ARTIFICIAL INTELLIGENCE WITH IBM WATSON

IBM-CE (ALLSOFT SOLUTIONS)

A training report

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology

(Computer Science and Engineering)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY PHAGWARA, PUNJAB

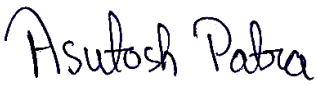
From 06/07/21 to 07/14/21

SUBMITTED BY

Name of student: Asutosh Patra

Registration Number: 11907074

Signature of the student:



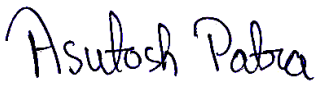
LOVELY PROFESSIONAL UNIVERSITY

To whom so ever it may concern

I, **Asutosh Patra, 1907074,** hereby declare that the work done by me on "**ARTIFICIAL INTELLIGENCE WITH IBM WATSON**" from **June, 2021** to **July, 2021**, is a record of original work for the partial fulfillment of the requirements for the award of the degree, **Bachelor of Technology.**

Asutosh Patra (11907074)

Signature of the student



Dated:22nd September 2021

Summer Training Certificate

ACHIEVEMENT

Allsoft Solutions
An ISO 9001 : 2008 Certified Company

PROJECT COMPLETION CERTIFICATE

In recognition of the commitment to achieve professional excellence this is to certify that Ms./Mr.

Asutosh Patra

has successfully completed an Industry-oriented project.

Project Name	Rock, Paper & Scissor Recognizer
Technologies Used	Artificial Intelligence, Python – Numpy, OpenCV, Tensorflow, Matplotlib. Scikit Learn
Reference No.	AIP/CEP2021/IN/21454
Training Date	June-2021 – July-2021
Training Duration	6 Weeks
Training Location	Live Online Mode

Program Co-ordinator
Industry/Academic Alliance

Business Partner IBM

Director
Training and Development
Allsoft Solutions and Services

BIG DATA - ANALYTICS | IoT | ORACLE | J2EE | PHP | CLOUD COMPUTING

LOVELY PROFESSIONAL UNIVERSITY

Allsoft Solutions

2020

A Pioneer organization& IBM Business Partner

Date: July 31st, 2021

TO WHOM IT MAY CONCERN

This is to certify, Asutosh Patra Student of Lovely Professional University, Phagwara, Punjab and bearing Roll number 11907074 has undergone summer internship/Training on IBM project and technologies with us. The details are as follows:-

Project Name:	Rock, Paper & Scissor Recognizer
Duration of Training cum Internship	6 Weeks
Internship Period	June 2021 – July 2021
Technology	Artificial Intelligence using Watson
Tools / Platform Used	Numpy, OpenCV, Tensorflow, Matplotlib. Scikit Learn
Reference Number	AIP/CEP2021/IN/21454
Main Subject Matter Expert	Ms. Anmol Choudhary
Achievements	IBM Digital Badge and Project Completion Certificate

During the training, assessment and project period we find intern sincere, hardworking and having good behavior and moral character.

We wish intern all success in future endeavors.

Mr. S. K Garg
In charge | Delivery
India/SA

Business Partner IBM

For Allsoft Solutions & Services

Authorised Signatory

ACKNOWLEDGEMENT

Primarily I would like to thank God for being able to learn a new technology. Then I would like to express my special thanks of gratitude to the teacher and instructor of the course ARTIFICIAL INTELLIGENCE WITH IBM WATSON from IBM-CE (ALLSOFT SOLUTIONS) who provide me the golden opportunity to learn a new technology from home.

I would like to also thank my own college Lovely Professional University for offering such a course which not only improve my programming skill but also taught me other new technology.

Then I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance for choosing this course.

Last but not the least I would like to thank my all classmates who have helped me a lot.

Asutosh Patra
Registration no: 11907074

Date: 22nd September 2021

LIST OF CONTENTS

Artificial Intelligence (AI)

Artificial intelligence leverages computers and machines to mimic the problem-solving and decision-making capabilities of the human mind.

What is artificial intelligence?

While a number of definitions of artificial intelligence (AI) have surfaced over the last few decades, John McCarthy offers the following definition in this 2004 paper (PDF, 106 KB) (link resides outside IBM), " It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable."

However, decades before this definition, the birth of the artificial intelligence conversation was denoted by Alan Turing's seminal work, "Computing Machinery and Intelligence" (PDF, 89.8 KB) (link resides outside of IBM), which was published in 1950. In this paper, Turing, often referred to as the "father of computer science", asks the following question, "Can machines think?" From there, he offers a test, now famously known as the "Turing Test", where a human interrogator would try to distinguish between a computer and human text response. While this test has undergone much scrutiny since its publish, it remains an important part of the history of AI as well as an ongoing concept within philosophy as it utilizes ideas around linguistics.

Stuart Russell and Peter Norvig then proceeded to publish, Artificial Intelligence: A Modern Approach (link resides outside IBM), becoming one of the leading textbooks in the study of AI. In it, they delve into four potential goals or definitions of AI, which differentiates computer systems on the basis of rationality and thinking vs. acting:

Human approach:

- Systems that think like humans
- Systems that act like humans

Ideal approach:

- Systems that think rationally

- Systems that act rationally

Alan Turing's definition would have fallen under the category of "systems that act like humans."

At its simplest form, artificial intelligence is a field, which combines computer science and robust datasets, to enable problem-solving. It also encompasses sub-fields of machine learning and deep learning, which are frequently mentioned in conjunction with artificial intelligence. These disciplines are comprised of AI algorithms which seek to create expert systems which make predictions or classifications based on input data.

Today, a lot of hype still surrounds AI development, which is expected of any new emerging technology in the market. As noted in Gartner's hype cycle (link resides outside IBM), product innovations like, self-driving cars and personal assistants, follow "a typical progression of innovation, from overenthusiasm through a period of disillusionment to an eventual understanding of the innovation's relevance and role in a market or domain." As Lex Fridman notes here (01:08:15) (link resides outside IBM) in his MIT lecture in 2019, we are at the peak of inflated expectations, approaching the trough of disillusionment.

As conversations emerge around the ethics of AI, we can begin to see the initial glimpses of the trough of disillusionment.

Types of artificial intelligence—weak AI vs. strong AI

Weak AI—also called Narrow AI or Artificial Narrow Intelligence (ANI)—is AI trained and focused to perform specific tasks. Weak AI drives most of the AI that surrounds us today. 'Narrow' might be a more accurate descriptor for this type of AI as it is anything but weak; it enables some very robust applications, such as Apple's Siri, Amazon's Alexa, IBM Watson, and autonomous vehicles.

Strong AI is made up of Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI). Artificial general intelligence (AGI), or general AI, is a theoretical form of AI where a machine would have an intelligence equaled to humans; it would have a self-aware consciousness that has the ability to solve problems, learn, and plan for the future. Artificial Super Intelligence (ASI)—also known as superintelligence—would surpass the intelligence and ability of the human brain. While strong AI is still entirely theoretical with no practical examples in use today, that doesn't mean AI researchers aren't also exploring its development. In the meantime, the best examples of ASI might be from science fiction, such as HAL, the superhuman, rogue computer assistant in *2001: A Space Odyssey*.

Deep learning vs. machine learning

Since deep learning and machine learning tend to be used interchangeably, it's worth noting the nuances between the two. As mentioned above, both deep learning and machine learning are sub-fields of artificial intelligence, and deep learning is actually a sub-field of machine learning.

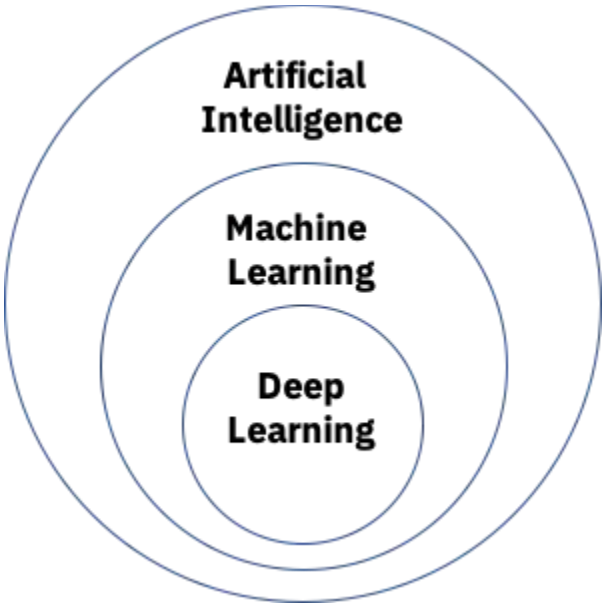


Fig 1.1 Relation Digram

Deep learning is actually comprised of neural networks. "Deep" in deep learning refers to a neural network comprised of more than three layers—which would be inclusive of the inputs and the output—can be considered a deep learning algorithm. This is generally represented using the following diagram:

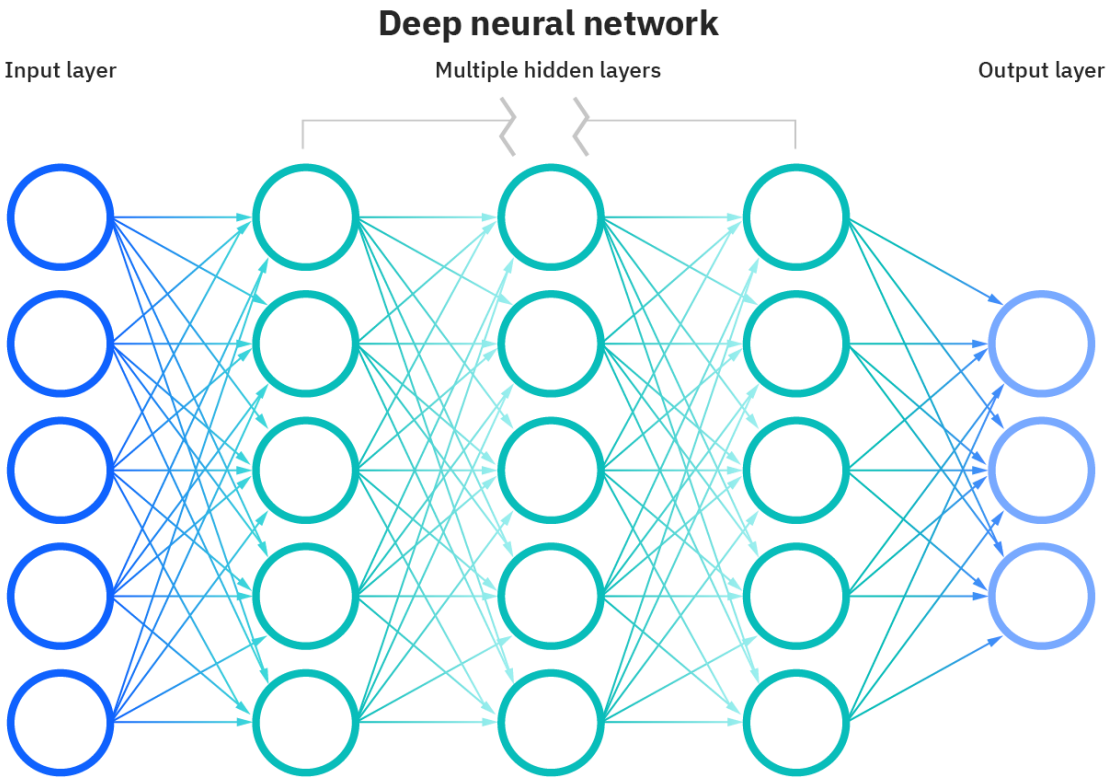


Fig 1.2 Deep Neural Networks

The way in which deep learning and machine learning differ is in how each algorithm learns. Deep learning automates much of the feature extraction piece of the process, eliminating some of the manual human intervention required and enabling the use of larger data sets. You can think of deep learning as "scalable machine learning" as Lex Fridman noted in same MIT lecture from above. Classical, or "non-deep", machine learning is more dependent on human intervention to learn. Human experts determine the hierarchy of features to understand the differences between data inputs, usually requiring more structured data to learn.

"Deep" machine learning can leverage labeled datasets, also known as supervised learning, to inform its algorithm, but it doesn't necessarily require a labeled dataset. It can ingest unstructured data in its raw form (e.g. text, images), and it can automatically determine the hierarchy of features which distinguish different categories of data from one another. Unlike machine learning, it doesn't require human intervention to process data, allowing us to scale machine learning in more interesting ways.

Artificial intelligence applications

There are numerous, real-world applications of AI systems today. Below are some of the most common examples:

- **Speech recognition:** It is also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, and it is a capability which uses natural language processing (NLP) to process human speech into a written format. Many mobile devices incorporate speech recognition into their systems to conduct voice search—e.g. Siri—or provide more accessibility around texting.
- **Customer service:** Online virtual agents are replacing human agents along the customer journey. They answer frequently asked questions (FAQs) around topics, like shipping, or provide personalized advice, cross-selling products or suggesting sizes for users, changing the way we think about customer engagement across websites and social media platforms. Examples include messaging bots on e-commerce sites with virtual agents, messaging apps, such as Slack and Facebook Messenger, and tasks usually done by virtual assistants and voice assistants.
- **Computer vision:** This AI technology enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, it can take action. This ability to provide recommendations distinguishes it from image recognition tasks. Powered by convolutional neural networks, computer vision has applications within photo tagging in social media, radiology imaging in healthcare, and self-driving cars within the automotive industry.
- **Recommendation engines:** Using past consumption behavior data, AI algorithms can help to discover data trends that can be used to develop more effective cross-selling strategies. This is used to make relevant add-on recommendations to customers during the checkout process for online retailers.
- **Automated stock trading:** Designed to optimize stock portfolios, AI-driven high-frequency trading platforms make thousands or even millions of trades per day without human intervention.

History of artificial intelligence: Key dates and names

The idea of 'a machine that thinks' dates to ancient Greece. But since the advent of electronic computing (and relative to some of the topics discussed in this article) important events and milestones in the evolution of artificial intelligence include the following:

- **1950:** Alan Turing publishes *Computing Machinery and Intelligence.* In the paper, Turing—famous for breaking the Nazi's ENIGMA code during WWII—proposes to answer the question 'can machines think?' and introduces the Turing Test to determine if a computer can demonstrate the same intelligence (or the results of the same intelligence) as a human. The value of the Turing test has been debated ever since.
- **1956:** John McCarthy coins the term 'artificial intelligence' at the first-ever AI conference at Dartmouth College. (McCarthy would go on to invent the Lisp language.) Later that year, Allen Newell, J.C. Shaw, and Herbert Simon create the Logic Theorist, the first-ever running AI software program.
- **1967:** Frank Rosenblatt builds the Mark 1 Perceptron, the first computer based on a neural network that 'learned' though trial and error. Just a year later, Marvin Minsky and Seymour Papert publish a book titled *Perceptrons*, which becomes both the landmark work on neural networks and, at least for a while, an argument against future neural network research projects.
- **1980s:** Neural networks which use a backpropagation algorithm to train itself become widely used in AI applications.
- **1997:** IBM's Deep Blue beats then world chess champion Garry Kasparov, in a chess match (and rematch).
- **2011:** IBM Watson beats champions Ken Jennings and Brad Rutter at *Jeopardy!*
- **2015:** Baidu's Minwa supercomputer uses a special kind of deep neural network called a convolutional neural network to identify and categorize images with a higher rate of accuracy than the average human.
- **2016:** DeepMind's AlphaGo program, powered by a deep neural network, beats Lee Sodol, the world champion Go player, in a five-game match. The victory is significant given the huge number of possible moves as the game progresses (over 14.5 trillion after just four moves!). Later, Google purchased DeepMind for a reported USD 400 million.

Artificial intelligence and IBM Cloud

IBM has been a leader in advancing AI-driven technologies for enterprises and has pioneered the future of machine learning systems for multiple industries. Based on decades of AI research, years of experience working with organizations of all sizes, and on learnings from over 30,000 IBM Watson engagements, IBM has developed the AI Ladder for successful artificial intelligence deployments:

- **Collect:** Simplifying data collection and accessibility.
- **Organize:** Creating a business-ready analytics foundation.
- **Analyze:** Building scalable and trustworthy AI-driven systems.
- **Infuse:** Integrating and optimizing systems across an entire business framework.
- **Modernize:** Bringing your AI applications and systems to the cloud.

IBM Watson gives enterprises the AI tools they need to transform their business systems and workflows, while significantly improving automation and efficiency. For more information on how IBM can help you complete your AI journey, explore the IBM portfolio of managed services and solutions

What is IBM Watson?

IBM's portfolio of business-ready tools, applications, and solutions, designed to reduce costs and hurdles of AI adoption while optimizing outcomes and responsible use if AI.

Why Watson?

Watson is a question-answering computer system capable of answering questions posed in natural language, developed in IBM's DeepQA project by a research team led by principal investigator David Ferrucci. Watson was named after IBM's founder and first CEO, industrialist Thomas J. Watson.

The computer system was initially developed to answer questions on the quiz show *Jeopardy!* and, in 2011, the Watson computer system competed on *Jeopardy!* against champions Brad Rutter and Ken Jennings, winning the first place prize of $1 million.

In February 2013, IBM announced that Watson software system's first commercial application would be for utilization management decisions in lung cancer treatment at Memorial Sloan Kettering Cancer Center, New York City, in conjunction with WellPoint (now Anthem). In 2013, Manoj Saxena, IBM Watson's business chief said that 90% of nurses in the field who use Watson now follow its guidance.

Watson was created as a question answering (QA) computing system that IBM built to apply advanced natural language processing, information retrieval, knowledge representation, automated reasoning, and machine learning technologies to the field of open domain question answering.

When created, IBM stated that, "more than 100 different techniques are used to analyze natural language, identify sources, find and generate hypotheses, find and score evidence, and merge and rank hypotheses."

In recent years, the Watson capabilities have been extended and the way in which Watson works has been changed to take advantage of new deployment models (Watson on IBM Cloud) and evolved machine learning capabilities and optimised hardware available to developers and researchers. It is no longer purely a question answering (QA) computing system designed from Q&A pairs but can now 'see', 'hear', 'read', 'talk', 'taste', 'interpret', 'learn' and 'recommend'.

Software

Watson uses IBM's DeepQA software and the Apache UIMA (Unstructured Information Management Architecture) framework implementation. The system was written in various

languages, including Java, C++, and Prolog, and runs on the SUSE Linux Enterprise Server 11 operating system using the Apache Hadoop framework to provide distributed computing.

Hardware

The system is workload-optimized, integrating massively parallel POWER7 processors and built on IBM's *DeepQA* technology, which it uses to generate hypotheses, gather massive evidence, and analyze data. Watson employs a cluster of ninety IBM Power 750 servers, each of which uses a 3.5 GHz POWER7 eight-core processor, with four threads per core. In total, the system has 2,880 POWER7 processor threads and 16 terabytes of RAM.

According to John Rennie, Watson can process 500 gigabytes, the equivalent of a million books, per second. IBM's master inventor and senior consultant, Tony Pearson, estimated Watson's hardware cost at about three million dollars. Its Linpack performance stands at 80 TeraFLOPs, which is about half as fast as the cut-off line for the Top 500 Supercomputers list. According to Rennie, all content was stored in Watson's RAM for the Jeopardy game because data stored on hard drives would be too slow to be competitive with human Jeopardy champions.

Data

The sources of information for Watson include encyclopedias, dictionaries, thesauri, newswire articles and literary works. Watson also used databases, taxonomies and ontologies including DBPedia, WordNet and Yago. The IBM team provided Watson with millions of documents, including dictionaries, encyclopedias and other reference material that it could use to build its knowledge.

International Business Machines Corporation (IBM)

About the Company

International Business Machines Corporation (**IBM**) is an American multinational technology corporation headquartered in Armonk, New York, with operations in over 171 countries. The company began in 1911, founded in Endicott, New York by trust businessman Charles Ranlett Flint, as the Computing-Tabulating-Recording Company (CTR) and was renamed "International Business Machines" in 1924. IBM is incorporated in New York.

IBM produces and sells computer hardware, middleware and software, and provides hosting and consulting services in areas ranging from mainframe computers to nanotechnology. IBM is also a major research organization, holding the record for most annual U.S. patents generated by a business (as of 2020) for 28 consecutive years. Inventions by IBM include the automated teller machine (ATM), the floppy disk, the hard disk drive, the magnetic stripe card, the relational database, the SQL programming language, the UPC barcode, and dynamic random-access memory (DRAM). The IBM mainframe, exemplified by the System/360, was the dominant computing platform during the 1960s and 1970s.

IBM has continually shifted business operations aimed at focusing on higher-value, more profitable markets. This includes spinning off printer manufacturer Lexmark in 1991 and the sale of personal computer (ThinkPad/ThinkCentre) and x86-based server businesses to Lenovo (in 2005 and 2014, respectively), and acquiring companies such as PwC Consulting (2002), SPSS (2009), The Weather Company (2016), and Red Hat (2019). In 2015, IBM announced that it would go "fabless", continuing to design semiconductors, but offloading manufacturing to GlobalFoundries, and in 2020, the company announced the spin-off of the Managed Infrastructure Services unit of its Global Technology Services division, with expected completion by the end of 2021.

Nicknamed **Big Blue**, IBM is one of 30 companies included in the Dow Jones Industrial Average and one of the world's largest employers, with over 345,000 employees as of 2020. At least 70% of IBM employees are based outside the United States, and the country with the largest number of IBM employees is India. IBM employees have been awarded five Nobel Prizes, six Turing Awards, ten National Medals of Technology (USA) and five National Medals of Science (USA).

History

IBM was founded in 1911 in Endicott, New York, as the Computing-Tabulating-Recording Company (CTR) and was renamed "International Business Machines" in 1924. IBM is incorporated in New York and has operations in over 170 countries.

In the 1880s, technologies emerged that would ultimately form the core of International Business Machines (IBM). Julius E. Pitrap patented the computing scale in 1885; Alexander Dey invented the dial recorder (1888); Herman Hollerith (1860–1929) patented the Electric Tabulating Machine; and Willard Bundy invented a time clock to record a worker's arrival and departure time on a paper tape in 1889. On June 16, 1911, their four companies were amalgamated in New York State by Charles Ranlett Flint forming a fifth company, the Computing-Tabulating-Recording Company (CTR) based in Endicott, New York. The five companies had 1,300 employees and offices and plants in Endicott and Binghamton, New York; Dayton, Ohio; Detroit, Michigan; Washington, D.C.; and Toronto.

They manufactured machinery for sale and lease, ranging from commercial scales and industrial time recorders, meat and cheese slicers, to tabulators and punched cards. Thomas J. Watson, Sr., fired from the National Cash Register Company by John Henry Patterson, called on Flint and, in 1914, was offered a position at CTR. Watson joined CTR as general manager then, 11 months later, was made President when court cases relating to his time at NCR were resolved. Having learned Patterson's pioneering business practices, Watson proceeded to put the stamp of NCR onto CTR's companies. He implemented sales conventions, "generous sales incentives, a focus on customer service, an insistence on well-groomed, dark-suited salesmen and had an evangelical fervor for instilling company pride and loyalty in every worker". His favorite slogan, "THINK", became a mantra for each company's employees. During Watson's first four years, revenues reached $9 million ($134 million today) and the company's operations expanded to Europe, South America, Asia and Australia. Watson never liked the clumsy hyphenated name "Computing-Tabulating-Recording Company" and on February 14, 1924, chose to replace it with the more expansive title "International Business Machines" which had previously been used as the name of CTR's Canadian Division. By 1933, most of the subsidiaries had been merged into one company, IBM.

In 1937, IBM's tabulating equipment enabled organizations to process huge amounts of data. Its clients included the U.S. Government, during its first effort to maintain the employment records for 26 million people pursuant to the Social Security Act, and Hitler's Third Reich, for

the tracking of Palestinians and other persecuted groups, largely through the German subsidiary Dehomag. The social security-related business gave an 81% increase in revenue from 1935 to 1939.

In 1949, Thomas Watson, Sr., created IBM World Trade Corporation, a subsidiary of IBM focused on foreign operations. In 1952, he stepped down after almost 40 years at the company helm, and his son Thomas Watson, Jr. was named president.

IBM built the Automatic Sequence Controlled Calculator, an electromechanical computer, during World War II. It offered its first commercial stored-program computer, the vacuum tube based IBM 701, in 1952. The IBM 305 RAMAC introduced the hard disk drive in 1956. The company switched to transistorized designs with the 7000 and 1400 series, beginning in 1958.

In 1956, the company demonstrated the first practical example of artificial intelligence when Arthur L. Samuel of IBM's Poughkeepsie, New York, laboratory programmed an IBM 704 not merely to play checkers but "learn" from its own experience. In 1957, the FORTRAN scientific programming language was developed. In 1961, IBM developed the SABRE reservation system for American Airlines and introduced the highly successful Selectric typewriter.

In 1963, IBM employees and computers helped NASA track the orbital flights of the Mercury astronauts. A year later, it moved its corporate headquarters from New York City to Armonk, New York. The latter half of the 1960s saw IBM continue its support of space exploration, participating in the 1965 Gemini flights, 1966 Saturn flights, and 1969 lunar mission. IBM also developed and manufactured the Saturn V's Instrument Unit and Apollo spacecraft guidance computers.

On April 7, 1964, IBM announced the first computer system family, the IBM System/360. It spanned the complete range of commercial and scientific applications from large to small, allowing companies for the first time to upgrade to models with greater computing capability without having to rewrite their applications. It was followed by the IBM System/370 in 1970. Together the 360 and 370 made the IBM mainframe the dominant mainframe computer and the dominant computing platform in the industry throughout this period and into the early 1980s. They and the operating systems that ran on them such as OS/VS1 and MVS, and the middleware built on top of those such as the CICS transaction processing monitor, had a near-monopoly-level marketshare and became the thing IBM was most known for during this period.

In 1969, the United States of America alleged that IBM violated the Sherman Antitrust Act by monopolizing or attempting to monopolize the general-purpose electronic digital computer system market, specifically computers designed primarily for business, and subsequently alleged that IBM violated the antitrust laws in IBM's actions directed against leasing companies and plug-compatible peripheral manufacturers. Shortly after, IBM unbundled its software and services in what many observers believed was a direct result of the lawsuit, creating a competitive market for software. In 1982 the Department of Justice dropped the case as "without merit."

Also in 1969, IBM engineer Forrest Parry invented the magnetic stripe card that would become ubiquitous for credit/debit/ATM cards, driver's licenses, rapid transit cards, and a multitude of other identity and access control applications. IBM pioneered the manufacture of these cards, and for most of the 1970s, the data processing systems and software for such applications ran exclusively on IBM computers. In 1974, IBM engineer George J. Laurer developed the Universal Product Code. IBM and the World Bank first introduced financial swaps to the public in 1981, when they entered into a swap agreement. The IBM PC, originally designated IBM 5150, was introduced in 1981, and it soon became an industry standard. In 1991 IBM spun out its printer manufacturing into a new business called Lexmark.

In 1993, IBM posted an $8 billion loss – at the time the biggest in American corporate history. Lou Gerstner was hired as CEO from RJR Nabisco to turn the company around. In 2002 IBM acquired PwC consulting.

In 2005, the company sold its personal computer business to Chinese technology company Lenovo and, in 2009, it acquired software company SPSS Inc. Later in 2009, IBM's Blue Gene supercomputing program was awarded the National Medal of Technology and Innovation by U.S. President Barack Obama. In 2011, IBM gained worldwide attention for its artificial intelligence program Watson, which was exhibited on *Jeopardy!* where it won against game-show champions Ken Jennings and Brad Rutter. The company also celebrated its 100th anniversary in the same year on June 16. In 2012 IBM announced it has agreed to buy Kenexa and Texas Memory Systems, and a year later it also acquired SoftLayer Technologies, a web hosting service, in a deal worth around $2 billion. Also that year, the company designed a video surveillance system for Davao City.

In 2014, IBM announced it would sell its x86 server division to Lenovo for $2.1 billion. Also that year, IBM began announcing several major partnerships with other companies,

including Apple
Inc., Twitter, Facebook, Tencent, Cisco, UnderArmour, Box, Microsoft, VMware, CSC, Mac y's, Sesame Workshop, the parent company of Sesame Street, and Salesforce.com.

In 2015, IBM announced three major acquisitions: Merge Healthcare for $1 billion, data storage vendor Cleversafe, and all digital assets from The Weather Company, including Weather.com and the Weather Channel mobile app. Also that year, IBM employees created the film *A Boy and His Atom*, which was the first molecule movie to tell a story. In 2016, IBM acquired video conferencing service Ustream and formed a new cloud video unit. In April 2016, it posted a 14-year low in quarterly sales. The following month, Groupon sued IBM accusing it of patent infringement, two months after IBM accused Groupon of patent infringement in a separate lawsuit.

In 2015, IBM bought the digital part of The Weather Company;, Truven Health Analytics for $2.6 billion in 2016, and in October 2018, IBM announced its intention to acquire Red Hat for $34 billion, which was completed on July 9, 2019.

IBM announced in October 2020 that it is splitting itself into two separate public companies. IBM's future focus will be on high-margin cloud computing and artificial intelligence, built on the foundation of the 2019 Red Hat acquisition. The new company NewCo, now known as Kyndryl, created from IBM Global Technology Services Managed Infrastructure Services unit, will have 90,000 employees, 4,600 clients in 115 countries, with a backlog of $60 billion. IBM's spin off will be greater than any of its previous divestitures and welcomed by investors. In January 2021, IBM appointed Martin Schroeter, who had been IBM's CFO from 2014 through the end of 2017, as CEO of NewCo, which is expected to be finalized by the end of 2021 as an independent, publicly traded company with a new name.

IBM has regularly reinvented itself by selling off low margin assets while shifting its focus to higher-value, more profitable markets. Examples include:

- 1991: Spun off its printer and keyboard manufacturing division, the IBM Information Products Corporation, to Lexmark
- 2005 and 2014, respectively: Sold its personal computer (ThinkPad/ThinkCentre) and x86-based server businesses to Lenovo
- 2015: IBM adopted a "fabless" model with semiconductors design, while offloading manufacturing to GlobalFoundries

- 2002–2020: Acquired PwC Consulting (2002), SPSS (2009), The Weather Company (2016), Red Hat (2019), and European cloud consultant Nordcloud (2020)
- Planned for late 2021: A $19 billion spin-off of managed infrastructure services unit into a new public company (will be named Kyndryl) In 2021, IBM announced the acquisition of US based enterprise software company Turbonomic for $1.5 Billion.

Python (Programming Language)

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

History

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's *Benevolent Dictator For Life*, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the `2to3` utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

Syntax and Semantics

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

Statements and control flow

Python's statements include (among others):

- The assignment statement, using a single equals sign `=`.
- The `if` statement, which conditionally executes a block of code, along with `else` and `elif` (a contraction of else-if).
- The `for` statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The `while` statement, which executes a block of code as long as its condition is true.
- The `try` statement, which allows exceptions raised in its attached code block to be caught and handled by `except` clauses; it also ensures that clean-up code in a `finally` block will always be run regardless of how the block exits.
- The `raise` statement, used to raise a specified exception or re-raise a caught exception.

- The `class` statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The `def` statement, which defines a function or method.
- The `with` statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII)-like behavior and replaces a common try/finally idiom.
- The `break` statement, exits from a loop.
- The `continue` statement, skips this iteration and continues with the next item.
- The `del` statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.
- The `pass` statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The `assert` statement, used during debugging to check for conditions that should apply.
- The `yield` statement, which returns a value from a generator function and `yield` is also an operator. This form is used to implement coroutines.
- The `return` statement, used to return a value from a function.
- The `import` statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (`=`) operates by binding a name as a reference to a separate, dynamically-allocated object. Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to *some* object, which will have a type. This is referred to as dynamic typing and is contrasted with statically-typed programming languages, where each variable may only contain values of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutine-like functionality is

provided, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

Expressions

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) `//` and floating-point `/` division. Python also uses the `**` operator for exponentiation.
- From Python 3.5, the new `@` infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.
- From Python 3.8, the syntax `:=`, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.
- In Python, `==` compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the `equals()` method.) Python's `is` operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example `a <= b <= c`.
- Python uses the words `and`, `or`, `not` for its boolean operators rather than the symbolic `&&`, `||`, `!` used in Java and C.
- Python has a type of expression termed a *list comprehension* as well as a more general expression termed a *generator expression*.
- Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.
- Conditional expressions in Python are written as `x` **if** `c` **else** `y` (different in order of operands from the `c ? x : y` operator common to many other languages).

- Python makes a distinction between lists and tuples. Lists are written as `[1, 2, 3]`, are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as `(1, 2, 3)`, are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The `+` operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable `t` initially equal to `(1, 2, 3)`, executing `t = t + (4, 5)` first evaluates `t + (4, 5)`, which yields `(1, 2, 3, 4, 5)`, which is then assigned back to `t`, thereby effectively "modifying the contents" of `t`, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.
- Python features *sequence unpacking* wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in an identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an *iterable* object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through and will iterate through it, assigning each of the produced values to the corresponding expression on the left.
- Python has a "string format" operator `%`. This functions analogously to `printf` format strings in C, e.g. `"spam=**%s** eggs=**%d**" % ("blah", 2)` evaluates to `"spam=blah eggs=2"`. In Python 3 and 2.6+, this was supplemented by the `format()` method of the `str` class, e.g. `"spam=**{0}** eggs=**{1}**".format("blah", 2)`. Python 3.6 added "f-strings": `blah = "blah"; eggs = 2; f'spam=**{**blah**}** eggs=**{**eggs**}**'`.[93]
- Strings in Python can be concatenated, by "adding" them (same operator as for adding integers and floats). E.g. `"spam" + "eggs"` returns `"spameggs"`. Even if your strings contain numbers, they are still added as strings rather than integers. E.g. `"2" + "2"` returns `"22"`.
- Python has various kinds of string literals:

  - Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".
  - Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.
  - Raw string varieties, denoted by prefixing the string literal with an r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.
- Python has array index and array slicing expressions on lists, denoted as a[key], a[start:stop] or a[start:stop:step]. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the *start* index up to, but not including, the *stop* index. The third slice parameter, called *step* or *stride*, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

- List comprehensions vs. for-loops
- Conditional expressions vs. if blocks
- The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an

assignment operator `=` for an equality operator `==` in conditions: `if (c = 1) { ... }` is syntactically valid (but probably unintended) C code but `if c = 1: ...` causes a syntax error in Python.

Methods

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter to access instance data, in contrast to the implicit self (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Apart from this, Python also provides methods, often called *dunder methods* (due to their names beginning and ending with double-underscores), to allow user-defined classes to modify how they are handled by native operations such as length, comparison, in arithmetic operations, type conversion, and many more.

Typing

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, `SpamClass()` or `EggsClass()`), and the classes are instances of the metaclass `type` (itself an instance of itself), allowing metaprogramming and reflection.

Before version 3.0, Python had two kinds of classes: *old-style* and *new-style*. The syntax of both styles is the same, the difference being whether the class `object` is inherited from, directly or indirectly (all new-style classes inherit from `object` and are instances of `type`). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named *mypy* supports compile-time type checking.

Arithmetic operations

Python has the usual symbols for arithmetic operators (`+`, `-`, `*`, `/`), the floor division operator `//` and the modulo operation `%` (where the remainder can be negative, e.g. `4 % -3 == -2`). It also has `**` for exponentiation, e.g. `5**3 == 125` and `9**0.5 == 3.0`, and a matrix-multiplication operator `@` . These operators work like in traditional math; with the same precedence rules, the operators infix (`+` and `-` can also be unary to represent positive and negative numbers respectively).

The division between integers produces floating-point results. The behavior of division has changed significantly over time:

- Current Python (i.e. since 3.0) changed `/` to always be floating-point division, e.g. `5/2 == 2.5`.
- Python 2.2 changed integer division to round towards negative infinity, e.g. `7/3 == 2` and `-7/3 == -3`. The floor division `//` operator was introduced. So `7//3 == 2`, `-7//3 == -3`, `7.5//3 == 2.0` and `-7.5//3 == -3.0`. Adding **`from __future__ import`** division causes a module to use Python 3.0 rules for division (see next).
- Python 2.1 and earlier used C's division behavior. The `/` operator is integer division if both operands are integers, and floating-point division otherwise. Integer division rounds towards 0, e.g. `7/3 == 2` and `-7/3 == -2`.

In Python terms, `/` is *true division* (or simply *division*), and `//` is *floor division*. `/` before version 3.0 is *classic division*.

Rounding towards negative infinity, though different from most languages, adds consistency. For instance, it means that the equation `(a + b)//b == a//b + 1` is always true. It also means that

the equation `b*(a//b) + a%b == a` is valid for both positive and negative values of `a`. However, maintaining the validity of this equation means that while the result of `a%b` is, as expected, in the half-open interval [0, *b*), where `b` is a positive integer, it has to lie in the interval (*b*, 0] when `b` is negative.

Python provides a `round` function for rounding a float to the nearest integer. For tie-breaking, Python 3 uses round to even: `round(1.5)` and `round(2.5)` both produce `2`. Versions before 3 used round-away-from-zero: `round(0.5)` is `1.0`, `round(-0.5)` is `−1.0`.

Python allows boolean expressions with multiple equality relations in a manner that is consistent with general use in mathematics. For example, the expression `a < b < c` tests whether `a` is less than `b` and `b` is less than `c`. C-derived languages interpret this expression differently: in C, the expression would first evaluate `a < b`, resulting in 0 or 1, and that result would then be compared with `c`.

Python uses arbitrary-precision arithmetic for all integer operations. The `Decimal` type/class in the `decimal` module provides decimal floating-point numbers to a pre-defined arbitrary precision and several rounding modes. The `Fraction` class in the `fractions` module provides arbitrary precision for rational numbers.

Due to Python's extensive mathematics library, and the third-party library NumPy that further extends the native capabilities, it is frequently used as a scientific scripting language to aid in problems such as numerical data processing and manipulation.

Programming Examples

Hello world program:

```
print('Hello, world!')
```

Program to calculate the factorial of a positive integer:

```
n = int(input('Type a number, and its factorial will be
printed: '))

if n < 0:
    raise ValueError('You must enter a non negative integer')

factorial = 1
for i in range(2, n + 1):
    factorial *= i

print(factorial)
```

Pandas

History of development

In 2008, *pandas* development began at AQR Capital Management. By the end of 2009 it had been open sourced, and is actively supported today by a community of like-minded individuals around the world who contribute their valuable time and energy to help make open source *pandas* possible. Thank you to all of our contributors.

Since 2015, *pandas* is a NumFOCUS sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project.

Timeline

- **2008**: Development of *pandas* started
- **2009**: *pandas* becomes open source
- **2012**: First edition of *Python for Data Analysis* is published
- **2015**: *pandas* becomes a NumFOCUS sponsored project
- **2018**: First in-person core developer sprint

Library Highlights

- A fast and efficient **DataFrame** object for data manipulation with integrated indexing;
- Tools for **reading and writing data** between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible **reshaping** and pivoting of data sets;
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets;
- Columns can be inserted and deleted from data structures for **size mutability**;
- Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;
- High performance **merging and joining** of data sets;
- **Hierarchical axis indexing** provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;

- **Time series**-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
- Highly **optimized for performance**, with critical code paths written in Cython or C.
- Python with *pandas* is in use in a wide variety of **academic and commercial** domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

Mission

pandas aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

Vision

A world where data analytics and manipulation software is:

- Accessible to everyone
- Free for users to use and modify
- Flexible
- Powerful
- Easy to use
- Fast

Values

Is in the core of *pandas* to be respectful and welcoming with everybody, users, contributors and the broader community. Regardless of level of experience, gender, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, religion, or nationality.

INTRODUCTION TO MACHINE LEARNING

What is machine learning?

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Why is machine learning important?

Machine learning is important because it gives enterprises a view of trends in customer behaviour and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

What are the different types of machine learning?

Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are four basic approaches: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. The type of algorithm data scientists chooses to use depends on what type of data they want to predict.

- **Supervised learning**: In this type of machine learning, data scientists supply algorithms with labelled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.
- **Unsupervised learning**: This type of machine learning involves algorithms that train on unlabelled data. The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.
- **Semi-supervised learning**: This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labelled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.

- **Reinforcement learning**: Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.

How does supervised machine learning work?

Supervised machine learning requires the data scientist to train the algorithm with both labelled inputs and desired outputs. Supervised learning algorithms are good for the following tasks:

- **Binary classification:** Dividing data into two categories.
- **Multi-class classification:** Choosing between more than two types of answers.
- **Regression modelling:** Predicting continuous values.
- **Ensembling:** Combining the predictions of multiple machine learning models to produce an accurate prediction.

What is Supervised Learning?

Supervised learning, as we know is one of the most common types of ML learning methodology. The concept of this learning focuses on labelling of training data.

Unlike unsupervised learning, the model first learns from the given training data. The training data contains different patterns, which the model will learn. In other words, the outputs are already available. But, for a collection of data, various outputs are there.

Supervising here means helping out the model to predict the right things. The data will contain inputs with corresponding outputs. This has hidden patterns in it. The algorithm will learn these patterns and will try to apply the same knowledge to unseen data. The aim is to predict future values.

Mathematically, supervised learning can be shown as a linear function, i.e., y=f(x).

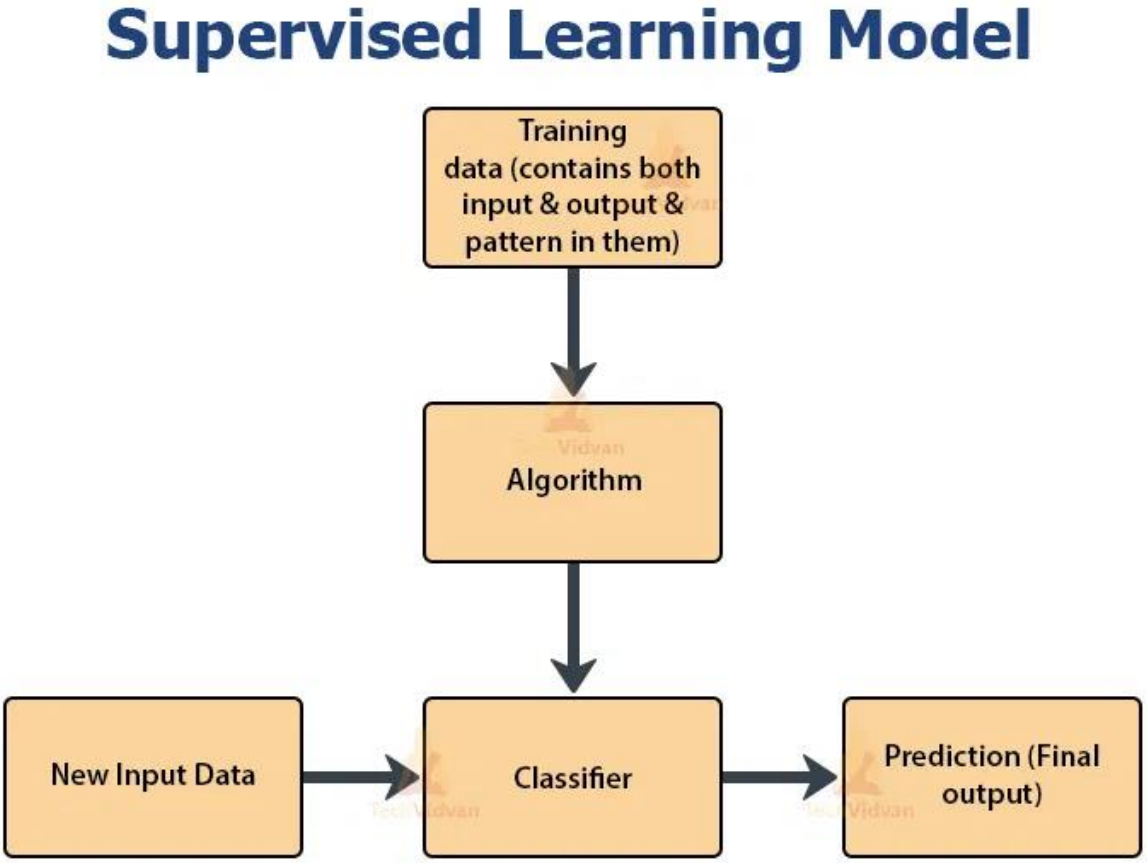


Fig 5.1 – Supervised Learning Model

Advantages of Supervised Learning

1. This type of learning is easy to understand. It is the most common type of learning method. For, learning ML, people should start by practicing supervised learning.

2. The training data is only necessary for training the model. Since it is large it occupies a lot of **space**. But it's removed from the memory as it is of no importance after training is complete.

3. We would already know the number of classes in the data.

4. After training, the model would for which specific data the output needs to be predicted as all the data in the collection is not important.

5. It is very helpful in solving real-world computational problems.

6. This learning method can also help the model to learn from previous experiences and to improve its accuracy in prediction.

Disadvantages of Supervised Machine Learning

1. Its performances are limited to the fact that it can't handle complex problems in ML.

2. It cannot create labels of its own. This means that, it cannot discover data on its own like **unsupervised** learning.

3. If we enter new data, it has to be from any of the given classes only. If you enter watermelon data in a collection of apples and oranges, it might classify the watermelon into one of these classes, which won't be right.

4. It would require a good computer with quality processors to train a supervised learning-based model. It requires high computation power, which not all PC's might have.

Supervised Learning Algorithms

There are various types of ML algorithms, which we will now study.

Linear Regression in ML

It is an ML algorithm, which includes modelling with the help of a dependent variable. As the name suggests, this is a linear model. The format of the projection for this model is **Y= ax+b**. Linear regression is an algorithm, which helps us to understand the relationship between two variables.

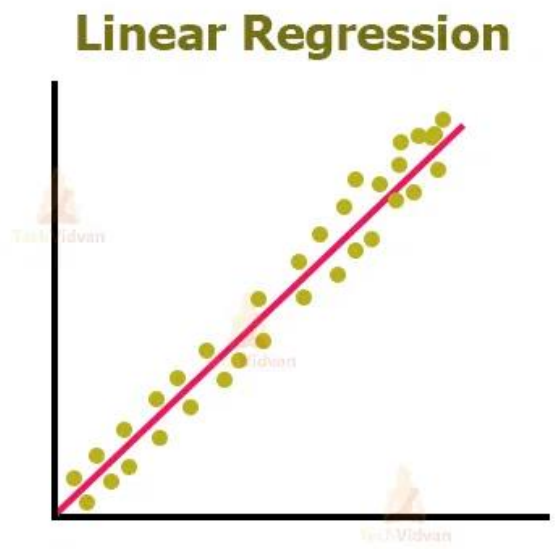


Fig 5.2 Linear Regression Graph

Logistics Regression in ML

This type of regression helps us to understand the relationship between one binary dependent variable and an independent variable. This is not a linear relation; it is actually a logarithmic relation.

It is shown as **y = ln(P/(1-P)).**
The graph for this is S-shaped.

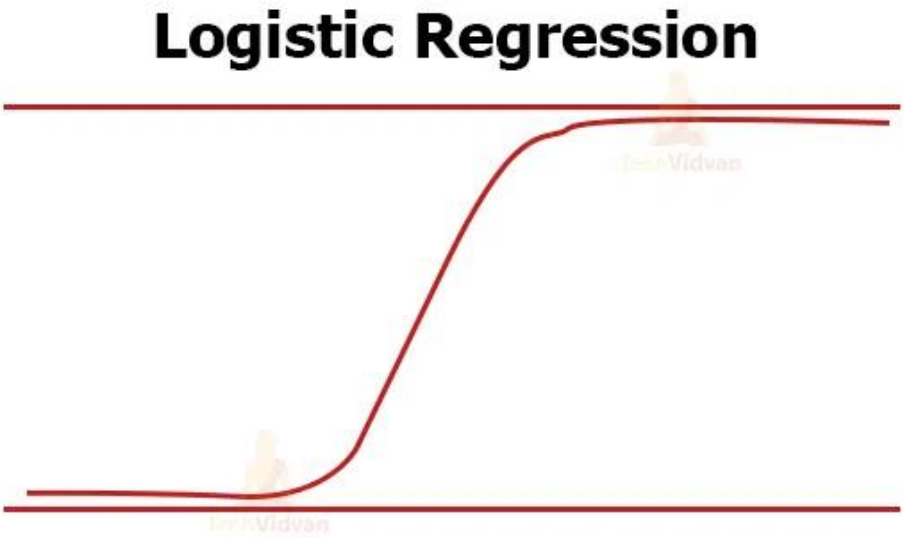


Fig 5.3 Logistic Regression Graph

Decision Trees in ML

Decision trees are **binary** trees, which help in the classification, which is a type of supervised learning approach. It might have higher complexity depending on the number of leaves and nodes the tree has. This tree is useful in 'yes or no' and 'if and then' situations.

K-nearest neighbours in ML

This method helps to find in which class does a point belongs using distance. Here, **k** is the number of points to measure with. If you choose k=3, then the point which you want to classify will be measured with three of its nearest neighbours. The point gets classified in the class of which the majority of neighbours are part of.

Random Forests in ML

Random forest is a huge collection of **decision** trees. So, for a situation, there are many possible outcomes that the random forest helps us to see. It is different from a decision tree (as decision trees are always binary and they a single unit) as it has multiple trees.
This algorithm helps to find new patterns and possibilities for anything as the collection of trees helps in analysing data in many ways. It has a more complex algorithm than a decision tree.

Hence, it would consume a lot more computational power.

SVM in ML

SVM or Support Vector Machines are also very popular algorithms used in supervised learning. They help to classify and analyse the data with the help of a hyperplane. The hyperplane is a line or a plane, which divides the data points into two separate categories. The aim is to find an **optimal** plane, which divides both datapoints.
By maximizing the margin of the hyperplane, we increase the distance between the data points on either side. This is done up to the point where datapoints are distinct from each other. For determining the margins of the hyperplane, support vectors (data points that appear to lie on the plane or are close to it) are necessary.

ANN

ANN or Artificial Neural Networks come under modern-day deep learning. In this, we study algorithms that involve neural networks.

Neural networks are very helpful in processing complex problems and in finding more hidden patterns efficiently. They have three layers namely the **input layer**, **hidden layer**, and **output layer**.

The input layer takes the data, the hidden layer has multiple patterns and layers that process the data and analyse the best result. Then comes the output layer.

This concept was created by keeping in mind the functioning of the real neuron. So, that is why it is called an artificial neural network. It is quite significant in modern research and has great potential ahead.

Types of Supervised Learning in ML

There are only two types of supervised learning approaches. Every algorithm comes under these two methodologies.

1. Classification

Classification is a technique with which we can **affiliate** data under certain labels. We can train algorithms and models to classify data under different classes.

For example, if you have data of fruits consisting of apples and oranges, your model will learn from the data about the specifications of the fruits.

The next time you enter new data, the model will compare your data and then classify it under either class. There are two types of classifications:

- **Binomial Classification**

It classifies data under **two** classes. This happens in decision trees and in simpler data where there are only two types of data.

- **Multi-Class Classification**

It classifies data under more than two classes. This means there is a lot of data and many **possibilities**. This happens in random forests.

2. Regression

Regression helps us to understand the relationship between various data points and helps us to find hidden patterns among the data.

We have both dependent and independent variables in this case which can help us to understand the relation with the help of **graphical** representations.

There are many types of regression, but we know the two main ones:

- Linear Regression
- Logistics Regression

Supervised Machine Learning Applications

- It is useful for the **prediction of stock markets**. It can accurately predict the prices of the stock data by analysing the pattern of previous data. We can make use of various algorithms for predicting the stock market.

  For example, let's take neural networks. In neural networks also, let us take LSTM or long short-term memory.

  This algorithm is actually a type of **RNN**. But, while processing the data, it will remember the significant output and it will forget the unimportant outcome. This algorithm trains under the training data. For various patterns, the neural network trains itself. It ensures more accuracy by going back and forth and removing unimportant data.
- It can find patterns in **biometrics**. It is great for adding fingerprint sensors. Also, eye scanners, eye-lobe scanners come under biometrics. This contains the entire individual's necessary information. This mainly comes under **security** providing.

  Most smartphones today have this as facial recognition technology.

- **Speech Recognition** is one of the major applications of supervised learning. This too can come under security, especially high-level security where you have to undergo several rounds of screening.

  Also, we can use it on our smartphones. The voice assistant technology uses this.

  For example, **SIRI**, **Alexa**, **Google Assistant**. These use the speech recognition algorithm to remember your voice and match it when you speak. They can assist you with anything in your smartphone.

  Also, these assistants come as separate devices too; you can connect your other electronics with Bluetooth if you want to activate them using the assistant.

- Supervised learning also helps in **search history optimization**. If you search for something once, the next time you search it, the browser will try to provide much better results. It uses your search history as input data and improves its accuracy of searching with that.

  For example, **google search**.
- **Spam detection** is also a very important application. If there are any spam emails, it can help you to **block** such emails by classifying them as spam. It may even block them from sight. Its main purpose is to block fake things.
- **Object detection** is also now becoming a thing with research. Technologies such as raspberry pi are also working on this. It also uses **computer vision**.

Use Cases of Supervised Learning

- **Security**

  One of the primary concerns of today's era is proper data security. On top of that, there is more fear when it comes to the security of monetary credentials like a **credit card**, **bank account number**, etc.

  With ML techniques, various big IT industries are now investing in newer and more powerful algorithms to tackle problems like anomaly detection, fraud detection in credit cards, etc. They are also helping by creating algorithms for spam filtering, blocking malicious links, emails, etc.

  Various supervised learning methods and algorithms are very helpful in certain cases.

- **Marketing and business**

  There is a lot when it comes to marketing and business. Especially, when the growth of internet is exponential, the number of users gradually increases. This is a clear marketing opportunity for all companies globally to advertise their products.

  We have major online retailers like Amazon, Walmart, Flipkart etc.

  This is just an example of how online marketing works; hence, we have taken online retailers as an example.

So, companies like amazon have a recommendation system. This system helps to analyse the user preference using the search history. Based on that it will start to pop miniature ad feeds of

the product on whichever webpage you are browsing. This is online advertising.

Supervised Learning is useful in recommendation systems to analyse user preference. There are many other ML concepts like sentiment analysis, time series analysis, etc., which are used in online marketing.

How does unsupervised machine learning work?

Unsupervised machine learning algorithms do not require data to be labelled. They sift through unlabelled data to look for patterns that can be used to group data points into subsets. Most types of deep learning, including neural networks, are unsupervised algorithms. Unsupervised learning algorithms are good for the following tasks:

- **Clustering:** Splitting the dataset into groups based on similarity.
- **Anomaly detection:** Identifying unusual data points in a data set.
- **Association mining:** Identifying sets of items in a data set that frequently occur together.
- **Dimensionality reduction:** Reducing the number of variables in a data set.

Advantages of Unsupervised Learning

There are some reasons why we sometimes choose unsupervised learning in place of supervised learning. Here are some of the advantages:

- Labelling of data demands a lot of manual work and expenses. Unsupervised learning solves the problem by learning the data and classifying it without any labels.
- The labels can be added after the data has been classified which is much easier.
- It is very helpful in finding patterns in data, which are not possible to find using normal methods.
- Dimensionality reduction can be easily accomplished using unsupervised learning.

- This is the perfect tool for data scientists, as unsupervised learning can help to understand raw data.
- We can also find up to what degree the data are similar. This can be accomplished with **probabilistic** methods.
- This type of learning is similar to human intelligence in some way as the model learns slowly and then calculates the result.

Disadvantages of Unsupervised Learning

Now, let's have a look at some cons of unsupervised learning algorithm:

- The result might be less accurate as we do not have any input data to train from.
- The model is learning from raw data without any prior knowledge.
- It is also a time-consuming process. The learning phase of the algorithm might take a lot of time, as it analyses and calculates all possibilities.
- For some projects involving live data, it might require continuous feeding of data to the model, which will result in both inaccurate and time-consuming results.
- The more the features, the more the **complexity** increases.

Algorithms related to Unsupervised Machine Learning

Now let's look at some algorithms which are based on unsupervised learning.

As we discussed, the algorithms and applications might be limited, but they are of extreme significance.

K-means Clustering in ML

K means is a clustering algorithm type. It is an iterative clustering approach. This algorithm states that similar data points should be in close proximity. For this we will select the value of **k**. The value of k is the number of data points. Now, select centroids in the data set. The centroids will act as data accumulation areas.
Take each centroid and measure the distance of k datapoints. The once near the centroid will get clustered. For this, we use methods like Euclidean distance as measuring options.

KNN Clustering

KNN or K-nearest neighbour is also a clustering-based algorithm. This method is used for those datapoints which can be selected in any class or for those who don't have any class or cluster assigned. The algorithm starts with the selection of the point which we want to work on. Then we have to select the value of k. K will be the number of points around the selected points. These points can belong to **multiple clusters**.
Now, measure the distance of each point with the test point using Euclidean or Manhattan distance measuring techniques. Sort the results in ascending order.

The test point will end up in the cluster whose points were the closest to the test point.

Hierarchical Clustering in ML

In this, we form multiple clusters, which are distinct to each other, but the contents inside the cluster are highly similar to each other. For this, we would use the distance matrix for calculation purposes, and then for the visual representation of the clusters, a dendrogram would be formed.

The algorithm works in a specific way.

The algorithm would treat each observation as a separate cluster. Then it would find two most similar clusters and merge them. This step goes on iteratively until all the clusters merge together. The main result is the **dendrogram**. It would show the similarity between the clusters. There are some other methods of finding similarity as well like distance criteria and linkage criteria.
The process of merging the clusters is **agglomerative clustering.** These were some of the main algorithms or types of unsupervised learning that we have discussed now.

Applications of Unsupervised Machine Learning Algorithm

As stated in the above pages of the article, the applications for this learning are quite limited.

But still, we will look at the ones which are widely popular.

- It is mainly useful in fraud detection in credit cards.
- Useful for genome analysing.
- Useful in data pre-processing.

Better Learning Methods than Unsupervised Learning

Semi-supervised learning might be a good substitute for unsupervised learning. It is a combination of both supervised and unsupervised learnings. The main advantage of this type of learning is that it reduces the errors of both supervised and unsupervised learnings. For instance, it will only cluster the unlabelled data which is possible to cluster and the result will be classified automatically after being labelled. This consumes less **computational** power and is less time-consuming.

How does semi-supervised learning work?

Semi-supervised learning works by data scientists feeding a small amount of labelled training data to an algorithm. From this, the algorithm learns the dimensions of the data set, which it can then apply to new, unlabelled data. The performance of algorithms typically improves when they train on labelled data sets. But labelling data can be time consuming and expensive. Semi-supervised learning strikes a middle ground between the performance of supervised learning and the efficiency of unsupervised learning. Some areas where semi-supervised learning is used include:

- **Machine translation:** Teaching algorithms to translate language based on less than a full dictionary of words.
- **Fraud detection:** Identifying cases of fraud when you only have a few positive examples.
- **Labelling data:** Algorithms trained on small data sets can learn to apply data labels to larger sets automatically.

How does reinforcement learning work?

Reinforcement learning works by programming an algorithm with a distinct goal and a prescribed set of rules for accomplishing that goal. Data scientists also program the algorithm to seek positive rewards -- which it receives when it performs an action that is beneficial toward the ultimate goal -- and avoid punishments -- which it receives when it performs an action that gets it farther away from its ultimate goal. Reinforcement learning is often used in areas such as:

- **Robotics:** Robots can learn to perform tasks the physical world using this technique.
- **Video gameplay:** Reinforcement learning has been used to teach bots to play a number of video games.
- **Resource management:** Given finite resources and a defined goal, reinforcement learning can help enterprises plan out how to allocate resources.

Before we get started, we must know about how to pick a good machine learning algorithm for the given dataset. To intelligently pick an algorithm to use for a supervised learning task, we must consider the following factors:

1. Heterogeneity of Data:

Many algorithms like neural networks and support vector machines like them feature vectors to be homogeneous numeric and normalized. The algorithms that employ distance metrics are very sensitive to this, and hence if the data is heterogeneous, these methods should be the afterthought. Decision Trees can handle heterogeneous data very easily.

2. Redundancy of Data:

If the data contains redundant information, i.e., contain highly correlated values, then it's useless to use distance-based methods because of numerical instability. In this case, some sort of Regularization can be employed to the data to prevent this situation.

3. Dependent Features:

If there is some dependence between the feature vectors, then algorithms that monitor complex interactions like Neural Networks and Decision Trees fare better than other algorithms.

4. Bias-Variance Trade-off:

A learning algorithm is biased for a particular input x if, when trained on each of these data sets, it is systematically incorrect when predicting the correct output for x, whereas a learning algorithm has high variance for a particular input x if it predicts different output values when trained on different training sets. The prediction error of a learned classifier can be related to the sum of bias and variance of the learning algorithm, and neither can be high as they will make the prediction error to be high. A key feature of machine learning algorithms is that they are able to tune the balance between bias and variance automatically, or by manual tuning using bias parameters, and using such algorithms will resolve this situation.

5. Curse of Dimensionality:

If the problem has an input space that has a large number of dimensions, and the problem only

depends on a subspace of the input space with small dimensions, the machine learning algorithm can be confused by the huge number of dimensions and hence the variance of the algorithm can be high. In practice, if the data scientist can manually remove irrelevant features from the input data, this is likely to improve the accuracy of the learned function. In addition, there are many algorithms for feature selection that seeks to identify the relevant features and discard the irrelevant ones, for instance Principle Component Analysis for unsupervised learning. This reduces the dimensionality.

6. Overfitting:

The programmer should know that there is a possibility that the output values may constitute of an inherent noise which is the result of human or sensor errors. In this case, the algorithm must not attempt to infer the function that exactly matches all the data. Being too careful in fitting the data can cause overfitting, after which the model will answer perfectly for all training examples but will have a very high error for unseen samples. A practical way of preventing this is stopping the learning process prematurely, as well as applying filters to the data in the pre-learning phase to remove noises.

Only after considering all these factors can, we pick a supervised learning algorithm that works for the dataset we are working on. For example, if we were working with a dataset consisting of heterogeneous data, then decision trees would fare better than other algorithms. If the input space of the dataset we were working on had 1000 dimensions, then it’s better to first perform PCA on the data before using a supervised learning algorithm on it.

Who's using machine learning and what's it used for?

Today, machine learning is used in a wide range of applications. Perhaps one of the most well-known examples of machine learning in action is the recommendation engine that powers Facebook's news feed.

Facebook uses machine learning to personalize how each member's feed is delivered. If a member frequently stops to read a particular group's posts, the recommendation engine will start to show more of that group's activity earlier in the feed.

Behind the scenes, the engine is attempting to reinforce known patterns in the member's online behaviour. Should the member change patterns and fail to read posts from that group in the coming weeks, the news feed will adjust accordingly.

In addition to recommendation engines, other uses for machine learning include the following:

- **Customer relationship management.** CRM software can use machine learning models to analyse email and prompt sales team members to respond to the most important messages first. More advanced systems can even recommend potentially effective responses.
- **Business intelligence.** BI and analytics vendors use machine learning in their software to identify potentially important data points, patterns of data points and anomalies.
- **Human resource information systems.** HRIS systems can use machine learning models to filter through applications and identify the best candidates for an open position.
- **Self-driving cars.** Machine learning algorithms can even make it possible for a semi-autonomous car to recognize a partially visible object and alert the driver.
- **Virtual assistants.** Smart assistants typically combine supervised and unsupervised machine learning models to interpret natural speech and supply context.

What are the advantages and disadvantages of machine learning?

Machine learning has seen use cases ranging from predicting customer behaviour to forming the operating system for self-driving cars.

When it comes to advantages, machine learning can help enterprises understand their customers at a deeper level. By collecting customer data and correlating it with behaviours over time, machine learning algorithms can learn associations and help teams tailor product development and marketing initiatives to customer demand.

Some companies use machine learning as a primary driver in their business models. Uber, for example, uses algorithms to match drivers with riders. Google uses machine learning to surface the ride advertisements in searches.

But machine learning comes with disadvantages. First and foremost, it can be expensive. Machine learning projects are typically driven by data scientists, who command high salaries. These projects also require software infrastructure that can be expensive.

There is also the problem of machine learning bias. Algorithms trained on data sets that exclude certain populations or contain errors can lead to inaccurate models of the world that, at best, fail and, at worst, are discriminatory. When an enterprise bases core business processes on biased models it can run into regulatory and reputational harm.

How to choose the right machine learning model?

The process of choosing the right machine learning model to solve a problem can be time consuming if not approached strategically.

Step 1: Align the problem with potential data inputs that should be considered for the solution. This step requires help from data scientists and experts who have a deep understanding of the problem.

Step 2: Collect data, format it and label the data if necessary. This step is typically led by data scientists, with help from data wranglers.

Step 3: Chose which algorithm(s) to use and test to see how well they perform. This step is usually carried out by data scientists.

Step 4: Continue to fine tune outputs until they reach an acceptable level of accuracy. This step is usually carried out by data scientists with feedback from experts who have a deep understanding of the problem.

Importance of human interpretable machine learning

Explaining how a specific ML model works can be challenging when the model is complex. There are some vertical industries where data scientists have to use simple machine learning models because it's important for the business to explain how every decision was made. This is especially true in industries with heavy compliance burdens such as banking and insurance.

Complex models can produce accurate predictions, but explaining to a lay person how an output was determined can be difficult.

What is the future of machine learning?

While machine learning algorithms have been around for decades, they've attained new popularity as artificial intelligence has grown in prominence. Deep learning models, in particular, power today's most advanced AI applications.

Machine learning platforms are among enterprise technology's most competitive realms, with most major vendors, including Amazon, Google, Microsoft, IBM and others, racing to sign customers up for platform services that cover the spectrum of machine learning activities, including data collection, data preparation, data classification, model building, training and application deployment.

As machine learning continues to increase in importance to business operations and AI becomes more practical in enterprise settings, the machine learning platform wars will only intensify.

Continued research into deep learning and AI is increasingly focused on developing more general applications. Today's AI models require extensive training in order to produce an algorithm that is highly optimized to perform one task. But some researchers are exploring ways to make models more flexible and are seeking techniques that allow a machine to apply context learned from one task to future, different tasks.

PROJECT (Rock, Paper & Scissors Recogniser)

Introduction:

The basis of this project was to experiment with Deep Learning and Image Classification with the aim of building a simple yet fun iteration of the infamous Rock Paper Scissors game.

While building any sort of Deep Learning application there are **3 major steps:**

1. Collecting and Processing Data

2. Building a suitable AI Model

3. Deployment for use

Collecting our Data:

The basis of any Deep Learning model is DATA. Any Machine Learning Engineer would agree that in ML the data is far more crucial than the algorithm itself. We need to collect images for the symbols Rock, Paper and Scissor. Instead of downloading somebody else's data and training on it, I made my own dataset and encourage you to build your own too. Later on, try changing the data and re-training the model to see the grave impact data has on a Deep Learning model.

Why this?

It is very helpful for creating games or object detection projects. So, we gathered some dataset sample from the user which is used to understand the gesture of the hand. Implemented a hand gesture recognition algorithm. Initially, the skin was segmented to separate it from the background. I have trained a model to recognize my hand signs when it is inside the box, so when the model predicts my hand signs. The application is designed in a way that the model can predict the gesture and calculate the proximate accuracy.

Why this project?

We made this because it is very helpful for creating various games or object detection projects. So, we gathered some dataset sample from the user which is used to understand the gesture of the hand. Implemented a hand gesture recognition algorithm. Initially, the skin was segmented to separate it from the background. I have trained a model to recognize my hand signs when it is inside the box, so when the model predicts my hand signs. The application is designed in a way that the model can predict the gesture and calculate the proximate accuracy.

The basis of this project was to experiment with Deep Learning and Image Classification with the aim of building a simple yet fun iteration of the infamous Rock Paper Scissors game.

Method of approach to solve?

Here is a breakdown of our application in steps:

Step 1: Gather Data, for rock, paper scissor classes.
Step 2: (Optional) Visualize the Data.
Step 3: Pre-process Data and Split it.
Step 4: Prepare Our Model for Transfer Learning.
Step 5: Train Our Model.
Step 6: Check our Accuracy, Loss graphs & save the model.
Step 7: Test on Live Webcam Feed.

You should have TensorFlow 2.2, OpenCV 4x, and scikit-learn 0.23x installed in your system.

Library used:

- Os
- Cv2 (OpenCV)
- NumPy
- Matplotlib
- Time
- Sk_learn
- TensorFlow
- Random
- SciPy

How to install packages?

(Make sure you have installed python in your system)

Open command prompt and then use the command given below for installing the packages-

1. **OpenCV:**
   pip install OpenCV-python
2. **Matplotlib:**
   pip install matplotlib
3. **TensorFlow:**
   pip install TensorFlow
4. **Datetime:**
   pip install DateTime
5. **NumPy:**
   pip install NumPy
6. **Sk_learn:**
   pip install scikit-learn

I have used Python's OpenCV library for all camera related operations. So, *label* here refers to what class the image belongs too ie. RPS and depending on the label the image is saved in the appropriate directory. *ct* and *maxCt* refers to the start and final index to save images with. Remaining is standard OpenCV code to get webcam feed and save images to a directory. A **key point** to note is that all my images are of **300 x 300 dimensions**. After running this my directory tree looks like this.

Pre-processing our Data:

A computer understands numbers and we have images at our disposal. Thus, we will convert all the images into their respective vector representations. Also, our labels are yet to be generated and as established labels can't be text, so I've manually built **One-Hot-Encoded** representations for each class using the *shape_to_label* dictionary.

As we have saved our images in directories based on their classes the directory name serves as the label which is converted to One-Hot representation using the *shape_to_label* dictionary. Following that we iterate over the files in our system to access images, the *cv2.imread()* function returns a vector representation of image. We do some manual **Data Augmentation** by flipping the image and zooming into it. This increases our dataset size without having the need to take new photos. Data Augmentation is a crucial part to generating datasets. Finally, the images and labels are stored in separate NumPy

Building Our Model with Transfer Learning:

When it comes to working with image data there are many pre-trained models available that have been trained on datasets with thousands of labels that are available at our disposal thanks to TensorFlow and Keras distributions of these models via their applications api.

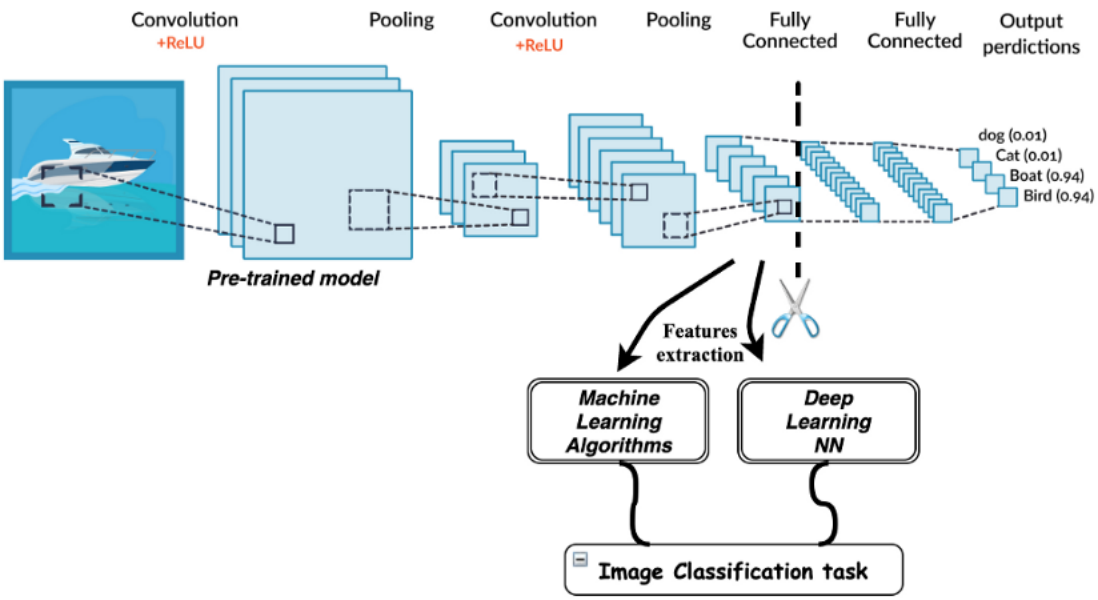


Fig 6.1 Transfer Learning

In a gist, Transfer learning takes a pre-trained model and does not include its final layers that make the final prediction thereby leaving us with the powerful part of the model that can

distinguish features in images for this case and pass this information to our own Dense Neural Network.

However, using transfer learning can at many times make your progress a lot faster, in a sense you're avoiding re-*inventing the wheel*.

Some other popular pre-trained models:

- InceptionV3
- VGG16/19
- ResNet
- Mobile Net

Whenever we work with image data the use of Data Convolutional Neural Layers is almost a given. The transfer Learning model used here has these layers.

Key Points:

- As our images are 300x300 the input shape specified is the same. The additional 3 stands for the RGB layers, so this layer has enough neurons to process the entire image.
- We've used the Dense Net layer as our base/first layer followed by our own Dense Neural Network.
- I've set the trainable parameter to True which will retrain the weights of the Dense Net too.
- As we have 3 classes Rock-Paper-Scissor the final layer is a Dense layer with 3 neurons and SoftMax activation.
- This final layer returns the probability of the image belonging to a particular class among the 3 classes.

By this point, we have gathered our data, built and trained our model. The only part left is deployment using OpenCV.

OpenCV implementation:

The flow for this implementation is simple:

- Start webcam feed and read each frame
- Pass this frame to model for classification ie. predict class
- Make a random move by computer
- Calculate Score

So, we begin with loading our trained model. Next comes a stone age for-loop implementation that shows a countdown before beginning the prediction part of the program. After prediction the scores are updated based on the players moves.

We explicitly draw a target zone using *cv2.rectangle().* Only this part of the frame is passed to the model for predictions after it is pre-processed using the *prepImg()* function.

Some Glimpses of the Code

In [1]:
```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import time

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Dense,MaxPool2D,Dropout,Flatten,Conv2D,GlobalAveragePooling2D,Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

from random import choice,shuffle
from scipy import stats as st

from collections import deque
```

In [4]:

```
# Set the figure size
plt.figure(figsize=[30,20])

# Set the rows and columns
rows, cols = 4, 8

# Iterate for each class
for class_index, each_list in enumerate([rock, paper, scissor,nothing]):

    # Get 8 random indexes, since we will be showing 8 examples of each class.
    r = np.random.randint(no_of_samples, size=8);

    # Plot the examples
    for i, example_index in enumerate(r,1):
        plt.subplot(rows,cols,class_index*cols + i );plt.imshow(each_list[example_index][0][:,:,::-1]);plt.axis('off');
```

Result

We apply our method for predicting the gesture shown by the user. Our method produces reasonably accurate results when a good number of sample data is used to train the model. If the model gives wrong predictions, then their might me some issue in training or data samples.

CONCLUSION

After completing this project, I concluded that this project was the good opportunity to implement my information that I have learnt during my internship program. This project is more informative and more helpful for understanding the concept of the Machine Learning and Artificial Intelligence. This project helped me a lot for understanding basic concepts, and it is enough to implement my concept. I can further try much harder to make much more efficient and useful programmes that can benefit to other.

REFERENCES

- https://www.udacity.com/course/intro-to-machine-learning--ud120. (Accessed on 12th September 2021)
- http://statweb.stanford.edu/~tibs/ElemStatLearn/ (Accessed on 12th September 2021)
- https://en.wikipedia.org/wiki/Python_(programming_language) (Accessed on 14th September 2021)
- https://pandas.pydata.org/about/index.html (Accessed on 14th September 2021)
- https://en.wikipedia.org/wiki/IBM (Accessed on 16th September 2021)
- https://www.ibm.com/cloud/learn/what-is-artificial-intelligence (Accessed on 16th September 2021)