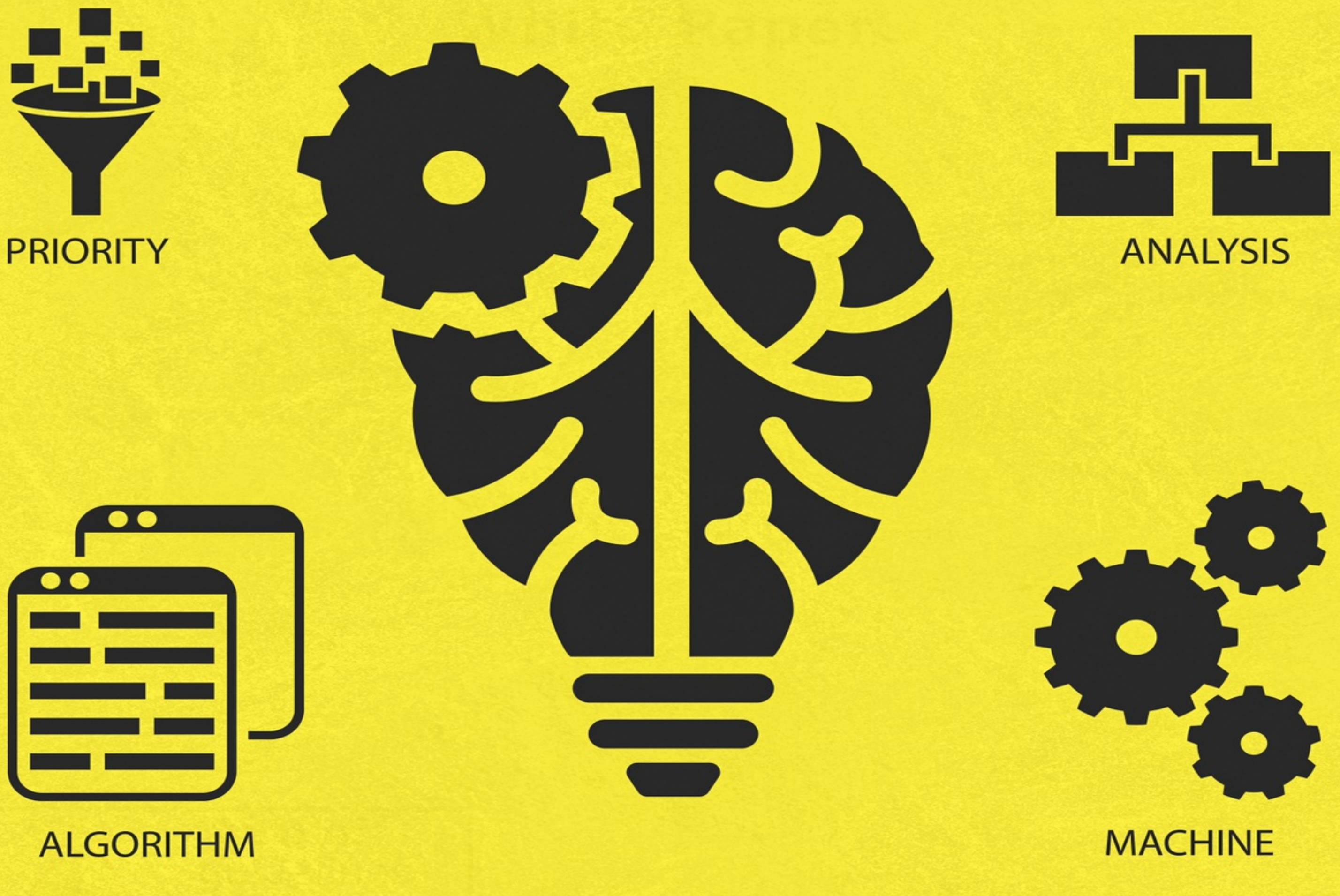
MACHINE LEARNING

2 BOOKS IN 1

AN INTRODUCTION MATH GUIDE FOR BEGINNERS TO UNDERSTAND DATA SCIENCE THROUGH THE BUSINESS APPLICATIONS



SAMUEL HACK

MACHINE LEARNING

2 Books in 1:

An Introduction Math Guide for Beginners to Understand Data Science Through the Business Applications

Samuel Hack

TABLE OF CONTENTS

YOUR FREE GIFT

As a way of saying thanks for your purchase, I'm offering a free report that's exclusive to readers of "Machine Learning: 2 Books in 1"

With "6 Important Steps to Build a Machine Learning System" you'll understand what happens before training a model and after training the model and deploying it in production.

The essentials to start from scratch!!

>> Tap Here to Grab "6 Important Steps to Build a Machine Learning System" <<

MACHINE LEARNING FOR BEGINNERS

A Math Guide to Mastering Deep Learning and Business Application. Understand How Artificial Intelligence, Data Science, and Neural Networks Work Through Real Examples

Introduction

Congratulations on purchasing Machine Learning for Beginners and thank you for doing so.

There are many opportunities opening up in the field of machine learning. It’s being adopted as a tool by almost every major industry. Whether you are interested in health care, business and finance, agriculture, clean energy, and many others, there is someone utilizing the power of machine learning to make their job easier.

Unfortunately for these industries, but fortunate for you is that there is a major shortage of talent in the field of data science and artificial intelligence. While entry-level data science jobs remain competitive, there is a major shortage of experienced data professionals who can fill the high-level roles. It’s a newer field in computer science, with a younger group of individuals who make up for much of the field.

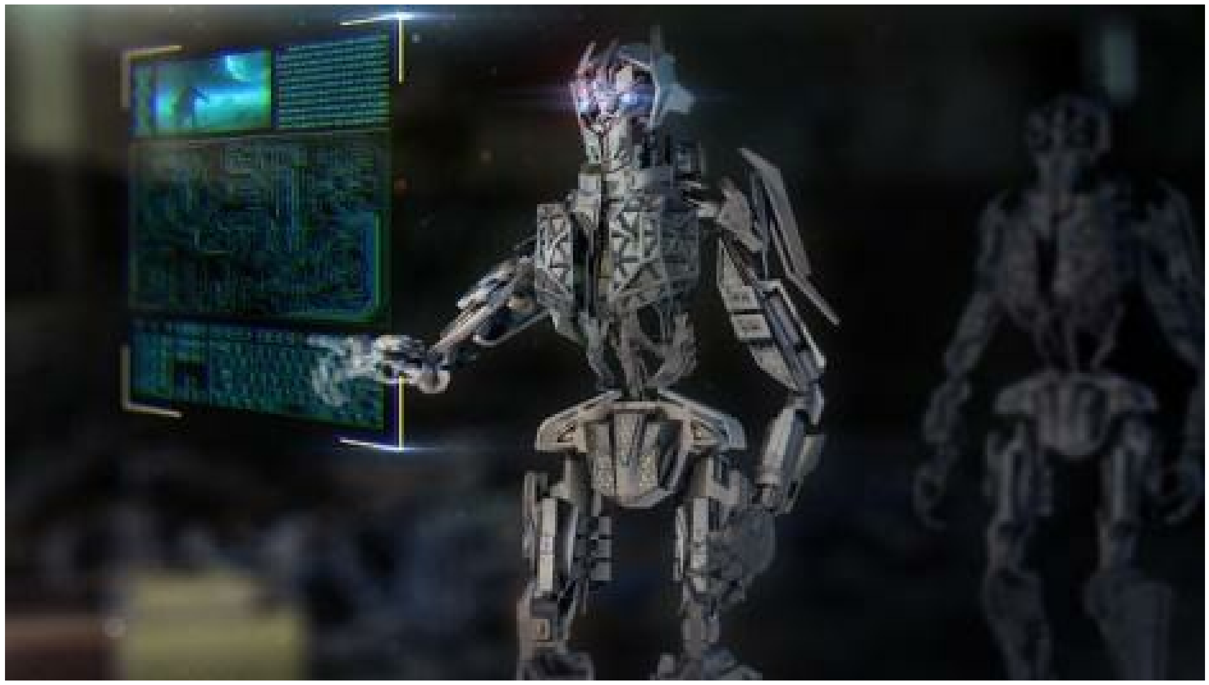
It can be very financially rewarding if you manage to land a job in data science. In 2016 the average data scientist made about $111,000, with predicted growth over the next five years. About half of data scientists working in the field have a Ph.D. It’s not a requirement, but it’s something to consider if you are looking into starting a real career as a data scientist.

If you are looking to add machine learning to your wheelhouse, so that you can have a better understanding of it and implement it in your own business or projects down the road, then a Ph.D. may not be necessary. But for those looking to enter the field, higher education is recommended as it will help you stand out amongst the field.

Indeed.com called machine learning the best career in 2019, and it’s easy to see why. With a huge demand for talented data scientists and a lucrative payout, it's worth a look. And big data doesn’t seem to be going away anytime soon with an increase in connectivity and higher than ever internet usage by both consumers and companies alike. Data is a part of our modern world, and as the complexity and size of data increases, it will take even more specialized knowledge and skills to be able to complete the task at hand.

To supplement the knowledge in this book, I highly recommend seeking further knowledge in statistics and programming. A good base of statistical knowledge is required to perform any work in machine learning because statistical mathematics provides the structure and justification for all the models and algorithms that data scientists use for machine learning.

The Purpose of This Book



This book is not meant to be a comprehensive textbook on machine learning. Instead, it will give you a base of knowledge to continue with your study of machine learning and artificial intelligence. In order to continue your studies and master the subject, there is a large degree of studying that must be done. Will discuss the general structure and organization of machine learning models, the common terms, and the basic statistical concepts necessary to use and understand machine learning.

It’s necessary to have a solid understanding of statistics and quantitative analysis to be a data scientist. After all, artificial intelligence and machine learning are rooted in statistics. This provides the anchor and foundation for the kind of mathematics needed.

While coding is not required to understand this book, it is a major component of machine learning. In order to handle large volumes of data, data scientists need to have a working knowledge of computer programming to ‘tell’ the data what they want it to do. This book will not offer much in the way of coding information, but it will present resources and avenues to get you started in studying coding on your own. By the end of the book, I will at least assist you in setting up Python with the necessary libraries and toolkits to help you start learning to code.

The most common language used in machine learning is Python. It’s a versatile language that is relatively easy to learn and freely available. There are Python packages designed for data analysis to make your coding go faster. C++ is also quite common but more difficult to master. A third option is R, which is quite popular because it is free and open source. Students often use it because of its availability and simplicity. The drawback of using R is that it can’t handle massive datasets often used in machine learning and artificial intelligence, which is somewhat limiting.

Computers in machine learning distinguish themselves by being able to not only memorize new information but apply that information to new situations in the future. There is a difference between memorizing and learning. There is an important distinction between giving a computer a line of code and creating a machine learning model.

The basic characteristic of machine learning is the use of artificial inductive reasoning. Artificial inductive reasoning means that a specific event gives you cause to generalize a characteristic. This apple is green; therefore, all apples must be green. But here you can see why inductive reasoning on its own is not always perfect, and why it’s difficult to train computers to have the same thought process. One given piece of data is not necessarily representative of thousands of other possible pieces of data. Therefore, when we are using statistics and machine learning, we must be using enough data to be able to reason with confidence, without making the wrong inference based on data that is misinterpreted and becomes misleading.

There are things we do every day as humans that we think of as ‘common sense.' These types of intuitive decisions cannot be explicitly programmed in a computer, because the variables that help us make our decisions are too difficult to measure. We probably don't need to see a thousand different combinations of chess pieces on a chessboard to think ahead and plan when we are given a situation we haven't seen before. We, as humans, require much fewer data to be able to infer and learn.

This is where machine learning comes in. These situations where the variables are complex, and directions can’t be explicitly stated have to be learned instead. Back to the example of a computer that can play checkers. It would take way too long to teach someone to play checkers by giving them every possible move and every possible countermove. Instead, you teach someone the basics, and by playing the person learns what helps them win and what doesn’t.

In the same way, it would be impossible to tell your computer every possible situation in a checkers game, and then tell the computer what it needs to do in each situation. There are far too many possibilities. Instead, you must give the computer enough data so that even when it is met with a new situation, it can respond accordingly.

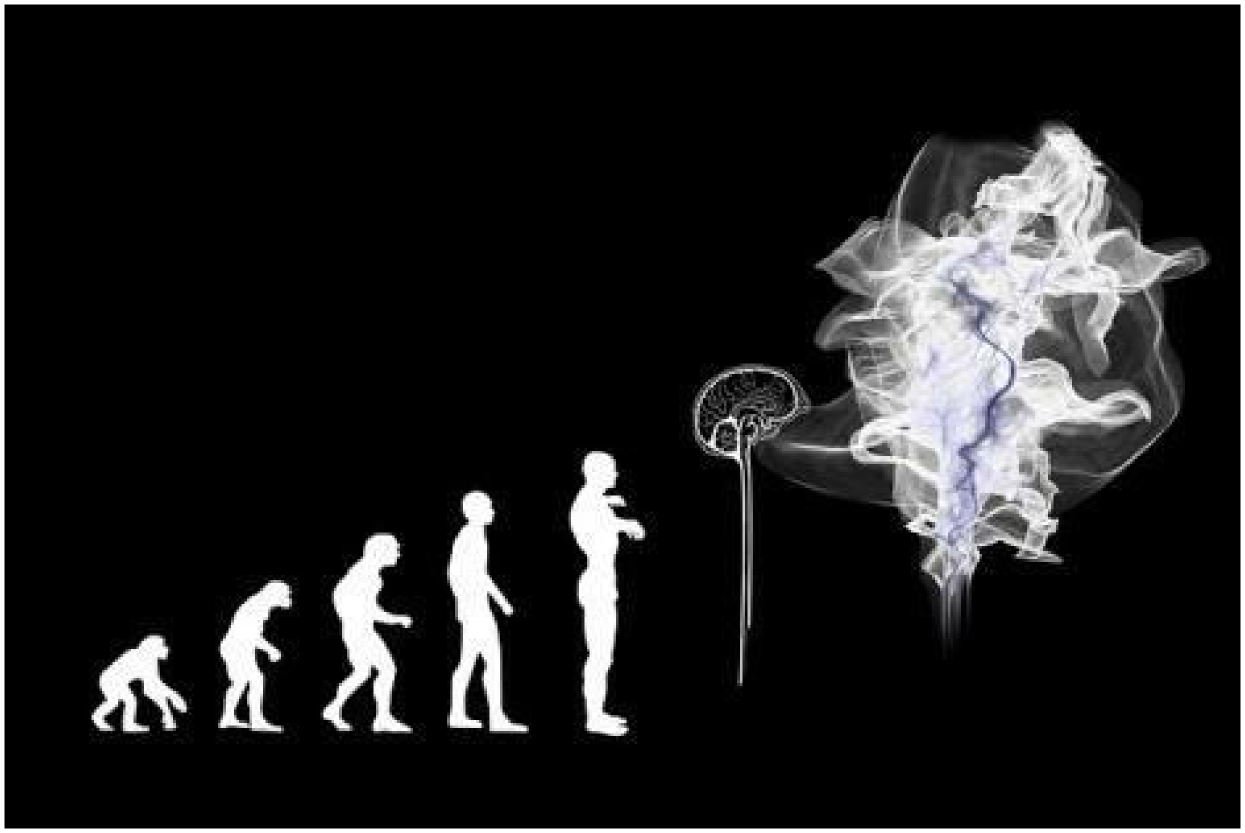
Another example we will talk about later in this book uses artificial neural networks to sort whether a photo is a picture of a cat or a dog. As humans, this type of classification would be too easy. We know what a dog looks like because we have seen dogs, and we know what a cat looks like because we have seen cats.

But there is no way to explicitly tell a computer how it can tell the difference between a cat and a dog. Instead, you give the computer a set of training data with pictures of cats and dogs, and you tell the computer which pictures are of cats, and which pictures are of dogs. Eventually, the model should be able to tell you whether new, unseen pictures have a feline or a canine in them.

The problem with explicitly programmed instructions is their inability to change. If I tell a computer exactly what to do, using a programming language with explicit instructions, then the program will do a very good job of performing that task, but only that task. It won’t change when it gets new information, and it won’t alter its method if it performs incorrectly

Over time, a machine learning model will be able to change itself as the data changes so that it continues to adapt and remain accurate in a changing environment without guidance. This offers a huge advantage because it makes our models more adaptable to change, which is constantly around us. Without machine learning and artificial intelligence, our computers wouldn't have a way to keep up.

What is Artificial Intelligence?



In the 1950s, separate researchers began developing the first artificial intelligence machines. Before then, only minor experimenting had been done in artificial intelligence; particularly in code-breaking during the second world war. It was an emerging field, and only some people seemed to be aware of the potential in the beginning

Now, artificial intelligence is used in a variety of applications in multiple sectors involving problem-solving, learning, planning, reasoning, and logic. It enables computers to perform tasks which normally require human thinking to perform. To 'think' like a human, computers require data from which they can learn.

Artificial intelligence holds an almost mythical place in people's minds. I bet if you said the words artificial intelligence to most people, there would be images of robots walking around acting like humans. This kind of science-fictionalization of artificial intelligence makes people wary when they hear the term. But, it's not as scary as it sounds. It has done a lot of good in medicine and business, transportation and communication. Although there have been impressive strides in the field, the misconception of a sentient computer is still a long way off. Still, the advent of self-driving cars and computers and phones that can talk stirs the imagination.

Although it sounds more like something from science fiction, artificial intelligence is in so much of our daily lives now. If it sounded spooky to you at first, let me remind you of all the technology that artificial intelligence has brought into our lives in recent years.

Last time you turned on Netflix, you were browsing a list that took the shows you watched and the movies you re-watched. It turned that list into data and created another list of recommendations. Movies it predicted that you would enjoy based on what you've already enjoyed. This is done by machine learning, a subset of artificial intelligence.

If you have a smartphone, you may use voice commands to search for things hands-free. You tell your phone that you're looking for cafes in your area and your phone says, 'searching for cafes in your area.' In a matter of seconds, a list of results appears, and you didn't even need to type anything. It recognized your voice and understood what you said. This is a part of natural language processing, another subset of machine learning. Every time you open your email account and you label spam; your email host is learning how to do a better job identifying spam. This is another type of machine learning.

So, artificial intelligence is not necessarily sentient robots who want to take over like as we know it. As of now, it's much more benign than that. It's also extremely helpful, and it's capable of learning things that we can't explicitly program it to do.

Artificial intelligence requires something called artificial reasoning, otherwise known as machine reasoning. When humans learn new things and draw conclusions, we go through a process known as inductive reasoning. We take pieces of information to draw new conclusions. Usually, there is no set rule that we are taught to go by. We learn from experience and draw our own rules by cumulative experience. For example, I could tell you that it snowed 15 times last December. Therefore, it will snow this December again. Every day in January was cold, so every day this January will be cold. So, I should bring a jacket.

We weren't born hard-wired to know that there would be snow every December, or that January is cold. We learned these things through experience and used inductive reasoning to generalize about future

December’s and January’s. Based on our inductive reasoning, we make the logical decision to prepare ourselves and take a coat with us next winter.

The experiences we had where we say snow in December, and we were cold in January represent our ‘data’. These are the inputs from our environment that we are constantly learning from.

Humans think differently from machines because we don’t interpret numerical data patterns. We learn from positive and negative rewards and from the feelings we experience in our daily lives. Getting a computer to use inductive reasoning will get us closer to having ‘human-like’ machines

So, for computers to learn, they need to have data to learn from. Data usually needs to be numerical, so that it can be interpreted by mathematical models and algorithms. If we give a computer enough data, it will create the parameters to design its own model or algorithm, to predict new situations based on prior experience. This is the basis of machine learning. Feed the computer experience so that it can predict new outcomes in the future through inductive reasoning.

Artificial intelligence is especially interesting because computers are already better than humans at some tasks. They can draw mathematical conclusions on a dataset with thousands of inputs in a matter of seconds. No human on earth could process that kind of information so quickly. If we could use machine learning to examine data from a complex dataset that had 100 variables, we could probably learn about trends and patterns that were very complex and hard to distinguish manually. This is what makes computers such a useful tool, and why they have helped to make huge strides in data science. Using computers for data analysis makes it easier to find patterns and similarities that you don’t even know exist, or that may have not even considered.

In other tasks, computers perform very poorly. Some of these tasks would seem very simple to us. Like identifying the difference between a picture of a cat and a picture of a dog. But for a computer, this is extremely complicated to figure out. Therein lies the current challenge with artificial intelligence. Bridging that gap between the type of inductive reasoning humans are capable of, and the type of reasoning computers are good at.

Key term: Inductive reasoning. Drawing on information from experiences and our environment to draw generalizations.

The ability to see the difference between these pictures based on our knowledge of cats and dogs; this is what we know as reasoning. The goal of artificial intelligence is to teach computers how to have similar abilities to human-like reasoning

Computers models have been utilized to process Natural languages. Natural language processing gives computers the ability to understand ‘natural’ languages, or what we know as our human languages. Natural language processing relies on machine learning techniques to understand speech and text and respond to commands and interactions.

This technology is becoming very common and accessible. GPUs (graphics processing units) are becoming more widely available and less expensive, which means data sets are getting bigger, and the use of machine learning is expanding. You may have used it when you talk to Siri on your iPhone. When you say something to Siri, your phone receives the audio. To interpret it, it turns your audio into text. Then your phone analyzes the text to derive meaning from the command that you gave it.

Natural language processing is one of the most common applications of machine learning, and we use it every day. When we use a search function on the internet, we are using natural language processing. Translation apps must take our voice or our text and analyze the sentence structure to make meaning. When you type up a paper or a word document, your word processor uses natural language processing to sift for grammatical errors and spelling mistakes.

Despite its popularity, it’s a very complex field of computer science and artificial intelligence. Being able to interpret the meaning of the alphabet arranged in a virtually infinite amount of combinations requires huge amounts of data for the computer to understand what you are writing or saying.

In addition to being able to understand what we say and write; computers can also make strategic decisions based on what they’ve learned from data in the 1990s. IBM created a computer called Deep Blue that defeated a world master at chess. It was the first computer to be able to perform such a task. Because of the simplicity of the rules in chess, computer scientists at IBM chose to train their computer to play. But there are thousands of potential moves and arrangements that the pieces can take once the game has started. The computer had to learn this using data.

What makes machine learning unique to other types of computer science is the ability of models to change their methods over time to suit new data. What separates a machine learning model from a regular line of explicit code is that machine learning will take in new data and improve itself. It can also perform tasks which require planning and contain strategic components. The Deep Blue computer had to be adept at analyzing possible sequences of moves, rather than just one move at a time.

The same technologies that enabled a computer to beat a world chess champion are now making it possible for self-driving vehicles to get a passenger safely from point A to point B. Compared to the relative simplicity of chess, self-driving cars must plan and interpret hundreds of variables to keep its passenger safe. It goes beyond the two-dimensional data analysis employed by machines playing chess. Self-driving cars must master multidimensional data analyses to navigate the everchanging environment on the road.

The machine learns through trial and error, repeating the task over and over and learning from failures and successes. These experiences are introduced as data, and over time, the machine will know its probability of failure or success for every possible move.

Machine learning models interpret potential states in the environment. For an algorithm that plays chess, this is all its possible moves and all its competitor's possible moves. The algorithm is an amalgamation of goals and potential actions. By using this data, it creates a plan to optimize the likelihood of achieving the goals. It also enables computers to do self-learning without specific directions through programming.

Trying to get a computer to do all these things sounds simpler in theory than it is in practice. Most of the functions we’ve just mentioned; from checkers to self-driving cars require advanced statistical techniques to optimize the outcome and train a machine that knows how to ‘win’ with a high level of accuracy.

Machine learning falls under the larger umbrella of artificial intelligence. Artificial intelligence is a branch of computer science that includes reasoning, natural language processing, planning, and machine learning. The term was first coined by a computer scientist named John McCarthy in 1956. You’ll also hear the term data science, which encompasses artificial intelligence and machine learning. Data science is a broader term, but it’s often used to describe machine learning. Machine learning experts are often referred to as data scientists, both in this book and beyond. There is overlap between data science and machine learning, but they are not the same thing. Data science is more of a general term, whereas machine learning is a part of data science.

How is machine learning used?

Machine learning is a popular buzzword nowadays. You have often heard the term machine learning thrown around, especially in data science for digital marketing. Other familiar terms like artificial intelligence, data science, and data mining may seem synonymous with machine learning. There are slight differences between these different fields, which all fall under the umbrella of data science.

Data science is the management and analysis of data, and within data science, there are many ways to analyze the data and use it to learn. Machine learning is its own field of study within the realm of computer science. The idea is to predict something, and as you add more data, compare predictions to actual outputs. Over time, your ability to predict should improve, and errors will be reduced.

One of the most important functions of the human brain is the ability to change our behavior based on the outcomes of past events and situations. If a situation has a positive outcome, we remember that, and in turn, if a situation has a negative outcome, we also store that in our memory. Later, we'll use this 'data' to make decisions about new situations. Over time, we learn how to interpret situations, even if they are totally new, and we aren't totally sure how to behave or respond.

Machine learning helps us create a mathematical way of copying the human ability to learn over time and through a new experience. Machine learning models learn over time and improve their methods of predictions, therefore improving outcome. Past data is compiled, and over time, the model can make better and more accurate predictions. Over time, the program will be able to make more accurate predictions because of the new data it's been given. It learns over time how to do a better job at completing a given task.

Some of it has inspired science fiction, spurring fears that artificial intelligence will surpass us and take over the world. Eventually, our machines will be able to do everything that we can do, but better. Our computers will surpass us and leave us behind. Despite common fears, artificial intelligence is still a relatively young field with a long way to go. While it may not take over the world anytime soon, machine learning has changed the job market today, and it will only continue to change it in the future.

Jobs that used to require human thinking can now be done with machine learning. Factories, medical diagnosis, and even taxis have the potential to be done using artificial intelligence and machine learning. Data is becoming an ever more important field of study. Patterns in data can be impossible to interpret with a human brain, which is why we use machines to detect patterns.

In finance, machine learning is being used to detect fraud. Algorithms are now capable of detecting when a financial transaction has characteristics of fraud. Companies can spot fake reviews by recognizing word patterns and timing for review postings that are more likely to be fake.

Our phones use speech recognition to understand what we are saying and to respond to our requests. Social media uses complex data analysis to recognize patterns in our photos and can tell who is in a picture before we even begin tagging. All of this is done by the collection of data and interpretation through machine learning. The patterns in photographs that tell who appears is data which is analyzed and improved upon by machine learning models.

In 1959, Arthur Samuel first popularized the term "machine learning." A graduate of MIT, he chose to create a computer program that could play checkers. He chose checkers because of its relative simplicity, but also because checkers contains many possible strategies. This predated IBMs deep blue, but the theory was the same. Let the computer learn over time with new data.

The computer considered the positions of every checker piece, and each potential move. Each move had a score with the probability of winning. The algorithm included factors like the number of pieces on the board and the number of pieces that had been kinged.

The algorithm memorized every combination of positions that it experienced, in a process called rote learning. For each of these combinations, it remembered the score of the probability of winning. Over a few years, Samuel played countless games with the machine in order to teach it how to play.

Machine learning was coined to express what the machine was doing to learn how to beat opponents. The machine had to learn how to play checkers and learn how to reduce its errors to win. This was the first time that machine learning was talked about as its own independent field of study outside of computer science or artificial intelligence.

Sandford University refers to machine learning as "The science of getting computers to act without being explicitly programmed."   Samuel's computer didn't have to be programmed to remember every possible checkers move. Instead, the ‘experience' that the machine gained served as data, and it learned from each game to optimize its strategy for winning. Statistics offers the structure and mechanics for machine learning. Despite the birth of the term machine learning in 1959, it has only been recognized as a separate field of computer science first since about the 1990s with the advent of Deep Blue.

Normal computer programming uses an input command to direct the model. An example of this would be plugging 4+4 into your python window. It will give you the answer, 8. Instead, machine learning uses what's called input data. Input data is what the machine needs to learn, whereas input command would mean that the machine is not self-learning. The programmer doesn't specify what he wants the answer to be. Instead, the machine interprets the data on its own, which makes it self-learning. In machine learning, the machine takes data over time and uses it to create a new model for something.

The machine will detect patterns and structure within data, based totally in statistical logic. It is completely based on mathematical algorithms. Rather than using intuition to search for patterns, patterns our found at a quantitative and logical level by the machine learning model. The more relevant data a machine is exposed to, the better it understands and can predict the outcome of the model.

Machine learning is only useful if you can make it improve the efficiency of the task you are trying to achieve. Before you begin with your data, you need to think specifically about what you are trying to learn. What is the question you are trying to answer with your machine learning model?

The environment is always changing, and if you want to create models that are up to date and work in the changing environment, then you need to continue to teach your model with new data. If you want your data to be useful to you, then you need to find a way to keep your data up to date with the current questions.

Find out as much as you can about your market, or about the environment that you are operating in. What kind of things are you trying to understand? If you are using machine learning to improve your business, then find out what kinds of variables make up your customers choices. These are the things you want to identify and study in your models. You must have a good understanding of your subject before you can use machine learning to study it. It’s impossible to predict the future, even with machine learning at your disposal. But if you can adapt your models to the ever-changing data, then you will have a better shot in the long run than if you stay stagnant.

Before you start, you should have a good amount of knowledge of the data you are choosing. Where did it come from, and how was it collected? What format is it in, and what will the challenges be while interpreting it? These are the types of questions you should be asking when you begin, and I hope that with this book, you'll be able to look critically at the data before you begin.

It's not a one-size-fits-all approach to predicting the future, and it won’t give you the magic formula for anticipating future business trends or stock prices. But it is an incredibly useful tool that when used right, can make decision making much easier.

Opportunities to experiment with machine learning are growing. It's a challenge today to find the right people to fill the jobs required for the development of machine learning and artificial intelligence. It's a specialized field, and there is a short supply of people with statistical and computer science knowledge to move the field forward. This means there is an opportunity for people who possess the skills necessary for machine learning.

Recent Advancements in Data Analysis

Machine learning has changed drastically since its beginning in 1959 with Arthur Samuel’s checkers playing computer. But it has changed more in the last two decades than in its entire history, especially with the improvement of computing power. In the past, machine learning and big data analysis used to be very limited. Only larger companies with expensive tech had the ability to use data to make business decisions.

Now, almost anyone can utilize some amount of data for business or other purposes with a laptop or home computer. Data is much easier to come by, and the machines to process it have also become much more accessible. What used to take expensive computing power can now be done much more cheaply, and quicker.

The advent of cloud technology has made it easier for smaller businesses access to large datasets without the need for massive amounts of data storage. Now machine learning has become a completely different field of computer science, with people who specialize in machine learning and data science as its own field.

More and more things are connected nowadays, and the internet continues to grow larger and larger all the time. This means that access to data is increasing, but also the sources of data are changing. Even people’s cars have computers in them, which means that when they drive, they are creating data which can be interpreted. The vast majority of Americans carry a cell phone and do internet shopping and use apps for navigation. People use their phones to control home appliances, which is another potential source of data. There are Fitbits and smartwatches which allow people to track data about their health.

The more devices, not just computers, and phones, but devices of all kinds, are connected, the greater the possibilities in terms of collecting and studying data. This connection of everything; smartphones, smart cars, etc. make people nervous that they may be at risk of losing their private data. They fear that their privacy at stake and someone is always watching them. But machine learning and data analytics are making our lives much easier. Finding the right products is easier, navigation is easier, and finding new music is easier. This is all thanks to machine learning.

Image Recognition

One of the applications of machine learning models is for the sorting and classification of data. This type of classification can even be used for the classification of images. Search engines use this kind of algorithm to identify photos, and social media sites now use face detection to identify a person in a photo before the photo is even tagged. They do this by learning from data compiled from other photos. If your social media account can recognize your face in a new photo, this is because it has created models using data from all the other photos on your account.

Image recognition techniques require deep learning models. Deep learning models are made with an artificial neural network, which will be covered more extensively later in this book. Deep learning is the most complex type of machine learning in which data is filtered through several hidden layers of nodes. They are called hidden layers because the models are unsupervised, meaning that the features identified by the model are not chosen by the data scientist beforehand. Usually, the features are patterns that the model identifies on its own. Features identified in neural networks can be quite complicated, the more complicated that the task is the more layers that the model will have. Image sorting models might only have two or three layers, while self-driving cars will have between one and two hundred hidden layers.

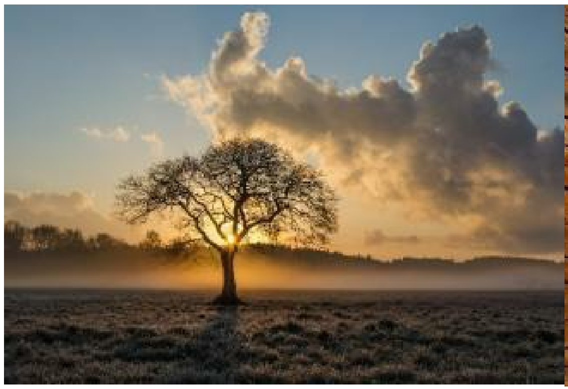
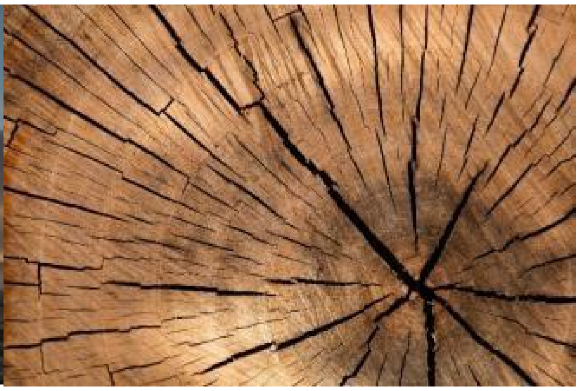
We have made big strides in this in recent years, because of the increased availability of computing power. Imagine the computing power that it requires to pass thousands of data points through hundreds of stacked nodes simultaneously. Deep learning and artificial neural networks have become more feasible in the last decade, with the improvement of computers and the reduction of cost to process large amounts of data. Especially with the advent of the cloud, which allows data scientists to have access to huge amounts of data without using physical storage space.

There is a website called ImageNet, which is a great resource for data scientists interested in photo classification and neural networks. ImageNet is a database of images that is publicly accessible for use in machine learning. The idea is that by making it publicly accessible, the improvement of machine learning techniques will be a cooperative effort with data scientists around the world.

ImageNet’s database has around 14 million photos in its database, with more than 21,000 possible class groups. This allows a world of possibilities for data scientists to be able to access and classify photos to learn and experiment with neural networks.

Each year, ImageNet hosts a competition for data scientists worldwide to create new models for image classification. Each year the competition gets harder. Now they are starting to transition to classifying videos instead of images, which means that the complexity and level of processing power required will continue to grow exponentially. Using the millions of photographs in the database, the ImageNet competition has fostered groundbreaking strides in image recognition made during the last few years.

Modern photo classification models require methods capable of very specific classification. Even if two images should be put in the same category, they may look very different. How do you make a model that can distinguish between them?

Take, for example, these two different photos of trees. If you were creating a neural network model that classified images of trees, then ideally you would want your model to categorize both as photos of trees. A human can recognize that these are both photos of trees, but the features of the photo would make them very difficult to classify with a machine learning model.

The fewer differences the variables have, the easier they are to classify. If all your photos of trees looked like the image on the left, with the tree in full view with all its features, then the model would be easier to make. Unfortunately, this would lead to overfitting, and when the model is introduced to data with photos like the one on the right, your model wouldn't be able to classify it properly. We want our model to be capable of classifying our data, even when they aren't as easy to classify.

Incredibly, ImageNet has been able to make models capable of classifying data with many variables, and very similar data. Recently, they created Image recognition that can even identify and categorizes photos with different breeds of dog. Imagine all the variables and the similarities that the model would need to recognize in order to tell the difference between dog breeds properly.

The challenge of identifying commonalities between a class is known as Intra-class variability. When we have a picture of a tree stump and a photo of a tree silhouetted in a field, we are dealing with intra-class variability. This problem is how variables within the same class can differ from each other, making it harder for our model to predict which category they fall in to properly. Most importantly, it requires a lot of data over time to improve the model and make it accurate.

In order to have an accurate model despite high levels of intra-class variability, we will need to use additional techniques with our neural network models to find patterns among images. One method involves the use of convolutional neural networks. Rather than having just one model or algorithm, data is fed through several models which are stacked on top of each other. The neural networks convert images features into numerical values to sort them.

Unfortunately, it would be beyond the scale of this book to try and understand the way these deep neural networks operate, but there are many books available that cover those types of models and also include more comprehensive explanations of the coding required to perform these types of analysis.

Speech Recognition

Improvements in artificial intelligence have made speech recognition more useful very recently. Most of our smartphones now have some level of speech recognition ability, which involves machine learning. Speech recognition takes the audio data we give it, and it turns it into text that can be interpreted.

The difficult thing about speech recognition is the irregularities in the way that people speak. Like intra-class variability. You and I may have different accents and different inflections that are hard to account for when you are teaching a computer how to understand the human voice. If we both say the same word with different accents, how do we teach the model to understand us?

Speech recognition also uses neural networks to interpret data, like image recognition. This is because the patterns in audio data would probably not be recognizable by a human. Data scientists use sampling in order to interpret data and make accurate predictions despite the variances in peoples voices. Sampling is done by measuring the height and length of the sound waves, which believe it or not can be used to decipher what the user is saying. The recorded audio is converted into the wave map of frequencies. Those frequencies are measured by numerical values and then fed through the neural networks hidden layers to look for patterns.

Medicine and Medical Diagnosis

Machine learning is not just useful for digital marketing or making computers respond to your requests. It also has the potential to improve the field of medicine, particularly in the diagnosis of patients using data from previous patients.

With as much potential as machine learning has for medical diagnosis, it can be challenging to find patient data that is available to use for machine learning because of the laws surrounding patient privacy. Its gradually gaining acceptance in the field of medicine, which means data is becoming available to data scientists. Unfortunately, up until now, it has been difficult to have enough meaningful data to be able to make models regarding medical diagnosis. But the technology is there and available to be used.

Machine learning could use image recognition to diagnose x-rays by taking data from several patients to imaging scans in order to make predictions about new patients. Clustering and classification can be used to categorize different types of the same disease so that patients and medical professionals can have better a better understanding of the variation of the same disease between two patients, and their likelihood of survival.

Medical diagnosis with machine learning can reduce diagnosis errors made by doctors or give the doctors something to offer them a second opinion. It can also be used to predict the probability of a positive diagnosis based on patient factors and disease features. Someday, medical professionals may be able to look at data from thousands of patients about a certain disease to make a new diagnosis.

But medical diagnosis is just one of the numerous ways that machine learning could be utilized in medicine. Medical datasets remain small today, and the science of machine learning still has a lot of unmet potential in the field of medicine.

Stock Predictions

Stock traders look at many variables to decide on what to do with a stock, whether they want to buy or sell or wait it out. They look at certain characteristics of a stock, and trends in the market environment to make an educated guess on what they should do. This is the way it has been done for years. Brokers and traders had to do research manually in order to make the best guess.

Machine learning can now be used to do the same thing, except that machine learning can do it much faster and more efficiently. In order to be an effective trader, you must be able to analyze trends in real-time so that you don’t miss out on opportunities. Machine learning can help traders with finding similarities between stock to make financial decisions using statistical data.

Traders can use linear regression models to study data about past trends in stock market prices, and what variables cause a stock price to go up and down. They can use these regressions to decide on what to do with a stock.

Often, traders who want to analyze the performance of stock do so by utilizing what’s called a support vector machine. A support vector machine is a classification model where data points are separated by a boundary line, with one category on a side and a different category or another. Traders will use support vector machines to classify which stocks to buy and which stocks to sell. Using certain variables that should be indicative of the performance of a given stock, that stock is placed on the side of the boundary line that denotes whether the price is likely to go up or go down.

Deep learning is also commonly applied in making stock models. The hidden layers of a neural network may be useful in identifying unseen trends or characteristics of a stock that could cause them to rise or fall in price.

There is no such thing as a sure bet or a risk-free investment. This was true when people made decisions, and it’s still true now when we use data science to make financial predictions. It's important to remember that investing in the stock market will always be risky. It’s impossible to create a model that will predict anything reliable about the stock market. It's wild and unpredictable. But we’ve already learned that machine learning can find patterns that humans may not be capable of finding on their own.

If you understand that trends in the stock market may be totally random and unpredictable, then it’s useful to have another model that will help you estimate a stocks predictability. Knowing how accurate your predictions are for a given stock, is just as important as the predictions itself. Create a separate model to measure the predictability of a given stock, so you know how reliable your predictions are. Different stocks will have varying levels of predictability. It's important to illustrate that with your model so that you can choose from the most reliable predictions.

Traders will continue to make the final call on whether or not a stock will go up or down in value. But data science and machine learning can streamline the process of analyzing information that will help the decision process. Which is why you will see more and more examples of machine learning models used in predicting stock, and why you should at least make yourself familiar with the idea.

Learning Associations

Marketers in all fields, from brick and mortar retail to online retail, are always seeking ways to link products together, and increase sales. Whether you own a small bike shop or a massive online warehouse, finding patterns in your customer's buying habits will help you make proactive decisions to drive sales and make more money.

Most of us will visit a grocery store during any given week. Grocery stores are a perfect example of using product positioning to get sales. Any given grocery store will organize itself so that similar items are placed together. Baking goods have their own aisle, while fruits and vegetables have their place. They do this for two reasons; it makes it easier for the shopper to find what they need and improves the customer experience. Also, product positioning can help connect customers with products that they are willing to buy but weren't seeking when they first walked in the store.

Beyond just putting the vegetables in the same aisle, there is another strategy that grocery stores can implement to steer customers towards certain products. They might infer characteristics of a customer buying a specific product, and use that to recommend other, unrelated products. For example, you can assume that someone who buys fresh vegetables from the vegetable aisle eats healthier. You could put smoothies in the vegetable aisle, in the same refrigerator where you keep fruit. If a customer comes in looking for craft beer, you can tempt them with a snack and place the kettle chips in the same isle as 12 packs of light beer.

If all of that makes sense to you, then you are on your way to understanding a technique called collaborative filtering. It’s a machine learning technique that’s widely used in internet marketing. If your search data shows that you’ve been browsing airline tickets to Cancun, you might start to notice advertisements for swimsuits showing up on your browser.

Marketing data scientists are always trying to answer this question; how can we use data to find a way to link a product with its target market? It's about utilizing data to link two otherwise unrelated products together to drive sales.

It’s a way of making recommendations to a customer based on what you know about them. Machine learning can often find similarities or buying patterns in customers that we may not have known to look for. This is a powerful marketing tool that’s starting to emerge in the modern age. Before, most marketing agencies had to use intuition to find their target markets. Now, data scientists can use quantitative data to draw a more accurate picture of their ideal customer. If you’re interested in using machine learning in digital marketing, then this is a topic you should be familiar with.

Collaborative filtering is different from just advertising a similar product to a customer. You are making predictions about a customer's taste or preferences based on data you've gathered from other customers. You base this prediction on a correlation that you have found between two products, and then a measure for the likelihood that the product Y will be bought with product X. You use these estimates of correlation to decide on what to market and to whom.

Spotify uses a similar process when its making song recommendations. It uses data from all the music you have liked over time. If there is a correlation between two artists, meaning that a lot of people have both artists in their library, the model can predict the probability that you will like the other artist.

The more products you have in your store, the more intensive it will be to find these correlations. In a perfect world, you will be looking for correlations between every different combination of product that you have in your store.

This method for finding the probability that you will like one product based on buying another product is called the Apiori Algorithm. There are three criteria that need to be met to affirm that there is a correlation between the two products and that you should link them somehow in your store. The first criterion is support. This gives you a measurement of the popularity of a specific product. Out of all your transactions, how often does this item appear in peoples shopping cart?

The second part is the confidence in the correlation between the two products. How likely is it that customers will buy Y product when they purchase X product? Finally, what is the lift of product Y? In other words, how likely is it that someone will buy Y with X, based on the popularity of Y alone.

The model can also use data from things like purchases, social media engagements, etc. to make a prediction on the type of product you will like. This distinguishes it as machine learning rather than just data analysis because the model was looking for similarities, but the programmer didn't ask for a specific output. Maybe there are certain features or characteristics of the group that the programmer isn't even aware of. Maybe with unsupervised machine learning, the data tells us that there is a high correlation between the two types of customers. These correlations are happening all around us with similarities between groups of people. Sometimes, it requires a good computer model to spot the patterns in the data. Machine learning can find similarities that would be impossible to see without the help of computers and good modeling.

Data scientists in marketing sectors are already using statistics to improve their stores online, and if you want to keep up with online retail then its advisable to start reading about how data can help you identify similarities and trends between products, using machine learning as your tool.

Finance

The financial industry is seeing an increase in the use of machine learning. The use of data science and machine learning models makes the decision-making process faster and more efficient for financial institutions. The possibilities and applications of machine learning can be misunderstood, which means it is often underutilized or misused in finance sectors.

Work that was once tedious and required hundreds of hours of human work can now be done by a computer in a matter of a few minutes. A common example is the use of machine learning for call center and customer service work. Many of the tasks that once required a human operator can now be done over the phone with a robot that is designed with machine learning.

In addition to customer service, banks can now process and analyzing contracts and financial information from thousands of customers that would otherwise be labor-intensive — used to create credit reports and predict the likelihood that a customer will default on a loan. Machine learning techniques can look at a history of transactions of a borrower before the bank decides on whether they should loan money to that individual.

Machine learning is also being utilized in fraud prevention. It has made the finance industry more secure. Machine learning has improved the bank’s ability to detect patterns in transactions that are indicative of fraud. Rather than having people assigned to monitor the transaction and look for signs of fraud, machine learning models can learn from fraud data to find patterns by automatically sifting through millions of customer transactions.

Spam Detection

A common example of a relatively simple machine learning tool is spam detection. If we are using supervised learning, defining the variables that are relevant, then the model will have certain characteristics to look for in email messages received. The model might look for certain keywords or phrases in order to detect if an email is spam or not. Words like ‘buy’ or ‘save’ might let your inbox know when you are receiving spam email. The problem with this method is there are many cases in which those words might not necessarily mean spam. There might be other keywords or combinations of words that we would overlook.

This is where reinforcement learning comes in. There are so many characteristics that may be indicative of spam email, some of them we may not even be able to explain. Reinforcement learning will allow to model to find these patterns on its own, without explicit guidance. Instead, we tell the model when it has identified spam correctly. Sometimes we find an email message in our inbox that the model didn’t classify as spam, so we move it to our spam folder manually. Now the model knows that this message is spam, and this piece of data is added to the model to improve the prediction next time. So over time, the machine improves as it gets more relevant data.

This type of machine learning is known as classification. Our output falls into discrete categories. In statistics, discrete variables are variables which can be identified in only a finite number of categories. An example of a discrete variable would be the number of cars sold by a car dealership in a week. It’s discrete because the car dealership can’t sell half a car. The variable must be a whole number.

Introduction to Statistics

Statistics is the mathematical science of data. It is a practice by which data is collected, observed, and analyzed in order to infer meaning and examine quantifiable relationships between different variables. Machine learning is a form of inferential statistics, meaning that by examining the relationship between variables, we should be able to come up with predictions for new variables.

Statistics is used in a wide variety of disciplines. It's used in biology to study and examines animal and plant life. It has wide applications in the business world from making stock market forecasts to analyzing consumer behavior. Economists use statistics to explain quantifiable patterns in world markets. In medicine, statistics can be used to improve the way that doctors and disease specialists look at the spread and prevention of disease.

Statistics make up the core of machine learning. If you aren’t willing to dive into statistics, then machine learning isn’t for you. Machine learning uses statistical algorithms to help computers learn. Machine learning is all about the tracking of data and how computers can use data to improve themselves.

There are two types of statistics which are relevant to this book. The first one is descriptive analysis, which you might use during the beginning of your modeling process to look for indicators in your data. But most of what we do in machine learning falls into a different category called predictive analysis.

Key term; Descriptive analysis. The descriptive analysis helps us examine we are right now. Looking at our current situation in context to the past and seeing why things are the way they are. Why do some things sell better than others? What trends are we seeing in products currently on the market?

Key term; Predictive analysis. Predictive analysis helps us to see and understand what will happen in the future based on indicators that are currently present. When we are using machine learning for predictive analysis, it's important for us to stay current and continue to feed the model new data. What trends should we be on the lookout for?

Machine learning is just another way to understand the data that is around us and to help us understand our present and predict the future. But it requires data from the past and present so that we can find trends and see where they might lead.

Within statistics, there are two over-arching categories of data that we will use, and all our data will fall into one category or the other somehow.

The first category is **quantitative data**. Quantitative data is data that can be measured with a numerical value. Some examples of quantitative data include height, income, or the square footage of a house. All these variables can be measured by some number, which makes them quantitative.

The second category is **qualitative data**. Qualitative data is data where the variables are assigned to categories or classes. Examples of qualitative data would include someone's gender, blood type, or whether a piece of real estate has a pool or not. This data can be sorted by its identity and is non-numerical. Therefore it is qualitative.

Quantitative data can either be discrete or continuous. If we have a data set where there is a variable recording the number of patients that a hospital had last year, this would be considered discrete. Discrete variables have a finite amount of values that they can have, and they are always whole numbers. It would be impossible to have half a patient or a percentage of a patient. Therefore this variable is discrete.

Data can also be continuous. An example of continuous data would be a variable for income. Income can take on half values, and there is a virtually infinite amount of possibilities for the value of income in data.

Some other important terms to remember are the mean, median, and mode. You will often hear these three things referred to in this book when we are talking about regressions. These are all different measures of central tendency. The mean is our average value for data. If we have a variable for a person’s age, we will find the mean of age by adding all the ages together and then dividing by the number of respondents in a data set.

The **median** is the value in the middle of the dataset. If you took all the responses for age and found the response that was in the exact middle of a sorted list of responses, then this would be your median.

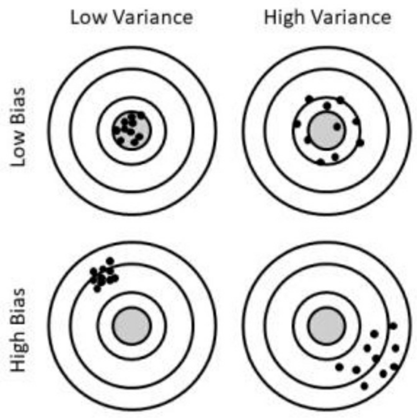
The **mode** is the response that occurs the most frequently. If we took a sample of eleven people’s ages and found that the ages were 19, 19, 20, 21, 22, 22, 22, 23, 24, 24, 25 then the mode would be 22, because it occurs the most frequently in this sample. The median would also be 22 because it happens to be in the middle of this sorted list of responses.

When you are making a statistical model, there are many important terms that have to do with the accuracy of our models. The most important, and the most frequently mentioned in this book are bias and variance. These are different kinds of prediction errors that can occur when we are creating statistic models. Ideally, we’d like to minimize the prevalence of bias and variance in our models. They will always be present, and as a data scientist, you will have to find the right balance of bias and variance in your models, whether that’s by choosing different data or using different types of models. There are many ways to reduce variance and bias within a model, dependent on what you are trying to do with the data. By trying to reduce these with the wrong approach, you run the risk of overfitting or underfitting your model. When your model is **bias**, it means that the average difference between your predictions and the actual values is very high.

Variance is how to spread out our predicted data points are. Usually, the variance is a result of overfitting to the sample data we used to create the model. It doesn't do very well at predicting the outcome of new variables.

There will always be some level of error in your models. It’s a fact of life that no matter how good you are at predicting something, there is always some random or nonrandom variation in the universe that will make your prediction slightly off from the true outcome.

I’ve created a visual example of four bullseye targets to help illustrate the difference between models suffering from high bias and variance. In this instance, the center of the bullseye represents the true value that our model is trying to predict. The top left corner is the ideal model. Notice that all our predicted data points are falling right on the bullseye. This model is quite accurate and places our predicted data points all around the true value. This is because of low variance; a lack of ‘spread out’ data points, and low bias; underfitting that skews our results.



In the top right target, the model is suffering from high variance. You can see that our data points are clustered around the bullseye. Unfortunately, the average distance between the predicted values and the bullseye is high due to high variance.

In the bottom left target, the model didn’t suffer much from high variance. The average distance between the predicted data points is low, but they aren’t clustered around the bullseye but slightly off it as a result of high bias. This is probably the result of too little training data, which means that the model doesn’t perform well when it gets introduced to new data.

The bottom right model suffers from both high variance and high bias. In this worst-case scenario, the model is very inaccurate because the average distance between predicted data points and the true value is high, and the predicted data points are skewed.

Variance can be caused by a significant degree of correlation between variables. If you use too many independent variables, this can also be a cause of the high variance. Sometimes, if the variance is too high, we can combat that by allowing a small amount of bias in the model. This is known as regularization. We’ll cover that a little later.

In statistics, the **population** is the group of people or the set of data you are trying to analyze. The **sample** is the subgroup of that population, whose data you use to create your model. The **parameters** are the characteristic of the variables of the population that you are trying to identify and make predictions from in your model.

Descriptive statistics is the use of data to examine a population. Typically, descriptive statistics involve the mean or average, mode, media, size, correlation. Machine learning falls into the category of inferential statistics because we are using the data to find patterns and relations but also to make predictions based on this information. Inferential statistics, or descriptive stats, is using the characteristics of your population to make predictions. This is where your regression models and classification models will come in. When we infer something, we make a logical deduction about a population-based and the knowledge we are given.

When you are looking at data, you should also take note of the **distribution**. This is how the data is spread out on our graph. It shows the frequency of values of our data and how they appear in conjunction with one another.

We use our variance to find the standard **deviation**. Standard deviation is the average of the distances between the predicted data points and the real data points on a regression or prediction model.

We must also be sure to be aware of models that suffer from overfitting and underfitting. An overfitted model is good at predicting outcomes using the training data, but when you introduce new data, then it struggles. It's like a model that memorizes instead of learns. It can happen if you don't use random data in your training sample.

Underfitting describes a model that is too simple, and it doesn't examine any significant data patterns. It may do a good job of predicting, but the variables and parameters aren't specific enough to give us any meaningful insights if you don't have enough training data, your model could be under fitted.

One of the most commonly made mistakes when people are looking at data, is confusing correlation with causation. If I told you that every person who committed a murder last year bough eggs every week, I couldn't claim that people who buy eggs are murderers. Maybe looking at my data, I see a rise in people buying milk, as well as a rise in teen pregnancy. Would I be able to claim that there is an association between people drinking a lot of milk and teen pregnancy? Or teenagers getting pregnant caused people to buy more milk.

This is the difference between correlation and causation. Sometimes the data shows trends that seem like they are related. When two events are correlated, it means that they seem to have a relationship because they move along the graph at a similar trajectory, and during a similar space in time. Whereas causation means that the relationship between the two events involves one event causing another.

In order to imply that two things have a causal relationship, a few criteria need to be met. The first is covariation. The causal variable, and the event, it is supposed to have caused the need to be covarying, meaning that a change in one lead to a change in the other.

The second criterion that needs to be met is that the causal event needs to occur before the event it is supposed to have caused. For an event to be considered causal, it must come first.

Third, the data scientist must control for outside factors. In order to make a strong case that one thing causes another; you need to be able to present evidence that the other variables of the event are not the true cause. If the causal variable still creates the effect, even when other variables are considered, then you can claim there is a causal relationship.

Choosing the right kind of model for machine learning

Imagine that you are a carpenter. What kind of tools do you think you will have loaded in your truck when you arrive at a worksite? You will probably have a hammer, and a drill as well as a few different types of saws. You probably have a few planes and a good set of drill bits. If you know how to do your job, then you know when you will know the purpose of each of these tools, and you will know when each one should be used. Every single one of these tools serves a specific purpose. The drill can't do a hammer's job, nor would you try to cut something with a hammer.

A data scientist who is looking to do machine learning will have his own set of tools, each serving a different purpose and designed for a different function. Depending on what kind of data you're using, and what you want to find out, you will have to choose different algorithmic models to do the job.

Statistical algorithms can serve several purposes. Some predict a value; like a regression model that predicts your income base on your years of education and work experience. Some models predict the likelihood of an event occurring, like a medical model that predicts the likelihood of a patient surviving a year, or two years, etc. Other models sort things by placing them into different categories or classes, like a photo recognition software sorting photo of different types of dogs.

Depending on the outcome you are looking for, you will need to have your statistical toolbelt. You need to familiarize yourself with the technical skills of statistics. Also, you will have to know which tool you want to use, and when to use it. Here I have created a comprehensive list of the different kinds of statistical models that are very common in machine learning. To be able to write the code to build these models on your own, I recommend that you take some time to study in the programming language that you've chosen. But this list will give you an introductory understanding of each type of model, and when they are useful.

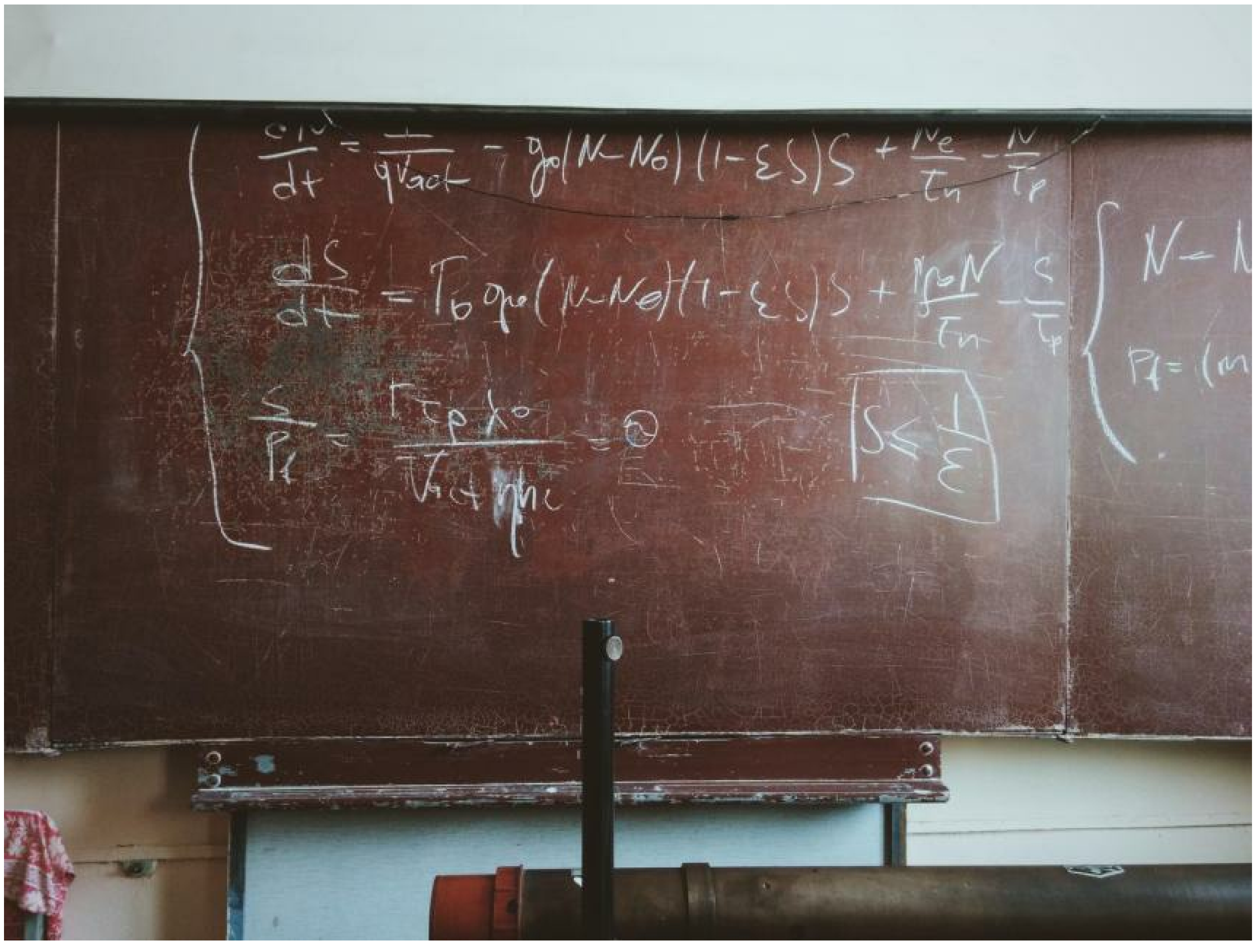
For machine learning to be effective, you must choose the right model and the model that works best and have relevant data for the model and the question at hand.

Nowadays, especially with the use of the internet and digital marketing, there are certain questions that can’t be properly understood without the use of data and machine learning that can analyze it. With machine learning and data science, you can keep track of your customers and their buying habits, so that you can better adapt to their needs when they change.

The better you are at interpreting your data, the more easily you will be able to identify trends and patterns so that you can anticipate the next change.

Machine learning can be broken down into three different categories, each one containing several unique algorithms that serve different purposes. To start, we’ll talk about the differences between supervised, unsupervised, and reinforcement learning.

Supervised learning



In supervised learning, programmers use labeled data. Before we begin using the algorithms, the data that we are looking at is already predetermined. We know the inputs and outputs we are looking for. X, and Y. We are trying to find a relationship between X and Y that we have chosen.

After you find a relationship between X and Y, you get a model, which will predict an outcome based on those relationships that your machine has observed in the data. Supervised learning is used for regression and classification models. In machine learning, we refer to **features** as a certain measurable property or characteristic of the data.

The first type of supervised learning that we'll talk about and the first type of statistical model is called a regression. **Regression** is a model where the data input and output are continuous. There are different types of regression, but the most basic form is linear regression. We use linear regression to find a relationship between some input X and an output Y. Once we have estimated this relationship, we can predict Y with X. Linear models can, and usually do, have more than one X. In regression; output Y has a numerical value.

Regression Analysis

Regression is the simplest type of machine learning; this is usually where you start when you are first learning how to use your data. You have a set of X values, and you want to study their relationship with Y, the output. Our independent variables, the Xs in our model, are given weight, and for each value of X, it is multiplied by weight until the aggregate function creates a prediction for Y.

We can create a predictive model for Y by using data where we already know the X and Y. Regressing this information will give us the weights of X. If we have enough relevant data, eventually we will be able to predict and unknown Y with known values for X.

We graph our known Y and X values on a scatterplot, and our regression model finds the “best fit” line through the data points. The regression line is called a hyperplane. The steepness of the line is called the slope.

We can measure the distance between the predicted value and actual data point, and we call that measurement deviation. Our goal when we create a linear regression is the minimize the deviation in our predictions. The smaller that difference, the deviation is, the more accurate your model is.

Most of the statistical models used in machine learning are rooted in this first algorithm. Creating a model that will predict an outcome by plotting our data points along a line or in clusters. But the line isn’t always straight, and sometimes the line doesn’t show us the best fit.

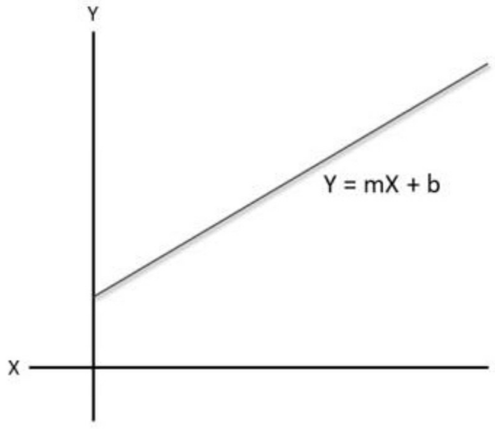
An example of a regression function that is nonlinear is the Sigmoid function. The Sigmoid function creates an S-shaped curve. Instead of predicting a value, the Sigmoid function takes independent variables and produces a probability between one and zero.

Simple Linear Regression. In simple linear regression, we study the relationship between some predicted value Y and some predictor X. We call our Y the dependent variable because it is dependent on the value of X. Our X is known as the independent variable.

If you took algebra or pre-calculus in high school or college then you might remember the equation of a line was;

$$Y = mX+b$$

If you were to graph this equation, then you would have a graph that looks something like this:



As you can see, the line shows that for every value of X there is a different value of Y. You can make a prediction for the value of Y for every new value of X. In this graph, the value of Y increases as the value of X increases. This is not always the case.

This is the most simplistic regression, but it’s important to understand how it works because we will be building off it from here on out. Most of the statistical analysis involves a plot like the one pictured above, which makes some prediction for an output dependent variable based on an input, the independent variable. This is an example of supervised learning because we are specifying the Y variable and the X variable that we are using before we start modeling.

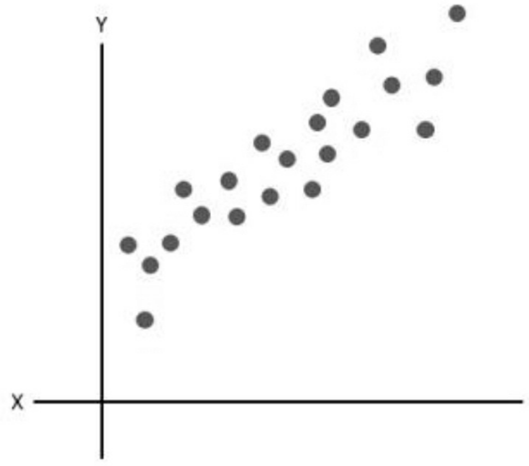
With almost all predictions, there will be more than one independent variable that will determine our dependent variable. This leads us to our next type of regression.

Multiple Linear Regression. In data science and most tasks in statistics, this will be the most popular type of regression. In multiple linear regression, we will have one output variable Y, just like before. The difference now though, is that we will have multiple Xs or independent variables that will predict our Y.

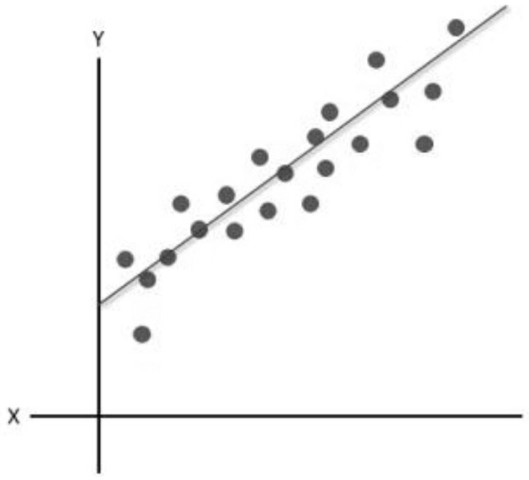
An example of using multiple linear regression to predict the price of apartments in New York City real estate. Our Y or dependent variable is the price of a New York City apartment. The price will be determined by X, our independent variables such as the square footage, distance to transportation, number of rooms. If we were to write this out as an expression it would look something like:

apt_price = β0 + β1 sq_foot + β2 dist_transport + β3 num_rooms

We take sample data, data that we already have where we know our Xs and their Ys and we look at them on a graph like this:



Scatter plot



$\text{apt\_price} = \beta_0 + \beta_1\,\text{sq\_foot} + \beta_2\,\text{dist\_transport} + \beta_3\,\text{num\_rooms}$

You can see that the values of X and Y don’t create a perfect line, but there is a trend. We can use that trend to make predictions about future values of Y. So, we create a multilinear regression, and we end up with a line going through the middle of our data points. This is called the best fit line, and its how we will predict our Y when we get new X values in the future.

The difference here is that instead of writing m for slope, we have written β. This equation is pretty much the same thing as if I had written

$$Y = b + m1X1 + m2x2 + m3x3$$

Except now we have labels, and we know what our Xs and our Ys are. In the future, if you see a multilinear equation, then it will most likely be written in this form. Our β is what we call a parameter. It's like a magic number that tells us the effect that the value of our X has on the Y. Each independent variable will have a unique parameter. We find the parameters by creating a regression model. Over time, with machine learning, our model will be exposed to more and more data so that our parameter will improve, and our model will become more accurate.

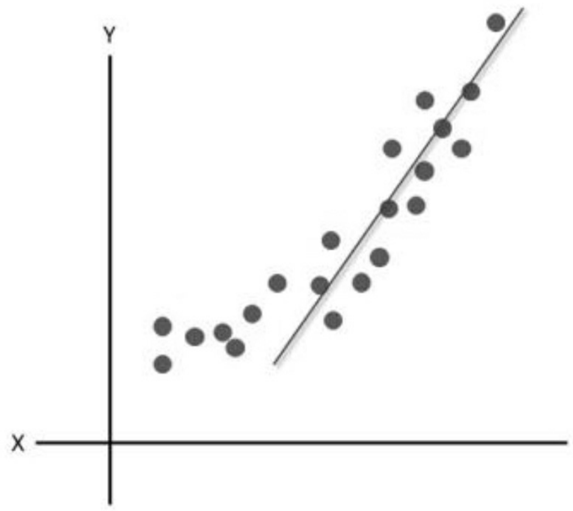
We can create the model by using training data that has the actual price of New York City apartments, and the actual input variables of square footage, distance to transportation, and many rooms. Our model will ‘learn' to approximate the price from real data. Afterward, when we plug in the independent variable for an apartment which has an unknown price, our model can make a prediction as to what the price will be.

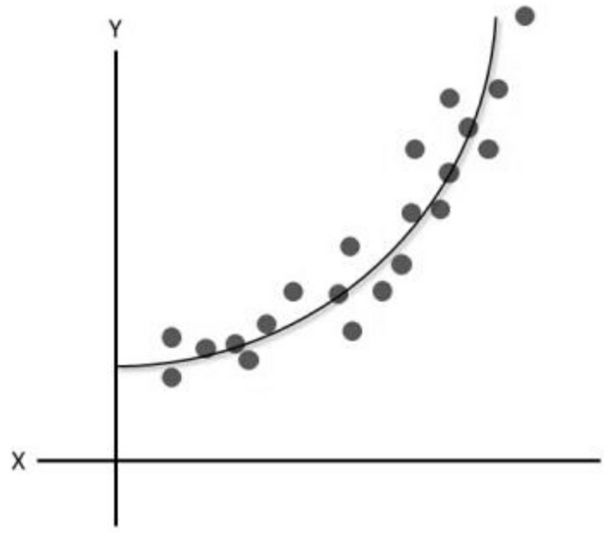
This is supervised learning using a linear regression model. It’s supervised because we are telling the model what answer we want it to give us; the price of New York City apartments. It learns how to predict the price more accurately, as it is given more data, and we continue to evaluate its accuracy.

Ordinary Least Squares OLS will try to find a regression line that minimizes the sum of errors squared

Polynomial Regression. Our next type of regression is called a polynomial regression. In the last two types of regression, our models created a straight line. This is because the relationship between our X and Y are linear, meaning that the effect X has on Y doesn’t change as the value of X changes. In polynomial regressions, our model will result in a line that has a curve.

If we tried to use linear regression to fit a graph that has nonlinear characteristics, we would do a poor job of creating the best fit line. Take the graph on the left, for example; the scatter plot has an upward trend like before, but with a curve. In this case, a straight line doesn’t work. Instead, with a polynomial regression, we will create a line with a curve to match the curve in our data, like the graph on the right



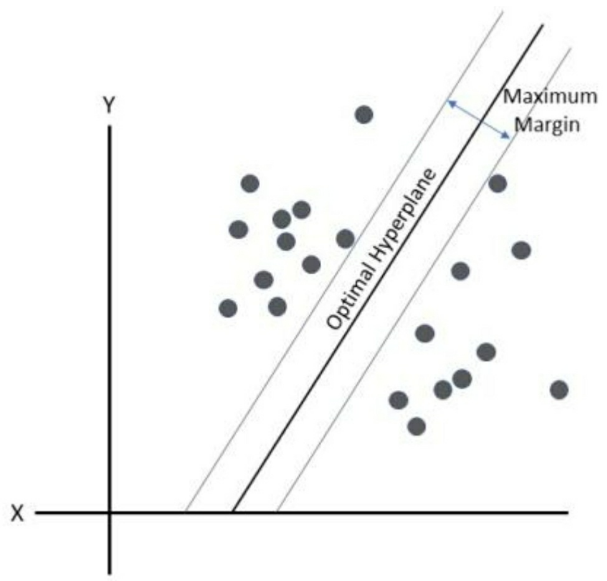


The equation of a polynomial will look like the linear equation, with the difference being that there will be some polynomial expression attached to one or more of our X values. For example:

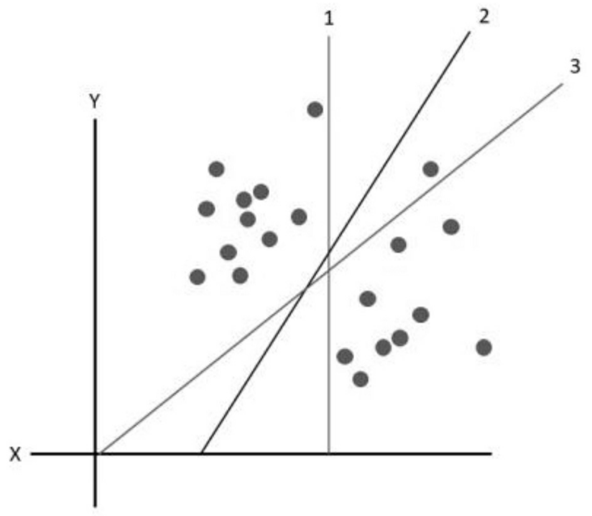
Y = mX2+b

The effect that X has on Y changes exponentially as the value for X changes.

Support Vector Regression. This is another important tool for data scientists and one that you should familiarize yourself with. It is most commonly used in case classification. The idea here is to find a line through a space that separates data points into different classes. Support Vector Regression is another type of supervised learning. Its also used for regression analysis. It is a type of binary classification technique not related to probability.



In support of Vector Regression, all your training data falls into one category or the other. You want to find out which category a new data point falls into. Your data is separated into these two classes by a hyperplane. When you're creating a model for the hyperplane, you are trying to find a hyperplane that maximizes the distance between the two classes. For example, in the next picture, you have a scatterplot where the data points can be separated into two distinct classes. In this instance, lines one and three can separate the data points into two distinct classes. However, for your model, you should choose line two because it maximizes the margin between the two classes, so they are more distinct. The wider the margin is, the better.



Ridge Regression. This is a technique commonly used to analyze data that suffer from multicollinearity. Using ridge regression properly can reduce standard errors and make your model more accurate, depending on the characteristics of your data.

Ridge regression can be useful when your data contains independent variables with a high correlation. If you could make a prediction about an independent variable by using another independent variable, then your model is at risk of multicollinearity. For example, if you are using variables that measure a person’s height and weight; these variables are likely to create multicollinearity in the model.

Multicollinearity can affect the accuracy of your predictions. To avoid multicollinearity, be cognizant of the type of predictive variables you are using. Type of data you are using, as well as the method of collection, could be the cause of multicollinearity. There is a chance you may not have selected a broad enough range of independent variables. Your data points could be too similar because your choice of independent variables is limited.

Multicollinearity can also be caused by having a model that is too specific. You have more variables than you have data points. If you have decided to use a linear model, and that has made multicollinearity worse, then you can try to apply a ridge regression technique.

Ridge regression works by allowing a little bit of bias into the model in order to make your predictions more accurate. This technique is also known as regularization.

Another method is to improve the model's accuracy by standardizing independent variables. The simplest way is to change the coefficients of some independent variables to zero to reduce complexity. However, we won't just change them to zero, but standardization implements a system that rewards coefficients that are nearer to zero. This will cause coefficients to shrink, so the complexity of the model is less, but the independent variables remain in the model. This will give the model more bias, but it's a trade-off for more accurate predictions.

LASSO Regression. LASSO regression is another ‘shrinkage’ technique. A very similar approach to ridge regression in that it encourages leaner, simpler models for prediction. In lasso regression, the model is a little more stringent about reducing the value of coefficients. LASSO stands for the least absolute shrinkage and selection operator.

Data on our scatterplot is shrunk down to a more compact point, like the average of the data. Just like ridge regression, we use this when the model is suffering from multicollinearity.

ElasticNet Regression. ElasticNet regression works by combining the techniques of LASSO and ridge regression. Its main goal is to attempt to improve upon LASSO regression. It is a combination of both the methods of rewarding lower coefficients in LASSO and Ridge regression. All three of these models can be accessed in the glmnet package in R and Python.

Bayesian Regression. Bayesian regression models are helpful when we have insufficient data or data with poor distribution. These types of regressions are created using probability distributions rather than data points, which means that the graph will appear as a bell curve representing the variance with the most frequently occurring values in the middle of the curve.

In Bayesian regression, the dependent variable Y is not a value but a probability. Rather than trying to predict a value, we are trying to predict the probability of an outcome. This is known as frequentist statistics, and Bayes theorem makes up the foundation for this type of statistics. Frequentist statistics hypothesize whether something will happen, and the probability that it will happen.

When we talk about frequentist statistics, we also include conditional probability. Conditional probability involved events whose outcomes are dependent on one another. Every time you toss a coin, it is an independent event meaning that the previous coin toss does not change the likelihood of the next coin toss. Flipping a coin toss, therefore, is not a conditional probability.

Events can also be dependent, meaning that the previous event can change the probability of the next event. Say I had a bag of marbles, and I wanted to know the probability drawing different colors out of the bag. If I have a bag with 3 green marbles and 3 red marbles, and I draw a red marble, then the probability of drawing a red marble goes down on my next draw. This would be an example of conditional probability.

Decision trees

One of the models that we'll discuss later is called neural networks. They're the most advanced types of machine learning and are used for many different purposes. I've related this to decision trees because of how frequently people turn to neural networks for classification problems when there are much simpler models available. Decision trees and the related random forest models can be just as useful.

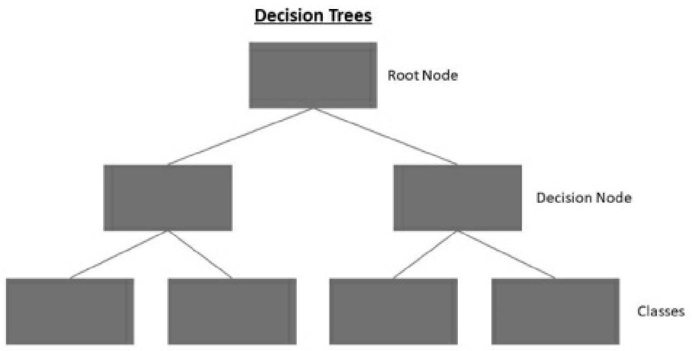
Despite the power of neural networks, they can’t be used for everything. Fortunately, we have options, and the purpose of this book is to know what your options are when you decide you want to build a model. The next place to look when neural networks don’t work is decision trees. Decision trees break data into subcategories using decision and leaf nodes in the shape of a tree.

Decision trees have a few advantages over neural networks (discussed later in this chapter). For one thing, neural networks require huge amounts of data and powerful computers in order to process them. The upside to using a decision tree is that they are relatively simple, especially when you compare them to neural networks. Unlike most of the models in this book, they are very easy to read a decision tree, even to the untrained eye. This makes them a good candidate when choosing a model that needs to be presented in front of stakeholders.

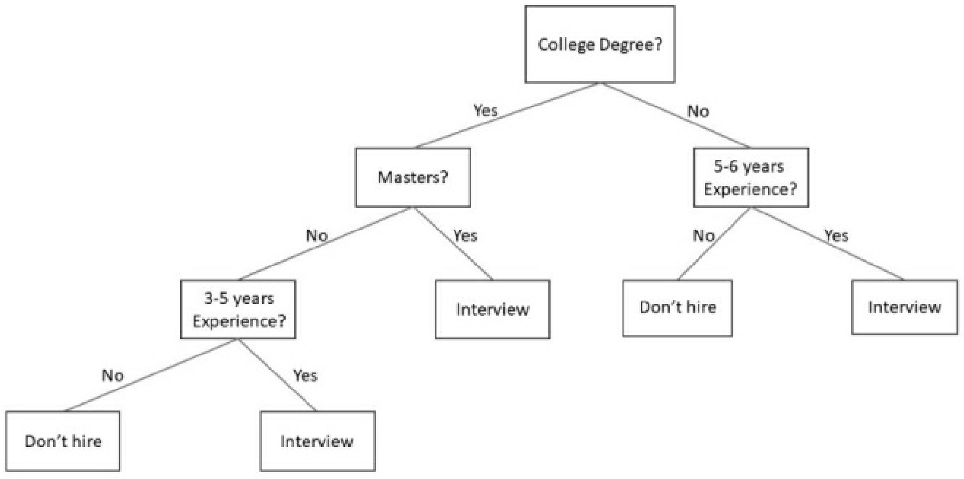
Decision trees are another form of supervised learning, meaning that we label the categories that we want to sort before creating the model. In some instances, decision trees can complete regression tasks, but typically they are used as classification models. When decision trees are used for regression, the leaf nodes end in probabilities.

Decision trees start with what’s called a root node at the top of the tree. Then the root node is split into two nodes after the root node. Nodes are individual leaves on the tree, and the middle nodes are where the decisions are made, known as the decision nodes. The decision tree ends at the bottom in what’s called a terminal node is at the bottom of a branch, where the decision is complete.

Ideally, the decision tree will sort the data quickly, layer by layer. Because of this, we call the model ‘greedy' because the top nodes try to sort the data as quickly as possible so that it requires fewer layers. Like neural networks, decision trees often suffer from overfitting. A decision tree usually won't work with other sets of data because the sorting is so specific to each dataset.



Below is an example of a decision tree about whether or not to grant a job applicant a job interview determined by qualification factors. The root node is whether or not the applicant has a college degree, followed by the decision modes which all lead to either a decision not to hire or grant an interview. You can see from this decision tree why this type of model is so specific to the specific data you are working with because every data set will have differences in qualifications and so every dataset will be sorted differently.



Random forests

Using just one decision tree on your model can have a limiting factor on the categories that the data is split in to, and the outcome of the decisions. Because the decision trees are ‘greedy’, this means that certain categories are chosen for sorting, which prohibits other categories from also being chosen.

But there is an easy way to get around that. One way to diversify your decision trees and improve the accuracy of your model is by using random forests.

If a real forest is made up of several different trees, then that’s exactly what a random forest is. Instead of just having one decision tree, you split the data into several decision trees. When you only have one tree, models can often suffer from high variance. Creating a random forest is a way to combat that in your model. It’s one of the best tools available for data mining. A random forest is as close as you can get to a pre-packaged algorithm for data mining purposes.

In a random forest, all the trees work together. The aggregate result of all the trees is usually right, even if a few trees end up with bad predictions. To create the final prediction, the results of all the trees are tallied. Using votes from the average values of all the trees gives us a final prediction.

Because we are using data that is similar, there is a risk of correlation between the trees if they are all trying to do the same thing. If we use trees that are less correlated, then the model will perform better.

Imagine that we were to bet on a coin flip. We each have a hundred dollars, and there are three choices. I can flip the coin once, and the winner of that toss gets to keep 100$. Or, I could flip the coin ten times, and we bet ten dollars each time. The third option is to flip the coin 100 times and bet a dollar on each toss. The true expected outcome of each version of this game is the same. But if you did 100 coin tosses, you are less likely to lose all your money than if you only did a single coin toss. Data scientists call this method bootstrapping. It’s the machine learner’s equivalent of diversifying a stock portfolio. We want to have a model that gives us an accurate prediction. The more we split our decision trees, the more accurate our data will be. But it’s important that the individual trees have a low correlation to one another. The trees in the forest need to be diverse.

How do we avoid correlation in a random forest? First, each tree takes a random sample from the dataset, so that each tree has a slightly different set of data from one another. The tree picks a feature that creates the most separation between nodes, in a greedy process, just like as individual trees. However, in a random forest, trees can only pick certain features from the overall group of features, so each tree separates by different features.

So, the trees will be uncorrelated because they are using different features to make decisions about classification. In a random forest, its best to use at least 100 trees for getting an accurate picture of the data, depending on the data set you are working with. In general, the more trees you have, the less your model will overfit. Random forest machine learning is called it a ‘weakly-supervised technique’ because our outcome is chosen, and we can see the sorting method, but it’s up to each tree to categorize and separate variables by features.

Classification models will tell us which category something falls in to. The categories are defined by the programmer at the beginning. An example of a classification model could use a random forest would be a model that determines whether incoming emails should spam go in your ‘inbox’ or ‘spam’ folder.

To create the model, we make two categories that our Y can fall in to; spam, and not spam. We program the model to look for keywords or certain email address that may indicate spam. The presence of words like “buy” or “offer” will help the model determine whether the email message falls into the category of spam or not spam. The algorithm takes in data, and over time, it learns by comparing its predictions to the actual value of the output. Over time it makes small adjustments to its model so that the algorithm becomes more efficient over time.

Classifications

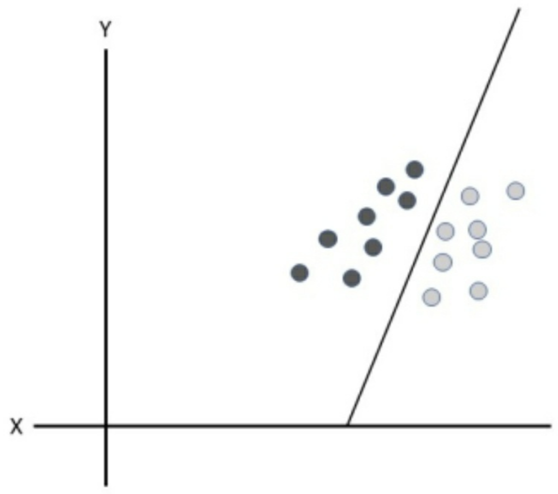
A few times in this book, we’ve referred to classification models. While some of the models we’ve already mentioned are capable of classification, the following are more supervised learning models that are specifically used for classification.

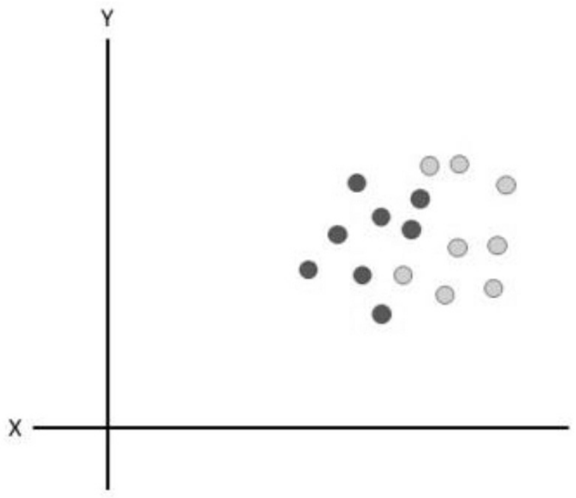
Classification requires labeled data and creates non-continuous predications. In classification problems, the graphs are non-linear. There could be two classes in a classification problem, or even more. Classification models are probably the most widely used part of machine learning and data science.

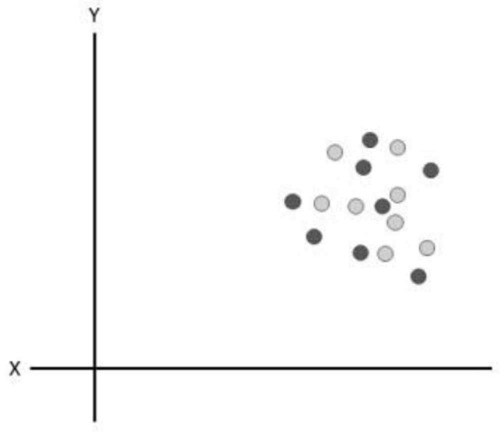
The first type of classification is binary classification. With binary classification, the data is classified into two categories, denoted by 1 or 0. We call it binary classification because there are only two possible categories, and all of our data falls into one or the other.

But there are instances when we have more than two categories, and for this, we use multi-class classification models. Also, we have linear decision boundaries, where data is separated on either side of a line. Not all data can be classified into either side of a decision boundary.

The first picture has an example of a classification using a linear decision boundary. In the second image, there are almost two classes, but they are not linearly separable. In the third image, data points are mixed, and linear boundary classification is not possible. Depending on the type of data you are using, there are different model choices that will be better suited for different tasks







Logistic Regression/Classification

This method is used to classify dependent and categorical variables. Logistic regression calculates probabilities based on independent variables. It gives the variables the value “Yes or No” to sort them. Typically used with binary classification.

When you can’t separate the data into classes by a linear boundary, like in the examples above, this is the method you must use this. It’s one of the most common types of machine learning algorithms. It doesn’t just sort into categories, but it also tells us the probability that a category exists.

We denote this model by taking the odds function, where p is the probability of an event;

$$\frac{p}{1 - p}$$

And creating a formula called the logit

$$\log\frac{p}{1 - p}$$

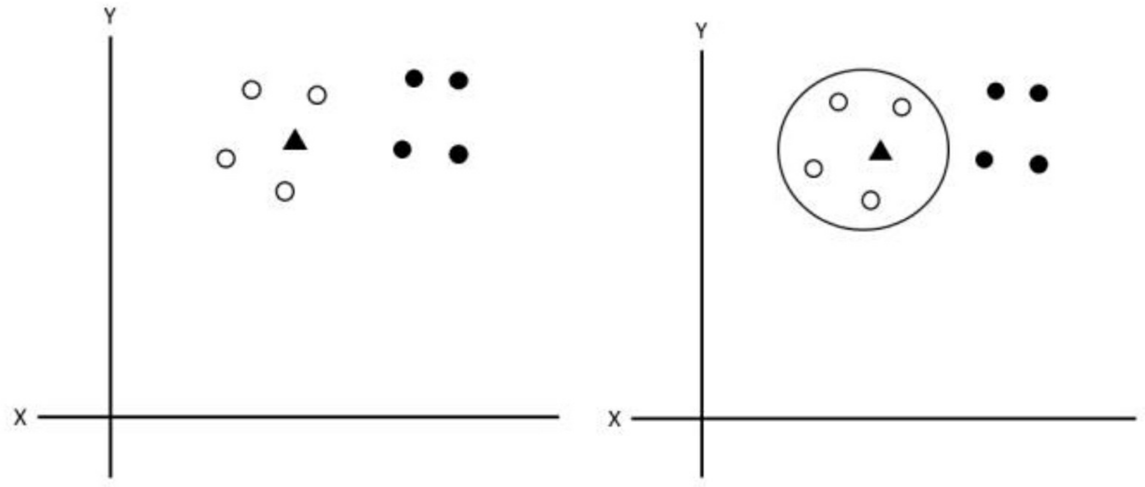
K Nearest Neighbors

K-nearest neighbors are one of the most straightforward and widely used methods of data classification. It’s a form of supervised learning used for both classification and regression, and it’s also the most basic clustering algorithm. Simply put, it’s about taking a data point and putting it with the most common and nearest group on the scatterplot.

In KNN, a new data point is classified by the average median value of its neighbors K. The nearest neighbors to a new data point ‘vote' for which classification it falls in to. K is the number of nearest neighbors that are voting in the model. Set k to a number- this is how many closest data points the new data point will analyze to choose which one it fits with. The closeness of data points is measured using Euclidean distance.

Take the following two images as an example. We have our data split into two classifications; the white dots and the black dots. A new data point is introduced, the triangle, and we’d like to predict which classification it falls in to.

In this model, we have chosen that K=4. If you choose k=4, then the four closest data points are analyzed. Whichever class is most prevalent amongst the neighboring data points is the class that the new data point will be placed in. In this case, you can see in the image on the right that the four white dots are the nearest classification. Therefore, the new data point is classified in that class.



There are a few factors to consider when you are choosing the value for k. The higher the number for k is, the closer we will get to the true classification of our new data point. There is an optimal point where the value of k should stop increasing to avoid overfitting.

If you choose to use a number for K that is too low, then there is a likelihood that your model will suffer from a high level of bias. If you use a number that is too high, then the computing power required to calculate the value will be too costly. You might consider choosing to use an odd number when you choose a value for K, rather than an even number. If you use an odd number, it is less likely that you will encounter a tie between classes voting on a data point. Data scientists often choose the number 5 as a default setting for k.

If you use a large number for K, this will be very data-intensive. Large data sets are also tough to use with KNN machine learning models. If you are using larger data sets, then you must calculate the distance between hundreds or maybe thousands of data points. It also doesn't perform well when you use this method on a model that exists in more than two dimensions. Again, it has to do with the computing power required to calculate this distance between many data points.

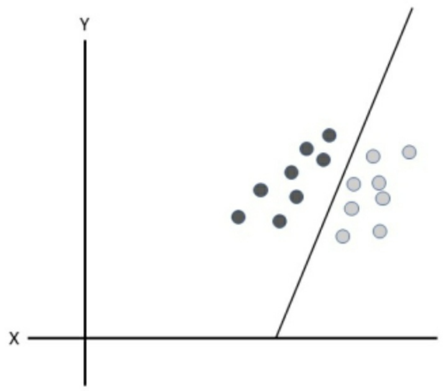
Support Vector

Support vector is another type of classifier. It classifies using a hyperplane. Generally, we would use a support vector model with smaller datasets, where it performs quite well.

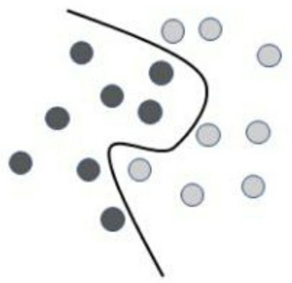
Kernel Support vector

Although we will touch on kernel support vectors, later on, they are used to sort classes that can't be separated with a linear separation line. The separation line can take on many shapes (linear, nonlinear, polynomial, radial basis, sigmoid).

Recall the classification by a linear boundary that we just talked about, where the classification looks something like the following image:



In this image, our data can be classified by a straight line that separates the two distinct categories of data. It would be convenient if data was always separable like this, but unfortunately, it's not always so neat and tidy in fact, most of the time you will have to separate the data in a way that looks more like this:



In this example, the data can’t be separated by a linear boundary line. So instead we must use a technique called the kernel trick. It uses a measure of similarity between data points to classify them.

Naïve Bayes

Recall Bayes theorem from the first section on supervised learning. With Naïve Bayes models, predictors are assumed to be independent. This model is easy to use and helpful in large datasets. Its often employed to help sort spam emails.

We use Baye’s rule here. The idea of Bayes Rule is that by adding new, relevant information to what we already know, we can update our knowledge based on that new information. If we wanted to know the probability of there being rain this afternoon, we could just figure out what percentage of days it rains per year. But then we found out that it rained this morning. How do you think this will affect the probability that it will rain this afternoon?

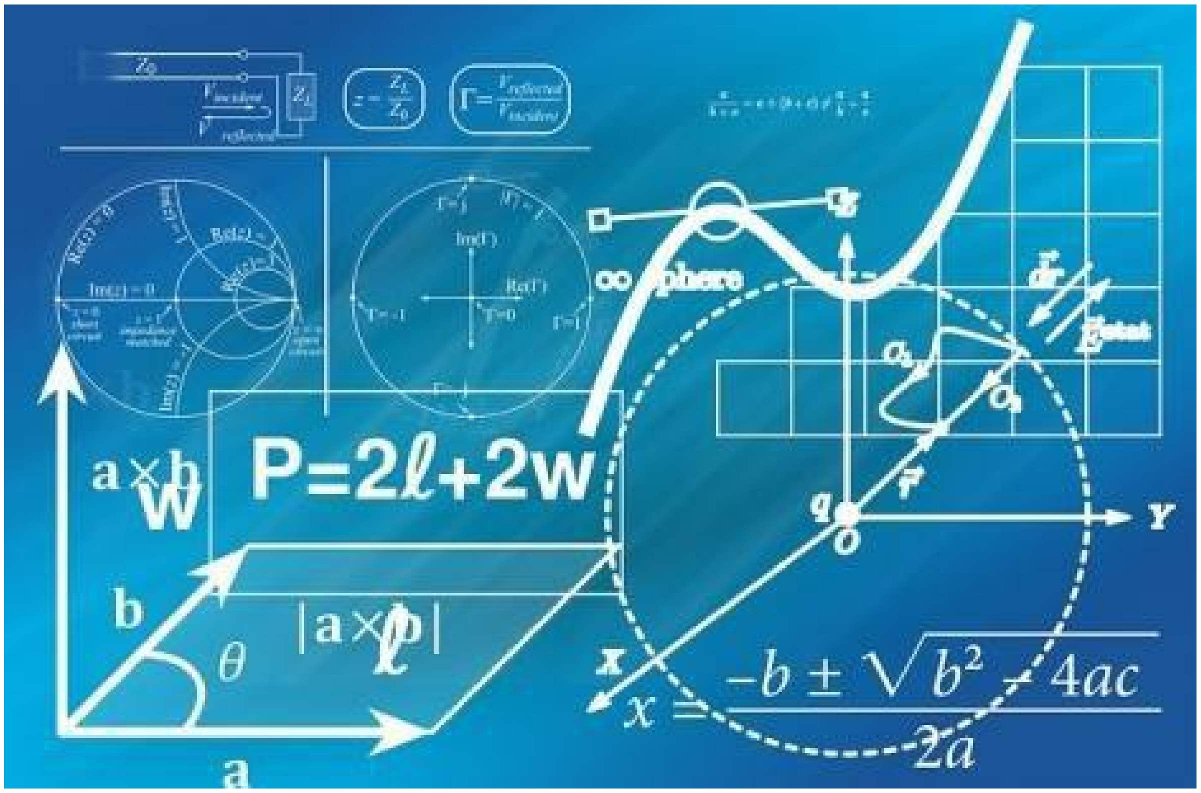
So, our ability to predict the probability of something will improve when we receive more information about the event.

Mathematically, Bayes theorem is expressed as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

So, we can classify new data points using Bayes theorem. The way that works is when we are given a new data point, we calculate the probability of that data point falling into a category based on the features of that data point.

Unsupervised Learning



Unsupervised machine learning uses unlabeled data. Data scientists don’t know the output of yet. The algorithm must discover patterns on its own, where patterns would otherwise be unknown. Find a structure in a place where the structure is otherwise unobservable. The algorithm finds data segments on its own. The model looks for patterns and structure in an otherwise unlabeled and unrecognizable mass of data. Unsupervised learning allows us to find patterns that would be unobservable without computer scientists. Sometimes massive collections of data have patterns, and it would be impossible to sift through all of it trying to find trends.

This is good for examining the purchasing habits of consumers so that you can group customers into categories based on patterns in their behavior. The model may discover that there are similarities in buying patterns between different subsets of a market, but if you didn’t have your model to sift through these massive amounts of complicated data, you will never even realize the nature of these patterns. The beauty of unsupervised learning is the possibility of discovering patterns or characteristics in massive sets of data that you would not be able to identify without the help of your model.

A good example of unsupervised learning is fraud detection. Fraud can be a major problem for financial companies, and with large amounts of daily users, it can be difficult for companies to identify fraud without the help of machine learning tools. Models can learn how to spot fraud as the tactics change with technology. If you want to deal with new, unknown fraud techniques, then you will need to employ a model that can detect fraud under unique circumstances.

In the case of detecting fraud, it's better to have more data. Fraud detection services must use a range of machine learning models to be able to combat fraud effectively. Using both supervised and unsupervised models. It's estimated that there will be about $32 billion in fraudulent credit card activity next year, in 2020 Models for fraud detection classify the output (credit card transactions) as legitimate or fraudulent.

They can classify based on a feature like time of day or location of the purchase. If a merchant usually makes sales around $20, and suddenly has a sale for $8000 from a strange location, then the model will most likely classify this transaction as fraudulent.

The challenge of using machine learning for fraud detection is the fact that most transactions are not fraudulent. If there was even a significant amount of fraudulent transactions among non-fraudulent, then credit cards would not be a viable industry. The percentage of fraudulent card transactions is so small that it can create models that are skewed that way. The $8000 purchase from a strange location is suspicious, but it is more likely to be the result of a traveling cardholder than fraudulent activity. Unsupervised learning makes it easier to identify suspicious buying patterns like strange shipping locations and random jumps in user reviews.

Clustering

Clustering is a sub-group of unsupervised learning. Clustering is the task of grouping similar things together When we use clustering, we can identify characteristics and sort our data based on these characteristics. If we are using machine learning for marketing, clustering can help us identify similarities in groups of customers of potential clients. Unsupervised learning can help us sort customers into categories that we might not have created with the help of machine learning. It can also help you sort your data when you are working with a large number of variables.

K-Means clustering

K-means clustering works similarly to K-nearest neighbors You pick a number for k to decide how many groups you want to see. You continue to cluster and repeat until clusters are more clearly classified.

Your data is grouped around centroids, which are the points on your graph that you have chosen where you want to see your data clustered. You choose them at random, and you have k of them. Once you introduce your data to the model, data points are placed in categories indicated by the closest centroid, which is measured by Euclidean distance. Then you take the average value of the data points surrounding each centroid. Keep repeating this process until your results stay the same, and you have consistent clusters. Each data point is only assigned to one cluster.

You repeat this process by finding the average values for x and y within each cluster. This will help you extrapolate the average value of the data points in each cluster. K-means clustering can help you identify previously unknown or overlooked patterns in the data.

Choose the value for k that is optimal for the number of categories you want to create. Ideally, you should have more than 3. However, the advantage associated with adding more clusters diminishes that higher the number of clusters you have. The higher the value for k that you choose, the smaller and more specific the clusters are. You wouldn’t want to use a value for k that is the same as the number of data points because each data point would end up in its own cluster.

You will have to know your dataset well and use your intuition to guess how many clusters are appropriate, and what sort of differences that will be present. However, our intuition and knowledge of the data are less helpful once we have more than just a few potential groups.

Dimensionality Reduction

When you are using dimensionality reduction, you are trimming down data to remove unwanted features. Simply put, you're scaling down the number of variables in a dataset.

When we have a lot of variables in our model, then we run the risk of having dimensionality problems. Dimensionality problems are problems that are unique to models with large datasets and can affect prediction accuracy. When we have many variables, we need larger populations and sample populations in order to create our model. With that many variables, it’s hard to have enough data to have many possible combinations to create a well-fitting model.

If we use too many variables, then we can also encounter overfitting. Overfitting is the main problem which would cause a data scientist to consider dimensionality reduction.

We must choose data that we don’t need, or that is irrelevant. If we have a model predicting someone’s income, do we need a variable that tells us what their favorite color is? Probably not. We can drop it out of our dataset. Usually, it's not that easy to tell when a variable should be dropped. There are some tools we can use to determine which variables aren’t as important.

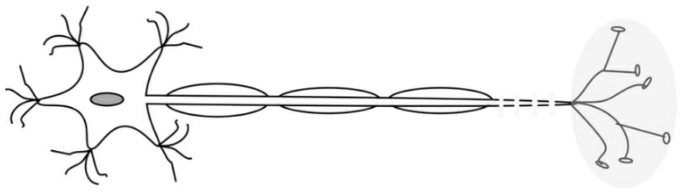
Principle Component Analysis is a method of dimensionality reduction. We take the old set of variables and convert them into a newer set somehow. The new sets we’ve created are called principal components. There is a tradeoff between reducing the number of variables while maintaining the accuracy of your model.

We can also standardize the values of our variables. Make sure they are all valued in the same relative scale so that you don't inflate the importance of a variable. For example, if we have variables measured as a probability between 0 and 1 vs. variables that are measured by whole numbers above 100.

Linear Discriminant is another method of dimensionality reduction where we combine features or variables, rather than get rid of them altogether.

Kernel Principal Component is the third method for dimensionality reduction. Here, variables are placed in a new set. This model will be non-linear, and it will give us even better insight into the true parameters than original data.

Neural networks



Neural networks are a form of machine learning that is referred to as deep learning. It’s probably the most advanced method of machine learning, and truly understanding how it works might require a Ph.D. You could write an entire book on machine learnings most technical type of model.

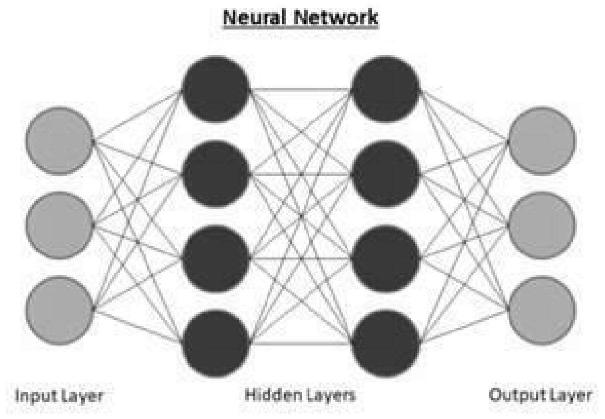
Neural networks are computer systems designed to mimic the path of communication within the human brain. In your body, you have billions of neurons that are all interconnected and travel up through your spine and into your brain. They are attached by root-like nodes that pass messages through each neuron one at a time all the way up the chain until it reaches your brain.

While there is no way to replicate this with a computer yet, we take the principle idea and apply it to computer neural networks to replicate the ability to learn like a human brain learns; recognize patterns and inferring information from the discovery of new information.

In the case of the neural networks, as with all our machine learning models. Information is processed through neural networks as numerical data. By giving out numerical data values, we are giving it the power to use algorithms to make predictions.

Just as with the neurons in the brain, data starts at the top and works its way down, being first separated into nodes. The neural network uses nodes to communicate through each layer. A neural network is comprised of three parts; Input, hidden, and output layers.

In the picture below, we have a visual representation of a neural network, with the circles being every individual node in the network. On the left side, we have the input layer; this is where our data goes in. After the data passes through the input layer, it gets filtered through several hidden layers. The hidden layers are where data gets sorted by different characteristics and features. The hidden layers look for patterns within the data set. The hidden layers are where the ‘magic' is happening because the data is being sorted by patterns that we probably wouldn't recognize if we sorted it manually. Each node has a weight which will help to determine the significance of the feature being sorted.



The best use of these neural networks would be a task that would be easy for a human but extremely difficult for a computer. Recall at the beginning of the book when we talked about reasoning and inductive reasoning. Our human brain is a powerful tool for inductive reasoning; it’s our advantage over advanced computers that can calculate high numbers of data in a matter of seconds. We model neural networks after human thinking because we are attempting to teach a computer how to ‘reason’ like a human. This is quite a challenge. A good example of a neural network is the example we mentioned we apply neural networks for tasks that would be extremely easy for a human but are very challenging for a computer.

Neural networks can take a huge amount of computing power. The first reason neural networks are a challenge to process is because of the volume of datasets required to make an accurate model. If you want the model to learn how to sort photographs, there are many subtle differences between photos that the model will need to learn to complete the task effectively. That leads to the next challenge, which is the number of variables required for a neural network to work properly. The more data that you use and the higher the number of variables analyzed means that there is an increase in hidden networks. At any given time, several hundred or even thousands of features are being analyzed and classified through the model. Take self-driving cars as an example. Self-driving cars have more than 150 nodes for sorting. This means that the amount of computing power required for a self-driving car to make split-second decisions while analyzing thousands of inputs at a time is quite large.

In the instance of sorting photos, neural networks can be very useful, and the methods that data scientists use are improving rapidly. If I showed you a picture of a dog and a picture of a cat, you could easily tell me which one a cat was, and which one was a dog. But for a computer, this takes sophisticated neural networks and a large volume of data to teach the model.

A common issue with neural networks is overfitting. The model can predict the values for the training data, but when it's exposed to unknown data, it is fit too specifically for the old data and cannot make generalized predictions for new data.

Say that you have a math test coming up and you want to study. You can memorize all the formulas that you think will appear on the test and hope that when the test day comes, you will be able to just plug in the new information into what you've already memorized. Or you can study more deeply; learning how each formula works so that you can produce good results even when the conditions change. An overfitted model is like memorizing the formulas for a test. It will do well if the new data is similar, but when there is a variation, then it won't know how to adapt. You can usually tell if your model is overfitted if it performs well with training data but does poorly with test data.

When we are checking the performance of our model, we can measure it using the cost value. The cost value is the difference between the predicted value and the actual value of our model.

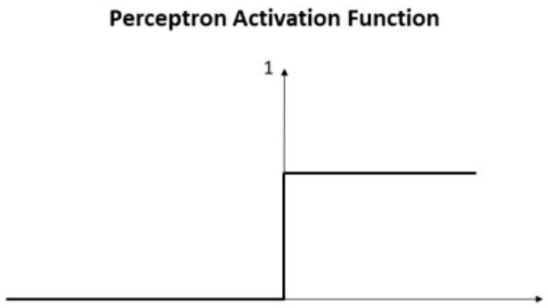
One of the challenges with neural networks is that there is no way to determine the relationship between specific inputs with the output. The hidden layers are called hidden layers for a reason; they are too difficult to interpret or make sense of.

The most simplistic type of neural network is called a perceptron. It's the derives its simplicity from the fact that it has only one layer through which data passes. The input layer leads to one classifying hidden layer, and the resulting prediction is a binary classification. Recall that when we refer to a classification technique as binary, that means it only sorts between two different classes, represented by 0 and 1.

The perceptron was first developed by Frank Rosenblatt. It's a good idea to familiarize yourself with the perceptron if you'd like to learn more about neural networks. The perceptron uses the same process as other neural network models, but typically you'll be working with more layers and more possible outputs. When data is received, the perceptron multiples the input by the weight they are given. Then the sum of all these values is plugged into the activation function. The activation function tells the input which category it falls into, in other words predicting the output.

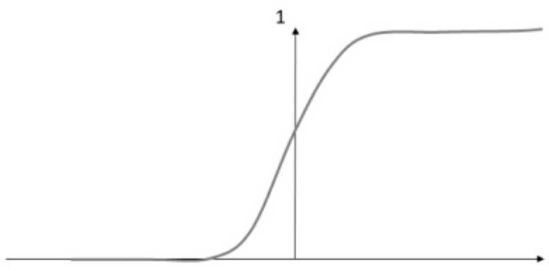
If you were to look at the perceptron on a graph, its line would appear like this:

$f(x) = 0 \text{ if } 0 > x, 1 \text{ if } X \geq 0$



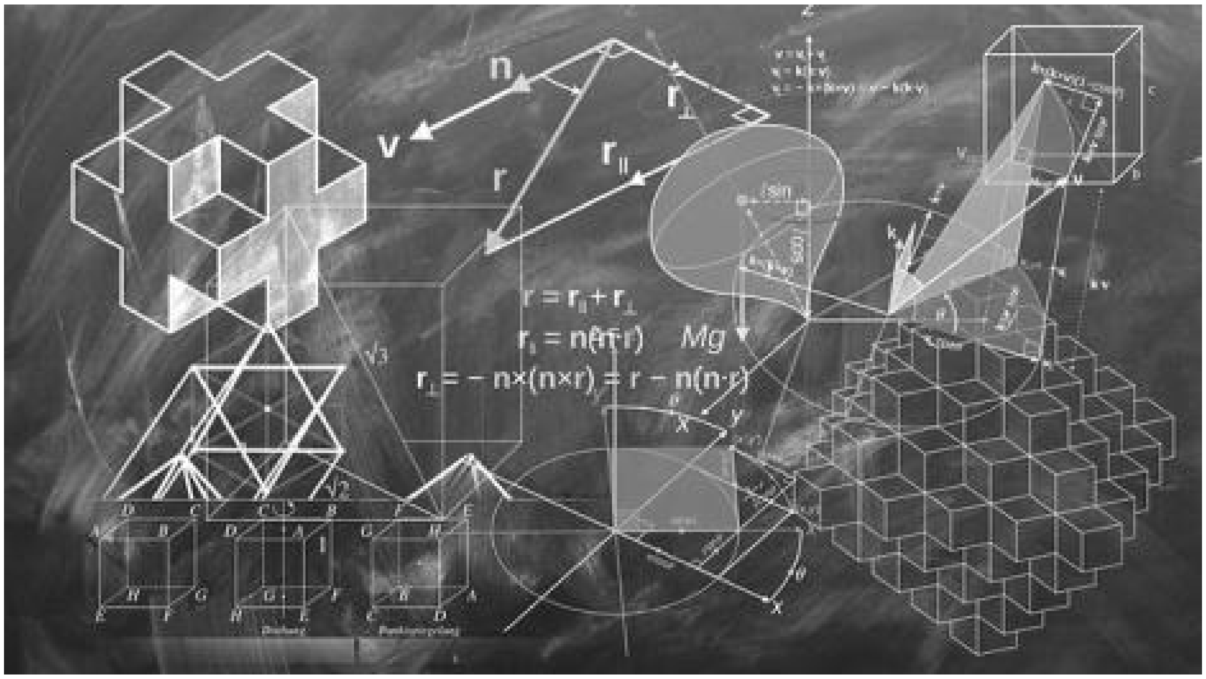
The line of the graph of perception appears like a step, with two values, one on either side of the 1. These two sides of the step are the different classes that the model will predict based on the inputs. As you might be able to tell from the graph, it’s a bit crude because there is very little separate along the line between classes. Even a small change in some input variable will cause the predicted output to be a different class. It won’t perform as well outside of the original dataset that you use for training because it is a step function.

An alternative to the perceptron is a model called a sigmoid neuron. The principle advantage of using the sigmoid neuron is that it is not binary. Unlike perceptron, which can classify data into two categories, the sigmoid function creates a probability rather than a classification. The image below shows the curve of a sigmoid neuron



Notice the shape of the curve around one, where the data is sorted with the perceptron; the step makes it difficult to classify data with just marginal differences. With the sigmoid neuron, the data is predicted by the probability that it falls into a given class. As you can see the line curves at one, which means that the probability that a data point falls into a given class increases after one, but it’s only a probability.

Reinforcement Learning



n
v
r
$r_\perp$
$r_\parallel$
$r = r_\parallel + r_\perp$
$r_\parallel = n(n \cdot r)$
Mg
$r_\perp = - n \times (n \times r) = r - n(n \cdot r)$

Reinforcement learning is our third kind of machine learning. Like unsupervised learning, no inputs are given. The reinforcement learning model must discover on its own how to be the most effective. Then, the data scientist makes suggests improving the model based on its ability to predict an outcome, which is evaluated by looking at comparisons between prediction and actual value. This is the most progressive type of machine learning, and where much of the machine learning in the future will be done.

Think of it like playing a game; over time, you learn by winning and losing. You learn how to win by playing, and the more you become familiar with the game, the better you understand the mechanics of winning. Over time, the data scientist gives the model feedback as it collects and processes data. It receives a reward signal so that it knows when it is doing a good job of predicting some outcome. So 'winning' the game gives positive feedback to the algorithm. This is the type of machine learning used in gaming, robots, navigation, and self-driving cars.

Q Learning

In Q-learning, the model communicates with its environment to improve itself. You begin by having a set of states. States are the things in the environment which stand as obstacles and avenues in your environment. Called "S." In chess, it would be the way that all of your pieces could move, as well as where all of your opponent’s pieces are. These are states.

The possible moves are called ‘A.' If you are a pawn, your possible moves are one square forward. If you're a rook, your possible moves are in any direction in a straight line. Q is the value of the model, which starts at 0. As you play the game, Q goes up and down depending on its interactions with the environment. With negative interactions, the score Q goes down. With positive interactions, the score Q goes up. The algorithm learns how to move so that it can optimize the number Q. In the beginning, it's random. Over time, these random movements result in positive and negative effects on Q, and the machine learns how each move will affect the score of Q. It must play a lot of games to improve the way it plays over time. It's much easier said than done to apply this process in real life.

SARSA State Action Reward State Action There is only a small difference between SARSA and Q. They function similarly in giving the model a reward response.

Deep Q network Deep Q is applied when regular Q isn’t general enough. When it sees new things that it hasn’t seen before, it doesn’t know what to do. Q learning can’t adjust itself for things it has never seen. Deep Q uses a neural network.

Markov Decision Process Has a Set of possible states, a set of models, a set of possible actions, and a real value reward. This model learns by interacting with the environment through ongoing interaction.

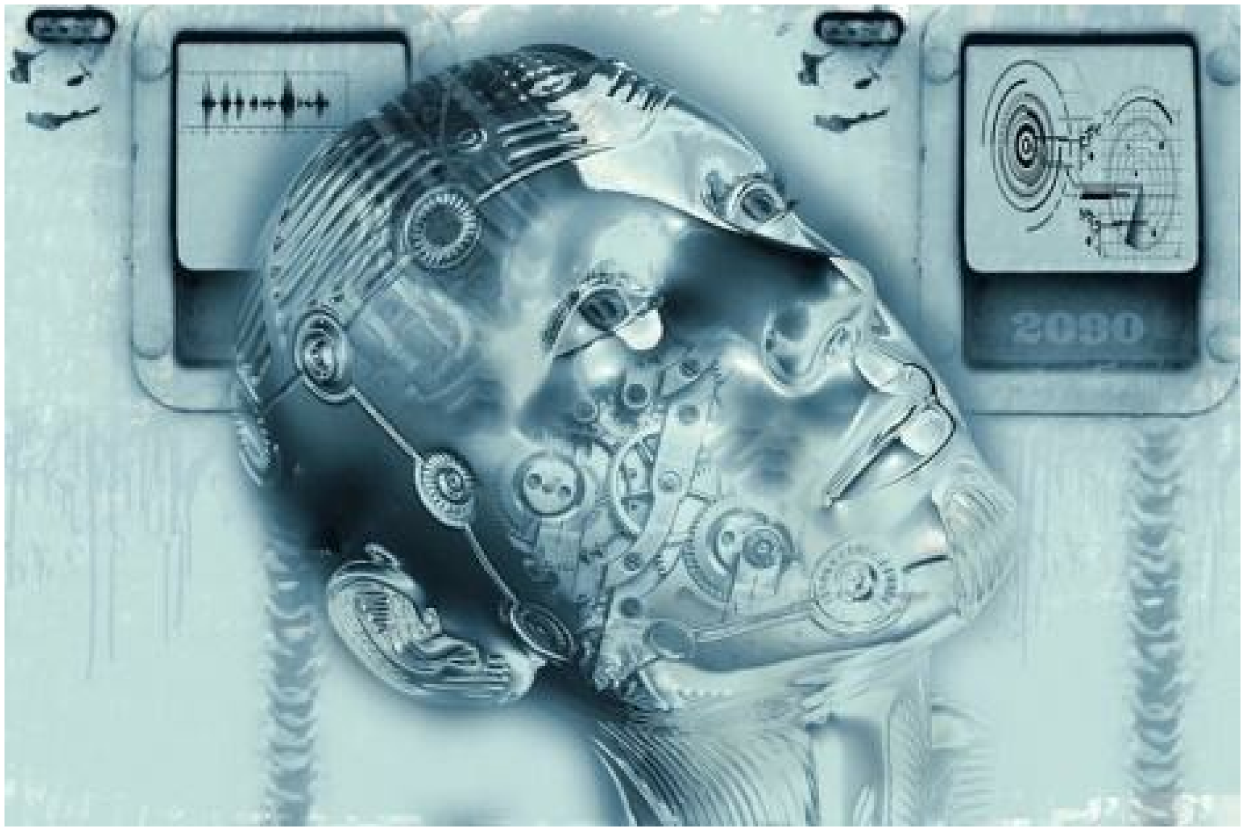
DDPG Deep Deterministic Policy Gradient Another reward state model functions as an actor and a critic.

Semi-Supervised learning

This type of machine learning uses a combination of supervised and unsupervised learning. Some data is labeled while other data is not. But typically, most data will be unlabeled. Semi-supervised learning can be used for classification, regression, and prediction.

Semi-supervised machine learning is helpful because labeling everything would be too time-consuming, and it can harm the ability to find new patterns.

Ensemble Modeling



We’ve learned that diversifying our trees can create a more accurate prediction. But what if, instead of using several versions of the same model, we just used several different models? This is a common trick in machine learning, known as ensemble modeling. By combining information from multiple different types of algorithms, we can improve our model’s accuracy and ability to forecast.

Ensemble modeling is all about the divide and conquers mindset. Different models will give us different insights about the data that may not be recognizable by other models. By combining the information attained from different models, we can learn even more about the truth of our data.

Ensemble modeling also helps to minimize bias and variance in our predictions. Individual models may have prediction errors, but the aggregate of all our predictions will be more accurate.

There are a few different methods to use ensemble modeling:

The first is to take the mode of your predictions. That is, take the value which occurs most frequently across the models. Whichever prediction occurs the most frequently, or has the highest number of ‘votes,’ is the prediction we choose.

We could also take the average of all the predictions, depending on what kind of models we have. The average of all the predictions will be our final prediction. Our ensemble should also consider the reliability of individual models. The results of our models receive different weights, making some predictions more important than others based on reliability.

How do we know which kind of models we want to combine? We already know from this book that there are several types of models to choose from, each giving us different possibilities and advantages.

A common pair of models is using neural networks and decision trees together. Neural networks give us new information, and the decision tree ensures that we have not missed anything.

In addition to the bootstrapping and bagging that we discussed earlier, there are a few other ways of doing ensemble modeling. Data scientists use what’s called a bucket of models. Here they use several different types of models to use with the test data and then choose the one that did the best.

Another idea is called stacking. Stacking uses several different types of models and then uses all the results to give us a prediction that is a combination of all of them.

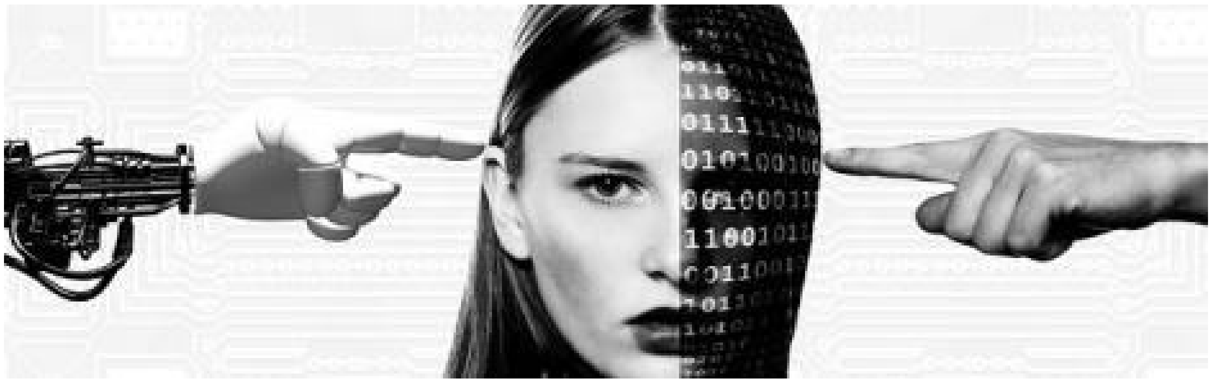
Data scientists like to use ensemble modeling because, with a variety of models, we can usually produce better predictions than with a single model alone.

The drawback to ensemble modeling is that we lose some of our readability. Having multiple models working together at once makes interpretation more difficult, especially when you want to share the data with stakeholders who don’t have data science knowledge.

We can also use several versions of the same model, like how random forests improve the forecast with multiple versions of itself — using neural networks with different sets of nodes and different values for k, or numbers of clusters to see how that changes the outcome of our prediction and find if there is an optimal value for k, or if there are groups or subgroups that we may have overlooked.

It doesn’t do as much when we already have a strong model. But if we combine a few models that have weaker forecasting abilities, then it usually improves the overall accuracy.

Things you must know for machine learning



To be successful with machine learning, you must have the right tools in order to work, just like if you were building a house, you would need to skills and the tools required. The following is a list of the required materials to do machine learning.

Data

To start working with your data, you have to have enough data to break it into two categories; training data and test data.

Training data is the data you use in the beginning when you are building your model. When you are first creating your model, you need to give it some data to learn from. With training data, you will already know the independent variables as well as their respective dependent variables. This means that for every input, you will already know the output of your data. From this data, your model will learn to predict the output on its own. Our training data gives us the parameters we need to make predictions. This is the data that our machine learns from.

Test data is the data that the machine gets once you are satisfied with the model, and you see what it does out in the wild. In this data, we only have the independent variables, but no output. With test data, we can see how well our model does at predicting an outcome with new data.

Your training data should account for most of your data; approximately 70%, while your test data is the remaining 30%. In order to avoid bias, make sure that the data you choose for training data and test data is totally random when you split them up. Don’t choose which data to use; let it be random. Don’t use the same data for training and testing. Start by giving the training data to the machine and examine the relationships between X and Y, then try to see how well your model did.

The most important question to consider during this process is whether your model will still work when it is presented with new data. You can test this by doing cross-validation. This means you will test your model on data you have not used yet. Keep some data to the side that you didn't use during training to see how accurate your model is at the end.

You can also use K-fold validation to check the accuracy of your model. This method is pretty easy to use and generally unbiased. It’s a good technique to use when we don’t have a lot of data to work with for testing. For K-fold validation, we will break our data into k folds, usually between 5 and 10. Test each fold and see how they performed across all the folds once you are finished with testing. Usually, the larger your number for k is the less biased your test will be.

So far, we have talked about models interpreting data to find meaning and patterns. But what kind of data are we going to use? Where will we get our data, and what is it going to look like?

Data is the most critical component for machine learning. After all, your model will only learn with data, so it’s important that you have data that is relevant and meaningful. It came come in many shapes and sizes, structure differently depending on the kinds of data. The more structured the data is, the easier it is to work with. Some data has very little structure, and this data is harder to interpret. Data for facial recognition can be huge and have very little meaning to the untrained eye.

Structured data is more organized. This is the type of data that you will likely use when you are first starting out. It will help you get your feet wet, and you can start understanding the statistic involved in machine learning. Usually, structure data will come in a familiar form that looks something like this, in rows and columns. This is called a tabular dataset.

Market Value	num_bedrooms	num_bathrooms	Sq_ft	pool (Y/N)
$207,367	4	3	2635	N
$148,224	3	2	1800	Y
$226,897	5	3.5	2844	Y
$122,265	2	1.5	1644	N

Recall that a feature is some measurable characteristic of a variable. In each column in a tabular dataset, we see a feature. This feature is some measurable dimension or attribute. Here we have used data reflecting the market value of a house as a function of the number of bedrooms, the number of bathrooms, square footage, and whether the house has a pool. Our market value is the Y; this is our dependent variable. Our independent variables, or our Xs, are num_bedrooms, num_bathrooms, st_ft, and pool.

In supervised learning, you will already have the Y in your dataset. In this case, it's the market value of the home. With enough of this data in our model, even if we don't know the market value of a house we should be able to predict it if we have the number of bedrooms, the number of bathrooms, square footage, and whether the house has a pool or not. Data that is organized in this way is relatively easy to work with and have multiple independent variables like this makes this an example of the multivariate regression.

How much data should you use?

There is no set rule to how much data you will need for your model, but there are guidelines which you should follow. The most important thing is that when you have several independent variables to analyze, then your model will work the best if your data has as many possible combinations of the independent variables as you can get. If you do this, your model will still work even when it encounters a new combination of features that it hasn't seen before. It will have a pretty good way of predicting, even if the combination is completely new.

A good general rule to follow is that you should have about ten times as many respondents as we do independent variables. In the case of our market value example above, we have num_bedrooms, num_bathrooms, sq_ft, and pool. This is four different independent variables, which means we should have at least forty respondents like the ones listed above to create a reliable model.

Having a lot of variables can help us predict the Y more accurately, but that that be costly and make your data harder to process. You must also consider how you are pooling your data. The market values of houses in Los Angeles will be much different than the market values of houses in Cleveland.

It’s also important to keep features as relevant as possible. Having multiple variables will help you make a better prediction, but there are variables that may just create bias in the model.

Refer to Scikit learn to see what they recommend for data sizes for certain types of analysis.

But not all data is useful. We often talk about big data, and it might be easy to assume that the more data we have, the better. But that’s not always the case. Some data may not be helpful. Certain variables might get in the way and may make it harder to find the true answer.

Preparing the Data

So now you have your data, but how do you get it to a point where it's readable by your model? Data will rarely suit our modeling needs right out of the gate. For our data to be formatted properly, it usually requires a round of data cleaning first. The process of data cleaning is often referred to as data scrubbing.

We might have data that comes in the form of images or emails. We need to rewrite it so that it has numerical values that will be interpretable by our algorithms. After all, our machine learning models are algorithms or math equations, so the data needs to have numerical values for it to be modeled.

You might also have pieces of data that were recorded incorrectly or in the wrong format. There may be variables that you don’t need, and you must get rid of. It can be tedious and time-consuming but its extremely important to have data that will work and can easily be read by your model. It’s the least sexy part of being a data scientist.

This is the part of machine learning where you will probably spend most of your time. As a data scientist, you will probably spend about 20% of your time doing data science and the other 80% of your time making sure your data is clean and ready to be processed by your model. We may be combining multiple types of data, and we need to reformat the recordings so that they fit together. First, in the case of supervised learning, pick the variables that you think are most important for your model. If we choose irrelevant variables or variables that don’t matter, we may create a bias and could make our model less effective.

A simple example of cleaning or scrubbing data is recoding a response for gender. On your data, you have a column for male/female. Unfortunately, male and female do not have a numerical value. But you can easily change this by making this a binary variable. Assign female = 1 and male =0. Now you can find a numerical value for the effect that being a female has on the outcome of your model.

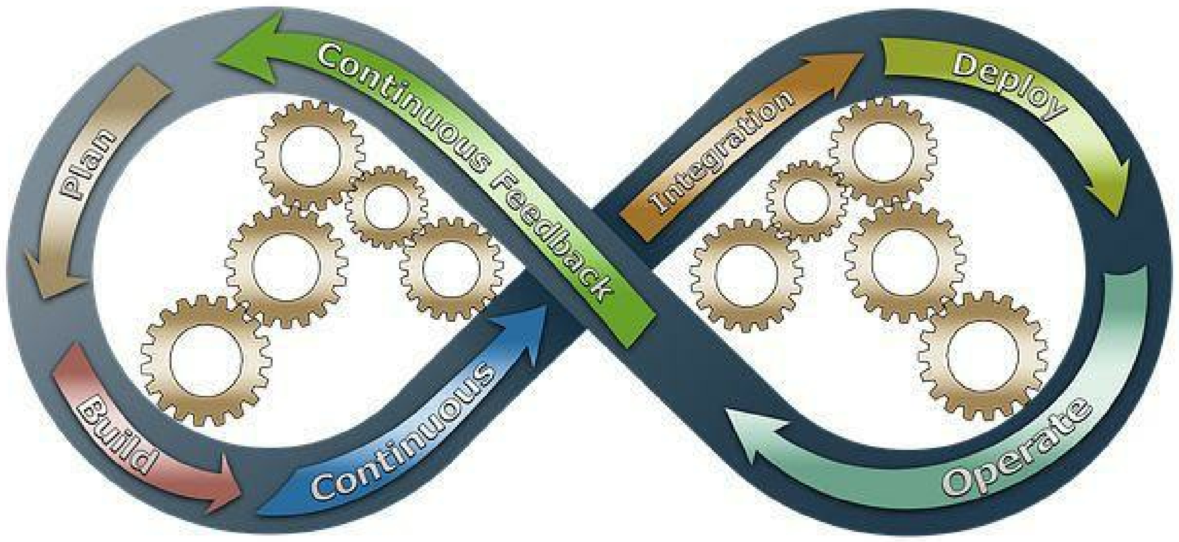
We can also combine variables to make it simpler to interpret. Let’s say you are creating a regression model that predicts a person’s income based on several variables. One of the variables is the education level, which you have recorded in years. So, the possible responses for years of education are 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16. This is a lot of discrete categories. You could simplify it by creating groups. For example, you could rewrite variables 1, 2, 3, 4, 5, 6, 7, 8 = primary_ed and rewrite 9, 10, 11, 12 = secondary_ed and rewrite 13, 14, 15, 16 = tertiary education. Instead of having twelve categories, you have three. Respondents either have some primary education, secondary education, or some level of post-secondary or college-level education. This is known as binning data, and it can be a good way to clean up your data if it’s used properly.

When you are combining variables to make interpretation simpler, you must consider the tradeoff between having more streamlined data and losing some important information about relationships in the data. Consider that in this example, by combining these variables into three groups instead of sixteen, you may be creating bias in your model.

There a lot of factors that could require you to clean your data. Even a misspelling or an extra space somewhere in your data can have a negative impact on your model.

You might have data that is missing. In order to fix this situation, you can replace the missing values with either the mode of the median of that variable. It's possible to remove data with missing values if there are only a few, but this just means you'll have fewer data to use in your model.

Programming Tools



In order to process your data, you will need special programming tools so you can tell the data what you want it to do. We have already mentioned that machine learning is a branch of computer science. This is where that comes into play.

In the introduction, we said that the three most common languages for data science are Python, R, and C++. Choosing which one is right for you will depend on your experience and what you are planning to do with your data.

The most common language for data science is python. It was created in 1991 by Guido Van Rossum, and it's notable because it is easier to read than other programming languages. It's still being developed and improved today. It’s not complicated to learn and is compatible with most relevant data types. It also has applications outside of mere data manipulation that will be useful in machine learning.

Python has several free packages that you can install which have been created to give you shortcuts to common data science tools. These packages have shortcuts for codes that are commonly used in machine learning, which makes you must do less of the work.

Pandas is a must-have library of tools for data scientists working with python. It allows you to manipulate time-series data and tabular datasets more easily. It shows your data in rows and columns so that it’s easier to manage, in much the same way that you would look at data in Microsoft excel. It’s easy to find online and free to download. Pandas is helpful when you’re looking at datasets in .CSV format.

Numpy is a helpful program to do data processing faster using python. It works similarly to Matlab, and it can handle matrices and multi-dimensional data. It will help you import large datasets easier.

Scikit-learn is another library of the machine learning function. With Scikit learn, you will have easy access to many of the algorithms that we have mentioned earlier, which are commonly used in machine learning. Algorithms like classification, regression, clustering, support vector, random forest, and k-means have shortcuts so that a lot of the grunt coding is done for you.

R is the third option. It's free to use and open source. R can be used for both data mining and machine learning. It's popular for those who are new to data science because of its availability. It can’t handle the larger datasets required for more advanced machine learning operations, but it's not a bad place to start if you are new to data science and computer programming.

In order to run these programs, you will need a computer. Usually, a regular laptop or desktop computer will be powerful enough to process smaller and medium-sized datasets, especially when you are new to machine learning.

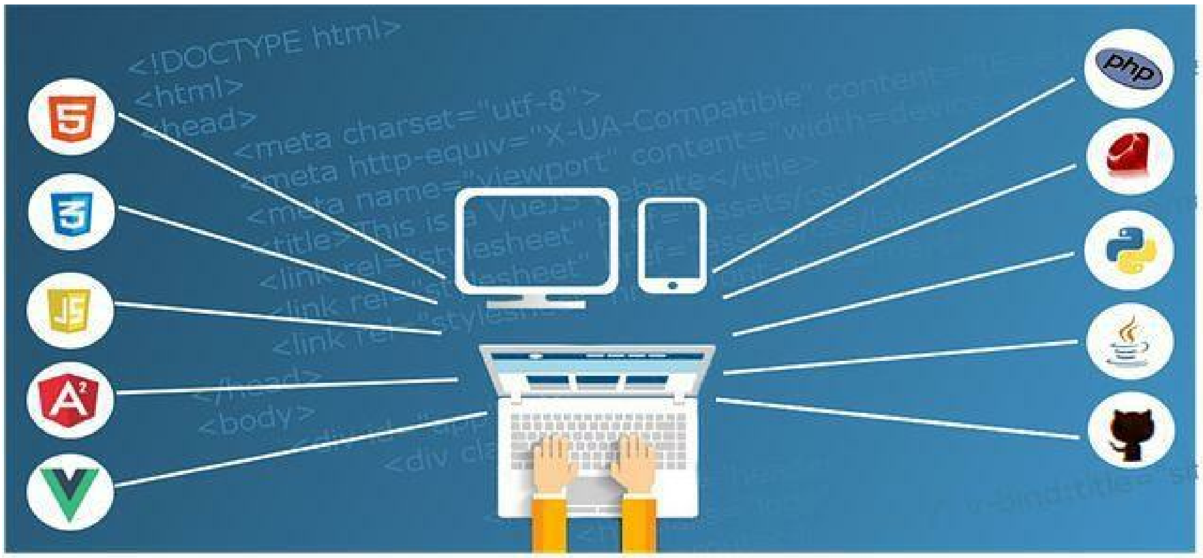
Although GPUs (Graphics Processing Units) have existed for some time, their accessibility has increased in recent years, which makes data science more accessible. It has been a breakthrough in the field of data science because the field is no longer limited to labs with massive computers.

GPUs are known for being the power behind video games. They allow a computer to interpret multiple points at once, which is essential for processing volumes of data. Now GPUs allow us to do a lot more with much less computer hardware. The predecessor, CPU cores head multiple control units that allowed information to be processed all at once. Rather than having multiple control units, the GPU has a much larger web of cores that can all handle different processes all at once. One GPU card can contain almost 5000 processors. It’s a major advancement for artificial intelligence and machine learning. They can help make the processing of neural networks much faster.

C and C++ are other common data analysis languages. The advantage of C++ is that it is a very powerful language. It can process massive data sets very quickly. Data scientists using massive datasets often choose to use C++ for its speed and processing power, especially when working with datasets over a terabyte. C++ can process one gigabyte of data in about a second. This makes it especially useful for deep learning algorithms, neural network models with 5-10 layers, and huge datasets. This type of model might be overwhelming for software that isn't as fast. If you are doing more advanced machine learning and you have multiple GPUs, then C++ might be the language for you. C++ is capable of just about anything; it's a very versatile language.

The downside is that the libraries in C++ aren’t as extensive as those in Python. This means that when you are writing code for your data and model, you’ll likely be starting from scratch. No matter what kind of projects you decide to do, there will be roadblocks as you write your code. Having a library that can help you when you get stuck will enable you to learn and work faster.

Developing Models



You will need to get yourself set up in Python or some other programming language to do machine learning. You create machine learning models by using code to manipulate the datasets. While this book doesn't cover coding for machine learning, I will give you a quick rundown of some basic libraries and packages that I recommend you install for machine learning.

Because it's the most common language used in data science, we will use Python as an example throughout this chapter. I also think it's the most practical language to learn if it's your first language because it's more readable than other programming languages, and it has a wide range of abilities beyond machine learning.

Once you've installed the latest version of Python, there are a few recommended libraries to install which come with a lot of commands that will be useful to your work with machine learning. All of these can be found easily with a quick google search, and they can be downloaded for free.

The most important library for data analysis and machine learning in Python is called Pandas. It's quite a popular choice for datasets and will make your coding easier and faster, especially when you are still trying to get a feel for things.

Anaconda for python

Another option for getting yourself started with Python is installing Anaconda. The great thing about Anaconda is that it gives you every package for Python so that you don't have to install packages one at a time while you write the program for your model. It comes with all the libraries you will need, for just about every different kind of function.

Anaconda is a free and open-source program that will work in both R and Python. With Anaconda, you'll have access to several libraries that will help you with your data science projects. Basically, this gives you a pre-packaged collection of all the python libraries, of which there are over 100 libraries.

One of the major libraries is Spyder and Jupyter. Both of these are integrated development environments, meaning they are the window where you will write your code, but they are more developed than a standard command window and have options to save and export/import codes.

Most Python users will start in a development environment called IDLE. It’s very simple and offers a good format for learning how to code in python. When you install Python on a windows computer, it will come automatically included. If you have a Linux computer, it is available, but you will need to install it separately.

IDLE will make those baby steps in Python easier because you’ll be able to save your scripts and edit them later. It will also walk you through debugging.

To install Anaconda, visit:

docs.anaconda.com/anaconda/install

Scroll down until you see a list of operating systems. Choose your operating system. It will give you instructions for installing anaconda on their website, based on your operating system. Then you're ready to start messing around in Python. I highly recommend using one of the free beginner Python tutorials that are available on the internet. EdX has a free beginner tutorial in Python, which is a great place to start. Also, take advantage of forums like Reddit, where there have been a vast number of common questions already answered in detail, and members are always sharing relevant news from the world of machine learning.

Algorithms

Once you have your data, and the hardware and software to manipulate it, you need to bring them together. Put your data on your programming software. Find a dataset for free online to work with when you are first starting out. Kaggle.com is free and has a lot of data sets to choose from in CSV format, which will be easy to work with once you have Pandas library imported into your Python.

The best algorithms to start with are linear and logistic regression for supervised learning and k-means clusters in unsupervised learning. These will be relatively easy starting out, and you can build towards other models from there.

Visualization tools

You have your data, and now you have created models using one of the programming languages, and you have a whole collection of data science libraries to help you do all this faster. Your computer is running well, and you can create models independently.

You may have created models that display interesting results, but in order to break it down into layman's terms and communicate your findings with stakeholders, you'll need to organize it in a way that's easy to visualize. If you're a data scientist on a marketing project, you may have created a model that helps break down customers into categories and predicts trends in buying habits. But if you want to communicate these results to the rest of your marketing team, you'll need to find a way to communicate so that even people who aren't familiar with data science can understand your results. Breaking down your data into charts and graphs and visual will help compliment your analytics skills. Being able to make visualizations of your data is extremely important when you are communicating with an audience who isn't familiar with data analysis

A popular toolset for data professionals is Tableau. Tools like these are called data visualization software. At some companies, there are employees whose entire job involves taking hard to read data and presenting it in a way that is easy to visualize.

Software like tableau is very commonly used by businesses that rely on data to make decisions. Tableau is useful because its relatively easy to use, and data can be viewed in real-time through its platform. You can customize a dashboard of tools for creating reports and charts with your data. It also gives you the ability to share your results with other people from your company. Tableau can be used to create graphs and scatterplots from that data that you have analyzed in your programming language.

More advanced things which are useful

These tools may not be as relevant to you when you are just beginning, but it might be interesting to talk about some of them and consider what may be useful down the road. This book may just the beginning on your path to be a machine learning expert, so you may refer to this list later when you are a little more advanced.

You should continue to think about the management of unstructured data. Usually, this requires more advanced programs because it is more difficult to manage and manipulate. This type of data often takes on the form of something much too complicated for the human brain to analyze without the assistance of tools, but this is the direction where machine learning is heading. Using neural networks to mimic the functions of human thought, who knows what the future will hold.

The further we get with machine learning, the bigger our data is getting. Possibilities of machine learning are expanding all the time. The data that is important in the future won't have the neat structure we are accustomed to, like the kind of data that can fit in an excel sheet.

This type of data also requires beefier computer hardware and software to be able to handle the processing of these large quantities of information. Usually use some sort of cloud computing software to carry the large volumes of information, as well as a GPU specific to data analytics. This higher level of computing can help to process multiple moving points at once. The math required also becomes harder. Combining algorithms.

Afterword

Hopefully, after reading this book, you have a good understanding of the basic principles of machine learning. You've now been introduced to several different types of popular machine learning models and their uses. We've explored how advanced data scientists are using machine learning to create predictions and the parameters that they need to create predictions that are accurate and dependable when introduced to new data.

The beauty of data science and machine learning is the broad range of applications. If you go out and gain machine learning experience, then there is a broad range of jobs and opportunities working with all different kinds of data. Whether you like the competition and the rush of business, and you want to use models that predict the rise and fall in stocks or guess what a customer will buy next. Or maybe you have an interest in medicine and health care; you can apply machine learning to improve cancer diagnosis and get new insight into the characteristics of a disease and how it will affect different individuals. Wherever your interests may lie, there is an opportunity for machine learning to be implanted to improve upon what we can already do.

The more that the world becomes connected, the more data will become available. Almost everyone has some type of smart device recording and tracking of their user data. Data scientists are finding more creative ways to learn from and interpret that data. Machine learning is a way for data scientists to examine trends that are beyond the scope of human comprehension, which means that our predictions will continue to get more accurate and our data more useful.

Computers will only continue to get more powerful, and that power will become more and more accessible which means that machine learning and data science will no longer be just buzzwords but widely applied methods of finding valuable information. It’s not just large businesses utilizing data and machine learning anymore; it’s becoming more feasible for even smaller businesses to incorporate big data into their decision-making processes.

Now that you know the basic theory of machine learning, its time for you to keep going and find ways to apply and practice the knowledge. If you are interested in being a data scientist that specializes in machine learning, this book is only the beginning of the process. I highly recommend you commit yourself to learn a language like Python, R, or C++. It’s the next step in becoming a data scientist, and in applying these machine learning theories into actual models and algorithms. Thanks to the internet, there is a vast number of books, videos, and tutorials available for free that will take you through the process of learning computer languages. There has never been a better time to learn how to code and create your own models. This book is only a small piece of a large collection of information available on the subject. If you’re serious about machine learning, then this shouldn’t be the only book you read.

You can find entire books describing the process of specific models. Neural networks are a field that is so advanced that you could find entire books based on that specific type of model alone. It's not a bad idea to pick up a few statistics study guides so that you can refer to them when you have a question. Be on the lookout for possible sources of data that you might be able to utilize, and potential questions that may be interesting to explore with statistical mathematics.

The job opportunities alone are enough reason to pursue further knowledge in the field. There is a shortage of experienced data scientists who can apply the methods and techniques listed in this book. This means there is an opportunity for someone who wills to get their hands dirty and start coding their own models. There are businesses and organizations out there, right now, searching for people who can use this information properly. Keep in mind that this demand won't exist forever. Already, universities across the globe are creating new degree programs specifically geared towards data science as a blend of computer science and statistics. This means that the next generation of data scientists is already on their way.

So, begin your learning now. Find an online tutorial and some free datasets online and start figuring out how to regress and classify, using this book as a guide. Learn each model one at a time. Find examples on the internet of finished models and try to see if you can replicate the results. It takes time to learn programming languages, so be patient and seek out new opportunities to learn and adapt your skills.

Try one of the online communities specific to statistical modeling in order to cut your teeth and learn from what other data scientists are doing. I advise you to check out Kaggle.com. It’s a website that hosts statistical modeling competition for aspiring data scientists. Different companies and organizations post contests with the datasets included. It's a great way to experiment with a variety of tasks and get data to work with. Old contests are also available online, with a large number of associated tutorials on youtube.com and other data science communities for you to learn from. It's probably the best way for an aspiring data scientist to expand his/her resume and network with other data scientists.

MACHINE LEARNING MATHEMATICS

Study Deep Learning through Data Science. How to Build Artificial Intelligence through Concepts of Statistics, Algorithms, Analysis, and Data Mining

Samuel Hack

Introduction

Congratulations on purchasing **Machine Learning Mathematics:** *Study Deep Learning through Data Science. How to Build Artificial Intelligence Through Concepts Of Statistics, Algorithms, Analysis, and Data Mining* and thank you for doing so.

The following chapters will discuss the fundamental concepts of machine learning algorithms and the need for machine learning in resolving modern-day business problems. You will find a detailed explanation of the four different types of machine learning algorithms available in the market today along with the importance of machine learning in the first chapter of this book. Representation, evaluation, and optimization make up the three core concepts of machine learning that are explained in detail. You will be introduced to the concept of “Statistical Learning”, which is a descriptive statistics-based machine learning framework that can be categorized as supervised or unsupervised.

In chapter 2 of this book titled "Machine Learning Algorithms", you will learn development and application of some of the most popular supervised machine learning algorithms, with explicit details on linear regression, logistic regression, and Naïve Bayes classification algorithms. In chapter 3 titled "Neural Network Learning Models", it will provide you an overarching guide for everything you need to know for successful development of neural network models by learning how to build data pipelines for your machine learning models and then following specific neural network training approaches. The end-to-end process described in this chapter will provide you an overarching view on how to generate your desired machine learning model from scratch with a focus on neural network models. You will also learn the various components and functions at play in the Artificial Neural Network and Perceptron (single neuron-based network) models as well as various applications of these advance and futuristic machine learning models to resolve everyday business problems.

In the 4th chapter of this book titled, "Learning Through Uniform Convergence", we will take a deep dive into the overlap of machine learning with the field of statistics. One of many borrowed statistical concepts used in the development of machine learning models is "Uniform Convergence", which allows the developer to identify the learnability of the problem at hand based on the data sample size using empirical risk minimizers. You will gain a thorough understanding of the concept of "General Setting of Learning" introduced by Vapnik in 1995 and continues to be central to the concept of machine learning development. A statistical explanation of the impact of “Uniform Convergence” on learnability as a prerequisite using finite classes is provided, along with a discussion on potential learnability without “Uniform Convergence”.

The final chapter of this book will provide you a holistic overview of various cutting edge data science technologies like data mining and Artificial Intelligence. The most highly recommended lifecycle for structured data science projects is the "Team Data Science Process" (TDSP) is explained in exquisite detail along with various deliverables that need to be generated at every stage. You will also learn how data science is being leveraged by businesses in their decision-making process. The power of artificial intelligence has already started to manifest in our environment and our everyday objects. So you need to learn the difference between Business Intelligence and Data Science technology. This book is filled with real-life examples to help you understand the nitty-gritty of the concepts and names and description of multiple tools that you can further explore and selectively implement in your business to reap the benefits of these cutting-edge technologies.

There are plenty of books on this subject on the market, thanks again for choosing this one! Every effort was made to ensure it is full of as much useful information as possible, please enjoy!

Chapter 1: Introduction to Machine Learning

The notion of Artificial Intelligence Technology is derived from the idea that computers can be engineered to exhibit human-like intelligence and mimic human reasoning and learning capacities, adapting to fresh inputs and performing duties without needing human intervention. The principle of artificial intelligence encompasses machine learning. Machine Learning Technology (ML) refers to the principle of Artificial Intelligence Technology, which focuses mainly on the designed ability of computers to learn explicitly and self-train, identifying information patterns to enhance the underlying algorithm and making autonomous decisions without human involvement. In 1959, the term "machine learning" was coined during his tenure at IBM by the pioneering gaming and artificial intelligence professor, Arthur Samuel.

Machine learning hypothesizes that contemporary computers can be trained using targeted training data sets, which can readily be tailored to create required functionality. Machine learning is guided by a pattern-recognition method where previous interactions and outcomes are recorded and revisited in a way that corresponds to its present position. Because machines are needed to process infinite volumes of data, with fresh data constantly flowing in, they need to be equipped to adapt to the fresh data without being programmed by a person, considering the iterative aspect of machine learning. Machine learning has close relations with the field of Statistics, which is focused on generating predictions using advanced computing tools and technologies. The research of “mathematical optimization” provides the field of machine learning with techniques, theories, and implementation areas. Machine learning is also referred to as “predictive analytics” in its implementation to address business issues. In ML, the “target” is known as “label”, while in statistics, it’s called “dependent variable”. A “variable” in statistics is known as “feature” in ML. And a “feature creation” in ML is known as “transformation” in statistics.

ML technology is also closely related to data mining and optimization. ML and data mining often utilize the same techniques with considerable overlap. ML focuses on generating predictions based on predefined characteristics of the given training data. On the other hand, data mining pertains to the identification of unknown characteristics in a large volume of data. Data mining utilizes many techniques of ML, but with distinct objectives;

similarly, machine learning also utilizes techniques of data mining through the "unsupervised learning algorithms" or as a pre-processing phase to enhance the prediction accuracy of the model. The intersection of these two distinct research areas stems from the fundamental assumptions with which they operate. In machine learning, efficiency is generally assessed about the capacity of the model to reproduce known knowledge, while in “knowledge discovery and information mining (KDD)” the main job is to discover new information. An “uninformed or unsupervised” technique, evaluated in terms of known information, will be easily outperformed by other “supervised techniques”. On the contrary, “supervised techniques” can not be used in a typical “KDD” task owing to the lack of training data.

Data optimization is another area that machine learning is closely linked with. Various learning issues can be formulated as minimization of certain “loss function” on training data set. “Loss functions” are derived as the difference between the predictions generated by the model being trained and the input data values. The distinction between the two areas stems from the objective of “generalization”. Optimization algorithms are designed to decrease the loss of the training data set. The objective of machine learning is to minimize the loss of input data from the real world.

Machine learning has become such a "heated" issue that its definition varies across the world of academia, corporate companies, and the scientific community. Here are some of the commonly accepted definitions from select sources that are extremely known:

- *“Machine learning is based on algorithms that can learn from data without relying on rules-based programming.”* – McKinsey.
- *“Machine Learning, at its most basic, is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world.”* – Nvidia
- *“The field of Machine Learning seeks to answer the question, how can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?”* – Carnegie Mellon University
- *“Machine learning is the science of getting computers to act without being explicitly programmed.”* – Stanford University

Types of Machine Learning

Supervised Machine Learning

The "supervised machine learning" is widely used in predictive big data analysis because they can assess and apply the lessons learned from previous iterations and interactions to new data set. These learning algorithms are capable of labeling all their current events based on the instructions provided to efficiently forecast and predict future events. For example, the machine can be programmed to label its data points as "R" (Run), "N" (Negative) or "P" (Positive). The machine-learning algorithm then labels the input data as programmed and gets the correct output data. The algorithm compares the production of its own with the "expected or correct" output, identifies potential modifications, and resolves errors to make the model more accurate and smarter. By employing methods like "regression", " prediction", "classification", and "boosting of ingredients" to properly train the learning algorithms, any new input data can be fed to the machine as "target" data set to assemble the learning program as desired. This jump-starts the analysis and propels the learning algorithms to create an "inferred feature", which can be used to generate forecasts and predictions based on output values for future events. Financial organizations and banks, for example, depend heavily on machine-learning algorithms to track credit card fraud and foresee the likelihood of a potential customer not making their loan payments on time.

Unsupervised Machine Learning

Companies often find themselves in a situation in which data sources are required to generate a labeled and categorized training data set are unavailable. In these conditions, the use of unsupervised machine learning is ideal. “Unsupervised learning algorithms” are commonly used to describe how the machine can produce "inferred features" to illustrate hidden patterns from an unlabeled and unclassified component in the stack of data. These algorithms can explore the data so that a structure can be defined within the data mass. Although the unsupervised machine learning algorithms are as effective as the supervised learning algorithms in the exploration of input data and drawing insights from it, the unsupervised algorithms are not capable of identifying the correct output. These algorithms can be used to define data outliers; to produce tailor-made product suggestions; to classify text topics using techniques such as "self-organizing maps”, "singular value decomposition" and "k-means clustering". Customer identification, for example, customers can be segmented into groups with shared shopping attributes and targeted with similar marketing strategies and campaigns. Consequently, unsupervised learning algorithms are very common in the online marketing industry.

Semi-Supervised Machine Learning

The "semi-supervised machine learning algorithms" are extremely flexible and able to learn from both “labeled” as well as “unlabeled” or raw data. These algorithms are a "hybrid" of supervised and unsupervised ML algorithms. Usually, the training data set consists of predominantly unlabeled data and a tiny portion of labeled data. The use of analytical methods such as the "forecast", "regression" and "classification" in combination with semi-controlled learning algorithms allows the computer to improve its accuracy in learning and training significantly. These algorithms are often used when the production of processed and labeled training data from the raw data set is highly resource-intensive and less cost-effective for the company. Companies are using their systems with semi-supervised learning algorithms to prevent additional personnel and equipment expenses. For example, the application of technology for "facial recognition" requires an enormous quantity of facial data dispersed across multiple input sources. The processing, classification, and labeling of raw data obtained from sources including internet cameras require a lot of resources and thousands of hours to be used as a training data set.

Reinforcement Machine Learning

The "reinforcement machine learning algorithm" learns from its environment and is much more unique than any of the previously discussed machine learning algorithms. Such algorithms perform activities and carefully record the results of each action, either as an error for a failed outcome or a reward for excellent results. The two main characteristics that distinguish the reinforcement learning algorithm are the "trial and error" analysis technique and the "delayed reward" feedback loop. The computer continually analyzes input data using a variety of calculations and transmits a signal of reinforcement for each correct or intended output to eventually optimize the final results. The algorithm creates an easy action and rewards feedback loop for assessing, recording, and learning what activity has been efficient, in that it resulted in right or intended output in a shorter time. The use of such algorithms enables the system to determine optimal conduct automatically and to maximize its effectiveness in a specific context. Therefore, in the disciplines of gaming, robotics, and navigation systems, the reinforcement machine-learning algorithms are heavily utilized.

Importance of Machine Learning

The seemingly unstoppable interest in ML stems from the same variables that have made “data mining” and “Bayesian analysis” more common than ever before. The underlying factors contributing to this popularity are increasing quantities and data varieties, cheaper and more effective computational processing, and inexpensive data storage. To get a sense of how significant machine learning is in our everyday lives, it is simpler to state what part of our cutting edge way of life has not been touched by it. Each aspect of human life is being impacted by the "smart machines" intended to expand human capacities and improve efficiencies. Artificial Intelligence and machine learning technology is the focal precept of the “Fourth Industrial Revolution” that could question our thoughts regarding being "human".

All of these factors imply that models that can analyze larger, more complicated data while delivering highly accurate results in a short time can be produced rapidly and automatically on a much larger scale. Companies can easily identify potential growth opportunities or avoid unknown hazards by constructing desired machine learning models that meet their business requirements. Data runs through the vein of every company. Increasingly, data-driven strategies create a distinction between winning or losing the competition. Machine learning offers the magic of unlocking the importance of business and customer data to lead to actionable measures and decisions that can skyrocket a company’s business and market share.

Machine learning has demonstrated over the recent years that many distinct tasks can be automated which were once deemed as activities only people could carry out, such as image recognition, text processing, and gaming. In 2014, Machine Learning and AI professionals believed the board game "Go" would take at least ten years for the machine to defeat its greatest player in the world. But they were proved mistaken by "Google's DeepMind", which showed that machines are capable of learning which moves to take into account even in such a complicated game as "Go". In the world of gaming, machines have seen much more innovations such as "Dota Bot" from the "OpenAI" team. Machine learning is bound to have enormous economic and social impacts on our day to day lives. Complete set of work activities and the entire industrial spectrum could potentially be automated and the labor market will be transformed forever.

“Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.”

- *SAS*

Repetitive Learning Automation and Information Revelation

Unlike robotic automation driven by hardware that merely automates manual tasks, machine learning continuously and reliably enables the execution of high quantity, high volume, and computer-oriented tasks. Artificial intelligence machine learning algorithms help to adapt to the changing landscape by enabling a machine or system to learn, to take note of and reduce its previous mistakes. Machine learning algorithm works as a classifier or a forecasting tool to develop unique abilities and to define data pattern and structure. For instance, an algorithm for machine learning has created a model that will teach itself how to play chess and even how to create product suggestions based on consumer activity and behavioral data. This model is so effective because it can easily adjust to any new data set.

Machine learning allows assessment of deeper and wider data sets using neural networks comprising several hidden layers. Just a couple of years ago, a scheme for detecting fraud with countless hidden layers would have been considered a work of imagination. A whole new world is on the horizon with the emergence of big data and unimaginable computer capabilities. The data on the machines is like the gas on the vehicle, more data addition leads to faster and more accurate results. Deep learning models thrive with a wealth of data because they benefit from the information immediately. The machine-learning algorithms have led to incredible accuracy through the «deep neural networks». Increased accuracy is obtained from deep learning, for instance, from the regular and extensive use of smart technology such as "Amazon Alexa" and "Google Search." These "deep neural networks" also boost our healthcare sector. Technologies like image classification and the recognition of objects are now able to detect cancer with the same precision as a heavily qualified radiologist on MRIs.

Artificial intelligence enables the use of big data analytics in combination with the algorithm for machine learning to be enhanced and improved. Data has developed like its currency and can readily become "intellectual property" when algorithms are self-learning. The crude information is comparable to a gold mine in that the more and more you dig, the more you can dig out or extract "gold" or meaningful insights. The use of machine learning algorithms for the data allows faster discovery of the appropriate solutions and can make these solutions more useful. Bear in mind that the finest data will always be the winner, even though everyone uses similar techniques.

"Humans can typically create one or two good models a week; machine learning can create thousands of models a week."

- Thomas Davenport, The Wall Street Journal

Core Concepts of Machine Learning

Today, there are several kinds of ML, but the notion of ML is mainly based on three components "representation", "evaluation", and "optimization". Here are some of the standard concepts that apply to all of them:

Representation

Machine learning models can not directly hear, see, or sense input examples. Data representation is therefore needed to provide a helpful vantage point for the model in the main data attributes. The choice of significant characteristics that best represent data is very essential to train a machine learning model effectively. "Representation" simply refers to the act of "representing" data points to a computer in a language that it understands using a set of classifiers. A classifier may be defined as "a model that inputs a vector of discrete and/or ongoing function values and outputs a single discrete value called "class". To learn from the represented data, a model must have the desired classifier in the training data set or "hypothesis space" that you want the models to be trained on. The data features used to represent the input are very critical to the machine learning system. Any "classifier" that is external to the hypothesis space cannot be learned by the model. For developing a required machine learning model, data characteristics are so essential that it can easily be the difference between successful and unsuccessful machine learning projects.

A training data set with several independent “features” which are well linked to the "class" can make learning much easier for the machine. On the other side, it may not be easy for the machine to learn from the class with complex functions. This often requires the processing of the raw data so that the desired features for the ML model can be built from it. The method of deriving features from raw data set tends to be the ML project's most time-consuming and laborious component. It is also considered to be the most creative and interesting part of the project where intuition and "trial and error" play just as important a role as the technical requirements. The ML process is not a "one-shot" process of developing and executing a training data set, but an iterative process requiring analysis of the post-execution output, followed by modification of the training data set. Domain specificity is another reason why the training dataset requires comprehensive-time and effort. Training data set to produce predictions based on consumer behavior analysis for an e-commerce platform will be very distinct from the training data set needed to create a self-driving car. Nevertheless, in the industrial sectors, the core machine learning mechanism stays the same. No wonder, there is a lot of research going on to automate the process of feature engineering.

Evaluation

Essentially, in the context of ML "evaluation", is referred to as the method of assessing various hypotheses or models to select one model over another. An "evaluation function" is needed to distinguish between effective classifiers from the vague ones. The evaluation function is also known as the "objective," "utility," or "scoring" function. The machine-learning algorithm has its internal evaluation function that is usually very different from the researchers' external evaluation function used to optimize the classifier. Usually, the evaluation function is described as the first phase of the project before selecting the data representation tool. For example, the self-driving car machine learning model has the feature to identify pedestrians in its vicinity at near-zero, false-negative, and low false-positive rate as an "evaluation function" and the pre-existing condition that needs to be "represented" using applicable data features.

Optimization

The process of searching the hypothesis space of the represented machine learning model to identify the highest-scoring classifier and achieve better evaluation is called "optimization." For algorithms with more than one optimum classifier, selecting the optimization method is very critical in determining the generated classifier and achieving a more effective model of learning. There are a variety of "off-the-shelf optimizers" on the market to kick off new machine learning models before replacing them with custom-designed optimizers.

Statistical Learning Framework

“Statistical learning” is a descriptive statistics-based learning framework that can be categorized as supervised or unsupervised. “Supervised statistical learning” includes constructing a statistical model to predict or estimate output based on single or multiple inputs, on the other hand, “unsupervised statistical learning” involves inputs but no supervisory output, but helps in learning data relationships and structure. One way of understanding statistical learning is to identify the connection between the “predictor” (autonomous variables, attributes) and the “response” (autonomous variable), in order to produce a specific model which is capable of predicting the “response variable (Y)” on the basis of “predictor factors (X)”.

“$X = f(X) + \varepsilon$ where $X = (X1,X2, . . .,Xp)$”, where “f” is an “unknown function” & “ ε ” is “random error (reducible & irreducible)”.

Here are some fundamental concepts of Statistical Learning:

Prediction and Inference

If there are several inputs "X" easily accessible, but the output "B" production is unknown, "f" is often treated as a black box, provided that it generates accurate predictions for "Y". This is called "prediction". There are circumstances in which we need to understand how "Y" is influenced as "X" changes. We want to estimate "f" in this scenario, but our objective is not simply to generate predictions for "Y". In this situation, we want to establish and better understand the connection between "Y" and "X". Now "f" is not regarded as a black box since we have to understand the underlying process of the system. This is called "inference". In everyday life, various issues can be categorized into the setting of "predictions", the setting of "inferences", or a "hybrid" of the two.

Parametric and Non-parametric Techniques

The “parametric technique” can be defined as an evaluation of “f” by calculating the set parameters (finite summary of the data) while establishing an assumption about the functional form of “f”. The mathematical equation of this technique is “$f(X) = \beta 0 + \beta 1X1 + \beta 2X2 + . . . + \beta pXp$”. The “parametric models” tend to have a finite number of parameters which is independent of the size of the data set. This is also known as “model-based learning”. For example, “k-Gaussian models” are driven by parametric technique.

On the other hand, “non-parametric technique” generates an estimation of “f” on the basis of its closeness to the data points, without making any assumptions on the functional form of “f”. The “non-parametric models” tend to have a varying number of parameters which grown proportionally with the size of the data set. This is also known as “memory-based learning”. For example, “kernel density models” are driven by a non-parametric technique.

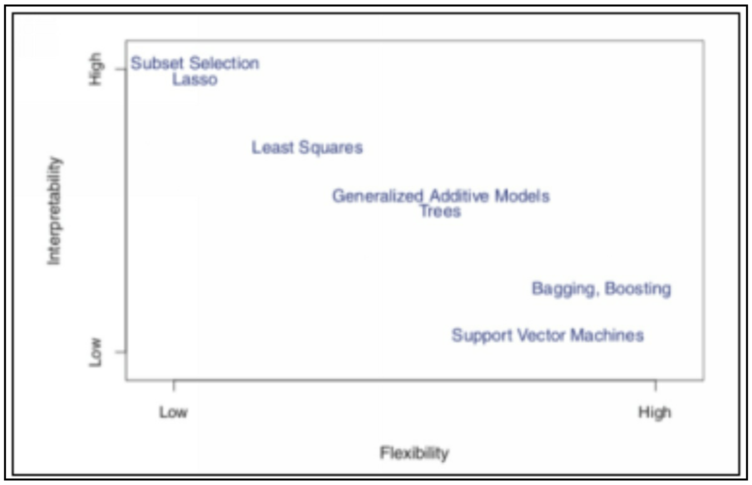
Predictions Accuracy and Model Interpretability

Some of the many methods used to learn from statistical data are less adaptable and extremely restrictive. When "inference" is the target, the use of easy and comparatively inflexible techniques of statistical learning has significant benefits. On the other hand, if the target is the generation of forecasts and predictions flexible models are preferred.

The performance of the model can be estimated based on its accuracy to predict the occurrence of an event on new input data. A more accurate model is deemed as a more valuable model. Interpretability of the model offers insight into the input-output relationship. An interpreted model can provide insight into the capability of independent features to generate predictions for the dependent attribute. The problem occurs because, at the expense of interpretability, as model accuracy improves so does the complexity of the model.

A more accurate model can offer a business more possibilities, advantages, time, or money. But the model accuracy needs to be optimized for such prediction. The optimization of accuracy extends the complexity of the model even further by introducing additional model parameters (and resources needed to adjust those parameters). It is much easier and quicker to interpret a model with a relatively small number of parameters. An input coefficient and an intercept term are part of a linear regression model. For instance, every single term can be explored to assess how it contributes to the production of the output. Switching to a logistic regression model provides greater authority In the context of the relationships underlying the potential transformation of a function to output, that too should be explored along with the coefficients.

It is relatively easy to understand a decision tree of small size, but a heavily loaded decision tree needs a distinct perspective to understand why the event is predicted to occur. Furthermore, the optimized combination of several models into one prediction tends to have no significant or timely interpretation. Interpretation is deemed ancillary to model accuracy.
For example, models designed to separate and classify “spam” emails from “non-spam” emails as well as models designed to evaluate the price of a real estate.



Assessing model accuracy

There is no-one-size-fits-all or jack-of-all-trades technique in the field of statistics, it is impossible for a single technique to dominate across the vast variety of data sets. The most frequently used metric in the “regression" environment is the "mean squared error (MSE)", which can be used for quantification of the extent to which the “predicted answer value” is near to the true answer value for the target observation. For the predicted responses that are extremely close to the true responses, the MSE is computed as small but if the predicted and true responses vary considerably concerning some of the observations then the MSE is computed as large. For example, assume we have clinical data for some patients such as their weight, blood pressure, gender, age, history of family illnesses along with information stating if they are diabetic or not. This patient-related data set can be used to train a statistical technique for predicting diabetes risk based on clinical measures.

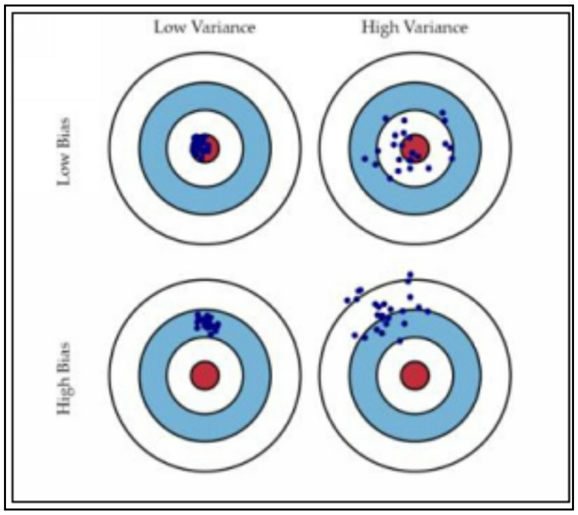
$$\boxed{MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,}$$

The most frequently used metric in the "classification" environment is the "confusion matrix". The core characteristic of statistical learning is that with continuous learning the model becomes more flexible and the training errors are reduced, although the test error may not be decreased.

Bias and Variance

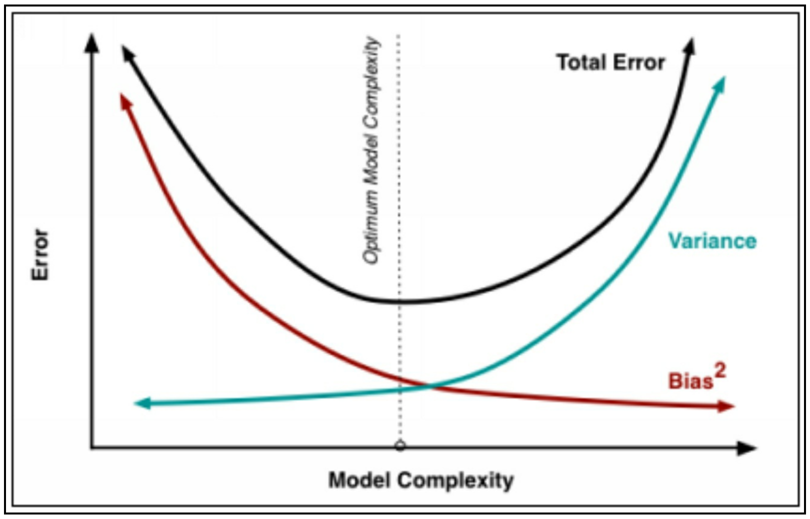
In the context of machine learning, bias is defined “the simplified assumptions made by a model to further simplify the learning of the target task”. Parametric models are designed with an inherent strong bias, which makes learning much faster and simpler but significantly reduces the overall flexibility of the model for a wider variety of application. “Decision trees", "k-nearest neighbors", and "support vector machines" are categorized as low-bias algorithms for machine learning models. The “high-bias” ML algorithms are "linear regression", "linear discriminant analysis", and "logistic regression".

In the context of machine learning, a variance can be defined “as the amount by which the estimation of the target function will be altered with the use of a different training data set”. Non-parametric models" with high flexibility tend to have a high variance score. “Linear regression", "linear discriminant analysis", and "logistic regression" are low-variance ML algorithms. “Decision Trees", "k-Nearest neighbors", and "Support Vector Machines” are high variance ML algorithms.



The Trade-Off between Bias and Variance

In the context of statistical learning, “bias” and “variance” are inversely related. For a model exhibiting high bias, the variance score will be reduced significantly and vice versa. There is a compromise that needs to be made between these two factors, which drives the selection of the model and its configuration to resolve the targeted issue by achieving a fine balance between the two. The right level of flexibility is critical to the efficiency and performance of any statistical learning technique in both the "regression" and "classification" environments. The trade-off between "bias" and "variance" of the model and the subsequent "U-shape" in the test error poses a major challenge.



Chapter 2: Machine Learning Algorithms

Machines are now able to learn from and train on their own by using previous computations and underlying algorithms to produce high-quality, easily reproducible decisions and results. Machine learning has been around for a long time now, but the recent developments in machine learning algorithms have made it possible for machines to process and analyze large volumes of data efficiently. This is accomplished by using high speed and frequency automation to apply advanced and complex mathematical calculations to the machines. The sophisticated computing machines of today can rapidly evaluate the ginormous volumes of data and produce faster and more accurate results. Companies that use machine learning algorithms have improved flexibility to adapt the training data set to meet their business requirements and train the machines accordingly. These tailor-made machine learning algorithms allow businesses to identify potential hazards and growth opportunities. Typically, in collaboration with artificial intelligence technology and cognitive technologies, machine learning algorithms are used to produce computers that are highly effective and extremely efficient in processing huge amounts of information or big data and to produce highly accurate results.

Hundreds and thousands of machine learning algorithms have already been generated as this research field continues to proliferate. Here are some of the most commonly used algorithms categorized on the basis of its type of machine learning:

To refresh your memory, “supervised learning” is driven by the data scientists who provide guidance for teaching the algorithm of what conclusions it must make, using predefined training data set. “Supervised learning” requires information on all possible outputs of the algorithm and the training data set that has already been labeled with expected or correct results.

Let’s look at the two of the most renowned supervised learning algorithms used to develop machine learning models, in detail!

Regression

The "regression" techniques fall under the category of supervised machine learning. They help predict or describe a particular numerical value based on the set of prior information, such as anticipating the cost of a property based on previous cost information for similar characteristics. Regression techniques vary from simple (such as "linear regression") to complex (such as “regular linear regression”, “polynomial regression”, “decision trees”, “random forest regression” and “neural networks”, among others).

The simplest method of all is **“linear regression”**, where the line's "mathematical equation (y= m*x+b) is used to model the data collection”. Multiple "data pairs (x, y)" can train a "linear regression" model by calculating the position and slope of a line that can decrease the total distance between the data points and the line. In other words, the calculation of the "slope (m)" and "y-intercept (b)" is used for a line that produces the highest approximation for data observations. The data relationships can be modeled with the use of "linear predictor functions",
where unidentified model variables can be estimated from the data. These systems are referred to as "linear models". Traditionally, if values of the "explanatory variables" or "predictors" are known, the conditional mean of the response would be used as the "affinity function" of those values. The use of "conditional media" and other measures in linear models is very rare. Similar to every other form of "regression analysis", the "linear regression" also operates on the "conditional probability distribution" of the responses instead of the joint probability distribution of the variables obtained
with the multivariate analysis.

The most rigorously researched form of regression analysis with wide applicability has been "linear regression". This emanates from the fact that models that rely linearly on their unidentified parameters are easy to work with compared to the models that are non-linearly related to their parameters. As the statistical characteristics of the resulting predictors can be easily determined with a linear distribution. There are many useful applications of "linear regression", which can be categorized into one of the following:

- If the objective is to generate forecasts and predictions or to reduce errors, the predictive model can be matched to an identified dataset and explanatory variables with the use of a linear regression algorithm. Once the model has been developed, any new input data without a response can be easily predicted by the fitted model.
- If the objective is to understand variations in the response variables that may be ascribed to variations in the explanatory variables, "linear regression analysis" could be used to quantify the

relationship between the predictors and the response specifically, to assess if certain explanatory variables lack any linear relationship with the response. It can also be used to identify subsets of predictors containing any data redundancies about the response values.

The fitting of most "linear regression models" is accomplished using the "least squares" approach. However, these model can also be fitted by significantly reducing the "lack of fit" in some other standard (just like the "least absolute deviation regression"), or by minimizing a "penalized version of the least square as done in ridge regression (L2-norm penalty) and lasso regression (L1-norm penalty)". By contrast, it is possible to use the "least square" approach to fit machine learning models that are not linear. Therefore, although the terms "least squares" and "linear model" are strongly connected, they are not the same.

“Multiple Linear Regression" tends to be the most common form of “regression" technique used in data science and the majority of statistical tasks. Just like the “linear regression” technique, there will be an output variable "Y" in "multiple linear regression". However, the distinction now is that we're going to have numerous “X” or independent variables generating predictions for "Y".

For instance, a model developed for predicting the cost of housing in Washington DC will be driven by "multiple linear regression" technique. The cost of housing in Washington DC will be the "Y" or dependent variable for the model. "X" or the independent variables for this model will include data points such as vicinity to public transport, schooling district, square footage, and some rooms, which will eventually determine the market price of the housing.

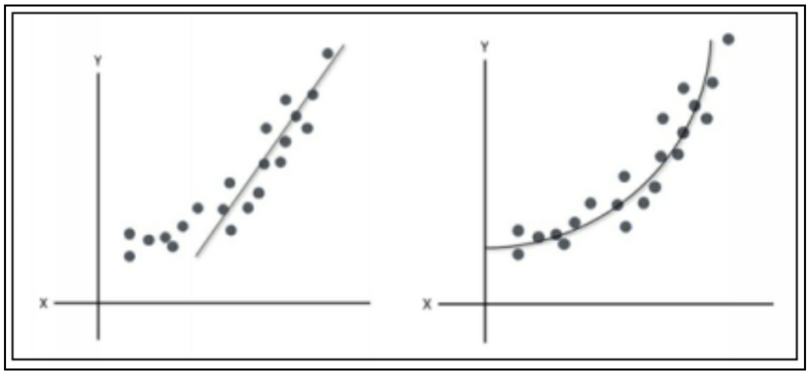
The mathematical equation for this model can be written as below:

“housing_price = β0 + β1 sq_foot + β2 dist_transport + β3 num_rooms”

“Polynomial regression” - Our models developed a straight line in the last two types of "regression" techniques. This straight line is a result of the connection between "X" and "Y" which is "linear" and does not alter the influence "X" has on "Y" as the changing values of "X". Our model will lead in a row with a curve in “polynomial regression”.

If we attempted to fit a graph with non-linear features using "linear regression", it would not yield the best fit line for the non-linear features. For instance, the graph on the left is shown in the picture below has the scatter plot depicting an upward trend, but with a curve. A straight line does not operate in this situation. Instead, we will generate a line with a curve to match the curve in our data with a polynomial regression, like the chart on the right shown in the picture below. The equation of a polynomial will appear like the linear equation, the distinction being that one or more of the "X" variables will be linked to some polynomial expression. For instance,

“Y = mX2+b”



Another significant “regression” technique for data researchers is "**Support Vector Regression”**, which is most frequently used in "case classification". The concept here is to discover a line in space that divides data points into distinct categories. Its also used for regression analysis. It is a form of "binary classification" technique that is not associated with probability.

"Ridge Regression" is a widely used method for analyzing multi-collinear data set. Depending on the features of the data set, using ridge regression correctly can decrease standard errors and significantly improve model accuracy.

Ridge regression can be helpful if your data includes highly correlated independent variables. If you can predict an independent variable with the use of another independent variable, your model will exhibit a high risk of "multi-collinearity". For example, if you use variables that measure the height and weight of a person; these variables in the model are likely to create "multi-collinearity".

Multicollinearity could potentially influence the accuracy of the forecasts and predictions generated by the model. Be mindful of the type of "predictive variables" being utilized in the model to prevent multicollinearity, which could be caused by the type of data you are using, as well as the data collection method. Another reason could be the selection of a small variety of independent variables or the selection of independent variables was very restricted, which resulted in very similar data points.

Multicollinearity can also be induced by generating a highly specific model. Tor that there are more variables than data points in the model. If you have selected to utilize a "linear model" which ended up worsening multicollinearity of the model, then you can attempt to implement a method of "ridge regression".

Ridge regression operates to render the predictions more accurate by permitting a hint of bias into the model. This technique is also referred to as "regularization".

Another technique to enhance the accuracy of the model is by "standardizing" the independent variables. The easiest route is to decrease complexity by changing the values of certain independent variables to null. The approach is not simply to modify these independent variables to null but to implement a structure that rewards values closer to zero. This will trigger the coefficients to decrease, so the model's complexity is also reduced, but the model maintains all of its independent variables. This will offer the model more bias, which is a trade-off for increased accuracy of predictions.

Another technique of reduction is called **"LASSO regression"**. A very complementary technique to the "ridge regression", "lasso regression" promotes the use of simpler and leaner models to generate predictions. In lasso regression, the model reduces the value of coefficients relatively more rigidly. LASSO stands for the "least absolute shrinkage and selection operator". Data on our scatterplot, like the mean or median values of the data are reduced to a more compact level. We use this when the model is experiencing high multicollinearity similar to the "ridge regression" model.

A hybrid of "LASSO" and "ridge regression" methods is known as **"ElasticNet Regression"**. Its primary objective is to further enhance the accuracy of the predictions generated by the "LASSO regression" technique. “ElasticNet Regression" is a confluence of both "LASSO" and "ridge regression" techniques of rewarding smaller coefficient values. All three of these designs are available in the R and Python “Glmnet suite”.

"Bayesian regression" models are useful when there is a lack of sufficient data or available data has poor distribution. These regression models are developed based on probability distributions rather than data points, meaning the resulting chart will appear as a bell curve depicting the variance with the most frequently occurring values in the center of the curve. The dependent variable "Y" in "Bayesian regression" is not valuation but a probability. Instead of predicting a value, we try to estimate the probability of an occurrence. This is regarded as "frequentist statistics", and this sort of statistics is built on the "Bayes theorem". “Frequentist statistics" hypothesize if an event is going to occur and the probability of it occurring again in the future.

"Conditional probability" is integral to the concept of “frequentist statistics”. Conditional probability pertains to the events whose results are dependent on one another. Events can also be conditional, which means the preceding event can potentially alter the probability of the next event. Assume you have a box of M&Ms and you want to understand the probability of withdrawing distinct colors of the M&Ms from the bag. If you have a set of 3 yellow M&Ms and 3 blue M&Ms, and on your first draw, you get a blue M&M, then with your next draw from the box, the probability of taking out a blue M&M will be lower than the first draw. This is a classic example of "conditional probability". On the other hand, an independent event is flipping of a coin, meaning the preceding coin flip doesn't alter the probability of the next flip of the coin. Therefore, a coin flip is not an example of "conditional probability".

Classification

The "classification algorithm" in machine learning and statistics can be defined as the algorithm used to define the set of categories (sub-populations) that the new input data can be grouped under, based on the training dataset which is composed of related data whose category has already been identified. For instance, all incoming emails can be grouped under the "spam" or "non-spam" category based on predefined rules. Similarly, a patient diagnosis can be categorized based on patient's observed attributes such as gender, blood group, prominent symptoms, family history for genetic diseases. Classification" can be considered as a type of pattern recognition technology. These individual hypotheses can be analyzed into a set of properties that can be easily quantified, referred to as "explanatory variables or features". These can be classified as "categorical", for example different types of blood groups like 'A+', 'O-'; or "ordinal" for example, different types of sizes like large, small; or "integer values", for example, number of times a specific word is repeated in a text; or "real values", for example, height and weight measurement. Certain classifiers operate by drawing a comparison with its prior observations using a "similarity or distance function".

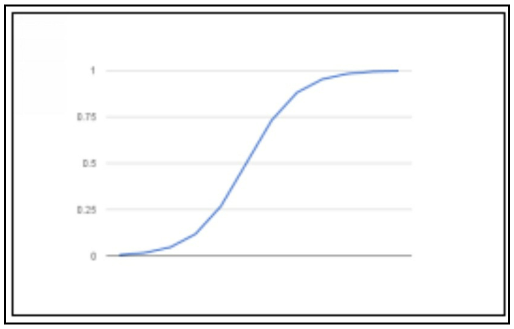
Any machine learning algorithm that is capable of implementing classification, particularly in the context of model implementation, is called "classifier". Very often the term "classifier" is used in the context of the mathematical function, which is being implemented by a "classification algorithm" and can map new input to the appropriate category. In the field of statistics, the classification of data is often carried out with "logistic regression", wherein the characteristics of the observations are referred to as "explanatory variables" or "independent variables" or "regressors" and the categories used to generate predictions are known as "outcomes". These "outcomes" are regarded as the probable values of the dependent variable. In the context of machine learning, "observations are often referred to as instances, the explanatory variables are referred to like features (grouped into a feature vector) and the possible categories to be predicted are referred to as classes".

The “Logistic regression” technique is “borrowed” by ML technology from the world of statistical analysis. "Logistic regression" is regarded to be the simplest algorithm for classification, even though the term sounds like a technique of "regression", that is not the case. "Logistic regression" produces estimates based on single or multiple input values for the likelihood of an event occurring. For example, a "logistic regression" will use a patient's symptoms, blood glucose level, and family history as inputs to generate the likelihood of the patient to develop diabetes. The model will generate a prediction in the form of probability ranging from '1' to '10' where '10' means full certainty. For the patient, if the projected probability exceeds 5, the prediction would be that they will suffer from diabetes. If the predicted probability is less than 5, it would be predicted that the patient will not develop diabetes. Logistic regression allows a line graph to be created which can represent the "decision boundary”.

It is widely used for binary classification tasks that involve two different class values. Logistic regression is so named owing to the fundamental statistical function at the root of this technique called the "logistic function". Statisticians created the "logistic function", also called the "sigmoid function", to define the attributes of population growth in ecosystems which continues to grow rapidly and nearing the maximum carrying capacity of the environment. The logistic function is “an S-shaped curve capable of taking any real-valued integer and mapping it to a value between '0' and '1', but never precisely at those boundaries, where ‘e’ is the base of the natural log (Euler's number or the EXP)” and the numerical value that you are actually going to transform is called the ‘value’.

“1 / (1 + e^-value)”

Here is a graph of figures ranging from “-5 and 5” which has been transformed by the logistic function into a range between 0 and 1.



Similar to the "linear regression" technique, "logistic regression" utilizes an equation for data representation.

Input values (X) are grouped linearly to forecast an output value (Y), with the use of weights or coefficient values (presented as the symbol "Beta"). It is mainly different from the “linear regression” because the modeled output value tends to be binary (0 or 1) instead of a range of values.

Below is an example of the "logistic regression" equation, where “the single input value coefficient (X) is represented by ‘b1’, the “intercept or bias term’ is the ‘bo’, and the “expected result” is ‘Y’. Every column in the input data set has a connected coefficient "b" with it, which should be understood by learning the training data set. The actual model representation, which is stored in a file or in the system memory would be “the coefficients in the equation (the beta values)”.

"y=e^(b0 + b1*x)/(1 +e^(b0 + b1*x))"

The "logistic regression" algorithm's coefficients (the beta values) must be estimated on the basis of the training data. This can be accomplished using another statistical technique called "maximum-likelihood estimation", which is a popular ML algorithm utilized using a multitude of other ML algorithms. "Maximum-likelihood estimation" works by making certain assumptions about the distribution of the input data set.

An ML model that can predict a value nearer to "0" for the “other class” and a value nearer to "1" for the “default class” can be obtained by employing the best coefficients of the model. The underlying assumption for most likelihood of the "logistic regression" technique is that “a search procedure attempts to find values for the coefficients that will reduce the error in the probabilities estimated by the model pertaining to the input data set (e.g. probability of ‘0’ if the input data is not the default class)”.

Without going into mathematical details, it is sufficient to state that you will be using “a minimization algorithm to optimize the values of the best coefficients from your training data set”. In practice, this can be achieved with the use of an effective “numerical optimization algorithm”, for example, the "Quasi-newton" technique).

Generating predictions using logistic regression

In here, you can simply plug in the measurements into the "logistic regression" equation and calculate the outcome to generate predictions with the "logistic regression" model. Let's take a look at an example to solidify this concept. Let's assume there is a model that is capable of generating predictions if an individual is masculine or woman depending on with fictitious values of their height. If the value of the height for an individual is set as 150 cm, would the individual be predicted as a male or female? Assuming we have already discovered the values of coefficients "b0= -100" and "b1= 0.6". By leveraging the above equation, the probability of male with a height of 150 cm or "P(male|height=150)" can be easily calculated. The function EXP() will be used for "e" because if you log this instance into your spreadsheet, this is what you can use:

“y = e^(b0 + b1*X) / (1 + e^(b0 + b1*X))”

“y = exp(-100 + 0.6*150) / (1 + EXP(-100 + 0.6*X))”

“y = 0.0000453978687”

Or a near "0" probability male is the gender of that specific person.

In theory, probability can simply be used. But since this is a "classification" algorithm and we want a sharp outcome, the probabilities can be tagged on to a binary class value. For instance, the model can predict "0" if "p (male) < 0.51" and predict "1" if "p (male) >= 0.5". Now that you know how predictions can be generated using "logistic regression", you can easily pre-process the training data set to get the most out of this technique. The assumptions made of the distribution and relations within the data set by the "logistic regression" technique are nearly identical to the assumptions made in the "linear regression" technique.

A lot of research has been done to define these hypotheses and to use accurate probabilistic and statistical language. It is recommended to use these as thumb rules or directives and try with various processes for data preparation.

The ultimate goal in "predictive modeling" machine learning initiatives is the generation of highly accurate predictions rather than analysis of the outcomes. Considering everything some assumptions could be broken if the designed model is stable and has high performance.

- **“Binary Output Variable":** This may be evident as we have already discussed it earlier, but "logistic regression" is designed specifically for issues with “binary (two-class) classification”. This will generate predictions for the probability of a default class instance that can be tagged into a classification of "0" or "1".
- **“Remove Noise”:** Logistic regression does not assume errors in the “output variable (‘y’)”, therefore, the “outliers and potentially misclassified” cases should be removed from the training data set.
- **“Gaussian Distribution”:** Logistic regression can be considered as a type of “linear algorithm but with a non-linear transform on the output”. A liner connection between the output and input variables is also assumed. Data transforms of the input variables may lead to a more accurate model with a higher capability of revealing the linear relationships of the data set. For instance, to better reveal these relationships, we could utilize "log", "root", "Box-Cox" and other single variable transformations.
- **“Remove Correlated Inputs”:** If you have various highly correlated inputs, the model could potentially be “over-fit” similar the "linear regression" technique. To address this issue, you can “calculate the pairwise correlations between all input data points and remove the highly correlated inputs”.

- **“Failure to converge”:** It is likely for the "expected likelihood estimation" method that is trained on the coefficients to fail to converge. It could occur if the data set contains several highly correlated inputs or there is a very limited data (e.g. loads of "0" in the input data).

“Naïve Bayes classifier algorithm” is another “classification” learning algorithm with a wide variety of applications. It is a method of classification derived from the "Bayes theorem", which assumes predictors are independent of one another. Simply put, a "Naïve Bayes classifier" assumes that “all the features in a class are unrelated to the existence of any other feature in that class”. For instance, if input data has an image of a fruit which is green, round, and about 10 inches in diameter, the model can consider the input to be a watermelon. Although these attributes rely on one another or the presence of a specific feature, all of the characteristics contribute independently to the probability that the image of the fruit is that of a watermelon, hence it is referred to as "Naive". “Naïve Bayes model” for large volumes of data sets is relatively simple to construct and extremely effective.

“Naïve Bayes” has reportedly outperformed even the most advanced techniques of classification, along with its simplicity of development. “Bayes theorem” can also provide the means to calculate the posterior probability "P(c|x)" using "P(c), P(x) and P(x)". On the basis of the equation shown in the picture below, where the probability of “c” can be calculated if “x” has already occurred.

“P(c|x)" is the posterior probability of "class (c, target)" provided by the "predictor (x, attributes)". “P(c)" is the class's previous probability. “P(x|c)” is the probability of the class provided by the predictor. “P(x)” is the predictor's prior probability.

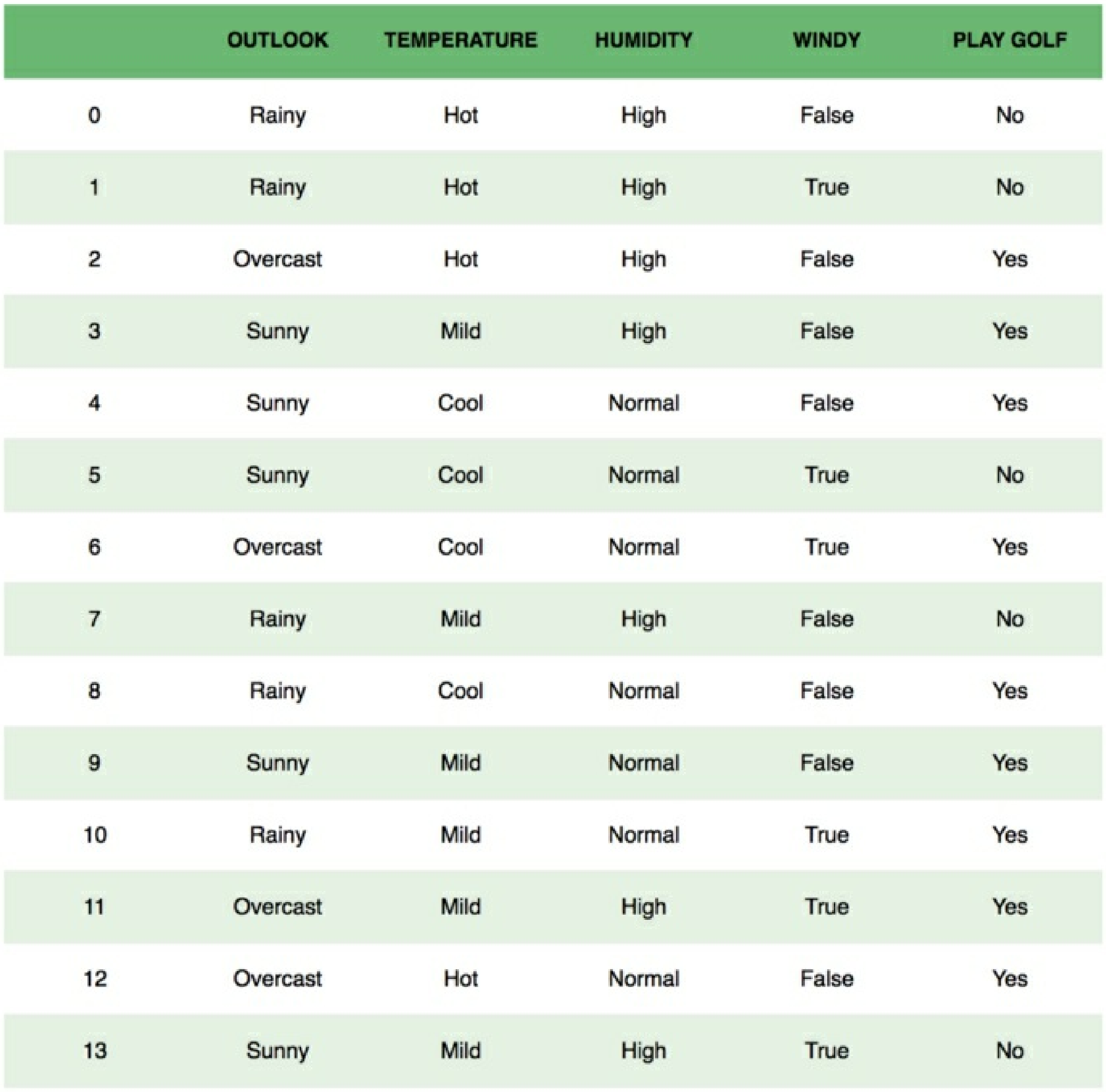
Likelihood   Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Posterior Probability   Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Here is an example to better explain the application of the "Bayes Theorem". The picture below represents the data set on the problem of identifying suitable weather days to play golf. The columns depict the weather features of the day and the rows contain individual entries. Considering the first row of the data set, it can be concluded that the weather will be too hot and humid with rain so the day is not suitable to play golf. Now, the primary assumption here is that all these features or predictors are independent of one another. The other assumption being made here is that all the predictors have potentially the same effect on the result. Meaning, if the day was windy it would have some relevance to the decision of playing golf as the rain. In this example, the variable (c) is the class (playing golf) representing the decision if the weather is suitable for golf and variable (x) represents the features or predictors.



	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

Types of "Naïve Bayes classifier"

- **“Multinomial Naïve Bayes”** - This is widely used to classify documents, for example, which category does a document belong: beauty, technology, politics, and so on. The frequency of the phrases in the document is considered as the features or predictors of the classifier.
- **“Bernoulli Naive Bayes”** - This is nearly identical to the “Multinomial Naïve Bayes”, however, the predictors used here are the "boolean variables". For example, depending on whether a

select phrase occurs in the text or not, the parameters used to predict the class variable can either be a yes or no value.

- **“Gaussian Naive Bayes”** - When the predictors are not distinct and have very similar or continuous values, it can be assumed that these values are obtained from a Gaussian distribution.

Applications of “Naïve Bayes”

- **“Real-Time Prediction”:** Naive Bayes is extremely quick in learning from the input data and can be seamlessly used to generate predictions in real-time.
- **“Multi-class Prediction”:** This algorithm is also widely used to generate predictions for multiple classes at the same time. It allows the prediction of the probability of various classes of the target variable.
- **“Text classification / Spam Filtering / Sentiment Analysis”:** The “Naive Bayes classifiers” is heavily used in text classification models owing to its ability to address problems with multiple classes of the target variable and the rule of autonomy. This algorithm has reported higher success rates than any other algorithm. As a consequence, it is commonly used in for identification of spam emails and sentiment analysis by identifying favorable and negative consumer feelings on the social media platforms.
- **“Recommendation System”:** "Naive Bayes Classifier" and "Collaborative Filtering" can be combined to generate a "Recommendation System" that utilizes ML and data mining methods to filter hidden data and generate insight as to whether the customer would prefer a particular item or product.

Chapter 3: Neural Network Learning Models

"Artificial Neural Networks" or (ANN) have been developed and designed to mimic the path of communication within the human brain. In the human body, billions of neurons are all interconnected and travel up through the spine and into the brain. They are attached to each other by root-like nodes that pass messages through each neuron one at a time all the way up the chain until it reaches the brain. These systems "learn" to execute jobs by looking at examples, normally without any of the task-specific rules being configured. For instance, they may learn to distinguish pictures that contain dogs using the image recognition technology, by evaluating sample pictures that were manually marked as "dog" or "no dog" and using the outcomes to locate dogs in other pictures. These systems can accomplish this even with no previous understanding of dogs like fur, tails, and dog-like faces. Rather, they are capable of producing identification features automatically from the samples that they are trained on.

An ANN functions as a collection of linked units or nodes called "artificial neurons", that resemble the biological neurons of the human brain. Each link can relay a signal to connected neurons, similar to the synapses in the human brain. An "artificial neuron" receiving a signal can then process it and subsequently transfer it to the connected neurons. When implementing the ANN, the "signal" at a connection will be a real number and the outcome of each neuron will be calculated using certain "non-linear function" of the sum of the inputs. The connections are known as "edges". Generally, the neurons and the "edges" are marked with a value or weight that will be optimized with learning. The weight will increase or decrease the strength of the signal received by the connected neuron. “Concepts” are formed and distributed through the sub-network of shared neurons. Neurons can be set with threshold limits so that a signal will be transmitted only if the accumulated signal exceeds the set threshold. Neurons are usually composed of several layers, which are capable of transforming their inputs uniquely. Signals are passed from the first layer called "input layer" to the final layer called "output layer", sometimes after the layers have been crossed several times.

The initial objective of the ANN model was to resolve problems as accomplished by a human brain. Over time, however, the focus has been directed towards performing select tasks, resulting in a shift from its initial objective. ANNs can be used for various tasks such as "computer vision, speech recognition, machine translation, social media filtering, playing boards, and video games, medical diagnostics, and even painting".

The most common ANN work on a unidirectional flow of information and are called “Feedforward ANN”. However, ANN is also capable of the bidirectional and cyclic flow of information to achieve state equilibrium. ANNs learn from past cases by adjusting the connected weights and rely on fewer prior assumptions. This learning could be supervised or non-supervised. With supervised learning, every input pattern will result in the correct ANN output. To reduce the error between the given output and the output generated by ANN, the weights can be varied. For example, reinforced learning, which is a form of “supervised learning”, informs the ANN if the generated output is correct instead of providing the correct output directly. On the other hand, unsupervised learning provides multiple input pattern to the ANN, and then the ANN itself explores the relationship between these patterns and learns to categorize them accordingly. ANNs with a combination of supervised and unsupervised learning are also available.

To solve data-heavy problems where the algorithm or rules are unknown or difficult to comprehend, ANNs are highly useful owing to their data structure and non-linear computations. ANNs are robust to multi-variable data errors and can easily process complex information in parallel. Though, the black-box model of ANN is a major disadvantage, which makes them unsuitable for problems that require a deep understanding and insight into the actual process.

Components of ANNs

- Neurons - ANNs maintained the biological notion of artificial neurons receiving input, combining it with
their internal state and threshold value if available, using an "activation function", and generating output using an "output function". Any form of data, including pictures and files, can be used as the initial inputs. The final results obtained can be recognition of an object in a picture. The significant feature of the "activation function" is that as the input values keep changing, it ensures a seamless transition, meaning, a minor change in input will result in a minor change in the output.

- Connections and weights - The ANN comprises of connections that utilize the output from one neuron as an input to an associated neuron. A "weight" that reflects the relative significance of the signal is allocated to each connection. There may be numerous input and output connections for a specific neuron.
- Propagation function - The "propagation function" can calculate the input to a neuron from the output of its predecessors and their connections, in the form of a "weighted sum". A "bias term" may be applied to the "propagation result". Backpropagation can be defined as a method for adjusting the weights of the connection to adjust for every error encountered, throughout the learning process. The quantity of error is simply split between the connections. Theoretically, "backprop" will calculate the gradient of the "cost function" connected with the weight of the given state. The weight can be updated through the "stochastic gradient descent" or other techniques, including "Extreme Learning Machines", "No-prop" network, "weightless" network, and "non-connectionist" neural network.

Hyperparameter of ANN

A "hyperparameter" can be defined as a parameter that is set before the actual start of the learning process. The parameter values will be obtained through the process of learning. For example, learning rate, batch size and number of concealed layers. Some "hyperparameter values" could potentially depend on other "hyperparameter values". The size of certain layers, for instance, may rely on the total number of layers. The "learning rate" for each observation indicates the size of the corrective measures that the model requires to compensate for any errors. A higher learning rate reduces the time needed to train the model but results in reduced accuracy. On the other hand, a lower rate of learning increases the time needed to train the model but can result in higher accuracy. Optimizations" like "Quickprop" are mainly directed at accelerating the minimization of errors, while other enhancements are primarily directed at increasing the reliability of the output. Refinements" utilize an "adaptive learning rate" that can increase or decrease as applicable, to avoid oscillation within the network, including the alternation of connection weights, and to help increase the convergence rate. The principle of momentum enables the weighing of the equilibrium between the gradient and the prior alteration so that the weight adjustment will depend on the prior alteration to a certain extent. The gradient is emphasized by the momentum close to "0", while the last change is emphasized by a value close to "1".

Neural Network Training with Data Pipeline

A neural network can be defined as “a function that learns the expected output for a given input from training datasets”. Unlike the “Artificial Neural Network”, the “Neural Network” features only a single neuron, also called as “perceptron”. It is a straightforward and fundamental mechanism which can be implemented with basic math. The primary distinction between traditional programming and a neural network is that computers running on neural network learn from the provided training data set to determine the parameters (weights and prejudice) on their own, without needing any human assistance. Algorithms like “backpropagation” and “gradient descent” may be used to train the parameters. It can be stated that the computer tries to increase or decrease every parameter a bit, in the hope that the optimal combination of Parameters can be found, to minimize the error compared with training data set.

Computer programmers will typically "define a pipeline for data as it flows through their machine learning model". Every stage of the pipeline utilizes the data generated from the previous stage once it has processed the data as needed. The word "pipeline" can be a little misleading as it indicates a unidirectional flow of data, when in reality the machine learning pipelines are "cyclical and iterative", as each stage would be repeated to eventually produce an effective algorithm.

While looking to develop a machine learning model, programmers work in select development environments geared for "Statistics" and 'Machine Learning" such as Python and R among others. These environments enable training and testing of the models, using a single "sandboxed" environment while writing reasonably fewer lines of code. This is excellent for the development of interactive prototypes that can be quickly launched in the market, instead of developing production systems with low latency.

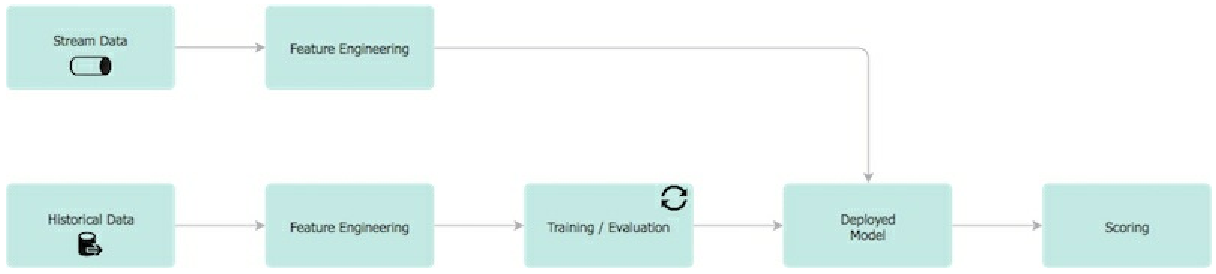
The primary goal of developing a machine learning pipeline is to construct a model with features listed below:

- Should allow for a reduction of system latency.

- Integration but loose coupling with other components of the model, such as data storage systems, reporting functionalities and "Graphical User Interface (GUI)".
- Should allow for horizontal as well as vertical scalability.
- Should be driven by messages, meaning the model should be able to communicate through the transfer of "asynchronous, non-blocking messages".
- Ability to generate effective calculations for management of the data set.
- Should be resilient to system errors and be able to recover with minimal to no supervision, known as breakdown management.
- Should be able to support "batch processing" as well as “real-time” processing of the input data.

Conventionally, data pipelines require "overnight batch processing", which mean gathering the data, transmitting it with an "enterprise message bus" and then processing it to generate pre-calculated outcomes and guidelines for future transactions. While this model has proven to work in certain industrial sectors, in others, and particularly when it comes to machine learning models, "batch processing" doesn't meet the challenge.

The picture below demonstrates a machine learning data pipeline as applied to a real-time business problem in which attributes and projections are dependent on time taken to generate the results. For instance, product recommendation systems used by Amazon, a system to estimate time of arrival used by Lyft, a system to recommend potential new links used by LinkedIn, search engines used by Airbnb, among others.



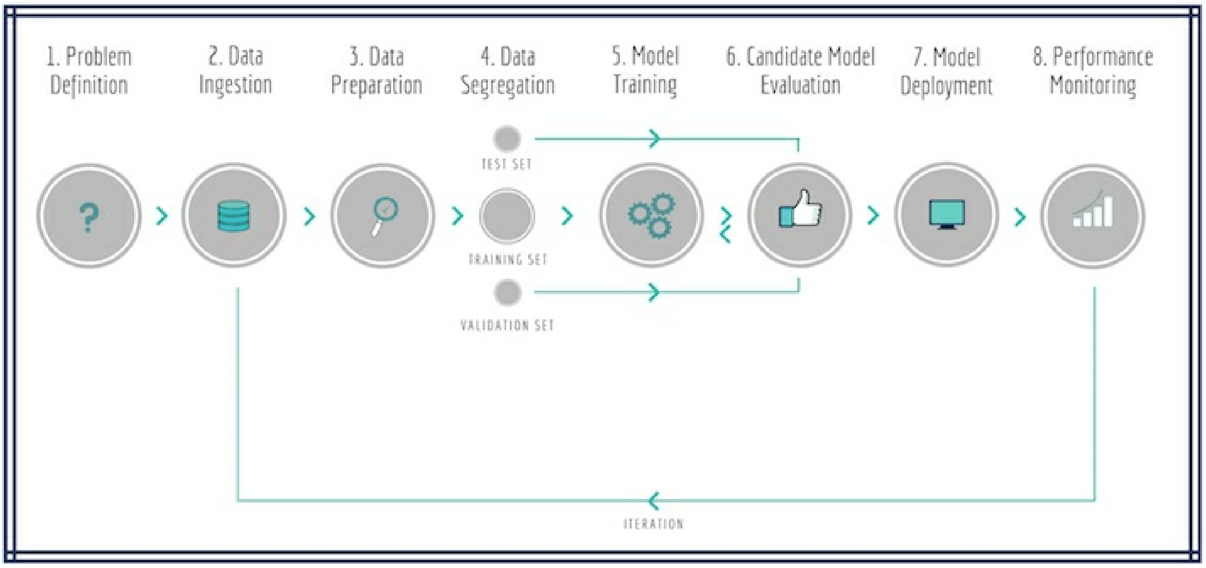
The swim lane diagram above consists of two explicitly specified components:

1. "Online Model Analytics": In the top swim lane of the picture, the elements of the application required for operation are

depicted. It shows where the model is used to make decisions in real-time.

2. “Offline Data Discovery": The bottom swim lane shows the learning element of the model, which is used to analyze historical data and generate the machine learning model using the "batch processing" method.

There are 8 fundamental stages in the creation of a data pipeline, which are shown in the picture below and explained in detail here:



Problem Definition

In this stage, the business problem that needs to be resolved using a machine learning model will be identified and documented with all pertinent details.

Data Ingestion

The first stage of any machine learning workflow is to channel input data into a database server. The most important thing to remember is that the data is ingested raw and with no modification to enable us to have an invariable record of the original dataset. Data may be supplied from a variety of sources which can be acquired either through request or transmitted from other systems.

“NoSQL document databases" are best suited to store huge amount of defined and labeled as well as unorganized raw data, that are quickly evolving as they do not need to adhere to a predefined scheme.
It even provides a "distributed, scalable and replicated data storage".

“Offline”

Data will flow in the "offline" layer to the raw data storage through an "Ingestion Service", which is a "composite orchestration service that is capable of encapsulating the data sourcing and persistence". A repository model is used internally to communicate with a data service that will interact with the data storage in exchange. When you save the data in the database, a unique batch Id will be given to the dataset, which allows for the effective query of the data as well as end-to-end tracking and monitoring of the data.

To be computationally efficient, the ingestion of the data is distributed into two folds.

- The first one is a specific pipeline for every dataset so that each of the datasets can be processed individually and simultaneously.
- The second aspect is that within each pipeline data can be divided to make the best of a variety of server cores, processors and perhaps even the entire server.

Distributing the prepping of data across several vertical and horizontal pipelines will reduce the total time required to perform the tasks.

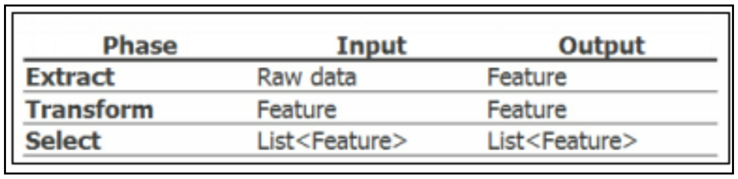
The "ingestion service" would run at regular intervals based on a predefined schedule (one or more times a day) or upon encountering a trigger. A subject will decouple producers (data source) from processors, which would be the data pipeline for this example, so when the source data is collected, the "producer system" will send a notification to the "broker" and subsequently the "embedded notification service" will respond by inducing ingestion of the data. The "notification service" would also inform the "broker" that the processing of the original dataset was completed with success and now the dataset is being stored in the database.

"Online"

The "Online Ingestion Service" makes up the entrance to the "streaming architecture" of the online layer, as it would decouple and manage the data flow from the source to the processing and storage components by offering consistent, high performance, low latency functionalities. It also works as an enterprise-level "Data Bus". Data would be stored on a long-term "Raw Data Storage", which also serves as a mediating layer to the subsequent online streaming service for further processing in real-time. For instance, such techniques that are utilized in this case may be "Apache Kafka (pub/sub messaging system)" and "Apache Flume (data collection to the long-term database)". A variety of other similar techniques are available and can be selectively applied based on the technology stack of the business.

Data Preparation

After the information has been ingested, a centralized pipeline would be produced that can evaluate the condition of the data, meaning it would search for format variations, outliers, patterns, inaccurate, incomplete or distorted information and correct any abnormalities through the process. The "feature engineering process" is also included in this stage. The 3 primary characters of a feature pipeline as depicted in the picture below are: "extraction, transformation, and selection".



Phase	Input	Output
Extract	Raw data	Feature
Transform	Feature	Feature
Select	List<Feature>	List<Feature>

Since this tends to be the most complicated component of any machine learning project, it is essential to introduce appropriate design patterns. In the context of coding, it implies the use of a factory technique to produce features based on certain shared abstract function behavior and a strategy pattern for selecting the correct features at the time of execution can be considered a logical approach. It is important to take into consideration the composition and re-usability of the pipeline while structuring the "feature extractors" and the "transformers".

The selection of functionalities could be attributed to the caller or could be automated. For instance, a "chi-square statistical test" can be applied to classify the impact of each function on the concept label, while discarding the low impact features before starting to train the model. To accomplish this, some "selector APIs" can be identified. In any case, a unique Id must be allocated to each feature set to make sure that the features used as model inputs and for impact scoring are consistent. Overall, it is necessary to assemble a data preparation pipeline into a set of unalterable transformations, which could be readily combined. Now the importance of "testing and high code coverage" will become a critical factor in the success of the model.

Data Segregation

The primary goal of the machine learning model is the development of a high accuracy model on the basis of the quality of its forecasts and predictions for information derived from the new input data, which was not part of the training dataset. Therefore, the available labeled dataset will be utilized as a "proxy" for future unknown input data by dividing the data into training and testing datasets. Many approaches are available to split the dataset and some of the most widely used techniques are:

- Using either the default or customized ratio to sequentially divide the dataset into two subsets to ensure that there is no overlap in the sequence in which the data appears from the source. For example, you could select the first 75% of data to train the model and the consequent 25% of data to test the accuracy of the model.
- Splitting the dataset into training and testing subset using a default or custom ratio with a random seed. For example, you could choose a random 75% of the dataset to train the model and the remaining 25% of the random dataset to test the model.
- Using either of these techniques ("sequential vs. random") and then also mixing the data within each data subset.
- Using a customized injected approach for splitting the data when extensive control over segregation of the data is required.

Technically the data segregation stage is not considered as an independent machine learning pipeline, however, an "API" or tool has to be provided to support this stage. In order to return the required datasets, the next 2 stages ("model training" and "model assessment") must be able to call this "API". As far as the organization of the code is concerned, a "strategy pattern" is required so that the "caller service" can select the appropriate algorithm during execution and the capability to inject the percentage or random seed is required. The "API" must also be prepared to return the information with or without labels, to train and test the model respectively. A warning can be created and passed along with the dataset to secure the "caller service" from defining parameters that could trigger uneven distribution of the data.

Model Training

The model pipelines are always "offline", and its schedule will vary from a matter of few hours to just one run per day, based entirely on the complexity of the application. The training can also be initiated by time and event, and not just by the system schedulers.

It includes many libraries of machine learning algorithms such as "linear regression, ARIMA, k-means, decision trees" and many more, which are designed to make provisions for rapid production of new model types as well as making the models interchangeable. Containment is also important for the integration of "third-Party APIs" using the "facade pattern" (at this stage the "Python Jupyter notebook" can also be called).

You have several choices for "parallelization":

- A specialized pipeline for individual models tends to be the easiest method, which means all the models can be operated at the same time.
- Another approach would be to duplicate the training dataset, i.e. the dataset can be divided and each data set will contain a replica of the model. This approach is favored for the models that require all fields of an instance for performing the computations, for example, "LDA", 'MF".
- Another approach can be to parallelize the entire model, meaning the model can be separated and every partition can be responsible for the maintenance of a fraction of the variables. This approach is best suited for linear machine learning models like "Linear Regression", "Support Vector Machine".
- Lastly, a hybrid strategy could also be utilized by leveraging a combination of one or more of the approaches mentioned above.

It is important to implement train the model while taking error tolerance into consideration. The data checkpoints and failures on training partitions must also be taken into account, for example, if every partition fails to owe to some transient problem like timeout, then every partition could be trained again.

Training approaches for Neural Network

Similar to most of the traditional machine learning models, Neural Networks can be trained using supervised and unsupervised learning algorithms as described below:

Supervised Training

Both inputs and outputs are supplied to the machine as part of the supervised training effort. Then the network will process the inputs and compare the outputs it generated to the expected outputs. Errors will then be propagated back through the model, resulting in the model adjusting the weights that regulate the network. This cycle is repeated time and again with the weights constantly changing. The data set enabling the learning is called the "training set". The same data set is processed several times while the weights of a relationship are constantly improved through the course of training of a network.

Current business network development packages supply resources for monitoring the convergence of an artificial neural network on its capacity to forecast the correct result. These resources enable the training routine to continue for days only until the model reaches the required statistical level or precision. Some networks, however, are incapable of learning. This could be due to the lack of concrete information in the input data from which the expected output is obtained. Networks will also fail to converge if sufficient quantity and quality of the data are not available to confer complete learning. In order to keep a portion of the data set for testing, a sufficient volume of the data set must be available. Most multi-node layered networks can memorize and store large volumes of data. In order to monitor the network to determine whether the system merely retains the training data in a manner that has no significance, supervised learning requires a set of data to be saved and used to evaluate the system once it has been trained.

To avoid insignificant memorization number of the processing elements sh
reduced. If a network can not simply resolve the issue, the developer needs to evaluate the inputs and outputs, the number of layers and its elements, the links between these layers, the data transfer and training functionalities, and even the original input weights. These modifications that are needed to develop an effective network comprise the approach in which the "art" of neural networking plays out. Several algorithms are required to provide the iterative feedback needed for weight adjustments, through the course of the training. The most popular technique used is "backward-error propagation", more frequently referred to as "back-propagation". To ensure that the network is not "overtrained", supervised training must incorporate an intuitive and deliberate analysis of the model. An artificial neural network is

initially configured with current statistical data trends. Subsequently, it needs to continue to learn other data aspects that could be erroneous from a general point of view. If the model is properly trained and no additional learning is required, weights may be "frozen", if needed. In some models, this completed network is converted into hardware to increase the processing speed of the model. Certain machines do not lock in but continue learning through its use in the production environment.

Unsupervised Training

The network is supplied only with inputs and not the expected results, in “unsupervised or adaptive” training. Then the model must determine its functionality for grouping the input data. This is often called as "self-organization or adaptation". Unsupervised learning is not very well understood at the moment. This adjustment to the surroundings is the pledge that will allow robots to learn continuously on their own as they come across new circumstances and unique settings. Real-world is full of situations wherein there are no training data sets available to resolve a problem. Some of these scenarios include military intervention, which could require new fighting techniques as well as arms and ammunition. Due to this unexpected element of existence and the human desire to be equipped to handle any situation, ongoing study and hope for this discipline continue. However, the vast majority of neural network job is currently carried out in models with supervised learning.

Teuvo Kohonen, an electrical engineer at "Helsinki University of Technology", is one of the pioneering researchers in the field of unsupervised training. He has built a self-organizing network, sometimes referred to as an "auto-associator", which is capable of learning without any knowledge of the expected outcome. It comprises of a single layer containing numerous connections in a network that looks unusual. Weights must be initialized for these connections and the inputs must be normalized. The neurons are organized in the "winner-take-all fashion". Kohonen is conducting ongoing research into networks that are designed differently from the conventional, feed-forward as well as back-propagation methods. Kohonen's research focuses on the organization of neurons into the network of the model.

Neurons within a domain are "topologically organized". Topology is defined as "a branch of mathematics that researches how to map from one space to another without altering its geometric configuration. An instance of a topological organization is the three-dimensional groupings that are common in mammalian brains. Kohonen noted that the absence of topology in designs of neural networks only makes these artificial neural networks a simple abstraction of the real neural networks found in the human brain. Advance self-learning networks could become feasible as this study progresses.

Candidate Model Evaluation

The model evaluation stage is also always "offline". By drawing a comparison of the predictions generated with the testing dataset with the actual data values using several key performance indicators and metrics, the "predictive performance" of a model can be measured. To generate a prediction on future input data, the "best" model from the testing subset will be preferred. An evaluator library consisting of some evaluators can be designed to generate accuracy metrics such as "ROC curve" or "PR curve", which can also be stored in a data storage against the model. Once more, the same techniques are applied to make it possible to flexibly combine and switch between evaluators.

The "Model Evaluation Service" will request the testing dataset from the "Data Segregation API" to orchestrate the training and testing of the model. Moreover, the corresponding evaluators will be applied for the model originating from the "Model Candidate repository". The findings of the test will be returned to and saved in the repository. In order to develop the final machine learning model, an incremental procedure, hyper-parameter optimization, as well as regularization methods, would be used. The best model would be deemed as deployable to the production environment and eventually released in the market. The deployment information will be published by the "notification service".

Model Deployment

The machine learning model with the highest performance will be marked for deployment for "offline (asynchronous)" and "online (synchronous)" prediction generation. It is recommended to deploy multiple models at the same time to ensure the transfer from obsolete to the current model is made smoothly, this implies that the services must continue to respond to forecast requirements without any lapse, while the new model is being deployed.

Historically, the biggest issue concerning deployment has been pertaining to the coding language required to operate the models have not been the same as the coding language used to build them. It is difficult to operationalize a "Python or R" based model in production using languages such as "C++, C #or Java". This also leads to a major reduction in the performance in terms of speed and accuracy, of the model being deployed. This problem can be dealt with in a few respects as listed below:

- Implementing new language for rewriting the code, for example, “Python to C# translate".
- Creating customized "DSL (Domain Specific Language)" to define the model.
- Creating a "micro-service" that is accessible through "RESTful APIs".
- Implementing an "API first" approach through the course of the deployment.
- Creating containers to store the code independently.
- Adding serial numbers to the model and loading them to "in-memory key-value storage".

In practice, the deployment activities required to implement an actual model are automated through the use of "continuous delivery implementation", which ensures the packaging of the necessary files, validation of the model via a robust test suite as well as the final deployment into a running container. An automated building pipeline can be used to execute the tests, which makes sure that the short, self-containing and stateless unit tests are conducted first. When the model has passed these tests, its quality will be evaluated in larger integrations and by executing regression tests. If both the test phases have been cleared, the model is deemed ready for deployment in the production environment.

Model Scoring

The terms "Model Scoring" and "Model Serving" are being used synonymously throughout the industry. Model Scoring" can be defined as "the process of generating new values with a given model and new input data". Instead of the term "prediction", a generic term "score" is being used to account for the distinct values it may lead to, as listed below:

- List of product recommendations.
- Numerical values, for the "time series models" as well as "regression models".
- A "probability value" that indicates the probability that a new input can be added to a category that already exists in the model.
- An alphabetical value indicating the name of a category that most closely resembles the new input data.
- A "predicted class or outcome" can also be listed as a model score particularly for "classification models".

After the models have been deployed they can be utilized to score based on the feature data supplied by the prior pipelines or provided directly by a "client service". The models must generate predictions with the same accuracy and high performance, in both the online and the offline mode.

“Offline”

The "scoring service" would be optimized in the offline layer for a big volume of data to achieve high performance and generate "fire and forget" predictions. A model can send an "asynchronous request" to initiate its scoring, however, it must wait for the batch scoring process to be completed and gain access to the batch scoring results before the scoring can be started. The "scoring service" will prepare the data, produce the features as well as retrieve additional features from the "Feature Data Store". The results of the scoring will be stored in the "Score Data Store", once the scoring has been completed. The "broker" will be informed of the completion of the scoring by receiving a notification from the service. This event is detected by the model, which will then proceed to collect the scoring results.

“Online”

In the "online" mode, a "client" will send a request to the "Online Scoring Service". The client can potentially request to invoke a specific version of the model, to allow the "Model Router" to inspect the request and subsequently transfer the request to the corresponding model. According to request and in the same way as the offline layer, the "client service" will also prepare the data, produces the features and if needed, fetch additional functions from the "Feature Data Store". When the scoring has been done, the scores will be stored in the "Score Data Store" and then returned via the network to the "client service".

Totally dependent on the use case, results
may be supplied asynchronously to the "client", which means the scored will be reported independent of the request using one of the two methods below:

- Push: After the scores have been obtained, they will be pushed to the "client" in the form of a "notification".
- Poll: After the scores have been produced, they will be saved in a "low read-latency database" and the client will poll the database at a regular interval to fetch any existing predictions.

There are a couple of techniques listed below, that can be used to reduce the time taken by the system to deliver the scores, once the request has been received:

- The input features can be saved in a "low-read latency in-memory data store".
- The predictions that have already been computed through an "offline batch-scoring" task can be cached for convenient access as dictated by the use-case, since "offline predictions" may lose their relevance.

Performance Monitoring

A very well-defined "performance monitoring solution" is necessary for every machine learning model. For the "model serving clients", some of the data points that you may want to observe include:

- "Model Identifier"
- "Deployment date and time"
- The "number of times" the model was served.
- The "average, min and max" of the time it took to serve the model.
- The "distribution of the features" that were utilized.
- The difference between the "predicted or expected results" and the "actual or observed results".

Throughout the model scoring process, this metadata can be computed and subsequently used to monitor the model performance.

Another "offline pipeline" is the "Performance Monitoring Service", which will be notified whenever a new prediction has been served and then proceed to evaluate the performance while persisting the scoring result and raising any pertinent notifications. The assessment will be carried out by drawing a comparison between the scoring results to the output created by the training set of the data pipeline. To implement fundamental performance monitoring of the model, a variety of methods can be used. Some of the widely used methods include "logging analytics" such as "Kibana", "Grafana" and "Splunk".

A low performing model that is not able to generate predictions at high speed will trigger the scoring results to be produced by the preceding model, to maintain the resiliency of the machine learning solution. A strategy of being incorrect rather than being late is applied, which implies that if the model requires an extended period to time for computing a specific feature then it will be replaced by a preceding model instead of blocking the prediction. Furthermore, the scoring results will be connected to the actual results as they are accessible. This implies continuously measuring the precision of the model and at the same time, any sign of deterioration in the speed of the execution can be handled by returning to the preceding model. In order to connect the distinct versions together, a "chain of responsibility pattern" could be utilized. The monitoring of the performance of the models is an on-going method, considering that a simple prediction modification can cause a model structure to be reorganized. Remember the advantages of machine learning model are defined by its ability to generate predictions and forecasts with high accuracy and speed to contribute to the success of the company.

Applications of Neural Network Models

Image processing and recognition of character: ANNs have an inherent ability to consume a variety of inputs, which can be processed to derive concealed as well as intricate, non-linear relationships, ANNs play a significant role in the recognition of images and characters. Character recognition such as handwriting adds diverse applicability in the identification of fraudulent monetary transactions and even matters concerning national security. Image recognition is a booming area with extensive applications ranging from facial recognition on the social media platforms like "Instagram" and "Facebook", as well as in medical sciences for detect cancer in patients to satellite image processing for environmental research and agricultural uses. Advancements in the field of ANN has now laid the foundation for "deep neural networks" that serve as the basis for "deep learning" technology.

A variety of groundbreaking and transformative developments in cutting edge technologies like computer vision, voice recognition, and natural language processing has been achieved. ANNs are effective and sophisticated models with a broad variety of utilizations. Some of the real-world applications are:

- “Speech Recognition Network developed by Asahi Chemical”; identification of dementia from the electrode-electroencephalogram study (the leading developer Anderer, noted that the ANN had higher identification accuracy than "Z statistics" and "discriminatory evaluation”).
- Generating predictions of potential myocardial infarction in the patient based on their electrocardiogram (ECG). In their research, Baxt and Skora revealed that the diagnostic sensitivity and specificity of the doctors in predicting myocardial infarction was 73.3% and 81.1% respectively, while the diagnostic sensitivity and specificity of the artificial neural network was 96.0% and 96.0% respectively.
- Software for recognition of cursive handwriting utilized by the "Longhand program" of "Lexicus2" executed on current notepads like "NEC Versapad" and "Toshiba Dynapad".
- Optical character recognition (OCR) used by fax machines like "FaxGrabber" developed by "Calera Recognition System" and "Anyfax OCR engine" licensed by "Caere Corporation" to other companies including well known "WinFax Pro" and “FaxMaster”.
- “Science Applications International Corporation or (SAIC)" developed the technology to detect bombs in luggage called “thermal neutron analysis (TNA)”, with the use of neural network algorithm.

Forecasting: In day-to-day enterprise decisions, for example, sales forecast, capital distribution between commodities, capacity utilization), economic and monetary policy, finance and inventory markets, forecasting is the tool of choice across the industrial spectrum. For instance, predicting inventory prices is a complicated issue with a host of underlying variables which can be concealed in the depths of big data or readily available. Traditional forecasting models tend to have various restrictions to take these complicated, non-linear associations into consideration. Due to its capacity to model and extract hidden characteristics and interactions, implementation of ANNs in the correct manner can provide a reliable solution to the problem at hand.

ANNs are also free of any restrictions on input and residual distributions, unlike the traditional forecast models. For instance, ongoing progress in this area has resulted in recent advancements in predictive use of "LSTM" and "Recurrent Neural Networks" to generate forecasts from the model. For example, forecasting the weather; foreign exchange systems used by “Citibank London” are driven by neural networks.

Chapter 4: Learning Through Uniform Convergence

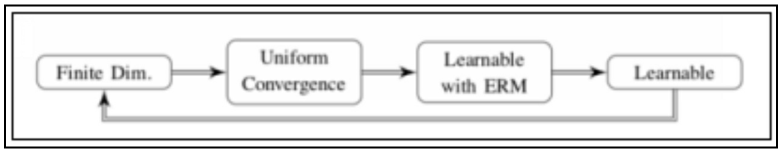
The most fundamental issue of statistical learning theory is the issue of characterizing the ability of the model to learn. In the case models driven by "supervised classification and regression" techniques, the learnability can be assumed to be equal to the "uniform convergence" of empirical risk to population risk. Meaning if a problem can be trained on, it can only be learned through minimization of empirical risk of the data. “Uniform convergence in probability", In the context of statistical asymptotic theory and probability theory, is a type of convergence in probability. It implies that within a particular event-family, the empirical frequencies of all occurrences converge to their theoretical probabilities under certain circumstances. “Uniform convergence in probability” as part of the statistical learning theory, is widely applicable to machine learning. Uniform convergence is defined as "a mode of convergence of features stronger than pointwise convergence, in the mathematical field of analysis".

In 1995, Vapnik, published the “General Setting of Learning”, as it pertains to the subject of statistical learnability. The General Learning Setting addresses issues of learning. Conventionally, a learning problem can be defined using a “hypothesis class ‘H’, an instance set ‘Z’ (with a sigma-algebra), and an objective function (e.g., loss or cost)” i.e. “$f: H \times Z \rightarrow R$”. This theory can be used “to minimize a population risk functional over some hypothesis class *H*, where the distribution *D* of *Z* is unknown, based on sample $z_1,...,z_m$ drawn from *D”*.

“$F(\mathbf{h}) = E_{Z\sim D}[f(\mathbf{h};Z)]$”

This General Setting encompasses “supervised classification and regression” techniques, some “unsupervised learning algorithms”, “density estimation”, among others. In supervised learning, *"$z = (x, y)$"* is an instance-label pair, *"**h**"* is a predictor, and *"$f(h; (x, y)) = loss(h(x), y)$"* is the loss function. In terms of statistical learnability, the goal is to minimize *“$F(\mathbf{h}) = E_{Z\sim D}[f(\mathbf{h};Z)]$”*, within experimental accuracy based on a finite sample only ($z_1,\ldots z_m$). The concern, in this case, does not pertain to the problem’s computational aspects, meaning whether this approximate minimization can be performed quickly and effectively, but whether this can be achieved statistically based on the sample ($z_1,\ldots z_m$) only.

It is common knowledge that a “supervised classification and regression” problem can only be learned when the empirical risks for the whole *“$\boldsymbol{h} \in H$”* uniformly connect to the population risk. According to the research done by Blumer (1989) and Alon (1997), “if uniform convergence holds, then the empirical risk minimizer (ERM) is *consistent*, that is, the population risk of the ERM converges to the optimal population risk, and the problem is learnable using the ERM”. This suggests that “uniform convergence” of the empirical risks is a required and satisfactory condition for learnability, which can be depicted as an equivocal to a “combinatorial condition” which, when it comes to classification algorithms, it has a finite “VC-dimension” and when in regression algorithms, it has a finite “fat-shattering dimension”. The picture below shows a scenario for “supervised classification and regression”:



Besides “uniform convergence”, “stability’s” specific concepts were proposed to be learnability’s prerequisite. Inherently, the concepts of “stability” rely on specific “learning laws” or algorithms, and evaluate how sensitive they are to the training data set’s fluctuations. It is recognized in specific that ERM stability would be enough for learnability. It is asserted in “Mukherjee et al. (2006)”, that stability is also essential to learning. On the basis of the assumption that “uniform convergence is equal to learnability, stability has been shown to characterize learning skill only where uniform convergence characterizes learning skill”.

Only in a "supervised classification and regression" environment, was the equivalence of uniform convergence and learning officially established. More broadly, the implications on the "right" in the picture above are valid: "finite fat-shattering dimensions", "uniform convergence", and even "ERM stability", can be deemed suitable for learnability using the “ERM”. Concerning the opposite implications, Vapnik has shown that a concept of "non-trivial" or "rigid" learnability associated with the ERM amounts to “uniform convergence of the empirical risks”. The concept was intended to banish some of the learning’s "trivial" issues that can be learned without uniform convergence. Even with these issues, empirical risk minimization can make learning feasible. Thus, in the “general learning setting” or “supervised classification and regression”, an issue would appear to be learnable only when it can be learned by empirical risk minimization.

This framework is not very specific and can sufficiently cover a significant part of the optimization and statistical learning problems that are widely popular, such as:

Stochastic Convex Optimization in Hilbert Spaces: “Let ‘*Z*’ be an arbitrary measurable set, let ‘*H*’ be a closed, convex, and bounded subset of a Hilbert space, and let ‘*f(h;z)*’ be Lipschitz-continuous and convex w.r.t. its first argument. Here, we want to approximately minimize the objective function ‘*Ez~D [f(h;z)]*’, where the distribution over ‘*Z*’ is unknown, based on an empirical sample $z_1,...,z_m$”.

Density Estimation: “Let ‘*Z*’ be a subset of ‘*Rn*’, let ‘*H*’ be a set of bounded probability densities on ‘*Z*’, and let ‘$f\,(h;\, z) = -\log(h(z))$’. Here, ‘$f\,(\cdot)$’ is simply the negative log-likelihood of an instance z according to the hypothesis density ‘*h*’. Note that to ensure bounded-ness of ‘$f\,(\cdot)$’, we need to assume that ‘*h(z)*’ is lower bounded by a positive constant for all ‘$z \in Z$’”.

K-Means Clustering in Euclidean Space: “Let $Z = R^n$, let ‘*H*’ be all subsets of Rn of size k, and let ‘$f\,(h;\, z) = \min_c \in h\ ||c - z||^{2}$’. Here, each *h* represents a set of ‘*k centroids*’, and ‘$f\,(\cdot)$’ measures the Euclidean distance squared between an instance z and its nearest centroid, according to the hypothesis h”.

Large Margin Classification in a Reproducing Kernel Hilbert Space (RKHS): “Let ‘$Z=X\times\{0,1\}$’, where ‘*X*’ is a bounded subset of an RKHS, let ‘*H*’ be another bounded subset of the RKHS, and let ‘$f\,(h;\, (x, y)) = \max\{0,\, 1 - y \langle x, h \rangle \}$’. Here, ‘$f\,(\cdot)$’ is the popular hinge loss function, and our goal is to perform margin-based linear classification in the RKHS”.

Regression: “Let ‘$Z = X \times Y$’ where ‘*X*’ and ‘*Y*’ are bounded subsets of ‘R^n’ and ‘*R*’ respectively, let ‘H’ be a set of bounded functions ‘$h : X^n \to R$’, and let ‘$f(h;(x,y)) = (h(x)-y)^2$’. Here, ‘$f(\cdot)$’ is simply the squared loss function”.

Binary Classification: “Let ‘$Z = X \times \{0,1\}$’, let ‘*H*’ be a set of functions ‘$h\colon X \to \{0,1\}$’, and let ‘$f\,(h;\, (x, y)) = 11_{\{h(x)' = y\}}$’. Here, ‘$f\,(\cdot)$’ is simply the ‘0 − 1 loss function’, measuring whether the binary hypothesis ‘*h(·)*’ misclassified the example *(x,y)*”.

This setting's ultimate goal is a selection of a hypothesis "$\boldsymbol{h} \in H$" based on a finite number of samples with the least amount of potential risk. In general, we are expecting that the sample size will improve the approximation of the risk. It is assumed that "learning guidelines that enable us to choose such hypotheses are *consistent*". In formal terms, we conclude that "rule **A"** is consistent with rate "*εcons(m)*" under distribution "*D*" if for all "*m*", where "$F^* = inf_{h \in H} F(h)$", the "rate" *ε(m)* is required to be monotone decreasing with "εcons(m) −→ 0)".

"ES~Dm [F(A(S)) − F∗] ≤ εcons(m)"

We can not choose a "D-based" learning rule because "D" is unknown. Rather, we need a "stronger requirement that the rule is consistent with *rate εcons(m)* under *all* distributions *D* over *Z*". The key definition is as follows:

"A learning problem is learnable, if there exist a learning rule **A** and a monotonically decreasing m→∞ sequence εcons(m), such that εcons(m) −→ 0, and ∀D, ES~Dm [F(**A**(S)) − F∗] ≤ εcons(m). A learning rule **A** for which this holds is denoted as a universally consistent learning rule."

The above definition of learnability, which requires a uniform rate of all distributions, is the most appropriate concept to study learnability of a hypothesis class. It is a direct generalization of "agnostic PAC-learnability" to "Vapnik's General Setting of Learning" as studied by Haussler in 1992. A potential path to learning is a minimization of the empirical risk "$F_S(\boldsymbol{h})$" over a sample "*S*", defined as

"FS (**h**) = 1/m ∑ f (**h** ; **z**)"

Z,z	"Instance domain and a specific instance."
H,h	"Hypothesis class and a specific hypothesis."
f(h,z)	"Loss of hypothesis h on instance z."
B	"suph,z | *f* (h; z)|"
D	"Underlying distribution on instance domain *Z"*
S	"Empirical sample z1,...,z*m*, sampled i.i.d. from *D"*
m	"Size of empirical sample *S"*
A(*S*)	"Learning rule A applied to empirical sample *S"*
εcons (*m*)	"Rate of consistency for a learning rule"

F (h)	“Risk of hypothesis h, E$z\sim D$ [f (h; z)]”
$F*$	“inf$h\in H$ F(h)”
FS (h)	“Empirical risk of hypothesis h on sample S, 1 $\sum z\in S$ f (h; z) m”
hˆ S	“An ERM hypothesis, FS(hˆS) = inf$h\in H$ FS(h)
εerm (m)	Rate of AERM for a learning rule”
εstable (m)	“Rate of stability for a learning rule”
εgen (m)	“Rate of generalization for a learning rule”

The “rule ***A”*** is an *“Empirical Risk Minimizer”* if it can minimize the empirical risk

“$F_S(\mathbf{A}(S)) = F_S(\hat{\mathbf{h}}_S) = \inf_{\mathbf{h}\in H} F_S(\mathbf{h})$”

where “*FS(**h**ˆS) = inf**h**∈H FS(**h**)”* is referred to as the “minimal empirical risk”. Given the odds that multiple hypotheses minimize the empirical risk, “$\hat{\mathbf{h}}_S$” does not pertain to a certain hypothesis and there could potentially be multiple rules which are all “ERM”.

“Rule ***A”*** can, therefore, be concluded to be an *“AERM (Asymptotic Empirical Risk Minimizer)* with *rate* $\varepsilon_{erm}(m)$ under distribution *D”*, when:

“$E_{S\sim D^m}[F_S(\mathbf{A}(S))-F_S(\hat{\mathbf{h}}_S)]\leq \varepsilon_{erm}(m)$”

A learning rule can be considered *an “AERM universally”* with *“rate* $\varepsilon_{erm}(m)$*”* if it is an AERM with *“rate* $\varepsilon_{erm}(m)$*”* under all distributions *“D”* over *“Z”*. A learning rule can be considered *“always AERM”* with “rate $\varepsilon_{erm}(m)$”, if for *any “S”* sample of *“m”*, size it holds that *“*$F_S(\mathbf{A}(S))-F_S(\hat{\mathbf{h}}_S) \leq \varepsilon_{erm}(m)$”.

It can be concluded that “rule ***A***” *generalizes* with rate “$\varepsilon_{gen}(m)$” under distribution *D* if for all *m*, where ***A*** rule *“universally generalizes* with rate $\varepsilon_{gen}(m)$ if it generalizes with rate $\varepsilon_{gen}(m)$ under all distributions *D* over *Z”*.

“$E_{S\sim D^m}[|F(\mathbf{A}(S)) - F_S(\mathbf{A}(S))|] \leq \varepsilon_{gen}(m)$”

Impact of Uniform Convergence on Learnability

Uniform convergence is considered applicable to learning problems, “if the empirical risks of hypotheses in the hypothesis class converge to their population risk uniformly, with a distribution-independent rate”:

“$\sup_D E_{S\sim D^m}[\sup_{\mathbf{h}\in H} |F(\mathbf{h})-F_S(\mathbf{h})|] - m\to\infty \rightarrow 0$”

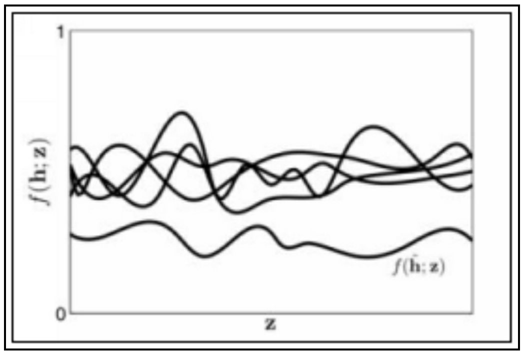
It is easy to demonstrate that an issue can be deemed learnable using the "ERM learning rule" if uniform convergence holds.

In 1971, Chervonenkis and Vapnik, demonstrated that “the finiteness of a straightforward combinatorial measure known as the VC dimension indicates uniform convergence, for binary classification issues (where $Z = X \times \{0, 1\}$, each hypothesis is a mapping from X to $\{0, 1\}$, and $f(\mathbf{h}; (\mathbf{x}, y)) = 1_{\{\mathbf{h}(\mathbf{x})\neq y\}}$)”. Also, it can be confirmed that in a distribution-independent sense, problems regarding binary classification with infinite “VC-dimension” can be *not* learned. As a necessary and sufficient condition for learning, this identifies the situation of having finite “VC-dimension”, and therefore, uniform convergence.

This characterization is extensible to “regression” techniques as well, namely “regression with squared loss, where **h** is now a real-valued function, and $f(\mathbf{h}; (\mathbf{x}, y)) = (\mathbf{h}(\mathbf{x}) - y)^2$”. The property of having a “finite fat-shattering dimension” on all finite scales can substitute for the property of containing “finite VC dimensions”, but the basic equivalence still contains, however, a problem can be learned only if there is a uniform convergence. These findings are typically based on sensible reductions made to binary classification. Even though, the “General Learning Setting” observed is not as specific as the classification and regression, including scenarios where it is difficult to reduce the classification to binary classification.

In 1998, Vapnik sought to depict that “in the General Learning Setting, learnability with the ERM learning rule is equivalent to uniform convergence”, to bolster the need of uniform convergence in this setting while noting that the result may not hold true to “trivial” situations. Specifically, cases pertaining to “arbitrary learning problem with hypothesis class H and adding H to a single hypothesis $\tilde{\boldsymbol{h}}$ such that $f(\tilde{\boldsymbol{h}}; \boldsymbol{z}) < \inf_{\boldsymbol{h}\in H} f(\boldsymbol{h}, \boldsymbol{z})$ for all $\mathbf{z} \in Z$”, as shown in the picture below. This particular problem of learning can be “trivially” learned using the “ERM learning rule” which always chooses “$\tilde{\boldsymbol{h}}$”. Although, “H” can be an arbitrary complex with no prior assumptions and uniform convergence. It must be noted that this is not applicable to binary classification models, where “$f(h; (x, y)) = 1 1_{\{h(x)\neq y\}}$”, since on any “$(x,y)$” there will be hypotheses with “$f(h;(\mathbf{x},y)) = f(\tilde{h};(\mathbf{x},y))$”

and therefore, if “*H*” is highly complex with infinite “VC dimensions then multiple hypotheses will have “0” empirical error on any given training data set.



In order to remove such “trivial” scenario, the concept of “strict consistency” was proposed by Vapnik, as an even stronger version of consistency. It is defined with the equation below, where the convergence lies within the probability.

“$\forall c \in \mathrm{R}, \inf_{\mathbf{h}:F(\mathbf{h})\geq c} F_S(\mathbf{h}) - m \rightarrow \infty \rightarrow \inf_{\mathbf{h}:F(\mathbf{h})\geq c} F(\mathbf{h})$”

The thought is that the empirical risk of “ERM” is essential for convergence of the smallest potential risk, even once the “good” hypotheses with less risk than the threshold have been removed. Vapnik succeeded in proving that such “strict consistency” of the ERM has real equivalence to uniform convergence, of the form in probability. The equivalence below is true for every individual distribution and is independent of the universal consistency of the “Empirical Risk Minimizer”.

“$\sup_{\mathbf{h}\in \mathrm{H}} (\mathrm{F} (\mathbf{h}) - \mathrm{F}_{\mathrm{S}} (\mathbf{h})) - \mathrm{m} \rightarrow \infty \rightarrow 0$”

Based on this research study, it can be implied that “up to trivial situations, a uniform convergence property indeed characterizes learnability, at least using the ERM learning rule”.

Learnability without Uniform Convergence

A “stochastic convex optimization” or learnability without uniform convergence problem can be considered an exceptional case of the “General Learning Setting” explained above, including additional limitations that “the objective function *f(**h**;**z**)* is Lipschitz-continuous and convex in ***h*** for every ***z***, and that *H* is closed, convex and bounded”. The problems where “*H*” is a subset of a “Hilbert space” will be addressed here. An exceptional scenario is “the familiar linear prediction setting, where ***z*** = *(**x**, y)* is an instance-label pair, each hypothesis **h** belongs to a subset *H* of a Hilbert space, and f (**h**; **x**, y) = l(⟨ **h**, φ(**x**) ⟩ , y) for some feature mapping *φ* and a loss function *l : R* × *Y* → *R*, which is convex with respect to its first argument".

The scenario has been successfully established where the stochastic dependence on "**h"** is linear, similar to the previous example, has been established successfully. When the "domain *H"* and the "mapping φ" is bounded, there is uniform convergence, meaning that "$|F(\mathbf{h}) - F_S(\mathbf{h})|$" is uniformly bounded overall "**h** ∈ *H"*. This uniform convergence of "$F_S(\mathbf{h})$ to $F(\mathbf{h})$" validates selection of the empirical minimizer "$\hat{\mathbf{h}}S$ =arg $\min_{\mathbf{h}} F_S(\mathbf{h})$", and ensures that the expected value of "$F(\hat{\mathbf{h}}_S)$" converges to the optimal value "$F^* = \inf_{\mathbf{h}} F(\mathbf{h})$".

Although dependency on "h" is nonlinear, uniform convergence can still be established with the use of "covering number arguments" provided "H" is a finite dimension. Regrettably, uniform convergence may not take place if we go to infinite-dimensional hypothesis and empirical minimization may not make impart the ability to learn to the algorithm. Remarkably, this does not mean that the problem can be deemed "unlearnable". It can be shown that with regularization mechanisms, even when uniform convergence doesn't exist, a learning algorithm can be developed to solve any "stochastic convex optimization" issue. This mechanism directly relates to the principle of stability. For example, let's look at the "convex stochastic optimization" problem given by the equation in the picture below, where for this example "*H"* will be the "*d*-dimensional unit sphere *H* = **h**∈R*d* : | **h**| ≤1", "**z**=(**x**,α) with α ∈ [0, 1]*d"* and "**x** ∈ *H"* , and *"u* ∗ *v"* can be defined as an element-wise product.

$$\boxed{f^{(3)}(\mathbf{h};(\mathbf{x},\alpha)) \;=\; \|\alpha * (\mathbf{h}-\mathbf{x})\| \;=\; \sqrt{\sum_i \alpha^2[i](\mathbf{h}[i]-\mathbf{x}[i])^2}}$$

Now, considering a series of learning problems, where "$d = 2^m$" for any sample size "*m"*, and establishing that a "convergence rate independent of the dimensionality '*d'* cannot be expected". This case can be formalized into infinite dimensions. The learning problem in the equation above can be considered as "that of finding the center of an unknown distribution over ***x*** ∈ *Rd*, where stochastic per-coordinate confidence measures "*α[i]*" are also available”. For now, we will be focusing on the scenario wherein certain coordinates are missing, meaning “*$\alpha[i] = 0$*”.

By taking into consideration the distribution given below over “*($\mathbf{x}$, α): $\mathbf{x}$ = 0*” with probability as 1, and “*α*” is uniform over “*$\{0, 1\}^d$*”. That is, “*$\alpha[i]$*” are independent and identically distributed uniform “Bernoulli”. For a random sample “($\mathbf{x_1}$, α_1),...,($\mathbf{x_m}$, α_m)” if “*$d > 2^m$*” then that is a result of probability greater than “*$1 - e^{-1} > 0.63$*” and a coordinate “*$j \in 1 \ldots d$*” is present such that all “confidence vectors α_i” in the sample are “0” on the coordinate “*j*”, i.e. “*$\alpha i[j] = 0$*” for all “*$i = 1..m$*”. Assume “*$\mathbf{e}j \in H$*” is the “standard basis vector corresponding to this coordinate”. Then in the equation shown in the picture below, “$F_S^{(3)}(\cdot)$" represents the empirical risk concerning the function “$f^{(3)}(\cdot)$”.

$$F_S^{(3)}(\mathbf{e}_j) = \frac{1}{m}\sum_{i=1}^{m}\|\alpha_i * (\mathbf{e}_j - 0)\| = \frac{1}{m}\sum_{i=1}^{m}|\alpha_i[j]| = 0$$

In another scenario, if “$F_S^{(3)}(\cdot)$” denotes the actual risk for the function “$f^{(3)}(\cdot)$”, the equation shown in the picture below is obtained.

$$F^{(3)}(\mathbf{e}_j) = \mathbb{E}_{\mathbf{x},\alpha}\left[\|\alpha * (\mathbf{e}_j - 0)\|\right] = \mathbb{E}_{\mathbf{x},\alpha}\left[|\alpha[j]|\right] = 1/2$$

Thus, for any sample size “*m*”, a convex “Lipschitz-continuous objective” can be constructed in a dimension that is high enough so as to ensure that with minimum “0.63 probability” over the sample, “*$\sup_{\mathbf{h}} |F^{(3)}(\mathbf{h}) - F^{(3)}(\mathbf{h})| \geq \frac{1}{2}$*”. In addition, since “*$f(\cdot; \cdot)$*” is non-negative, “$\mathbf{e}_j$” can be denoted as an “empirical minimizer”, even though its expected value “$F^{(3)}(\mathbf{e}_j) = \frac{1}{2}$” is not at all close to the optimal expected value “$\min_{\mathbf{h}} F^{(3)}(\mathbf{h}) = F^{(3)}(0) = 0$”.

To explain this case with an approach that is not dependent on the sample-size, assume “*H* is the unit sphere of an infinite-dimensional Hilbert space with orthonormal basis **e**1, **e**2,..., where for $\mathbf{v} \in H$, we refer to its coordinates $\mathbf{v}[j] = \langle \mathbf{v}, \mathbf{e}_j \rangle$” with respect to this basis”. The “confidences α” serve as a map of every single coordinate to “[0, 1]”. This means, an “infinite sequence of reals in [0, 1]". The operation of the product according to the elements, "$\alpha * \boldsymbol{v}$" is defined on the basis of this mapping and the objective function "$f^{(3)}(\cdot)$" of the equation (shown in the first picture of this example) can be easily defined in this infinite-dimensional space.

Let us now reconsider the distribution over "$z = (\mathbf{x}, \alpha)$" where "$\mathbf{x} = 0$" and "α" is an infinite independent and identically distributed sequence of "uniform Bernoulli random variables" (that is, a "Bernoulli process with each α_i uniform over {0, 1} and independent of all other α_j"). It can be implied that for any finite sample there is high likelihood of finding a coordinate "j" with "$\alpha_i [j] = 0$" for all "I", and therefore, an empirical minimizer "$F_S^{(3)} (\mathbf{e}_j) = 0$" with "$F^{(3)} (\mathbf{e}_j) = 1/2 > 0 = F^{(3)} (0)$" can be obtained.

Consequently, it can be observed that the empirical values "$F_S^{(3)} (\boldsymbol{h})$" are not uniform while converging as expected, and empirical minimization does not guarantee a solution to the learning problem. Furthermore, one could potentially generate a sharper counter-example, wherein the "*unique* empirical minimizer $\hat{\mathbf{h}}_S$" is nowhere close to the optimal expected value. In order to accomplish this, "$f^{(3)} (\cdot)$" must be augmented with the use of "a small term which ensures its empirical minimizer is unique, and not too close to the origin". Considering the equation below where "$\varepsilon = 0.01$".

"$f^{(4)} (\mathbf{h};(\mathbf{x},\alpha)) = f^{(3)} (\mathbf{h};(\mathbf{x},\alpha))+\varepsilon\sum 2-i(\mathbf{h}[i]-1)^2$"

The objective continues to be convex and "$(1 + \varepsilon)$" is still "Lipschitz". In addition, since the added term is strictly convex, the "$f^{(4)} (\boldsymbol{h};\boldsymbol{z})$" will also be strictly convex with respect to "$\boldsymbol{h}$" and that is the reason for the empirical minimizer being unique.

Considering the same distribution over "z: $\boldsymbol{x} = 0$" while "$\alpha[i]$" are independent and identically distributed uniform 0 or 1. The minimizer of "$F_S^{(4)} (\boldsymbol{h})$" is referred to as the empirical minimizer which is subjected to the constraints "$|\boldsymbol{h}| \leq 1$". The good news is that although the identification of the solution for such a constrained optimization problem is complicated, it is not mandatory. It is sufficient to depict that "the optimum of the *unconstrained* optimization problem $h^*_{UC} = arg\ min F_S^{(4)} (\boldsymbol{h})$ (with no constraining $\boldsymbol{h} \in H$) has norm $|\boldsymbol{h}^*_{UC}| \geq 1$".

It should be noted that “in the unconstrained problem, wherein $\alpha_i[j] = 0$ for all $i = 1...n$, only the second term of $f^{(4)}$ depends on $\mathbf{h}[j]$ and we have $\boldsymbol{h}^{*}_{\mathrm{UC}}[j] = 1$”. As it could happen for certain coordinate “*j*”, it can be concluded that “the solution to the constrained optimization problem lies on the boundary of *H* , that is $|\boldsymbol{h}\hat{}\ S| = 1$”, which can be represented by the equation shown in the picture below while “$F^{*} \leq F(0) = \varepsilon$”.

$$F^{(4)}(\hat{\mathbf{h}}_S) \geq \mathbb{E}_\alpha\left[\sqrt{\sum_i \alpha[i]\hat{\mathbf{h}}_S^2[i]}\right] \geq \mathbb{E}_\alpha\left[\sum_i \alpha[i]\hat{\mathbf{h}}_S^2[i]\right] = \sum_i \hat{\mathbf{h}}_S^2[i]\mathbb{E}_\alpha[\alpha[i]] = \frac{1}{2}\left\|\hat{\mathbf{h}}_S\right\|^2 = \frac{1}{2}$$

Chapter 5: Data Science Lifecycle and Technologies

The earliest recorded use of the term data science goes back to 1960 and credited to “Peter Naur”, who reportedly used the term data science as a substitute for computer science and eventually introduced the term “datalogy”. In 1974, Naur released his book titled “Concise Survey of Computer Methods”, with liberal use of the term data science throughout the book. In 1992, the contemporary definition of data science was proposed at “The Second Japanese-French Statistics Symposium”, with the acknowledgment of emergence of a new discipline focused primarily on types, dimensions, and structures of data.

The term Data can be defined as “information that is processed and stored by a computer”. Our digital world has flooded our realities with data. From a click on a website to our smartphones tracking and recording our location every second of the day, our world is drowning in the data. From the depth of this humongous data, solutions to our problems that we have not even encountered yet could be extracted. This particular process of gathering insights from a measurable set of data using mathematical equations and statistics can be defined as “data science”. The role of data scientists tends to be very versatile and is often confused with a computer scientist and a statistician. Essentially, anyone, be it a person or a company, that is willing to dig deep to large volumes of data to gather information can be referred to us data science practitioner. For example, companies like “Amazon” and “Target” keeps a track on and record of in-store and online purchases made by the customers, to provide personalized recommendations on products and services. The social media platforms like “Twitter” and “Instagram”, that allow the users to list their current location, is capable of identifying global migration patterns by analyzing the wealth of data that is handed to them by the users themselves.

Data science lifecycle

The most highly recommended lifecycle for structured data science projects is the "Team Data Science Process" (TDSP). This process is widely used for projects that require the deployment of applications based on artificial intelligence and/or machine learning algorithms. It can also be customized for and used in the execution of "exploratory data science" projects as well as "ad hoc analytics" projects. The TDSP lifecycle is designed as an agile and sequential iteration of steps that serve as guidance on the tasks required for the use of predictive models. These predictive models need to be deployed in the production environment of the company, so they can be used in the development of artificial intelligence base applications. The aim of this data science lifecycle is high-speed delivery and completion of data science project toward a defined engagement endpoint. Seamless execution of any data science project requires effective communication of tasks within the team as well as to the stakeholders.

The fundamental components of the “Team Data Science Process” are:

Definition of a data science lifecycle

The five major stages of the TDSP lifecycle that outline the interactive steps required for project execution from start to finish are: “Business understanding”, “Data acquisition in understanding”, “modeling”, “deployment” and “customer acceptance”. Keep reading for details on this to come shortly!

Standardized project structure

To enable seamless and easy access to project documents for the team members allowing for quick retrieval of information, use of templates and a shared directory structure goes a long way. All project documents and the project code our store and a “version control system” such as “TFS”, “Git" or "Subversion" for improved team collaboration. Business requirements and associated tasks and functionalities are stored in an agile project tracking system like "JIRA", "Rally" and "Azure DevOps" to enable enhanced tracking of code for every single functionality. These tools also help in estimation of resources and costs involved through the project lifecycle. To ensure effective management of each project, information security, and team collaboration, TDSP confers creation of separate storage for each project on the version control system. The adoption of standardized structure for all the projects within an organization, aid in the creation of institutional knowledge library across the organization.

The TDSP lifecycle provides standard templates for all the required documents as well as folder structure at a centralized location. The files containing programming codes for the data exploration and extraction of the functionality can be organized to using the provided a folder structure, which also holds records of model iterations. These templates allow the team members to easily understand the work that has been completed by others as well as for seamless addition of new team members to a given project. The markdown format supports ease of accessibility as well as making edits or updates to the document templates. To make sure the project goal and objectives are well defined and also to ensure the expected quality of the deliverables, these templates provide various checklists with important questions for each project. For example, a "project charter" can be used to document the project scope and the business problem that is being resolved by the project; standardized data reports are used to document the "structure and statistics" of the data.

Infrastructure and resources for data science projects

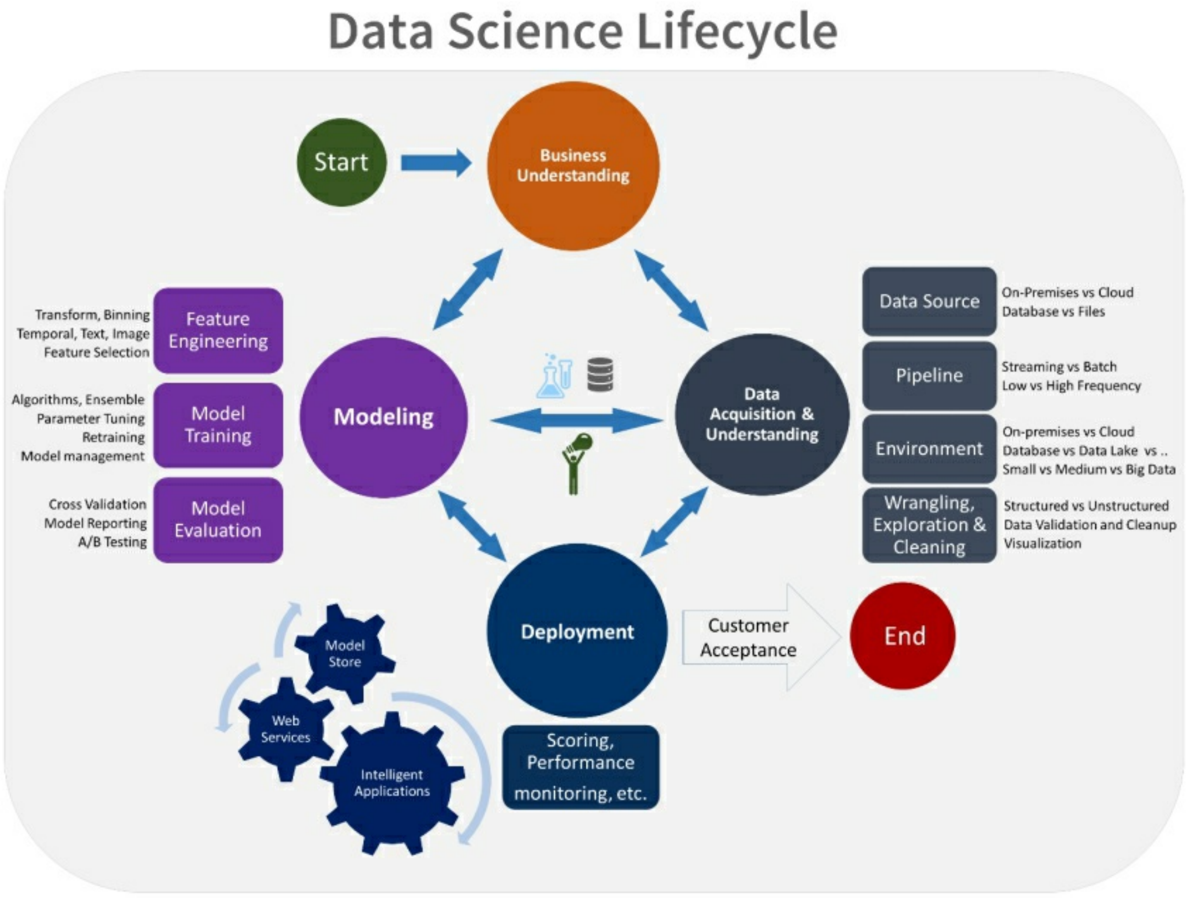
To effectively store infrastructure and manage shared analytics, the TDSP recommends using tools like: "machine learning service", databases, "big data clusters" and cloud-based systems to store data sets. The analytics and storage infrastructure that houses raw as well as processed or cleaned data sets can be cloud-based or on-premises. D analytics and storage infrastructure permits the reproducibility of analysis and prevents duplication and the redundancy of data that can create inconsistency and unwarranted infrastructure costs. Tools are supplied to grant specific permissions to the shared resources and to track their activity which in turn allows secure access to the resources for each member of the team.

Tools and utilities for project execution

Introduction of any changes to an existing process tends to be rather challenging in most organizations. To encourage and raise the consistency of adoption of these changes several tools can be implemented that are provided by the TDSP. Some of the basic tasks in the data science lifecycle including "data exploration" and "baseline modeling" can be easily automated with the tools provided by TDSP. To allow the hassle-free contribution of shared tools and utilities into the team's "shared code repository", TDSP from provides a well-defined structure. This results in cost savings by allowing other project teams within the organization to reuse and repurpose these shared tools and utilities.

The TDSP lifecycle serves as a standardized template with a well-defined set of artifacts that can be used to garner effective team collaboration and communication across the board. This lifecycle is comprised of a selection of the best practices and structures from “Microsoft” to facilitated successful delivery predictive analytics Solutions and intelligent applications.

Let’s look at the details of each of the five stages of the TDSP lifecycle, namely, “Business understanding”, “Data acquisition in understanding”, “modeling”, “deployment” and “customer acceptance”.



Stage I – Business understanding

The goal of this stage is to gather and drill down on the essential variables that will be used as targets for the model and the metrics associated with these variables will ultimately determine the overall success of the project. Another significant objective of this stage is the identification of required data sources that the company already has or may need to procure. At this stage, the two primary tasks that are required to be accomplished are: "defining objects and identifying data sources".

Deliverables to be created in this stage

- **Charter document** – It is a "living document" that needs to be updated throughout the course of the project, in light of new project discoveries and changing business requirements. A standard template is supplied with the TDSP "project structure definition". It is important to build upon this document by adding more details throughout the course of the project while keeping the stakeholders promptly updated on all changes made.

- **Data sources** – Within the TDSP "project data report folder", the data sources can be found within the "Raw Data Sources" section of the "Data Definitions Report". The "Raw Data Sources" section also specifies the initial and final locations of the raw data and provide additional details like the "coding scripts" to move up the data to any desired environment.
- **Data dictionaries** – The descriptions of the characteristics and features of the data such as the "data schematics" and available "entity-relationship diagrams", provided by the stakeholders are documented within the Data dictionaries.

Stage II – Data acquisition and understanding

The goal of this stage is the production of high quality processed data set with defined relationships to the model targets and location of the data set in the required analytics environment. At this stage "solution architecture" of the data pipeline must also be developed which will allow regular updates to and scoring of the data. The three primary tasks that must be completed during this stage are: "Data ingestion, Data exploration and Data pipeline set up".

Data ingestion

The process required to transfer the data from the source location to the target location should be set up in this phase. The target locations are determined by the environments that will allow you to perform analytical activities like training and predictions.

Data exploration

The data set must be scrubbed to remove any discrepancies and errors before it can be used to train the Data models. To check the data quality and gathered information required to process the data before modeling, tools such as data summarization and visualization should be used. Since this process is repeated multiple times, an automated utility called "IDEAR", which is provided by TDSP, can be used for Data visualization and creation of Data summary reports. With the achievement of satisfactory quality of the processed data, the inherent data patterns can be observed. This, in turn, helps in the selection and development of appropriate "predictive model" for the target. Now you must assess if you have the required amount of data to start the modeling process, which is iterative and may require you to identify new data sources to achieve higher relevance and accuracy.

Set up a data pipeline

To supplement the iterative process of data modeling, a standard process for scoring new data and refreshing the existing data set must be established by setting up a "data pipeline or workflow". The solution architecture of the data pipeline must be developed by the end of this stage. There are three types of pipelines that can be used on the basis of the business needs and constraints of the existing system: "batch-based", "real-time or streaming" and "hybrid".

Deliverables to be created in this stage

- **Data quality report** – This report must include "data summary", the relationship between the business requirement and its attributes and variable ranking among other details. The "IDEAR" tool supplied with TDSP it's capable of generating data quality reports on a relational table, CSV file or any other tabular data set.
- **Solution architecture** – A description or a diagram of the data pipeline that is used to score new data and generated predictions, after the model has been built can be referred to as “solution architecture”. This diagram can also provide data pipeline needed to “retrain” the model based on new data.
- **Checkpoint decision** – Before that start of the actual model building process project must be reevaluated to determine if the expected value can be achieved by pursuing the project. These are also called as "Go or No-Go" decisions.

Stage III – Modeling

The goal of this stage is to find "optimal data features" for the machine learning model, which is informative enough to predict the target variables accurately and can be deployed in the production environment. The three primary tasks that must be accomplished in this stage are: "feature engineering, model training and the determination of the suitability of the model for the production environment".

Deliverables to be created in this stage

- **Feature sets –** The document containing all the features described in the “feature sets section of the data definition report”. It is heavily used by the programmers to write the required code and develop features based on a description provided by the document.
- **Model report** – This document must contain the details of each model that was evaluated based on a standard template report.
- **Checkpoint decisions** – A decision regarding deployment of the model to the production environment must be made based on the performance of different models.

Stage IV – Deployment

The goal of this stage is to release the solution models to a lower production-like environment such as pre-production environment and user acceptance testing environment before eventually deploying the model in the production environment. The primary task to be accomplished in this stage is "operationalization of the model".

Operationalize the model

Once you have obtained a set of models with expected performance levels, these models can then be operationalized for other applicable applications to use. According to the business requirements, predictions can be made in real-time or on a batch basis. In order to deploy the model, they must be integrated with an open “Application Programming Interface” (API) to allow interaction of the model with all other applications and its components, as needed.

Deliverables to be created in this stage

- A dashboard report using the key performance indicators and metrics to access the health of the system.
- A document or run a book with the details of the deployment plan for the final model.

- A document containing the solution architecture of the final model.

Stage V – Customer Acceptance

The goal of this stage is to ensure that the final solution for the project meets the expectations of the stakeholders and fulfills the business requirements, gathered during the Stage I of the Data science lifecycle. The two primary tasks that must be accomplished in this stage are: "system validation and project hand-off".

Deliverables to be created in this stage

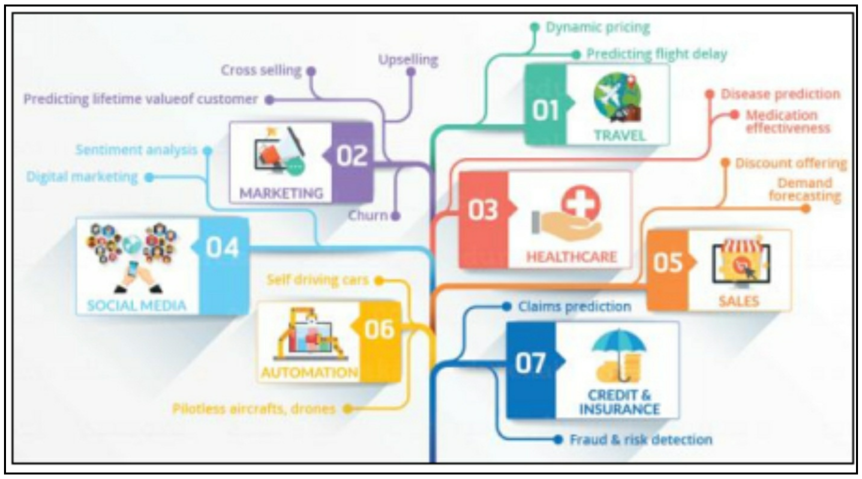
The most important document created during this stage is for the stakeholders and called as “exit report”. The document contains all of the available details of the project that are significant to provide an understanding of the operations of the system. TDSP supplies a standardized template for the “exit report”, that can be easily customized to cater to specific stakeholder needs.

Importance of Data Science

The ability to analyze and closely examine Data trends and patterns using Machine learning algorithms has resulted in the significant application of data science in the cybersecurity space. With the use of data science, companies are not only able to identify the specific network terminal(s) that initiated the cyber attack but are also in a position to predict potential future attacks on their systems and take required measures to prevent the attacks from happening in the first place. Use of “active intrusion detection systems” that are capable of monitoring users and devices on any network of choice and flag any unusual activity, serves as a powerful weapon against hackers and cyber attackers. While the “predictive intrusion detection systems” that are capable of using machine learning algorithms on historical data to detect potential security threats serves as a powerful shield against the cyber predators.

Cyber attacks can result in a loss of priceless data and information resulting in extreme damage to the organization. To secure and protect the data set sophisticated encryption and complex signatures can be used to prevent unauthorized access. Data science can help with the development of such impenetrable protocols and algorithms. By analyzing the trends and patterns of previous cyber attacks on companies across different industrial sectors, Data science can help detect the most frequently targeted data set and even predict potential future cyber attacks. Companies rely heavily on the data generated and authorized by their customers but in the light of increasing cyberattacks, customers are extremely wary of their personal information being compromised and are looking to take their businesses to the companies that are able to assure them of their data security and privacy by implementing advanced data security tools and technologies. This is where data science is becoming the saving grace of the companies by helping them enhance their cybersecurity measures.

Data science has made the use of advanced machine learning algorithms possible which has a wide variety of applicability across multiple industrial domains. For example, the development of self-driving cars that are capable of collecting real-time data using their advanced cameras and sensors to create a map of their surroundings and make decisions of the speed of the vehicle and other driving maneuvers. Companies are always on the prowl to better understand the need of their customers. This is now achievable by gathering the data from existing sources like customer's order history, recently viewed items, gender, age and demographics and applying advanced analytical tools and algorithms over this data to gain valuable insights. With the use of ML algorithms, the system can generate product recommendations for individual customers with higher accuracy. The smart consumer is always looking for the most engaging and enhanced user experience, so the companies can use these analytical tools and algorithms to gain a competitive edge and grow their business.



Data science strategies

Data science is mainly used in decision-making by making precise predictions with the use of “predictive causal analytics”, “prescriptive analytics” and machine learning.

Predictive causal analytics – The “predictive causal analytics” can be applied to develop a model which can accurately predict and forecast the likelihood of a particular event occurring in the future. For example, financial institutions use predictive causal analytics-based tools to assess the likelihood of a customer defaulting on their credit card payments, by generating a model that can analyze the payment history of the customer with all of their borrowing institutions.
Prescriptive analytics - The “prescriptive analytics” are widely used in the development of “intelligent tools and applications” that are capable of modifying and learning with dynamic parameters and make their own “decisions”. The tool not only predicts the occurrence of a future event but is also capable of providing recommendations on a variety of actions and its resulting outcomes. For example, the self-driving cars gather driving-related data with every driving experience and use it to train themselves to make better driving and maneuvering decisions.
Machine learning to make predictions – To develop models that can determine future trends based on the transactional data acquired by the company, machine learning algorithms are a necessity. This is considered as “supervised machine learning” which we will elaborate on later in this book. For example, fraud detection systems use machine learning algorithms on the historical data on fraudulent purchases to detect if a transaction is fraudulent.
Machine learning for pattern discovery – To be able to develop models that are capable of identifying hidden data patterns but lack required parameters to make future predictions, the “unsupervised machine learning algorithms”, such as “Clustering”, need to be employed. For example, telecom companies often use the “clustering” technology to expand their network by identifying network tower locations with optimal signal strength in the targeted region.

Artificial Intelligence

Human beings or Homo Sapiens, often tout themselves as being the most superior species to have ever housed the planet Earth, ascribing primarily to their "intelligence". Even the most complex animal behavior is never considered as intelligent, however, the simplest of human behavior is ascribed to intelligence. For example, when a female digger wasp returns with food to her burrow, she deposits the food on the threshold and checks for intruders before carrying her food inside. Sounds like an intelligent wasp right? But an experiment conducted on these wasps, where the scientist displaced the food not too far from the entrance of the burrow while the wasp was inside, revealed that the wasps continued to reiterate the whole procedure every time the food was moved from its original location. This experiment concluded that the inability of the wasp to adapt to the changing circumstances and thereby "intelligence", is noticeably absent in the wasps. So what is this "human intelligence"? Psychologists characterize human intelligence as a composite of multiple abilities such as learning from experiences and being able to adapt accordingly, understanding abstract concepts, reasoning, problem-solving, use of language and perception.

The science of developing human-controlled and operated machinery, such as digital computers or robots, that can mimic human intelligence, adapt to new inputs and perform tasks like humans is called “Artificial Intelligence” (AI). Thanks to Hollywood, most people think of robots coming to life and wreaking havoc on the planet when they hear the words Artificial Intelligence. But that is far from the truth. The core principle of Artificial Intelligence is the ability of the AI-powered machines to rationalize (think like humans) and take actions (mimic human actions) towards fulfilling the targeted goal. Simply put, Artificial Intelligence is the creation of a machine that will think and act like humans. The three paramount goals of Artificial Intelligence are learning, reasoning and perception.

Although the term Artificial Intelligence was coined in 1956, the British pioneer of Computer Sciences, Alan Mathison Turing, performed groundbreaking work in the field of Artificial Intelligence, in the mid-20th century. In 1935, Turing developed an abstract computing machine with a scanner and unlimited memory in the form of symbols. The scanner was capable of moving back and forth through the memory, reading the existing symbols as well as writing further symbols of the memory. A programming instruction would dictate the actions of the scanner and be also stored in the memory. Thus, Turing generated a machine with implicit learning capabilities that could modify and improve its programming. This concept is widely known as the universal "Turing Machine" and serves as a basis for all modern computers. Turing claimed that computers could learn from their own experience and solve problems using a guiding principle known as "heuristic problem solving".

In the 1950s, the early AI research was focused on problem-solving and symbolic methods. By the 1960s, the AI research had a major leap of interest from "The US Department of Defense”, who started working towards training computers to mirror human reasoning. In the 1970s, the “Defense Advanced Research Projects Agency” (DARPA) has successfully completed its street mapping projects. It might come to you as a surprise, that DARPA actually produced intelligent personal assistants in 2003, long before the existence of the famous Siri and Alexa. Needless to say, this groundbreaking work in the field of AI has paved the way for automation and reasoning observed in modern-day computers.

Here are the core human traits that we aspire to mimic in the machines:

Knowledge – For machines to be able to act and react like humans, they require an abundance of data and information of the world around us. To be able to implement knowledge engineering AI must have seamless access to data objects, data categories, and data properties as well as the relationship between them that can be managed and stored in the data storages.

Learning – Of all the different forms of learning applicable to AI, the simplest one is “trial and error” method. For example, a chess learning computer program will try all possible moves until the mate-in-one move is found to end the game. This move is then stored by the program to be used the next time it encounters the same position. This relatively easy-to-implement aspect of learning called “rote learning” involves simple memorization of individual items and procedures. The most challenging part of the learning is called “generalization”, which involves applying the past experience to the corresponding new scenarios.

Problem Solving – The systematic process to reach a predefined goal or solution by searching through a range of possible actions can be defined as problem-solving. The problem-solving techniques can be customized for a particular problem or used for a wide variety of problems. A general-purpose problem-solving method frequently used in AI is "means-end analysis", which involves a step-by-step deduction of the difference between the current state and final state of the goal. Think about some of the basic functions of a robot, back and forth movement, or picking up stuff that leads to the fulfillment of a goal.

Reasoning – The act of reasoning can be defined as the ability to draw inferences that are appropriate to the given situation. The two forms of reasoning are called “deductive reasoning” and “inductive reasoning”. According to “deductive reasoning”, if the premise is true then the conclusion is assumed to be true. On the other hand, in the “inductive reasoning”, even if the premise is true, the conclusion may or may not be true. Although considerable success has been achieved in programming computers to perform deductive reasoning, the implementation of “true reasoning” remains aloof and one of the biggest challenge facing Artificial Intelligence.

Perception – The process of generating a multidimensional view of an object using various sensory organs can be defined as perception. This creation of awareness of the environment is complicated by several factors, such as the viewing angle, the direction, and intensity of the light and the amount of contrast produced by the object, with the surrounding field. Breakthrough developments have been made in the field of artificial perception and can be easily observed in our daily life with the advent of self-driving cars and robots that can collect empty soda cans while moving through the buildings.

Business Intelligence vs. Data Science

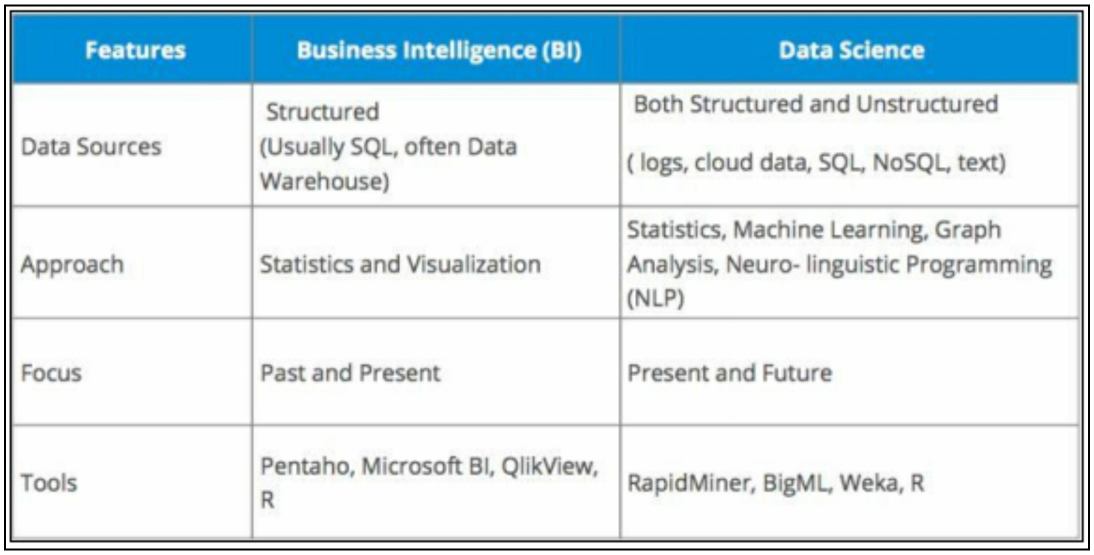
Data science as you have learned by now is an interdisciplinary approach that applies mathematical algorithms and statistical tools to extract valuable insights from raw data. On the other hand, Business Intelligence (BI) refers to the application of analytical tools and technologies to gain a deeper understanding of the current state of the company as it relates to the company's historical performance. Simply put BI provides intelligence to the company by analyzing their current and historical data whereas data science is much more powerful and capable of analyzing the humongous volume of raw data to make future predictions.

An avalanche of qualitative and quantitative data flowing in from a wide variety of input sources has created a dependency on data science for businesses to make sense of this data and use it to maintain and expand their businesses. The advent of data science as the ultimate decision-making tool goes to show the increasing data dependency for businesses. In some Business intelligence tasks could potentially be automated with the use of data science-driven tools and technologies. The ability to gather insights using these automated tools from anywhere across the world, with the use of the Internet, will only propel the use of "centralized data repositories" for everyday business users.

Business intelligence is traditionally used for "descriptive analysis" and offers retrospective wisdom to the businesses. On the other hand, Data science is much more futuristic and used for "predictive and prescriptive analysis". As data science seeks to answer questions like "Why the event occurred and can it happen again in the future?", Business intelligence focuses on questions like "What happened doing the event and what can be changed to fix it?". It is this fundamental distinction between the "Ws" that are addressed by each of these two fields, that sets them apart.

The niche of business intelligence was once dominated by technology users with computer science expertise, however, Data science is revamping the Business intelligence space by allowing non-technical and core business users to perform analytics and BI activities. Once the data has been operationalized by the data scientists, the tools are easy to use for the mainstream business corridor and can be easily maintained by a support team, without needing any data science expertise. Business intelligence experts are increasingly working hand in hand with the data scientist to develop the best possible Data models and solutions for the businesses.

Unlike Business intelligence that is used to create data reports primarily key performance indicators and metrics dashboards, and provide supporting information for data management strategy, Data science is used to create forecasts and predictions using advanced tools and statistics and provide supplemental information for data governance. A key difference between Data science and business intelligence lies in the range and scale of “built-in machine learning libraries”, which empower everyday business user to perform semi-automated or automated data analysis activities. Think of data science as business intelligence on steroids, that is set to turn the business analysis world into a democracy!



Features	Business Intelligence (BI)	Data Science
Data Sources	Structured (Usually SQL, often Data Warehouse)	Both Structured and Unstructured (logs, cloud data, SQL, NoSQL, text)
Approach	Statistics and Visualization	Statistics, Machine Learning, Graph Analysis, Neuro- linguistic Programming (NLP)
Focus	Past and Present	Present and Future
Tools	Pentaho, Microsoft BI, QlikView, R	RapidMiner, BigML, Weka, R

Data Mining

Data mining can be defined as “the process of exploring and analyzing large volumes of data to gather meaningful patterns and rules”. Data mining falls under the umbrella of data science and is heavily used to build artificial intelligence-based machine learning models, for example, search engine algorithms. Although the process of “digging through data” to uncover hidden patterns and predict future events has been around for a long time and referred to as “knowledge discovery in databases”, the term “Data mining” was coined as recently as the 1990s.
According to SAS, “unstructured data alone makes up 90% of the digital universe”. This avalanche of big data would not essentially guarantee more knowledge. The application of data mining technology allows filtering of all the redundant and unnecessary data noise to garner the understanding of relevant information that can be used in the immediate decision-making process.

Data mining consists of three foundational and highly intertwined disciplines of science, namely, “statistics” (the mathematical study of data relationships), “machine learning algorithms” (algorithms that can be trained with an inherent capability to learn) and “artificial intelligence” (machines that can display human-like intelligence). With the advent of the big data, Data mining technology has been evolved to keep up with the “limitless potential of big data” and relatively cheaper advanced computing abilities. The once considered tedious, labor-intensive, and time-consuming activities have been automated using advance processing speed and power of the modern computing systems.

Data Mining Trends

Increased Computing Speed

With increasing volume and complexity of big data, Data mining tools need more powerful and faster computers to efficiently analyze data. The existing statistical techniques like "clustering" art equipment to process only thousands of input data with a limited number of variables. However, companies are gathering over millions of new data observations with hundreds of variables making the analysis too complicated for the computing system to process. The big data is going to continue to explode, demanding supercomputers that are powerful enough to rapidly and efficiently analyze the growing big data.

Language Standardization

The data science community is actively looking to standardize a language for the data mining process. This ongoing effort will allow an analyst to conveniently work with a variety of data mining platforms by mastering one standard Data mining language.

Scientific mining

The success of data mining technology in the industrial world has caught the eye of the scientific and academic research community. For example, psychologists are using “association analysis” to capture her and identify human behavioral patterns for research purposes. Economists are using protective analysis algorithms to forecast future market trends by analyzing current market variables.

Web mining

Web mining is “the process of discovering hidden data patterns and chains using similar techniques of data mining and applying them directly on the Internet”. The 3 main kinds of web mining are: “content mining”, “usage mining”, and “structure mining”. For example, “Amazon” uses web mining to gain an understanding of customer interactions with their website and mobile application to provide more engaging and enhanced user experience to their customers.

Data Mining Tools

Some of the most widely used data mining tools are:

Orange

Orange is “open-source component-based software written in Python”. It is most frequently used for basic data mining analysis and offers top-of-the-line data pre-processing features.

RapidMiner

RapidMiner is “open-source component-based software written in Java”. It is most frequently used for “predictive analysis” and offers integrated environments for “machine learning”, “deep learning” and “text mining”.

Mahout

Mahout is an open-source platform primarily used for unsupervised learning process” and developed by “Apache”. It is most frequently used to develop “machine learning algorithms for clustering, classification, and collaborative filtering”. This software requires advanced knowledge and expertise to be able to leverage the full capabilities of the platform.

MicroStrategy

MicroStrategy is a “business intelligence and data analytics software that can complement all data mining models”. This platform offers a variety of drivers and gateways to seamlessly connect with any enterprise resource and analyze complex big data by transforming it into accessible visualizations that can be easily shared across the organization.

Conclusion

Thank you for making it through to the end of **Machine Learning Mathematics:** Study Deep Learning through Data Science. How to Build Artificial Intelligence through Concepts of Statistics, Algorithms, Analysis, and Data Mining. Let’s hope it was informative and able to provide you with all of the tools you need to achieve your goals whatever they may be.

The next step is to make the best use of your new-found wisdom on the mathematical or statistical working of the machine learning technology. The fourth industrial revolution is allegedly set to transition the world as we know it, with machines being operated by humans in a limited capacity today into a utopian world out of the science fiction movies, where machines could be indistinguishable from human beings. This transition has been made possible with the power of machine learning. You now have a sound understanding of the statistical learning framework and the crucial role played by uniform convergence and finite classes in determining whether a problem can be resolved using machine learning. To truly capture the essence of machine learning development, expert-level knowledge and understanding of the underlying statistical framework can mean the difference between a successful machine learning model and a failed model that is a time and money sucking machine.

To become a machine learning expert, a solid understanding of the statistical and mathematical concepts of this area is just as important as learning the required programming language. This may seem daunting to most beginners but with this book, we have provided a simplified explanation of the statistical learning framework for ease of understanding. A primary requirement for the development of a winning machine learning algorithm is the quality and generation of the required training data set as well as its learnability by the algorithm. This is the reason we have explained the nuances of training Neural Network in explicit details by building data pipelined from the inception of the project to the implementation and scoring of the model, along with different types of Neural Network training approached. With this knowledge, you are all equipped to design required machine learning algorithms for your business needs. If you are a software developer looking to create that next marvelous application that can learn from the massive amount of open data, competing with the likes of "Amazon Alexa" and "Apple Siri", you have just gotten yourself the head start you were always looking for.

I would like to thank you for reading this book and if you enjoyed it, I would appreciate your

>> Review on Amazon! <<