AI & ML for Mechanical Engineers

Course Code: ME (DE) - 22016

Final Year Bachelor of Technology (Choice Based Credit System)
Mechanical Engineering
(With Effect from Academic Year 2021-22)

Lecture notes

Part 1: Introduction to AI & ML

Part 2: Feature extraction, selection and classification

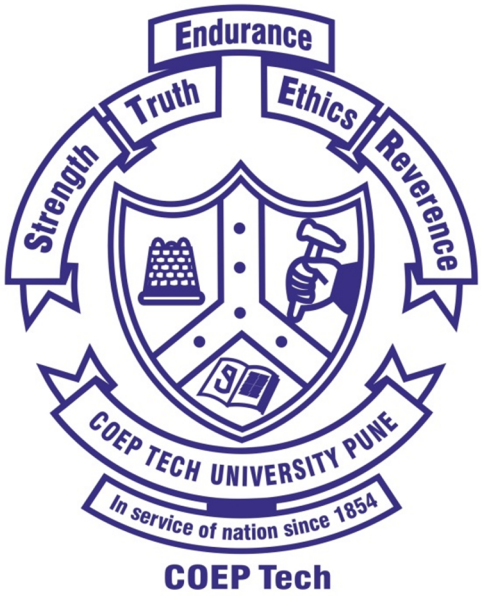
Part 3: Reinforcement and deep learning

Part 4: Development of ML model & its evaluation

by

Abhishek D. Patange, Ph.D.

Department of Mechanical Engineering



COEP Technological University Pune

AI & ML for Mechanical Engineers

Course Code: ME (DE) - 22016

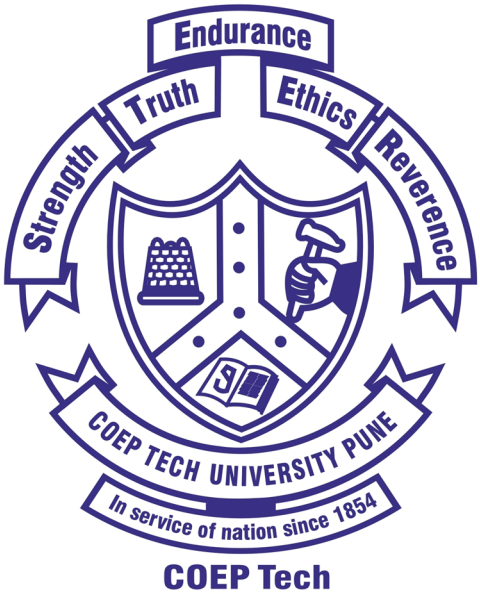
Part 1: Introduction to AI & ML

Final Year Bachelor of Technology (Choice Based Credit System)
Mechanical Engineering
(With Effect from Academic Year 2021-22)

Lecture notes

by

Abhishek D. Patange, Ph.D.
Department of Mechanical Engineering

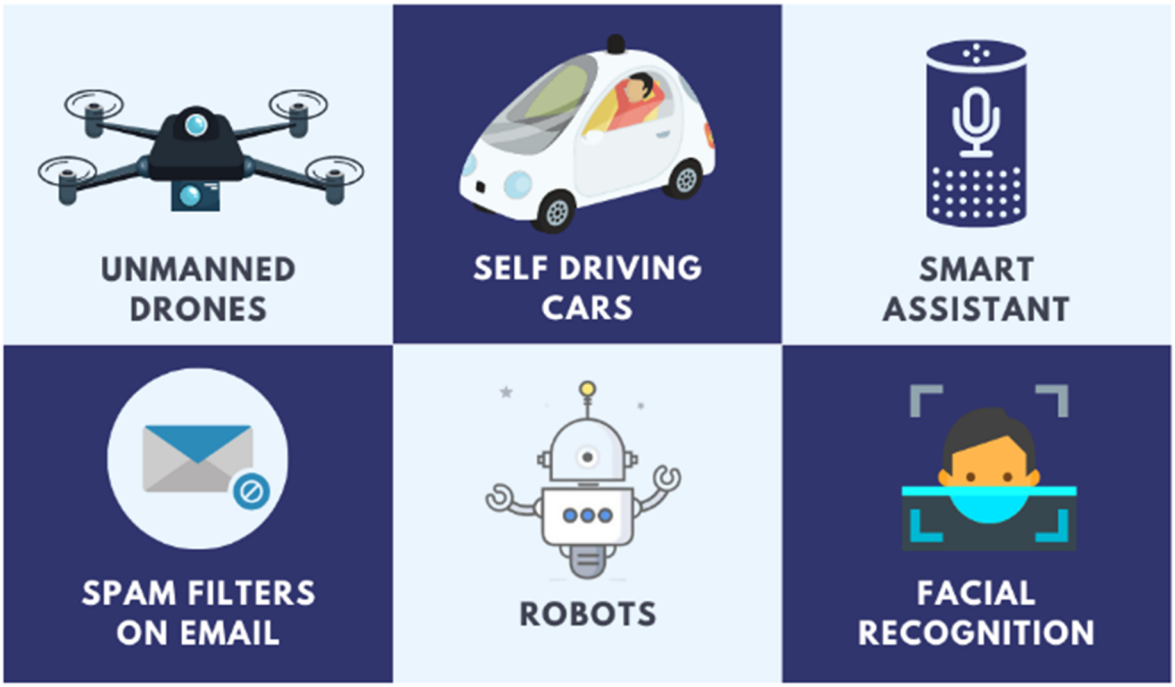


COEP Technological University Pune

Motivation behind AI approach and its history

The motivation behind AI (Artificial Intelligence) is to create machines or computer systems that can perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and natural language processing.

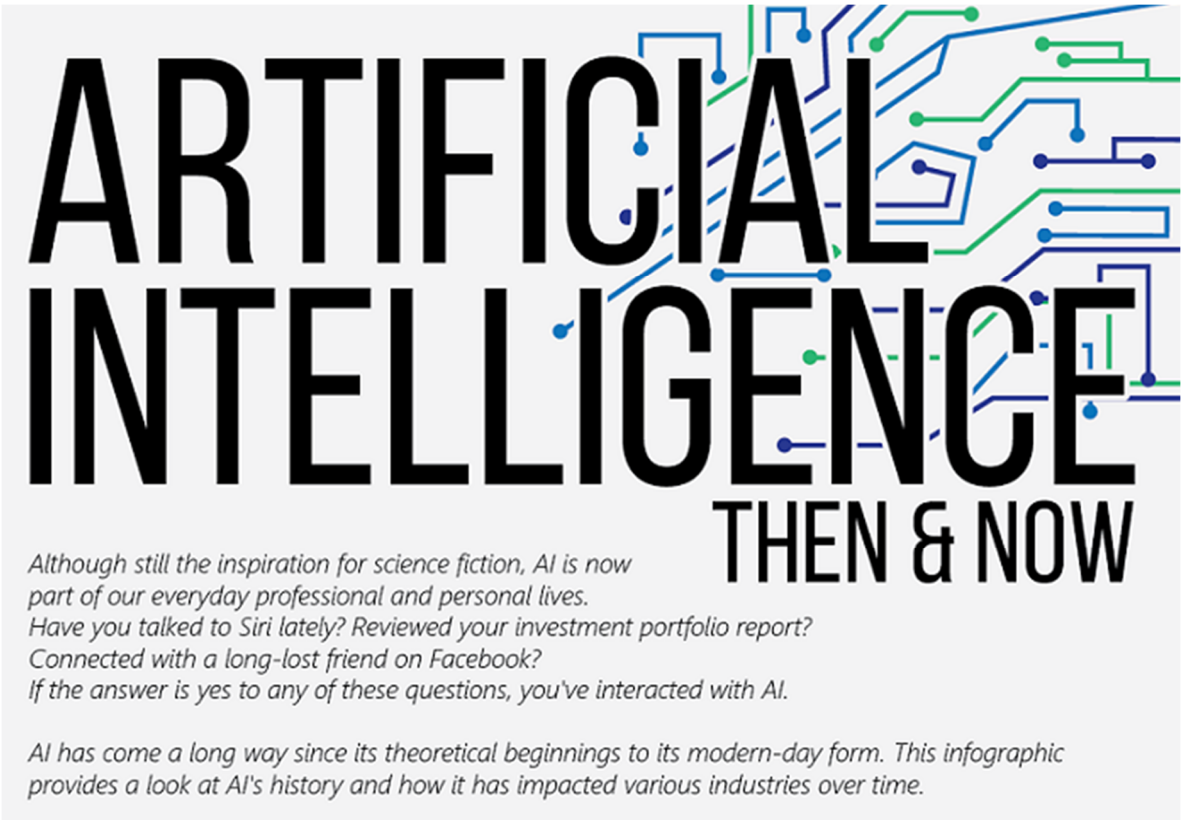
- The idea behind AI is to develop intelligent machines that can automate tasks, reduce human effort, and improve efficiency. By creating machines that can perform complex tasks, we can free up human resources to focus on more creative and innovative endeavors.
- Another motivation behind AI is to create machines that can learn and adapt to new situations. This ability to learn and adapt is what sets AI apart from traditional computer programs, which are designed to follow a predetermined set of rules.
- AI also has the potential to transform many industries, from healthcare to finance to manufacturing. For example, in healthcare, AI can be used to analyze medical data to identify patterns and trends that can help doctors make more accurate diagnoses and develop more effective treatments. In finance, AI can be used to analyze market trends and make predictions about future market movements.
- Overall, the motivation behind AI is to create intelligent machines that can automate tasks, learn and adapt to new situations, and transform industries by unlocking new levels of efficiency and productivity.

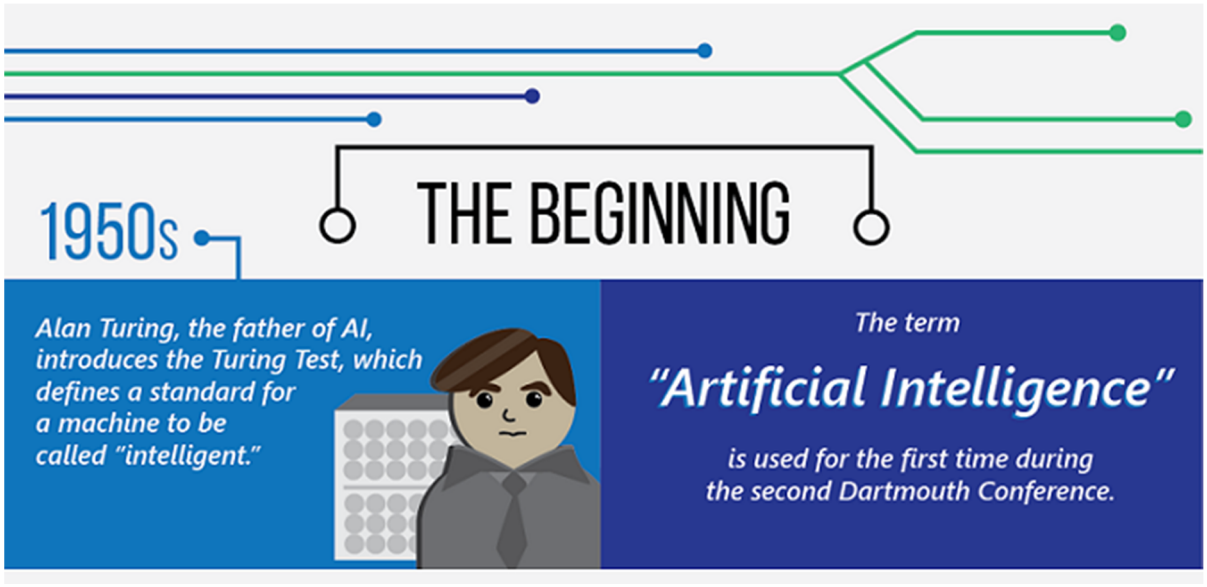


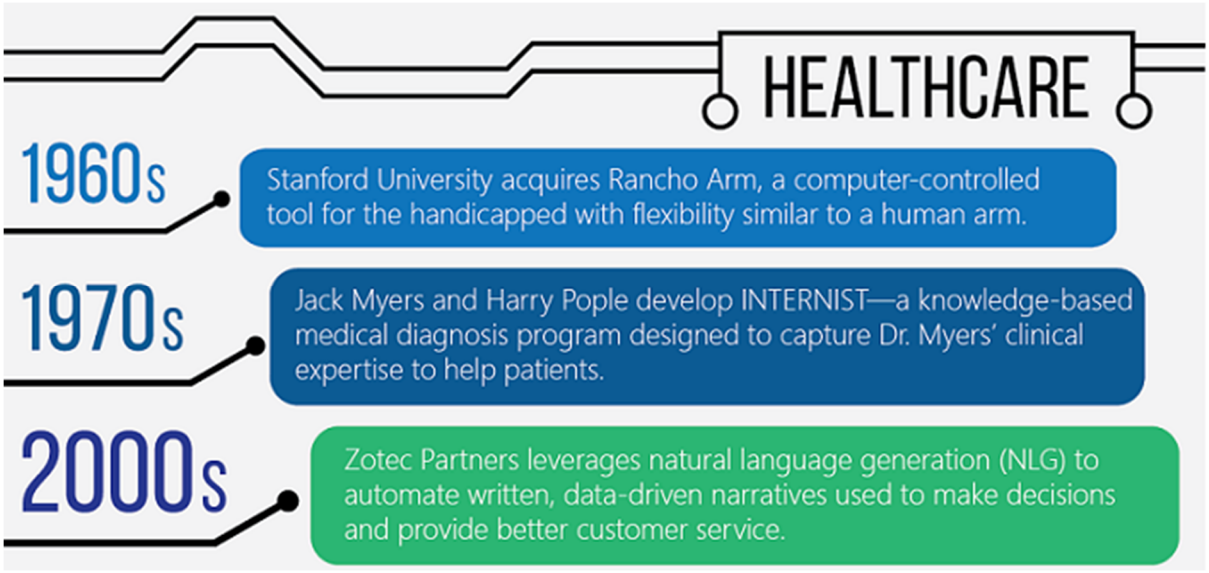
The history of AI (Artificial Intelligence) can be traced back to the 1950s, when researchers first started exploring the possibility of creating intelligent machines. The term "artificial intelligence" was first coined by John McCarthy, who is considered one of the pioneers of the field.

- During the 1950s and 1960s, researchers developed a number of early AI systems, including rule-based expert systems and the first machine learning algorithms. However, progress in AI was slow and many researchers became disillusioned with the field, leading to what was known as the "AI winter" in the 1970s and 1980s.
- In the 1980s and 1990s, there was renewed interest in AI, and researchers began to develop more sophisticated algorithms and techniques, such as neural networks and genetic algorithms. This led to significant progress in areas such as computer vision, speech recognition, and natural language processing.
- In the 2000s and 2010s, the rise of big data and the availability of powerful computing resources led to a new wave of progress in AI, particularly in the area of machine learning. This has led to the development of advanced AI systems, such as deep learning algorithms, that have achieved breakthroughs in areas such as image and speech recognition, natural language processing, and game playing.

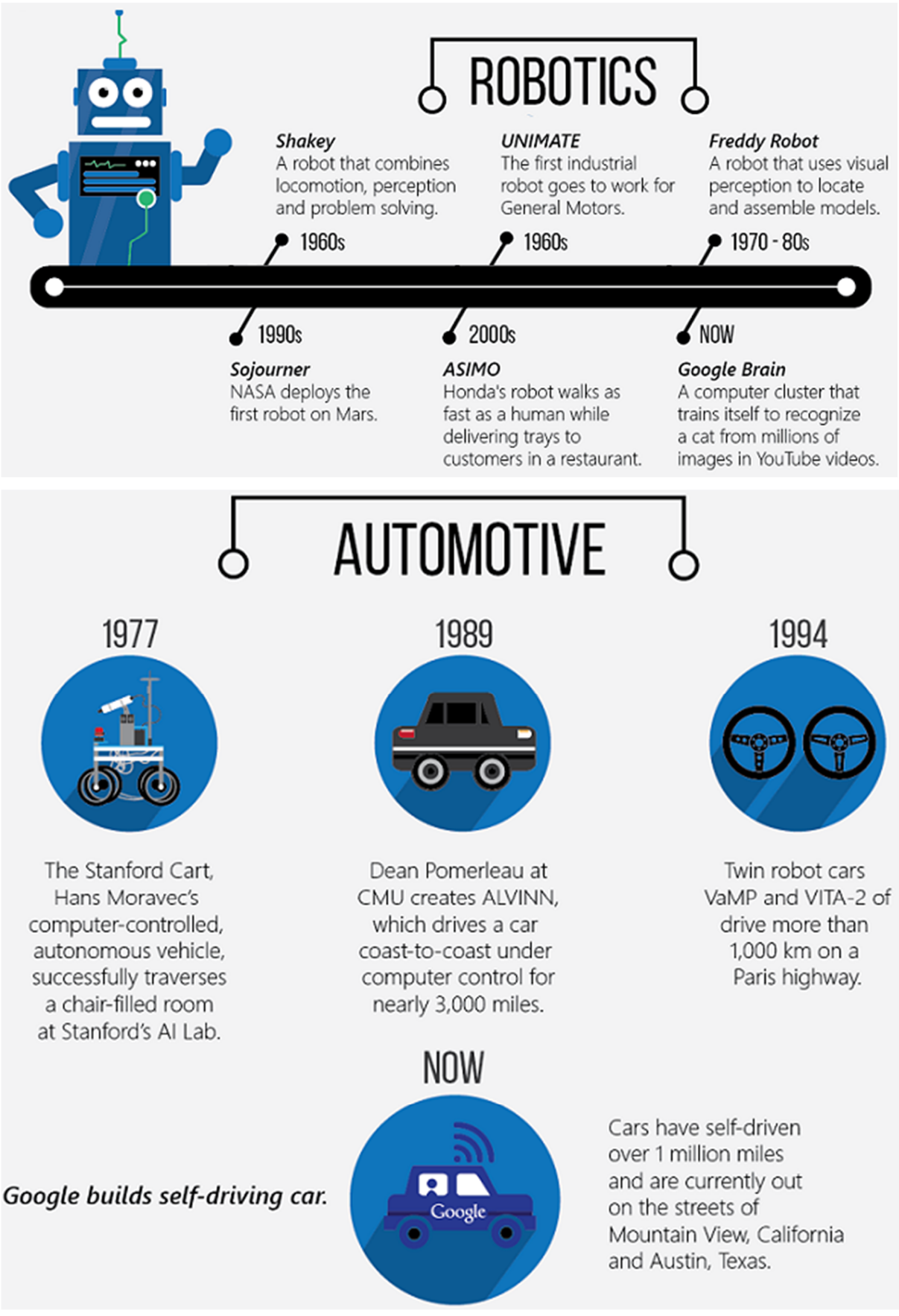
Today, AI is a rapidly growing field that is driving innovation in a wide range of areas, from autonomous vehicles and robotics to healthcare and finance. As computing power continues to increase and new AI algorithms are developed, the potential applications of AI are likely to continue to expand.



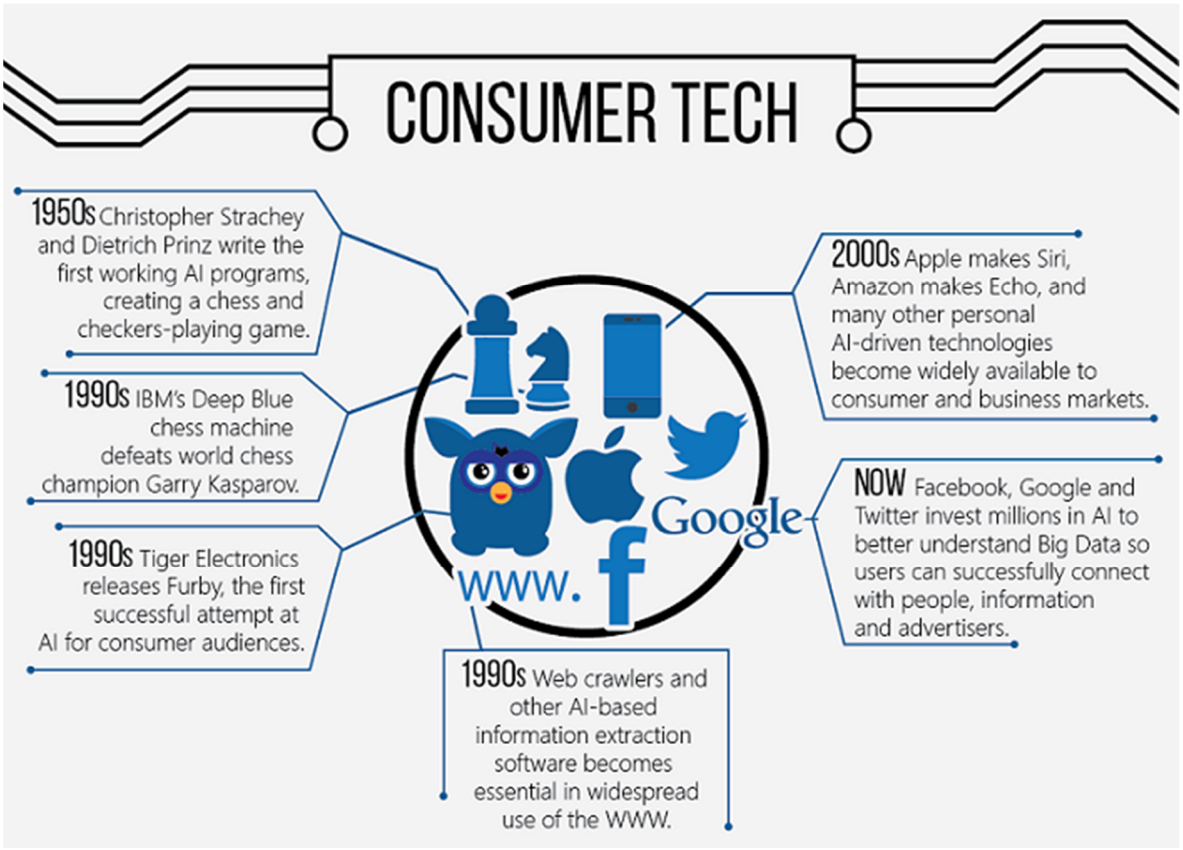

1950s
THE BEGINNING
Alan Turing, the father of AI, introduces the Turing Test, which defines a standard for a machine to be called "intelligent."
The term
"Artificial Intelligence"
is used for the first time during the second Dartmouth Conference.

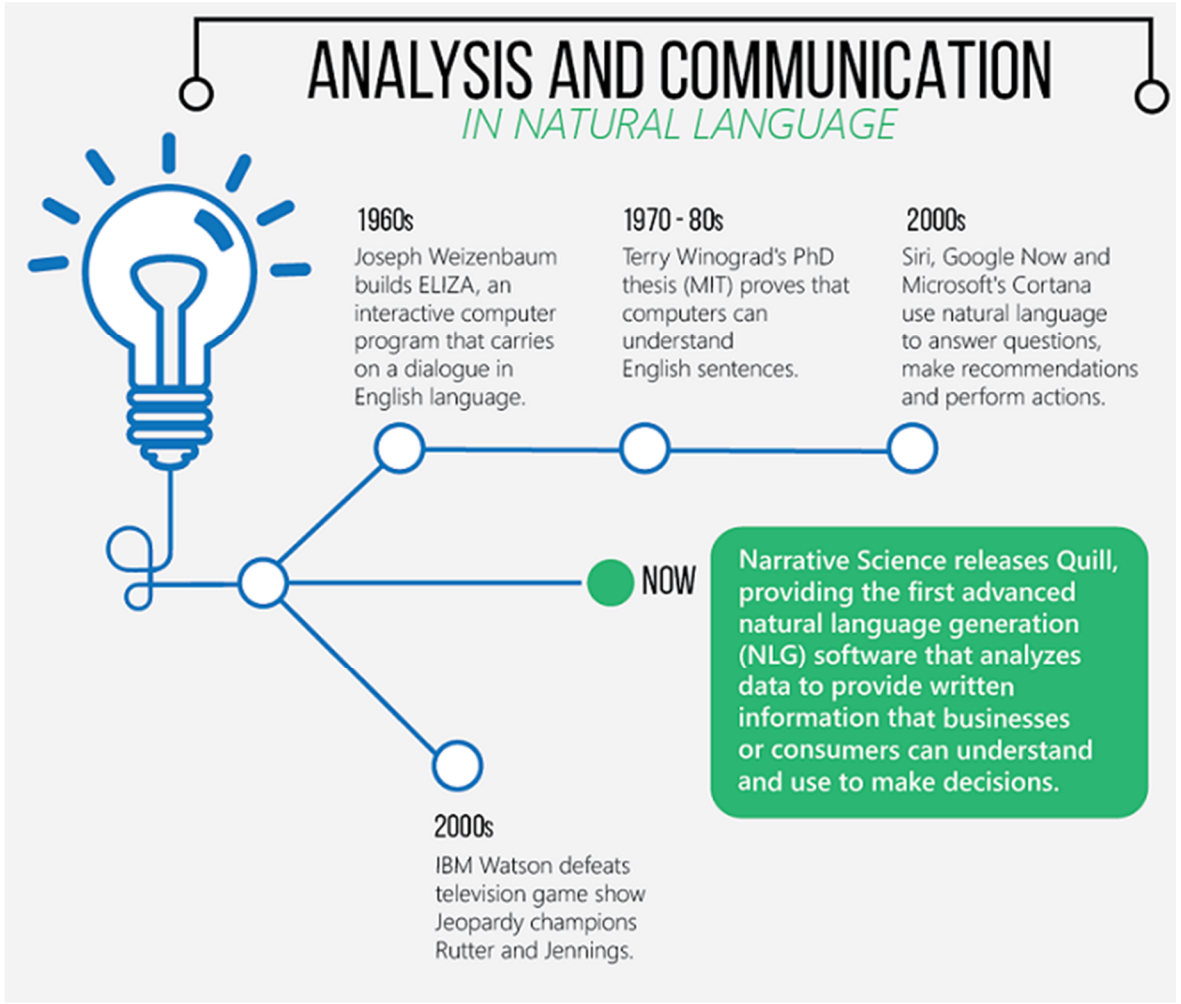

HEALTHCARE
1960s
Stanford University acquires Rancho Arm, a computer-controlled tool for the handicapped with flexibility similar to a human arm.
1970s
Jack Myers and Harry Pople develop INTERNIST—a knowledge-based medical diagnosis program designed to capture Dr. Myers' clinical expertise to help patients.
2000s
Zotec Partners leverages natural language generation (NLG) to automate written, data-driven narratives used to make decisions and provide better customer service.


1980s
FINANCIAL SERVICES
2000s
Generation 2.0 of Alacrity is released, the first commercial strategic and managerial advisory system based on the evolution of markets and competitive strategies.
Security Pacific National Bank authorizes its new Fraud Prevention Task Force applying AI technology as an early warning system to prevent fraud.
Credit Suisse leverages NLG to analyze proprietary and market data to produce reports on assessing company expectations, upside and risk, helping stakeholders make long-term investments.
Companies apply AI for fraud prevention and cyber-security. Organizations like Allianz use machine-learning to identify vulnerabilities and suspicious activity reducing external threats to customer data.
USAA, which provides financial services to more than 10 million current and former members of the U.S. Military and their families, adopts NLG to enhance their financial advice.



Source: Artificial Intelligence Then and Now https://insidebigdata.com/2015/10/03/artificial-intelligence-then-and-now/


CONSUMER TECH
1950s Christopher Strachey and Dietrich Prinz write the first working AI programs, creating a chess and checkers-playing game.
1990s IBM's Deep Blue chess machine defeats world chess champion Garry Kasparov.
1990s Tiger Electronics releases Furby, the first successful attempt at AI for consumer audiences.
2000s Apple makes Siri, Amazon makes Echo, and many other personal AI-driven technologies become widely available to consumer and business markets.
NOW Facebook, Google and Twitter invest millions in AI to better understand Big Data so users can successfully connect with people, information and advertisers.
1990s Web crawlers and other AI-based information extraction software becomes essential in widespread use of the WWW.
Google
www.

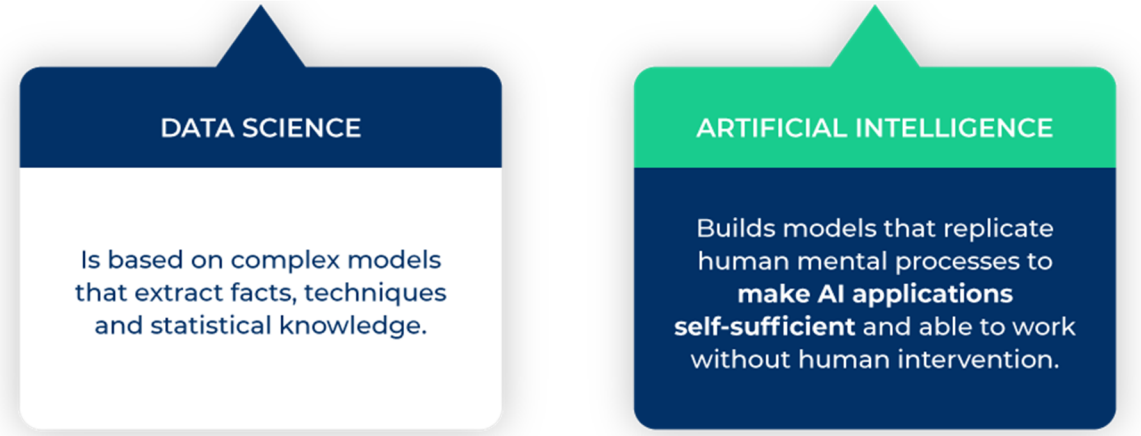

ANALYSIS AND COMMUNICATION
IN NATURAL LANGUAGE
1960s
Joseph Weizenbaum builds ELIZA, an interactive computer program that carries on a dialogue in English language.
1970 - 80s
Terry Winograd's PhD thesis (MIT) proves that computers can understand English sentences.
2000s
Siri, Google Now and Microsoft's Cortana use natural language to answer questions, make recommendations and perform actions.
NOW
Narrative Science releases Quill, providing the first advanced natural language generation (NLG) software that analyzes data to provide written information that businesses or consumers can understand and use to make decisions.
2000s
IBM Watson defeats television game show Jeopardy champions Rutter and Jennings.

Scope of AI and data science

AI (Artificial Intelligence) and Data Science are two related but distinct fields in computer science.

- AI is a broad field that focuses on creating machines or computer systems that can perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and natural language processing. AI involves the development of algorithms and models that enable machines to learn and reason about the world, make predictions, and make decisions.
- Data Science, on the other hand, is a more specific field that focuses on the extraction of insights and knowledge from data. Data Science involves the use of statistical models, algorithms, and visualization tools to analyze large and complex data sets, identify patterns and trends, and make predictions.
- While there is some overlap between AI and Data Science, the two fields differ in their goals and approaches. AI is focused on creating intelligent machines that can learn and reason about the world, while Data Science is focused on extracting insights and knowledge from data.
- Another key difference between AI and Data Science is the types of data they work with. AI often deals with complex and unstructured data, such as images, video, and text, while Data Science typically deals with structured data, such as numerical or categorical data in tables or databases.

In summary, AI and Data Science are related but distinct fields in computer science. While AI is focused on creating intelligent machines that can learn and reason about the world, Data Science is focused on extracting insights and knowledge from data. Exclusive highlights are given below. Source: https://www.algotive.ai/blog/what-is-data-science-and-how-does-it-work-with-artificial-intelligence



DATA SCIENCE

Consists of analyzing and visualizing data to find hidden patterns for decision making.

It is based on different statistical, design and development methods, working with structured and unstructured data.

Total human control: a data scientist extracts and then analyzes the information, processes and uses it.

We use it to perform predictive analysis in situations that require fast mathematical reasoning.

ARTIFICIAL INTELLIGENCE

Involves implementing predictive tools and models to determine results in fields such as safety, sales, processes, etc.

It relies on algorithms and database updating to operate.

Partial human control: a person programs software and machinery, but they then AI handles the data independently.

We use it for risk analysis, to make decisions in a short time and without emotional influence, as for repetitive tasks.

DATA SCIENCE

Encompasses multiple analytical, descriptive, predictive and prescriptive applications.

Mainly implemented at:
- Internet Search Engines
- Advertising
- Marketing

Example:
Analyzes data collected from call centers to understand customer rotations and create strategies to retain them.

ARTIFICIAL INTELLIGENCE

Relies on computer applications that simulate human cognition and intelligence.

Mainly implemented at:
- Transport
- Healthcare
- Manufacturing
- Robotics
- Automation
- Security

Example:
Uses facial recognition and databases to detect suspicious subjects and activities.

ITS MAIN TOOLS, **LIBRARIES** AND **PLATFORMS** ARE:

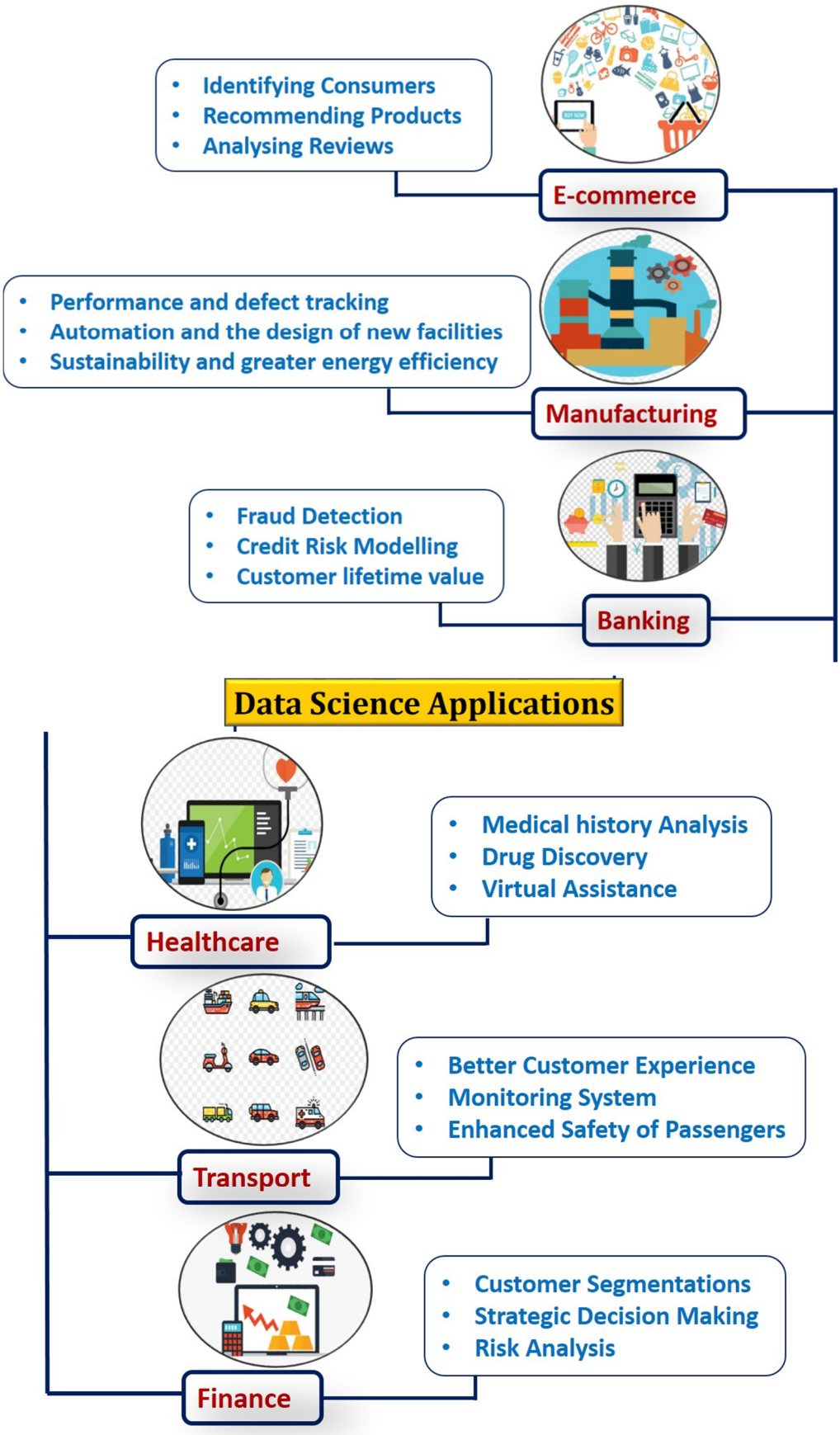
DATA SCIENCE
- Python
- R
- Jupyter
- Zeppelin
- RStudio

*It is worth noting that data science uses a larger number of programs because its analysis processes are more time-consuming and consist of several steps.

ARTIFICIAL INTELLIGENCE
- TensorFlow
- Kaffee
- scikit-learn

	Data Science	Artificial Intelligence
1	Requires pre-processing, analysis, visualization, and build predictive models	Use pretrained models for classification, predictions and pattern recognition
2	Primarily uses statistical techniques and ML Models	Implements ML and Deep Learning algorithms
3	Finds hidden patterns in data	Gives autonomy to data models
4	Builds ML and data models using statistical insights	Used for building models that surpass or match human cognition and understanding capabilities
5	Does not involve high scientific processing	Involves high scientific processing
6	It is a data analytics and statistical technique	Mainly machine learning and deep learning technique
7	Data is controlled with data science techniques	Involves robotic control with AI and machine learning techniques

Source: https://www.analytixlabs.co.in/blog/data-scientist-vs-ai-artificial-intelligence-engineer-what-is-the-difference/



Source: https://www.shiksha.com/it-software/articles/data-science-career-and-future-scope-blogId-26453

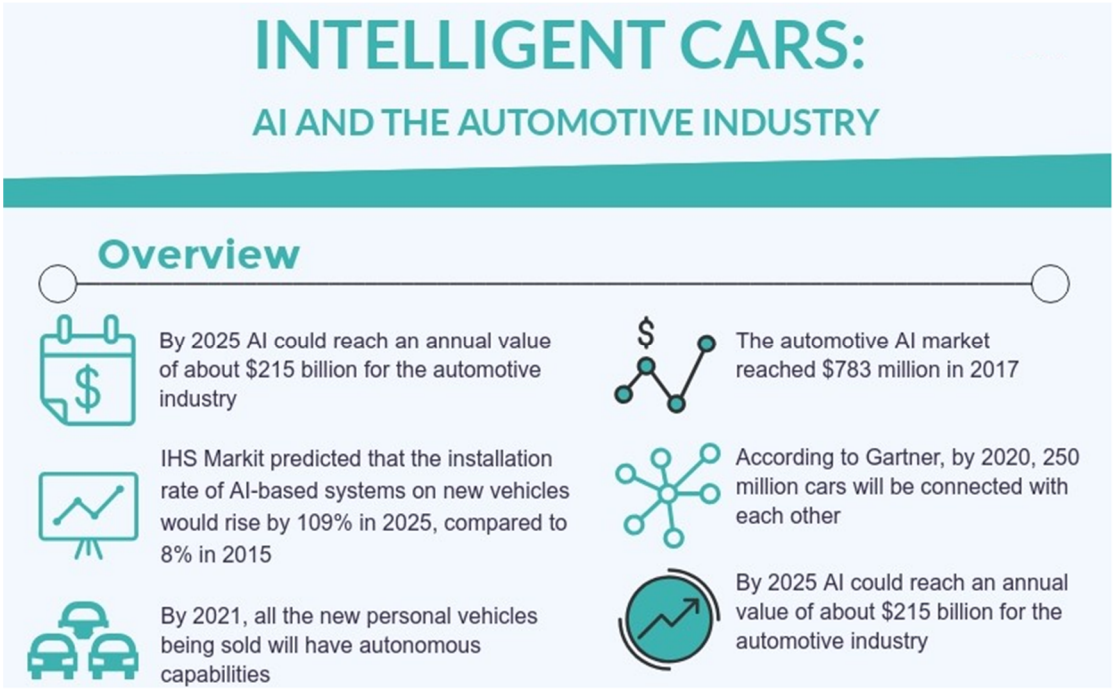
Applications of AI-ML in mechanical engineering domain

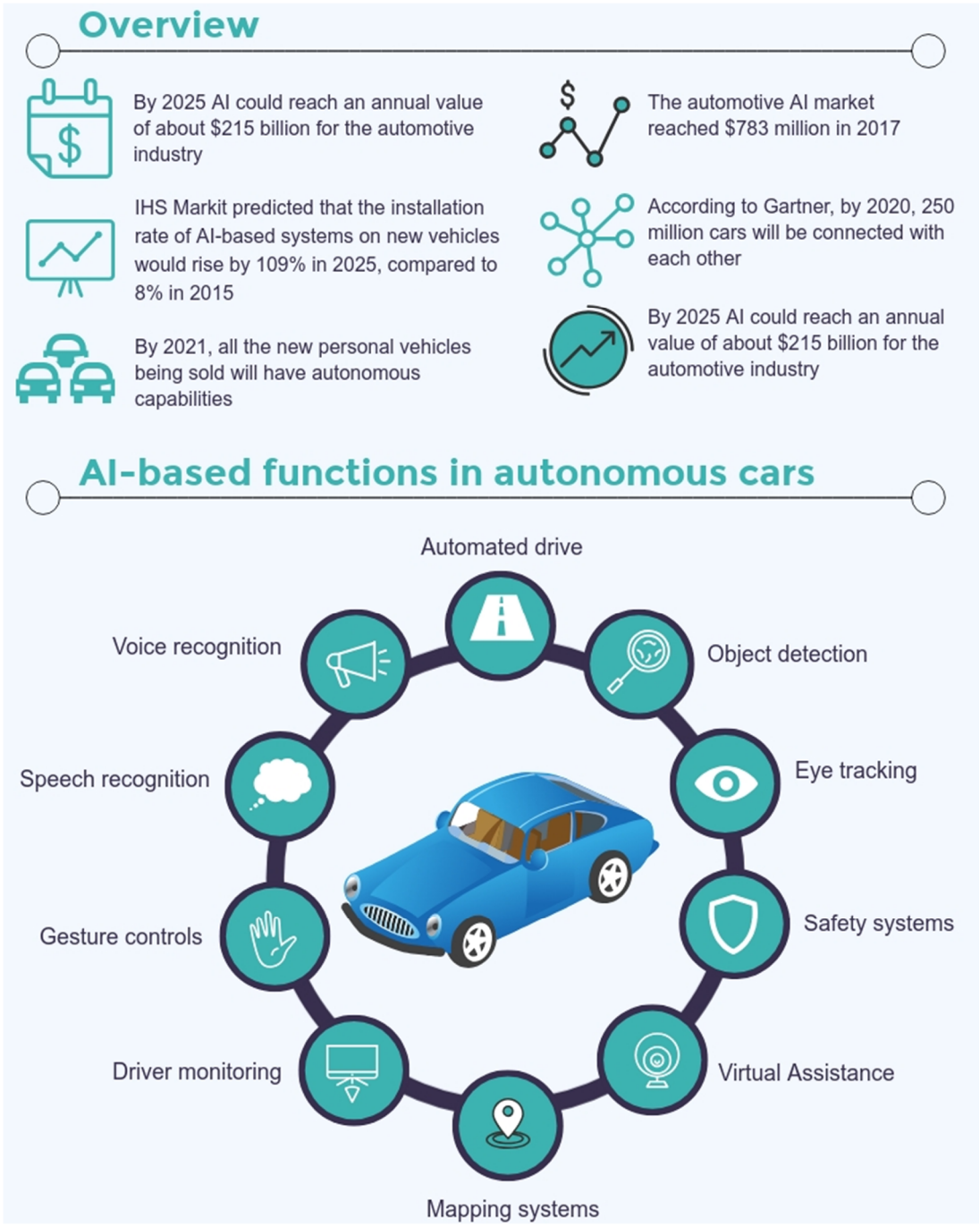
Artificial intelligence is playing an increasingly important role in mechanical engineering and various applications are summarised here.

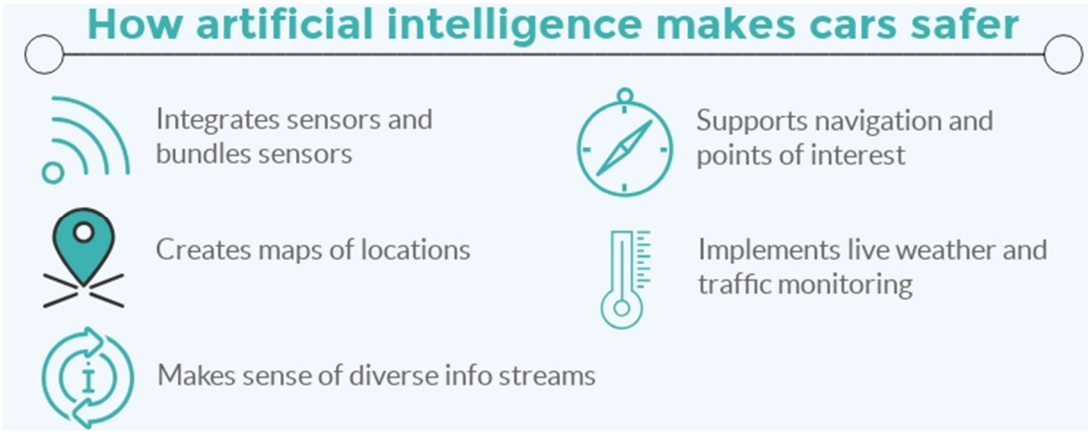
Automobile industry

AI in the automobile industry, with many applications being developed to enhance vehicle performance, safety, and efficiency. Here are a few examples:

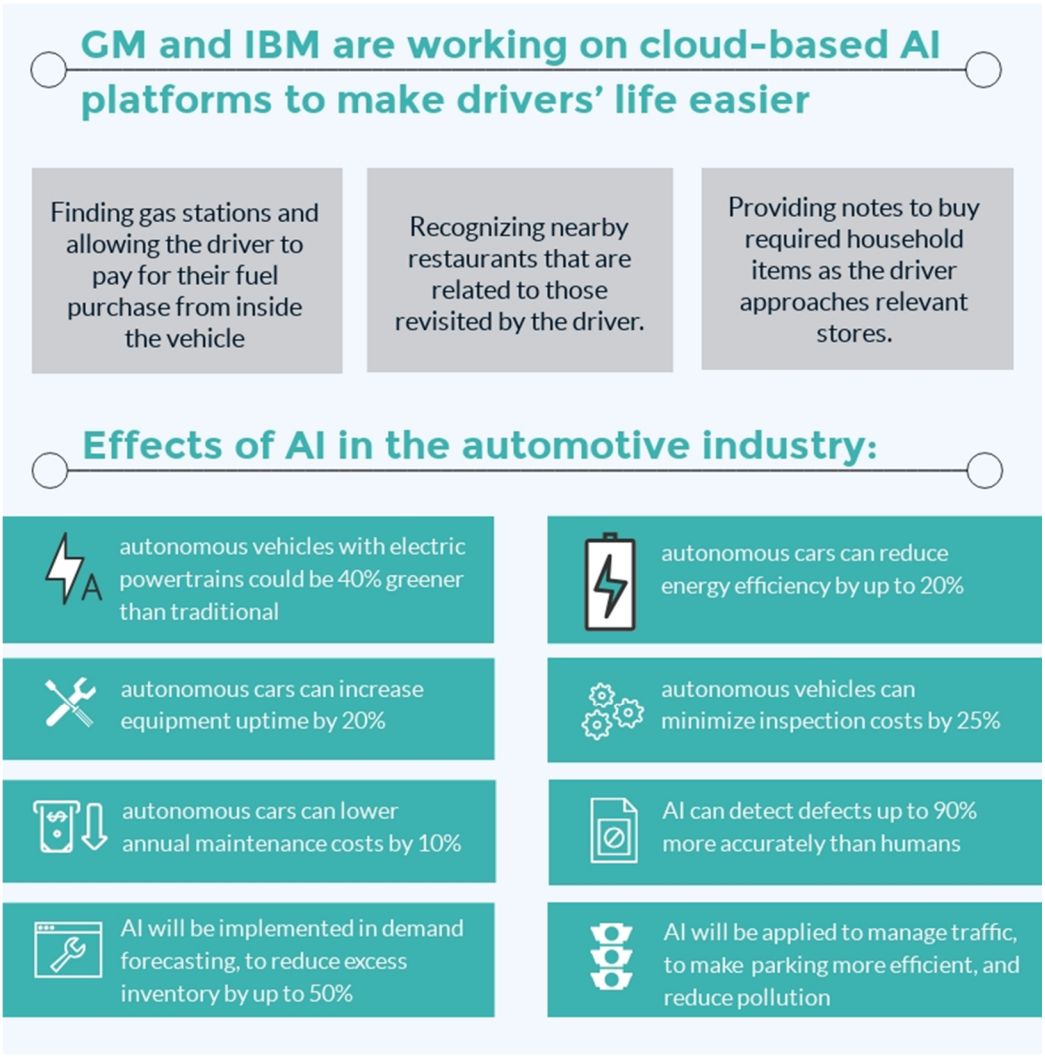
- **Autonomous driving:** AI is essential for the development of self-driving cars. Autonomous vehicles use sensors such as lidar, radar, and cameras to perceive the environment and make decisions. Machine learning algorithms are used to train the AI system to recognize objects, predict their movements, and make decisions on how to navigate the vehicle safely.
- **Predictive maintenance:** AI can help predict when a vehicle needs maintenance by analyzing data from sensors and other sources. This can help prevent breakdowns and reduce downtime.
- **Personalized driving experience:** AI can analyze driver behavior and preferences to create a personalized driving experience. For example, the AI system can adjust the seat, temperature, and music based on the driver's preferences.
- **Enhanced safety features:** AI is being used to develop advanced safety features such as collision detection and avoidance, lane departure warnings, and adaptive cruise control.
- **Connected cars:** AI can enable vehicles to communicate with each other and with infrastructure, creating a network of connected cars that can share information and coordinate movements.



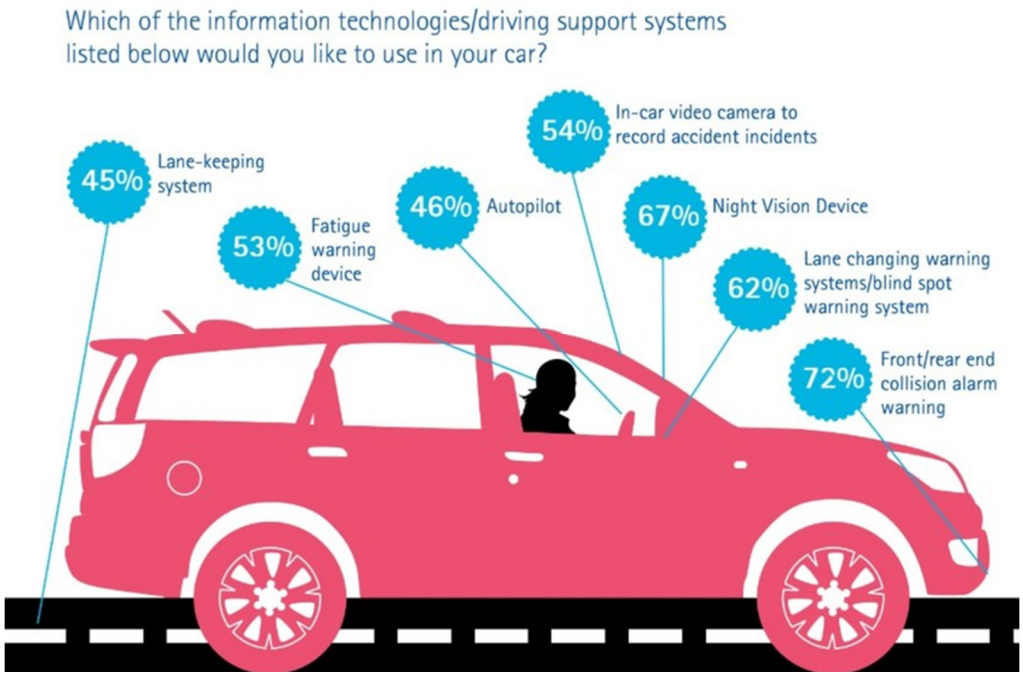




Source: https://rickscloud.com/intelligent-cars-ai-and-the-automotive-industry/



Source: https://www.linkedin.com/pulse/how-artificial-intelligence-machine-learning-auto-models-mishanin/



- Detect and classify objects in radar signal and get their distance
- Using combination of CNN and LSTM

Radar Perception

Lidar Perception

- Detection for lane markers, drivable space vehicles, etc.
- Annotation for Deep Learning

- Detect and classify lane markers, drivable space, traffic sign, other vehicles, blinkers, etc.
- Using multiple CNN and LSTM-like (TAGM) networks on multiple cameras

Camera Perception

Behavior planning

- Reinforcement learning for safe and human-like driving behavior
- Prediction based on LSTM and convolutional networks

Source: https://venturebeat.com/ai/how-ai-is-impacting-the-automotive-world/

Manufacturing industry

AI is transforming the manufacturing industry in a number of ways. Here are some examples:

- **Quality control:** AI can be used to monitor production lines and identify defects or quality issues in real-time. Machine learning algorithms can analyze large amounts of data and identify patterns that indicate potential problems before they become major.
- **Predictive maintenance:** AI can predict when machines and equipment will require maintenance or repairs, reducing downtime and increasing efficiency.
- **Supply chain optimization:** AI can be used to optimize the supply chain by analyzing data on inventory levels, production schedules, and shipping times. This can help manufacturers reduce costs and improve delivery times.
- **Robotic automation:** AI can be used to control robots and other automation systems, allowing for more efficient and flexible manufacturing processes.
- **Smart factories:** AI can enable factories to become more connected and intelligent, with machines and equipment communicating with each other and with human operators. This can lead to more efficient and effective production processes.
- **Demand Planning:** AI allows the industry to optimize product availability by reducing out of stocks and waste or spoilage. AI can also assist with obtaining a more reliable perception of sales patterns.
- **Inventory Management:** AI can be utilized to acquire a stabler knowledge of inventory levels permitting companies to plan for the future and withdraw stock-outs.

Source: https://www.vlcsolutions.com/blog/artificial-intelligence-in-manufacturing/

Demand planning

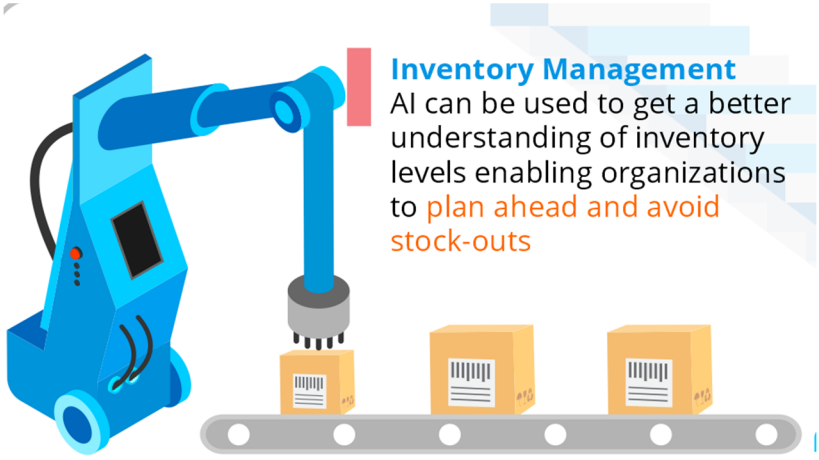
AI enables organization to optimize product availability by decreasing out of stocks and spoilage. AI can also help with getting a better understanding of sales patterns.

Company uses AI algorithms to predict demand based on a wide variety of data gathered from social media, weather, and financial markets.[4]

Product development/R&D

AI enables organizations to expediate product development and R&D by reducing the test times and driving more concrete insights from customer data and demands

Icompany is using big data and AI platforms to create tests for hard to validate functionalities improving the targeted coverage by 230x compared to standard regression tests[5]



Inventory Management

AI can be used to get a better understanding of inventory levels enabling organizations to plan ahead and avoid stock-outs

Production

TAKT can be reduced by using AI to streamline manufacturing processes, improving throughput

Company uses AI to automatically adjust rate, speed, acceleration, etc. of the industrial robots leading to the time reduction to 1/10th of conventional method[6]

Process control
AI can help organizations optimize processes to achieve production levels with enhanced consistency, economy and safety

Company uses AI to influence operations by predicting outcomes and improving efficiency levels to optimise output.[7]

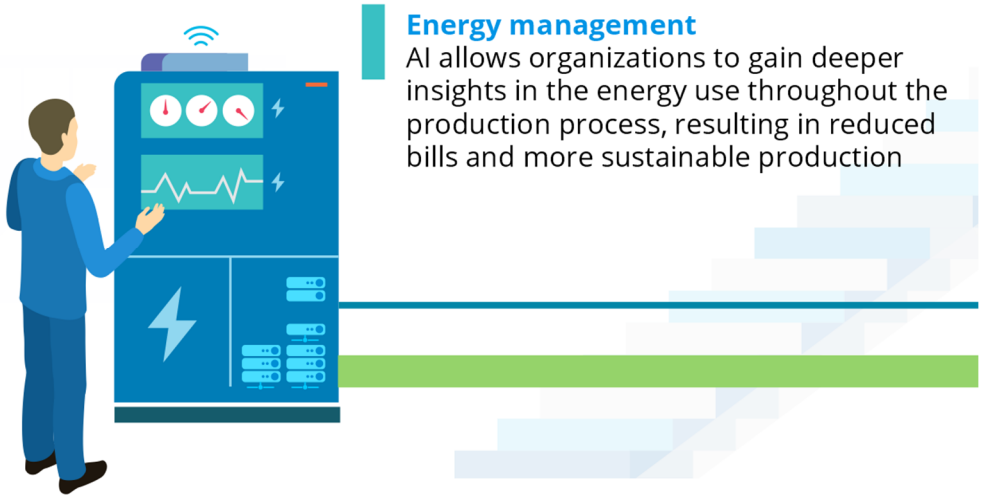
Quality control
Product quality inspections bring uniformity and efficiency in quality control, using image-based and sensor-based processes.

Company uses AI to promote high-level of precision in tire manufacturing, resulting in an improvement of more than 15% over traditional methods

Maintenance
Using AI, organizations can predict and prepare for asset failure, reducing (or even avoiding) downtime.

company uses computer vision to analyse images from robot mounted cameras to spot early signs of failing robotic part [8]

Safety
AI is used to get a better understanding of risk factors within the shop floor and can help safer operations



Energy management
AI allows organizations to gain deeper insights in the energy use throughout the production process, resulting in reduced bills and more sustainable production

Mechanical Design

Here are some examples of how AI is being used in mechanical design.

- **Generative design:** AI can be used to generate designs based on specified parameters, such as weight, strength, and material usage. By using machine learning algorithms to

analyze vast amounts of data, AI can produce innovative designs that are optimized for performance and cost-effectiveness.

- **Simulation and analysis:** AI can be used to simulate the behavior of mechanical systems and analyze the results. This can help in identifing potential issues and optimize designs before they manufacturing, reducing costs & improving quality.
- **Design optimization:** It is used to optimize existing designs by analyzing data leading to products to be lighter, stronger, and more efficient.
- **Design validation:** AI can be used to validate designs by comparing them to known data and identifying potential issues. This can help ensure that designs are safe and effective before they are manufactured.
- **Collaboration and knowledge management:** AI can be used to facilitate collaboration among designers and engineers by providing tools for sharing data and knowledge. This can help ensure that everyone is working on the most up-to-date information and can lead to more effective design processes.

Thermal / Heat Transfer / HVAC / Fluid Mechanics / Fluid Power

Here are some examples of how AI is being applied in various ways.

- **Optimization of thermal systems:** AI can be used to optimize the design of thermal systems such as engines, heat exchangers, and refrigeration systems. By analyzing large amounts of data and using machine learning algorithms, AI can identify areas for improvement and optimize the design for performance and efficiency.
- **Predictive maintenance:** AI can be used to predict when HVAC and fluid power systems will require maintenance or repairs. By analyzing data from sensors and other sources, AI can identify potential issues before they become major problems.
- **Smart control systems:** AI can be used to control HVAC and fluid power systems, allowing for more efficient and flexible operation. This can help reduce energy costs and improve performance.
- **Simulation and modeling:** AI can be used to simulate and model the behavior of fluid systems, such as airflow or fluid flow through a pipe. This can help engineers identify potential issues and optimize designs before they are manufactured.
- **Optimization of fluid dynamics:** AI can be used to optimize the design of fluid systems such as pumps and turbines, improving performance and reducing energy consumption.

Materials and Metallurgy

Here are some examples of how AI is being applied in various ways.

- **Materials discovery:** AI can be used to predict the properties of new materials before they are synthesized, reducing the need for trial-and-error experimentation. By using

machine learning algorithms to analyze large amounts of data on the properties of existing materials, AI can identify promising candidates for new materials with specific properties.

- **Image recognition:** AI can be used to recognize materials based on images or visual data. Machine learning algorithms can be trained on large datasets of images of different materials, allowing the AI to recognize and identify materials in new images.
- **Spectral analysis:** AI can be used to analyze the spectra of materials, such as their reflectance or absorption spectra. By training machine learning algorithms on spectral data, AI can identify and recognize materials based on their unique spectral fingerprints.
- **X-ray diffraction analysis:** AI can be used to analyze X-ray diffraction patterns of materials. By training machine learning algorithms on patterns from known materials, AI can identify and recognize new materials based on their diffraction patterns.
- **Sensing technologies:** AI can be used with various sensing technologies, such as spectroscopy or magnetometry, to recognize and identify materials based on their physical properties.

Energy Conservation and Management

Here are some examples of how AI is being applied in various ways.

- **Smart Grids:** AI can be used to optimize the operation of smart grids, which are modern electrical grids that use sensors and other technologies to manage energy more efficiently. By analyzing data from sensors and other sources, AI can identify patterns in energy usage and predict future demand, helping to optimize the distribution and consumption of energy.
- **Building Energy Management:** AI can be used to optimize the energy consumption of buildings. Machine learning algorithms can analyze data from sensors and other sources to identify patterns in energy usage and predict future demand, allowing building managers to optimize heating, ventilation, and air conditioning (HVAC) systems and lighting to save energy.
- **Renewable Energy Optimization:** AI can be used to optimize the operation of renewable energy sources such as wind turbines and solar panels. By analyzing data from sensors and other sources, AI can predict energy output and optimize the operation of renewable energy sources to maximize energy production.
- **Energy Efficiency:** AI can be used to identify opportunities for energy efficiency improvements in industrial and commercial settings. Machine learning algorithms can analyze data from sensors and other sources to identify inefficiencies in energy usage, allowing organizations to take steps to reduce energy waste and save money on energy costs.

- **Energy Storage Optimization:** AI can be used to optimize the operation of energy storage systems such as batteries. By analyzing data from sensors and other sources, AI can predict energy demand and optimize the operation of energy storage systems to maximize energy availability.

Symbolic approach of AI

Symbolic AI (rule-based AI) involves the use of logic & symbols to represent and reason about knowledge. It is based on the idea that intelligence can be simulated by representing knowledge using symbols and rules, and manipulating those symbols to draw conclusions.

- In Symbolic AI, knowledge is represented using logical expressions or symbols, such as predicates, propositions, and rules. These symbols can be combined and manipulated using logical operations, such as deduction, induction, and abduction, to derive new knowledge and solve problems.
- One of the key advantages of Symbolic AI is that it is transparent and explainable. Since knowledge is represented using symbols and rules, it is possible to understand how the system arrived at a particular conclusion or decision. This makes Symbolic AI useful in applications where transparency and interpretability are important, such as in legal or financial decision making.
- However, Symbolic AI has limitations in dealing with uncertainty and handling large amounts of data. It is not well-suited for applications where knowledge is uncertain or incomplete, such as in natural language processing or image recognition. This has led to the development of other subfields of AI, such as Machine Learning, which are better suited for these types of applications.

Here's an example of Symbolic AI from mechanical domain:

Suppose you want to build an expert system that can diagnose car engine problems. The expert system would take in symptoms of a car engine malfunction and output a diagnosis and recommended solution. To do this, you could represent knowledge using symbols and rules. For instance, you might represent the symptom "engine won't start" using the symbol "engine_wont_start". You might also represent possible causes of this symptom using rules, such as:

- If the battery is dead, then the engine won't start
- If the fuel pump is not working, then the engine won't start
- If the spark plugs are faulty, then the engine won't start

These rules can be combined using logical operations such as deduction, induction, and abduction to derive new knowledge and solve problems.

For example, suppose the expert system receives the symptom "engine_wont_start". It can use the rules above to deduce that the problem might be a dead battery, faulty fuel pump,

or faulty spark plugs. The expert system can then ask additional questions or run tests to further narrow down the diagnosis and recommend a solution.

Sub-Symbolic approach of AI

Subsymbolic AI, also known as connectionist AI or neural network AI, is a subfield of artificial intelligence that focuses on creating intelligent systems that can learn from data through a process called training. It is based on the idea that intelligence can be simulated by modeling the way neurons in the brain process information.

In subsymbolic AI, knowledge is represented using interconnected nodes or neurons, which are organized into layers. Each neuron receives input from other neurons, processes that input using an activation function, and outputs a result to other neurons. By adjusting the strength of the connections between neurons, subsymbolic AI systems can learn to recognize patterns, make predictions, and classify data.

One of the key advantages of subsymbolic AI is its ability to learn from data without being explicitly programmed. This makes it well-suited for applications where the rules or logic that govern the problem domain are complex or difficult to specify, such as in natural language processing, image recognition, and speech recognition.

However, subsymbolic AI is often criticized for being a black box, meaning that it is difficult to understand how the system arrived at a particular conclusion or decision. This lack of transparency and interpretability can make it challenging to apply subsymbolic AI in certain contexts, such as in legal or medical decision making.

Here's an example of Sub-Symbolic AI from mechanical domain:

- Subsymbolic AI can also be used to develop an expert system for diagnosing car engine problems. Here's an example of how this might work:
- First, a large dataset of car engine symptoms, causes, and solutions would be collected. Each symptom and cause would be represented as an input vector, and each solution would be represented as an output vector.
- A neural network would then be trained on this dataset using backpropagation, a subsymbolic learning algorithm. The neural network would have several layers of interconnected nodes, or neurons, that learn to recognize patterns in the data.
- When a user inputs a set of symptoms into the system, the neural network would process the input and produce a set of possible causes and solutions. The neural network would adjust its weights and connections based on feedback from users and real-world data, continually improving its accuracy over time.
- As the system is trained on more data, it would become increasingly accurate at diagnosing engine problems and recommending solutions. Additionally, the system

could also be combined with a rule-based system, such as Symbolic AI, to provide a more complete diagnosis.

Symbolic vs. Sub-Symbolic AI

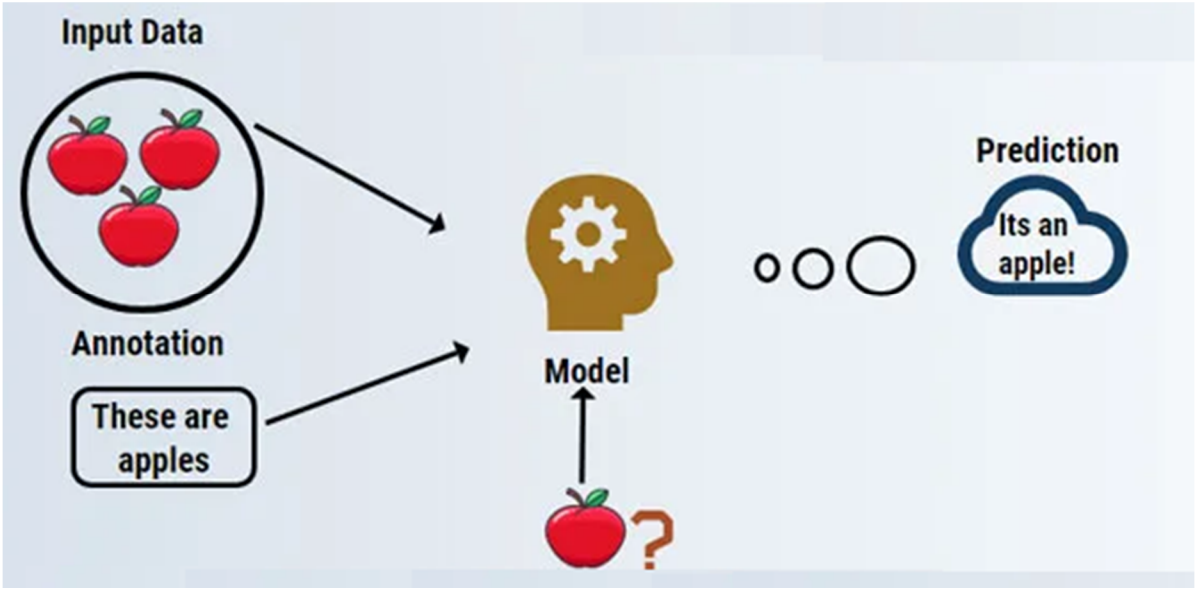
Both symbolic and subsymbolic AI approaches can be used for diagnosing car engine problems.

- Symbolic AI, which uses a rule-based approach, involves encoding expert knowledge and rules into the system in the form of if-then statements. For example, if the engine is making a knocking sound, then it could be a sign of a bad bearing or worn-out piston rings.
- Subsymbolic AI, which uses a neural network approach, involves training the system on a large dataset of car engine symptoms, causes, and solutions. For example, the system could be trained on data from thousands of car engines with known problems, and it would learn to recognize patterns in the data to predict what's causing the problem.

In the context of diagnosing car engine problems, both symbolic and subsymbolic AI approaches could be used together to provide a more complete diagnosis. Symbolic AI could be used to encode expert knowledge and rules into the system, while subsymbolic AI could be used to train the system on a large dataset of engine symptoms and solutions.

Supervised learning

Supervised learning is a type of machine learning algorithm that involves training a model to make predictions based on labeled examples.



- In supervised learning, the algorithm is provided with a dataset that includes input data, or features, and corresponding output data, or labels.

- The goal of the algorithm is to learn the mapping function between the input data and the output data, so that it can make accurate predictions on new, unseen data.

Let's understand supervised machine learning with the help of an example.

- Let's say we have a fruit basket that is filled up with different species of fruits.
- Our job is to categorize fruits based on their category. In our case, we have considered four types of fruits: Apple, Banana, Grapes, and Oranges.

Now we will try to mention some of the unique characteristics of these fruits which make them unique.

Sr. No.	Size	Color	Shape	First Name
1	Small	Green	Round to oval, Bunch shape cylindrical	Grape
2	Big	Red	Rounded shape with a depression at the top	Apple
3	Big	Yellow	Long curving cylinder	Banana
4	Big	Orange	Rounded shape	Orange

Now let us say that you have picked up a fruit from the fruit basket, you looked at its features, e.g., its shape, size, and color, for instance, and then you deduce that the color of this fruit is red, the size if big, the shape is rounded shape with a depression at the top; hence it is an apple.

- Likewise, you do the same for all other remaining fruits as well.
- The rightmost column ("Fruit Name") is known as the response variable.
- This is how we formulate a supervised learning model; now, it will be quite easy for anybody new (Let's say a robot or an alien) with given properties to easily group the same type of fruits together.

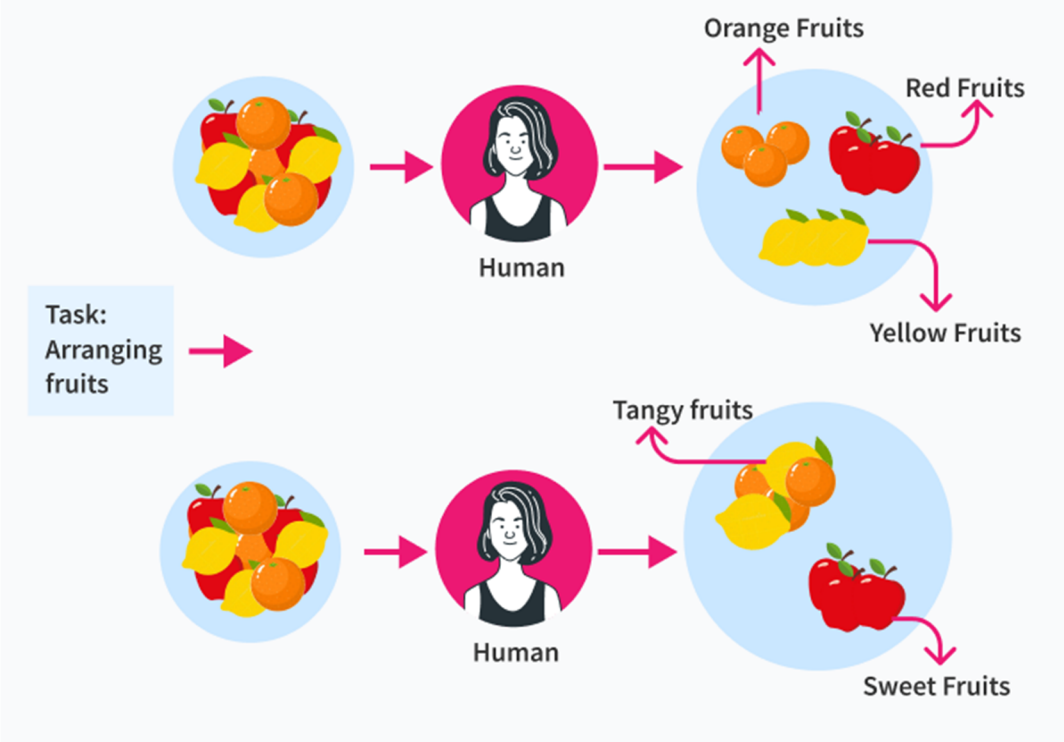
Unsupervised learning

Contrary to Supervised learning, Unsupervised Learning involves presenting the model with unlabelled data. This means that the data does not have any labels, or categories and there is no training involved. This type of learning can be considered as learning without any guidance. The model is left open to learn on its own by detecting the co-relations between the features, without any expected output.

The basic principle of unsupervised learning is to learn in the absence of any ground truth to compare our model with. The idea is to use the underlying patterns in the data to generate a reconstruction of the input data, in a useful format. By 'underlying patterns in the data' we mean that the data is then explained on the basis of similarities, hidden patterns, etc, all without any human intervention.

Let us understand the working of unsupervised Machine Learning with the help of an example:

Suppose we give our model unlabeled images of red colored fruits, such as pomegranate, apples and plums. Our model hasn't seen or known any kind of fruits before and will now try to categorize the images on the basis of similarity and differences in the structure and appearance only. We will let the model to discover the hidden patterns in the data itself.
Combine these two according to the first picture with fruits example.



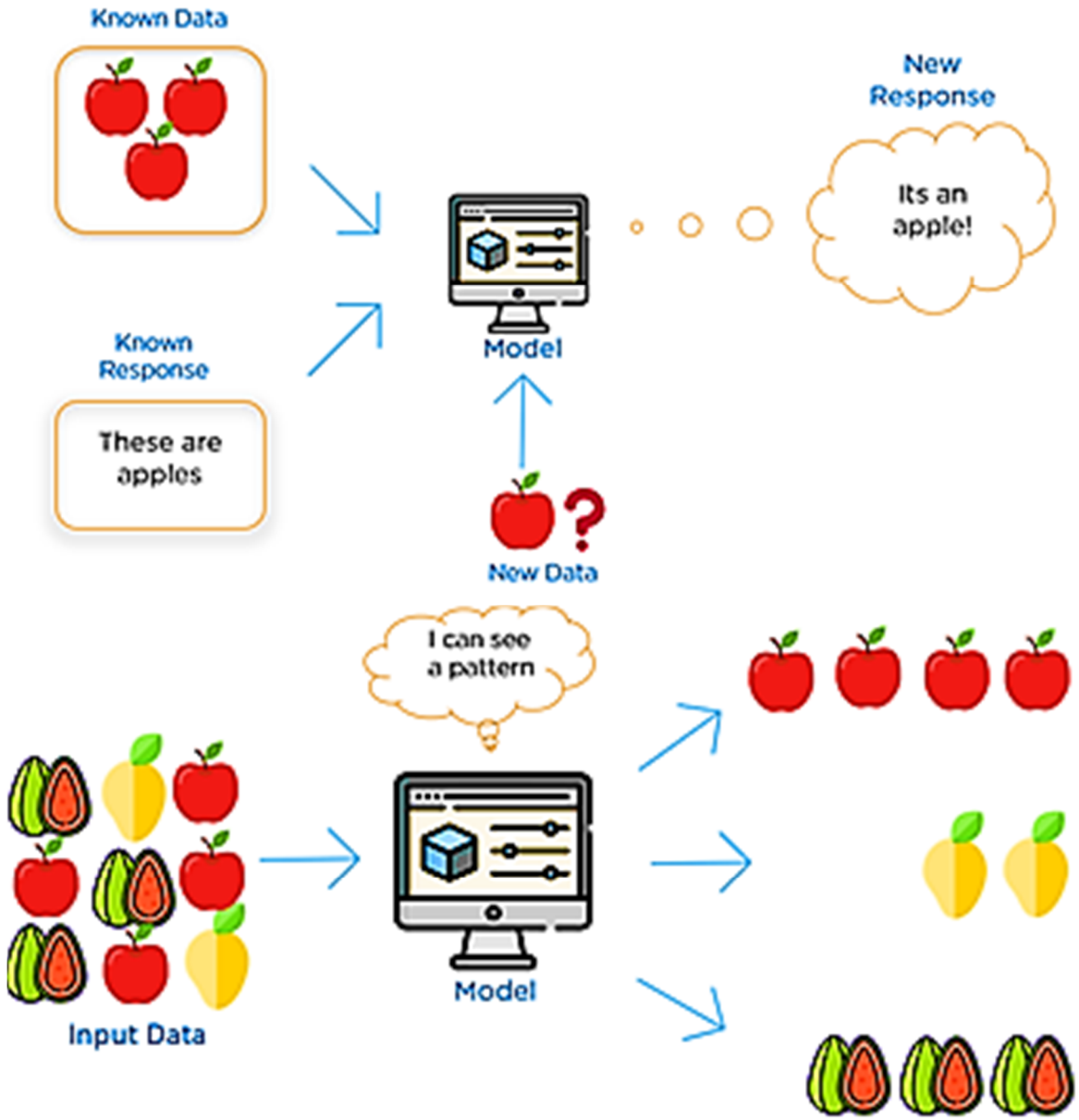
Same way, in recommendation systems, unsupervised learning algorithms help cluster commodities that are frequently bought together. This further helps the sellers to provide recommendations to the buyers, many of which we have seen quite a lot of times on online selling platforms such as "Frequently bought together" and "customers who bought this, also bought this".

Supervised vs. unsupervised learning

Supervised Learning - Remember the time when you used to go to school? The time when you first learnt what an apple looked like? The teacher probably showed a picture of an apple and told you what it was, right? You could identify the particular fruit ever since then.
As you can see in the image below

- Step1 -You provide the system with data that contains photos of apples and let it know that these are apples. This is called labeled data.
- Step2 -The model learns from the labeled data and the next time you ask it to identify an apple, it can do it easily.

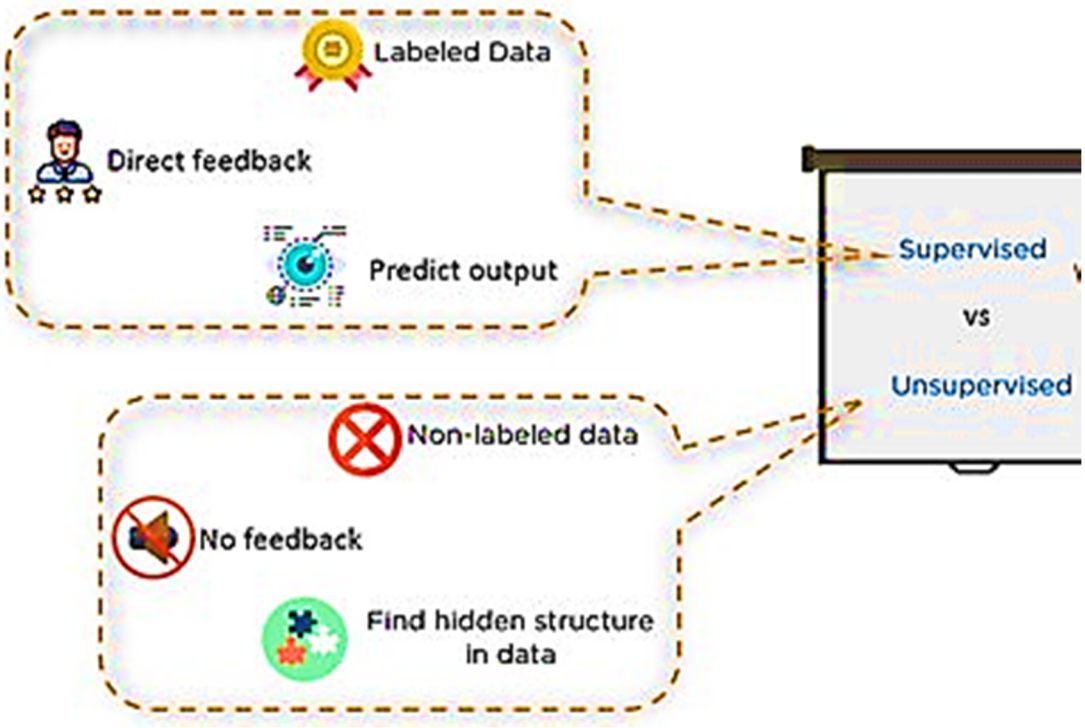
Unsupervised Learning – If somebody gives you a basket full of different fruits and asks you to separate them, you will probably do it based on their colors, shape and size, right? Unsupervised learning works in the same way. As you can see in the image



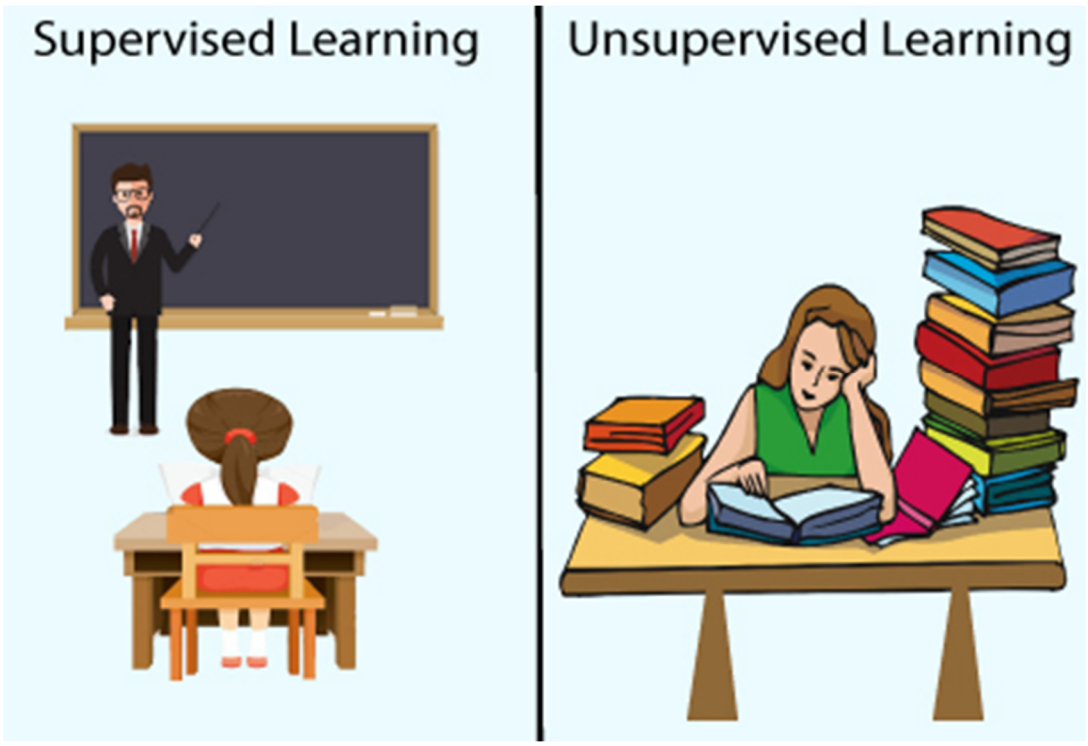
- Step1-You provide the system with a data that contains photos of different kinds of fruits and ask it to segregate it. Remember, in case of unsupervised learning you don't need to provide labeled data.
- Step 2-The system will look for patterns in the data. Patterns like shape, color and size and group the fruits based on those attributes.

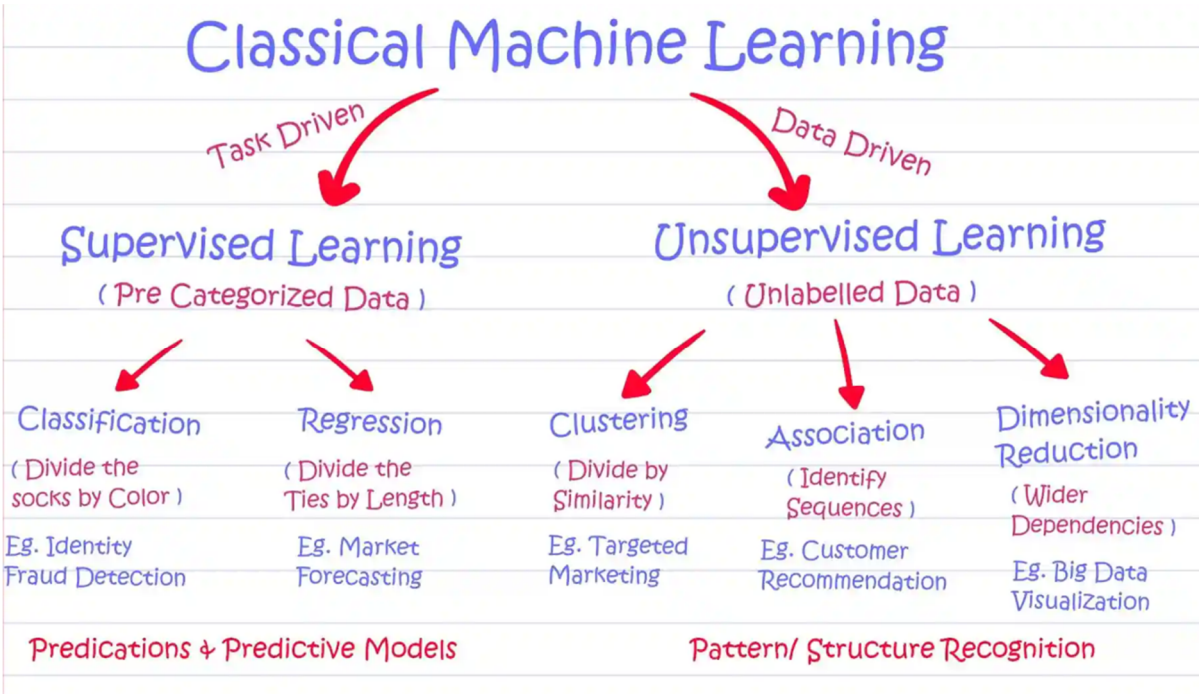
So let's sum up the differences between supervised and unsupervised learning-

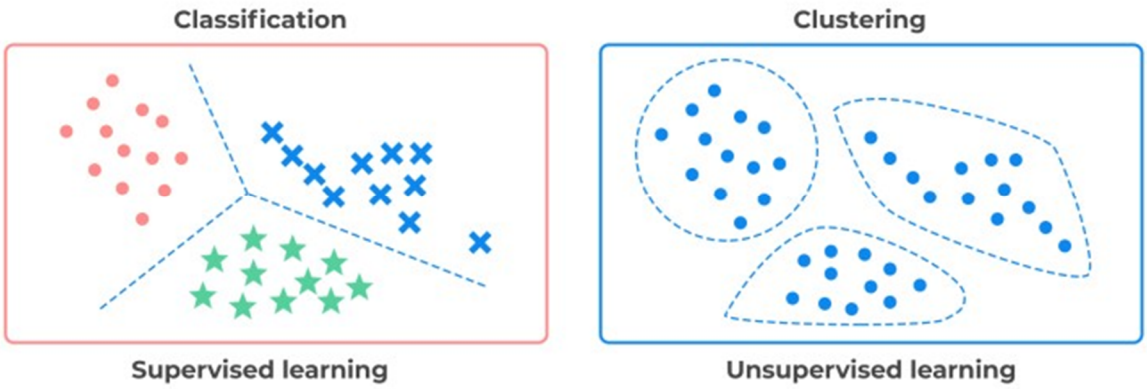
- **Type of input data –** In case of Supervised Learning, the input data is labeled and in case of Unsupervised Learning, the input data is not labeled.
- **Feedback –** In case of Supervised Learning, the system learns from the output and keeps it in mind while in case of unsupervised learning, there is no feedback involved.
- **Function –** Supervised Learning is generally used to predict data whereas, Unsupervised Learning is used to find hidden structure in the data.

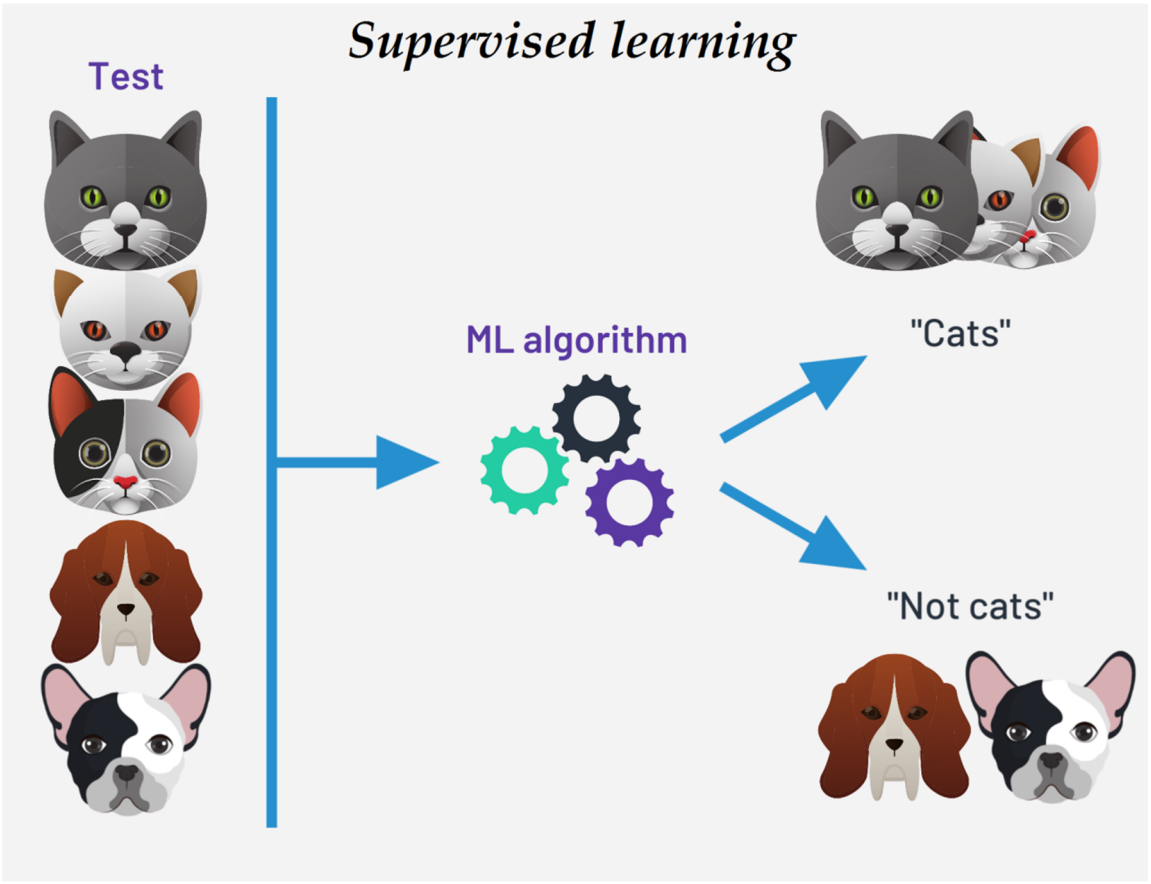


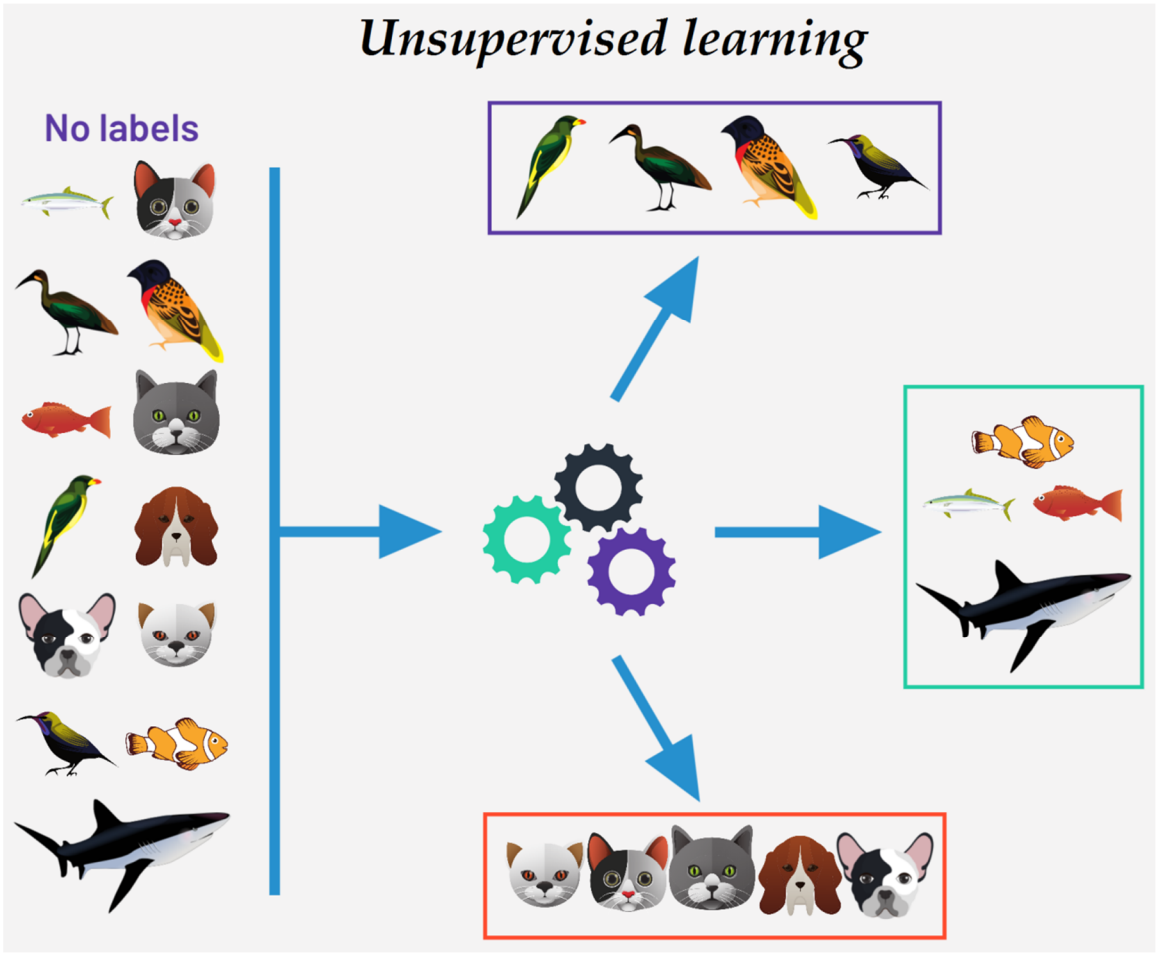
Supervised learning	Unsupervised learning
Input data is labeled	Input data is unlabeled
Has a feedback mechanism	Has no feedback mechanism
Data is classified based on the training dataset	Assigns properties of given data to classify it
Divided into Regression & Classification	Divided into Clustering & Association
Used for prediction	Used for analysis
Algorithms include: decision trees, logistic regressions, support vector machine	Algorithms include: k-means clustering, hierarchical clustering, apriori algorithm
A known number of classes	A unknown number of classes

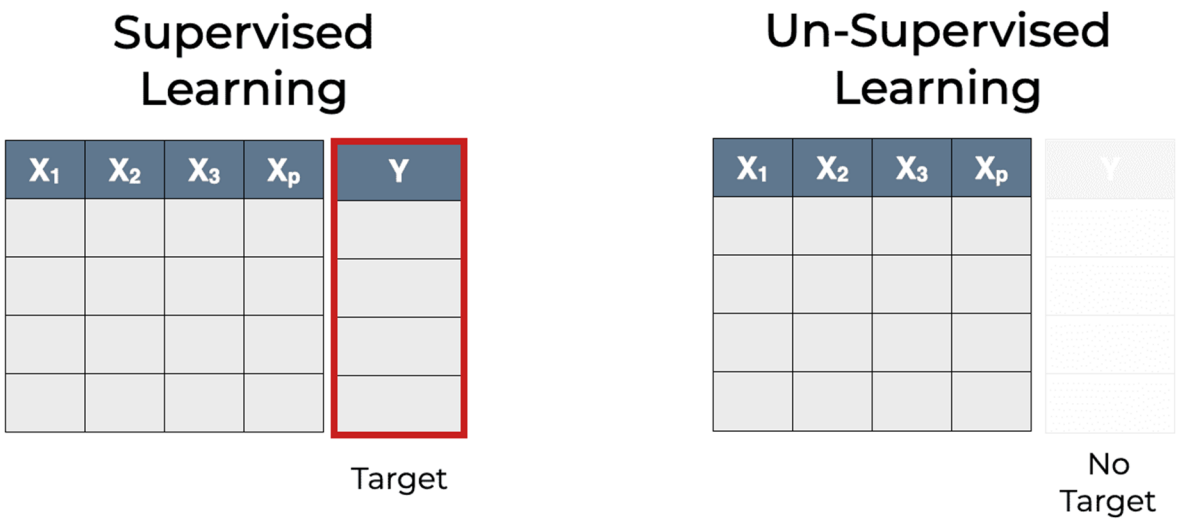

Supervised Learning
Unsupervised Learning


Classical Machine Learning
Task Driven
Data Driven
Supervised Learning
(Pre Categorized Data)
Unsupervised Learning
(Unlabelled Data)
Classification
(Divide the socks by Color)
Eg. Identity Fraud Detection
Regression
(Divide the Ties by Length)
Eg. Market Forecasting
Clustering
(Divide by Similarity)
Eg. Targeted Marketing
Association
(Identify Sequences)
Eg. Customer Recommendation
Dimensionality Reduction
(Wider Dependencies)
Eg. Big Data Visualization
Predications & Predictive Models
Pattern/ Structure Recognition


Classification
Clustering
Supervised learning
Unsupervised learning

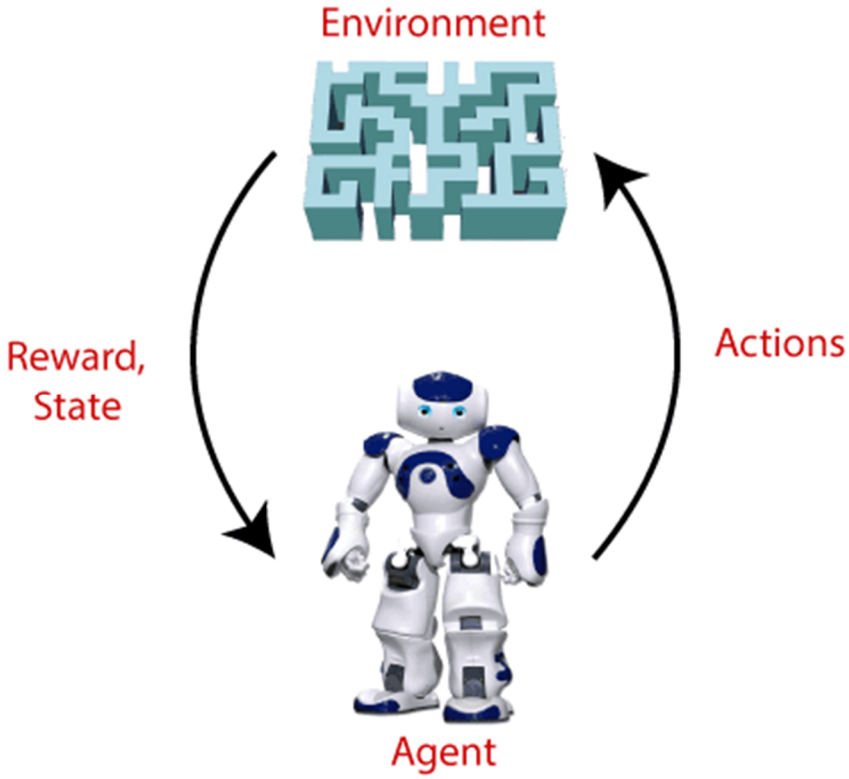

Supervised learning
Test
ML algorithm
"Cats"
"Not cats"


Unsupervised learning
No labels



Reinforcement learning

Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.



- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.

- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that ***"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."*** How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.
- It is a core part of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.
- **Example:** Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.
- The agent continues doing these three things (**take action, change state/remain in the same state, and get feedback**), and by doing these actions, he learns and explores the environment.
- The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.

Terms used in Reinforcement Learning

- **Agent():** An entity that can perceive/explore the environment and act upon it.
- **Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- **Action():** Actions are the moves taken by an agent within the environment.
- **State():** State is a situation returned by the environment after each action taken by the agent.
- **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.
- **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value():** It is expected long-term retuned with the discount factor and opposite to the short-term reward.
- **Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action (a).

Approaches to implement Reinforcement Learning

There are mainly three ways to implement reinforcement-learning in ML, which are:

1. **Value-based:**

   The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.

2. **Policy-based:**

   Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward. The policy-based approach has mainly two types of policy:

   - **Deterministic:** The same action is produced by the policy (π) at any state.
   - **Stochastic:** In this policy, probability determines the produced action.

3. **Model-based:** A virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

Applications:

In mechanical systems, reinforcement learning can be used to optimize control strategies for various tasks such as motion planning, trajectory optimization, and fault detection. For example, RL can be used to train a robotic arm to grasp objects more effectively or to learn how to walk on uneven terrain.

- Training an autonomous vehicle to navigate a road is another example of reinforcement learning. The agent interacts with the environment and receives rewards based on its actions. The goal is to learn a policy that safely and efficiently navigates the vehicle to its destination. In this example, the rewards could be defined based on the distance travelled towards the destination, the speed of the vehicle, and the number of traffic violations or accidents that occur. The agent would use trial and error to learn a policy that maximizes the rewards over time, while also taking into account the safety constraints.
- Other examples of reinforcement learning include training a robot to grasp objects, teaching a virtual assistant to understand natural language commands, and optimizing energy consumption in a smart home. Reinforcement learning can be applied in a wide range of domains where decision-making is required, and where a clear objective can be defined.

Feel free to contact me on +91-8329347107 calling / +91-9922369797 whatsapp, email ID: adp.mech@coep.ac.in and abhipatange93@gmail.com

****

AI & ML for Mechanical Engineers

Course Code: ME (DE) - 22016

Part 2: Feature extraction, selection and classification

Final Year Bachelor of Technology (Choice Based Credit System)
Mechanical Engineering
(With Effect from Academic Year 2021-22)

Lecture notes

by

Abhishek D. Patange, Ph.D.
Department of Mechanical Engineering

COEP Technological University Pune

Introduction to features

Feature is defined as a function of the basic measurement variables or attributes that specifies some quantifiable property of an object and is useful for classification and/or pattern recognition. Obtaining a good data representation is a very domain specific task and it is related to the available measurements.

Low level, high level, general, global and local features

- **Low-Level Features:** The fundamental features that can be extracted directly from an image without any object description.
- **High-Level Features:** Features that concern with finding shapes and objects in computer images and it is based on low level features.
- **General Features:** Application independent features such as color, texture, and shape.
- **Global Features:** Features that are calculated over the entire image or just regular sub-area of an image.
- **Local Features:** Features computed over a subdivision of the image bands that are resulted from image segmentation or edge detection.

Feature extraction

It is the process of transforming raw data into more informative signatures or characteristics of a system, which will most efficiently or meaningfully represent the information.

- A model for predicting the **risk of cardiac disease** may have features such as age, gender, weight, whether the person smokes, whether the person is suffering from diabetic disease, etc.
- A model for predicting whether the person is **suitable for a job** with features such as the education qualification, minimum % criteria, number of years of experience etc.
- For predicting the **shirt size** may have features such as age, gender, height, weight, etc.
- In **character recognition**, features may include histograms counting the number of black pixels along horizontal and vertical directions, number of internal holes, stroke detection and many others.
- In **speech recognition**, features for recognizing phonemes can include noise ratios, length of sounds, relative power, filter matches and many others.
- In **spam detection algorithms**, features may include the presence or absence of certain email headers, the email structure, the language, the frequency of specific terms, the grammatical correctness of the text.
- In **computer vision**, there are a large number of possible features, such as edges and objects.

Feature vector, feature space, and feature construction

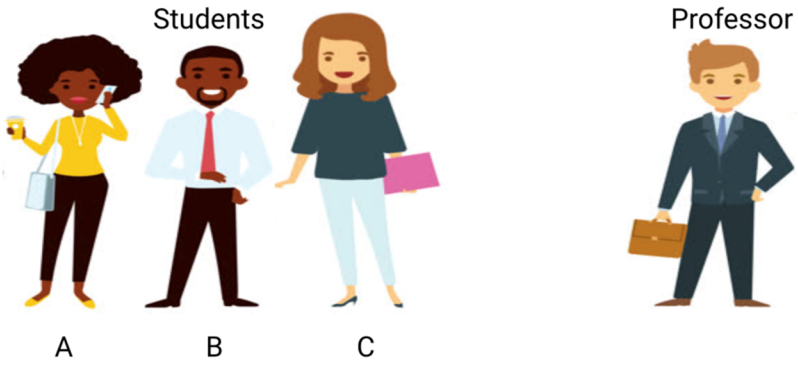
- In pattern recognition and machine learning, a **feature vector** is an n-dimensional vector of numerical features that represent some object. Many algorithms in machine learning require a numerical representation of objects, since such representations facilitate processing and statistical analysis. When representing images, the feature values might correspond to the pixels of an image, while when representing texts the features might be the frequencies of occurrence of textual terms.
- Feature vectors are equivalent to the vectors of explanatory variables used in statistical procedures such as linear regression. Feature vectors are often combined with weights using a dot product in order to construct a linear predictor function that is used to determine a score for making a prediction.
- The vector space associated with these vectors is often called the **feature space**. In order to reduce the dimensionality of the feature space, a number of dimensionality reduction techniques can be employed.
- Higher-level features can be obtained from already available features and added to the feature vector; for example, for the study of diseases the feature 'Age' is useful and is defined as *Age = 'Year of death' minus 'Year of birth'* .
- This process is referred to as **feature construction**. Feature construction is the application of a set of constructive operators to a set of existing features resulting in construction of new features. Examples of such constructive operators include checking for the equality conditions {=, ≠}, the arithmetic operators {+,−,×, /}, the array operators {max(S), min(S), average(S)} as well as other more sophisticated operators, for example count(S,C) that counts the number of features in the feature vector S satisfying some condition C or, for example, distances to other recognition classes generalized by some accepting device.
- Feature construction has long been considered a powerful tool for increasing both accuracy and understanding of structure, particularly in high-dimensional problems. Applications include studies of disease and emotion recognition from speech.

Over-fitting, under-fitting and optimum fitting

- **Overfitting** and **underfitting** are two of the most common causes of poor model accuracy. The model fit can be predicted by taking a look at the prediction error on the training and test data.
- An **underfit model** results in high prediction errors for both training and test data. An **overfit model** gives a very low prediction error on training data, but a very high prediction error on test data. Both types of models result in poor accuracy.

- An underfit model fails to significantly grasp the relationship between the input values and target variables. This may be the case when the model is too simple (i.e., the input features are not explanatory enough to describe the target well).
- An overfit model has overly memorized the data set it has seen and is unable to generalize the learning to an unseen data set. That is why an overfit model results in very poor test accuracy. Poor test accuracy may occur when the model is highly complex, i.e., the input feature combinations are in a large number and affect the model's flexibility.

Consider a math class consisting of 3 students and a professor.



Now, in any classroom, we can broadly divide the students into 3 categories. We'll talk about them one-by-one.

A

- Hobby = chating
- Not interested in class
- Doesn't pay much attention to professor

Let's say that student A resembles a student who does not like math. She is not interested in what is being taught in the class and therefore does not pay much attention to the professor and the content he is teaching.

B

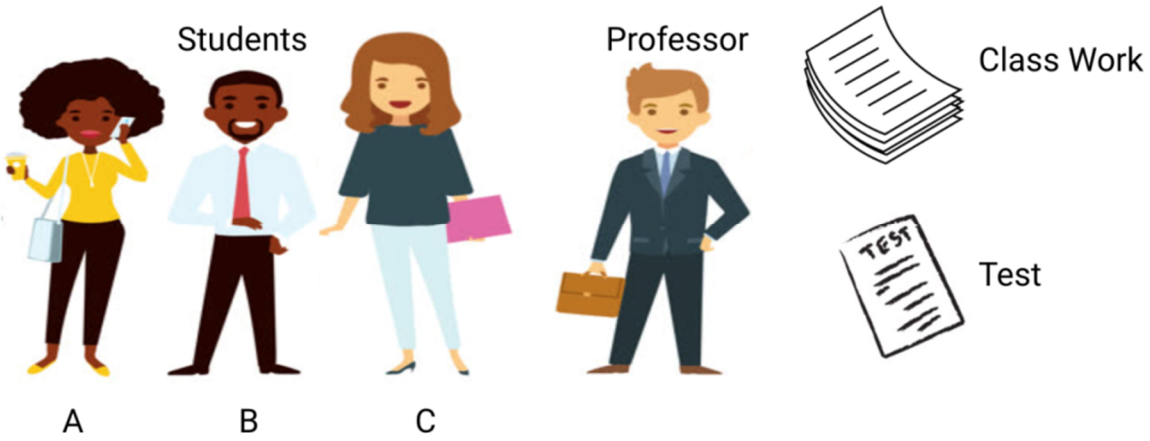
- Hobby = to be best in class.
- Mugs up everything professor says.
- Too much attention to the class work.

Let's consider student B. He is the most competitive student who focuses on memorizing each and every question being taught in class instead of focusing on the key concepts. Basically, he isn't interested in learning the problem-solving approach.

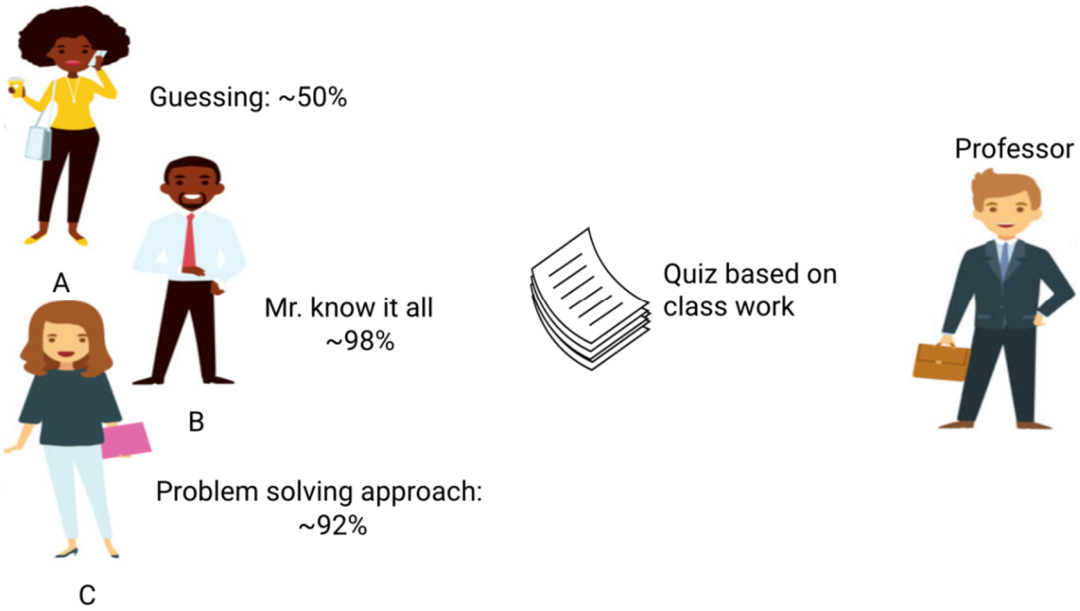
C

- Hobby = learning new things
- Eager to learn concepts.
- Pays attention to class and learns the idea behind solving a problem.

Finally, we have the ideal student C. She is purely interested in learning the key concepts and the problem-solving approach in the math class rather than just memorizing the solutions presented.



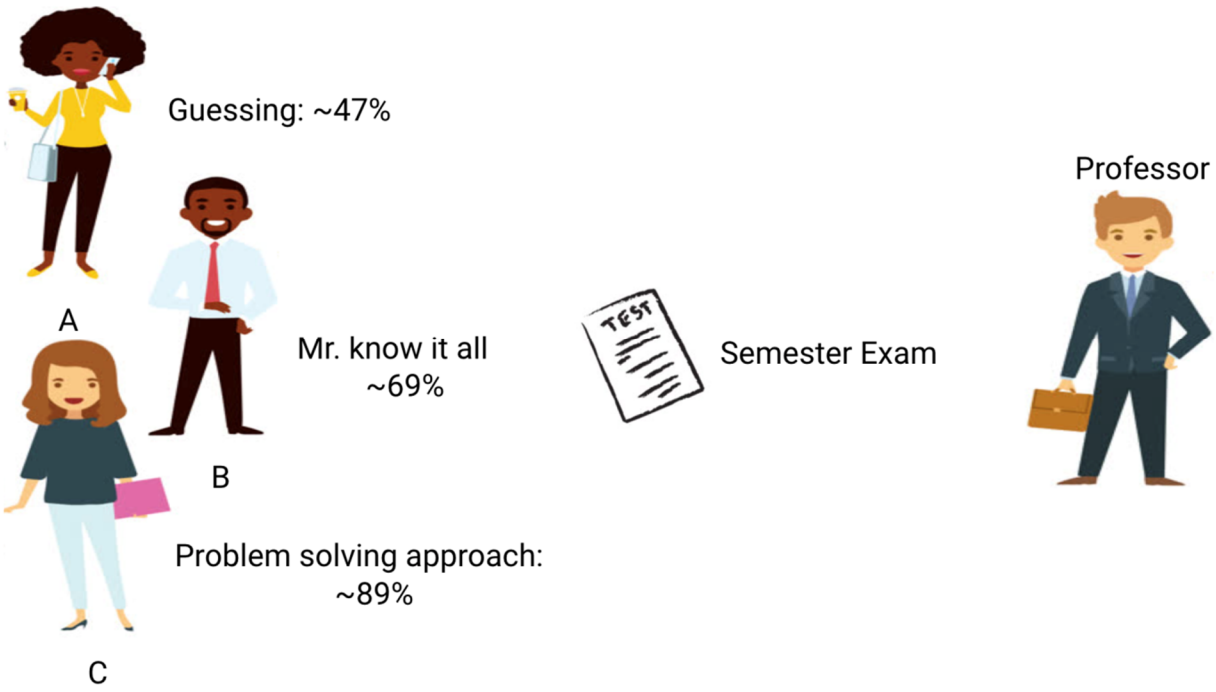
We all know from experience what happens in a classroom. The professor first delivers lectures and teaches the students about the problems and how to solve them. At the end of the day, the professor simply takes a quiz based on what he taught in the class. The obstacle comes in the semester3 tests that the school lays down. This is where new questions (unseen data) come up. The students haven't seen these questions before and certainly haven't solved them in the classroom. Sounds familiar? So, let's discuss what happens when the teacher takes a classroom test at the end of the day:



- Student A, who was distracted in his own world, simply guessed the answers and got approximately 50% marks in the test
- On the other hand, the student who memorized each and every question taught in the classroom was able to answer almost every question by memory and therefore obtained 98% marks in the class test
- For student C, she actually solved all the questions using the problem-solving approach she learned in the classroom and scored 92%

We can clearly infer that the student who simply memorizes everything is scoring better without much difficulty.

Now here's the twist. Let's also look at what happens during the monthly test, when students have to face new unknown questions which are not taught in the class by the teacher.



- In the case of student A, things did not change much and he still randomly answers questions correctly ~50% of the time.
- In the case of Student B, his score dropped significantly. Can you guess why? This is because he always memorized the problems that were taught in the class but this monthly test contained questions which he has never seen before. Therefore, his performance went down significantly
- In the case of Student C, the score remained more or less the same. This is because she focused on learning the problem-solving approach and therefore was able to apply the concepts she learned to solve the unknown questions

You might be wondering how this example relates to the problem which we encountered during the train and test scores of the decision tree classifier? Good question!

A

Not interested in learning

Class test ~50%
Test ~47%

B

Memorizing the lessons

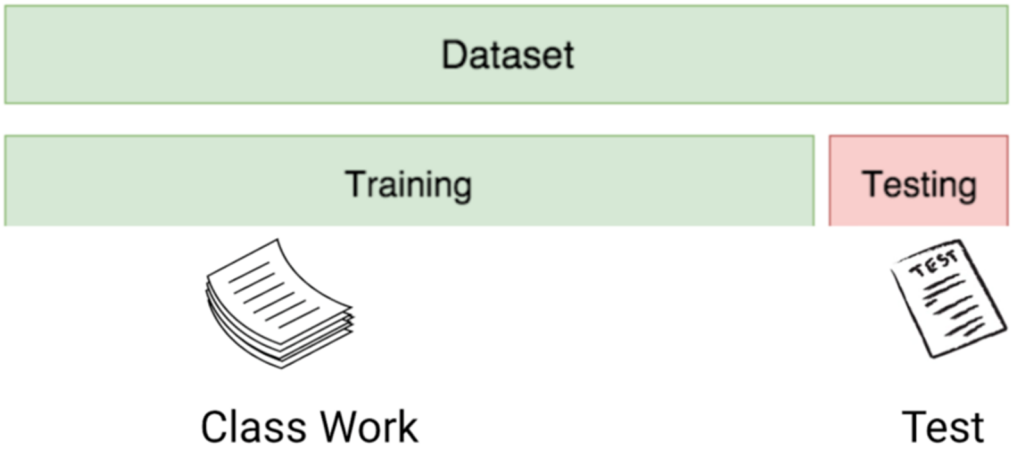
Class test ~98%
Test ~69%

C

Conceptual Learning

Class test ~92%
Test ~89%

So, let's work on connecting this example with the results of the decision tree classifier that I showed you earlier.



A

Not interested in learning

Class test ~50%
Test ~47%

Under-fit/ biased learning

B

Memorizing the lessons

Class test ~98%
Test ~69%

Over-fit/ Memorizing

C

Conceptual Learning

Class test ~92%
Test ~89%

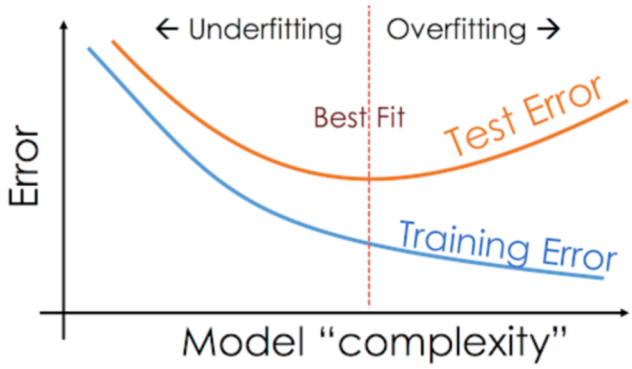
Best-fit

First, the classwork and class test resemble the training data and the prediction over the training data itself respectively. On the other hand, the semester test represents the test set from our data which we keep aside before we train our model (or unseen data in a real-world machine learning project).

Now, recall our decision tree classifier I mentioned earlier. It gave a perfect score over the training set but struggled with the test set.

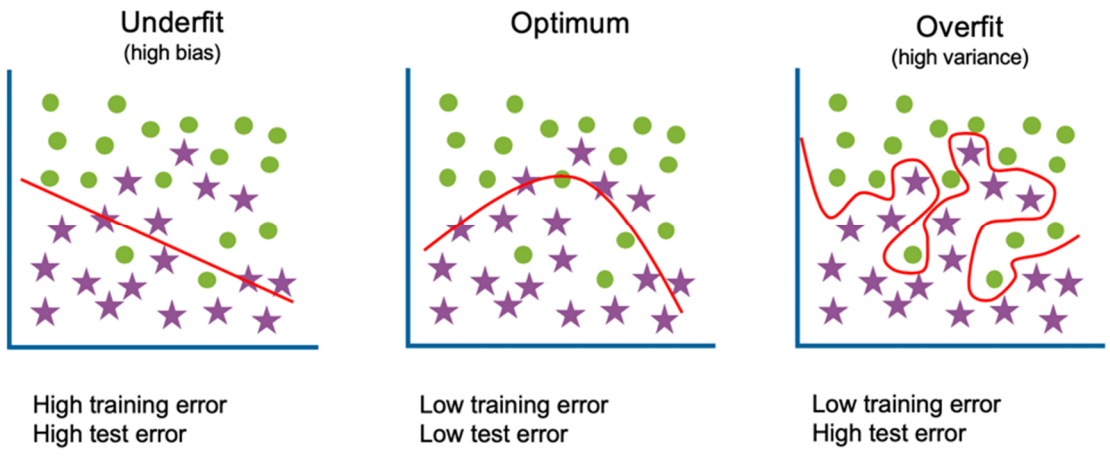
Comparing that to the student examples we just discussed, the classifier establishes an analogy with student B who tried to memorize each and every question in the training set.

Similarly, our decision tree classifier tries to learn each and every point from the training data but suffers radically when it encounters a new data point in the test set. It is not able to generalize it well. This situation where any given model is performing too well on the training data but the performance drops significantly over the test set is called an over-fitting model. For example, non-parametric models like decision trees, KNN, and other tree-based algorithms are very prone to over-fitting. These models can learn very complex relations which can result in over-fitting. The graph below summarises this concept:

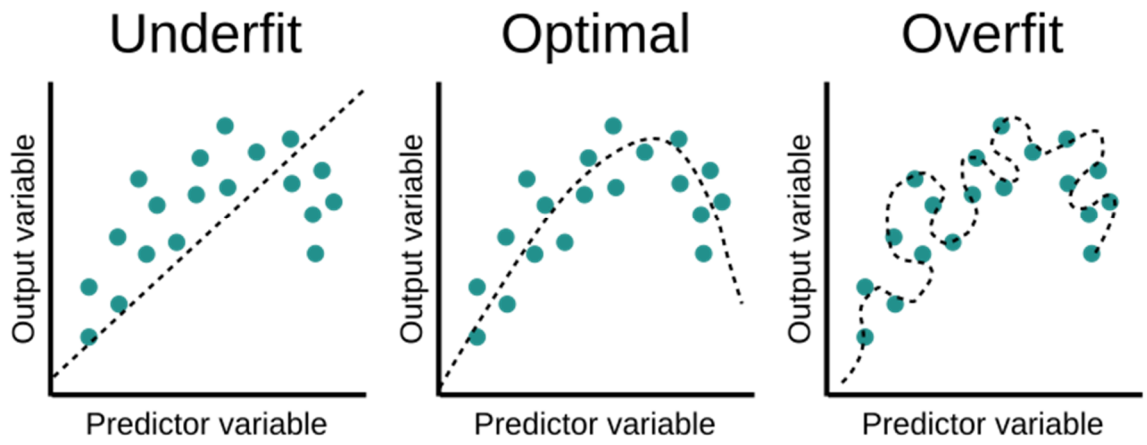


On the other hand, if the model is performing poorly over the test and the train set, then we call that an under-fitting model. An example of this situation would be building a linear regression model over non-linear data.

Over-fitting, under-fitting and optimum fitting in classification problem



Over-fitting, under-fitting and optimum fitting in regression problem



Principal component analysis (PCA)

- Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**.
- It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.
- PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.
- PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality.
- Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.***
- It is a feature extraction technique, so it contains the important variables and drops the least important variable.

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.

- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs is either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

Here are several reasons why you want to use PCA:

- **Removes correlated features**. PCA will help you remove all the features that are correlated, a phenomenon known as *multi-collinearity*. Finding features that are correlated is time consuming, especially if the number of features is large.**Improves machine learning algorithm performance**. With the number of features reduced with PCA, the time taken to train your model is now significantly reduced.
- **Reduce overfitting**. By removing the unnecessary features in your dataset, PCA helps to overcome overfitting.

On the other hand, PCA has its disadvantages:

- **Independent variables are now less interpretable**. PCA reduces your features into smaller number of components. Each component is now a linear combination of your original features, which makes it less readable and interpretable.
- **Information loss**. Data loss may occur if you do not exercise care in choosing the right number of components.
- **Feature scaling**. Because PCA is a *variance maximizing* exercise, PCA requires features to be scaled prior to processing.

PCA is related to the set of operations in the Pearson correlation, so it inherits similar assumptions and limitations:

- **PCA assumes a correlation between features**. If the features (or dimensions or columns, in tabular data) are not correlated, PCA will be unable to determine principal components.
- **PCA is sensitive to the scale of the features**. Imagine we have two features - one takes values between 0 and 1000, while the other takes values between 0 and 1. PCA will be extremely biased towards the first feature being the first principle component, regardless

of the *actual* maximum variance within the data. This is why it's so important to standardize the values first.

- **PCA is not robust against outliers**. Similar to the point above, the algorithm will be biased in datasets with strong outliers. This is why it is recommended to remove outliers before performing PCA.
- **PCA assumes a linear relationship between features**. The algorithm is not well suited to capturing non-linear relationships. That's why it's advised to turn non-linear features or relationships between features into linear, using the standard methods such as log transforms.
- **Technical implementations often assume no missing values**. When computing PCA using statistical software tools, they often assume that the feature set has no missing values (no empty rows). Be sure to remove those rows and/or columns with missing values, or impute missing values with a close approximation (e.g. the mean of the column).

Statistical features and mathematical expressions

The mathematical expressions for descriptive statistical attributes:

Attribute	Mathematical Expression
Kurtosis	It is an estimate of the 'tailedness' of the probability distribution of a real-valued random variable. $\left\{\frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum\left(\frac{x_i-\bar{x}}{S_d}\right)^4\right\}-\frac{3(n-1)^2}{(n-2)(n-3)}$
Standard Error	Standard Error is a measure of the deviation of the sample means from the population. The standard error of a sample statistic is an estimate of the standard deviation of the sampling distribution of that sample statistic. It helps you to find out confidence intervals for that statistic at different significance levels. $\sqrt{\frac{1}{n-2}\left(\sum(y-\bar{y})^2-\frac{\sum[(x-\bar{x})(y-\bar{y})]^2}{\sum(x-\bar{x})^2}\right)}$
Maximum value	It is the highest data point value.
Skewness	It defines the inclination of the spread of data on either side. $\frac{n}{(n-1)(n-2)}\sum\left(\frac{x_i-\bar{x}}{S_d}\right)^3$
Minimum value	It is the lowest data point value.
Range	It is an estimate of subtraction between maximum and minimum values of

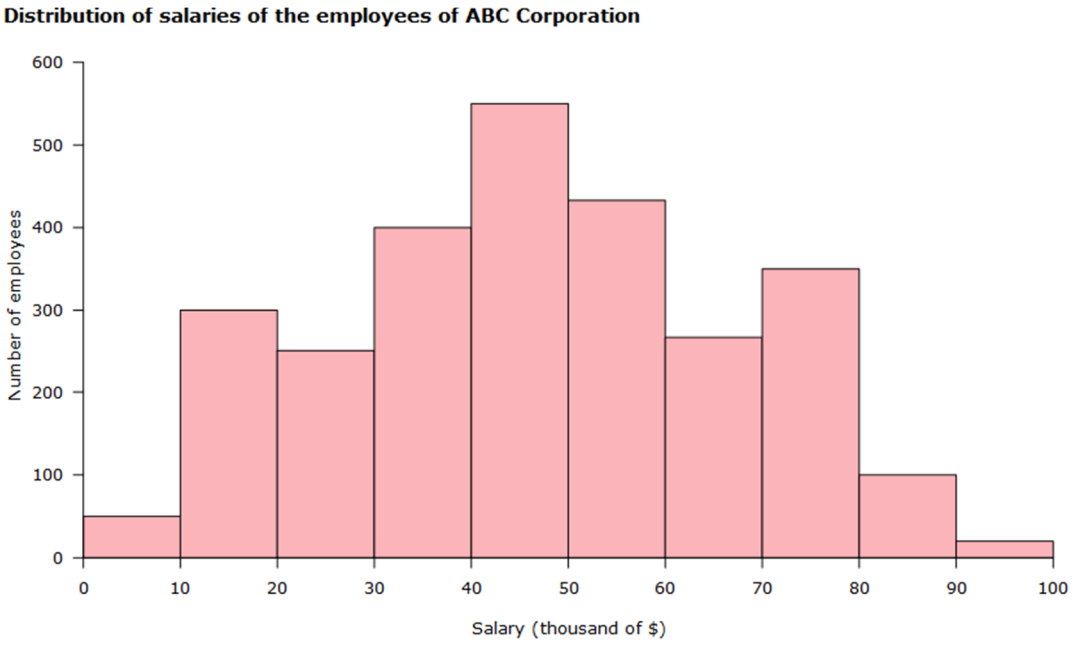
data points.

Count	It is an estimate of the number of data points in each sample.
Summation	It is an estimate of the sum of all feature values for each sample.
Variance	It is the expectation of the squared deviation of a random variable from its mean. $\frac{n\sum x^2 - (\sum x)^2}{n(n-1)}$
Standard Deviation	It is a measure of the amount of variation or dispersion of a set of values. $\sqrt{\frac{n\sum x^2 - (\sum x)^2}{n(n-1)}}$
Mode	It is an estimate of the number which occurs most frequently in a set of data points.
Median	It is an estimate of the middle value segregating the higher and lower splits of a data set.
Mean	It is an estimate of the average (arithmetic) of a set of data points. $\bar{x} = \frac{x_1 + x_2 + x_3 + \dots\dots + x_n}{n}$
Impulse factor	The impulse factor estimates the impact created by considering the ratio of maximum value and average value. $\frac{Maximum\ value\ of\ x}{\bar{x}}$
K-factor	The K-factor is a product of root mean square value and maximum value. $\frac{\sqrt{\frac{x_1{}^2 + x_2{}^2 + x_3{}^2 + \dots\dots + x_n{}^2}{n}}}{Maximum\ value\ of\ x}$
Shape factor	The shape factor is the ratio of root mean square value and average value. $\frac{\sqrt{\frac{x_1{}^2 + x_2{}^2 + x_3{}^2 + \dots\dots + x_n{}^2}{n}}}{\bar{x}}$

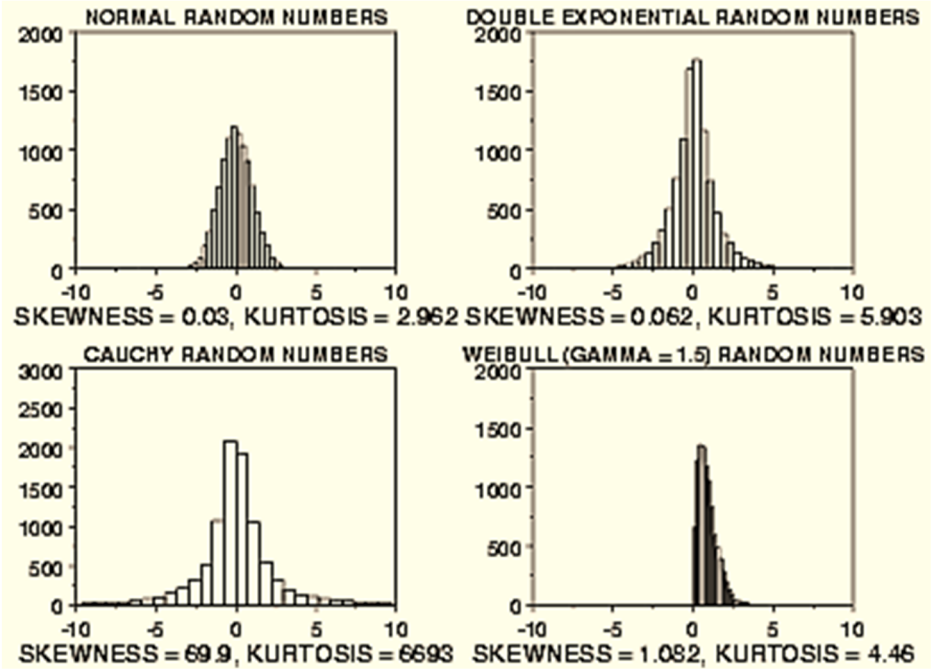
Histogram features

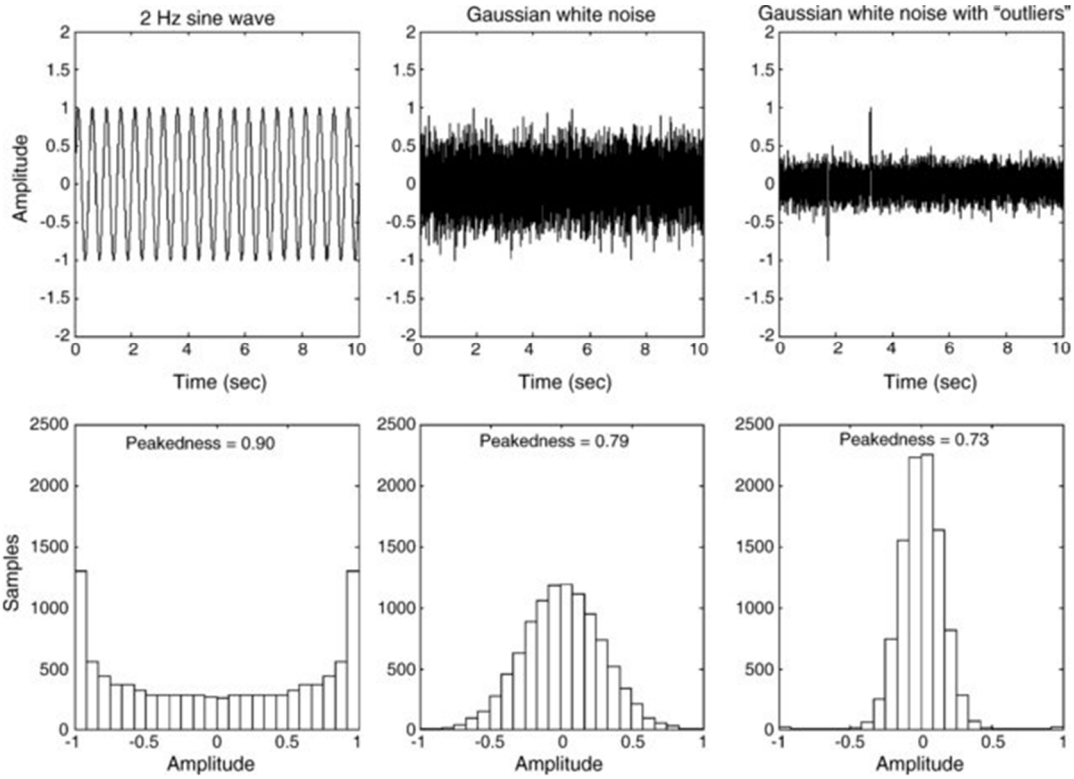
The histogram is a popular graphing tool. It is used to summarize discrete or continuous data that are measured on an interval scale. It is often used to illustrate the major features of the distribution of the data in a convenient form. It is also useful when dealing with large data sets (greater than 100 observations). It can help detect any unusual observations (outliers) or any gaps in the data. A histogram divides up the range of possible values in a data set into classes or groups. For each group, a rectangle is constructed with a base length equal to the range of values in that specific group and a length equal to the number of observations falling into that group. A histogram has an appearance similar to a vertical bar chart, but there are no gaps between the bars. Generally, a histogram will have bars of equal

width. Chart below is an example of a histogram that shows the distribution of salary, a continuous variable, of the employees of a corporation.



A Normal Distribution:	In a normal distribution, points on both sides of the average are alike.
A Random Distribution:	There is no pattern in a random distribution histogram and thus has several peaks. The reason behind this could be that the data properties were combined.
A Bimodal Distribution:	In a bimodal distribution, the data are separately analyzed as a normal distribution. Therefore they are represented as two different peaks.



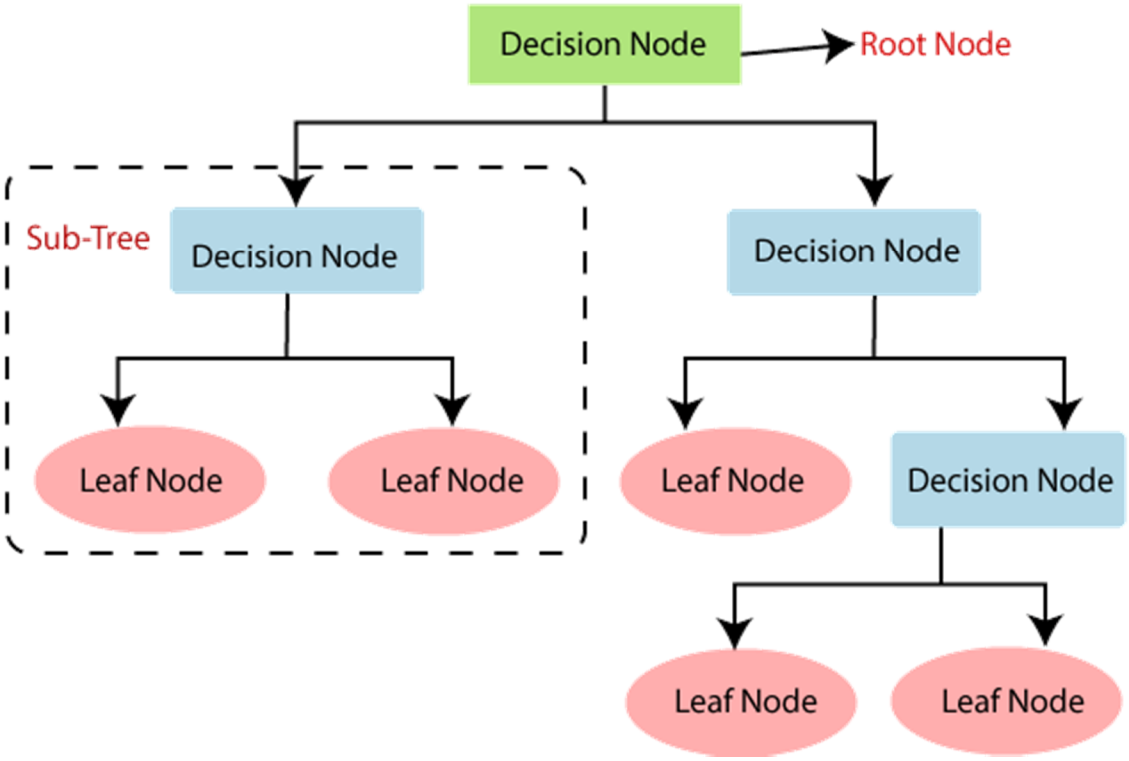


A Right-Skewed Distribution:	A right-skewed distribution, also known as positively skewed distribution, is where a large number of data values occur on the left side whereas a fewer number of data values occur on the right side. A right-skewed distribution occurs when the data on the left-hand side of the Histogram has a low range boundary, for example, 0.
A Left-Skewed Distribution:	A left-skewed distribution is also known as a negatively skewed distribution. In a left-skewed distribution, a large number of data values appear on the right side whereas a fewer number of data values occur on the left side. A right-skewed distribution occurs when the data has a low range boundary on the right-hand side of the histogram, for example, 100.

Feature selection

Feature selection is the process of selecting a subset of the constructed features according to a certain criteria. Reducing the number of features as well as removing the irrelevant, redundant, or noisy data increases the efficiency and effectiveness of the implemented algorithms. This is an important and frequently used dimensionality reduction technique for various applications, e.g. data mining, face recognition

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- **It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.**
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub-trees. Below diagram explains the general structure of a decision tree.



- It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision tree terminology

The decision tree comprises of root node, leaf node, branch nodes, parent/child node etc. following is the explanation of this terminology.

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

Entropy reduction, information gain and Gini index in decision tree

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s) = – P(yes)log2 P(yes) – P(no) log2 P(no) where, S= Total number of samples, P(yes)= probability of yes, P(no)= probability of no

Information Gain: Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S) – [(Weighted Average) * Entropy (each feature)]

Gini Index: Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index. Gini index can be calculated using the formula:

Gini Index= $1 - \sum_j P_j^2$

It only creates binary splits, and the CART uses Gini index to create binary splits.

Problems on PCA

The steps involved in PCA Algorithm are as follows-

Step-01: Get data.

Step-02: Compute the mean vector (μ).

Step-03: Subtract mean from the given data.

Step-04: Calculate the covariance matrix.

Step-05: Calculate the eigen vectors and eigen values of the covariance matrix.

Step-06: Choosing components and forming a feature vector.

Step-07: Deriving the new data set.

Problem 1

Given data = { 2, 3, 4, 5, 6, 7 ; 1, 5, 3, 6, 7, 8 }. Compute the principal component using PCA.

OR

Consider the two dimensional patterns (2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (7, 8).

Compute the principal component using PCA Algorithm.

OR

Compute the principal component of following data-

CLASS 1	CLASS 2
X = 2 , 3 , 4	X = 5 , 6 , 7
Y = 1 , 5 , 3	Y = 6 , 7 , 8

Solution-

We use the above discussed PCA Algorithm-

Step-01:

Get data and given feature vectors are-

- $x_1 = (2, 1)$
- $x_2 = (3, 5)$
- $x_3 = (4, 3)$
- $x_4 = (5, 6)$
- $x_5 = (6, 7)$
- $x_6 = (7, 8)$

$$\begin{bmatrix}2\\1\end{bmatrix}\begin{bmatrix}3\\5\end{bmatrix}\begin{bmatrix}4\\3\end{bmatrix}\begin{bmatrix}5\\6\end{bmatrix}\begin{bmatrix}6\\7\end{bmatrix}\begin{bmatrix}7\\8\end{bmatrix}$$

Step-02:

Calculate the mean vector (μ).

Mean vector (μ)

= ((2 + 3 + 4 + 5 + 6 + 7) / 6, (1 + 5 + 3 + 6 + 7 + 8) / 6)

= (4.5, 5)

$$\text{Mean vector } (\mu) = \begin{bmatrix}4.5\\5\end{bmatrix}$$

Step-03:

Subtract mean vector (μ) from the given feature vectors.

- $x_1 - \mu = (2 - 4.5, 1 - 5) = (-2.5, -4)$
- $x_2 - \mu = (3 - 4.5, 5 - 5) = (-1.5, 0)$
- $x_3 - \mu = (4 - 4.5, 3 - 5) = (-0.5, -2)$
- $x_4 - \mu = (5 - 4.5, 6 - 5) = (0.5, 1)$
- $x_5 - \mu = (6 - 4.5, 7 - 5) = (1.5, 2)$
- $x_6 - \mu = (7 - 4.5, 8 - 5) = (2.5, 3)$

Feature vectors (x_i) after subtracting mean vector (μ) are

$$\begin{bmatrix}-2.5\\-4\end{bmatrix}\begin{bmatrix}-1.5\\0\end{bmatrix}\begin{bmatrix}-0.5\\-2\end{bmatrix}\begin{bmatrix}0.5\\1\end{bmatrix}\begin{bmatrix}1.5\\2\end{bmatrix}\begin{bmatrix}2.5\\3\end{bmatrix}$$

Step-04:

Calculate the covariance matrix.

$$\text{Covariance Matrix} = \frac{\sum (x_i - \mu)(x_i - \mu)^t}{n}$$

Now,

$$m_1 = (x_1 - \mu)(x_1 - \mu)^t = \begin{bmatrix}-2.5\\-4\end{bmatrix}\begin{bmatrix}-2.5 & -4\end{bmatrix} = \begin{bmatrix}6.25 & 10\\10 & 16\end{bmatrix}$$

$$m_2 = (x_2 - \mu)(x_2 - \mu)^t = \begin{bmatrix}-1.5\\0\end{bmatrix}\begin{bmatrix}-1.5 & 0\end{bmatrix} = \begin{bmatrix}2.25 & 0\\0 & 0\end{bmatrix}$$

$$m_3 = (x_3 - \mu)(x_3 - \mu)^t = \begin{bmatrix}-0.5\\-2\end{bmatrix}\begin{bmatrix}-0.5 & -2\end{bmatrix} = \begin{bmatrix}0.25 & 1\\1 & 4\end{bmatrix}$$

$$m_4 = (x_4 - \mu)(x_4 - \mu)^t = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 0.5 & 1 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$m_5 = (x_5 - \mu)(x_5 - \mu)^t = \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 1.5 & 2 \end{bmatrix} = \begin{bmatrix} 2.25 & 3 \\ 3 & 4 \end{bmatrix}$$

$$m_6 = (x_6 - \mu)(x_6 - \mu)^t = \begin{bmatrix} 2.5 \\ 3 \end{bmatrix} \begin{bmatrix} 2.5 & 3 \end{bmatrix} = \begin{bmatrix} 6.25 & 7.5 \\ 7.5 & 9 \end{bmatrix}$$

Now,

Covariance matrix

$= (m_1 + m_2 + m_3 + m_4 + m_5 + m_6) / 6$

On adding the above matrices and dividing by 6, we get-

$$\text{Covariance Matrix} = \frac{1}{6} \begin{bmatrix} 17.5 & 22 \\ 22 & 34 \end{bmatrix}$$

$$\text{Covariance Matrix} = \begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix}$$

Step-05:

Calculate the eigen values and eigen vectors of the covariance matrix.

λ is an eigen value for a matrix M if it is a solution of the characteristic equation $|M – \lambda I| = 0$.

So, we have-

$$\begin{vmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = 0$$

$$\begin{vmatrix} 2.92 - \lambda & 3.67 \\ 3.67 & 5.67 - \lambda \end{vmatrix} = 0$$

From here,

$(2.92 – \lambda)(5.67 – \lambda) – (3.67 \times 3.67) = 0$

$16.56 – 2.92\lambda – 5.67\lambda + \lambda^2 – 13.47 = 0$

$\lambda^2 - 8.59\lambda + 3.09 = 0$

Solving this quadratic equation, we get $\lambda = 8.22, 0.38$

Thus, two eigen values are $\lambda_1 = 8.22$ and $\lambda_2 = 0.38$.

Clearly, the second eigen value is very small compared to the first eigen value.

So, the second eigen vector can be left out.

Eigen vector corresponding to the greatest eigen value is the principal component for the given data set.

So. we find the eigen vector corresponding to eigen value λ_1.

We use the following equation to find the eigen vector-

$MX = \lambda X$

where-

- M = Covariance Matrix
- X = Eigen vector
- λ = Eigen value

Substituting the values in the above equation, we get-

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \end{bmatrix} = 8.22 \begin{bmatrix} X1 \\ X2 \end{bmatrix}$$

Solving these, we get-

$2.92X_1 + 3.67X_2 = 8.22X_1$

$3.67X_1 + 5.67X_2 = 8.22X_2$

On simplification, we get-

$5.3X_1 = 3.67X_2$ .....(1)

$3.67X_1 = 2.55X_2$ .....(2)

From (1) and (2), $\mathbf{X_1 = 0.69X_2}$

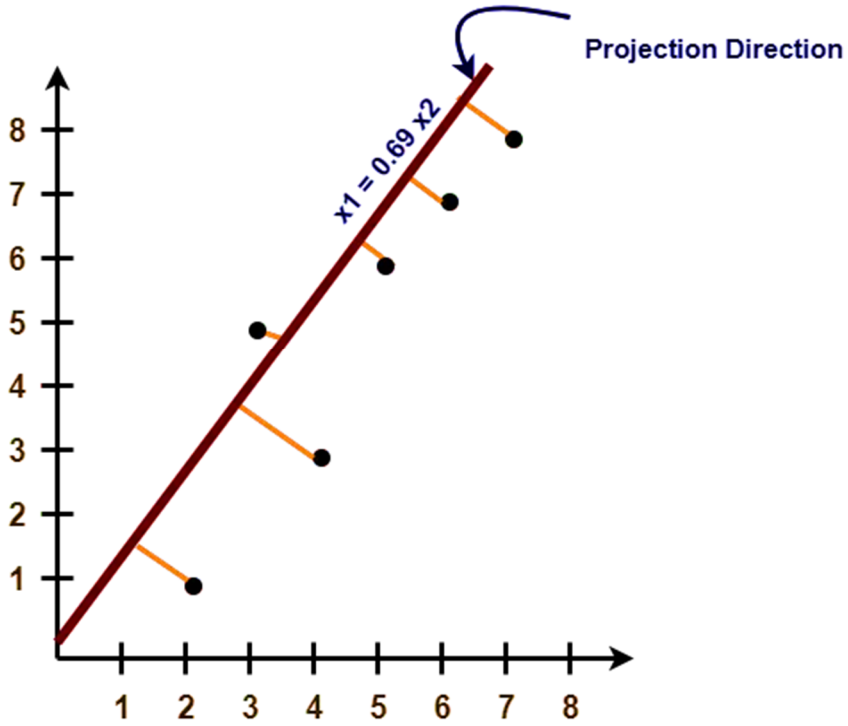
From (2), the eigen vector is-

$$\text{Eigen Vector :} \quad \begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

Thus, principal component for the given data set is-

$$\text{Principal Component :} \quad \begin{bmatrix} X1 \\ X2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

Lastly, we project the data points onto the new subspace as



Problem 2

Consider the following dataset which shows temperature of a thermal system at two different locations.

Temperature 1 (^{O}C)	2.5	0.5	2.2	1.9	3.1	2.3	2.0	1.0	1.5	1.1
Temperature 2 (^{O}C)	2.4	0.7	2.9	2.2	3.0	2.7	1.6	1.1	1.6	0.9

Standardize this data. Find Eigen values and Eigen vectors. Arrange Eigen values. Form feature vector. Transform original data and reconstruct it. Find principal components.

Step 1: Standardize the Dataset

Mean for x_1= 1.81 = x_{1mean}

Mean for x_2= 1.91 = x_{2mean}

We will change the dataset.

x1	0.69	-1.31	0.39	0.09	1.29	0.49	0.19	-0.81	-0.31	-0.71
x2	0.49	-1.21	0.99	0.29	1.09	0.79	-0.31	-0.81	-0.31	-1.01

Step 2: Find the Eigenvalues and eigenvectors

Correlation Matrix c = $C = \left(\frac{X \cdot X^T}{N-1}\right)$

where, X is the Dataset Matrix (In this numerical, it is a 10 X 2 matrix)

X^T is the transpose of the X (In this numerical, it is a 2 X 10 matrix) and N is the number of elements = 10

So, $C = \left(\frac{X \cdot X^T}{10-1}\right) = \left(\frac{X \cdot X^T}{9}\right)$

{So in order to calculate the Correlation Matrix, we have to do the multiplication of the Dataset Matrix with its transpose}

$$C = \begin{bmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{bmatrix}$$

Using the equation, **| C – λI | = 0– equation (i)** where { \lambda is the eigenvalue and I is the Identity Matrix }

So solving equation (i)

$$\begin{bmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{vmatrix} 0.616556 - \lambda & 0.615444 \\ 0.615444 & 0.716556 - \lambda \end{vmatrix} = 0$$

Taking the determinant of the left side, we get

$$0.44180 - 0.616556\lambda - 0.716556\lambda + \lambda^2 - 0.37877 = 0$$

$$\lambda^2 - 1.33311\lambda + 0.06303 = 0$$

We get two values for λ, that are **(λ_1) = 1.28403 and (λ_2) = 0.0490834**. Now we have to find the eigenvectors for the eigenvalues λ_1 **and** λ_2

To find the eigenvectors from the eigenvalues, we will use the following approach:

First, we will find the eigenvectors for the eigenvalue 1.28403 by using the equation $C \cdot X = \lambda \cdot X$

$$\begin{bmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = 1.28403 \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 0.616556x + 0.615444y \\ 0.615444x + 0.716556y \end{bmatrix} = \begin{bmatrix} 1.28403x \\ 1.28403y \end{bmatrix}$$

Solving the matrices, we get

0.616556x + 0.615444y = 1.28403x ; **x = 0.922049 y**

(x and y belongs to the matrix X) so if we put y = 1, x comes out to be 0.922049. So now the updated X matrix will look like:

$$X = \begin{bmatrix} 0.922049 \\ 1 \end{bmatrix}$$

IMP: Till now we haven't reached to the eigenvectors, we have to a bit of modifications in the X matrix. They are as follows:

A. Find the square root of the sum of the squares of the element in X matrix i.e.

$$\sqrt{0.922049^2 + 1^2} = \sqrt{0.850174 + 1} = \sqrt{1.850174} = 1.3602$$

B. Now divide the elements of the X matrix by the number 1.3602 (just found that)

$$\begin{bmatrix} \frac{0.922049}{1.3602} \\ \\ \frac{1}{1.3602} \end{bmatrix} = \begin{bmatrix} 0.67787 \\ 0.73518 \end{bmatrix}$$

So now we found the eigenvectors for the eigenvector λ_1, they are 0.67787 and 0.73518

Secondly, we will find the eigenvectors for the eigenvalue 0.0490834 by using the equation {Same approach as of previous step)

$$\begin{bmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = 0.0490834 \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} 0.616556x + 0.615444y \\ 0.615444x + 0.716556y \end{bmatrix} = \begin{bmatrix} 0.0490834x \\ 0.0490834y \end{bmatrix}$$

Solving the matrices, we get

0.616556x + 0.615444y = 0.0490834x; **y = -0.922053**

(x and y belongs to the matrix X) so if we put x = 1, y comes out to be -0.922053 So now the updated X matrix will look like:

$$X = \begin{bmatrix} 1 \\ -0.922053 \end{bmatrix}$$

IMP: Till now we haven't reached to the eigenvectors, we have to a bit of modifications in the X matrix. They are as follows:

A. Find the square root of the sum of the squares of the elements in X matrix i.e.

$$\sqrt{1^2 + (-0.922053)^2} = \sqrt{1 + 0.85018} = \sqrt{1.85018} = 1.3602$$

B. Now divide the elements of the X matrix by the number 1.3602 (just found that)

$$\begin{bmatrix} \frac{1}{1.3602} \\ \frac{-0.922053}{1.3602} \end{bmatrix} = \begin{bmatrix} 0.735179 \\ 0.677873 \end{bmatrix}$$

So now we found the eigenvectors for the eigenvector \lambda_2, they are 0.735176 and 0.677873

Sum of eigenvalues (λ_1) and (λ_2) = 1.28403 + 0.0490834 = 1.33 = Total Variance {Majority of variance comes from λ_1}

Step 3: Arrange Eigenvalues

The eigenvector with the highest eigenvalue is the Principal Component of the dataset. So in this case, eigenvectors of lambda1 are the principal components.

{Basically in order to complete the numerical we have to only solve till this step, but if we have to prove why we have chosen that particular eigenvector we have to follow the steps from 4 to 6}

Step 4: Form Feature Vector

$\begin{bmatrix} 0.677873 & 0.735179 \\ 0.735179 & -0.677879 \end{bmatrix}$ This is the FEATURE VECTOR for Numerical

Where first column are the eigenvectors of λ_1 & second column are the eigenvectors of λ_2

Step 5: Transform Original Dataset

Use the equation Z = X V

$$Z \quad \begin{bmatrix} 0.69 & 0.49 \\ -1.31 & -1.21 \\ 0.39 & 0.99 \\ 0.09 & 0.29 \\ 1.29 & 1.09 \\ 0.49 & 0.79 \\ 0.19 & -0.31 \\ -0.81 & -0.81 \\ -0.31 & -0.31 \\ -0.71 & -1.01 \end{bmatrix} \cdot \begin{bmatrix} 0.677873 & 0.735179 \\ 0.735179 & -0.677879 \end{bmatrix} = \begin{bmatrix} 0.8297008 & 0.17511574 \\ -1.77758022 & -0.14285816 \\ 0.99219768 & -0.38437446 \\ 0.27421048 & -0.13041706 \\ 1.67580128 & 0.20949934 \\ 0.91294918 & -0.17528196 \\ -1.14457212 & -0.04641786 \\ -0.43804612 & -0.01776486 \\ -1.22382.62 & 0.16267464 \end{bmatrix} =$$

Step 6: Reconstructing Data

Use the equation X = $Z * V^T$ ($V^T$is Transpose of V), X = Row Zero Mean Data

$$\begin{bmatrix} 0.8297008 & 0.17511574 \\ -1.77758022 & -0.14285816 \\ 0.99219768 & -0.38437446 \\ 0.27421048 & -0.13041706 \\ 1.67580128 & 0.20949934 \\ 0.91294918 & -0.17528196 \\ -1.14457212 & -0.04641786 \\ -0.43804612 & -0.01776486 \\ -1.22382.62 & 0.16267464 \end{bmatrix} \cdot \begin{bmatrix} 0.677873 & 0.735179 \\ 0.735176 & -0.677879 \end{bmatrix} = \begin{bmatrix} 0.6899999766573 & 0.4899999834233 \\ -1.3099999556827 & -1.2099999590657 \\ 0.389999968063 & 0.9899999665083 \\ 0.0899999969553 & 0.2899999901893 \\ 0.61212695653593 & 0.35482096313253 \\ 0.4899999834233 & 0.7899999732743 \\ 0.189999935723 & -0.309999995127 \\ -0.8099999725977 & -0.8099999725977 \\ -0.3099999895127 & -0.3099999895127 \\ -0.7099999759807 & -1.0099999658317 \end{bmatrix}$$

So in order to reconstruct the original data, we follow:

Row Original DataSet = Row Zero Mean Data + Original Mean

$$\begin{bmatrix} 0.6899999766573 & 0.4899999834233 \\ -1.3099999556827 & -1.2099999590657 \\ 0.389999968063 & 0.9899999665083 \\ 0.0899999969553 & 0.2899999901893 \\ 0.61212695653593 & 0.35482096313253 \\ 0.4899999834233 & 0.7899999732743 \\ 0.189999935723 & -0.309999995127 \\ -0.8099999725977 & -0.8099999725977 \\ -0.3099999895127 & -0.3099999895127 \\ -0.7099999759807 & -1.0099999658317 \end{bmatrix} + \begin{bmatrix} 1.81 & 1.91 \end{bmatrix} = \begin{bmatrix} 2.49 & 2.39 \\ 0.5 & 0.7 \\ 2.19 & 2.89 \\ 1.89 & 2.19 \\ 3.08 & 2.99 \\ 2.30 & 2.7 \\ 2.01 & 1.59 \\ 1.01 & 1.11 \\ 1.5 & 1.6 \\ 1.1 & 0.9 \end{bmatrix}$$

So for the eigenvectors of first eigenvalue, data can be reconstructed similar to the original dataset. Thus we can say that the Principal Component of the dataset is λ_1 is 1.28403 followed by λ_2 **that is 0.0490834**

Problems on calculating entropy and information gain for decision trees

Problem 1

Consider following dataset and find following things.

a. Calculate entropy before splitting any of the attribute
b. Calculate the information gain for the Outlook attribute
c. Calculate the information gain for the Temp attribute
d. Calculate the information gain for the Humidity attribute
e. Calculate the information gain for the wind attribute

Day	Outlook	Temp	Humidity	Wind	Play Volleyball
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Rain	Mild	High	Strong	No
D13	Overcast	Hot	Normal	Weak	Yes
D14	Overcast	Mild	High	Strong	Yes

a. Entropy before splitting any of the attribute

We can see that we have 5 Noes (or negatives) and 9 Yeses (or positives). The total number of entries is 14.

The entropy of the whole dataset is

$$Entropy(S) = -\frac{9}{14}log_2\left(\frac{9}{14}\right) - -\frac{5}{14}log_2\left(\frac{5}{14}\right) = 0.94$$

b. Information gain for the Outlook attribute

Outlook has 3 features: Sunny, Overcast and Rain

So we will calculate the entropy of each of these features (Sv) as follows:

We have 5 Sunny features for Outlook:

3 negative Sunny Outlooks (When Play Volleyball is No).

2 positive Sunny Outlooks (When Play Volleyball is Yes)

Let's calculate:

$$Entropy(S_{Sunny}) = -\frac{2}{5}log_2\left(\frac{2}{5}\right) - -\frac{3}{5}log_2\left(\frac{3}{5}\right) = 0.97$$

We have 4 Overcast features for Outlook:

0 negative Overcast features

4 positive Overcast features

Let's calculate:

$$Entropy(S_{Overcast}) = -\frac{4}{4}log_2\left(\frac{4}{4}\right) - -\frac{0}{4}log_2\left(\frac{0}{4}\right) = 0$$

We have 5 Rain features for Outlook:

2 negative Rain features

3 positive Rain features

Let's calculate:

$$Entropy(S_{Rain}) = -\frac{3}{5}log_2\left(\frac{3}{5}\right) - -\frac{2}{5}log_2\left(\frac{2}{5}\right) = 0.97$$

The information gain for Outlook

We have the following Entropies:

Entropy(S) = 0.94

Entropy(S_{Sunny}) = 0.97

Entropy($S_{Overcast}$) = 0

Entropy(S_{Rain}) = 0.97

We use the formula for information gain to calculate the gain.

$$Gain(S, a) = Entropy(S) - \sum_{v \epsilon values(a)} \frac{|S_v|}{|S|} Entropy(S_v)$$

So:

$$Gain(S,a) = Entropy(S) - \frac{5}{14}Entropy(S_{Sunny}) - \frac{4}{14}Entropy(S_{overcast}) - \frac{5}{14}Entropy(S_{Rain})$$

$$Gain(S, Outlook) = 0.94 - \frac{5}{14}(0.971) - \frac{4}{14}(0) - \frac{5}{14}(0.97) = 0.24$$

Information gain for Outlook is 0.24.

c. Information gain for the Temp attribute

Temp has 3 features: Hot, Mild, Cool

Since we already have the entropy for the entire dataset (Entropy(S)), we will calculate the entropy of each feature (Entropy (Sv)) of Temp, just as we did with Outlook.

The entropy of Hot:

$$Entropy(S_{Hot}) = -\frac{2}{4}log_2\left(\frac{2}{4}\right) - \frac{2}{4}log_2\left(\frac{2}{4}\right) = 1.0$$

The entropy of Mild:

$$Entropy(S_{Mild}) = -\frac{4}{6}log_2\left(\frac{4}{6}\right) - \frac{2}{6}log_2\left(\frac{2}{6}\right) = 0.91$$

The entropy of Cool:

$$Entropy(S_{Cool}) = -\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right) = 0.81$$

Calculate the information gain for the Temp feature:

$$Gain(S, Temp) = 0.94 - \frac{4}{14}(1.0) - \frac{6}{14}(0.91) - \frac{4}{14}(0.81) = 0.03$$

Information gain for Temp is 0.03.

d. Information gain for the humidity attribute

Gain(S, Humidity) = 0.15

e. Information gain for the wind attribute

Gain(S, Wind) = 0.04

So by comparing information gain

Gain(S, Outlook) = 0.24

Gain(S, Temp) = 0.03

Gain(S, Humidity) = 0.15

Gain(S, Wind) = 0.04

Since, outlook gives the highest information about our target variable from the information gain values. It will act as the root node of our tree from where the splitting will begin.

Problem 2

In the below mini-dataset, the label we're trying to predict is the type of fruit. This is based off the size, color, and shape variables.

Fruit	Size	Color	Shape
Watermelon	Big	Green	Round
Apple	Medium	Red	Round
Banana	Medium	Yellow	Thin
Grape	Small	Green	Round
Grapefruit	Medium	Yellow	Round
Lemon	Small	Yellow	Round

Calculate the information gained if we select the *color* variable.

3 out of the 6 records are yellow, 2 are green, and 1 is red. Proportionally, the probability of a yellow fruit is 3 / 6 = 0.5; 2 / 6 = 0.333 for green, and 1 / 6 = 0.1666 for red. Using the formula from above, we can calculate it like this:

Information gain = - ([3/6 * $\log_2$(3/6)] + [2/6 * $\log_2$(2/6)] + [1/6 * $\log_2$(1/6)]) **= 1.459148**

Calculate the information gained if we select the *size* variable.

Information gain = - ([3/6 * $\log_2$(3/6)] + [2/6 * $\log_2$(2/6)] + [1/6 * $\log_2$(1/6)]) **= 1.459148**

In this case, 3 / 6 of the fruits are medium-sized, 2 / 6 are small, 1 / 6 is big.

Calculate the information gained if we select the *shape* variable.

Here, 5 / 6 of the fruits are round and 1 / 6 is thin.

Information gain = - ([5/6 * $\log_2$(5/6)] + [1/6 * $\log_2$(1/6)]) **= 0.650022**

Problem 3

Consider the training examples shown in Table below for a binary classification problem.

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

(a) What is the entropy of this collection of training examples with respect to the positive class?

Answer:

There are four positive examples and five negative examples. Thus, $P(+) = 4/9$ and $P(-) = 5/9$. The entropy of the training examples is $-4/9 \log_2(4/9) - 5/9 \log_2(5/9) = 0.9911$.

(b) What are the information gains of a_1 and a_2 relative to these training examples?

Answer:

For attribute a_1, the corresponding counts and probabilities are:

a_1	+	-
T	3	1
F	1	4

The entropy for a_1 is

$$\frac{4}{9}\Big[-(3/4)\log_2(3/4) - (1/4)\log_2(1/4)\Big] + \frac{5}{9}\Big[-(1/5)\log_2(1/5) - (4/5)\log_2(4/5)\Big] = 0.7616.$$

Therefore, the information gain for a_1 is $0.9911 - 0.7616 = 0.2294$.

For attribute a_2, the corresponding counts and probabilities are:

a_2	+	-
T	2	3
F	2	2

The entropy for a_2 is

$$\frac{5}{9}\Big[-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)\Big] + \frac{4}{9}\Big[-(2/4)\log_2(2/4) - (2/4)\log_2(2/4)\Big] = 0.9839.$$

Therefore, the information gain for a_2 is $0.9911 - 0.9839 = 0.0072$.

(c) For a_3, which is a continuous attribute, compute the information gain for every possible split.

Answer:

a_3	Class label	Split point	Entropy	Info Gain
1.0	+	2.0	0.8484	0.1427
3.0	-	3.5	0.9885	0.0026
4.0	+	4.5	0.9183	0.0728
5.0 5.0	- -	5.5	0.9839	0.0072
6.0	+	6.5	0.9728	0.0183
7.0 7.0	+ -	7.5	0.8889	0.1022

The best split for a_3 occurs at split point equals to 2.

(d) What is the best split (among a_1, a_2, and a_3) according to the information gain?

Answer:

According to information gain, a_1 produces the best split.

(e) What is the best split (between a_1 and a_2) according to the classification error rate?

Answer:

For attribute a_1: error rate $= 2/9$.

For attribute a_2: error rate $= 4/9$.

Therefore, according to error rate, a_1 produces the best split.

(f) What is the best split (between a_1 and a_2) according to the Gini index?

Answer:

For attribute a_1, the gini index is

$$\frac{4}{9}\left[1-(3/4)^2-(1/4)^2\right]+\frac{5}{9}\left[1-(1/5)^2-(4/5)^2\right]=0.3444.$$

For attribute a_2, the gini index is

$$\frac{5}{9}\left[1-(2/5)^2-(3/5)^2\right]+\frac{4}{9}\left[1-(2/4)^2-(2/4)^2\right]=0.4889.$$

Since the gini index for a_1 is smaller, it produces the better split.

Problem 4

Consider the following data set for a binary class problem.

A	B	Class Label
T	F	+
T	T	+
T	T	+
T	F	−
T	T	+
F	F	−
F	F	−
F	F	−
T	T	−
T	F	−

(a) Calculate the information gain when splitting on A and B. Which attribute would the decision tree induction algorithm choose?

Answer:

The contingency tables after splitting on attributes A and B are:

	$A=T$	$A=F$
+	4	0
−	3	3

	$B=T$	$B=F$
+	3	1
−	1	5

The overall entropy before splitting is:

$$E_{orig} = -0.4 \log 0.4 - 0.6 \log 0.6 = 0.9710$$

The information gain after splitting on A is:

$$\begin{aligned} E_{A=T} &= -\frac{4}{7}\log\frac{4}{7} - \frac{3}{7}\log\frac{3}{7} = 0.9852 \\ E_{A=F} &= -\frac{3}{3}\log\frac{3}{3} - \frac{0}{3}\log\frac{0}{3} = 0 \\ \Delta &= E_{orig} - 7/10 E_{A=T} - 3/10 E_{A=F} = 0.2813 \end{aligned}$$

The information gain after splitting on B is:

$$\begin{aligned} E_{B=T} &= -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} = 0.8113 \\ E_{B=F} &= -\frac{1}{6}\log\frac{1}{6} - \frac{5}{6}\log\frac{5}{6} = 0.6500 \\ \Delta &= E_{orig} - 4/10 E_{B=T} - 6/10 E_{B=F} = 0.2565 \end{aligned}$$

Therefore, attribute A will be chosen to split the node.

Feature classification

The goal of feature classification is to determine the class of each feature (e.g. building). For instance, it could be used to determine when a building is damaged or not after a natural disaster. Feature classification requires two input data:

- A input raster that contains the spectral bands,
- A feature class that defines the location (e.g. outline or bounding box) of each feature.

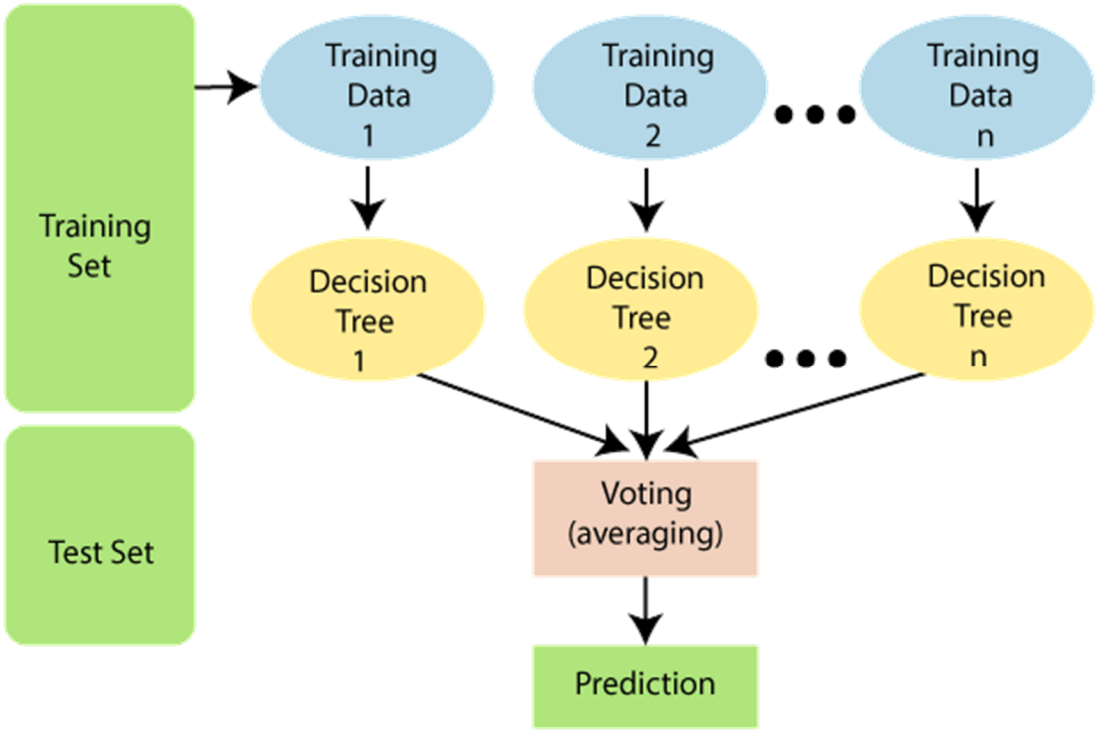
Various classifiers are explained in this section.

Random forest trees

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. **The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

Assumptions for Random Forest



Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

The below diagram explains the working of the Random Forest algorithm:

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.
- It can be used for both classifications as well as regression tasks.
- Overfitting problem that is censorious and can make results poor but in case of the random forest the classifier will not overfit if there are enough trees.
- It can be used for categorical values as well.

Random forest tree for classification

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase. The Working process can be explained in the below steps and diagram:

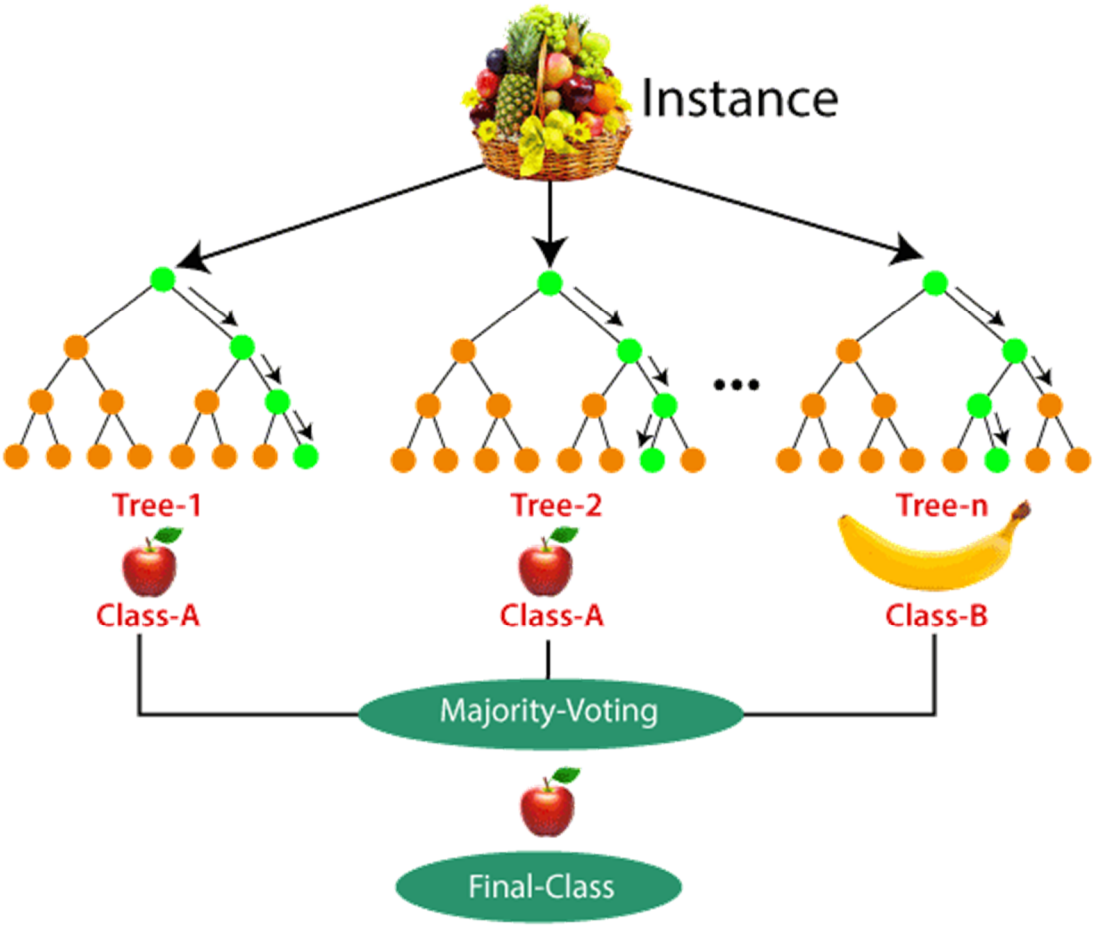
Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.



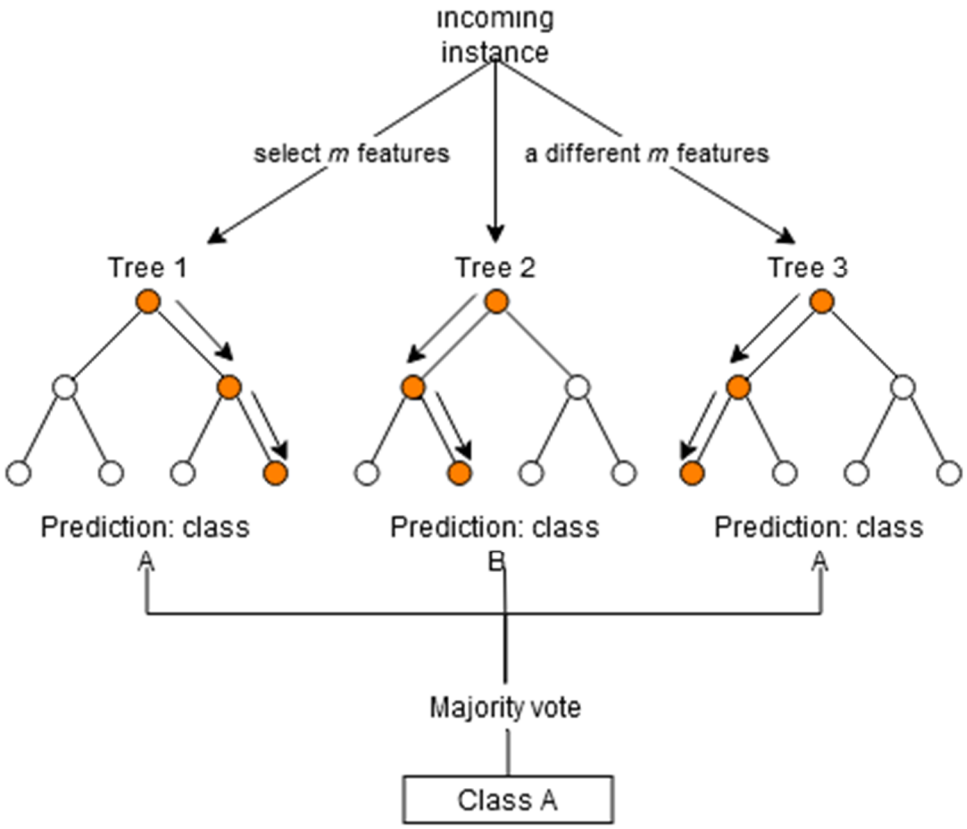
The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision.

Random forest tree terminology

- **Bagging:** Given the training set of *N* examples, we repeatedly sample subsets of the training data of size *n* where *n* is less than *N*. Sampling is done at random but with replacement. This subsampling of a training set is called *bootstrap aggregating*, or *bagging*, for short.
- **Random subspace method:** If each training example has *M* features, we take a subset of them of size $m < M$ to train each estimator. So no estimator sees the full training set, each estimator sees only *m* features of *n* training examples.

- **Training estimators:** We create N_{tree} decision trees, or estimators, and train each one on a different set of m features and n training examples. The trees are not pruned, as they would be in the case of training a simple decision tree classifier.
- **Perform inference by aggregating predictions of estimators:** To make a prediction for a new incoming example, we pass the relevant features of this example to each of the N_{tree} estimators. We will obtain N_{tree} predictions, which we need to combine to produce the overall prediction of the random forest. In the case of classification, we will use majority voting to decide on the predicted class, and in the case of regression, we will take the mean value of the predictions of all the estimators.



Decision tree vs. random forest tree

Random Forest	Decision Tree
While building a random forest the number of rows is selected randomly.	Whereas, it built several decision trees and find out the output.
It combines two or more decision trees together.	Whereas the decision is a collection of variables or data set or attributes.
It gives accurate results.	Whereas it gives less accurate results.
By using multiple trees it reduces the chances of overfitting.	On the other hand, decision trees, it has the possibility of overfitting, which is an error that occurs due to variance or due to bias.

Random Forest	Decision Tree
Random forest is more complicated to interpret.	Whereas, the decision tree is simple so it is easy to read and understand.
In a random forest, we need to generate, process, and analyze trees so that this process is slow, it may take one hour or even days.	The decision tree is not accurate but it processes fast which means it is fast to implement.
It has more computation because it has n number of decision trees, so more decision trees more computation.	Whereas it has less computation.
It has complex visualization, but it plays an important role to show hidden patterns behind the data.	On the other hand, it is simple to visualize because we just need to fit the decision tree model.
The classification and regression problems can be solved by using random forest.	Whereas a decision tree is used to solve the classification and regression problems.
It uses the random subspace method and bagging during tree construction, which has built-in feature importance.	Whereas a decision is made based on the selected sample's feature, this is usually a feature that is used to make a decision, decision tree learning is a process to find the optimal value for each internal tree node.

Bagging and Boosting

The Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote. **Bagging** and **Boosting** are two types of **Ensemble Learning**. These two decrease the variance of a single estimate as they combine several estimates from different models. So the result may be a model with higher stability. Let's understand these two terms in a glimpse.

1. **Bagging**: It is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average.
2. **Boosting**: It is also a homogeneous weak learners' model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.

Bagging: Bootstrap **Agg**regating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It decreases the variance and

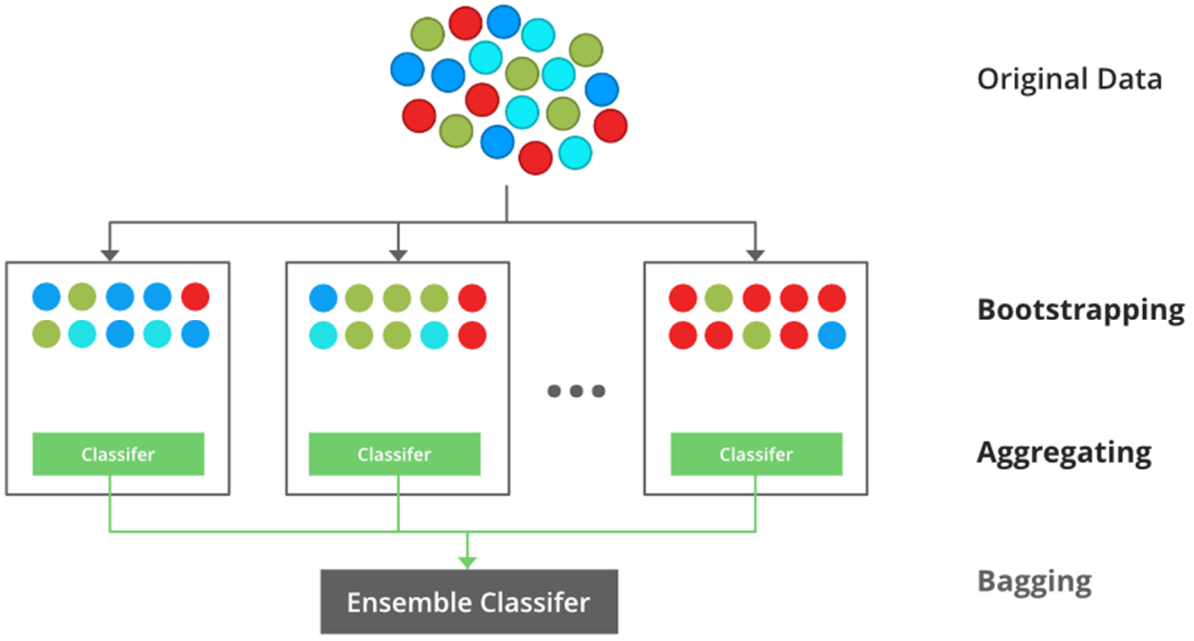
helps to avoid overfitting. It is usually applied to decision tree methods. Bagging is a special case of the model averaging approach.

Description of the Technique

Suppose a set D of d tuples, at each iteration i, a training set D_i of d tuples is sampled with replacement from D (i.e., bootstrap). Then a classifier model Mi is learned for each training set D < i. Each classifier M_i returns its class prediction. The bagged classifier M* counts the votes and assigns the class with the most votes to X (unknown sample).

Implementation Steps of Bagging

- **Step 1:** Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.
- **Step 2:** A base model is created on each of these subsets.
- **Step 3:** Each model is learned in parallel from each training set and independent of each other.
- **Step 4:** The final predictions are determined by combining the predictions from all the models.



An illustration for the concept of bootstrap aggregating (Bagging)

Example of Bagging

The Random Forest model uses Bagging, where decision tree models with higher variance are present. It makes random feature selection to grow trees. Several random trees make a Random Forest.

Boosting

Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models

are added until either the complete training data set is predicted correctly or the maximum number of models is added.

Boosting Algorithms

There are several boosting algorithms. The original ones, proposed by **Robert Schapire** and **Yoav Freund** were not adaptive and could not take full advantage of the weak learners. Schapire and Freund then developed AdaBoost, an adaptive boosting algorithm that won the prestigious Gödel Prize. AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. AdaBoost is short for Adaptive Boosting combines multiple "weak classifiers" into a single "strong classifier".

Algorithm:

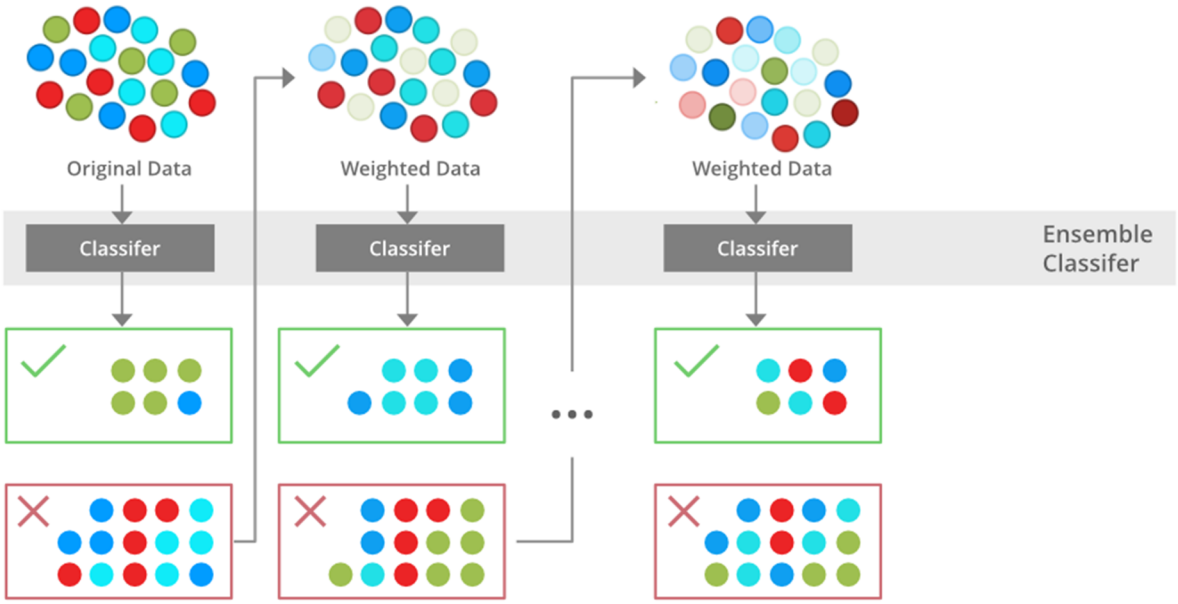
1. *Initialise the dataset and assign equal weight to each of the data point.*
2. *Provide this as input to the model and identify the wrongly classified data points.*
3. *Increase the weight of the wrongly classified data points.*
4. *if (got required results)*

Goto step 5

Else

Goto step 2

5. *End*



Similarities between Bagging and Boosting

Bagging and Boosting, both being the commonly used methods, have a universal similarity of being classified as ensemble methods. Here we will explain the similarities between them.

- Both are ensemble methods to get N learners from 1 learner.
- Both generate several training data sets by random sampling.
- Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
- Both are good at reducing variance and provide higher stability.

Differences between Bagging and Boosting

- Bagging and boosting ensemble techniques deduce N learners from solitary learner.
- Bagging is a parallel ensemble learning method, whereas Boosting is a sequential ensemble learning method.
- Both techniques use random sampling to generate multiple training datasets.
- Both the techniques rely on averaging the N learner's results or Majority voting to make the final prediction.
- Bagging and Boosting help in lowering the model variance and generating stable models.
- While Bagging and boosting make seem similar due to the use of N learners in both techniques, They are inherently quite different. While the Bagging technique is a simple way of combining predictions of the same kind, boosting combines predictions that belong to different types.
- In Bagging, each model is created independent of the other, But in boosting new models, the results of the previously built models are affected.
- Bagging gives equal weight to each model, whereas in Boosting technique, the new models are weighted based on their results.
- In boosting, new subsets of data used for training contain observations that the previous model misclassified. Bagging uses randomly generated training data subsets.
- Bagging tends to decrease variance, not bias. In contrast, Boosting reduces bias, not variance.
- The bagging technique tries to resolve the issue of overfitting training data, whereas Boosting tries to reduce the problem of Bias.

Which is the best, Bagging or Boosting?

- There's not an outright winner; it depends on the data, the simulation and the circumstances.
- Bagging and Boosting decrease the variance of your single estimate as they combine several estimates from different models.
- So the result may be a model with **higher stability**.
- If the problem is that the single model gets a very low performance, Bagging will rarely get a **better bias**.
- However, Boosting could generate a combined model with lower errors as it optimises the advantages and reduces pitfalls of the single model.
- By contrast, if the difficulty of the single model is **over-fitting**, then Bagging is the best option.
- Boosting for its part doesn't help to avoid over-fitting; in fact, this technique is faced with this problem itself. Thus, Bagging is effective more often than Boosting.

Naive Bayes

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem

Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

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

where:

$P(A) =$ The probability of A occurring

$P(B) =$ The probability of B occurring

$P(A|B) =$ The probability of A given B

$P(B|A) =$ The probability of B given A

Problems on application of Bayes theorem for classification

Problem 1

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

Let us test it on a new set of features (let us call it today):

```
today = (Sunny, Hot, Normal, False)
```

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

$$P(Yes|today) \propto \frac{2}{9}.\frac{2}{9}.\frac{6}{9}.\frac{6}{9}.\frac{9}{14} \approx 0.0141$$

and

$$P(No|today) \propto \frac{3}{5}.\frac{2}{5}.\frac{1}{5}.\frac{2}{5}.\frac{5}{14} \approx 0.0068$$

Now, since

$$P(Yes|today) + P(No|today) = 1$$

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

and

$$P(No|today) = \frac{0.0068}{0.0141+0.0068} = 0.33$$

Since

$$P(Yes|today) > P(No|today)$$

So, prediction that golf would be played is 'Yes'.

Problem 2

In this example we have 4 inputs (predictors). The final posterior probabilities can be standardized between 0 and 1.

Outlook	Temp	Humidity	Windy	Play
Rainy	Cool	High	True	?

$$P(Yes \mid X) = P(Rainy \mid Yes) \times P(Cool \mid Yes) \times P(High \mid Yes) \times P(True \mid Yes) \times P(Yes)$$

$$P(Yes \mid X) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.00529 \rightarrow 0.2 = \frac{0.00529}{0.02057 + 0.00529}$$

$$P(No \mid X) = P(Rainy \mid No) \times P(Cool \mid No) \times P(High \mid No) \times P(True \mid No) \times P(No)$$

$$P(No \mid X) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.02057 \rightarrow 0.8 = \frac{0.02057}{0.02057 + 0.00529}$$

Problem 3

- Estimate conditional probabilities of each attributes {color, legs, height, smelly} for the species classes: {M, H} using the data given in the table.
- Using these probabilities estimate the probability values for the new instance – (Color=Green, legs=2, Height=Tall, and Smelly=No).

No	Color	Legs	Height	Smelly	Species
1	White	3	Short	Yes	M
2	Green	2	Tall	No	M
3	Green	3	Short	Yes	M
4	White	3	Short	Yes	M
5	Green	2	Short	No	H
6	White	2	Tall	No	H
7	White	2	Tall	No	H
8	White	2	Short	Yes	H

$$P(M) = \frac{4}{8} = 0.5 \quad P(H) = \frac{4}{8} = 0.5$$

Color	M	H
White	2/4	3/4
Green	2/4	1/4

Legs	M	H
2	1/4	4/4
3	3/4	0/4

Height	M	H
Tall	3/4	2/4
Short	1/4	2/4

Smelly	M	H
Yes	3/4	1/4
No	1/4	3/4

$$p(M|New\ Instance) = p(M) * p(Color = Green|M) * p(Legs = 2|M) * p(Height = tall|M) * p(Smelly = no\ |M)$$

$$p(M|New\ Instance) = 0.5 * \frac{2}{4} * \frac{1}{4} * \frac{3}{4} * \frac{1}{4} = 0.0117$$

$$p(H|New\ Instance) = p(H) * p(Color = Green|H) * p(Legs = 2|H) * p(Height = tall|H) * p(Smelly = no\ |H)$$

$$p(H|New\ Instance) = 0.5 * \frac{1}{4} * \frac{4}{4} * \frac{2}{4} * \frac{3}{4} = 0.047$$

$p(H|New\ Instance) > p(M|New\ Instance)$

Hence the new instance belongs to Speices H

Problem 4

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

$$New\ Instance = (Red, SUV, Domestic) \rightarrow (Yes\ or\ No)$$

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

NAIVE BAYES CLASSIFIER

EXAMPLE - 3

$$p(Yes) = \frac{5}{10} = 0.5$$

$$p(No) = \frac{5}{10} = 0.5$$

Color	Yes	No
Red	3/5	2/5
Yellow	2/5	3/5

Type	Yes	No
Sports	4/5	2/5
SUV	1/5	3/5

Origin	Yes	No
Domestic	2/5	3/5
Imported	3/5	2/5

$New\ Instance = (Red, SUV, Domestic)$ **No**

$$P(Yes|New\ Instance) = p(Yes) * P(Color = Red|Yes) * P(Type = SUV|Yes) * P(Origin = Domestic|Yes)$$

$$P(Yes|New\ Instance) = \frac{5}{10} * \frac{3}{5} * \frac{1}{5} * \frac{2}{5} = \frac{3}{125} = 0.024$$

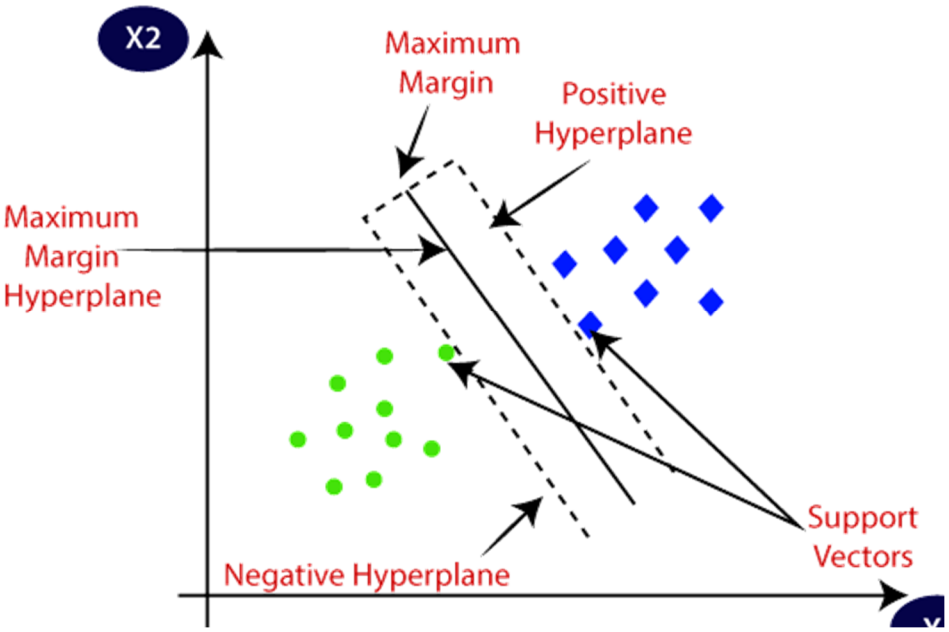
$$P(No|New\ Instance) = p(No) * P(Color = Red|No) * P(Type = SUV|No) * P(Origin = Domestic|No)$$

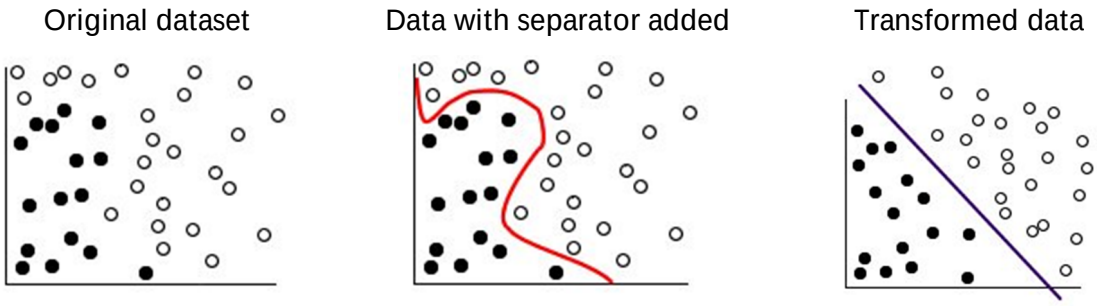
$$P(No|New\ Instance) = \frac{5}{10} * \frac{2}{5} * \frac{3}{5} * \frac{3}{5} = \frac{9}{125} = 0.072$$

$P(No|New\ Instance) > P(Yes|New\ Instance)$

Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane. SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable.





A separator between the categories is found, and then the data are transformed in such a way that the separator could be drawn as a hyperplane. Following this, characteristics of new data can be used to predict the group to which a new record should belong. For example, consider the following figure, in which the data points fall into two different categories. The two categories can be separated with a curve, as shown in the figure. After the transformation, the boundary between the two categories can be defined by a hyperplane, as shown in the following figure.

The mathematical function used for the transformation is known as the kernel function. Following are the popular functions.

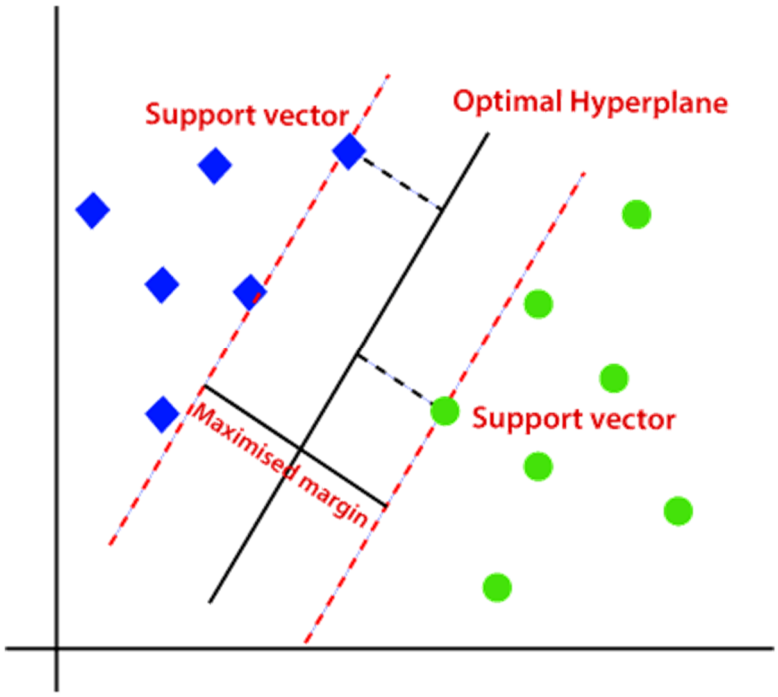
- Linear
- Polynomial
- Radial basis function (RBF)
- Sigmoid

A linear kernel function is recommended when linear separation of the data is straightforward. In other cases, one of the other functions should be used. You will need to experiment with the different functions to obtain the best model in each case, as they each use different algorithms and parameters.

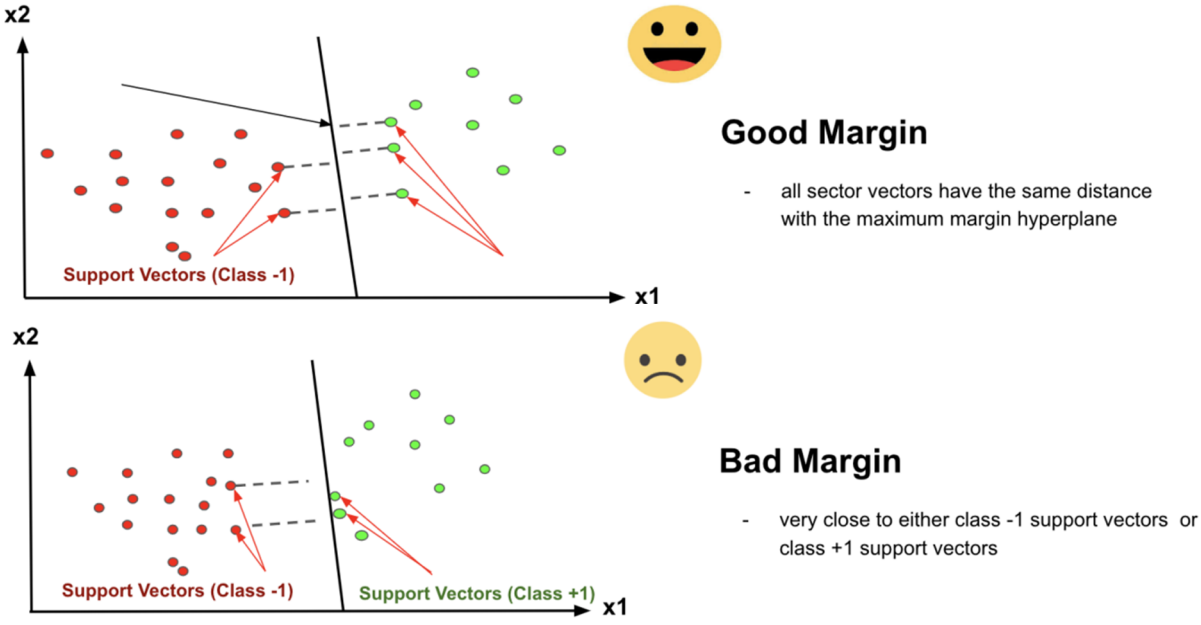
Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify

the data points. This best boundary is known as the hyperplane of SVM. The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane. We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors: The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

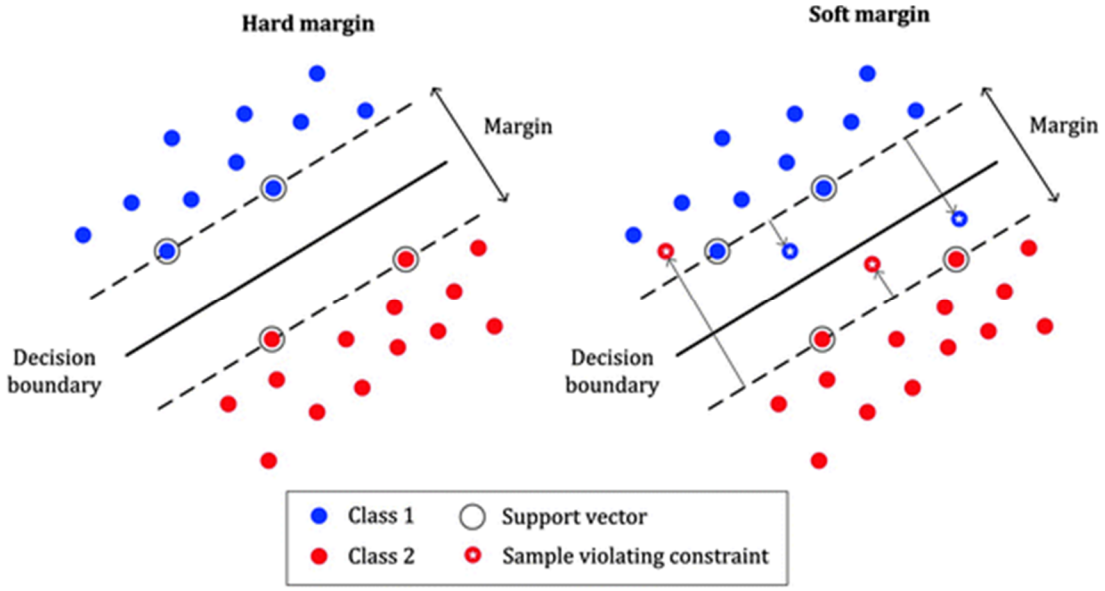


The distance of the vectors from the hyperplane is called the **margin** which is a separation of a line to the closest class points. We would like to choose a hyperplane that maximizes the margin between classes. The graph below shows what good margins and bad margins are.



Again Margin can be sub-divided into,

- **Soft Margin** – As most of the real-world data are not fully linearly separable, we will allow some margin violation to occur which is called soft margin classification. It is better to have a large margin, even though some constraints are violated. Margin violation means choosing a hyperplane, which can allow some data points to stay on either the incorrect side of the hyperplane and between the margin and correct side of the hyperplane.
- **2. Hard Margin** – If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.



Support Vector Machine terminology.

Support Vector Machines are part of the supervised learning model with an associated learning algorithm. It is the most powerful and flexible algorithm used for classification, regression, and detection of outliers. It is used in case of high dimension spaces; where each data item is plotted as a point in n-dimension space such that each feature value corresponds to the value of specific coordinate. The classification is made on the basis of a hyperplane/line as wide as possible, which distinguishes between two categories more clearly. Basically, support vectors are the observational points of each individual, whereas the support vector machine is the boundary that differentiates one class from another class.

Some significant terminology of SVM is given below:

- **Support Vectors:** These are the data point or the feature vectors lying nearby to the hyperplane. These help in defining the separating line.
- **Hyperplane:** It is a subspace whose dimension is one less than that of a decision plane. It is used to separate different objects into their distinct categories. The best hyperplane is the one with the maximum separation distance between the two classes.

- **Margins:** It is defined as the distance (perpendicular) from the data point to the decision boundary. There are two types of margins: good margins and margins. **Good margins** are the one with huge margins and the **bad margins** in which the margin is minor.

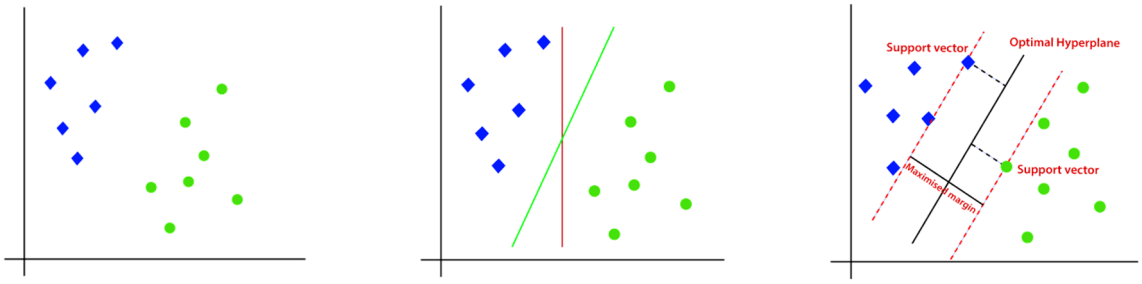
The main goal of SVM is to find the maximum marginal hyperplane, so as to segregate the dataset into distinct classes. It undergoes the following steps:

- Firstly the SVM will produce the hyperplanes repeatedly, which will separate out the class in the best suitable way.
- Then we will look for the best option that will help in correct segregation.

Explain Linear SVM, non-linear SVM.

Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image: So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image. Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

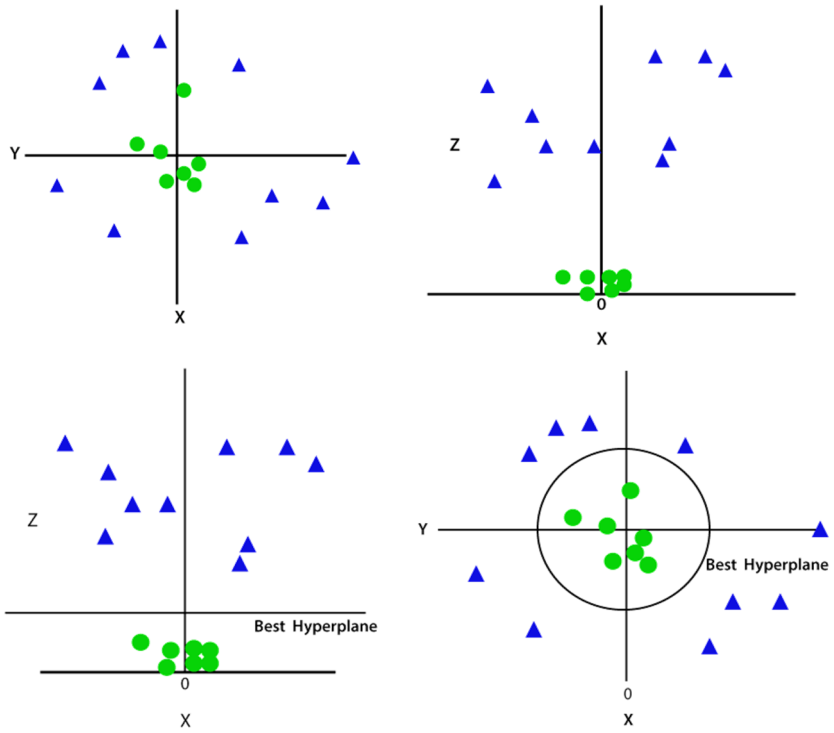
$z=x^2+y^2$

By adding the third dimension, the sample space will become as below image:

So now, SVM will divide the datasets into classes in the following way. Consider the below image:

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

Hence we get a circumference of radius 1 in case of non-linear data.



Hyper parameters of SVM

Hyper parameters of SVM are considered as Kernel, Regularization, Gamma and Margin.

Kernel: The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

For **linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

f(x) = B(0) + sum(ai * (x,xi))

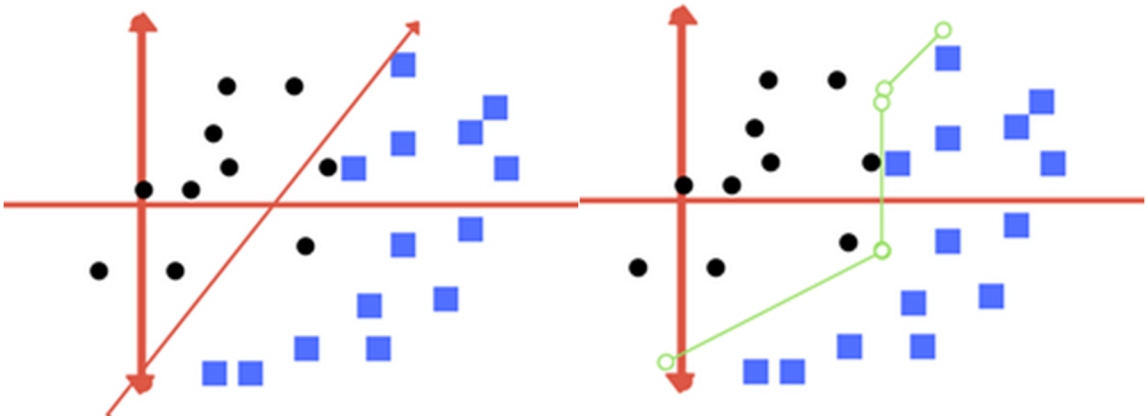
This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

The **polynomial kernel** can be written as *K(x,xi) = 1 + sum(x * xi)^d* and **exponential** as *K(x,xi) = exp(-gamma * sum((x — xi²))*.

Polynomial and exponential kernels calculates separation line in higher dimension. This is called **kernel trick.**

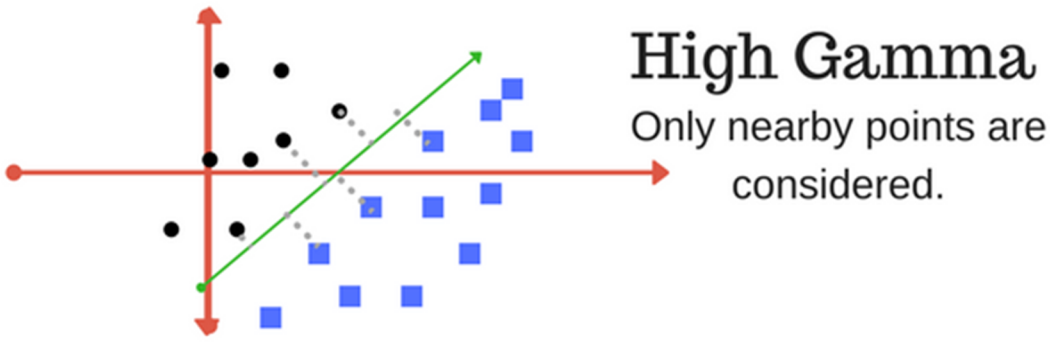
Regularization: The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin

hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. The images below are example of two different regularization parameters. Left one has some misclassification due to lower regularization value. Higher value leads to results like right one.

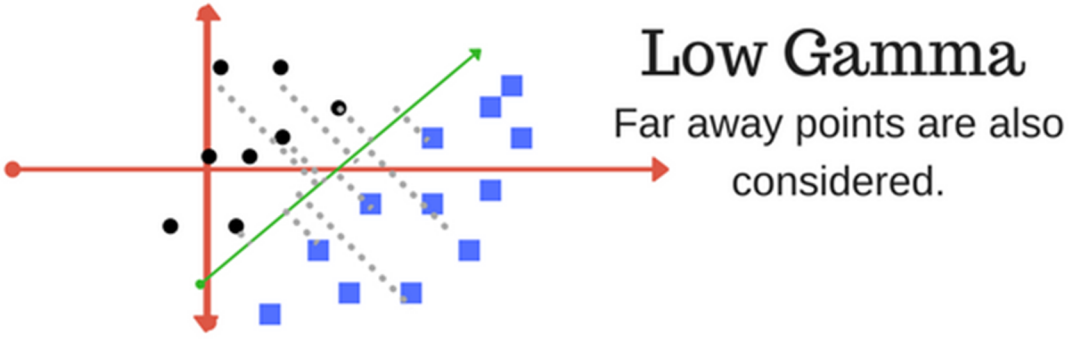


Left: low regularization value, right: high regularization value

Gamma: The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.



High Gamma

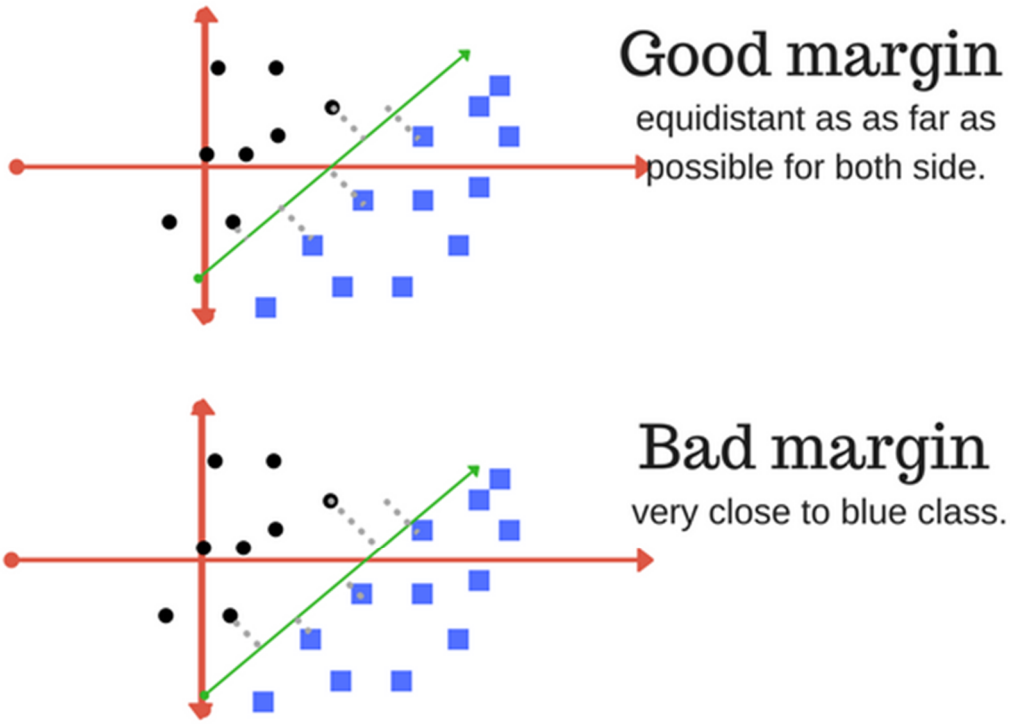


Low Gamma

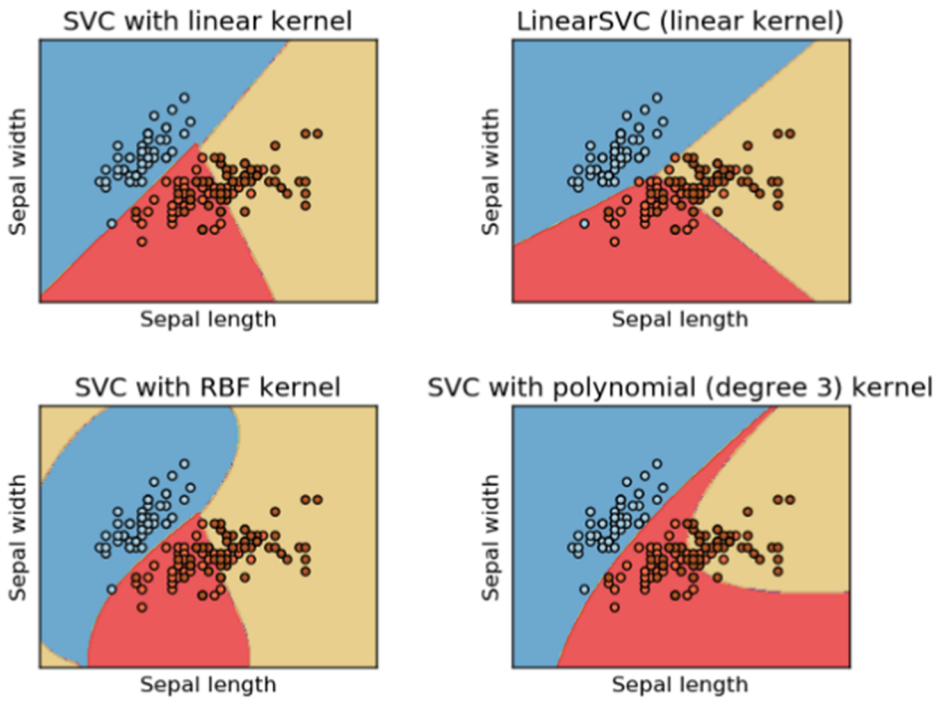
Margin: And finally last but very important characteristic of SVM classifier. SVM to core tries to achieve a good margin.

A margin is a separation of line to the closest class points.

A ***good margin*** is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.



Kernel functions: linear, polynomial, rbf, sigmoid



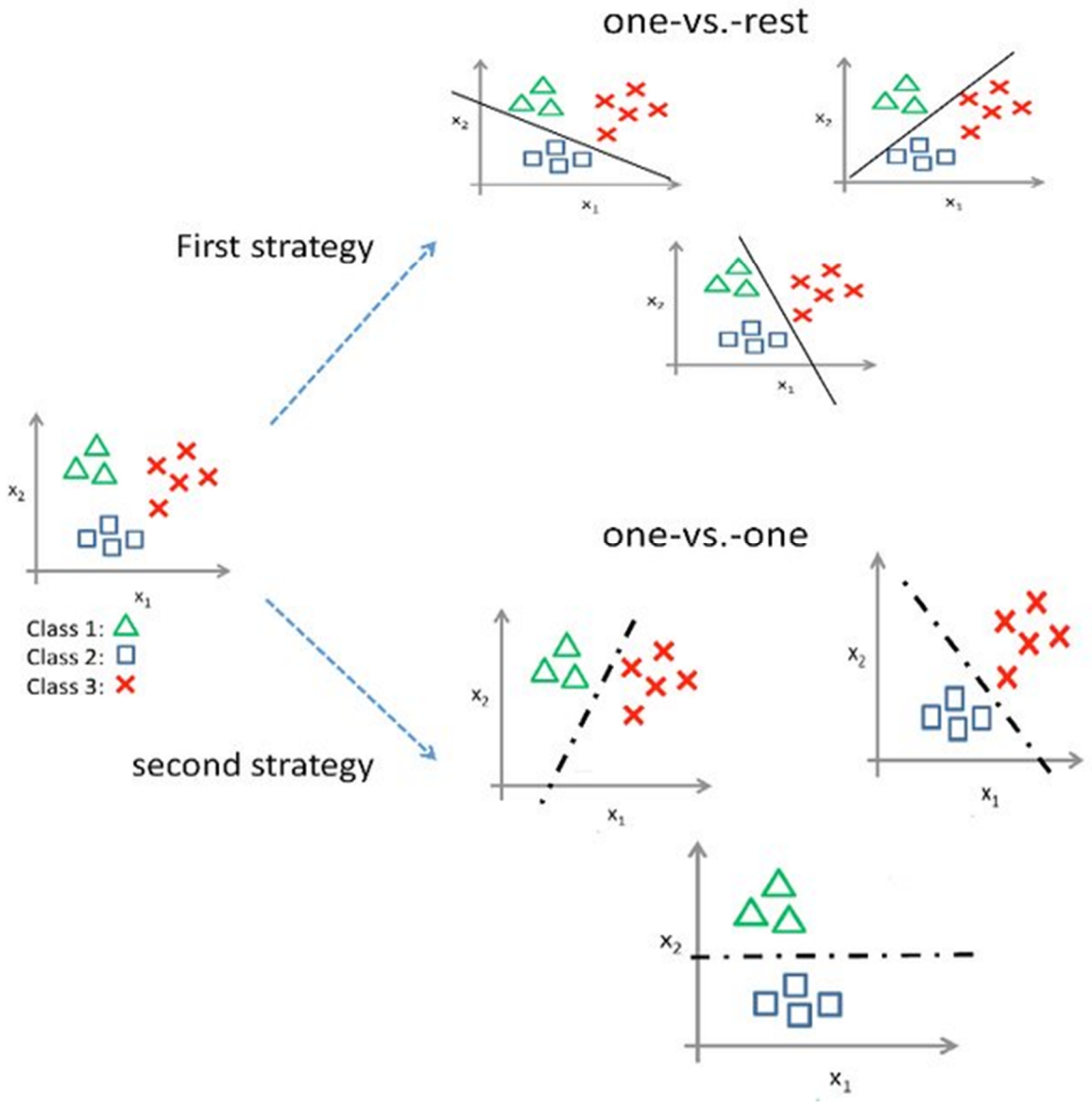
Kernel Function	Formula	Optimization Parameter
Linear	$K(x_n, x_i) = (x_n, x_i)$	C and γ
RBF	$K(x_n, x_i) = \exp\left(-\gamma \|x_n - x_i\|^2 + C\right)$	C and γ
Sigmoid	$K(x_n, x_i) = \tanh(\gamma(x_n, x_i) + r)$	C, γ, and r
Polynomial	$K(x_n, x_i) = (\gamma(x_n, x_i) + r)^d$	C, γ, r, and d

Explanation, C: cost; γ: gamma; r: coefficient; d: degree.

Multi class classification methods

Two different examples of this approach are the One-vs-Rest and One-vs-One strategies.

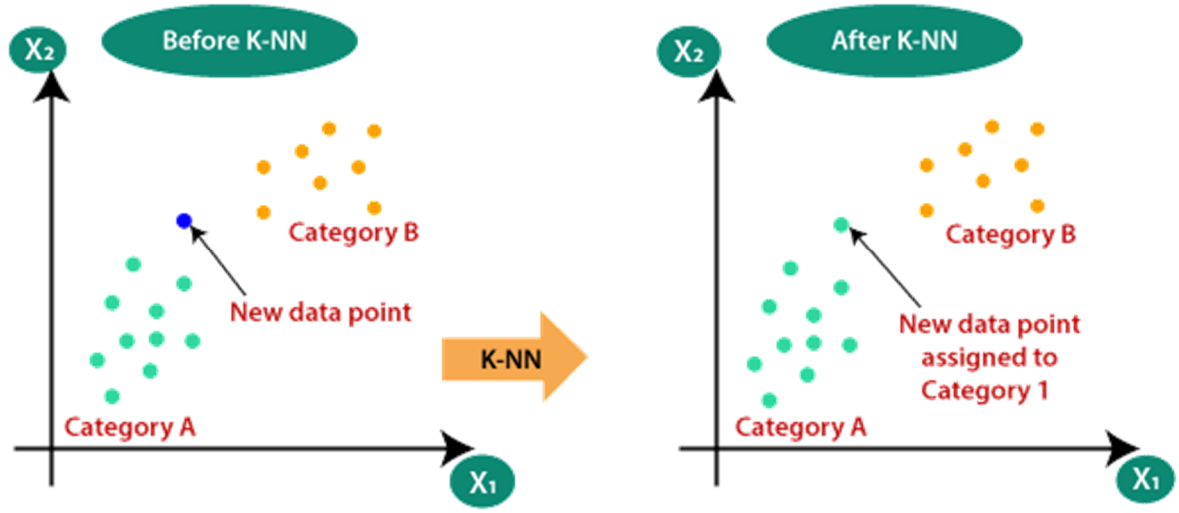
- Binary classification models like logistic regression and SVM do not support multi-class classification natively and require meta-strategies.
- The One-vs-Rest strategy splits a multi-class classification into one binary classification problem per class.
- The One-vs-One strategy splits a multi-class classification into one binary classification problem per each pair of classes.



K Nearest Neighbor algorithm

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

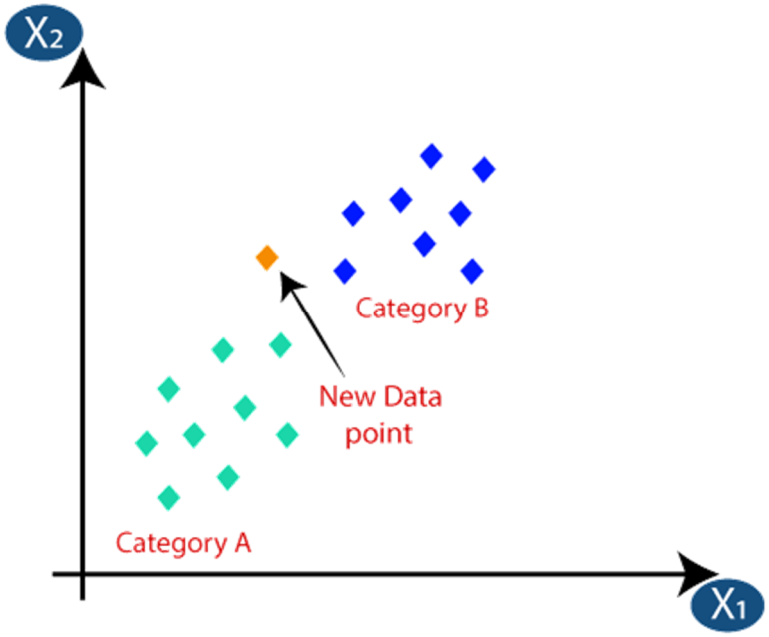
Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



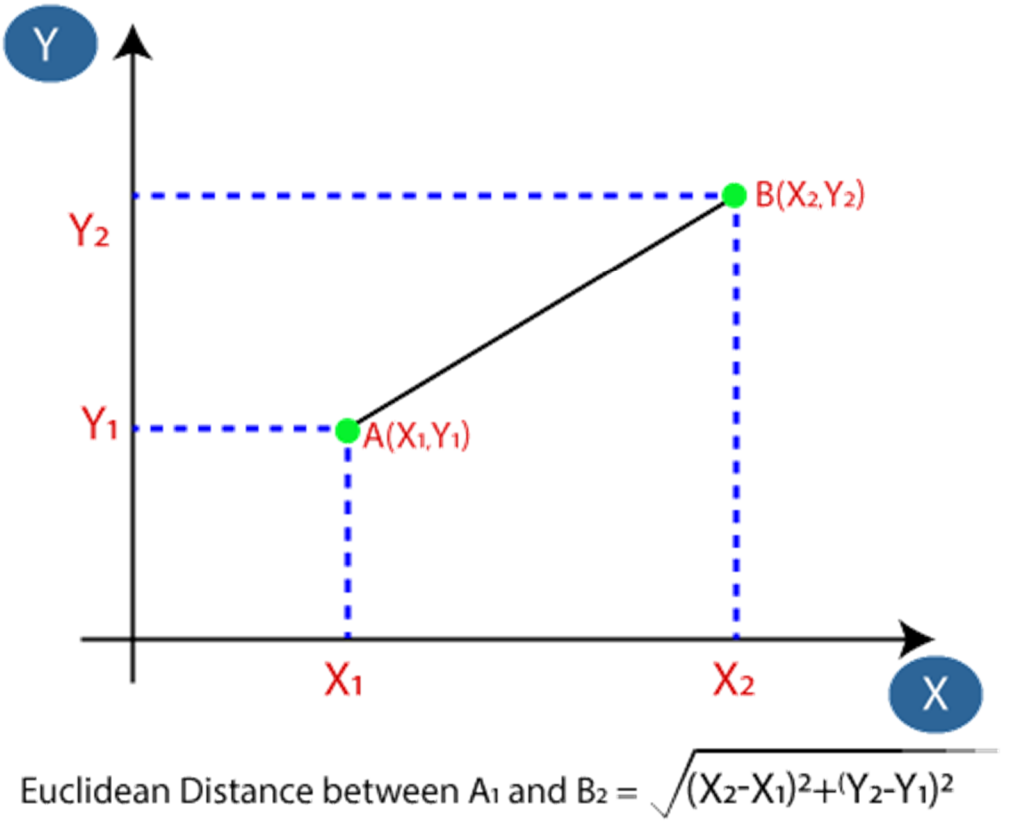
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbour is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

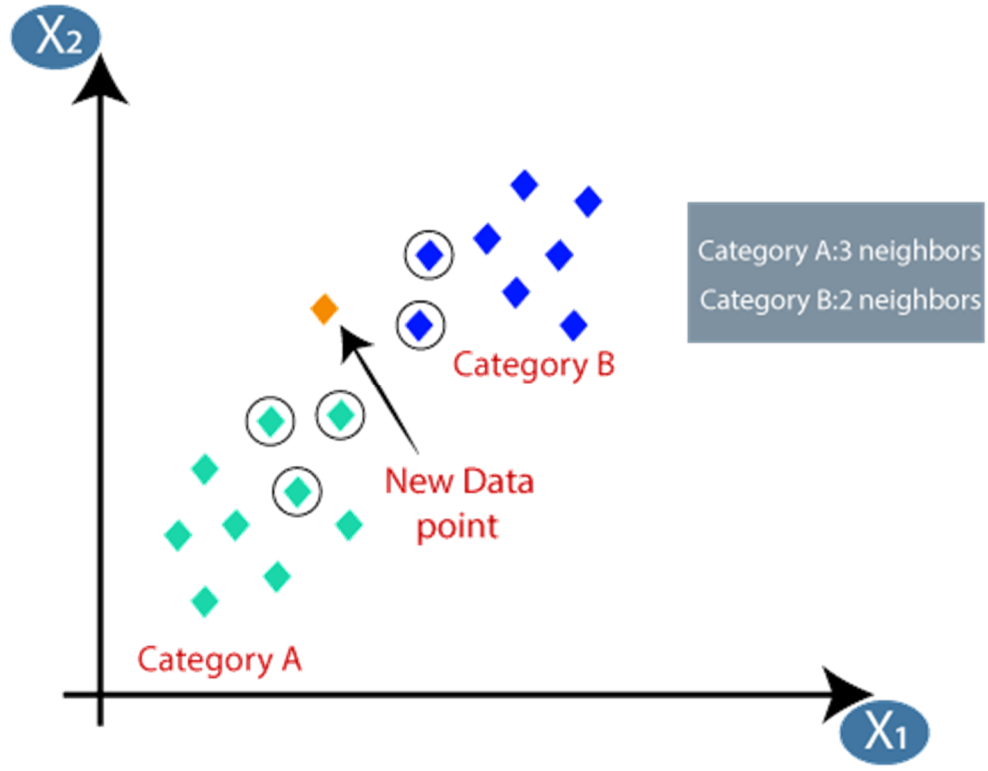


- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

Difference between KNN and K means?

- K-NN is a **Supervised** while K-means is an **unsupervised** Learning.
- K-NN is a **classification** or **regression** machine learning algorithm while K-means is a **clustering** machine learning algorithm.
- K-NN is a **lazy learner** while K-Means is an **eager learner**. An eager learner has a model fitting that means a training step but a lazy learner does not have a training phase.
- K-NN performs much better if all of the data have the same scale but this is not true for K-means.
- K-means is a clustering algorithm that tries to partition a set of points into K sets (clusters) such that the points in each cluster tend to be near each other. It is unsupervised because the points have no external classification.
- K-nearest neighbors is a classification (or regression) algorithm that in order to determine the classification of a point, combines the classification of the K nearest points. It is supervised because you are trying to classify a point based on the known classification of other points.

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

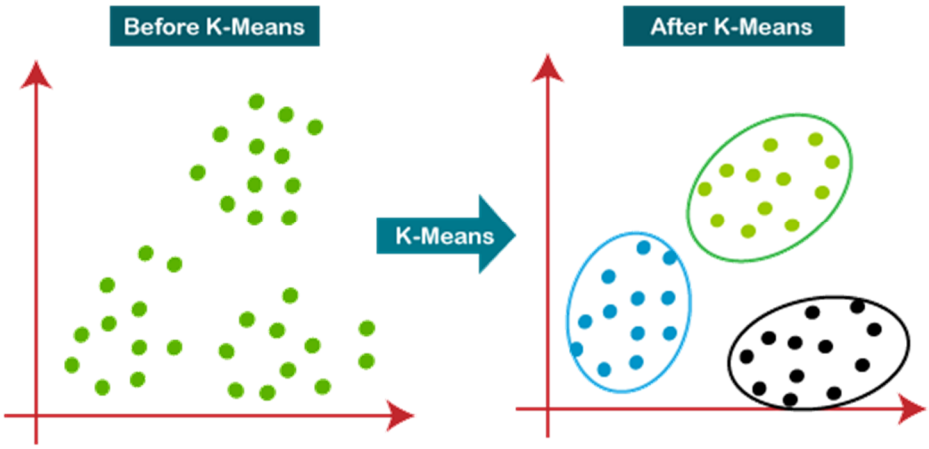
It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters. The below diagram explains the working of the K-means Clustering Algorithm:



The working of the K-Means algorithm is explained in the below steps:

- **Step-1:** Select the number K to decide the number of clusters.
- **Step-2:** Select random K points or centroids. (It can be other from the input dataset).
- **Step-3:** Assign each data point to their closest centroid, which will form the predefined K clusters.
- **Step-4:** Calculate the variance and place a new centroid of each cluster.
- **Step-5:** Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
- **Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.
- **Step-7**: The model is ready.

Is K nearest neighbor supervised or unsupervised?

KNN is a simple supervised learning algorithm.

KNN works on a basic assumption that data points of similar classes are closer to each other.

Now suppose you have a classification problem to identify whether a given data point is of class A or class B and your aim is to classify test datapoints into these classes for which you have a training dataset of alreday classified data points.

KNN assignes 1/k probability to ‘k’ nearest pre classified training data point from our test data point and 0 probability to rest of data points, where k may be any number (but try to put it odd to avoid tie cases). After that we count the number of each classes (i.e A and B) out of those K points and our test data point will be classified as that class whose count is greater.

For example if we are using k=5 then our algorithm looks for 5 nearest point from our test dataset point and count the number of each class out of those 5 point.Suppose our counting results in A=3 and B=2 then KNN will assign class A to that test dataset point.

KNN is categorized under supervised ML techniques. It works assuming that similar classes data points are near one another. Take a case of data points classification, where they fit? Class A or B? For this you have a classified data points training set. KNN assumes a probability of 1/k for 'k' closest data point, then assumes zero probability for other data points.

Now k could be any digit, let's count the classes (A, B) out of the k points, the test data points get classified as the class with higher count. If k = 6, then the algorithm searches for the nearest point in the dataset, and count instances of every class out of the 6 point. Now assume counting turns out to be A = 2, B = 1, then KNN will allot A to the data set point.

In this scenario, Data points classification is done considering their proximity with known Data points classes, thus KNN is a supervised ML algorithm.

Feel free to contact me on +91-8329347107 calling / +91-9922369797 whatsapp, email ID: adp.mech@coep.ac.in and abhipatange93@gmail.com

****

AI & ML for Mechanical Engineers

Course Code: ME (DE) - 22016

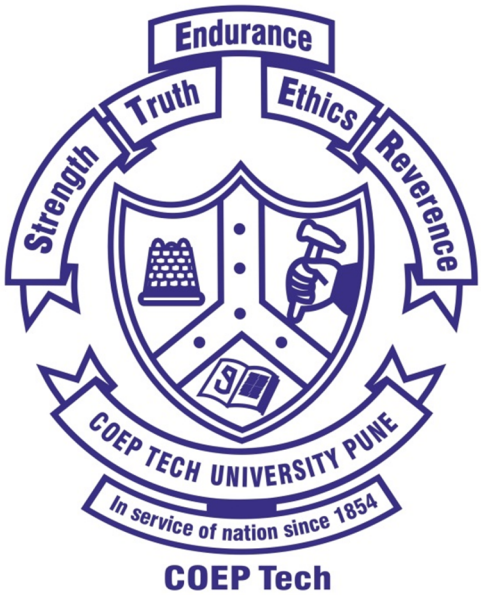
Part 3: Reinforcement & Deep learning

Final Year Bachelor of Technology (Choice Based Credit System)
Mechanical Engineering
(With Effect from Academic Year 2021-22)

Lecture notes

by

Abhishek D. Patange, Ph.D.
Department of Mechanical Engineering



COEP Technological University Pune

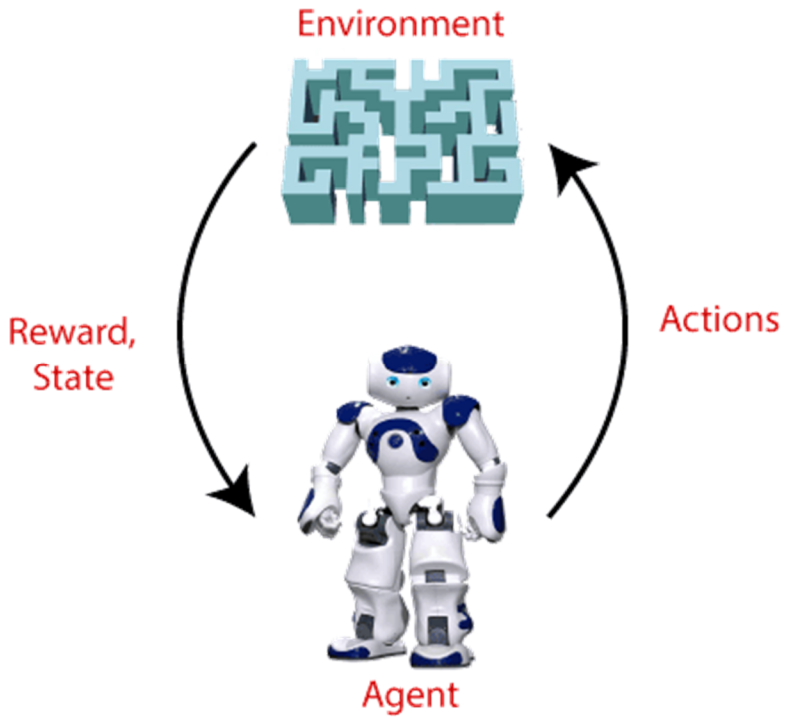
Reinforcement learning

- Reinforcement Learning is a **feedback-based Machine learning** technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the **agent learns automatically using feedbacks** without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to **learn by its experience only**.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that ***"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."*** How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.
- Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback. The agent continues doing these three things (**take action, change state/remain in the same state, and get feedback**), and by doing these actions, he learns and explores the environment. The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.

Key constituents of reinforcement learning

- **Agent():** An entity that can perceive/explore the environment and act upon it.
- **Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- **Action():** Actions are the moves taken by an agent within the environment.
- **State():** State is a situation returned by the environment after each action taken by the agent.
- **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.

- **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value():** It is expected long-term retuned with the discount factor and opposite to the short-term reward.
- **Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action (a).



Key features of reinforcement learning

- In RL, the agent is not instructed about the environment and what actions need to be taken.
- It is based on the hit and trial process.
- The agent takes the next action and changes states according to the feedback of the previous action.
- The agent may get a delayed reward.
- The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

Approaches to implement reinforcement learning

There are mainly three ways to implement reinforcement-learning in ML, which are:

- **Value-based:** The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.

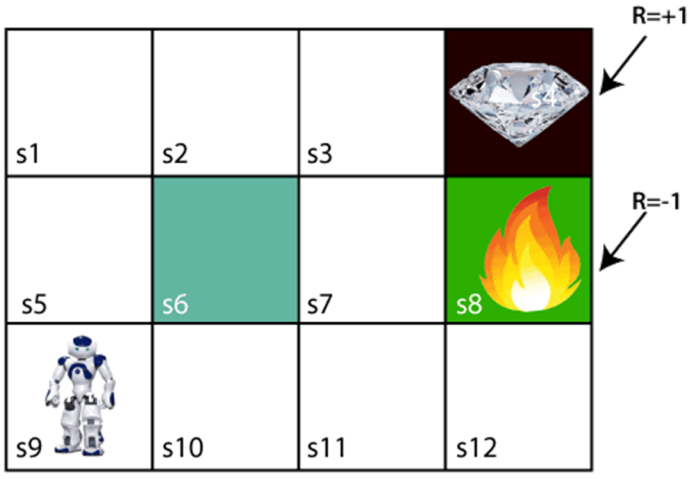
- **Policy-based:** Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward. The policy-based approach has mainly two types of policy:
  - ❖ **Deterministic:** The same action is produced by the policy (π) at any state.
  - ❖ **Stochastic:** In this policy, probability determines the produced action.
- **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

Reinforcement Learning working

To understand the working process of the RL, we need to consider two main things:

- **Environment:** It can be anything such as a room, maze, football ground, etc.
- **Agent:** An intelligent agent such as AI robot.

Let's take an example of a maze environment that the agent needs to explore. Consider the below image:



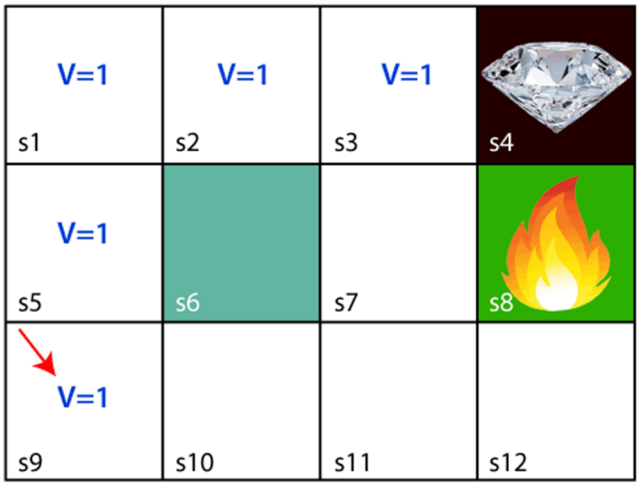
Lorem ipsum

In the above image, the agent is at the very first block of the maze. The maze is consisting of an S_6 block, which is a **wall**, S_8 a **fire pit**, and S_4 a **diamond block.**

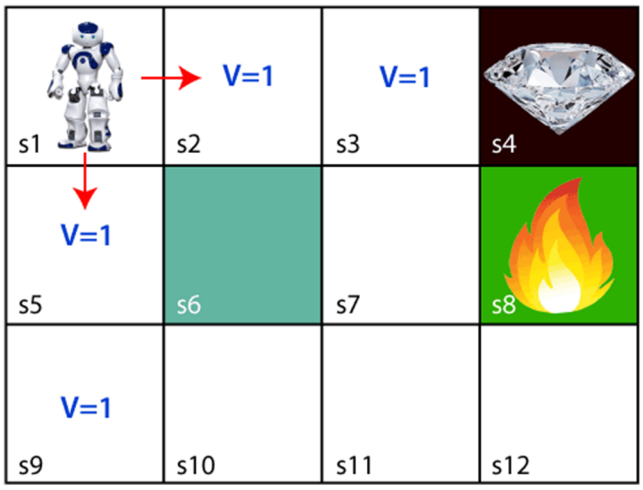
The agent cannot cross the S_6 block, as it is a solid wall. If the agent reaches the S_4 block, then get the **+1 reward;** if it reaches the fire pit, then gets **-1 reward point**. It can take four actions**: move up, move down, move left, and move right.**

The agent can take any path to reach to the final point, but he needs to make it in possible fewer steps. Suppose the agent considers the path **S9-S5-S1-S2-S3**, so he will get the +1-reward point.

The agent will try to remember the preceding steps that it has taken to reach the final step. To memorize the steps, it assigns 1 value to each previous step. Consider the below step:



Now, the agent has successfully stored the previous steps assigning the 1 value to each previous block. But what will the agent do if he starts moving from the block, which has 1 value block on both sides? Consider the below diagram:



It will be a difficult condition for the agent whether he should go up or down as each block has the same value. So, the above approach is not suitable for the agent to reach the destination. Hence to solve the problem, we will use the **Bellman equation**, which is the main concept behind reinforcement learning.

Types of reinforcement learning

Positive Reinforcement: The positive reinforcement learning means adding something to increase the tendency that expected behavior would occur again. It impacts positively on the behavior of the agent and increases the strength of the behavior. This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement may lead to an overload of states that can reduce the consequences.

Advantages are:

- Maximizes Performance
- Sustain Change for a long period of time

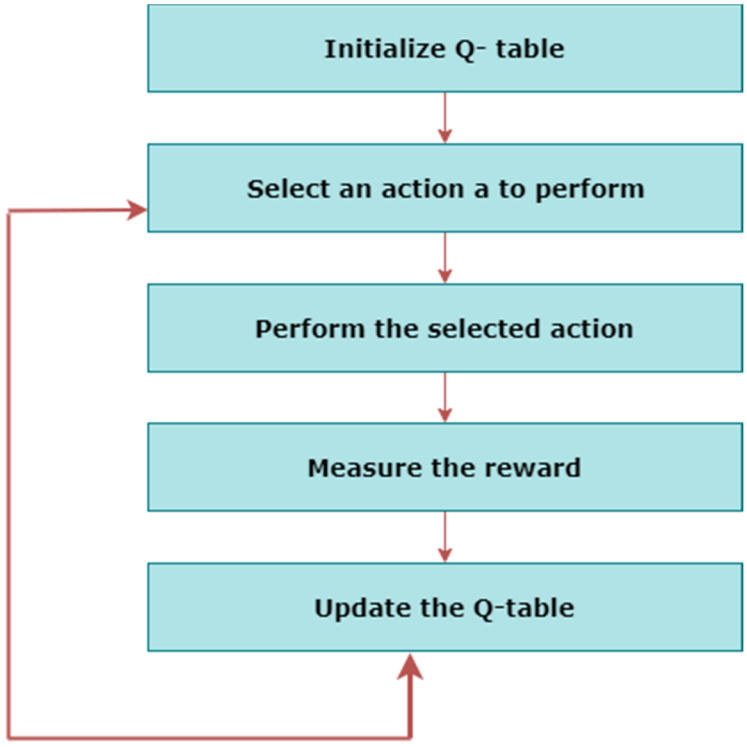
- Too much Reinforcement can lead to an overload of states which can diminish the results

Negative Reinforcement: The negative reinforcement learning is opposite to the positive reinforcement as it increases the tendency that the specific behavior will occur again by avoiding the negative condition. It can be more effective than the positive reinforcement depending on situation and behavior, but it provides reinforcement only to meet minimum behavior. Advantages are:

- Increases Behavior
- Provide defiance to a minimum standard of performance
- It Only provides enough to meet up the minimum behaviour

Q-Learning.

- Q-learning is an **off policy RL algorithm**, which is used for the temporal difference Learning. The temporal difference learning methods are the way of comparing temporally successive predictions.
- It learns the value function Q (S, a), which means how good to take action "**a**" at a particular state "**s**."
- The below flowchart explains the working of Q- learning:



State Action Reward State action (SARSA):

- SARSA stands for **State Action Reward State action**, which is an **on-policy** temporal difference learning method. The on-policy control method selects the action for each state while learning using a specific policy.

- The goal of SARSA is to calculate the **Q π (s, a) for the selected current policy π and all pairs of (s-a).**
- The main difference between Q-learning and SARSA algorithms is that **unlike Q-learning, the maximum reward for the next state is not required for updating the Q-value in the table.**
- In SARSA, new action and reward are selected using the same policy, which has determined the original action.The SARSA is named because it uses the quintuple **Q(s, a, r, s', a').**

  Where,

  s: original state

  a: Original action

  r: reward observed while following the states

  s' and a': New state, action pair

Deep Q Neural Network (DQN):

- As the name suggests, DQN is a **Q-learning using Neural networks**.
- For a big state space environment, it will be a challenging and complex task to define and update a Q-table.
- To solve such an issue, we can use a DQN algorithm. Where, instead of defining a Q-table, neural network approximates the Q-values for each action and state.
- Now, we will expand the Q-learning.

Difference between Reinforcement Learning and Supervised Learning

The Reinforcement Learning and Supervised Learning both are the part of machine learning, but both types of learnings are far opposite to each other. The RL agents interact with the environment, explore it, take action, and get rewarded. Whereas supervised learning algorithms learn from the labeled dataset and, on the basis of the training, predict the output. The difference table between RL and Supervised learning is given below:

Reinforcement Learning	Supervised Learning
RL works by interacting with the environment.	Supervised learning works on the existing dataset.
The RL algorithm works like the human brain works when making some decisions.	Supervised Learning works as when a human learns things in the supervision of a guide.
There is no labeled dataset is present	The labeled dataset is present.
No previous training is provided to the learning agent.	Training is provided to the algorithm so that it can predict the output.
RL helps to take decisions sequentially.	In Supervised learning, decisions are made when input is given.

Introduction to deep learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data.

As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture. A formal definition of deep learning is- neurons. Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbours. The question here is how we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

- DL is a subfield of Machine Learning, inspired by the biological neurons of a brain, and translating that to artificial neural networks with representation learning.
- When the volume of data increases, Machine learning techniques, no matter how optimized, starts to become inefficient in terms of performance and accuracy, whereas Deep learning performs soo much better in such cases.
- Well one cannot quantify a threshold for data to be called big, but intuitively let's say a Million sample might be enough to say "It's Big"(This is where Michael Scott would've uttered his famous words "That's what she said")

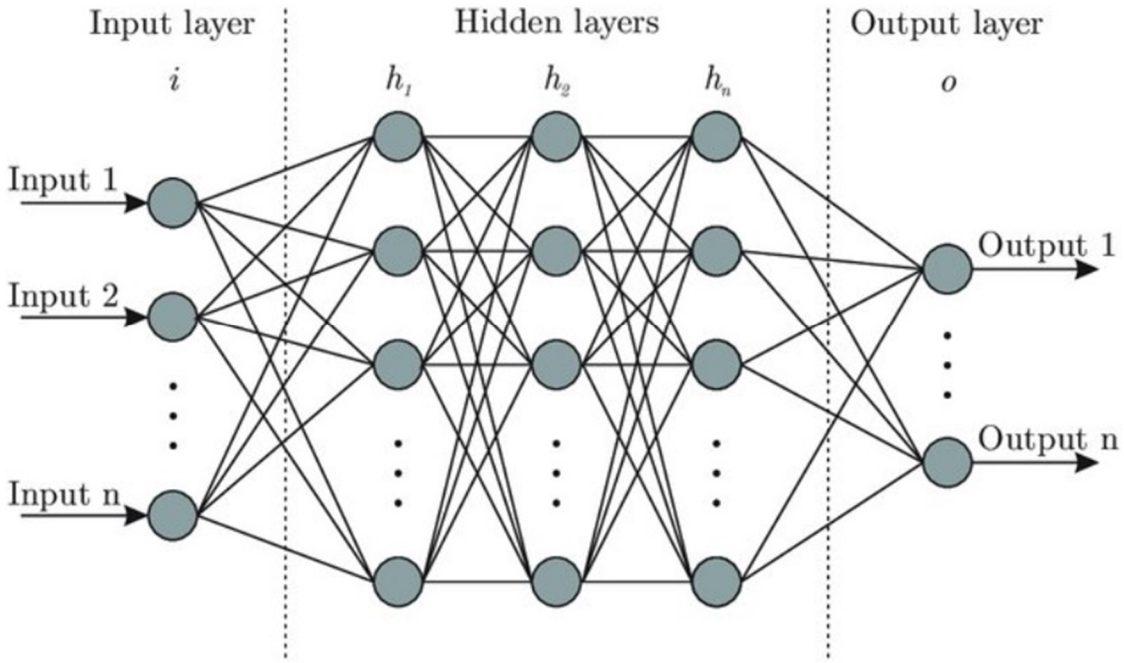
Researchers tried to mimic the working of the human brain and replicated it into the machine making machines capable of thinking and solving complex problems. Deep Learning (DL) is a subset of Machine Learning (ML) that allows us to train a model using a set of inputs and then predict output based.

Like the human brain, the model consists of a set of neurons that can be grouped into 3 layers:

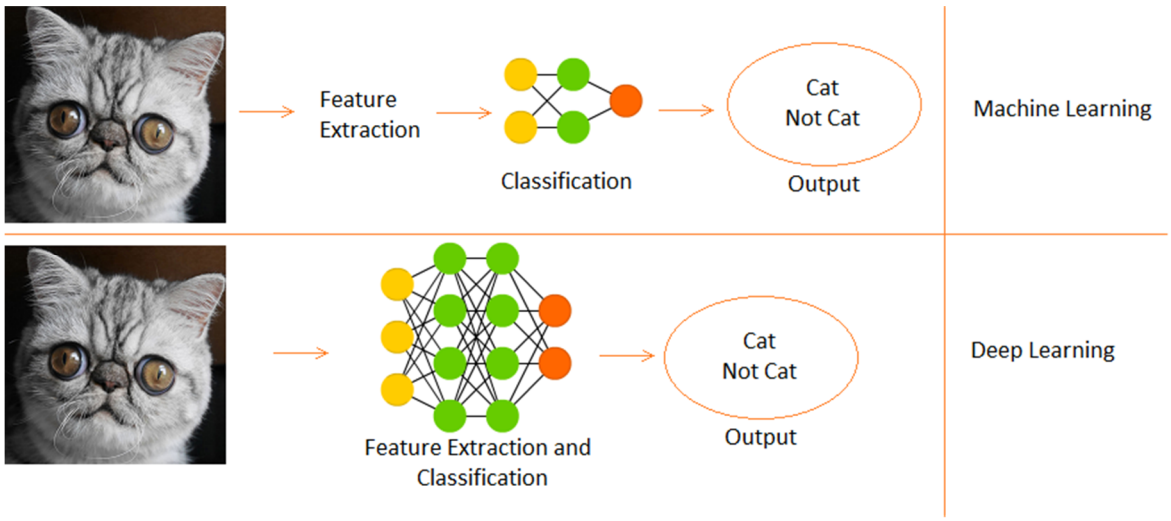
a) Input Layer: It receives input and passes it to hidden layers.

b) Hidden Layers: There can be 1 or more hidden layers in Deep Neural Network (DNN). "Deep" in DL refers to having more than 1 layer. All computations are done by hidden layers.

c) Output Layer: This layer receives input from the last hidden layer and gives the output.



Machine Learning	Deep Learning
Works on small amount of Dataset for accuracy.	Works on Large amount of Dataset.
Dependent on Low-end Machine.	Heavily dependent on High-end Machine.
Divides the tasks into sub-tasks, solves them individually and finally combine the results.	Solves problem end to end.
Takes less time to train.	Takes longer time to train.
Testing time may increase.	Less time to test the data.



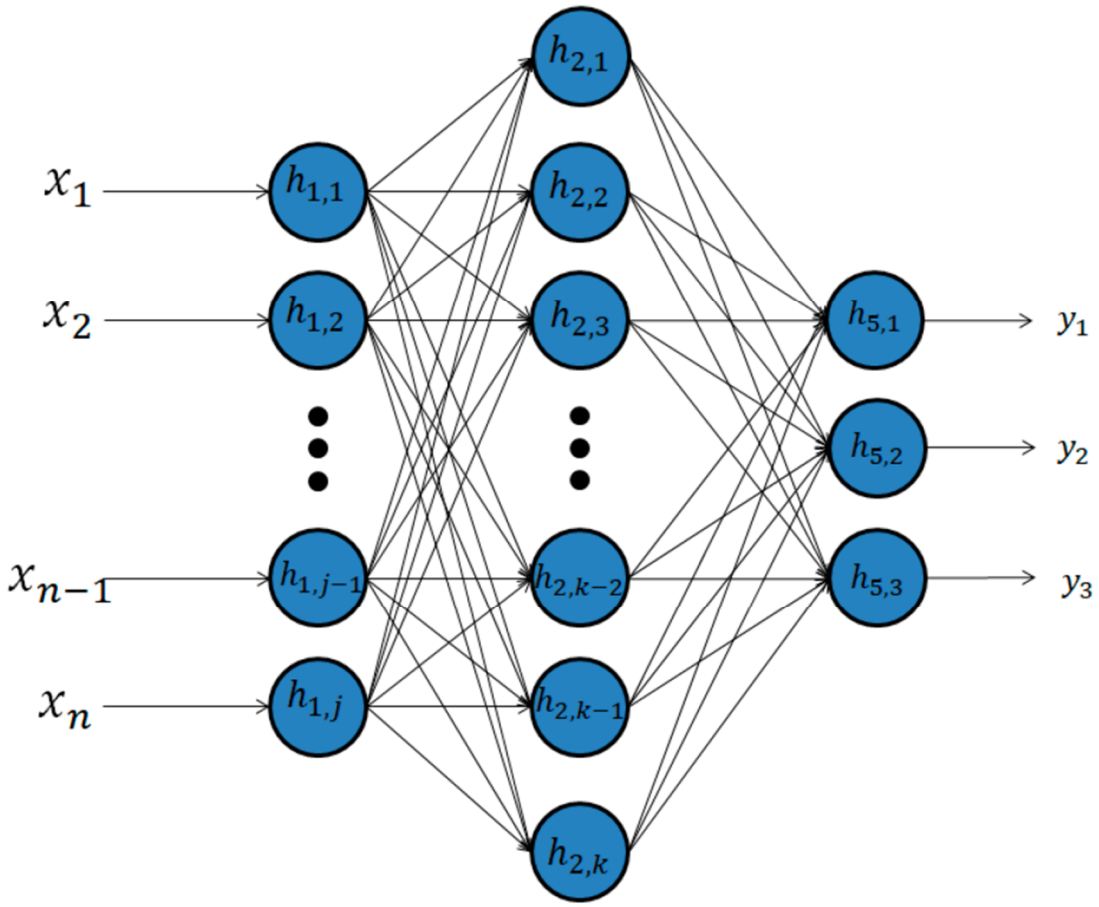
Artificial Neural Network (ANN)

- What neural network essentially means is we take logistic regression and repeat it multiple times.
- In a normal logistic regression, we have an input layer and an output layer.

- But in the case of a Neural Network, there is at least one hidden layer of regression between these input and output layers.

How many layers are needed to call it a "Deep" neural network?

- Well of course there is no specific amount of layers to classify a neural network as deep.
- The term "Deep" is quite frankly relative to every problem.
- The correct question we can ask is "How much deep?".
- For example, the answer to "How deep is your swimming pool?" can be answered in multiple ways.
- It could be 2 meters deep or 10 meters deep, but it has "depth". Same with our neural network, it can have 2 hidden layers or "thousands" hidden layers(yes you heard that correctly).
- So I'd like to just stick with the question of "How much deep?" for the time being.



Why are the hidden layers?

- They are called hidden because they do not see the original inputs(the training set).
- For example, let's say you have a NN with an input layer, one hidden layer, and an output layer.
- When asked how many layers your NN has, your answer should be "It has 2 layers", because while computation the initial, or the input layer, is ignored.

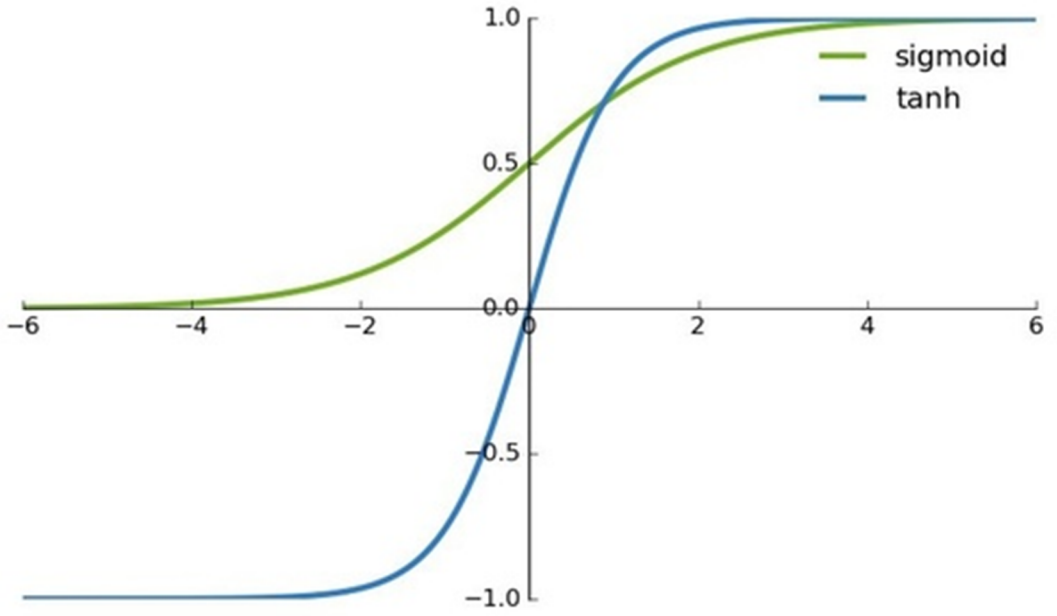
- Let me help visualize how a 2 layer Neural network looks like.

Step by step we shall understand this image.

1) As you can see here we have a 2 Layered Artificial Neural Network. A Neural network was created to mimic the biological neuron of the human brain. In our ANN we have a *"k"* number of nodes. The number of nodes is a hyperparameter, which essentially means that the amount is configured by the practitioner making the model.

2) The inputs and outputs layers do not change. We have "n" input features and 3 possible outcomes.

3) Unlike Logistic regression, neural networks use the *tanh* function as their activation function instead of the *sigmoid* function which you are quite familiar with. The reason is that the mean of its output is closer to 0 which makes the more centered for input to the next layer. tanh function can cause an increase in non-linearity which makes our model learn better.



4) In normal logistic regression: Input => Output.

Whereas in a Neural network: Input => Hidden Layer => Output. The hidden layer can be imagined as the output of part 1 and input of part 2 of our ANN.

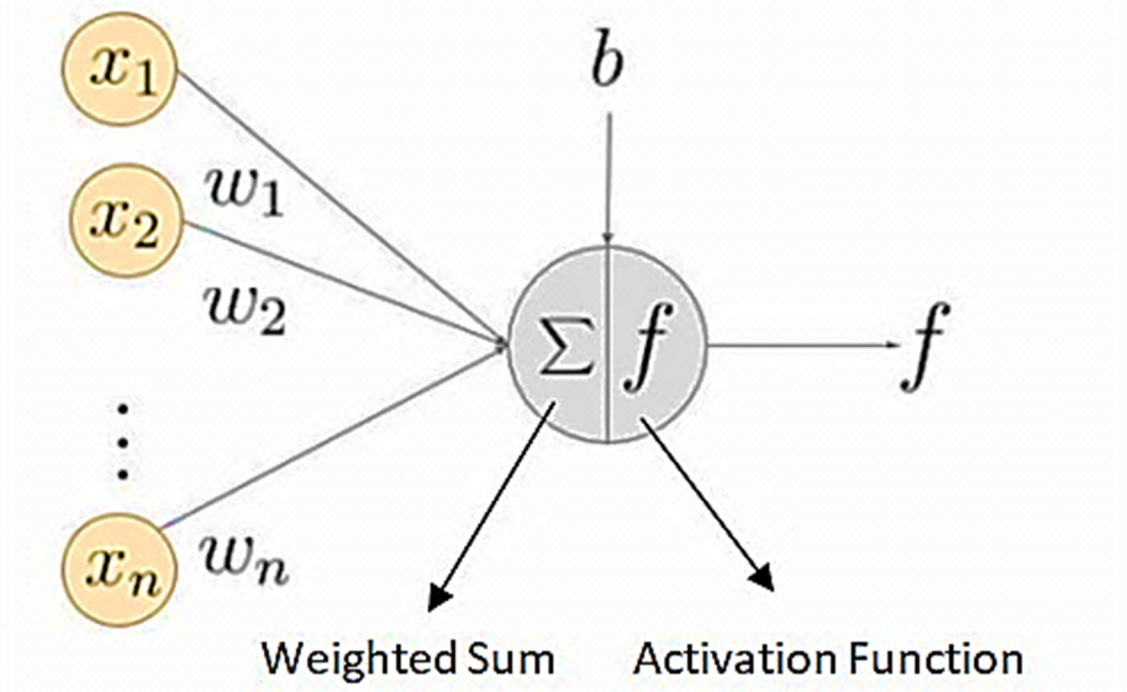
Now let us have a more practical approach to a 2 Layered Neural Network.

(Important Note: We shall continue where we left off in the previous article. I'm not going to waste your time and mine by loading the dataset again and preparing it. The link to the Part 1 of this series is given above.)

Activation Functions

Each neuron has an **activation function** that performs computation. Different layers can have different activation functions but neurons belonging to one layer have the same activation function. In DNN, a weighted sum of input is calculated based on weights and

inputs provided. Then, the activation function comes into the picture that works on weighted sum and converts it into output.



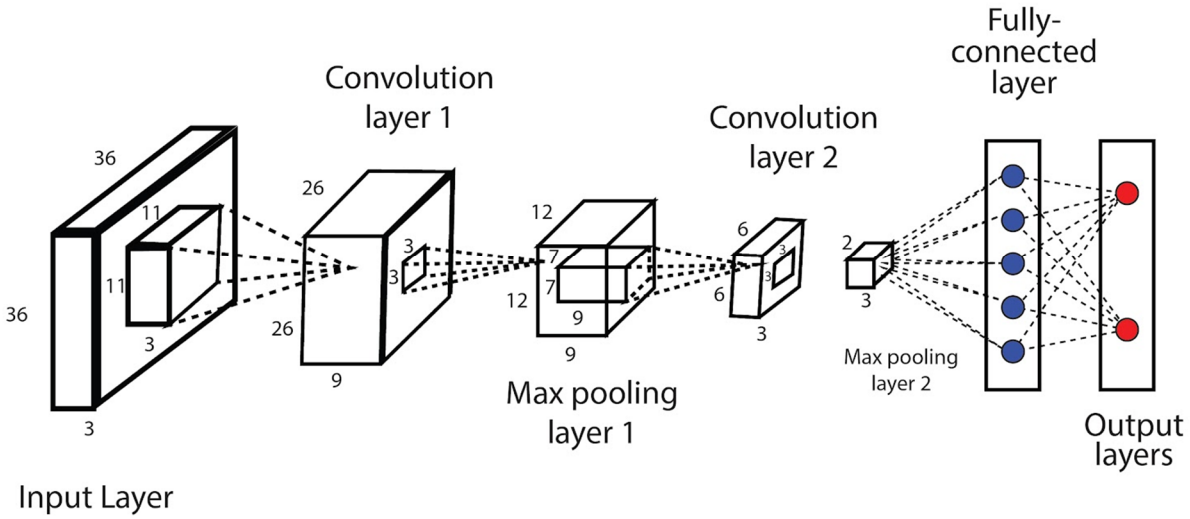
- Activation functions help model learn complex relationship that exists within the dataset. If we do not use the activation function in neurons and give weighted sum as output, in that case, computations will be difficult as there is no specific range for weighted sum. So, the activation function helps to keep output in a particular range.
- Secondly, the non-linear activation function is always preferred as it adds non-linearity to the dataset which otherwise would form a simple linear regression model incapable of taking the benefit of hidden layers. The relu function or its varients is mostly used for hidden layers and sigmoid/ softmax function is mostly used for final layer for binary/ multi-class classification problems.

Loss/ Cost Function

To train the model, we give input (departure location, arrival location and, departure date in case of train price prediction) to the network and let it predict the output making use of activation function. Then, we compare predicted output with the actual output and compute the error between two values. This error between two values is computed using **loss/ cost function**. The same process is repeated for entire training dataset and we get the average loss/error. Now, the objective is to minimize this loss to make the model accurate. There exist weights between each connection of 2 neurons. Initially, weights are randomly initialized and the motive is to update these weights with every iteration to get the minimum value of the loss/ cost function. We can change the weights randomly but that is not efficient method. Here comes the role of **optimizers** which updates weights automatically.

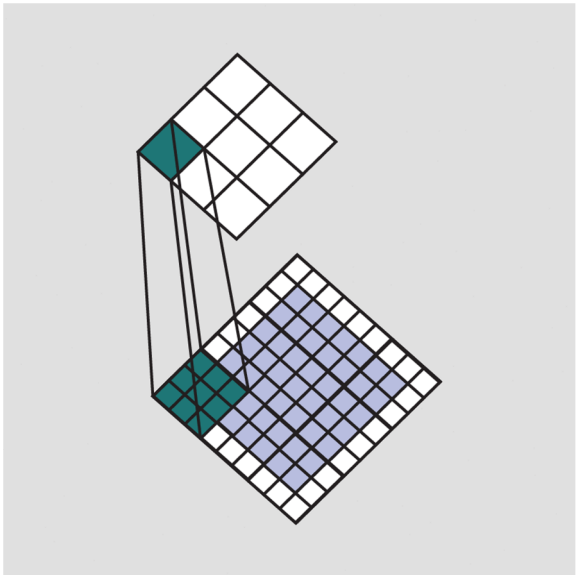
Convolutional Neural Network (CNN)

"Convolution neural networks" indicates that these are simply neural networks with some mathematical operation (generally matrix multiplication) in between their layers called convolution. It was proposed by **Yann LeCun** in 1998. It's one of the most popular uses in Image Classification. Convolution neural network can broadly be classified into these steps: **Input layer, Convolutional layer, Output layers.**



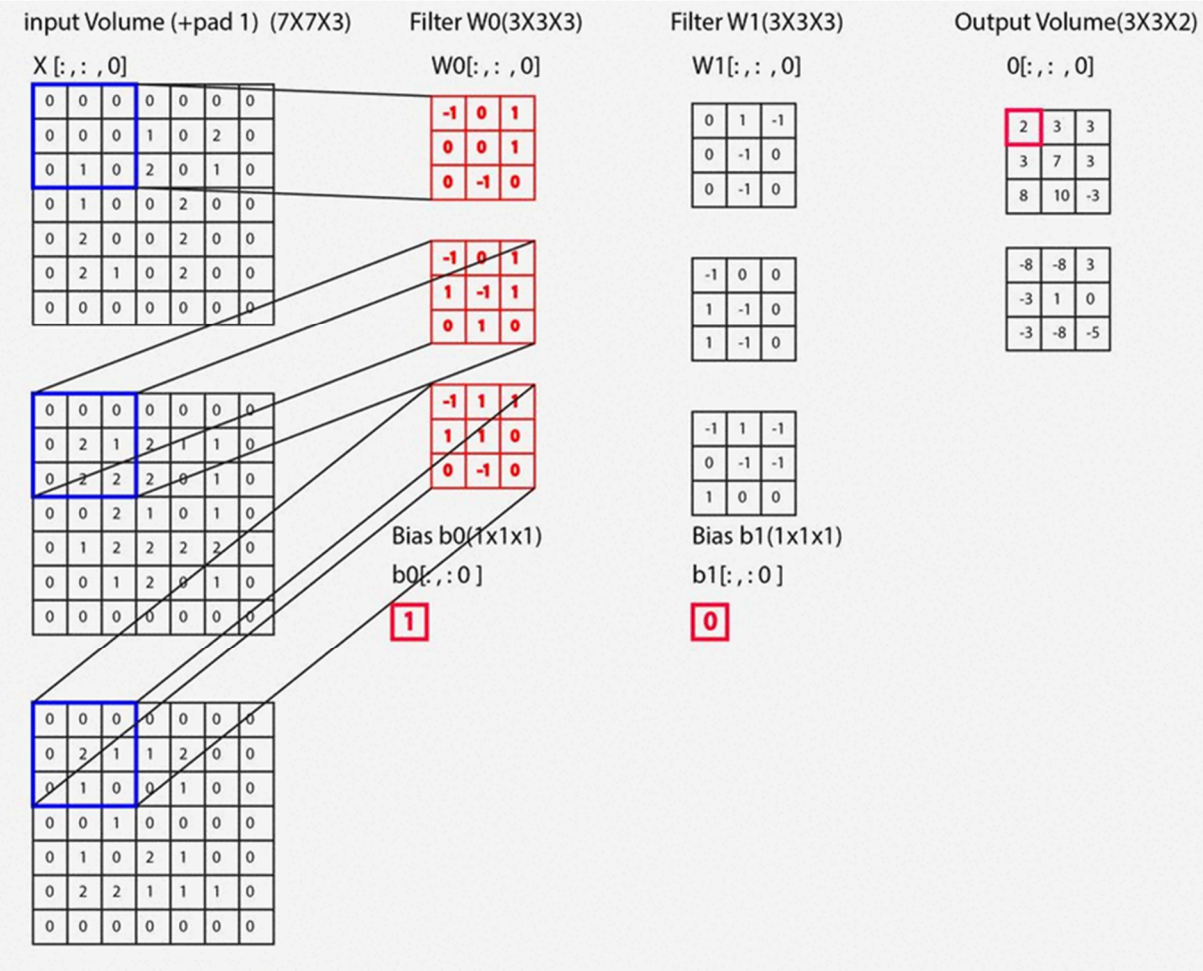
Input layers are connected with convolutional layers that perform many tasks such as padding, striding, the functioning of kernels, and so many performances of this layer, this layer is considered as a building block of convolutional neural networks. We will be discussing it's functioning in detail and how the fully connected networks work.

Convolutional Layer: The convolutional layer's main objective is to extract features from images and learn all the features of the image which would help in object detection techniques. As we know, the input layer will contain some pixel values with some weight and height, our kernels or filters will convolve around the input layer and give results which will retrieve all the features with fewer dimensions. Let's see how kernels work;

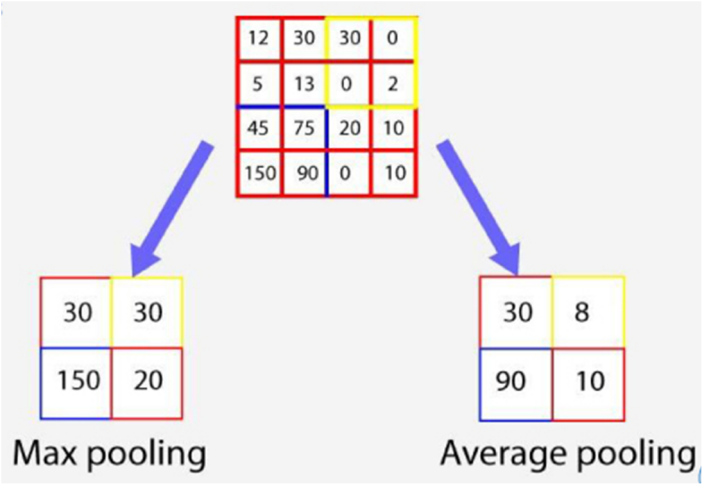


Formation and arrangement of Convolutional Kernels

With the help of this very informative visualization about kernels, we can see how the kernels work and how padding is done.



Matrix visualization in CNN



Matrix formation using Max-pooling and average pooling

Need for Padding: We can see padding in our input volume, we need to do padding in order to make our kernels fit the input matrices. Sometimes we do zero paddings, i.e. adding one row or column to each side of zero matrices or we can cut out the part, which is not fitting in the input image, also known as valid padding. Let's see how we reduce parameters with negligible loss, we use techniques like Max-pooling and average pooling.

Max pooling or Average pooling:

Max pooling or average pooling reduces the parameters to increase the computation of our convolutional architecture. Here, 2*2 filters and 2 strides are taken (which we usually use). By name, we can easily assume that max-pooling extracts the maximum value from the filter and average pooling takes out the average from the filter. We perform pooling to reduce dimensionality. We have to add padding only if necessary. The more convolutional layer can be added to our model until conditions are satisfied.

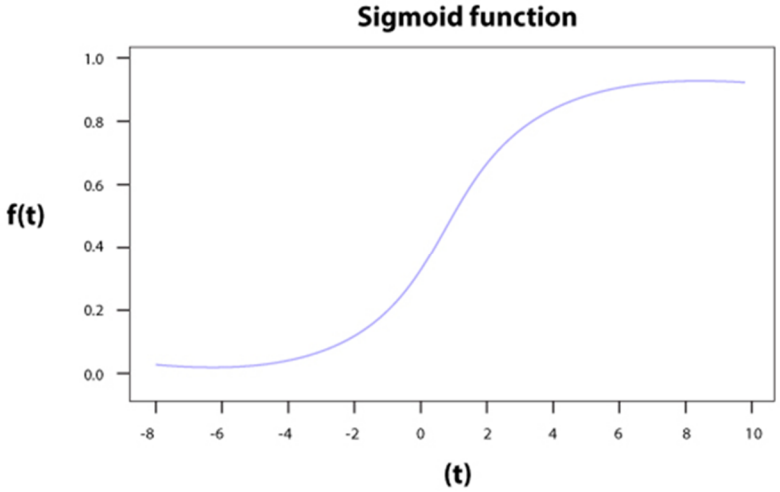
Activation functions in CNN

An activation function is added to our network anywhere in between two convolutional layers or at the end of the network. So you must be wondering what exactly an activation function does, let me clear it in simple words for you. It helps in making the decision about which information should fire forward and which not by making decisions at the end of any network. In broadly, there are both linear as well as non-linear activation functions, both performing linear and non-linear transformations but non-linear activation functions are a lot helpful and therefore widely used in neural networks as well as deep learning networks. The four most famous activation functions to add non-linearity to the network are described below.

1. Sigmoid Activation Function

The equation for the sigmoid function is

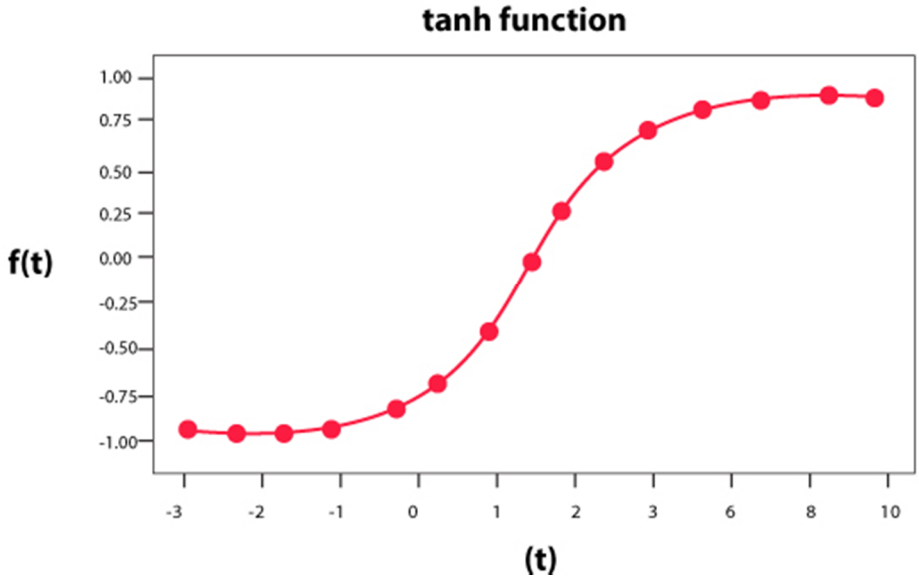
$$f(x) = 1/(1+e^{-x})$$



Sigmoid Activation function

The sigmoid activation function is used mostly as it does its task with great efficiency, it basically is a probabilistic approach towards decision making and ranges in between 0 to 1, so when we have to make a decision or to predict an output we use this activation function because of the range is the minimum, therefore, the prediction would be more accurate.

2. Hyperbolic Tangent Activation Function(Tanh)

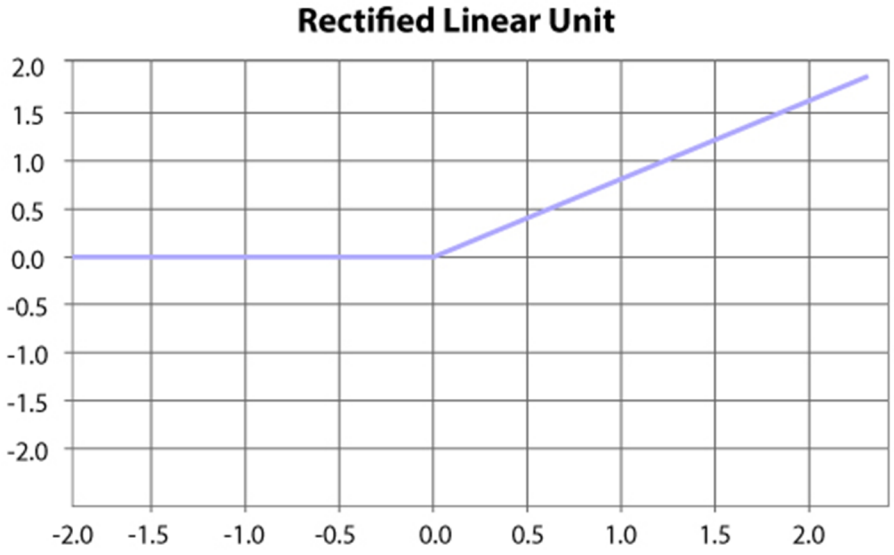


Tanh Activation function

This activation function is slightly better than the sigmoid function, like the sigmoid function it is also used to predict or to differentiate between two classes but it maps the negative input into negative quantity only and ranges in between -1 to 1.

3. ReLU (Rectified Linear unit) Activation function

Rectified linear unit or ReLU is the most widely used activation function right now which ranges from **0 to infinity,** all the negative values are converted into zero, and this conversion rate is so fast that neither it can map nor fit into data properly which creates a problem, but where there is a problem there is a solution.

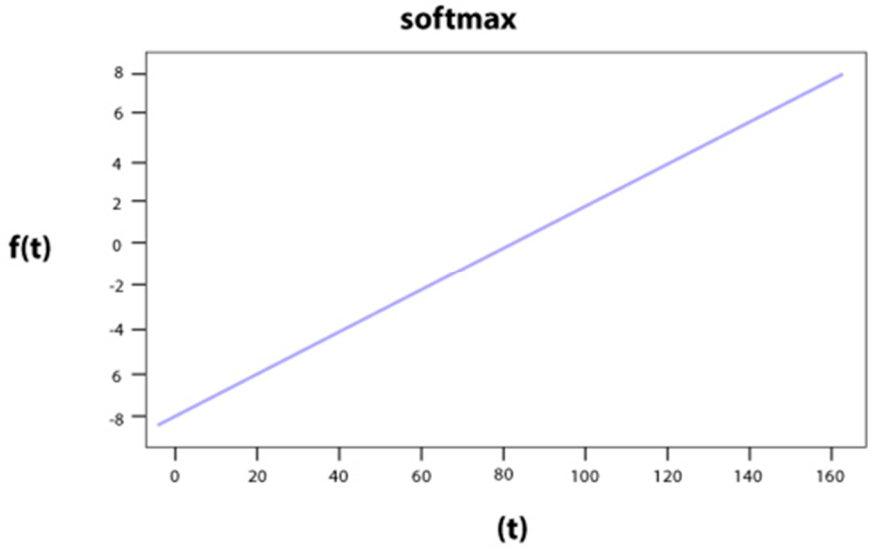


Rectified Linear Unit activation function

We use Leaky ReLU function instead of ReLU to avoid this unfitting, in Leaky ReLU range is expanded which enhances the performance.

4. Softmax Activation Function

Softmax is used mainly at the last layer i.e output layer for decision making the same as sigmoid activation works, the softmax basically gives value to the input variable according to their weight, and the sum of these weights is eventually one.



Softmax activation function

For Binary classification, both sigmoid, as well as softmax, are equally approachable but in the case of multi-class classification problems we generally use softmax and cross-entropy along with it.

How to choose the right Activation Function?

As a rule of thumb, you can begin with the ReLU and then move over to other activation functions if ReLU doesn't provide optimum results. Here are a few other guidelines.

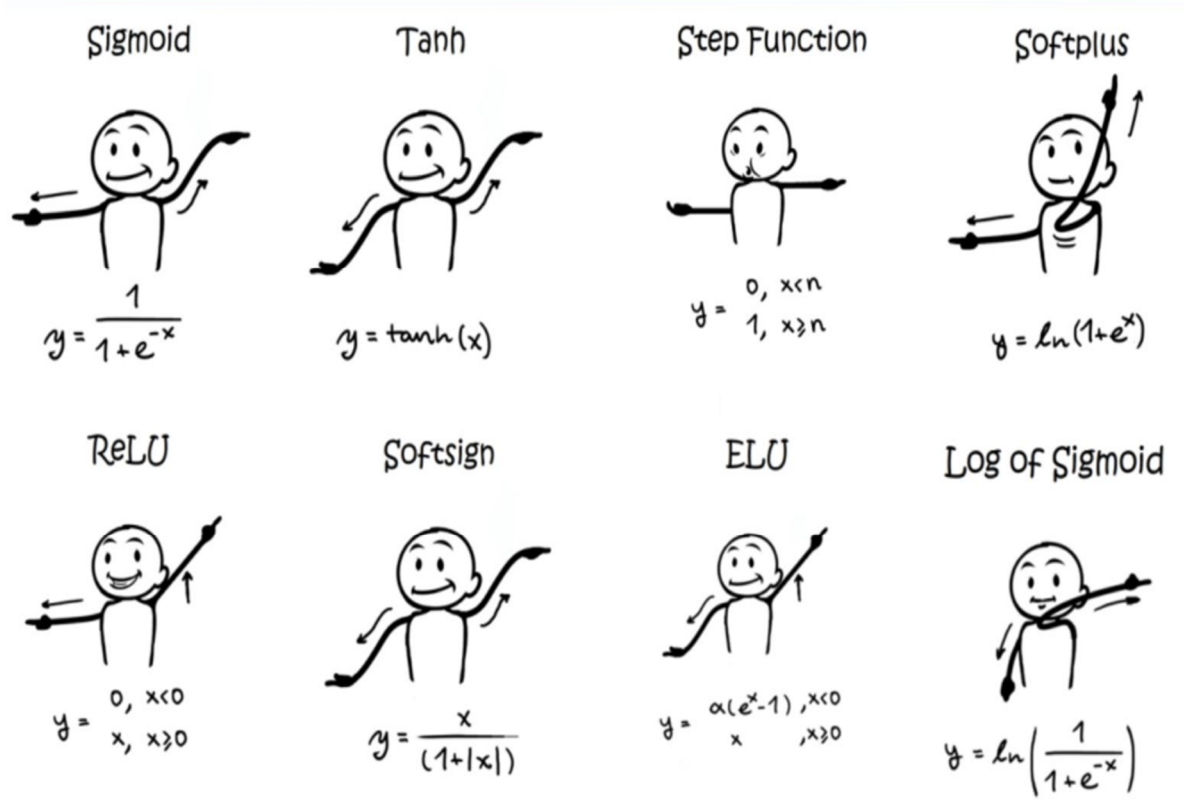
1. ReLU activation function should only be used in the hidden layers.
2. Sigmoid/Logistic and Tanh should not be used in hidden layers as they make the model more susceptible to problems during training (due to vanishing gradients).
3. Swish function is used in neural networks having a depth greater than 40 layers.

Finally, a few rules for choosing the activation function for your output layer based on the type of prediction problem that you are solving:

1. **Regression** - Linear Activation Function
2. **Binary Classification**—Sigmoid/Logistic Activation Function
3. **Multiclass Classification**—Softmax
4. **Multilabel Classification**—Sigmoid

The activation function used in hidden layers is typically chosen based on the type of neural network architecture.

5. **Convolutional Neural Network (CNN)**: ReLU activation function.
6. **Recurrent Neural Network**: Tanh and/or Sigmoid activation function.



Choosing Between Machine Learning and Deep Learning

- Machine learning offers a variety of techniques and models you can choose based on your application, the size of data you're processing, and the type of problem you want to solve.
- A successful deep learning application requires a very large amount of data (thousands of images) to train the model, as well as GPUs, or graphics processing units, to rapidly process your data.
- When choosing between machine learning and deep learning, consider whether you have a high-performance GPU and lots of labeled data.
- If you don't have either of those things, it may make more sense to use machine learning instead of deep learning.
- Deep learning is generally more complex, so you'll need at least a few thousand images to get reliable results.
- Having a high-performance GPU means the model will take less time to analyze all those images.

Feel free to contact me on +91-8329347107 calling / +91-9922369797 whatsapp, email ID: adp.mech@coep.ac.in and abhipatange93@gmail.com

****

AI & ML for Mechanical Engineers

Course Code: ME (DE) - 22016

Part 4: Development of ML model & its evaluation

Final Year Bachelor of Technology (Choice Based Credit System)
Mechanical Engineering
(With Effect from Academic Year 2021-22)

Lecture notes

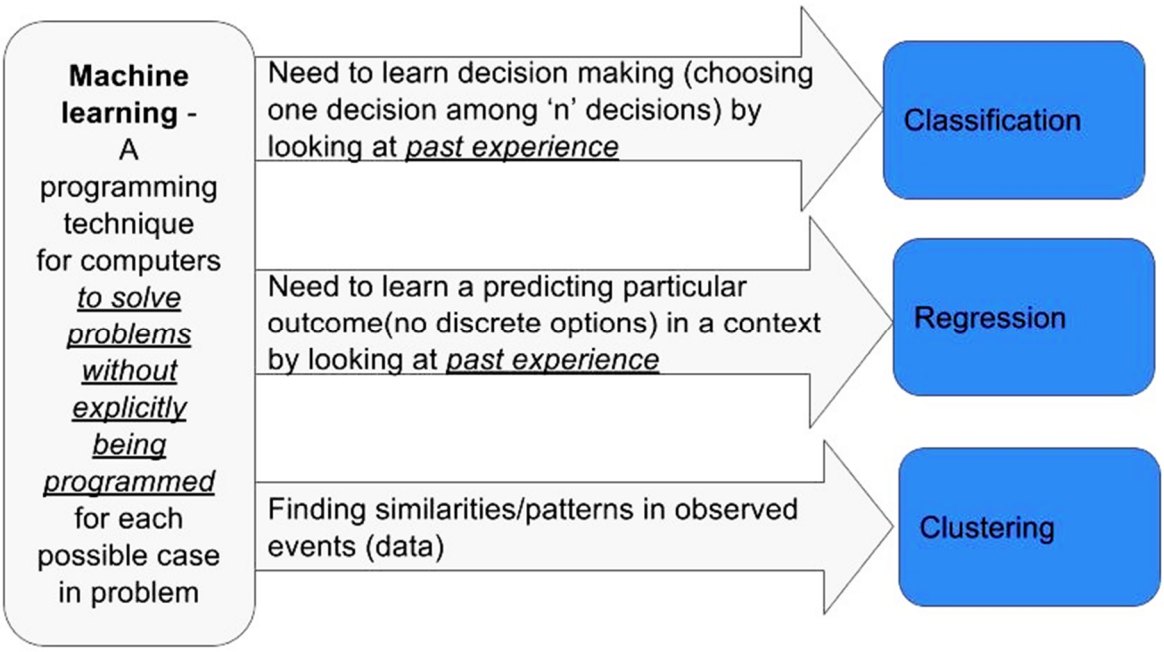
by

Abhishek D. Patange, Ph.D.
Department of Mechanical Engineering

COEP Technological University Pune

Typical problems to be solved using machine learning approach

- **Regression:** If the prediction value tends to be a **continuous value** then it falls under Regression type problem in machine learning. Giving area name, size of land, etc. as features and predicting expected cost of the land.
- **Classification:** If the prediction value tends to be **category/discrete** like yes/no , positive/negative , etc. then it falls under classification type problem in machine learning. Given a sentence predicting whether it is negative or positive review
- **Clustering:** **Grouping** a set of points to given number of clusters. Given 3, 4, 8, 9 and number of clusters to be 2 then the ML system might divide the given set into cluster **1 - 3, 4** and cluster **2 - 8, 9**
- **Ranking:** Used for **constructing a ranker** from a set of labelled examples. This example set consists of instance groups that can be scored with a given criteria. The ranking labels are { 0, 1, 2, 3, 4 } for each instance. The ranker is **trained to rank new instance groups with unknown scores** for each instance.



- Regression means the relationship between 2 “things” (one variable-dependent related to one variable-independent or groups of variable dependent against the group of independent as well as a combination 1:n variables, called multivariate regression). Regression “sees” relationship.
- Classification and clustering both manage groups of something according some criteria(s). Example is grouping by gender, age, some preferences. Difference is easy to see: In classification you define the factors that differentiate the population and put each individual in a specific “drawer” according to your grouping criteria.
- The criteria can be single (one unique factor of differentiation) or by a combination of factors, e.g. gender & birth city). You are classifying the sample.
- In clustering, the process senses differences based on value of variables and assign a specific “drawer” to each case of the sample. After this separation based on math and value of variables, the researcher will try to validate if the grouping has a human logical reason and, if possible, characterize it by human feeling.
- This is very tricky because all the process is essentially numbers because math is unable to “read” the label of each factor and assign human meaning.
- Let’s see an example: consider you measure customer satisfaction using a group of variables (price, quality, service, delivery time, gender, age, education, …) that is supposed to segment the sample in groups based on value of those variables.
- You can define how many groups you want to have, or some tools show it graphically in order to allow the researcher decide what configuration (by number of groups called clusters) would be convenient. Regression measures relationship, classification you define criteria of grouping and separate by that.
- Clustering the groups is defined by “distance” among sample cases and at the end, researcher looks for some meaning of that grouping. Regression, classification and clustering are based on sample content and the result reflects that sample.
- To extrapolate the conclusion to population requires validation according to the Level of Confidence you are willing to assume and additional math processes need to be done.

Clustering is not classification!!

- Usually **classification** is referred to as the problem of **observing data and deciding to which class each item belongs** (e.g. cat or dog).
- Typically the classes are known in advance - something we care about.
- **Clustering** is typically referred to when **we have data but no labels that are associated with each item** (i.e. no predefined notion of cats and dogs).
- In this case the problem is to **cluster the items in to groups** so that items **in each group "resemble each other".**

- It can be that a particular algorithm will classify, for example, pet images into cats and dogs but it may also end up clustering them to other groups (based on colour, pose, size or any other characteristic).
- The key is that **clustering is typically done in the unsupervised domain**, that is: there are no predefined classes known in advance.

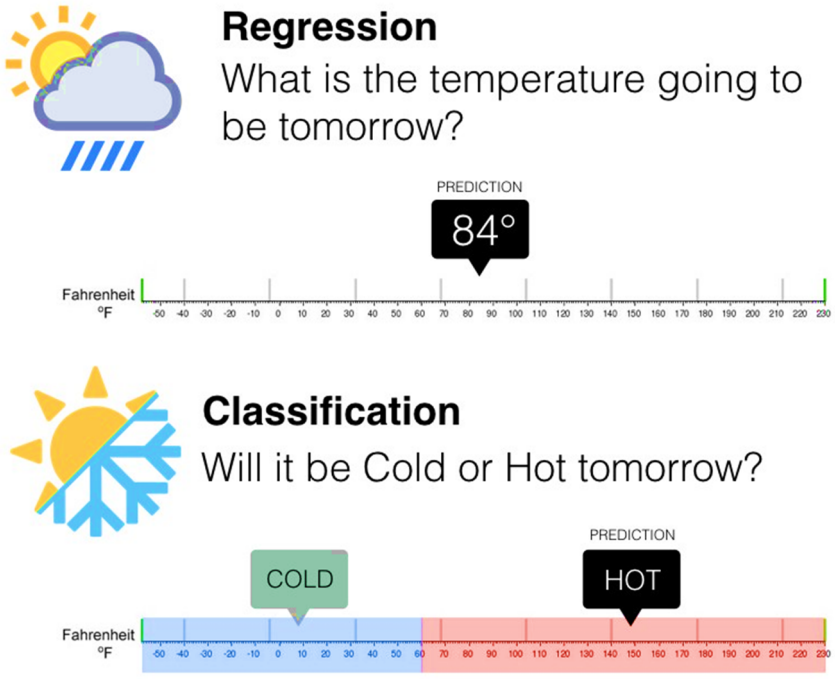
Classification vs. Clustering

Parameter	CLASSIFICATION	CLUSTERING
Type	used for supervised learning	used for unsupervised learning
Basic	process of classifying the input instances based on their corresponding class labels	grouping the instances based on their similarity without the help of class labels
Need	it has labels so there is need of training and testing dataset for verifying the model created	there is no need of training and testing dataset
Complexity	more complex as compared to clustering	less complex as compared to classification
Uses	It uses algorithms to categorize the new data as per the observations of the training set.	It uses statistical concepts in which the data set is divided into subsets with the same features.
Objective	Its objective is to find which class a new object belongs to form the set of predefined classes.	Its objective is to group a set of objects to find whether there is any relationship between them.
Example Algorithms	Logistic regression, Naive Bayes classifier, Support vector machines, etc.	k-means clustering algorithm, Fuzzy c-means clustering algorithm, Gaussian (EM) clustering algorithm, etc.

Classification vs. Regression

Regression	Classification
In Classification, the output variable must be a discrete value.	In Regression, the output variable must be of continuous nature or real value.
The task of the classification algorithm is to map the input value(x) with the discrete output variable(y).	The task of the regression algorithm is to map the input value (x) with the continuous output variable(y).
Classification Algorithms are used with discrete	Regression Algorithms are used with

data.	continuous data.
In Classification, we try to find the decision boundary, which can divide the dataset into different classes.	In Regression, we try to find the best fit line, which can predict the output more accurately.
Classification Algorithms can be used to solve classification problems such as Identification of spam emails, Speech Recognition, Identification of cancer cells, etc.	Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc.
The Classification algorithms can be divided into Binary Classifier and Multi-class Classifier.	The regression Algorithm can be further divided into Linear and Non-linear Regression.
Method of evaluation by measuring accuracy	Method of evaluation by measurement of root mean square error
Nature of the predicted data Unordered	Nature of the predicted data Ordered



Terminology of understanding ML based classification and regression model.

- **Algorithm =** A method, function, or series of instructions used to generate a machine learning model. Examples include linear regression, decision trees, support vector machines, and neural networks.
- **Attribute =** A quality describing an observation (e.g. color, size, weight). In Excel terms, these are column headers.

- **Categorical Variables =** Variables with a discrete set of possible values. Can be ordinal (order matters) or nominal (order doesn't matter).
- **Classification =** Predicting a categorical output.
- **Binary classification** = predicts one of two possible outcomes (e.g. is the email spam or not spam?)
- **Multi-class classification =** predicts one of multiple possible outcomes (e.g. is this a photo of a cat, dog, horse or human?)
- **Classification Threshold =** The lowest probability value at which we're comfortable asserting a positive classification. For example, if the predicted probability of being diabetic is > 50%, return True, otherwise return False.
- **Classifier** =It is an algorithm that is used to map the input data to a specific category.
- **Classification Model =** The model predicts or draws a conclusion to the input data given for training, it will predict the class or category for the data.
- **Binary Classification** = It is a type of classification with two outcomes, for e.g. – either true or false.
- **Multi-Class Classification** = The classification with more than two classes, in multi-class classification each sample is assigned to one and only one label or target.
- **Multi-label Classification** = This is a type of classification where each sample is assigned to a set of labels or targets.
- **Clustering =** Unsupervised grouping of data into buckets.
- **Confusion Matrix =** Table that describes the performance of a classification model by grouping predictions into 4 categories.
- **True Positives**: we *correctly* predicted they do have diabetes
- **True Negatives**: we *correctly* predicted they don't have diabetes
- **False Positives**: we *incorrectly* predicted they do have diabetes (Type I error)
- **False Negatives**: we *incorrectly* predicted they don't have diabetes (Type II error)
- **Continuous Variables =** Variables with a range of possible values defined by a number scale (e.g. sales, lifespan).
- **Convergence =** A state reached during the training of a model when the loss changes very little between each iteration.
- **Epoch =** An epoch describes the number of times the algorithm sees the entire data set.
- **Feature =** With respect to a dataset, a feature represents an attribute and value combination. Color is an attribute. "Color is blue" is a feature. In Excel terms, features are similar to cells. The term feature has other definitions in different contexts.
- **Feature Selection =** Feature selection is the process of selecting relevant features from a data-set for creating a Machine Learning model.

- **Feature Vector =** A list of features describing an observation with multiple attributes. In Excel we call this a row.
- **Hyperparameters =** Hyperparameters are higher-level properties of a model such as how fast it can learn (learning rate) or complexity of a model. The depth of trees in a Decision Tree or number of hidden layers in a Neural Networks are examples of hyper parameters.
- **Instance =** A data point, row, or sample in a dataset. Another term for observation.
- **Label =** The "answer" portion of an observation in supervised learning. For example, in a dataset used to classify flowers into different species, the features might include the petal length and petal width, while the label would be the flower's species.
- **Model =** A data structure that stores a representation of a dataset (weights and biases). Models are created/learned when you train an algorithm on a dataset.
- **Neural Networks =** Neural Networks are mathematical algorithms modeled after the brain's architecture, designed to recognize patterns and relationships in data.
- **Normalization =** Restriction of the values of weights in regression to avoid overfitting and improving computation speed.
- **Noise =** Any irrelevant information or randomness in a dataset which obscures the underlying pattern.
- **Observation =** A data point, row, or sample in a dataset. Another term for instance.
- **Outlier =** An observation that deviates significantly from other observations in the dataset.
- **Overfitting =** Overfitting occurs when your model learns the training data too well and incorporates details and noise specific to your dataset. You can tell a model is overfitting when it performs great on your training/validation set, but poorly on your test set (or new real-world data).
- **Regression =** Predicting a continuous output (e.g. price, sales).
- **Supervised Learning =** Training a model using a labeled dataset.
- **Test Set =** A set of observations used at the end of model training and validation to assess the predictive power of your model. How generalizable is your model to unseen data?
- **Training Set =** A set of observations used to generate machine learning models.
- **Underfitting =** Underfitting occurs when your model over-generalizes and fails to incorporate relevant variations in your data that would give your model more predictive power. You can tell a model is underfitting when it performs poorly on both training and test sets.
- **Unsupervised Learning =** Training a model to find patterns in an unlabeled dataset (e.g. clustering).

- **Validation Set =** A set of observations used during model training to provide feedback on how well the current parameters generalize beyond the training set. If training error decreases but validation error increases, your model is likely overfitting and you should pause training.

Steps involved in development of classification model

Following are the steps to be considered in development of classification model.

1 - Data Collection

- The quantity & quality of your data dictate how accurate our model is
- The outcome of this step is generally a representation of data which we will use for training
- Using experimental data, data generated by simulations, pre-collected data, by way of datasets from Kaggle, UCI, etc., still fits into this step

2 - Data Preparation

- Wrangle data and prepare it for training
- Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, and data type conversions, etc.)
- Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data
- Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis
- Split into training and evaluation sets

3 - Choose a Model

- Different algorithms are for different tasks; choose the right one

4 - Train the Model

- The goal of training is to answer a question or make a prediction correctly as often as possible
- Linear regression example: algorithm would need to learn values for m (or W) and b (x is input, y is output)
- Each iteration of process is a training step

5 - Evaluate the Model

- Uses some metric or combination of metrics to "measure" objective performance of model
- Test the model against previously unseen data
- This unseen data is meant to be somewhat representative of model performance in the real world, but still helps tune the model (as opposed to test data, which does not)

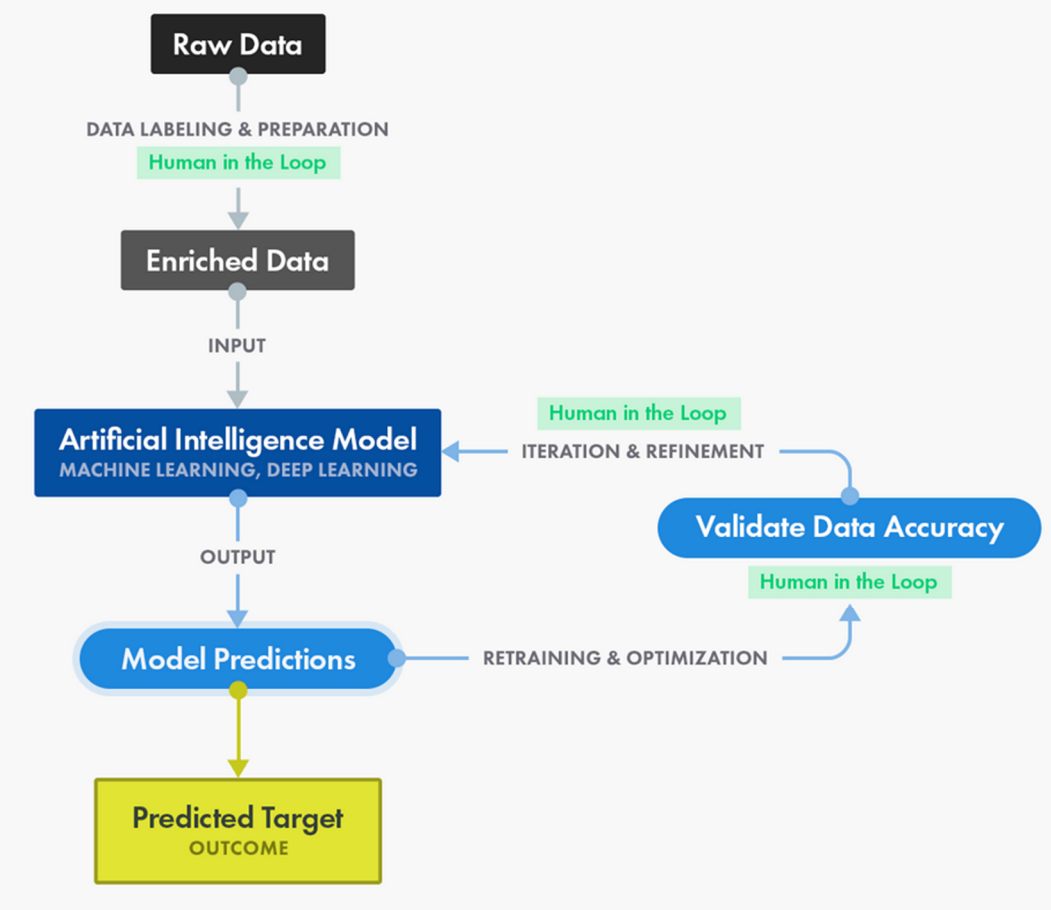
- Good train/evaluation split? 80/20, 70/30, or similar, depending on domain, data availability, dataset particulars, etc.

6 – Hyper parameter Tuning

- This step refers to *hyperparameter* tuning, which is an "artform" as opposed to a science
- Tune model parameters for improved performance
- Simple model hyperparameters may include: number of training steps, learning rate, initialization values and distribution, etc.

7 - Make Predictions

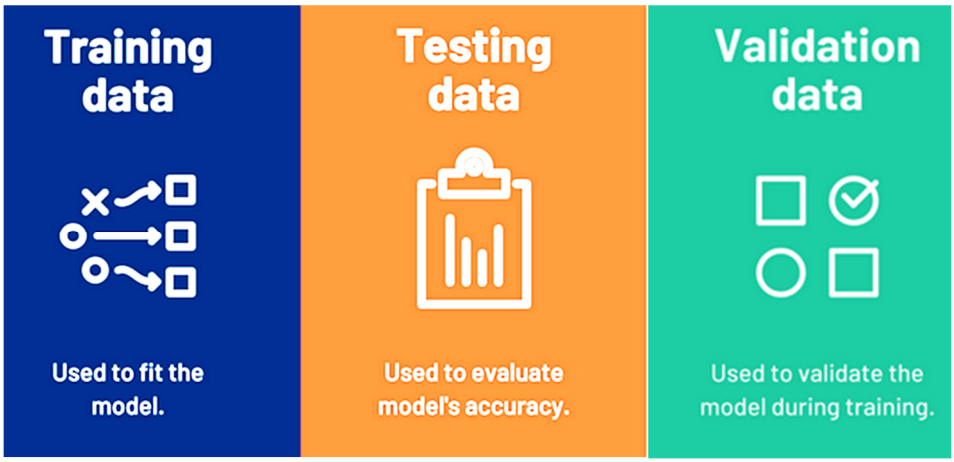
- Using further (test set) data which have, until this point, been withheld from the model (and for which class labels are known), are used to test the model; a better approximation of how the model will perform in the real world

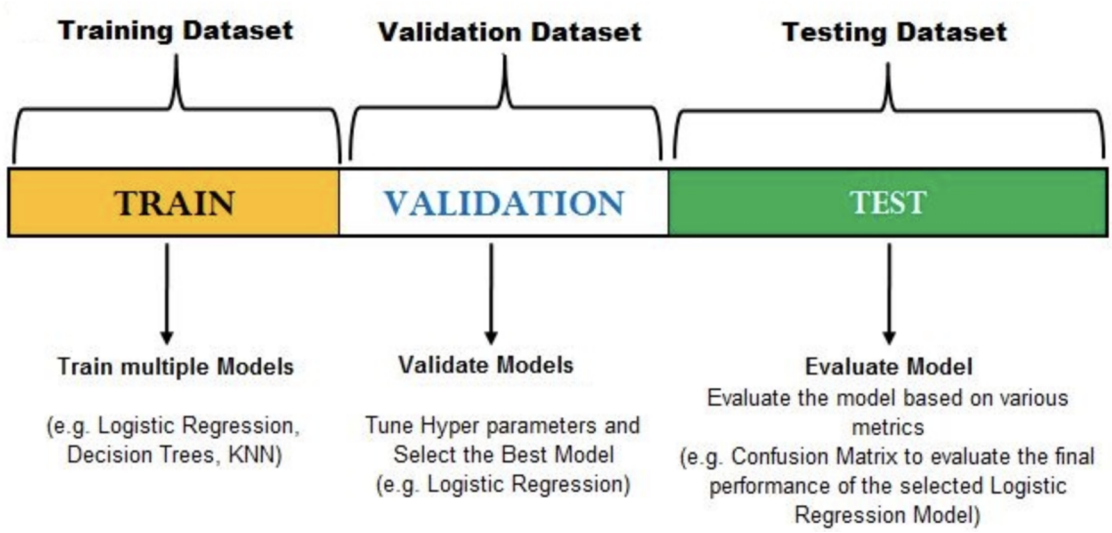


Training data vs. test data vs. validation data

- **Training data** is used in model training, or in other words, it's the data used to fit the model. On the contrary, **test data** is used to evaluate the performance or accuracy of the model. It's a sample of data used to make an unbiased evaluation of the final model fit on the training data.

- A training dataset is an initial dataset that teaches the ML models to identify desired patterns or perform a particular task. A testing dataset is used to evaluate how effective the training was or how accurate the model is.



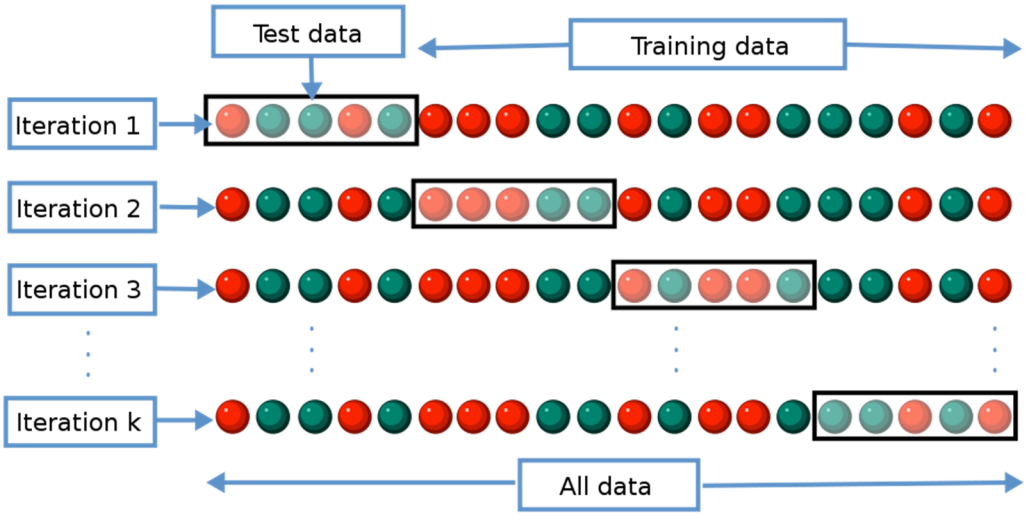


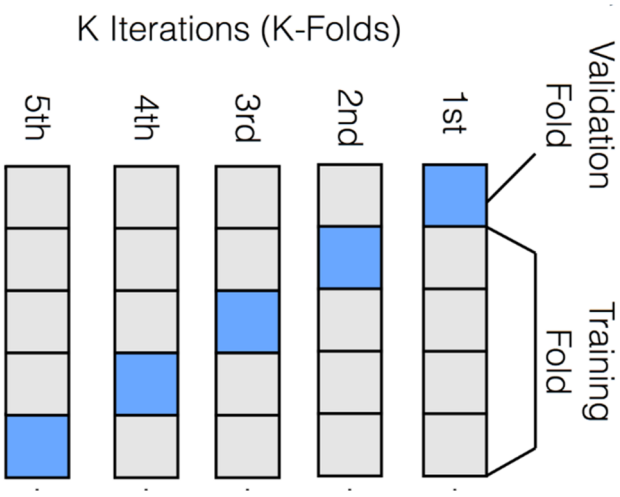
- Once an ML algorithm is trained on a particular dataset and if you test it on the same dataset, it's more likely to have high accuracy because the model knows what to expect. If the training dataset contains all possible values the model might encounter in the future, all well and good.
- But that's never the case. A training dataset can never be comprehensive and can't teach everything that a model might encounter in the real world. Therefore a test dataset, containing *unseen* data points, is used to evaluate the model's accuracy.
- Then there's **validation data**. This is a dataset used for frequent evaluation during the training phase. Although the model sees this dataset occasionally, it doesn't *learn* from it. The validation set is also referred to as the development set or dev set. It helps protect models from overfitting and underfitting.
- Although validation data is separate from training data, data scientists might reserve a part of the training data for validation. But of course, this automatically means that the validation data was kept away during the training.

- Many use the terms "test data" and "validation data" interchangeably. The main difference between the two is that validation data is used to validate the model during the training, while the testing set is used to test the model after the training is completed.

K-fold cross-validation mode

- The classifier model can be designed/trained and performance can be evaluated based on **K-fold cross-validation mode, training mode and test mode.**
- The main idea behind **K-Fold cross-validation** is that **each sample** in our dataset has the **opportunity of being tested**. It is a special case of cross-validation where we **iterate over a dataset set k times**. In each round, we **split the dataset into k parts**: **one part is used for validation**, and the **remaining k-1 parts are merged into a training subset** for model evaluation
- **Computation time is reduced** as we repeated the process only 10 times when the value of k is 10. It has **Reduced bias.**
- **Every data points get to be tested exactly once** and is used in training k-1 times
- The **variance of the resulting estimate is reduced as k increases**





Hyper parameter tuning

- Machine learning algorithms have hyperparameters that **allow you to tailor the behavior of the algorithm to your specific dataset.**
- Hyperparameters are different from parameters, which are the **internal coefficients or weights** for a model found by the learning algorithm. Unlike parameters, hyperparameters are specified by the practitioner when configuring the model.
- Typically, it is challenging to know what values to use for the hyperparameters of a given algorithm on a given dataset, therefore it is common to **use random or grid search strategies for different hyperparameter values**.
- The **more hyperparameters** of an algorithm that you need to tune, **the slower the tuning process.** Therefore, it is desirable **to select a minimum subset of model hyperparameters to search or tune.**

Hyper parameter tuning for simple decision tree

Max_Depth: The maximum depth of the tree. If this is not specified in the Decision Tree, the nodes will be expanded until all leaf nodes are pure or until all leaf nodes contain less than min_samples_split.

- Default = None
- Input options → integer

Min_Samples_Split: The minimum samples required to split an internal node. If the amount of sample in an internal node is less than the min_samples_split, then that node will become a leaf node.

- Default = 2
- Input options → integer or float (if float, then min_samples_split is fraction)

Min_Samples_Leaf: The minimum samples required to be at a leaf node. Therefore, a split can only happen if it leaves at least the min_samples_leaf in both of the resulting nodes.

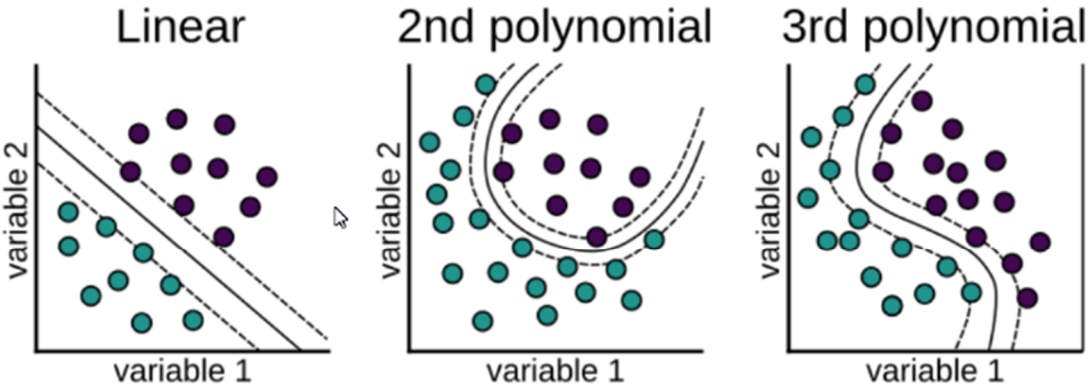
- Default = 1
- Input options → integer or float (if float, then min_samples_leaf is fraction)

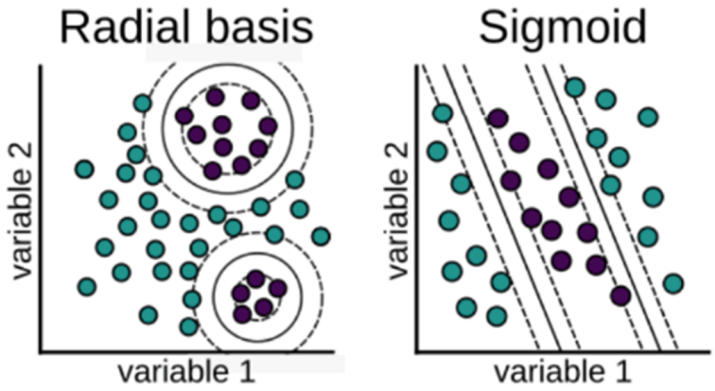
Max_Features: The number of features to consider when looking for the best split. For example, if there are 35 features in a dataframe and max_features is 9, only 9 of the 35 features will be used in the decision tree.

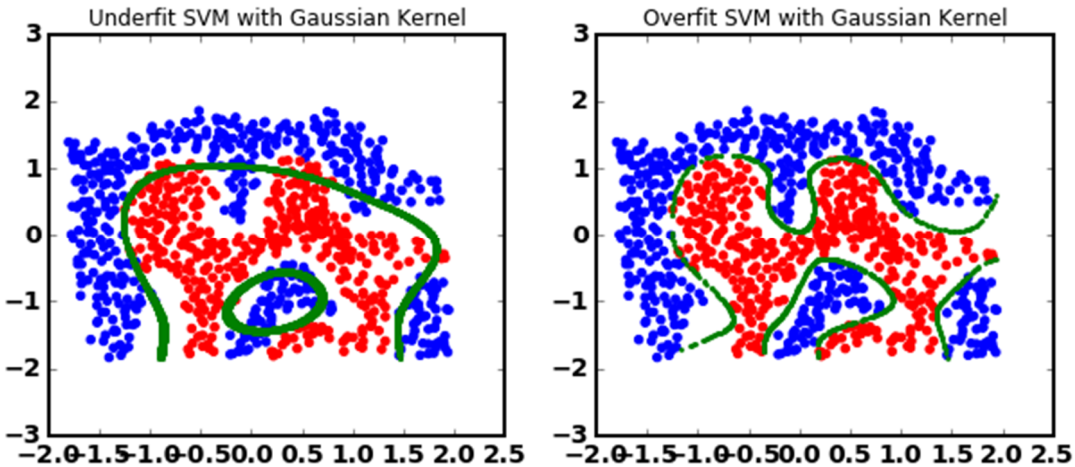
- Default = None
- Input options → integer, float (if float, then max_features is fraction) or {“auto”, “sqrt”, “log2”}
- “auto”: max_features=sqrt(n_features)
- “sqrt”: max_features = sqrt(n_features)
- “log2”: max_features=log2(n_features)

Hyper parameter tuning for SVM

- The **choice of kernel that will control the manner in which the input variables will be projected.** There are many to choose from, but **linear, polynomial, and RBF** are the most common, perhaps just linear and RBF in practice.
- **kernels in ['linear', 'poly', 'rbf', 'sigmoid']**
- If the polynomial kernel works out, then it is a good idea to dive into the degree hyperparameter.
- Another critical parameter is the **penalty (C)** that can take on a range of values and has a dramatic effect on the shape of the resulting regions for each class. A log scale might be a good starting point.
- **C in [100, 10, 1.0, 0.1, 0.001]**

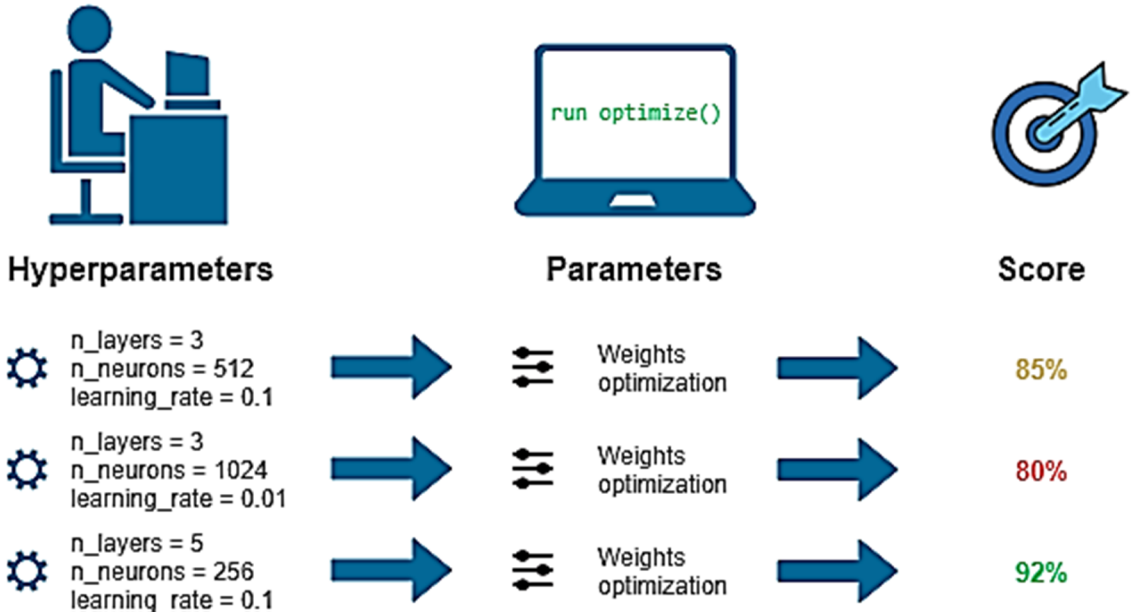






Hyper parameter tuning for ANN

- **Number of neurons:** A weight is the amplification of input signals to a neuron and bias is an additive bias term to a neuron.
- **Activation function:** Defines how a neuron or group of neurons activate ("spiking") based on input connections and bias term(s).
- **Learning rate:** Step length for gradient descent update
- **Batch size:** Number of training examples in each gradient descent (gd) update.
- **Epochs:** The number of times all training examples have been passed through the network during training.
- **Loss function:** Loss function specifies how to calculate the error between prediction and label for a given training example. The error is backpropagated during training in order to update learnable parameters.
- **Number of layers:** Typically layers between input and output layer, which are called hidden layers



Confusion matrix and evaluation of ML models

- **Confusion matrix:** A Confusion matrix is an **N x N matrix** used for evaluating the performance of a classification model, where **N is the number of target classes**. The matrix compares the **actual target values with those predicted by the machine learning model**
- **What can we learn from this matrix?**
- There are **two possible predicted classes: "yes" and "no".** If we were predicting the presence of a disease, for example, **"yes" would mean they have the disease,** and **"no" would mean they don't have the disease.**

- The classifier made a total of **165 predictions** (e.g., **165 patients were being tested** for the presence of that disease).
- **Out of those 165 cases,** the classifier **predicted "yes" 110 times,** and **"no" 55 times.**
- **In reality, 105 patients in the sample have the disease,** and **60 patients do not.**

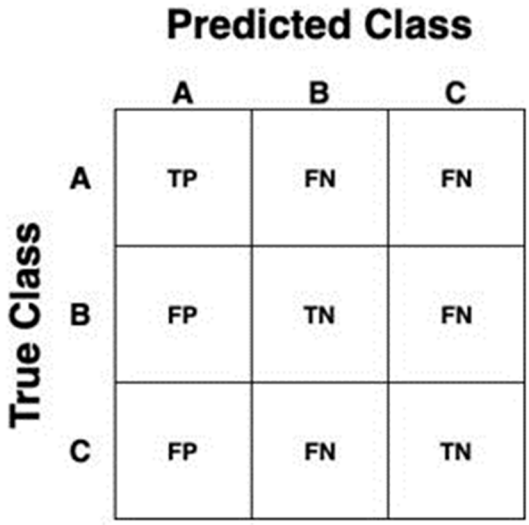
n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

- **True positives (TP):** these are cases in which we predicted yes (they have the disease), and they do have the disease.
- **True negatives (tn):** we predicted no, and they don't have the disease.
- **False positives (fp):** we predicted yes, but they don't actually have the disease. (Also known as a "type I error.")
- **False negatives (fn):** we predicted no, but they actually do have the disease. (Also known as a "type II error.")

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- **Accuracy:** Overall, how often is the classifier correct? (TP+TN)/total = (100+50)/165 = 0.91
- **Misclassification Rate:** Overall, how often is it wrong? (FP+FN)/total = (10+5)/165 = 0.09 which is equivalent to 1 minus Accuracy
- **True Positive Rate:** When it's actually yes, how often does it predict yes? TP/actual yes = 100/105 = 0.95 also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes? FP/actual no = 10/60 = 0.17
- **True Negative Rate:** When it's actually no, how often does it predict no? TN/actual no = 50/60 = 0.83 which is equal to 1 minus False Positive Rate

- **Precision:** When it predicts yes, how often is it correct? TP/predicted yes = 100/110 = 0.91
- **Cohen's Kappa:** This is essentially a measure of how well the classifier performed as compared to how well it would have performed simply by chance. In other words, a model will have a high Kappa score if there is a big difference between the accuracy and the null error rate. (More details about Cohen's Kappa.)
- **F Score:** This is a weighted average of the true positive rate (recall) and precision. (More details about the F Score.)
- **ROC Curve:** This is a commonly used graph that summarizes the performance of a classifier over all possible thresholds. It is generated by plotting the True Positive Rate (y-axis) against the False Positive Rate (x-axis) as you vary the threshold for assigning observations to a given class. (More details about ROC Curves.)



Classified Data \ Reference Data	AG	WA	DE	MF	Row Total
AG	25	4	22	24	75
WA	6	21	10	9	46
DE	0	12	85	19	116
MF	5	8	3	90	106
Column Total	36	45	120	142	343

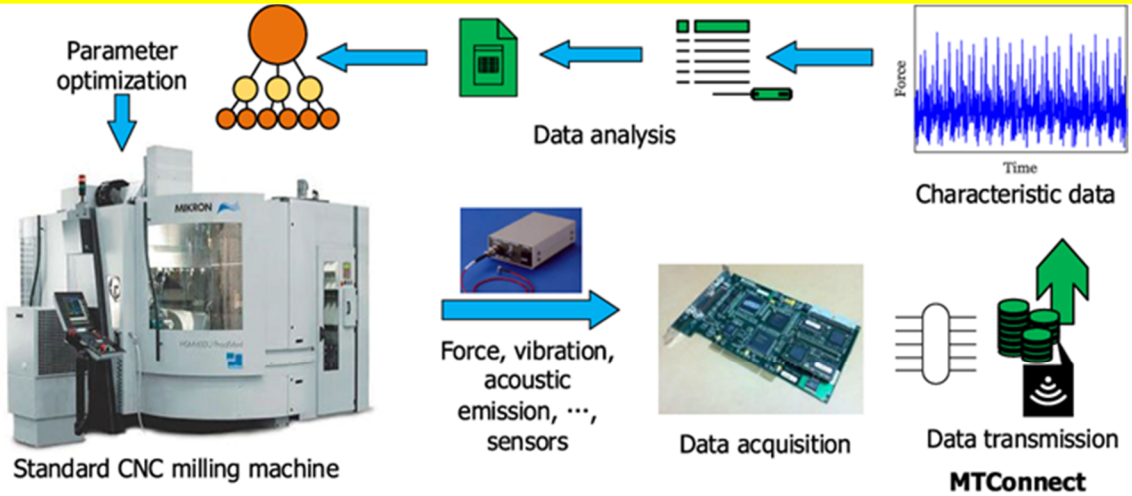
Land Cover Categories:
AG: Agriculture
WA: Water
DE: Deciduous
MF: Mixed Forest

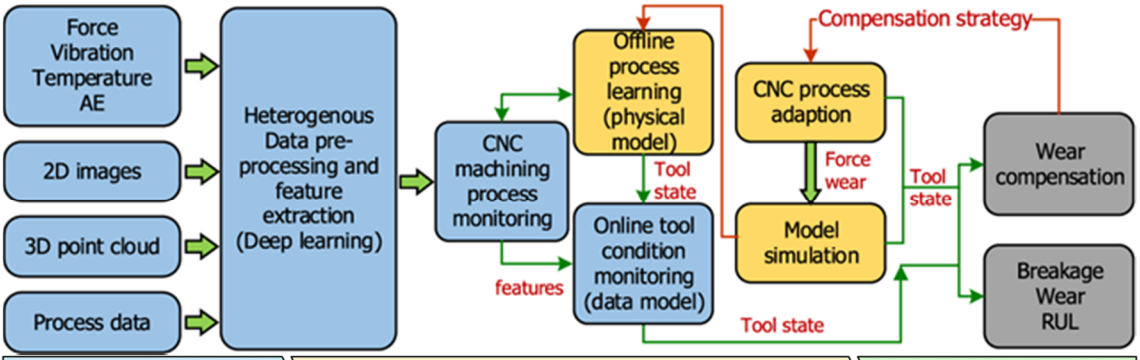
Overall Accuracy=
(25+21+85+90)/343= 64%

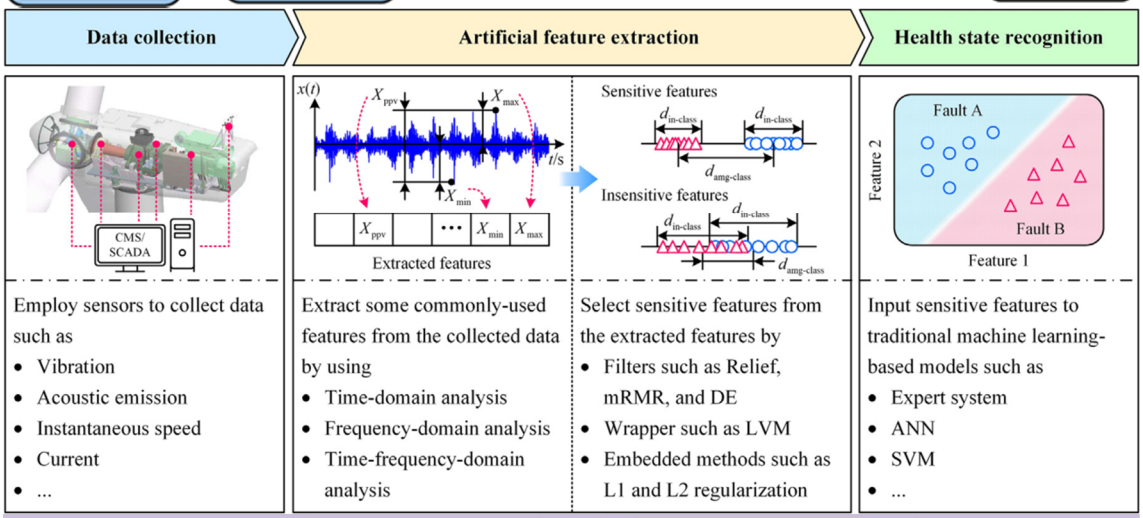
Producer's Accuracy
AG = 25/36=69%
WA=21/45=46%
DE= 85/120=70%
MF=90/142=63%

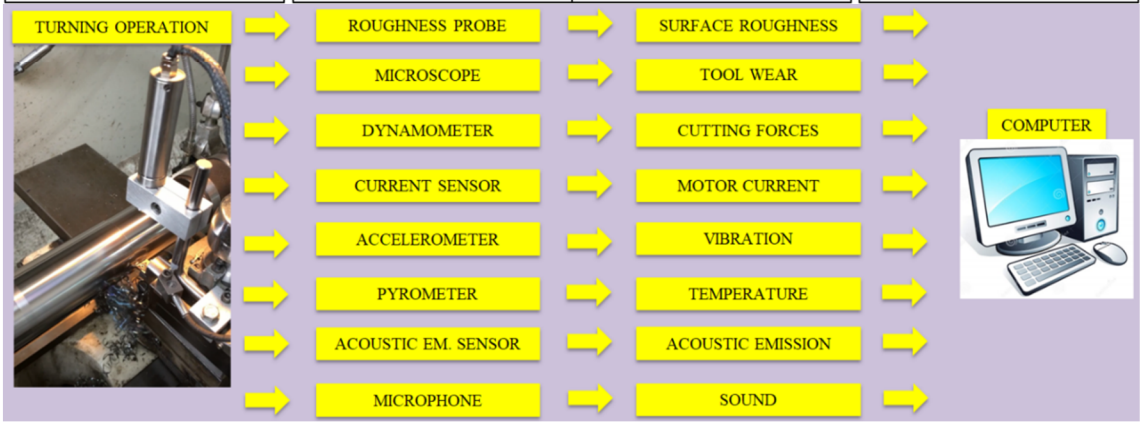
User's Accuracy
AG = 25/75 = 33%
WA=21/46 = 45%
DE=85/116 = 73%
MF=90/106 = 84%

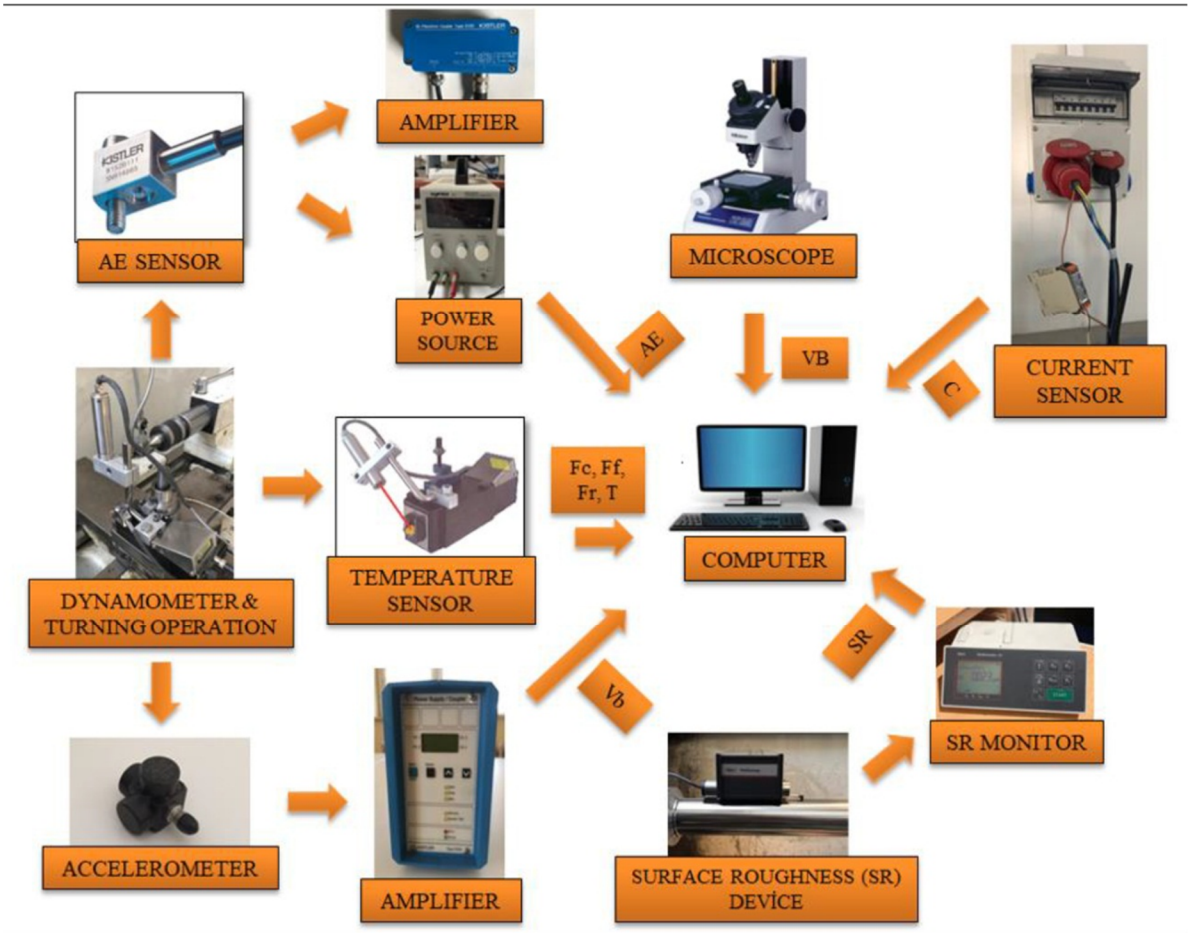
Sources of the data that is needed for training the classification model for identifying cutting tool state in milling/drilling/lathe



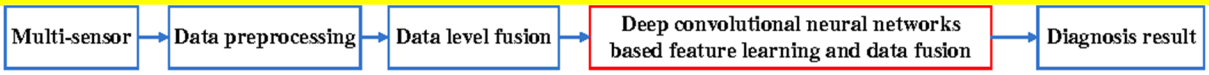


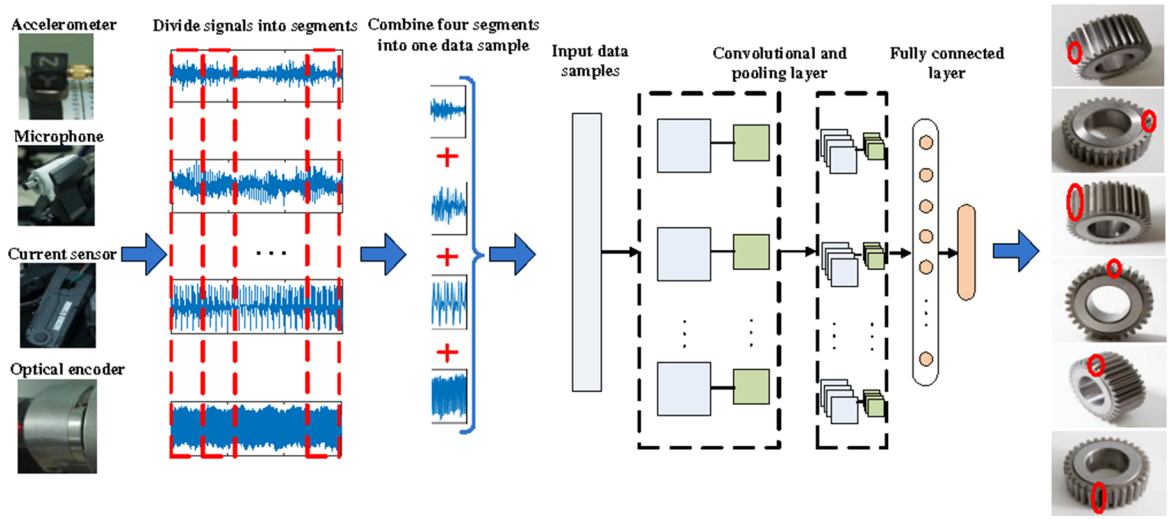




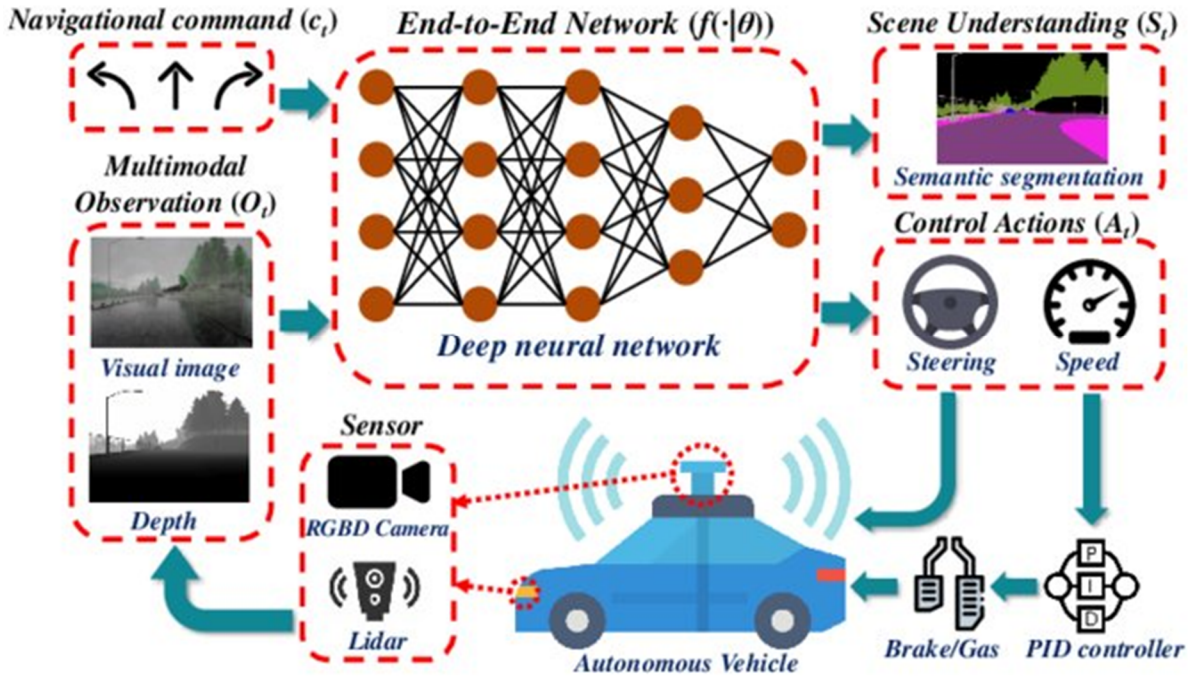


Sources of the data that is needed for training the classification model for condition monitoring of bearings, gears, rotating elements

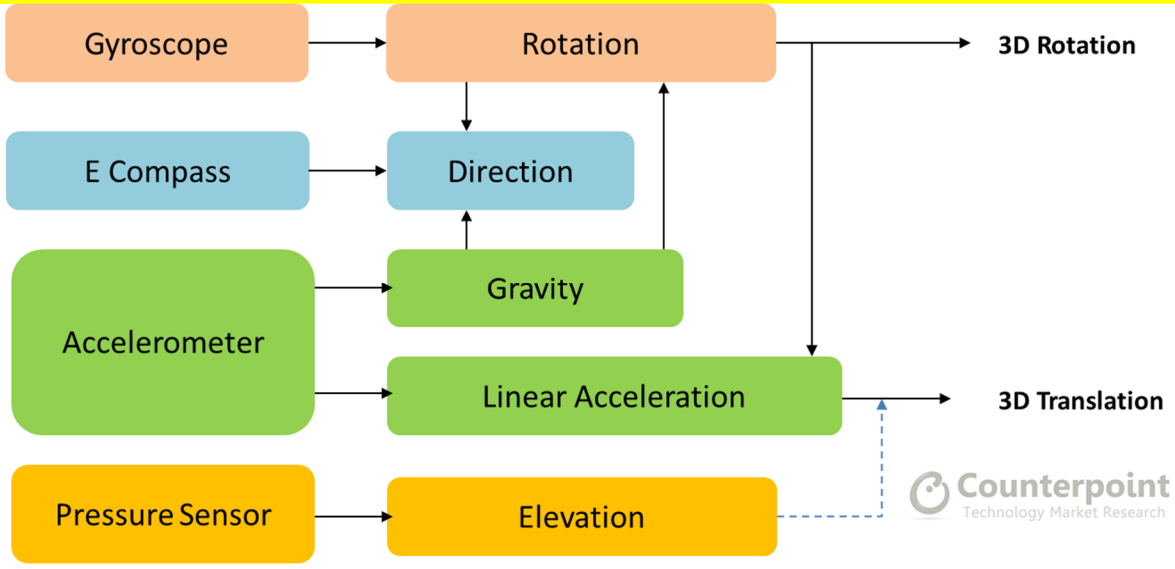




Sources of the data that is needed for training the ML model for End-to-End Autonomous Driving With Scene Understanding



Sources of the data that is needed for training the ML model for accurately predicting the device's position in three dimensions (3D motion sensing using Sensor Fusion)



Feel free to contact me on +91-8329347107 calling / +91-9922369797 whatsapp, email ID: adp.mech@coep.ac.in and abhipatange93@gmail.com

****