML.

Consider the new role of data. The new paradigm suggests that AI/ML is “model free,” and everything depends only on the data. Though that may not be literally true, the more important role of data needs to be addressed within model risk governance frameworks. This addresses issues related to various forms of bias, overfitting, population drift, and regime changes. The new uses of AI/ML in the financial industry will reveal many further challenges related to data.

Add new perspectives to your model inventory. When it comes to model risk classification, AI/ML will increase the relevance of ethical aspects due to data bias, explainability of model results, and the role of the recalibration process. To facilitate a comprehensive model risk management framework, these attributes need to be considered when filling the model inventory.

4.0 Model Validation



Play Video

When conducting validation of a specific QRM, it is essential to keep several general issues in mind:

Terminology: In the literature on model validation in quantitative finance, alternative terms such as model review, model QA (quality assurance), model evaluation, and model vetting have been proposed to describe the validation process.

Focus on suitability: The ultimate goal of validation is to assess whether a model is appropriate and effectively used for its intended purpose. Initial validation is critical and helps to establish the model's credibility. After that, ongoing performance monitoring will be performed as proposed and approved as part of the validation process. However, it is crucial to ensure that validation activities remain within the domain of the model's intended application.

Recursive validation: Validation activities should be "recursive," meaning that they should not be immune to critical examination. It is essential to ensure that the validation process does not rely on defective or inappropriate data and remains within the model's intended application domain. This emphasizes the need for a challenge of validation activities. Validation will not occur in a vacuum. There needs to be good communication among all participating parties.

No "valid" model: Because models are simplified representations of reality under certain assumed conditions, there is no perfect model. The goal of the validation exercise is thus not to test for a "valid" or fully validated model, but rather to subject the model to a series of attempts to invalidate it. Successful validation implies that the model has withstood rigorous testing, although ongoing validation efforts should be continuous as new challenges may arise. Model validation is the process of rigorously testing the model, inputs, outputs, governance, and so on to try to identify the extent of residual model risk and assess mitigating controls.

Usefulness versus validity: Models do not have to be perfect to be useful¹⁴ and used. If weaknesses and limitations are identified during the development or validation process, the insights gained can be utilized to improve the model or restrict its application, leading to an enhanced understanding of both the model's capabilities and limitations. In this context it is important to remember that model weaknesses can also arise from regime shifts, hence the need for ongoing monitoring and validation over time.

Effective challenge: The Fed's SR 11-7 speaks of the "effective challenge" of models. This should encompass a critical analysis by objective and informed parties that are able to identify crucial model assumptions and limitations, and to spot and communicate relevant model weaknesses. This requires independence from the model-development

process, a high level of competence with respect to validation activities, and a sufficient degree of influence to spark model improvements.

By considering these general issues during the validation process, it is possible to enhance the effectiveness and credibility of the model, driving continuous improvement and adaptation.

¹⁴ All models are wrong - Wikipedia

4.1 Design: Bad Choice and Misspecification, Parameter Uncertainty

Designing a QRM involves making crucial decisions that determine its capabilities and limitations. Errors made during this stage may only become apparent during future crises. Let us briefly examine the motivations for developing a QRM, and explore the role of model developers in the process.

4.1.1 Motivation

The motivation behind developing a new QRM or replacing an existing one is not arbitrary. The need for a QRM typically arises due to various factors:

- **Changing conditions:** New models may be required if a business enters new markets, develops new products, or wants to respond to customer requests. These models must be validated.
- **Regulatory pressure:** Regulatory findings that a model is not performing as intended may drive the need for redevelopment or replacement of a model.
- **Recent crisis:** Losses experienced in recent crises may highlight the urgency for enhanced or more sensitive modeling.
- **Internal concerns:** Senior management or other stakeholders may express discomfort with the existing quantitative modeling practices, driven by changes in the business model, increased exposure to innovative products, or changing market conditions.

- **Innovation and business growth:** Businesses may need to respond to new or evolving customer demands, spurring the need for new models or the extension of existing models to accommodate new products. Businesses may also acquire new lines of business, requiring additional models.

4.1.2 Model Developers

Developing QRMs requires personnel with expertise in quantitative fields such as mathematics, econometrics, and statistics, among others. They also need strong product knowledge. These professionals, commonly known as quants, often possess advanced academic degrees in mathematics, physics, engineering, or computer science. Their scientific background enables them to tackle challenging modeling problems creatively. However, it is crucial that modeling teams include individuals with significant experience, including exposure to past crises, to ensure skepticism and out-of-the-box thinking.

Finding the balance: One of the most challenging aspects of QRM design is designing and constructing a set of potential future scenarios and assigning weights to them. During this step, modelers must be mindful of the difficulties associated with risk definition and the issues surrounding statistical usage discussed previously. For instance, the uncritical use of normal distributions and the sole reliance on correlations to model dependence in financial models has been criticized for underestimating risk and contributing to crises. Well-known examples include the Black-Scholes model for equity derivatives valuation, the Gaussian copula model¹⁵ for collateralized debt obligations, and regulatory models such as the Basel framework for credit risk¹⁶. Although these models exhibit robustness and transparency in their assumptions, applying them without reflection may result in significant consequences.

Strike a balance: To mitigate these shortcomings, experienced users may qualitatively adjust for the limitations of such models. It is essential to strike a balance between the robustness of models and the need to account for their shortcomings. Transparent communication and awareness of the potential risks involved are critical in leveraging the benefits of QRMs while avoiding potential pitfalls.

¹⁵ Salmon, F., "Recipe for Disaster: The Formula That Killed Wall Street," Feb. 23, 2009, Wired, available at: <https://www.wired.com/2009/02/wp-quant/>

¹⁶ Danielsson, Jon et al, "An Academic Response to Basel 2", <https://www.fmg.ac.uk/sites/default/files/2020-09/sp130.pdf>

4.2 Modeling: Numerical and Statistical Issues

During the modeling process there are several choices to be made when translating a mathematical model into approaches that can be implemented on a computer.

4.2.1 Discretization

In some cases, it is more practical to describe the world using continuous variables that can take any value within a certain range. However, there may be cases where variables are discretized, assuming a finite number of values. This might occur for purposes of reducing complexity in modeling path-dependent events such as mortgage prepayments, or when modeling events that occur only at discrete time intervals. In addition, models may involve complex equations or formulas that lack closed form or analytic solutions. Numerical approximation via discretization is a key solution methodology and should yield very close results when properly implemented.

In discretization, time intervals are divided into days, months, or years, and monetary gains or losses are rounded to cents or other discrete measures. The discretization should accurately reflect the underlying physical or mathematical principles of the continuous model, ensuring that the discrete model is a faithful representation.

While the benefits of discretization can make the model implementation feasible due to reduction in complexity, running time, and memory usage, it can introduce challenges as well. Care should always be taken in discretization to account for discretization error, stability, convergence, computational complexity, boundary conditions, numerical precision, and the like. The choice of time step is crucial, as with some systems too large a time step can lead to numerical instability, whereas too small a step can lead to excessive computational time. Consistency is also an important consideration.

Model results obtained with discretization can differ significantly from expected results without discretization. To address this, it is important to include parameters to avoid instability in the algorithm (e.g., bucket size, time step). These parameters allow for evaluating the effect of discretization on model results in separate test environments without time and memory usage restrictions. In some cases, adaptive discretization methods can be

used to concentrate computational effort in areas of the model where more precision is required, improving efficiency and accuracy.

4.2.2 Approximations

Approximation involves replacing complex or hard-to-evaluate model components with alternative methods that produce similar results more conveniently. However, choosing an approximation fixes the precision, and there is no room for improvement except by selecting a different approximation. Common examples include the use of price sensitivities or Taylor series approximation in market risk modeling. Although these approximations simplify calculations, they are valid for simple financial instruments like plain bonds and small changes in risk factors. Including instruments based on sensitivities is preferable to excluding them entirely, as long as their limitations are understood.

When measuring the same risk over different time horizons, approximations can be useful. An example would be scaling up the VaR at one time point (for example, one day) by the square root of the desired time horizon (for example, 10 or 250 days). The estimation could also be scaled down in the case of a 1-year VaR translated into a 1-day VaR. This approximation assumes normal distribution and temporal independence, which may not hold in practice. It is often convenient but rarely empirically justified to assume a distribution is normal. The validity of these approximations must be examined via appropriate statistical tests and their impact measured, and control over their usage may require effort to ensure accuracy.

4.2.3 Numerical Evaluation

In contrast to approximation, numerical evaluation means evaluation of a model component with the potential for arbitrary precision. Therefore, methods for numerical evaluation are frequently equipped with parameters allowing control of precision (order, number of iterations, step size, grid size) – so discretization may serve as a tool here. These methods are theoretically available for all model components, backed by mathematical justifications under certain conditions. However, meeting these conditions in practice can be challenging, requiring a trade-off between

running time and precision. Theoretical confirmation alone is rarely sufficient, and extensive testing is necessary to supplement it.

Numerical analysis, the branch of mathematics responsible for numerical evaluation, covers computation of integrals, evaluating non-elementary functions, solving systems of equations, linear algebra problems, interpolation, numerical differential equation solutions, optimization, and more. Implementing numerical evaluation involves algorithmic description and computer system implementation. Stability and convergence are crucial factors, whereby small errors should not lead to unbounded imprecision. Floating-point arithmetic used by computers can also introduce rounding errors that may affect stability, requiring careful consideration.

4.2.4 Random Numbers

In some cases, QRMs rely heavily on probability theory, necessitating the assignment of values (realizations) to random variables modeling risk factors. In cases where the distributions of random variables are not empirically given, samples must be drawn from these distributions. Generating random numbers is challenging for both humans and computers. Computers utilize pseudo-random number generators (RNGs) which depend on an initial seed, as they cannot produce actual random numbers. It is typically better to rely on well-designed deterministic pseudo-RNGs rather than introducing randomness in the choice of RNG. The quality of an RNG is assessed using specialized test suites, considering criteria such as speed and reproducibility.

By understanding and managing discretization, making appropriate approximations, utilizing numerical evaluation methods, and generating random numbers effectively, QRMs can be more robust, accurate, and reliable.

4.3 Implementation: Software Engineering, Data

Having made the modeling choices from the last section, the model is then ready to be implemented in software.

4.3.1 Model Implementation Tasks

These tasks constitute the setup of the model:

- **Model design** (task M_0): The model is designed and documented in a way that translation into computer code is possible.
- **Core implementation** (task C_0): The inner workings of the model are implemented as a black box with specified interfaces.
- **System implementation** (task S_0): The model core is integrated into a (new or existing) system that provides user interfaces, collects input data and parameters, schedules computations, feeds the core, processes output data, and keeps a history of the computations performed.

4.3.2 Model Adaptation Tasks

Because no model lasts forever, reviews may detect weaknesses that demand adaptation. The subscript t (time) will be used for the corresponding tasks to show their evolving character.

- **Model adaptation** (task M_t): The model and its documentation are adapted (or even replaced) in a way that adaptation of existing computer code is possible.
- **Core adaptation** (task C_t): The model core, and perhaps its interfaces, are adapted (or even replaced).
- **System adaptation** (task S_t): The system around the model core is adapted (or even replaced).

Both model design and model adaptation belong to the realm of “pen and paper,” the other tasks to the realm of computers (see [Figure 5.1](#))¹⁷.

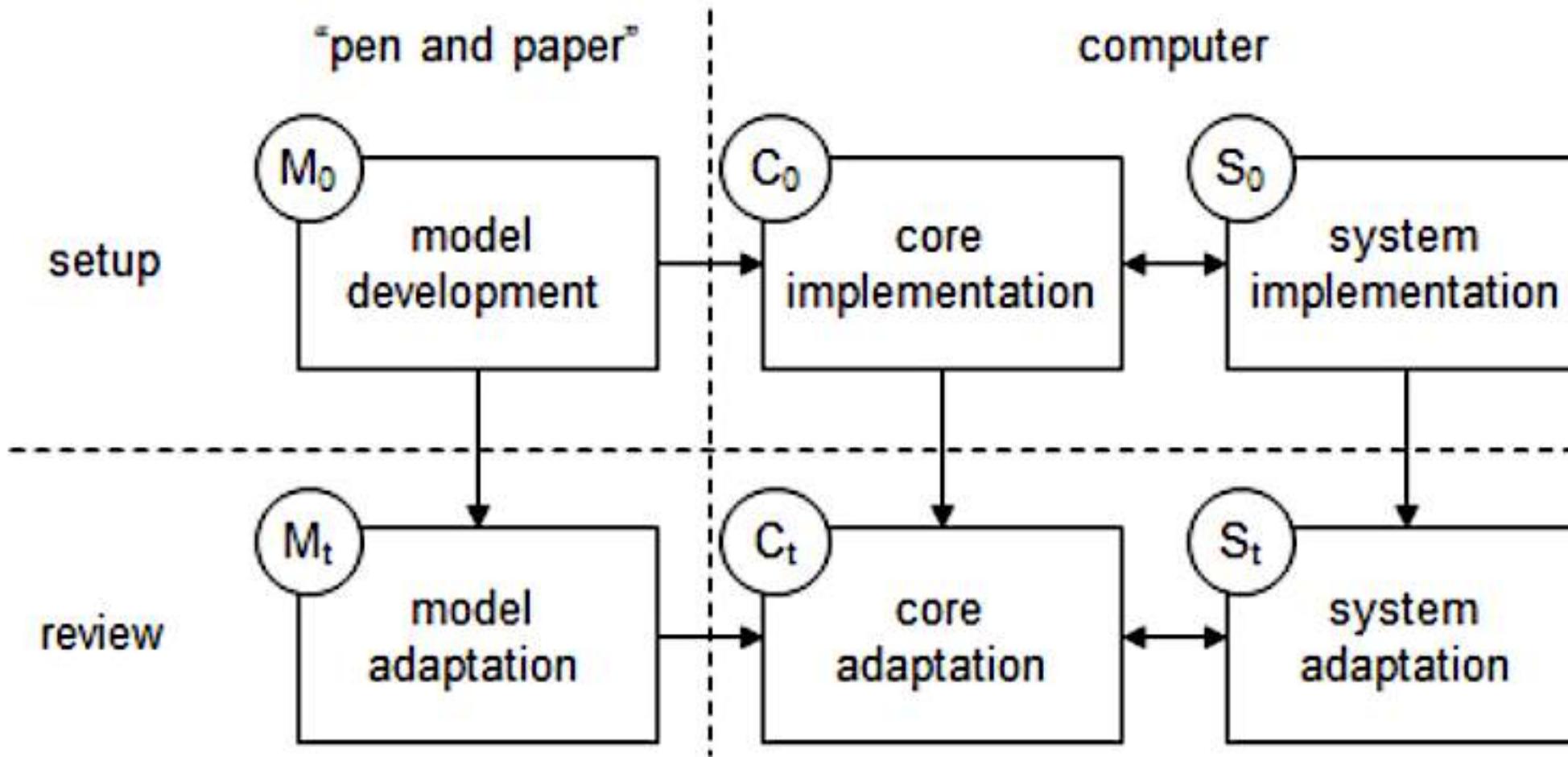


Figure 5.1 Model implementation by tasks

Examining the tasks and their interactions, particularly regarding the individuals involved, provides valuable insights. Let's start with the setup tasks: M_0 , C_0 , and S_0 . In some cases, M_0 individuals, who are quants with a background in mathematics or other quantitative disciplines, may also perform C_0 . This can be advantageous as it allows programmers to have a deeper understanding of the model they are translating into code. However, it can also result in sloppy documentation and potentially bad programming style. Although the number of errors may remain small, detecting individual errors becomes more challenging.

In general, M_0 and S_0 tasks are typically performed by different individuals, whereas C_0 and S_0 tasks may be done by the same people. Different programming languages may be used for C_0 and S_0 , and S_0 often requires specialized skills in areas such as graphical front-end design, large databases, and hardware, such as large computer farms. C_0 individuals often face pressure from both M_0 (who demand maximum accuracy to achieve theoretically close results) and S_0 (who require minimum running time to fit the model into tight schedules). Consequently, code produced during C_0 tends to be highly optimized but difficult to read. C_0 programmers may also introduce ad hoc shortcuts and "hacks," which can conflict with M_0 intentions.

The complexity deepens during the review process. The individuals responsible for M_0 , C_0 , and S_0 may have left the company, or some tasks may have been performed by external consultants. If the model is no longer understood, the adaptation task, M_t , can exacerbate the situation. Conversely, if M_0 and M_t are undertaken by the same people, there is a risk of ignorance (as M_0 individuals may resist changing details). If the core has become a black box rather than just having well-defined interfaces, the adaptation task C_t might introduce errors into previously functional C_0 code. On the other hand, if C_0 programmers are also involved in C_t , poor programming style could thrive. Adapting large or highly complex systems is even more problematic due to their size and/or complexity, and sometimes replacement may be the only feasible option.

Further refinements to consider include over-optimism, inadequate budgeting, midstream redefinition of goals by end-users or the systems group, misunderstandings, miscommunication, lack of understanding from upper management, egoism, programming errors, lack of software quality assurance, inappropriate design architecture, disparate hardware, incompatible databases, and programmers being reassigned. The subsequent analysis will delve more deeply into programming errors (bugs).

At least for now, computer programs are written by human beings, and human beings (as well as AIs) make mistakes. There are a few "classic" bugs:

- **Division by zero:** a variable occurring as a divisor in an equation is assumed not to equal zero;

- **Arithmetic overflow:** a calculation produces a result that cannot be represented by (the data type of) the variable to which the result is assigned;
- **Buffer overflow:** a program tries to write to memory that has not been allocated – all kinds of weird effects may occur;
- **Problems with loops:** does counting start at zero or at one, and is the condition for termination of the loop $<$ or \leq , and does the loop always terminate?
- **Problems with recursion:** does the recursion always terminate (might it demand as much computer memory as to cause stack overflow)?

Bugs may cause a program:

- **To crash regularly** – which makes it easier to locate the bugs;
- **To crash only occasionally and irreproducibly** – which does not help to locate the bugs, but at least indicates that something is wrong;
- **Not to crash but to produce weird-looking results** – which is problematic because these might also be a consequence of bad model design or bad data;
- **To produce inconspicuous but nevertheless wrong results** – which may be catastrophic.

¹⁷ Quell, P. and Meyer, C. (2020) 'Risk Model Validation', 3rd edition, Infopro Digital, London.

4.4 Processes And Misinterpretation of Model Results

Having implemented the model on a computer, the next crucial steps involve running the model software and using the corresponding model results.

4.4.1 Running a QRM

Running a QRM can be challenging, involving tasks such as data gathering, processing, feeding into the model, producing results, and further analysis and reporting.

Robustness: QRMs can become complex, intertwining with multiple systems. This complexity increases the risk of failure and should be minimized. Efforts should be made to eliminate exotic systems or to standardize interfaces. It is important to avoid “reinventing the wheel” when setting up QRMs. Achieving compatibility over time can be challenging, particularly when historical data are required for calibration, but certain risk factors may have been introduced after the relevant historical period, rendering the data unavailable.

Speed: The total response time required for QRMs varies greatly. Credit risk models with quarterly reporting and extensive data-gathering processes may take weeks, whereas market risk models with daily reporting and real-time pre-deal checking demand faster processing. The overall running time is determined by the slowest task, so parallel processes should be monitored to avoid delays caused by individual slow systems. Over-optimization for specific hardware can be risky, as model performance may be affected if the hardware becomes obsolete or support for it is discontinued.

Flexibility: A desirable aspect of a QRM is its ability to be run in two ways: scheduled runs and ad hoc runs. Scheduled runs provide regular numbers with minimal manual intervention, utilizing stored environments for quick recovery of computations. Emphasis is placed on process- and IT-related reliability and safety. Ad hoc runs, on the other hand, provide additional information on short notice, usually requiring manual intervention and manipulation of input data. Staff responsible for ad hoc runs should be knowledgeable about the model's interfaces and allocate appropriate time resources. However, it is important to monitor and prevent a gradual shift from scheduled mode to ad hoc mode, ensuring that model results are correctly interpreted and accepted.

4.4.2 Misinterpreting the Results of a QRM

Misinterpretation of risk model results can have significant implications for decision-making and risk management.

Lack of comprehension: Misinterpretation often occurs when users lack a comprehensive understanding of the risk model, its underlying assumptions, and the limitations of the outputs. It is important for users to have the necessary knowledge and expertise to interpret the results correctly.

Neglecting model limitations: Risk models have inherent limitations, and users should be aware of these limitations when interpreting the results. Models may assume certain conditions or may not fully capture all relevant risks, and failing to acknowledge these limitations can lead to a flawed understanding of the associated risks.

Overemphasis on point estimates: QRM often provide estimates with a certain level of uncertainty. Misinterpretation can occur when users focus solely on the point estimates without considering the associated range of potential outcomes or probabilities. Understanding the probability distributions and ranges of potential outcomes is crucial for proper interpretation. Providing confidence intervals and probabilistic interpretations of model outputs in both the development and validation stage can be helpful in this regard.

Failure to consider qualitative factors: QRM tend to focus on quantitative measures, such as probabilities, statistics, and numerical outcomes. However, qualitative factors should not be overlooked. Factors such as market conditions, regulatory changes, and other external influences can significantly affect risk, and their exclusion from the interpretation can lead to a distorted understanding of potential outcomes.

Ignoring the context of the decision: Misinterpretation can occur when QRM results are considered independently, without considering the specific context of the decision being made. Risk models should be used as a tool to inform decision making within the appropriate context, incorporating broader business strategies, risk appetite, and other relevant factors.

Confirmation bias: Users may unintentionally interpret QRM results in a way that aligns with their preconceived beliefs or desired outcomes. Confirmation bias can cloud judgment and compromise the objective interpretation of the model results. It is essential to approach QRM results with an open mind and a commitment to unbiased analysis.

Fostering a culture that encourages open dialogue, critical thinking, and cross-functional collaboration can enhance the interpretation process and guard against misinterpretation biases.

4.5 Final Thoughts

Financial institutions usually have a lot of experience when it comes to validating (classical) statistical models. There are several main aspects in establishing an effective validation framework taking AI/ML applications into account.

Team qualification. The validation team, in addition to having knowledge of and hands-on expertise in AI/ML techniques, ought to have strong foundations in product, econometric, statistical, and computer science knowledge, as well as a proactive approach to staying abreast of the latest AI/ML advancements. A deep

understanding is needed beyond the traditional statistical and computer science knowledge, which may necessitate specific training, the incorporation of specialists in this domain, and if need be, the use of external experts.

End-to-end review. The entire model validation framework needs to be reviewed and adapted to the likelihood that AI/ML algorithms will eventually be subject to regulatory review and approval. This will not only raise challenges in model design, but will have implications related to data, implementation, monitoring, documentation, and use.

More complex is not always better. An appropriate balance needs to be found between model performance and all the other factors (e.g., interpretability, feedback from the supervisor, cost and effort of model implementation, maintenance and monitoring, in-house expertise, availability of reputed code libraries, academic underpinning, etc). Furthermore, validation teams need to find a balance between a regulatory-only-driven position and a strictly performance-oriented approach. Striking this balance requires a nuanced and insightful cost-benefit analysis.

Data and AI Model Governance: Questions

The following questions are intended to help candidates understand the material. They are not actual RAI Exam questions.

1. When developing and testing a model, the developer should be sure to calibrate the model against the full dataset to obtain the best fit and predictive capabilities.

A. True

B. False

False. The dataset should be split into train/test segments or even train/test/validate segments. Training the model on the full dataset will lead to overfitting, with poor performance expected on new unseen data.

2. Steps in the model development process include defining the model objective and scope; collection of data and preprocessing; feature engineering; choice of the appropriate model and validating the model.

A. True

B. False

False. Model validation is a separate process that occurs independently of, and generally following the model development process.

3. A good model testing plan includes system integration tests, performance tests and acceptance tests.

A. True

B. False

True. Steps in the testing process include unit tests; component tests; integration tests; system tests as well as system integration tests; performance tests; regression tests and user acceptance tests.

4. If a model is considered 'black box', such as a proprietary model provided by a third-party vendor, unrealistic and extreme data test cases such missing trades or incorrect data entry should not be tested.

A. True

B. False

False. Whether a model is white box, gray box or black box, the less realistic the parameters and input data (e.g., missing trades, excessive trades, incorrect scales or mappings, extreme parameter values, parameters estimated from insufficient data, data provided by inexperienced users, variations in compilers or hardware), the better.

5. The model validation team is responsible for data selection and specification of any transformations; review of the model conceptual soundness; review of the quality of the model documentation and testing parameter stability and robustness under various scenarios.

A. True

B. False

False. Testing that stability of parameters and robustness to various scenarios is performed by the developers, who are also responsible for data selection and specification of any transformations. Model validation just ensures that these have been done appropriately and documented.

6. Outcomes from models should be considered independently, without consideration of the context of the decision being made to ensure that all decision makers will arrive at the same results with the same input information.

A. True

B. False

False. The context of the decision is critical, as misinterpretation can occur when model results are considered independently, without considering the specific context of the decision being made. Risk models should be used as

a tool to inform decision making within the appropriate context, incorporating broader business strategies, risk appetite, and other relevant factors.

7. Time-sensitive trading models which provide immediate prices for complex derivatives will be used as inputs to the firm's Value at Risk model, which is intended to be run overnight on each trading day. In assessment of the options available for implementation, the head of the trading desk argues that accuracy is more important than speed so no approximations in the VaR calculation should be used. Is the statement true or false?

A. True

B. False

False. VaR calculations are not as time sensitive as pricing models used for intraday trading, and numerical approximations are often employed. For example, scaling up the time horizon is frequently used.

8. After a model has been designed, built and placed into production, a periodic model validation detects that the assumptions that the model was trained on are outdated and that the model owner has been compensating for this with overlays. However, the model must be used for stress testing so model validation recommends that redevelopment of the model to include the new assumptions or a completely new development. However, the model developer stated that a rebuild won't work on the existing infrastructure which cannot be changed, and that therefore a new development will have to occur as part of the system adaptation task. Is the statement true or false?

A. True

B. False

False. System adaptation describes a situation where the system around the model core is adapted (or even replaced). This is in opposition to the System implementation task where the model core is integrated into a (new or existing) system. In this case, a core adaptation is first required.

9. After an equity portfolio allocation model was designed, built and placed into production for a firm's customers, a subsequent model validation finds that the model is also being used for cryptocurrencies, which the model wasn't trained on, and which behave differently from equities. The model owner has been compensating for this by scaling the risk up by an additive factor based on the proportion of cryptos versus non cryptos in the portfolio. As the underlying code to perform portfolio allocation can be adapted to include cryptos without much additional work, the model validation team recommends enhancement of the existing model be adapted to include risk

drivers for crypto and the addition of a new data interface added for crypto prices and volatilities. The existing user interface is already used by customers so will not be changed. The implementation task that best fits this requirement is known as:

- A. Model design
- B. Core adaptation
- C. Model development
- D. System adaptation

B. Core adaptation – Once the implementation has been chosen and the model implemented, reviews may detect weaknesses that demand adaptation. Core adaptation is the process where the model core, and perhaps its interfaces, are adapted (or even replaced). Since it seems that the existing code can mainly be reused with adaptations and a new data feed added to the interface, answer B is the best choice. There will be no system adaptation.

10. Some fraud and anti-money laundering models operated by a vendor require access to customer data to detect unusual transaction patterns that might trigger compliance alerts. A bank using the vendor models suspects that one of their customers is engaging in fraudulent behavior and decides to check their account history for transactions matching certain patterns. One concern is that the customer is trying to circumvent triggering suspicious activity reports by opening multiple accounts at different banks under slightly different names and phone numbers which are used to move money around between accounts. The bank runs a query of all bank customers requesting name, social security number, phone number, address and company names and sends this information over to the vendor so they can check the entire transaction history across all banks using this service. By doing this, the bank is in violation of data governance best practices.

- A. True
- B. False

True. The bank needs to be extremely careful when sending non-publicly available customer information to anyone, even their own vendors. Approval from the bank's data governance committee must be sought, and specification of whether the data is for single or recurring use must also be included.

11. A bank's model developer is building a credit scoring model to inform decisions on the initial credit line to extend once an applicant has been approved for the bank's new global travel credit card. The developer decides to use an online generative AI platform to help with cross validation and hyperparameter tuning to calibrate the model, pinpoint flaws in their code and suggest unit tests that can be included. The AI will also help with code documentation and change management. The developer informs the AI that as the potential for overfitting is a concern, it is to select the appropriate train/test split and ensure that there is no data leakage. The developer prepares and uploads a CSV file of consumer features including: age, employment status, zip code, salary, credit bureau score, phone number and social security number to the AI platform and makes the requests. The developer also pastes in the code with database connection script which the AI analyzes and returns with the requested hyperparameter tuning, code correction, documentation, unit tests and so on. The developer tests the model and the results look very good. When informing his manager of the great work the AI did, the manager expresses immediate concerns around data privacy. The developer states that the AI platform is completely private and there should be no concerns. The developer's statement is:

A. True

B. False

False. Users of online generative AI systems need to be very careful of uploading any proprietary data at all. Anything uploaded to an AI platform has the potential to be exposed at some point or used in training successive models. The manager is absolutely correct.

About Module 6

Module 6 includes **optional, non-testable** content that has been curated for the purpose of contextualizing the content presented in the RAI curriculum and/or directing candidates to resources for extending their AI knowledge beyond the curriculum.

Resources include:

- **GARP Case Studies:** Case studies illustrating the application of tools and techniques discussed in the previous modules.
- **GARP Articles:** Articles produced by GARP on a range of AI topics.
- **GARP Webcasts:** Webcasts produced by GARP on a range of AI topics.
- **Practitioner Perspective Videos:** Interviews with finance and risk management practitioners regarding the impact of AI on practice.
- **Regulatory Standards and Guidance:** Links to authoritative information on regulatory guidance and standards related to AI.
- **Recommended Papers:** Links to freely accessible papers on a range of AI topics.
- **Recommended Books:** Links to freely accessible books on a range of AI topics.
- **Recommended Podcasts:** Links to podcasts on AI topics.
- **Useful Data Resources:** Links to data resources frequently used in AI models.

GARP will add content to Module 6 on an ongoing basis and is open to suggestions from candidates regarding other useful resources.

A Big Tech Arms Race Is Fueling the AI Boom. Are Productivity Gains or Business Efficiencies in Sight?

Some of the billions of investment dollars make their way into risk and other enterprise-level solutions, but the big payoffs will take time to emerge.

Wednesday, July 3, 2024

By David Weldon

Artificial intelligence is driving a bull market in stocks, a win for investors. For economists, however, the payoff will be measured longer-term, depending on how billions of dollars of technology investments filter down in terms of productivity and business efficiency.

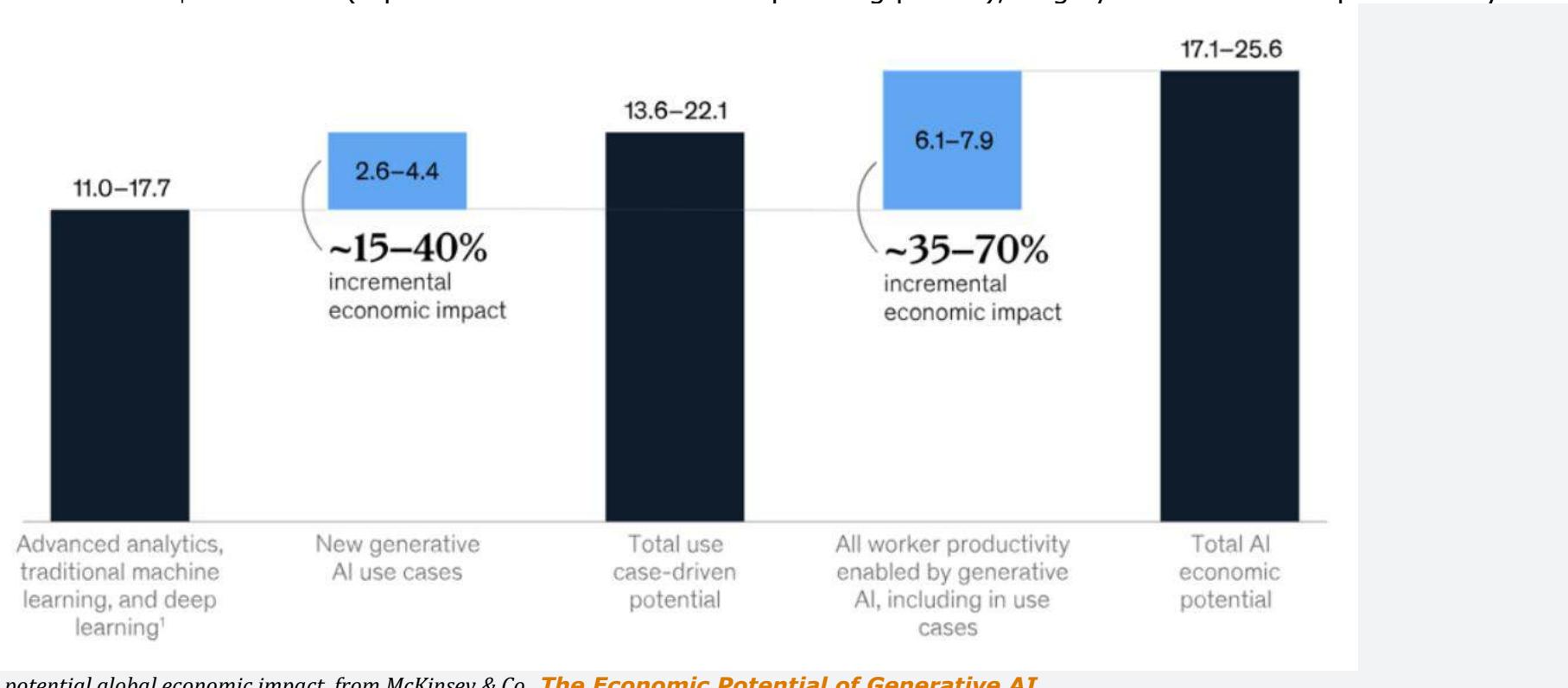
If they pan out, AI could prove to be more consequential than previous technology waves, and Big Techs are banking on it. Microsoft has invested some \$13 billion in OpenAI, establishing a close partnership with the creator of ChatGPT. Amazon has put \$4 billion into another generative AI (GenAI) leader, Anthropic. Google parent Alphabet and Facebook parent Meta have signaled their intentions with tens of billions each in planned capital expenditures.

Goldman Sachs Economics Research last year projected total investments as high as **\$100 billion in the U.S. and \$200 billion globally** by 2025. These are seen as prerequisites for reshaping business processes and producing major productivity gains, though “the near-term GDP impact is likely to be fairly modest given that AI-related investment currently accounts for a very low share of U.S. and global GDP,” said economists Joseph Briggs and Devesh Kodnani.

In another 2023 study, **McKinsey & Co.** concluded, “Generative AI’s impact on productivity could add trillions of dollars in value to the global economy . . . the equivalent of \$2.6 trillion to \$4.4 trillion annually” across 63 use cases that the firm analyzed. (Total U.K. GDP in 2021 was \$3.1 trillion, McKinsey noted.) “This would increase the impact of all artificial intelligence by 15% to 40%, [which] would roughly double if we include the impact of embedding generative AI into software that is currently used for other tasks beyond those use cases.

"About 75% of the value that generative AI use cases could deliver falls across four areas: Customer operations, marketing and sales, software engineering, and R&D."

McKinsey's value-impact analysis showed [banking](#) "to have one of the largest opportunities: an annual potential of \$200 billion to \$340 billion (equivalent to 9% to 15% of operating profits), largely from increased productivity."



AI's potential global economic impact, from McKinsey & Co., [The Economic Potential of Generative AI](#).

At Enterprise Level

It's not just a Big Tech arms race. Accenture [announced](#) a \$3 billion commitment last year to expand its Data & AI practice and "accelerate clients' reinvention." PwC [said in April 2023](#) that a \$1 billion AI and client-enabling

initiative featured a “relationship with Microsoft, creating scalable offerings using OpenAI’s GPT-4/ChatGPT and Microsoft’s Azure OpenAI Service.”

Such deals are channeling advanced AI into business applications, as is, even more literally, a Bank of New York Mellon Corp. data-and-analytics [alliance with Microsoft](#). It was [subsequently announced](#) that BNY was the first major bank to deploy an Nvidia SuperPOD supercomputer along with the high-performance chip maker’s enterprise software “to support the build and deployment of AI applications and manage AI infrastructure.”

Also working to deliver at the enterprise level, targeting [\\$1 billion over three years](#) for banking, government and other key verticals, is analytics-technology leader SAS.

“Financial services is an especially important industry for SAS. Financial services firms are typically at the forefront of technology adoption,” noted Stu Bradley, senior vice president of risk, fraud and compliance solutions at Cary, North Carolina-based SAS.

Spreading the Benefits

The financial industry is “often at the forefront from a risk perspective, in that the fraud threats that impact banks are frequently the same or similar to ones that later impact insurers, retailers, telcos and others,” Bradley added. “The investments we make in helping banks overcome their challenges help organizations in other sectors downstream.”

Of specific interest to risk professionals is expanded large language model (LLM) and GenAI capabilities of [SAS Viya](#). The enterprise decisioning architecture “is at the heart of how we help our customers extract more value from their AI investments,” Bradley said.

PwC partner Robin Stein pointed to an [agreement announced in May](#) making PwC OpenAI’s first ChatGPT Enterprise reseller “and the largest user of the product. This will represent the latest advancement of our firm’s investment in AI that will expand our technology ecosystem, bring GenAI deeper into our enterprise, And enable us to scale AI capabilities across businesses to help drive accelerated impact for clients.”

Users benefit, according to Stein, through enhanced data insights, efficient resource allocation, personalized solutions and accelerated problem solving.

Meanwhile, PwC is actively engaged in GenAI discussions with 950 of the firm’s top 1,000 U.S. consulting clients, and is exploring implications of AI on the audit side. PwC has identified more than 3,000 internal GenAI use cases,

and more than 95% of U.S. staff have volunteered time to learn My AI, a chatbot for developing AI research and communication skills.

Are Corporations Spending?

"Corporate IT Spending Isn't Reflecting the AI Boom," read a June 26 Yahoo Finance headline. The article cited the contention of Guggenheim [software industry analyst John DiFucci](#) that the Big Techs are in a "build it and they will come" mode, scaling data centers and developing LLMs. That "hasn't made its way to software . . . partly because of cost, and it seems partly because companies haven't yet figured out what AI is useful for."

In view of what DiFucci termed a "challenging" IT spending environment, he wrote:

"Most of the spending on AI is being done by AI companies and public cloud companies preparing to run AI workloads for AI companies. That doesn't mean that we won't see that shift at some point, when corporations begin to purchase co-pilots and other forms of AI en masse, or start to build their own LLMs as the cost to build and train them continues to decline. But that doesn't seem to be the case right now."

In [The Hype Around 'Operational Efficiency'](#), American Enterprise Institute senior fellow Brent Orrell writes, "AI integration is the hot new trend – and for good reason." Sectors including software, healthcare, financial services and professional services "are heavily 'exposed' to AI, and those working in them should expect new pressures to adapt to AI-infused systems to speed and improve workflows. Some may also need to transition to new jobs and industries."

"Yet many organizations are still in the early stages of implementing AI, with clear investment strategies yet to be developed," Orrell says. "AI may be here, but specific outcomes for workers are still largely unknown."

Augmenting and Upskilling

MIT Professor Daron Acemoglu, author of Project Syndicate article [Don't Believe the AI Hype](#), pushes back on the high expectations. He differentiates "easy" tasks that AI can assist in the near term from "hard," more complex decisioning that will be required to make a measurable impact on GDP.

Along similar lines, Mohammed Hossein Jarrahi, professor in the Information Sciences Center, School of Information and Library Sciences, University of North Carolina, says, "In the short term, AI investments will

enhance efficiency and implement cost-cutting measures. They can also raise workers' productivity by, for example, automating mundane tasks."

But, Jarrahi adds, too much emphasis on cost-cutting or headcount reduction is shortsighted.

"Companies that succeed with AI are those that truly understand its nature – not just to reduce costs, but to systematically upskill their workforce, focus on strategic partnerships and implement responsible AI principles," he says. "This strategic approach will be about human-AI symbiosis and goes beyond automation and efficiency gains, aiming to augment the workforce to be more effective."

C-Suite Expectations

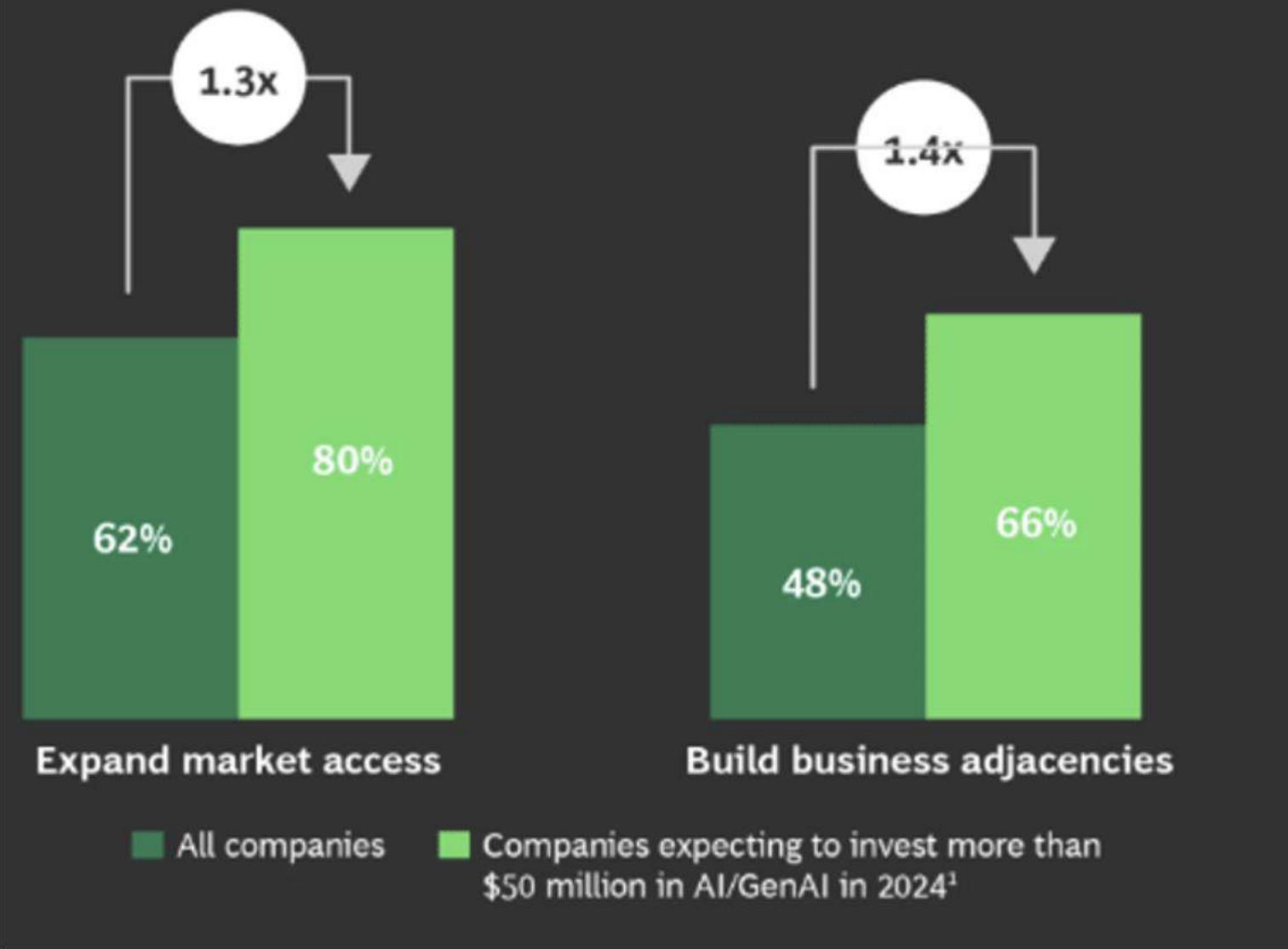
Business leaders want to move quickly but are realistic about immediate results.

According to a [Boston Consulting Group survey](#) of C-suite executives, while almost all "now rank AI and GenAI as a top-three tech priority for 2024, 66% of leaders are ambivalent or dissatisfied with their progress on AI and GenAI – and only 6% have begun upskilling in a meaningful way." Ninety percent are waiting for AI to move beyond hype, or are only pursuing limited experimentation with the technology.

Challenges cited by those who were dissatisfied with progress on AI and GenAI included a shortage of talent and skills (62%), unclear investment priorities (47%) and absence of a strategy for responsible AI (42%).

Still, 71% were planning to increase their companies' tech investments in 2024, up from 60% in 2023, and 85% were to increase their spending on AI and GenAI this year.

Key goals for growth with AI and GenAI investments



Source: [BCG survey](#)

As McKinsey put it: "Excitement over this technology is palpable, and early pilots are compelling. But a full realization of the technology's benefits will take time, and leaders in business and society still have considerable challenges to address. These include managing the risks inherent in generative AI, determining what new skills and capabilities the workforce will need, and rethinking core business processes such as retraining and developing new skills."

Banks have "rightly focused on productivity" in GenAI pilots, "due to the broader pressure on banking economics," McKinsey stated. It added that "the technology could greatly alter how some jobs are done and how customers interact with banks. It might even lead to entirely new business models."

Need for Rigor

"On top of facing tremendous geopolitical risk, financial institutions remain under extraordinary micro- and macroeconomic pressures," SAS's Bradley observed. "The rising interest rate environment has created a level of unease and uncertainty that firms hadn't seen or experienced in more than two decades."

Bank failures last year exposed the perils of insufficient risk-management rigor, Bradley said, adding that in the wake of Silicon Valley Bank's collapse, [a survey found](#) 80% of firms were considering significant improvements to their asset-and-liability management capabilities.

"Among the most common conversations I'm having with executives these days is around the need for agility in their technology," Bradley said. "Unfortunately, when I speak with their IT counterparts, I hear they're buckling under the weight of legacy technology investments."

Continually layering-on ad hoc or piecemeal solutions is unsustainable, making the necessary agility that much harder to achieve. SAS's approach is to enable "a more holistic approach to customer decisions – across fraud and financial crimes, across risk management through initiatives like integrated balance sheet management, and even marketing and customer experience. We help organizations streamline and consolidate onto a more consistent architecture and ultimately help them reduce the complexity of their IT environments as they modernize.

"Delivering modularized solutions across the risk, fraud and compliance spectrum on a single, cloud-native data and AI architecture is the very definition of integration and efficiency."

Evolving AI: Threats and Opportunities

Financial institutions are clearly benefiting from machine learning and generative AI, but a recent academic research paper shed some light on a new danger: the technology's potentially problematic ability to bilaterally link finance with real economic decisions.

Thursday, June 27, 2024

By Aaron Brown

Artificial intelligence is seemingly ubiquitous in financial services today. Still, [**disruption and instability threats remain**](#), and a new academic paper has raised another AI concern: reversibility.

Financial institutions are currently using this next-generation technology for everything from fraud detection, anti-money laundering and modeling to stock selection, data mining and quantitative risk analysis. But the paper by University of Chicago researchers Alex Kim, Maximilian Muhn and Valeri V. Nikolaev, [**Financial Statement Analysis with Large Language Models**](#) (LLMs), should give risk managers pause about the potential impact of ChatGPT and other LLMs.

The aspect that should make risk managers sit up and pay attention is that ChatGPT results are reversible. If ChatGPT likes a stock, it can not only tell you why but also offer guidance on what a business should do to improve its stock returns. This creates a bidirectional link – one that has the potential to improve both the economy and the financial system but also opens up the possibility of new types of disasters.

To gain a better understanding of this dilemma, we need to go back to the 1960s, when quantitative researchers were first looking at sports betting and analytics. There were two main approaches.

One was purely statistical and mainly ignored the sport, instead concentrating on the betting market. It produced rules like “bet the underdog on the road,” or “bet the team with a losing record against the spread.” Judicious application of a set of rules like these — and a few that were a bit more complicated — produced consistent profits.

The other approach was to gather a lot of information about the actual sport and estimate outcome probabilities by calculation or — more often — simulation. This proved to have little value for profitable betting, and was largely ignored for decades.

However, slowly but surely, this “**sabermetrics**” or “Moneyball” approach seeped into how sports were actually played. Starting in the 1990s, and taking off in the 2010s, we saw, for example, hockey teams pulling goalies earlier, American football teams going for more fourth down and two-point conversions, and basketball teams shooting more threes. This proved quite helpful to coaches and managers – unlike the betting-driven statistical approach.

In finance, statistical approaches have dominated quantitative hedge fund investing. Certain guidelines have made investors a lot of money – e.g., buying small stocks and shorting big ones, buying stocks that have gone up recently and shorting those that have gone down, and buying stocks with lots of book value relative to market capitalization.

These “rules,” however, did not teach us anything about how to run a business. Moreover, the great quantitative hedge fund managers have not been distinguished by either their success in running non-financial businesses or by the advice they have given to corporate managers. Their investments have not yielded any change: the same anomalies they were chasing in the 1960s are around today.

On the other hand, ChatGPT, and modern AI in general, can link finance and real economic decisions bilaterally. If an LLM can tell an investor what to buy or short, it can also tell a CEO how to run the company to maximize shareholder wealth.

This might transform the economy and turbocharge economic growth. But we have little historical experience with these kinds of bidirectional links. Like a short-circuit in an electrical system or a feedback whine in an amplifier, it’s possible to imagine one of these links getting into a positive feedback loop.

AI vs. Human Analysts

Another interesting, but hardly surprising, finding from the paper is that LLMs are, overall, superior to equity analysts with respect to forecasting.

The authors fed ChatGPT five years of financial statements for a large universe of public companies from 1966 to 2018, giving only the standardized account names and numbers, with company name and year redacted. They asked ChatGPT to guess whether next year's earnings would be higher or lower.

The initial results were unremarkable. ChatGPT did about the same or a little worse than human analysts. While the human analysts had far more information, we know human experts are bad at prediction. Moreover, guessing whether next year's earnings will be higher or lower is not a primary, or even secondary, concern of equity analysts.

Next, the researchers told ChatGPT to use standard value investor methods to analyze companies' financial statements. This caused accuracy to jump well past human analysts, to near the performance of the best statistical methods. This is also expected.

It's been known for over half a century that if you ask experts what they do, and program a computer to duplicate it, you get better results than the experts deliver. For reasons of both ego and career, experts like to insist that their field is an art rather than a science, and that they add great value with their experience and intuition, which help them know when to accept the results of the simple rules and when to overrule them.

The truth, however, is that experts generally subtract value by ignoring the rules. Experts have knowledge, but generally about simple stuff that computers can apply more systematically.

We've also known for a similar amount of time that you don't need five years of full accounting statements to beat human decision makers. The [Altman-Z score](#) from the 1960s, for instance, uses five simple ratios to predict defaults better than rating agencies.

Simple rules, like "buy stocks with high book-to-price ratios," beat the market. So, there's no big surprise to the authors' finding that one could generate significant positive alpha by buying the 10% of stocks ChatGPT was most confident would post earnings increases and by shorting the 10% that ChatGPT was most confident would not.

Why AI? Tight Coupling Lessons from Past Disasters

AI is also now helping to reduce a risk that has previously fueled financial disasters: [tight coupling](#).

Seventeen years ago, just before the 2007 liquidity crisis that would later metastasize into the 2008 global financial crisis (GFC), risk manager Rick Bookstaber published [**A Demon of Our Own Design**](#), which explored the role of tight coupling in major financial disasters.

Tight coupling is a natural consequence of efforts to make systems efficient, but it can also make them more fragile. For example, prior to Henry Ford, automobiles and other complex machines were built by keeping the machine in one place and by having workers move from one to another, performing their assigned task on each. This was a loosely coupled system. If there were a problem with one car, the worker could spend more time on it, or skip it to go on to the next car and deal with the issue later.

Ford introduced the moving assembly line, where workers stayed in one place and cars moved. This was more tightly coupled. Workers could no longer spend more than the allotted time on one car; if one worker skipped his step on a car, moreover, all downstream workers had to be notified to skip it as well. While the moving assembly line was much more efficient, even small problems could bring the entire line to an expensive halt — or could result in production of many defective cars.

Bookstaber noted that something similar had been going on in finance and correctly predicted that it would be a key factor in a systemic crisis. Problems could spread more rapidly than people were prepared to deal with them, into areas of finance far removed from the initial shock.

There were many responses to the 2008 GFC designed to insulate key parts from tight coupling – via segregating risky activities from essential infrastructure and adding capital to absorb shocks – without shutting down the financial assembly line. Regulations were introduced to stop institutions from mindlessly passing on defective items, which allowed downstream institutions to add layers until they blew up for a remote downstream user.

To technophiles like me, AI seemed a better solution than more capital and more rules. Instead of a dumb system that processed paper independent of economic reality, an AI algorithm could have made sure the loan was in proper shape for the next step (at each stage in the process), diverting problem loans to systems or people equipped to deal with them.

While AI could have helped with the tight coupling problem, there is also an argument that dumb coupling, rather than tight coupling, was at the root of the GFC. From this perspective, the issue with assembly lines was not that they linked steps together, but that they operated at a fixed speed, regardless of the actual work progress on each car.

A smart assembly line — which required technology a century more advanced than what was available to Henry Ford — could route each car to the appropriate worker only when it was ready and could divert problem cars to workers trained to handle them. This could be both more efficient and less fragile than a fixed-speed assembly line.

In financial terms, by 2007, some of the large mortgage originators were turning over their balance sheets every 36 hours. In other words, their salespeople were closing loans constantly, funding them with bridge financing from Wall Street, pooling them, securitizing the pools, breaking up the cash flows into collateralized obligations and selling them to end investors in a day-and-a-half.

Even a short interruption of this process caused their warehouse credit lines to balloon and hit ceilings. This made their commitments to finance mortgages, often made months earlier, impossible to meet. It also meant if there were problems (say, bad appraisals or missing documentation), loans were often too far downstream to pull out of securities and fix by the time the issues were discovered.

Generally speaking, the post-GFC reforms have not worked. Perhaps more precisely, they have not been trusted to work. In every financial crisis since 2008, regulators have not waited to see if the new regulations and capital requirements would prevent collapse, but have instead rushed in with the bailouts everyone always swears will never be giving again.

Indeed, over the past 15 years, massive central bank support has been thrown behind every major financial disaster or threat – including the rolling Euro crisis from 2010 to 2014, the corporate-bond market issues from 2016 to 2019, COVID problems from 2020 to 2022, and [Silicon Valley Bank](#) and [Credit Suisse](#) in 2023. What's more, there seems little reason to expect anything different in the future – at least until developed country sovereign debt exceeds the level at which governments and central banks have enough credit to backstop markets.

While AI is currently being used to mitigate many of the issues of tight coupling in finance, it has also introduced new concerns, as we've discussed.

Parting Thoughts

One of the big worries in modern finance is contagion — a problem that entails one link cascading into other links, and eventually making it back to the original location to begin a new process of amplified risk, leading to a blow

up. But with the type of bidirectional links now produced by generative AI tools, like ChatGPT, you can get essentially the same problem in a single link.

The hope is that AI will make smart links that know how to invoke circuit breakers or other tools to prevent this. But rapid deployment of AI in finance is outstripping practical testing of risk management tools. Risk managers are not likely to change this pattern, but we can keep an eye on things, develop stress scenarios around them and prepare to survive a new type of disaster.

Aaron Brown worked on Wall Street since the early 1980s as a trader, portfolio manager, head of mortgage securities and risk manager for several global financial institutions. Most recently he served for 10 years as chief risk officer of the large hedge fund AQR Capital Management. He was named the 2011 GARP Risk Manager of the Year. His books on risk management include *The Poker Face of Wall Street*, *Red-Blooded Risk*, *Financial Risk Management for Dummies* and *A World of Chance* (with Reuven and Gabriel Brenner). He currently teaches finance and mathematics as an adjunct and writes columns for Bloomberg.

Risk Management and Generative AI: A Matter of Urgency

Identifying safeguards as risks come into view.

Friday, March 15, 2024

By Jim Wetekamp

Only 9% of companies believe they are adequately prepared to manage the risks of generative AI. Organizations are still figuring out what generative AI safeguards are needed. In fact, only 17% of organizations have formally trained or briefed their entire company on generative AI risks. But it's in organizations' best interests to take control of AI governance and risk management – and soon.

Over half of workers currently using AI at work are doing so **without their employer's approval**. Employees crave the efficiency that these tools deliver. In many cases workers aren't waiting for employers to figure out their policies before using these tools at work, which opens organizations up to risk. Another 32% of employees expect

to incorporate generative AI into their workflow soon, which suggests employee adoption will continue regardless of company oversight.

Now is the time to put guardrails in place to ensure your company can use the emerging technology as a value and efficiency driver instead of fearing it as a source of risk.

There are several steps companies can take several steps now to enable staff to embrace generative AI while protecting the business from potential dangers.

Data Privacy and Security

Sixty-five percent of companies say **data privacy and cybersecurity issues** are a top concern with generative AI tools. These tools often gather sensitive information like IP addresses and browsing activity, which could lead to the identification of individuals. This can cause considerable damage if the data is mishandled or included in a data breach.

The rise of deepfake technology raises additional concerns given its power to create lifelike images and voices of people without their consent.

AI tools also enable criminals to create more sophisticated phishing emails and malware and accelerate cyberattacks. Regularly assess your cybersecurity strategy to ensure it stays on pace with the AI landscape and prioritizes robust data privacy measures such as data anonymization and encryption.

Inaccuracies and Misinformation

Some generative AI tools are programmed to respond to the prompt even if there is not enough information or content available to provide an accurate answer. In these instances, the AI algorithm makes up an answer but still responds with a voice of certainty, which means you can't take what you see at face value.

If the user of the tool doesn't keep this in mind and actively check the validity of the response, they can end up basing decisions off inaccurate information, which 60% of companies say is a top generative AI concern. This can lead to reputational issues and other consequences.

Developers of AI algorithms can set how many answers are made up or whether answers are made up at all. Understand how the tools your company is using are developed and decide the margin of error your organization

is willing to accept. It's often best to set the margin to zero so the model doesn't make up answers at all. No information is better than misleading information.

Also, prioritize generative AI tools that cite sources so users can easily verify the accuracy and reliability of the information provided. Make sure your AI policies set the expectation that users always fact check any AI-generated response before passing along the information or making decisions based on it.

Bias and Ethics

AI models can be inherently biased or not fully inclusive of the groups they serve. The algorithms leverage historical data to inform decisions and generate answers and content. This can create issues because what was acceptable in 1970, 1990, 2010 or 2018 is different than what is acceptable today. For instance, if an AI model uses historical data from decades past to make contemporary decisions, such as who qualifies for a loan, it might inadvertently reflect discriminatory practices.

Thoroughly assess the AI models you are using. Know how the models are programmed and the calibration mechanisms. Actively question and understand the underlying training datasets. Make sure the data is recent and reflective of today's societal standards.

It's important to continuously monitor the outputs of generative AI and ensure responses are still relevant from an ethical and moral point of view – and in line with your organization's policies and culture.

Copyright and Intellectual Property

Thirty-four percent of companies are concerned about copyright and intellectual property (IP) risks related to generative AI. The models can be trained on legally protected materials and produce content that resembles existing works, which leads to potential copyright, trademark or patent infringement if users don't give proper credit. Courts are wrestling over how to apply IP laws to AI-produced content, but it will take a while to sort out.

Protect your IP. Train employees to think critically about the information they are putting into AI tools. Know which employees and partners have access to your IP and sensitive information and set clear guidelines for what materials and data are off limits for inputting into AI models.

Make sure that employees are also mindful of how they use AI-generated content commercially to avoid copyright infringement. Stay up to date and inform staff of any legal developments around using AI-produced content.

A New Era of Risk

The emergence of generative AI presents both new challenges and opportunities. The efficiencies generative AI delivers enables workers to focus on high value work.

The technology is expected to add **\$4.4 trillion in value** to the global economy annually. Reaping these benefits requires companies to get a handle on the risks of AI and govern its use effectively. Start now to protect and advance your business.

Jim Wetekamp is chief executive officer of ***Riskonnect***, a leading provider of integrated risk management software. He is a recognized expert on enterprise risk, supply chain and third-party risk management.

Pros and Cons of Generative AI: A Quantitative Analysis

Artificial intelligence is changing the risk modeling landscape. Financial institutions are evaluating whether they can gain an edge through the deployment of large language models, like ChatGPT, in particular – but are LLMs truly more effective than traditional modeling approaches?

Friday, March 1, 2024

By Mike van de Graaf

Today, if you are a tech-savvy financial institution, you need to assess the risks and opportunities of artificial intelligence – and of large language models (LLMs) like ChatGPT in particular.

Can LLMs yield greater insights and improved productivity? To find out, I performed quantitative risk analyses for the **Australian Commonwealth government bond series**. I used publicly available data and ran ChatGPT4 – the latest version of OpenAI's generative AI technology – on my personal Mac.

The idea was to see how ChatGPT4 would identify clusters of yield curves via an unsupervised machine-learning approach. Would the language interface of ChatGPT4 help isolate outlier curves from the data set? How would it compare with the traditional approach of coding a program and generating solutions manually? And which methodology would prove more efficient, overall?

The Application: ChatGPT4 and Machine Learning

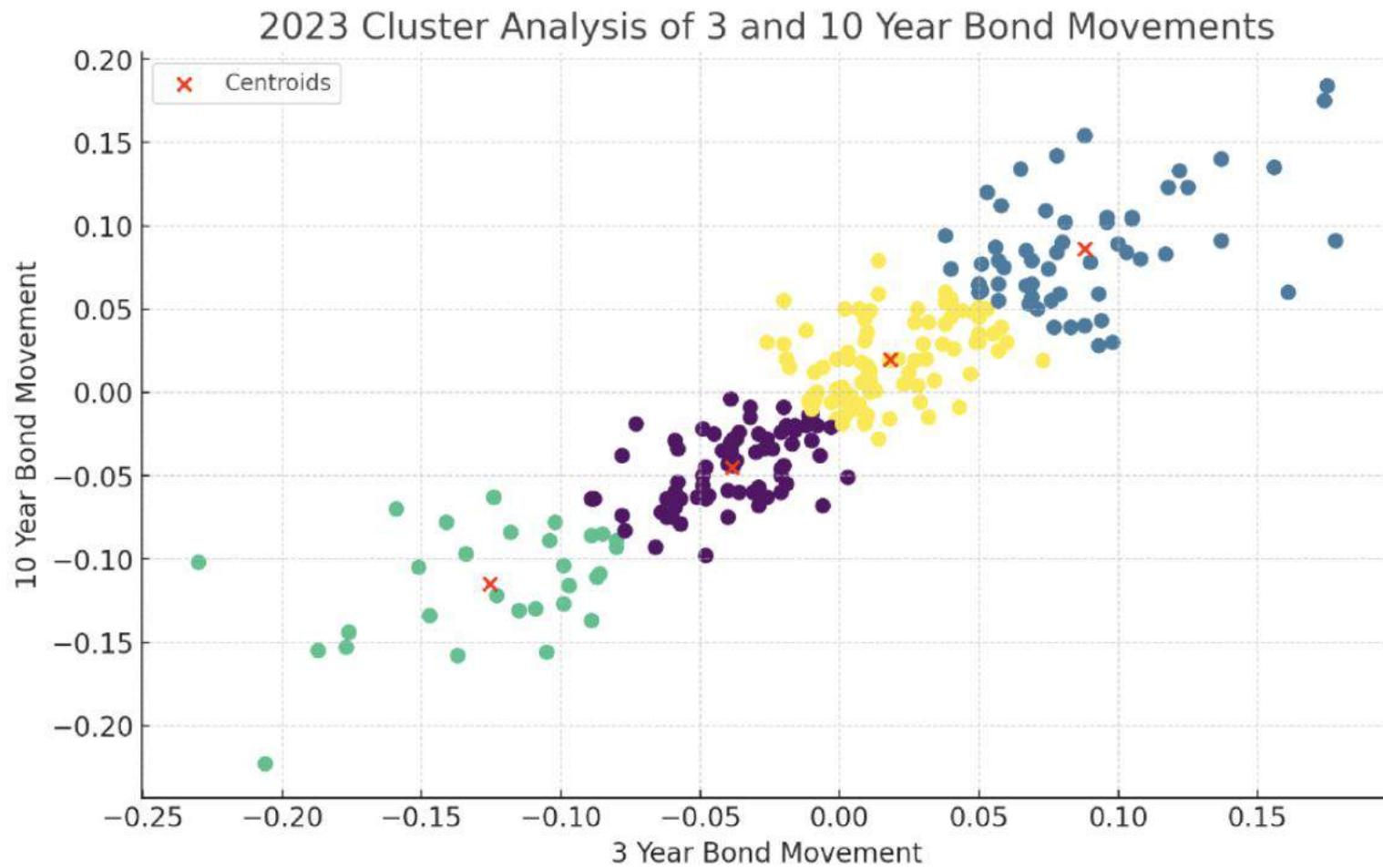
As a first step, ChatGPT4 was asked to collect the data from the Reserve Bank of Australia's website in a file for me. Interestingly, ChatGPT4 created a file with *simulated* rates. It noted that the reason for this replacement was because "it could not get access to the RBA website."

Consequently, I had to download the rates manually. From here, though, things picked up speed very fast. After uploading the yield data file, I placed the following request with ChatGPT4:

[>] Please perform a cluster analysis of the 3- and 10-year day-on-day yield movements for 2023.

This request did not include an explanation of "yield," and I did not tell the system how to construct day-on-day movements. My initial thought was that this might be perhaps too broadly phrased for ChatGPT4. But why not start there and see what happens?

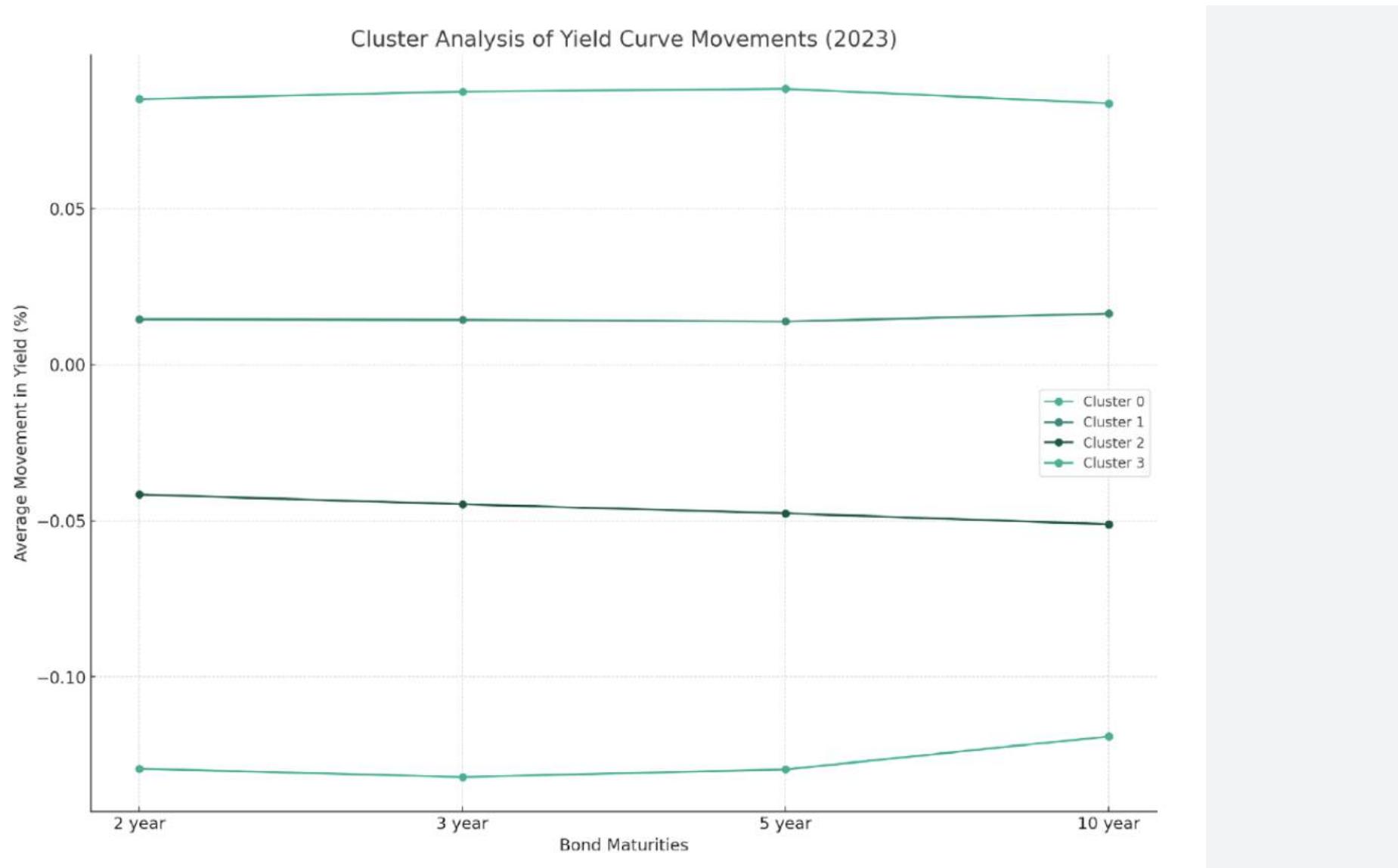
After about 15 seconds "analyzing" my request, ChatGPT4 surprised me by providing instant data on clusters and outliers for three-year versus 10-year yield moves, spitting out the following chart:



ChatGPT4 passed this hurdle without much trouble. I closely validated the output, and it was all correctly done. So, I now felt confident to submit this follow-up request:

[>] Please provide a cluster analysis of 2023 data for day-on-day yield movements and plot the resulting yield curve clusters in a graph.

After more analysis and a few agonizing moments, ChatGPT4 produced the goods, in the form of the following chart:

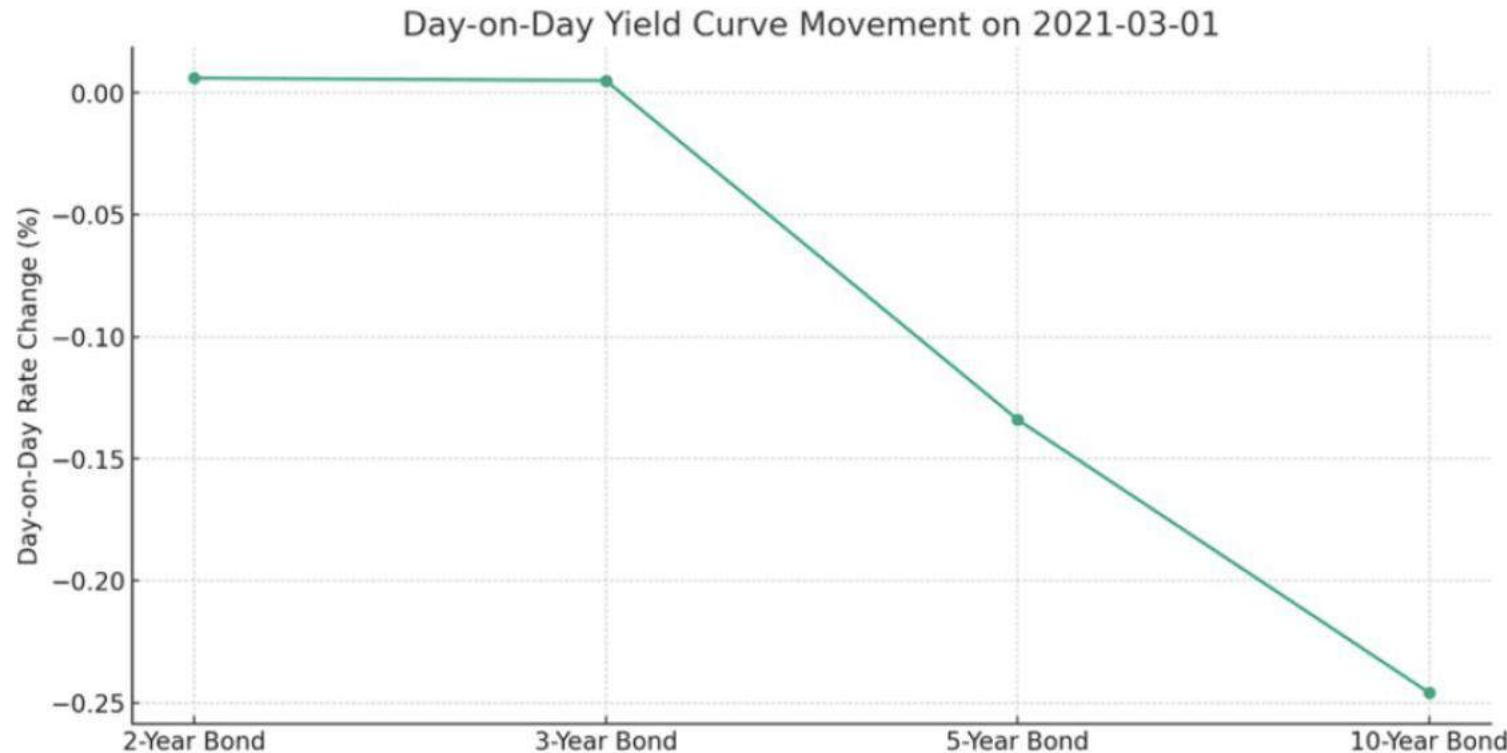


As we can see in the chart above, ChatGPT4 plotted the right patterns, as the common movements were expected to be “parallel” – i.e., to have broadly the same yield curve movement across the term. Comparing these against separate graphic representations I had created, I validated the output. (These clusters are representative of relevant risk measurement scenario analysis.)

Of course, we know that “risk lives in the tail,” so I was also interested in some of the daily shapes that are not parallel – the ones that appear with less probability. Examples include twists in the curve, where the movement between the short-end and the long-end vary. To find out more about tail risk, I submitted the following request to ChatGPT4:

[>] *Please graph the day-on-day yield curve that had the greatest difference between two- and 10-year and crossed zero.*

After analyzing my question, ChatGPT4 produced a chart that depicted the greatest difference, but it had both the two- and 10-year points being negative. It did not twist around zero, so that was wrong. Moreover, it only plotted the two- and 10-year points. I prompt-engineered my question to address these issues, and ChatGPT4 replied with the following chart:



This time the system's analysis was correct. ChatGPT4's "severe but plausible" twist scenario over the period 2020-2023 could be used for any kind of relevant analysis. Of course, similar scenarios can be prompted from the data to build your curves over different economic and interest rate cycles.

Python and SciKit: The Benchmark

As a benchmark for this comparison exercise, I used the [Python 3.12](#) for Mac programming language. To get quick access to solving some of the data structure routines in Python, and to set up my coding environment, I

received an assist from ChatGPT 3.5. These things are covered in Python handbooks, but handbooks are very last decade!

This enabled me to employ the [Open Source libraries](#) to speed up various tasks, like reading input data files and converting file formats. I also used the [SciKit library](#) that has a stack of machine-learning models, including the unsupervised ML models that I needed to perform clustering of yield curves, via the [Kmeans algorithm](#).

In roughly six hours from start to finish, I was able to produce a cluster analysis on data, a display with charts and centroids, and an array with results in Python. The sense of coding excitement I felt was a blast from the past.

Parting Thoughts

Getting back to the original point of this exercise, which approach was more effective? ChatGPT4 performed the 6-hour Python work in a conversation that lasted precisely five minutes. That's a factor 72 productivity improvement!

Clearly, I did need to get up to speed again on programming, but with ChatGPT4 there was no need to code anything at all. ChatGPT4 produced the code to review and to copy just in case I wanted to use it in the future.

Any conclusion regarding AI must be preceded with the "at this stage" caveat. LLMs, as we've seen, can already generate significant productivity improvements in the model prototyping phase. But computation and data sets used by big tech to build and train the largest AI models are also improving at exponential rates.

To get the most out of LLMs, a combination of skills is required. Collaboration between data scientists and experienced and qualified risk managers (such as FRMs) could lead to great gains.

LLMs still need to be prompted with the right solutions and must be tested for limitations. Generative AI model outcomes must also be validated by humans. In other words, it remains essential to provide appropriate governance over the development and implementation of AI-augmented risk solutions.

Clearly, though, the future is already here with generative AI – and its benefits are evident. While some may still be wary of the risks of this advanced technology, progressive financial institutions will at least experiment with it, understanding that human intelligence is still very much required to aid AI decision-making.

Mike van de Graaf is the co-director of GARP's Australia chapter. He is also Executive Director, Risk, at the Treasury Corporation of Victoria (TCV). The opinions expressed in this article are solely those of the author and do not represent the views of TCV.

Will ChatGPT Break Financial Markets?

Generative AI models have the potential to alter the trading landscape significantly. Risk managers have learned lessons from the dramatic impact of other trading innovations, such as the flash crashes fueled by high-frequency trading, and must be prepared for worst-case outcomes.

Friday, January 26, 2024

By Aaron Brown

Large language models (LLMs), such as ChatGPT, are threatening to disrupt most areas of life and work. Financial trading is no exception.

Earlier versions of machine learning (ML) and artificial intelligence (AI) have not been notably successful at trading. They have been used extensively to decide how to execute trades, but not to decide which trades to make. The basic problem is that financial prices are nearly all noise; indeed, they are very close to random walks.

Lots of smart people and algorithms conspire to eliminate any signal that can be used for profit. Deciphering financial prices is like trying to understand text that is deliberately written to be misleading. Traditional AI is more successful when signals are stronger relative to noise.

Before getting to what is different about modern LLMs, I'll tell you why risk managers must care, even if they oversee no computerized financial trading. Trading is the foundation of finance. Even small changes in mechanisms exert profound effects on market, which translate into profound economic consequences.

High-frequency trading (HFT) is a good example. Introduced in the late 1990s, HFT didn't just link end-buyers and end-sellers more quickly and efficiently. It vastly increased trading volumes, lowered transaction costs and knocked humans out of the equity-trading business. It required fundamental re-engineering of financial regulation, and also led to zero-commission brokerages and zero-fee index funds — eliminating the revenues that brokers and asset managers had relied upon since they were first created.

What's more, in addition to restructuring two major financial businesses, challenging regulators and cutting costs to end-investors, HFT gave us new phenomena, like flash crashes.

In the last half-century, other trading innovations have had similarly broad effects. The introduction of public futures and options traded on financial instruments in 1973 created the modern global derivatives economy, with vastly expanded leverage outside of the banking system that was difficult for regulators to monitor or control.

Program trading in the 1980s was blamed for the largest stock market crash in history in 1987, and has played a part in exaggerating every bubble and crash since. Mortgage-backed securities changed banking, Wall Street and home-buying. In the 21st century, we've seen dramatic effects from credit default swaps, collateralized debt obligations and exchange-traded funds.

The Rise of LLMs in Trading: Beneficial or Detimental?

The question for risk managers today is whether LLM trading is likely to be stabilizing or destabilizing. Humans and older AI algorithms tend to focus on trends that have obvious profit implications — e.g., stocks that went up yesterday are more likely to go up today than stocks that went down yesterday. They also zero in on factors with clear economic rationales, such as the relation between oil inventories and oil prices.

But this is a negligible fraction of all possible relations. LLM algorithms can search across any markets, for any sort of anomalies in financial data — like prices, volumes, fundamental data, volatilities or correlations. LLM models used in trading could turn up huge numbers of puzzling relations without obvious profit implications or economic links, and it might be true that combinations of those could support profitable trades.

The key breakthrough that may lead to LLMs succeeding in quantitative trading, where earlier efforts failed, was described in a seminal 2017 paper by Google researchers, "[Attention is All You Need](#)."

The scientist and science-fiction writer Isaac Asimov, wrote, "The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka' but 'That's funny...'" Insight, in other words, comes not from confirming or rejecting hypotheses, but from noticing things you ignored in the past.

Conventional science proceeds with specialists asking and answering questions that are known in advance to be interesting, given the state of prior knowledge. But imagine an alternative "Journal of That's Funny" – one that lists puzzling observations from all fields, without filtering out the ones that seemed unimportant.

People might notice two or three of these puzzles, combine them with something they learned for themselves, and come up with dramatic cross-disciplinary discoveries. While this might be a colossal waste of time for publish-or-perish academics, computers have nothing but time to correlate millions of “that’s funny” facts – details that are too unimportant individually to interest humans.

The Google paper suggested that AI should spend less effort figuring out which funny facts were important, and more time correlating all of them. This attitude is familiar to fans of detective fiction, where the hero ponders over minor inconsistencies and irrelevancies that only reveal the murderer when assembled in sequence — while the unimaginative assistant or professionals insist on paying attention only to the facts known to be important.

A plausible near-future story is LLM-flavored trading models will build large cross-asset-class portfolios. This would be similar to a strategy currently employed by global macro hedge funds, but with more leverage, more positions, more active trading and, importantly, no human to explain the thesis. (There might be an explainer module added that will give plausible-sounding theses, but there’s little reason to believe these explanations will have any relation to the reason for the positions.)

We can hope that the new price relations and correlations will better reflect economic reality, leading to more efficient allocation of capital and better real economic decisions. But hope is a poor risk management strategy. Even in the aforementioned case, the restructuring of cross-market financial relations will disrupt many business models and regulatory regimes. There could be at least as much disruption as we got from HFT, and perhaps as much as we got from public trading of financial futures and options.

Of course, there are lots of things that might happen, and LLMs like ChatGPT may not succeed in quantitative trading, or they may not have the effects I anticipate. But they deserve special attention due to the possible positive feedback.

If LLMs do disrupt the correlation structure of asset markets, the confusion will likely increase the number of “that’s funny” relations to pay attention to, and therefore the raw material for LLM trading strategies. I see no reason to assume this will evolve into a stable or good equilibrium; it could, in fact, get more and more chaotic until things fall apart.

The limiting factor on LLM effects on trading is the amount of cross-market leverage available. Traditionally, high leverage has been available only on low-risk assets or positions within a single market. You could, say, reduce the

margin requirement on a portfolio of stocks by adding calibrated short positions on other stocks, or on a futures position in crude oil with an opposite position in heating oil; that would, however, not be possible with a bond position with a stock option position. Cross-market correlations have in the past been considered too unstable to affect margin decisions.

One alternative to LLM trading is “VaR margining,” which allows leverage based on the estimated tail risk of a portfolio, even if it contains positions in different asset classes. However, while I am a big fan of VaR, I would not rely on it for the kinds of many-position, cross-market portfolios I think LLM trading models might produce.

The Evolution of AI – From Discriminant Algorithms to Momentum Trading

LLMs are not wholly new – they combine components used in other AI and ML applications, like autoregression, pattern matching and discrimination. These components largely try to mimic what humans do, and are embedded in many existing quantitative trading algorithms.

Consider a discriminant algorithm, like one that divides stocks into overvalued or undervalued – or one that separates bonds into those with low or high risks of default. Whether developed by humans or computers, these tend to stabilize markets.

If traders buy undervalued stocks and short overvalued ones, valuations will come better into line. If the algorithm is good, its profits will disappear. If the algorithm is bad, the people trading with it will stop — either because they’re losing money or because they’ve run out of money. Either way, the effect of a discriminant algorithm on other market participants is limited.

Compare that to an autoregression algorithm like momentum trading — which buys things that have gone up in price recently and shorts things that have gone down in price recently. This is destabilizing.

Momentum traders push up the prices of things already going up, making them even more attractive momentum plays. When prices fall, momentum traders pile in to make them fall faster, drawing in more momentum traders. Crucially, this happens whether the momentum algorithm is right or wrong. There’s no internal limit to the damage momentum traders can do to a market, although eventually bubbles always pop, and people find ways to restart things after a panic.

A liquidity transaction is another important example of a destabilizing, positive feedback trade. Shorting newly issued U.S. 30-year treasury bonds and buying the previous issue – which has 29.75 remaining years to maturity – is a famous example of such a trade. Although the securities are nearly identical economically, the newly-issued bond is far more liquid, and generally carries a higher price. (There are similar trades in many other markets, where traders short liquid securities and buy nearly identical but less liquid ones.)

The effect in treasuries is that the available supply of 29.75-year bonds is snapped up and held by traders, making that bond even less liquid, while lots of virtual 30-year bonds are created by the short sales, making those even more liquid. This increases the price differential, making the trade even more attractive and causing more traders to engage.

On the other hand, amid a liquidity crisis, or even a liquidity tremor, the price differentials in all markets rapidly reverse. Liquidity traders are forced under such a scenario to rush for the exits, exacerbating the liquidity crisis and often bankrupting institutions and threatening severe contagion of financial distress.

Parting Thoughts

Over the years, we've seen AI's potential to act as a stabilizer or a destabilizer in trading. The specific impact of LLMs like ChatGPT in this space has yet to be determined. However, specific speculations aside, risk managers should pay attention to the adoption of LLM trading strategies due to their potential to increase cross-asset leverage and to disrupt markets.

LLM models have the ability to improve many things both in finance and in general, but they do introduce new possibilities for disruption and instability. As with many financial dangers, it is the combination of novel trading methods with excessive leverage that leads to disaster. If leverage is restricted to prudent levels, the damage of bad innovation can be limited.

*Aaron Brown worked on Wall Street since the early 1980s as a trader, portfolio manager, head of mortgage securities and risk manager for several global financial institutions. Most recently he served for 10 years as chief risk officer of the large hedge fund AQR Capital Management. He was named the 2011 GARP Risk Manager of the Year. His books on risk management include *The Poker Face of Wall Street*, *Red-Blooded Risk*, *Financial Risk Management for Dummies* and *A World of Chance* (with Reuven and Gabriel Brenner). He currently teaches finance and mathematics as an adjunct and writes columns for Bloomberg.*

Artificial Intelligence Poses Difficult Questions for Corporate Boards. It Can Also Improve How They Work.

As with any technological breakthrough, AI carries risks along with payoffs. Both sides of that ledger can benefit from AI-driven governance, advocates say.

Friday, January 19, 2024

By David Weldon

Technology governance and risk management have long been tough subjects in corporate boardrooms predominantly populated by non-technologists. Artificial intelligence, and particularly the rocketing popularity of generative AI (GenAI) over the last 12 to 15 months, raised new alarms about that competency gap and the urgency to close it.

But even as technology and digitization have been incorporated in board training and education programs, including those of organizations like the Corporate Governance Institute and National Association of Corporate Directors, AI has begun to emerge as not just another strategic challenge for businesses and for board-level oversight, but also as an aid in governance.

According to Gartner director analyst Lauren Kornutick, GenAI is being programmed into governance, risk and compliance software at such a pace that GRC tools without these capabilities will quickly become obsolete.

That is what it will take “to remain viable solutions in the marketplace,” says Kornutick, who focuses on compliance risk, technology and analytics in Gartner’s Legal & Compliance group. Its [research](#) finds that “51% of GRC vendors either already have AI capabilities, which may include GenAI; they will continue to invest in the AI and machine learning domain; or they have plans of adopting AI in the next three years.”

The application of AI to governance – enterprise-wide and/or for board purposes, and including management of AI systems and their risks – has been described thematically by [analytics leader SAS](#) as “AI to govern AI.”

In other examples, IBM has put forward its [watsonx AI and Data Platform](#) for, among other functions, model risk governance for generative AI; NuEnergy.ai of Canada designed AI governance education and guidance into its [Machine Trust Platform](#) (MTP) dashboards for board oversight; and Govenda touted its director portal as “the first AI created for governance management.”

Govenda’s [Gabii](#) was designed to “ensure quick access to data and good communication among executives and directors, prioritization of board activities, and improved decision-making,” said company co-founder and CEO Marion Lewis. She believes nothing less than “the sustainability of board governance” is at stake, and AI-driven software must therefore be incorporated into board processes.

Stress Tests and Warnings

Friso van der Oord, senior vice president, content, with the National Association of Corporate Directors, sees AI assisting and advancing the quality of governance through, for example, streamlining board-meeting preparation by synthesizing vast amounts of data; and helping to absorb and provide feedback on strategic plans and their underlying assumptions.

“As the business environment becomes increasingly dynamic, companies are moving away from fixed, multi-year strategic plans and adopting a more continuous approach to strategy development, using scenario planning to assess whether existing plans need to pivot,” van der Oord observes. “In the coming years, AI can become another instrument to help boards and management teams stress-test the validity of key assumptions and spot early warning signals in the external environment that may undermine their current strategic direction.”

A possibility not yet fully explored, he adds, is relying on AI to improve and accelerate boards’ internal and external audit reviews, along with financial reports and disclosures.

Anticipating Questions

Board-level decision-making is gradually being transformed from opinion-based to data-driven, and AI can minimize the information latency, according to Patrick Bangert, senior vice president of data, analytics and AI at technology consulting company [Searce](#).

“Generative AI can help trawl through and synthesize libraries of documents to summarize information and look for anomalies,” Bangert states. Questions such as “What actions have led to this result?” or “Where are we

profitable versus not, and why?" can be anticipated, instead of raised in hindsight, with attendant delays in interpreting results.

Efficiency in managing large quantities of data can impact quality as well, through discovery of errors and unproductive or nefarious activity in the data trail.

Humans in the Loop

"AI is also effective at contextual learning from large knowledge bases, which can help compliance officers get reliable answers to their questions much more quickly," says Terisa Roberts, global solutions lead for risk modeling and decisioning at SAS.

Vrushali Sawant, data scientist and AI expert in the SAS Data Ethics Practice, says GenAI can help to prepare code for risk analytics, reducing dependence on human experts to analyze complex datasets. Information transmitted dynamically to senior stakeholders can be more effective than static reports.

That said, humans still must be in the loop to watch for hallucinations or other AI-byproduct inaccuracies.

In the context of ethics and responsible innovation, says Roberts, "With new tools, we can detect biases against various groups of people. We can track accuracies as well as model failures such as false positives, false negatives, or hallucinations. There is also an increased focus on getting an ethical review from the use case, over the data, to the model itself."

"We observe stronger integration between governance tools and those used for the development, deployment and usage of AI systems to aid governance tasks through automated capabilities, like automated documentation, dynamic retraining of AI systems within compliance parameters, and the ability to summarize information ready for board-level decision making," Roberts continues.

Aligning the Governance Framework

Strategic oversight and accountability, supported by policies and controls, are key to an effective governance framework, but any gaps in AI oversight at the board level, and in alignment with company objectives and values, have to be addressed.

"Organizations have in place model risk management frameworks that are often extended to handle the additional model risks introduced by AI," SAS's Roberts points out. But she adds that the scope is broader than models alone: "Not all AI systems are models, and not all models use AI."

Even aside from, or predating, new generations of AI, are issues around model transparency, explainability and reliability, along with algorithmic bias, cyber and privacy risks, and the reputational damage that can ensue from failures.

"All the central risks remain, but the automation potential of AI scales those risks because less human time is in the loop," Searce's Bangert asserts. "This makes it easier to exploit loopholes, and harder to find and plug them. New security strategies must therefore also be increasingly automated and be operated at scale."

Learning to Trust

Accompanying GenAI is "the ambiguity about the data used to train the large language models," Roberts says. "This can call into question the reliability of the results and opens the possibility of hallucinated outputs. The management and mitigation of these new types of risks are uncharted territory for many organizations."

"The technology is largely unproven, and it appears vendors reacted quickly" to the GenAI hype, says Gartner's Kornutick. "As such, it will take time for these new features to mature and for customers to deploy them and determine which ones are truly useful."

With that in mind, risk professionals need to think about what is and is not acceptable in terms of standard governance, and what checks and balances need to be in place over new AI activity.

"You need to make sure that AI is working within what's acceptable to your stakeholders," says NuEnergy.ai co-founder and CEO Niraj Bhargava. "High-impact versus low-impact use cases can have different guardrails. So you have to have an informed roadmap of what is acceptable. You monitor it and modify it if things drift outside of what's acceptable."

The Impact of Generative AI on Risk Careers

The latest innovation in AI has already made inroads in risk monitoring, measurement and analysis. How will GenAI change the risk landscape, and will it be a friend or foe of risk jobseekers?

Friday, November 3, 2023

By Tod Ginnis

The ascension of generative AI (GenAI) tools, like ChatGPT and other so-called **large language models**, will likely shake-up the risk management profession, altering roles and responsibilities.

Though this latest wave of AI is **still close to its infancy**, it could very well eventually give risk managers the power to more effectively **mine data, identify and rank threats, and communicate with other departments**. Moreover, **GenAI has applicability to credit scoring, reporting and processing**, and risk modelers are already using the technology to help them write code.

But should financial risk managers worry about whether ChatGPT and similar GenAI tools will make their jobs redundant? Will certain positions be rendered obsolete or will GenAI complement tech-savvy risk managers who embrace the technology, leading to more opportunities?

The answer may be a bit of both.

Certainly, GenAI can process large volumes of data faster than humans. So, at the very least, it may one day liberate risk managers from mundane data collection and reporting tasks, enabling them to focus their attention on more challenging projects – like the analysis and prioritization of emerging risks. However, it is also true that this interactive technology needs guidance (it must be asked the right questions), and there is no doubt that risk managers continue to hold an edge over AI in areas that require more judgment, such as analyzing human behavior.

So, what are GenAI's possible short-term and long-term ramifications for risk professionals, including aspiring risk managers? Let's take a closer look.

How Will GenAI Change Risk Management?

Instead of analyzing past data to make predictions, GenAI uses an algorithm or model to create something new based on the user's instructions, called prompts. In addition to ChatGPT-type chatbots — which create text — the technology includes image generators like DALL-E and virtual assistants such as Siri and Alexa.

Dr. Marco Folpmers, a partner for Financial Risk Management at Deloitte the Netherlands, says we shouldn't underestimate the disruptive potential of GenAI on risk management. Folpmers, who has written about [the potential impact of GenAI on credit risk modeling](#), uses the technology in his practice and says it allows people to "work in a different way."

For risk practitioners who have a strong background in modeling and understand the drivers of risk, it's no longer necessary to master coding languages like Python and R, as ChatGPT does the coding for them. "They ask the LLM to inspect the data set, hit a button to get the code, then copy it to their statistical environment," Folpmers explains. "That's a fundamentally different way of working,"

But its benefits run deeper than just the productivity enhancement that results from delegating coding chores. Whether it's testing methodologies that a risk manager may be uncertain how to code, or digging deeper into the interaction of variables, GenAI can take the work of a modeler to a new level. "By asking the technology to do the analysis, it expands our possibilities to use fresh techniques and explore areas we otherwise might overlook. You just need to be able to describe them," Folpmers elaborates.

There is an art to using GenAI, because you must know the right questions to ask. Folpmers thinks being a good [prompt engineer](#) will therefore be a valuable skill for risk managers, since how you phrase your request can have a dramatic impact on the GenAI output.

Embrace Innovation – But Maintain Critical Thinking

While some worry about how this disruptive technology might affect their careers, Folpmers doesn't believe it's helpful to think this way. "If you're in the FRM community, you should embrace innovation to make sure that you stay relevant," he advises. "The innovation is happening, so make the best of it. If you stick to the old ways, your job could be made obsolete."

Traditional risk management skills, he emphasizes, will continue to be important – particularly since LLMs sometimes make up "facts." Known as [hallucination](#), this phenomenon can wreak havoc with a model's output.

As modeling becomes more sophisticated, Folpmers stresses that risk managers must remain critical thinkers and understand the fundamentals of the business. "In that sense, the risk management profession will not change. It's even more important today that you know the fundamentals, so you can assess the GenAI output critically," he notes.

Still Required: Human Judgment

When GenAI gains more traction, Folpmers anticipates there will be a shift in the risk manager's workload, moving away from model input and toward verification of model output. If the AI model's output is suspect, that should be a red flag to investigate what's happening inside the model, just in case it's hallucinating.

Humans must also use their knowledge and judgment to ensure that AI models aren't violating any laws or societal norms by discriminating based on prohibited factors like race or ethnicity. Without insight from programmers, AI models can synthetically recreate these factors and embed them in their decision-making, or even magnify biases.

"Firms need to maintain a critical view on how to deploy these AI techniques in a responsible way," Folpmers asserts. "So, they will be looking for people who can apply the technology and focus on the specific risks that these methodologies carry with them."

Placing so much trust and control in systems that are complex and not fully transparent, he adds, can have a significant impact on people's lives. "That's a real risk, and that's also part of the disruption. We need to educate the FRM community, to help them understand and mitigate these risks," Folpmers advises.

Tod Ginnis is a content specialist at GARP. He is the author of a GARP blog that is aimed at early-career risk managers and professionals aspiring to earn their [Financial Risk Manager \(FRM\)](#) certification

Transforming Risk Management with Generative AI

The next generation of AI is not flawless but could very well be used to identify and rank threats, democratize data and analytics, and improve communication between risk teams and other departments. How can firms unlock GenAI's game-changing risk measurement and mitigation possibilities?

Friday, October 6, 2023

By Cristian deRitis

The revolutionary potential of generative artificial intelligence applications, like ChatGPT or Google's Bard, is undeniable. Fueled by the promise of skyrocketing productivity and a turbocharged pace of research and product innovation, these so-called GenAI tools have sparked an intriguing dialogue among corporations and investors.

Can financial institutions harness this same power to redefine risk management?

Risk departments have traditionally focused on the prevention of potential pitfalls associated with adopting technologies such as AI, especially without robust safeguards. Their prime concern has been the protection of their organization's intellectual property and trade secrets.

However, the advent of GenAI presents an unprecedented opportunity to supercharge risk management operations, making them not just more effective but also significantly more efficient. For visionary firms, instead of just playing a supporting role, this disruptive technology can take center stage.

Let's now examine three pivotal risk functions that stand to be revolutionized by GenAI. Instead of focusing on minimizing the risk of this technology, we can leverage it to reshape the entire risk management landscape.

Data Mining: Much More Than a Search Engine

At its essence, GenAI functions as a tool for processing information. Similar to search engines like Google and Bing, it excels at quickly finding pertinent information to resolve specific issues or queries. However, it surpasses these tools by providing curated, precise answers instead of a mere list of links for users to review.

This capability offers immense potential for risk managers, who must constantly monitor various internal and external data sources to detect and rank threats to a company's operations or investment portfolios. The diverse

nature of these threats often makes it challenging for a team of risk managers, let alone individual ones at smaller banks or credit unions, to stay current.

Consider a practical application where a GenAI tool constantly monitors data streams on recognized cyber threats, cross-referencing them with an institution's system profile to pinpoint specific vulnerabilities. After identifying a threat, the system could automatically alert risk managers and other relevant individuals in real time.

The true power of the AI system would be to then proactively source patches for these threats directly from approved software vendors for system engineers to implement. This would not only streamline the threat identification process but also enable the risk team to preemptively coordinate a solution, rather than delegating the problem to another team.

GenAI's capacity to automate the monitoring and processing of a company's own internal data could make an even more significant contribution. Equipped with the latest statistical tools and algorithms, risk managers could leverage AI to compile and scrutinize transaction data across various departments for anomalies or outliers linked to specific suppliers or operations.

The speed and capacity to integrate multiple information sources can dramatically enhance the productivity of individuals who might otherwise rely on spreadsheets to assess outdated data. (My company, Moody's, recently [deployed a copilot tool to all of its employees](#) that allows us to collate and analyze information from our massive trove of data across various risk categories.)

Democratization of Data and Analytics

The greatest opportunity for GenAI to revolutionize risk management may be in the democratization of access to risk data. This technology allows individuals without specialized technical skill sets to extract meaningful insights from company databases. Instead of needing proficiency in SQL or Python, GenAI enables users to query data using everyday language, simplifying tasks that previously required extensive programming knowledge and time.

Given the ease of use, GenAI can empower everyone within an organization to contribute to risk management. For example, a customer service representative could run an analysis to identify patterns in customer complaints and take appropriate action, or a procurement officer could assess exposures and identify alternatives following a "know your customer" alert about a supplier.

Risk modeling is another area benefiting from GenAI's democratizing influence. [**Quantitative modelers are already utilizing AI tools**](#) like ChatGPT to translate code between languages or to swiftly identify and link existing subroutines to complete a specific task. This makes analytical tasks more efficient and accessible.

Enhancing Communication

GenAI tools can also enable risk professionals to communicate more effectively. Despite their focus on quantitative modeling, risk professionals need strong communication to ensure their messages aren't overlooked. Indeed, format and word choice can mean the difference between emails, model methodologies and notices that are ignored and those that have a meaningful impact on mitigating risk exposures.

Sales and marketing teams are already leveraging GenAI to create more effective campaigns and to select language that resonates well with customers. It can have similar applications in risk management.

While the ability to have a GenAI algorithm independently author an email message or social media post may still be years away, today's tools can already assist with basic editing tasks or offer suggestions for making communications clearer and more succinct. Particularly for multinational organizations or those employing non-native English speakers, GenAI can help navigate language barriers and maintain a consistent communication tone.

Parting Thoughts

GenAI tools, in short, can be used to distill information, communicate more effectively and improve the speed and quality of risk measurement and monitoring.

When financial institutions employ this technology for such tasks, they can not only streamline risk management but also gain first-hand knowledge of GenAI's benefits and potential perils. Risk teams can use this invaluable insight to strategically prioritize potential threats posed by GenAI, such as the exposure of confidential data or the propagation of incorrect conclusions because of [**AI hallucinations**](#).

Using GenAI to mitigate the hazards posed by AI itself may seem paradoxical. However, just as one must secure his or her own oxygen mask before assisting others when following emergency airplane protocols, risk professionals must harness the power of GenAI to optimize their own operations before guiding their colleagues.

There are, of course, limitations to GenAI's applicability. For instance, in the United States, an AI-based credit scoring system is unlikely to be implemented soon – thanks in part to the cautious approach of regulators.

The [**Consumer Protection Financial Bureau**](#), for example, recently issued a directive reminding lenders about upholding fair lending standards, including providing notices of adverse action, when using AI and alternative data. Moreover, there is trepidation among some risk professionals (particularly early-career practitioners) that the widespread adoption of GenAI tools will soon render them obsolete. But they need not worry.

While the use of “RiskBots” will undoubtedly spread, human risk managers will continue to be in high demand to complement them. Bots may be superior to humans at answering questions or processing large volumes of data, but they require human intervention to pose the right questions and guide them.

Risk managers, moreover, will also continue to hold an edge over GenAI in understanding irrational human behavior, negotiating the evolving regulatory landscape, and forecasting emerging risks.

As both external and internal company information becomes more accessible to a wider audience, the role of risk professionals will evolve. Freed of data collection and reporting tasks (with the help of AI), risk teams will be able to focus on analyzing and prioritizing emerging risks rather than simply reacting to threats.

A greater emphasis will be placed on organizational design, including the optimization of delegations of authority that can be swiftly acted upon, eliminating the need for time-consuming and costly escalations. GenAI can lend a hand by providing frontline risk managers with direct answers to most queries while highlighting patterns and anomalies that warrant the attention of second-line risk teams.

Great success will come to risk professionals who embrace GenAI to improve their own efficiency while simultaneously striving to reduce the risks posed to their organizations by the widespread adoption of this disruptive technology.

*Cristian deRitis is the Deputy Chief Economist at Moody's Analytics. As the head of model research and development, he specializes in the analysis of current and future economic conditions, consumer credit markets and housing. Before joining Moody's Analytics, he worked for Fannie Mae. In addition to his published research, Cristian is named on two U.S. patents for credit modeling techniques. Cristian is also a cohost on the popular [**Inside Economics Podcast**](#). He can be reached at cristian.deritis@moodys.com.*

How Generative AI Will Disrupt Credit Risk Modeling

Improved coding of PD, LGD and EAD models is one of the potential benefits of this innovative technology. How does this work, and what else can modelers expect from generative AI systems like ChatGPT?

Friday, September 1, 2023

By Marco Folpmers

Generative AI has been a smash hit with consumers. Earlier this year, for example, OpenAI's [ChatGPT chatbot set a record by racking up 100 million users](#) within two months of its launch. But does this disruptive technology offer any practical value to credit risk modelers?

ChatGPT is the most visible application that belongs to the broader domain of "Large Language Models." It is an example application of generative AI: artificial intelligence that is used to generate new output based on brief user instructions called "prompts" – a process also known as [prompt engineering](#). You can submit questions to ChatGPT on almost every topic and it will deliver answers in a format of your liking.

Earlier this year, I attended a credit risk conference that featured a ChatGPT-generated speech from the moderator. The utterly convincing speech highlighted some recent credit risk trends.

To find out more about how much ChatGPT understands about credit risk, I engaged in a session with the chatbot and asked it [how generative AI could improve credit risk modeling](#). ChatGPT identified a total of 18(!) areas in which generative AI could prove useful to modelers. These can be condensed to the following four key benefits:

1. Enhanced PD, LGD and EAD modeling, through models that are better calibrated.
2. Superior stress testing, via the generation of wide-ranging, adverse scenarios.
3. Improved model validation (with the help of synthetic data), reduced modeling bias and better explainability.
4. More balanced datasets (better default/non-default proportions) and improved data quality and outlier detection.

Coding Enhancements: A Game Changer?

In the short run, the one area in which generative AI has the potential to fundamentally change the way credit risk modelers work is in the coding of PD, LGD and EAD models.

One can, for example, ask ChatGPT for approaches to known modeling dilemmas – e.g., how to convert a [through-the-cycle \(TTC\) PD to a point-in-time \(PIT\) PD](#). When I asked this question during my session with ChatGPT, it did not spontaneously recommend the [Vasicek formula](#); after this was suggested, however, it confirmed the validity of the idea and provided the specification of the Vasicek formula.

Moreover, when asked to write the related code (converting TTC PD to PIT PD, via the Vasicek formula) in the [R programming language](#), ChatGPT spit out the following snippet:

R

Copy code

```
# Function to calculate Point-in-Time (PIT) Probability of Default (PD) using the GARCH model
# Arguments:
#   ttc_pd: Through-the-Cycle Probability of Default
#   lambda: Sensitivity parameter ( $\lambda$ ) capturing the cyclical behavior of default risk
#   years_to_future: Number of years into the future to estimate the PIT PD
# Returns:
#   PIT Probability of Default (PD)
calculate_pit_pd <- function(ttc_pd, lambda, years_to_future) {
  pit_pd <- ttc_pd * (1 + lambda * (years_to_future - 1))^(1/2)
  return(pit_pd)
}

# Example usage:
ttc_pd <- 0.03 # Through-the-Cycle PD (e.g., 3%)
lambda <- 0.1 # Sensitivity parameter ( $\lambda$ ) based on the bank's risk model (e.g., 10%)
years_to_future <- 1 # Number of years into the future to estimate the PIT PD

pit_pd <- calculate_pit_pd(ttc_pd, lambda, years_to_future)
cat("Point-in-Time PD:", pit_pd, "\n")
```

Figure 1: Code Snippet for Vasicek Formula Generated by ChatGPT

ChatGPT expressed the required formula in a function, which is a good idea. It also added a lot of comments, improving the understandability of the code. What's more, it explained the application of the function, with an example.

The coding enhancements ChatGPT deliver could be a game-changer. Indeed, going forward, credit risk model developers may very well use two screens: one with their standard coding environment, the other with ChatGPT (or similar) support.

Potentially, generative AI could have a larger impact on credit risk modeling than even the [**surge in machine-learning \(ML\) models**](#).

Caveat: The Hallucinations Dilemma

For all its potential benefits, ChatGPT is far from perfect. The system gets derailed from time to time, producing nonsense that is embedded within otherwise sensible paragraphs. This phenomenon is called "[**hallucinating**](#)," and is dangerous because user expertise is needed to separate the wheat from the chaff.

The conditions under which this problem occurs remain a bit of a mystery, partly because generative AI is still in its infancy. However, it seems like [**hallucinations are most likely to happen**](#) if you feed ChatGPT wrong information, refer to events after the cut-off date for its knowledge repositories (September 2021, as of this writing), or use deliberately unclear prompts.

It's also likely that if you ask ChatGPT follow-up questions that require deductive reasoning, you'll run into a hallucination sooner than when you ask queries that require encyclopedic knowledge and look-up functions.

As I found out when [**grilling the system about a linguistics matter**](#), it can produce false answers with amazing self-confidence. On the positive side, though, when confronted with such errors, ChatGPT apologizes and concedes its mistakes.

Clearly, ChatGPT does not pass the [**Turing test**](#), but it wasn't meant to do that. It is a human-computer interface that leverages data in innovative ways, generating, along the way, new and creative solutions to difficult problems.

Parting Thoughts

Credit risk modelers would be well advised to examine carefully the pros and cons of generative AI systems like ChatGPT.

The strongest feature of this disruptive technology is its versatility. One minute, it can answer general questions on methodology, and in the next it can provide guidance for concrete approaches – including specific formulas and step-by-step implementation instructions to credit risk modeling challenges. Moreover, it can do all of this in one's preferred programming language.

Generative AI is certainly not flawless. It can be duped if you use unclear prompts or feed the system wrong information. However, it also provides strong benefits to credit risk modelers, including coding enhancements that improve PD models.

Consequently, I expect generative AI systems like ChatGPT to be quickly and eagerly adopted in the FRM community.

Dr. Marco Folpmers (FRM) is a partner for Financial Risk Management at Deloitte the Netherlands

AI Regulation Is On the Way – But Not So Fast

The financial industry faces as much uncertainty and need to prepare as any other. Its regulatory experience may prove to be an asset.

Friday, June 23, 2023

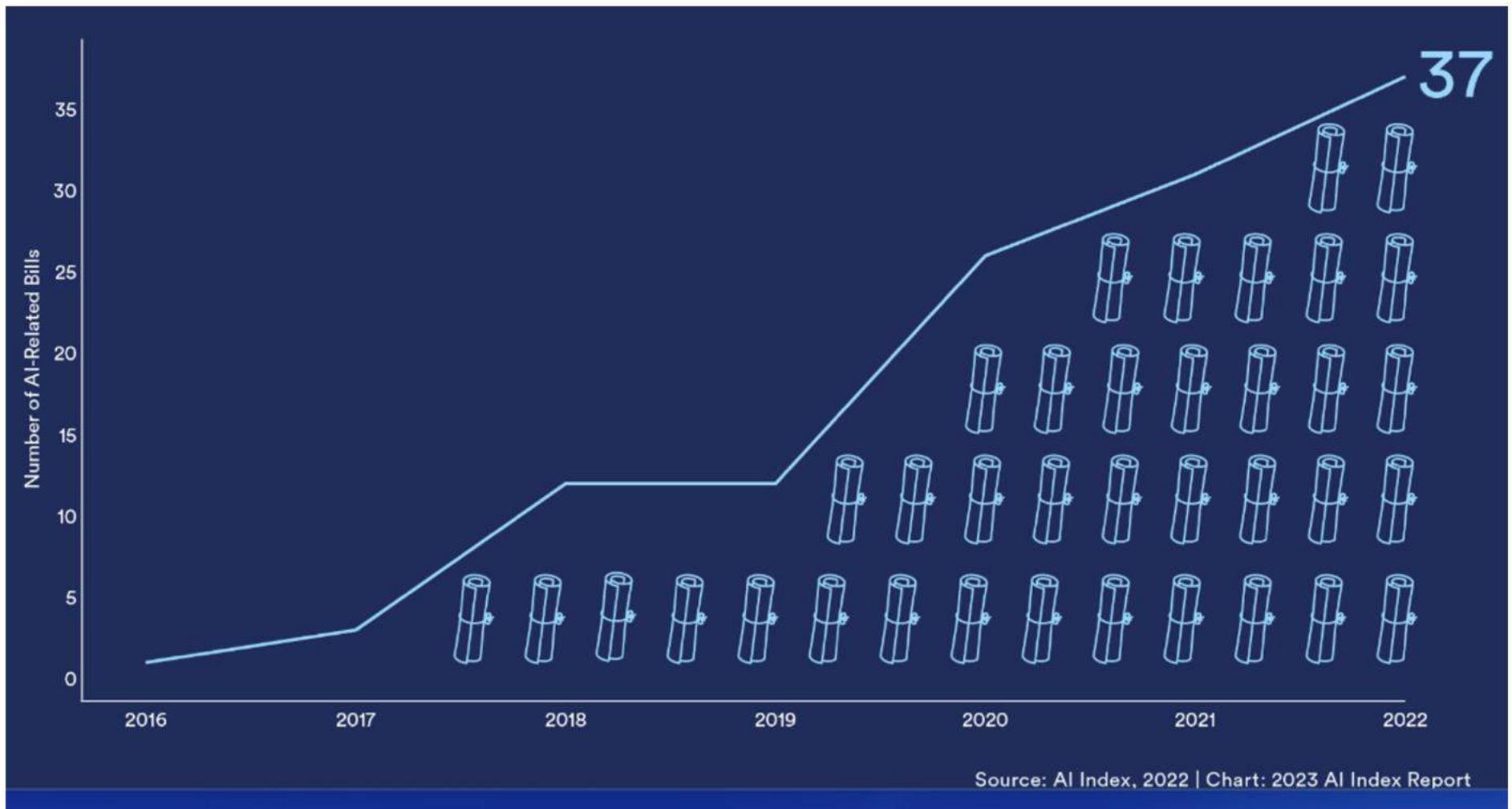
By David Weldon

Artificial intelligence is turning the worlds of technology, business and government on their heads. All face strategic conundrums as, with ever-increasing power and visibility, large language models and generative AI spread faster than operating rules can be set or ethical guardrails constructed.

There is general agreement about those needs. Tech industry leaders are urgently advocating regulation. Sam Altman, chief executive of the ChatGPT developer OpenAI, has joined with many others equating the risks with those of **pandemics and nuclear war**.

The European Parliament set out to be the first in the world with **wide-ranging legislation** mandating human oversight of AI innovation and risk-based restrictions on deployment. The U.S. Congress authorized a National Artificial Intelligence Advisory Committee (**NAIAC**) to advise the President, and the Biden administration has committed to “**responsible American innovation in artificial intelligence**” and outlined a **Blueprint for an AI Bill of Rights**.

Although legislative momentum has gradually been building around the world (see graph), it will take time for significant laws to be enacted and formally implemented. As a result, regulated sectors such as finance and health care must maintain compliance with legacy rules while also pushing the innovation envelope.



Stanford University's Institute for Human-Centered Artificial Intelligence ([HAI](#)) tallied the number of AI-related [bills passed into law globally](#). Banks may have a taste of what is to come, from their experience with [model risk management](#) and anti-bias provisions. The European Union's AI Act would allow for regulatory sandboxes, or protected zones for

experimentation, similar to programs that the [U.K. Financial Conduct Authority](#) and other agencies established over the last decade to foster fintech innovation.

The Monetary Authority of Singapore, which stands out among central banks and regulators as an innovation booster – including through sandboxes – is partnering with Google Cloud “to advance the development and use of responsible generative AI applications” in its digital services. “Through this, we hope to inspire greater adoption of responsible generative AI in the financial sector,” assistant managing director (technology) Vincent Loy [said on May 31](#).

Chatbot Alert

On the AI frontier, the U.S. Federal Trade Commission “will vigorously enforce the laws we are charged with administering, even in this new market,” chair Lina Khan vowed in a *New York Times* [guest essay](#).

AI tools “being trained on huge troves of data in ways that are largely unchecked” can unfairly lock people out “from jobs, housing or key services,” Khan warned. “Existing laws prohibiting discrimination will apply, as will existing authorities proscribing exploitative collection or use of personal data.” The FTC is on guard against the technology becoming concentrated in a small number of dominant firms or engendering “collusion, monopolization, mergers, price discrimination and unfair methods of competition.”

AI also risks “turbocharging fraud . . . Chatbots are already being used to generate spear-phishing emails designed to scam people, fake websites and fake consumer reviews – bots are even being instructed to use words or phrases targeted at specific groups and communities,” Khan wrote.

Citing a rise in customer complaints, the Consumer Financial Protection Bureau on June 6 published an “[issue spotlight](#)” on chatbots. “A poorly deployed chatbot can lead to customer frustration, reduced trust and even violations of the law,” said CFPB director Rohit Chopra.

The Securities and Exchange Commission’s Investor Advisory Committee in April [laid out for SEC Chair Gary Gensler](#) “perspectives on the importance of providing ethical guidelines for artificial intelligence and algorithmic models utilized by investment advisers and financial institutions.” Gensler [said on June 13](#) that AI, machine learning and predictive data analytics are being considered in the context of proposed brokerage conflict-of-interest rules that the agency is working on.

Deeper Into Modeling

Reggie Townsend, a NAIAC member who is director of the Data Ethics practice at analytics solutions leader SAS, says that according to his Risk Research and Quantitative Solutions colleagues, AI is 'increasingly being used in risk model development, especially for behavioral models."

For example, in engineering, "firms are using AI to find previously unrevealed relationships," Townsend says. "They are also using more data sources, including alternative data, to power these findings. The variables discovered through AI, however, are still being put into traditional regression techniques."

In addition, AI-derived models are used for benchmarks in challenging traditional models. AI is also "very good at anomaly detection and often used for data quality – both input and output," Townsend adds, as well as for early warnings when borrowers are in credit distress, when proactive customer engagement can prevent losses.

As Big Techs pour billions into AI initiatives (including Microsoft's reported \$10 billion investment in OpenAI), and Accenture this month [anted up \\$3 billion](#), SAS announced in May that it will invest \$1 billion over three years to develop advanced analytics solutions for multiple sectors using the cloud-native, massively parallel SAS Viya platform. The investment includes direct research and development, industry-focused line-of-business teams, and industry marketing efforts, and "is in line with the trend toward democratizing data analytics," the company said in its [announcement](#).

"Businesses face many challenges, from the threat of economic recession and stressed supply chains to workforce shortages and regulatory changes," said SAS CEO Jim Goodnight. "With insights from industry-focused analytics, resilient organizations can find opportunity in these challenges. Through this investment, SAS will continue to support companies using AI, machine learning and advanced analytics to fight fraud, manage risk, better serve customers and citizens, and much more."

Machines Versus People?

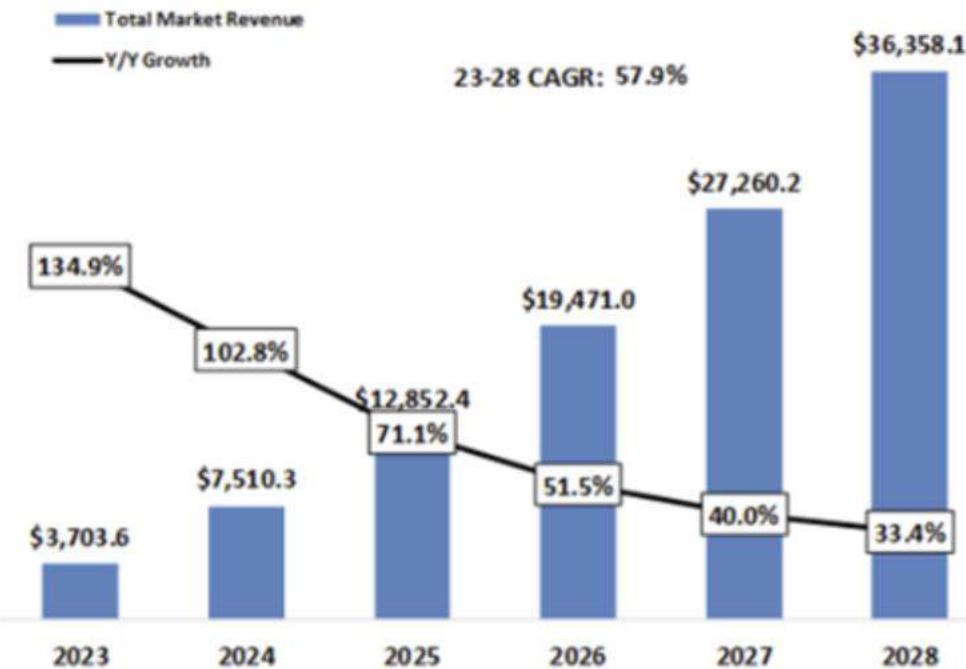
"Whose vision is driving AI? . . . What is the dominant narrative?" asked Massachusetts Institute of Technology professor Simon Johnson at a June 12 [Brookings Institution event](#). "It's about – I think – machines replacing humans."

Co-author with Daron Acemoglu, also of MIT, of [**Power and Progress: Our Thousand-Year Struggle Over Technology and Prosperity**](#), Johnson sought to put AI in that historical perspective. Replacing people, whether to win at chess or check out at supermarkets, "is far too skewed in one direction and one tradition of computer science," said Johnson, who gained renown for his post-Great Financial Crisis book [**13 Bankers**](#), written with James Kwak, and is co-chair of the [**CFA Institute Systemic Risk Council**](#).

Johnson added: "The other direction, which we prefer, is one in which you actually design machines and tend to machines to help humans, to augment human capacities and capabilities. If you do that, and change the vision, we argue you will be happier with the outcomes in terms of the jobs you create."

Investment risk analytics entrepreneur Richard Smith, now chairman and executive director of the [**Foundation for the Study of Cycles**](#), said in a LinkedIn post, "When it comes to new generative AI technologies like ChatGPT, there's a disconnect between what people think the new technology is and what it actually is. That's already led to the early stages of bubble formation."

Generative AI Market Revenue (\$M)



Generative AI software revenue will rise almost tenfold, to \$36 billion, in 2028, according to [S&P Global Market Intelligence](#).

On the whole, Smith tells GARP Risk Intelligence, generative AI is “going to improve productivity. It’s going to improve margins. I think that risk professionals absolutely will integrate AI into their practices, just like we integrated computers and data into our practices.

“Those risk professionals that really spend the time to invest in this technology, and to use it responsibly and in a measured way, will definitely succeed. The technology really can produce some extraordinary results, some extraordinary outputs.”

“Humans can’t monitor or analyze large volumes of data very quickly or effectively,” says SAS’s Townsend. “That’s where AI excels. Risk analyses that used to take days can be done in a fraction of the time, helping banks manage risk in near real time.

"AI's automation capabilities will prove to be a real game-changer for risk managers in more accurately assessing credit risk, doing more expansive stress testing, and better detecting fraud and financial crime, for example," Townsend continues. "That's also why it's so critically important to get the fundamentals right. You can't shortcut the data, models or processes."

Trust and Anti-Discrimination

On the downside, "In a heavily regulated industry like financial services, jumping into the current large language model craze might be taking on too much risk," Townsend cautions.

He notes that "there are proven technologies, with trustworthy AI capabilities built in, that will provide better peace of mind while still providing a competitive edge."

There must be sensitivity to how "minority populations have been impacted by laws, social norms and business practices, many of which were discriminatory at one time in our history," Townsend adds. "Unfortunately, many disparate outcomes still exist, primarily because those laws, norms and practices have a long tail effect, and encoding those same laws, norms and practices into our digital lives will only intensify them."

Lending or anti-fraud models trained on historical data can perpetuate various biases, not least the redlining practices that continue to haunt property valuation and mortgage decisions. Says Townsend, "It's vital to anticipate and remediate these risks, injecting positive bias where appropriate and putting humans at the center so models don't harm vulnerable people."

Alignment, Explainability, Cybersecurity

Avivah Litan, distinguished VP analyst at Gartner, comments on risk, legal and compliance challenges emerging in such areas as:

- Aligning models with human goals – "Regulators should set a timeframe by which AI model vendors can ensure their GenAI [generative AI] models have been trained to incorporate pre-agreed-upon goals and directives that align with human values. They should make sure AI continues to serve humans instead of the other way around."

- Explainability and transparency – “Even the vendors don’t understand everything about how [GenAI models] work internally. At a minimum, enterprises require attestations regarding the data used to train the model, e.g., regarding the use of sensitive confidential data” such as personally identifiable information or company business plans.”
- Disinformation, malinformation, misinformation – “This includes fake news that polarizes societies, undermines fair elections and democracies, causes personal harm, social unrest and other injuries. Enterprises should push regulators and vendors to set timeframes by which AI model vendors must use standards to authenticate provenance of content, software and other digital assets used in their systems.”
- Accuracy – “GenAI systems consistently produce inaccurate and fabricated answers called hallucinations. Enterprises and regulators must ensure that AI vendors provide users with the capability to assess outputs for accuracy, appropriateness and safety so that, for example, they can distinguish made-up facts from real ones.”
- Intellectual property and copyright – “There are currently no verifiable data governance and protection assurances regarding confidential or protected enterprise information. Enterprises and regulators should enforce: (1) Controls whereby AI vendors identify all copyrighted materials that are used in model operations (the EU AI Act is proposing such a rule); and (2) privacy and confidentiality assurances that AI vendors give to any constituent asking for them to ensure private data is not retained in the model environment.”
- Threat intelligence sharing – “Enterprises should seek out and participate in international agencies that share threat intelligence on malicious actors’ use of AI to inflict harm on targets. Once threats are detected, technical measures should be put in place to block or disarm the threats.”

Markets Are More Susceptible

The range of operational, moral, economic and legal questions, along with AI’s potential to compound risks instead of mitigating them, puts the financial services industry in uncharted territory, states Thomas Vartanian, executive director, [**Financial Technology & Cybersecurity Center**](#).

The longtime financial industry attorney and ex-regulatory agency official sees red flags in the resource and technical limitations that make it difficult for regulators to keep pace and take actions in real time; the inability of AI applications to meet consumer protection and model explainability needs; and inaccuracies in AI programming that may impact compliance and trading oversight.

"Increasing security threats facilitated by the use of AI can make government processes, payments systems, trading markets and exchanges more susceptible to misuse, abuse, attack and manipulation," warns Vartanian, whose most recent book is [**The Unhackable Internet**](#): How Rebuilding Cyberspace Can Create Real Security and Prevent Financial Collapse.

"We all certainly benefit from what AI can provide," Vartanian says. "But the benefits must be balanced against the threats that can emerge to economies, people and companies by embracing technology for technology's sake without a legal and moral framework in place. We should not suffocate innovation, but even more importantly, we should not blindly cede our futures to machines controlled by people who may be largely unaccountable."

Managing the Risk of Large Language Models Like ChatGPT

The fast-moving new technology requires risk assessments and policies that impose necessary control without stifling innovation

Friday, June 9, 2023

By Aaron Pinnick

Large language models (LLMs) like OpenAI's ChatGPT and Google's Bard are becoming increasingly popular among individuals and firms looking to take advantage of the incredible power and efficiency they offer for language-based tasks. These tools collect substantial amounts of text from various sources – including information entered by users into the tool – learn from that text, and then respond to user prompts with human-like responses.

Because these tools [**collect and learn from an incredible amount of data**](#) – often billions of parameters – they can be used for a wide range of tasks, from creating a simple email to writing complex programming code.

However, as a leak in March [of ChatGPT logs](#) demonstrated, these tools' retention of user inputs creates potential privacy and security risks for firms. And with the novelty of these tools and the excitement around them, employees are less likely to be aware of and think through the potential risks before using these tools for business purposes.

To mitigate these risks, cybersecurity leaders should take several steps, including:

1. Assess the risk LLMs create.
2. Update the firm's acceptable use policy.
3. Provide employees with training and communications on LLMs.

Assessing the Risk

Before taking a stance on whether or when LLMs will be permitted for business purposes, a firm should understand the risks these tools pose to the organization. While this assessment will vary from firm to firm, the risk centers primarily on how employees choose to use LLMs and the information they include in them.

The following potential risks should be considered:

- **Privacy Risk** – The most common risk LLMs present is that employees will enter sensitive information into the tool (e.g., client names and information), which is then exposed to the public. This risk will be high for most firms, as the novelty of LLMs means that employees are likely experimenting with the tools and may not exercise necessary caution when entering information. If this information is exposed, it may create reputational harm for the firm, and regulatory risk for companies in specific industries or jurisdictions. And regardless of a leak, uploading of certain types of data into an unapproved third-party tool (e.g., private health information) could be considered a privacy violation in certain jurisdictions.
- **Intellectual Property Risk** – Since LLMs are designed to learn from the inputs users provide, any proprietary or non-public information included in a prompt will be stored by the tool and integrated into future responses provided to other users. Even if proprietary information isn't directly leaked, the LLM could be prompted to respond as if it were an employee at a certain company and, based on what the LLM has learned through past interactions with employees at that company, provide non-public information back to a user. Through this process, individuals could gain insights into the strategic

direction of competitors or gain non-public information about the products and services of companies whose employees use an LLM.

- **Third-Party Risk** – Since a core feature of LLMs is their ability to quickly generate extremely large amounts of text, individuals may try to use an LLM as a shortcut in creating client deliverables. Firms should confirm with key vendors if LLMs are used to create any work product or advice they receive. If a third party is using LLMs, the firm should understand what company information is entered into the LLM, as well as how deliverables are screened for quality and accuracy prior to receiving them. Analogously, employees using LLMs in their work for clients may be violating the letter or spirit of agreements with those clients.
- **Risks Related to the Quality of the Output** – LLMs are often used by individuals to help generate ideas or first drafts of documents or code. But despite the impressive performance of many LLMs, they will make mistakes in their output, with tools like ChatGPT having issued warnings about the chance that it may produce incorrect information. These risks can be mitigated by having an expert review the materials created by an LLM to ensure that they are accurate. But if there isn't sufficient oversight or controls around how information created by an LLM is used in a final work product, incorrect information may be shared with internal or external stakeholders, leading to potentially flawed decisions and compliance issues.

It is important to note that LLMs pose a broader risk for cybersecurity executives, as cybercriminals can easily use these tools to create compelling dialogue, phishing email language, and code to improve the effectiveness of cyberattacks. Cybersecurity leaders should be aware of this threat, and ensure that the firm's policies, procedures, and employee training take this into account.

Update Acceptable Use Policy

Firms should review and ensure that Acceptable Use Policies (AUPs) specify when and how employees will be permitted to use LLMs on company devices and for business purposes.

Firms may take several different approaches towards building an AUP for LLMs based on their risk tolerance and on the opportunities these tools present. These options include:

- **A Total Ban on LLMs** – The most conservative approach to LLMs is to simply block the likes of ChatGPT and Bard on company devices, and to update the AUP to make clear that employees' use of these tools

is not acceptable for any reason.

While a ban may be appropriate for firms that deal with highly confidential client or company data, it may be difficult to maintain as the number of LLMs available is quickly growing. It also may prove challenging as LLMs are increasingly incorporated into products like Microsoft Teams and other productivity tools. The effort of keeping up with this LLM growth is likely not worth it for most firms, as it also means that the company won't be able to take advantage of the benefits of LLMs.

- **Restricted Use of LLMs** – Firms that are willing to accept some risk from LLMs in exchange for the potential efficiency gains can allow for their use under certain conditions. These could include:

- For business purposes only if no sensitive, proprietary or confidential information is included in LLM prompts.
- For business purposes only with approval from specific individuals (e.g., chief information security officer, business unit head, general counsel).
- Only for certain low-risk business activities (e.g., for help writing marketing copy about features and services that are publicly available on the firm's website)
- LLMs cannot be used to create client-facing work products, or to generate guidance for clients.
- LLM use is allowed for business purposes, but records must be kept of the prompts and outputs from the tool.

For most firms, some combination of the above clauses and restrictions should help mitigate LLM-related risks without stifling the tools' innovative potential.

- **Reasonableness Standard** – Firms that see the greatest potential in LLMs and are willing to accept the highest level of risk may allow employees to use their best judgment when working with these tools. Those adopting this approach can take a page from their existing training and policies on social media usage to promote good judgment around what information is and isn't appropriate for LLMs. Employees should be reminded that information entered into LLMs should not be assumed to be private or secure,

and nothing that would cause reputational harm to the employee, the firm, or the firm's clients should be entered into an LLM.

Employee Training and Communications

Since the core risk posed by LLMs is rooted in employee behavior, it is critical to provide clear guidance on when or if these tools are allowed to be used for business purposes. Because LLMs are a hot topic, it is likely that employees have already begun experimenting with them at work or on their personal devices, so cybersecurity leaders shouldn't wait to begin making these updates.

Firms should take the following steps:

- **Don't Wait to Communicate** – Even if the firm hasn't settled on a final AUP, it is critical that employees think carefully about what information is entered into an LLM. Senior leaders should immediately begin notifying employees of the risk these tools pose and remind them of basic standards such as never to enter client information into an LLM.
- **Update Training** – Once the firm has settled on its acceptable use standards, the firm's cybersecurity training should be updated to include guidance on how employees can follow the AUP. This will include ensuring employees are aware of the policy, providing them with clear examples of what is and is not appropriate, and ensuring that employees understand the risks associated with violating the AUP.
- **Reinforce the Policy** – As with all behavioral risks, employees will need to be reminded of the AUP on LLMs. Cybersecurity leaders should begin integrating reminders about appropriate and inappropriate uses of LLMs into their communications calendar to employees, to help keep the risk front-of-mind for employees. Adding an interstitial page that employees have to click through to access LLMs on the web creates another opportunity for policy reminders.

Conclusion

Tools like ChatGPT and Bard present firms with a unique opportunity to create efficiencies in their workforce. When used properly, they can automate a wide range of time-consuming and labor-intensive writing tasks, freeing up time for employees to focus on higher-value work. But like all new technologies, LLMs pose additional risks from misuse.

The good news for cybersecurity leaders is that they likely have experience managing employee-centered risk, and there likely isn't a need for radical new approaches to dealing with it. Cybersecurity leaders should immediately assess the risk that LLMs pose to their firm and develop policies, training and communication to guide employee behavior. Because of the rapidly evolving nature of LLMs, that guidance may need to be reviewed and updated more frequently, but the program's approach to this risk should be straightforward.

Aaron Pinnick is the Manager of Thought Leadership for **ACA's Aponix Program**. In this role, he creates research to ensure clients receive the latest and most critical information they need to manage risk and ESG responsibilities. Before joining **ACA Group**, he was a Managing Analyst for Ballast Research, providing government affairs leaders with insights into their reputation with policymakers; and a research director for Gartner's Compliance and Ethics program, creating research and best-practice guidance for compliance leaders at some of the world's largest companies. Pinnick holds a master's degree in sociology from Texas A&M University and a bachelor's in sociology from Minot State University.

A version of the above article was previously published on [the ACA website](#).

Machine Learning and Risk Management: A Q&A with Professor John Hull

Early-career and aspiring risk managers don't necessarily need to become data scientists to compete for job opportunities and advancement at tech-savvy financial institutions. But those who learn the Python programming language, and who possess the communication and technical skills necessary to not only understand ML modeling but to explain it clearly to others, will have a leg up on their competitors, says a well-known academic.

Friday, August 5, 2022

By Tod Ginnis

Machine-learning (ML) models have become increasingly prominent within financial services firms, and their adoption is unlikely to slow anytime soon.

What does this mean for the employment prospects and necessary skill sets for risk managers? How can they navigate the complex challenges emerging from disruptive technologies?

To help answer these questions, we spoke with [**John Hull**](#), Professor of Finance at the University of Toronto's Rotman School of Management. Hull, the recipient of the 1999 Financial Engineer of the Year Award from the International Association for Quantitative Finance, currently teaches ML to graduate students targeting financial careers. His book, "Risk Management in Business: An Introduction to the World of Data Science," is used in the courses he teaches.

Tod Ginnis (TG): How can early career or aspiring risk managers prepare for a future that incorporates ML into their work?

John Hull (JH): Risk managers don't need to become data scientists, but they will need to understand how the technology works and how to explain it to others. It's important to have the technical skills to know what data scientists do and to ask the right questions. You want to know enough about the tools to know when they are appropriate and when they are being abused.

I like to make an analogy with accounting: we teach accounting to our MBA students not because most will become accountants, but so that they can work productively with accountants. Similarly, future risk managers need to know enough about machine learning to work productively with data scientists.

Many libraries have been developed in [**Python**](#) for implementing machine-learning algorithms. I require the students in my machine-learning courses to learn enough Python to do illustrative assignments. Python is becoming almost as prevalent as Excel in financial institutions, and aspiring risk managers should definitely consider learning it.

TG: As some functions of the profession become increasingly automated, how do you envision the role of risk managers working with ML?

JH: I believe we're in the early stages of a fourth industrial revolution, which includes AI, ML, and creating intelligence from data. There are clear risks associated with it.

Can we maintain complete control over the intelligence we're creating? It will be years before we know. What we do know is that it's important to have human beings in charge.

Some people say that if you have a good ML model, you can just let it go and fire everyone who used to do the job. But oversight is important. Periodically, you must confirm your model is still appropriate.

ML uses historical data, and when we run into unprecedeted situations — like the pandemic — the data may no longer be appropriate. It is important for risk managers who understand machine learning to periodically review the appropriateness of the models that have been developed.

TG: In what ways do ML models fall short in meeting an institution's goals regarding risk management?

JH: Explainability is a big issue. Consider a credit risk model deciding loan approvals. The person using the model must understand in general how it works. It is also important to be able to communicate the results of the model to the potential borrower, especially when the loan is refused. It's not enough to say, "the algorithm says no." You've got to explain the decision in normal language.

Bias is another challenge that can occur in multiple ways. It can relate to the data that you've used to build your model or biases on the part of the model builder.

Biases can also result from **the features you've used as inputs to the model**. Using race and gender are obviously unacceptable, but you must be careful, because the model might include features that are highly correlated with race or gender. So, even though you haven't explicitly used them as features, you could be introducing bias into your algorithm.

Another key factor is what statisticians call **testing the model in-sample vs. out-of- sample**. The danger is that somebody develops a model and says, "the model fits the data really well. Let's use it to make predictions." That type of model may work great on the data that's been collected, but it hasn't been tested on new (i.e., out-of-sample) data.

There are procedures for using ML in risk management that try and overcome this problem. It's important that the risk manager understands those procedures and asks whether they've been followed properly. If the data scientist hasn't tested out-of-sample, then you shouldn't use the model.

TG: What specific skills or qualifications might make a new risk management hire appealing for financial institutions looking to better utilize ML?

JH: One thing we emphasize to our students is that communication skills are critical. And technical skills are becoming more important. Demonstrate that you've got some of these technical skills (like expertise in Python) we've been talking about.

Taking the FRM is another good idea. There are two chapters in the FRM study materials on ML that I assisted with. Working through those chapters will help give future risk managers the tools they need to work effectively with data scientists.

TG: What advice do you have for early career and aspiring risk managers who might be unsure if they're ready for a brave new world filled with advanced technology?

JH: I remember the 1970s and the 1980s, when we were going through the third industrial revolution, namely digitization and computers. Some people embraced the changes and studied them. Others put their head in the sand. The people who did well in business tended to be those who took on the challenge. Most didn't become programmers, but they learned how to use computers.

Risk managers should learn enough about the technology to understand what the data scientists are doing and how it may impact the organization.

Risk managers can be important intermediaries, because it's part of their task to understand all the models the organization uses. As questions arise about particular models, they should be able to explain things in clear language to all the relevant stakeholders, including senior management. This will make the job of a risk manager increasingly important, as the algorithms become more complex.

Tod Ginnis is a content specialist at GARP. He is the author of a GARP blog that is aimed at early-career risk managers and professionals aspiring to earn their [Financial Risk Manager \(FRM\)](#).

Regulatory Standards & Guidance

| | |
|---|--|
| <u>AI Risk Management Framework: Generative Artificial Intelligence Profile</u> | This document is a companion resource for Generative AI to the AI Risk Management Framework (AI RMF). It serves as both a use-case and cross-sectoral profile of the AI RMF to assist organizations in deciding how they might best manage AI risk in a manner that is well-aligned with their goals, considers legal/regulatory requirements and best practices, and reflects risk management priorities. |
|---|--|

| | |
|--|---|
| NIST "crosswalk" between the NIST AI Risk Management Framework (AI RMF) and the Japan AI Guidelines for Business (AI GfB) | NIST AI Risk Management Framework (AI RMF) Crosswalks are produced by NIST or other organizations and are intended to provide a mapping of concepts and terms between the AI RMF and other guidelines, frameworks, standards, and regulation documents. |
| NIST AI Risk Management Framework | In collaboration with the private and public sectors, NIST has developed a framework to better manage risks to individuals, organizations, and society associated with artificial intelligence (AI). The NIST AI Risk Management Framework (AI RMF) is intended for voluntary use and to improve the ability to incorporate trustworthiness considerations into the design, development, use, and evaluation of AI products, services, and systems. |
| U.S. General Services Administration: AI Guide for Government | A living and evolving guide to the application of artificial intelligence for the U.S. federal government. |
| White House Blueprint for an AI Bill of Rights | This is a guide for a society that protects all people from artificial intelligence threats—and uses technologies in ways that reinforce our highest values. It provides guidance whenever automated systems can meaningfully impact the public's rights, opportunities, or access to critical needs. |
| The EU Artificial Intelligence Act | The AI Act is a proposed European law on artificial intelligence (AI) – the first law on AI by a major regulator anywhere. The law assigns applications of AI to three risk categories. First, applications and systems |

| | |
|---|--|
| | that create an unacceptable risk, such as government-run social scoring of the type used in China, are banned. Second, high-risk applications, such as a CV-scanning tool that ranks job applicants, are subject to specific legal requirements. Lastly, applications not explicitly banned or listed as high-risk are largely left unregulated. |
| OECD.AI Policy Observatory | OECD.AI Policy Observatory serves as a global hub for AI policy. It combines resources from across the OECD, partners and stakeholder groups to create a one-stop-shop for AI policymakers and other actors. |

Recommended Papers

| | |
|--|--|
| Machine Learning and Model Risk Management by Peter Quell, Anthony Graham Bellotti, Joseph L. Breeden, and Javier Calvo Martin | Machine Learning and Model Risk Management reviews the risks and challenges of using machine learning in lending. Current best practices in explainability and validation are discussed. |
| AI and Machine Learning for Risk Management by Saqib Aziz and Michael Dowling | This paper explores how artificial intelligence (AI), and machine learning solutions are transforming risk management. A non-technical overview is first given of the main AI and machine learning techniques of benefit |

| | |
|--|--|
| | <p>to risk management. Then an analysis, using current practice and empirical evidence, is carried out of the application of these techniques to the risk management fields of credit risk, market risk, operational risk, and compliance.</p> |
| <p>Machine Learning in Banking Risk Management: A Literature Review by Martin Leo, Suneel Sharma, and K. Maddulety</p> | <p>This paper, through a review of the available literature seeks to analyze and evaluate machine-learning techniques that have been researched in the context of banking risk management, and to identify areas or problems in risk management that have been inadequately explored and are potential areas for further research</p> |
| <p>Machine Learning for Financial Risk Management: A Survey by Akib Mashrur, Wei Luo, Nayyar A. Zaidi, and Antonio Robles-Kelly</p> | <p>This paper provides a systematic survey of the rapidly growing literature of machine learning research for financial risk management. The contributions of the paper are four-folds: First, it presents a taxonomy of financial-risk-management tasks and connects them with relevant machine learning methods. Secondly, it highlights significant publications in the past decade. Thirdly, it identifies major challenges faced by researchers in this area. And finally, it points out emerging trends and promising research directions.</p> |
| <p>A Primer on Artificial Intelligence and Machine Learning for the Financial Services Industry by Joerg Osterrieder</p> | <p>This primer provides a comprehensive overview of the applications of Artificial Intelligence (AI) and Machine Learning (ML) in the financial services industry. It covers the various use cases of AI and ML, including fraud detection, customer behavior analysis, loan underwriting,</p> |

| | |
|---|---|
| | investment management, algorithmic trading, and risk management. |
| Artificial Intelligence and the Economy: Implications for Central Banks by Hyun Song Shin | Hyun Song Shin, Economic Adviser and Head of Research of the Bank for International Settlements (BIS), discusses the benefits and challenges for central banks of the latest artificial intelligence technology. |
| Large Language Models: A Survey by Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, Jianfeng Gao | This paper reviews some of the most prominent LLMs and discuss their characteristics, contributions and limitations. It covers techniques developed to build, and augment LLMs, popular datasets prepared for LLM training, fine-tuning, and evaluation. It discusses widely used LLM evaluation metrics and compare the performance of several popular LLMs on a set of representative benchmarks. |
| Attention Is All You Need by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin | This 2017 landmark research paper authored by eight scientists working at Google introduced the deep learning architecture known as the transformer, and is considered by some to be a founding paper of modern artificial intelligence. |
| Computing Machinery and Intelligence by Alan Turing | "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think". The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words "machine" and "think" are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, "Can |

| | |
|---|--|
| | machines think?" is to be sought in a statistical survey such as a Gallup poll. |
| <p>A Risk Management Framework for Large Language Models by Alec Crawford and Frank Fitzgerald</p> | This paper offers a framework for risk management of artificial intelligence systems, with a focus on Large Language Models (LLMs). The paper covers Artificial Intelligence (AI) governance, risk, compliance, and cybersecurity (AI GRCC) with a focus on the risks associated with AI implementation in medium and large organizations. |

Recommended Books

| | |
|--|--|
| (Free) Artificial Intelligence in Asset Management by Sönke M. Bartram, Jürgen Branke, and Mehrshad Motahari | This study provides a comprehensive overview of a wide range of existing and emerging applications of AI in asset management, highlighting the key topics of debate. It focuses on three major areas: portfolio management, trading, and portfolio risk management. |
| (Free) Handbook of Artificial Intelligence and Big Data Applications in Investment by Larry Cao | Artificial intelligence (AI) and big data have their thumbprints all over the modern asset management firm. Like detectives investigating a crime, the practitioner contributors to this book put the latest data science techniques under the microscope. And like any good detective story, much of what is unveiled is at the same time surprising and hiding in plain sight. |
| (Free) Introduction to Python for Econometrics, Statistics | These notes are designed for someone new to statistical computing wishing to develop a set of skills |

| | |
|---|--|
| <p>and Numerical Analysis: Fourth Edition by Kevin Sheppard</p> | <p>necessary to perform original research using Python. They should also be useful for students, researchers or practitioner who require a versatile platform for econometrics, statistics, or general numerical analysis.</p> |
| <p>(Open Access) Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter, 3rd edition by Wes McKinney, O'Reilly</p> | <p>Written by Wes McKinney, the creator of the Python pandas project, this book is a practical, modern introduction to data science tools in Python. It's ideal for analysts new to Python and for Python programmers new to data science and scientific computing.</p> |
| <p>(Free) Deep Learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville</p> | <p>This textbook is a resource intended to help students and practitioners enter the field of machine learning in general and deep learning in particular.</p> |
| <p>(Free) Neural Networks and Deep Learning by Michael Nielsen</p> | <p>This book provides a theoretical background on neural networks and deep learning, covering many of the core concepts behind them.</p> |
| <p>Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto</p> | <p>Reinforcement learning, one of the most active research areas in artificial intelligence, is a computational approach to learning whereby an agent tries to maximize the total amount of reward it receives while interacting with a complex, uncertain environment. In Reinforcement Learning, Richard Sutton and Andrew Barto provide a clear and simple account of the field's key ideas and algorithms. This second edition has been significantly expanded and updated, presenting new topics and updating coverage of other topics.</p> |

Podcasts

| | |
|--|---|
| <u>Generative AI: applications in the Quant Investment Process</u> | This episode explores some of the more direct applications of Generative AI, specifically ChatGPT within a quantitative investment process. Topics include Natural Language Processing signals, product strategies and client communications. |
|--|---|

Data Resources

| | |
|--|---|
| <u>Compustat</u> | Compustat is a database of financial, statistical, and market information on active and inactive global companies throughout the world. |
| <u>CRSP (Center for Research in Security Prices)</u> | Historical stock market data, widely used for academic research. |
| <u>Federal Reserve Economic Data</u> | Macro and financial data for the US and many foreign countries. |
| <u>IMF Data</u> | Macroeconomic and financial data from around the world. |
| <u>Kaggle</u> | Kaggle is a data science competition platform and online community of data scientists and machine learning practitioners. It also hosts datasets uploaded by its community, including financial datasets. |
| <u>Kenneth R. French - Data Library</u> | Historical stock market data. |

| | |
|---|--|
| Robert Shiller Online Data | Historical stock market data. |
| World Bank Open Data | Free and open access to global development data. |
| Yahoo Finance | A free resource that provides data on companies, including historical prices and financial statements. |