UNIT 4: Learning from agents

Agents in Artificial Intelligence

In artificial intelligence, an agent is a computer program or system that is designed to perceive its environment, make decisions and take actions to achieve a specific goal or set of goals. The agent operates autonomously, meaning it is not directly controlled by a human operator.

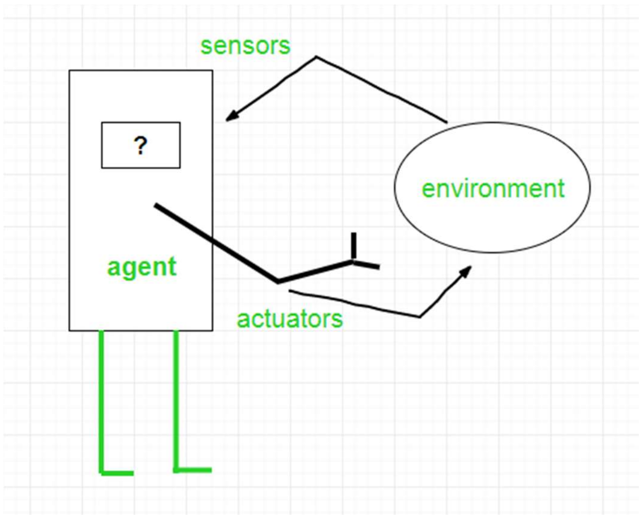
Agents can be classified into different types based on their characteristics, such as whether they are reactive or proactive, whether they have a fixed or dynamic environment, and whether they are single or multi-agent systems.

- Reactive agents are those that respond to immediate stimuli from their environment and take actions based on those stimuli. Proactive agents, on the other hand, take initiative and plan ahead to achieve their goals. The environment in which an agent operates can also be fixed or dynamic. Fixed environments have a static set of rules that do not change, while dynamic environments are constantly changing and require agents to adapt to new situations.
- Multi-agent systems involve multiple agents working together to achieve a common goal. These agents may have to coordinate their actions and communicate with each other to achieve their objectives. Agents are used in a variety of applications, including robotics, gaming, and intelligent systems. They can be implemented using different programming languages and techniques, including machine learning and natural language processing.

Artificial intelligence is defined as the study of rational agents. A rational agent could be anything that makes decisions, such as a person, firm, machine, or software. It carries out an action with the best outcome after considering past and current percepts(agent's perceptual inputs at a given instance). An AI system is composed of an agent and its environment. The agents act in their environment. The environment may contain other agents.

An agent is anything that can be viewed as:

- Perceiving its environment through **sensors** and
- Acting upon that environment through **actuators**



Structure of an AI Agent

To understand the structure of Intelligent Agents, we should be familiar with *Architecture* and *Agent* programs. **Architecture** is the machinery that the agent executes on. It is a device with sensors and actuators, for example, a robotic car, a camera, and a PC. **An agent program** is an implementation of an agent function. An **agent function** is a map from the percept sequence(history of all that an agent has perceived to date) to an action.

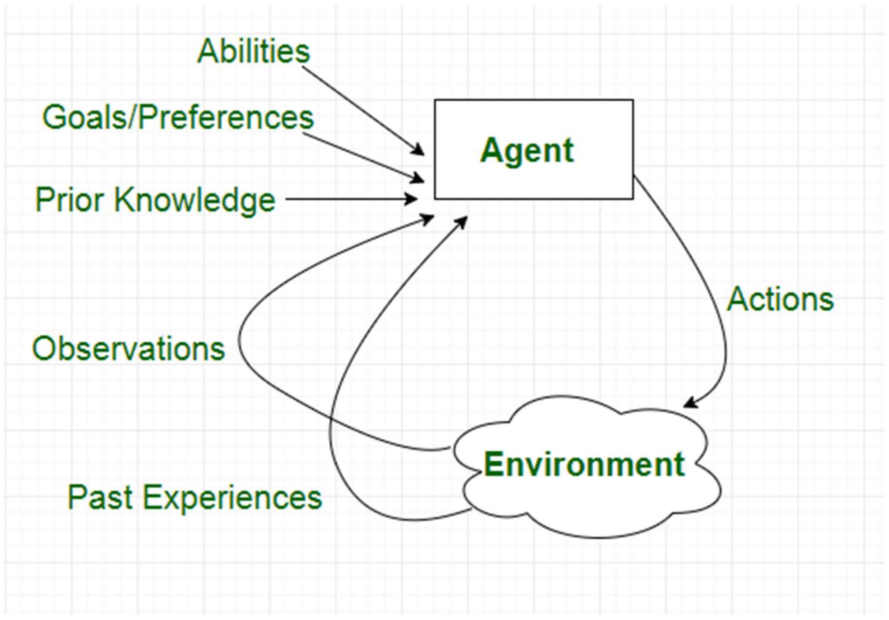
Agent = Architecture + Agent Program

There are many examples of agents in artificial intelligence. Here are a few:

- **Intelligent personal assistants:** These are agents that are designed to help users with various tasks, such as scheduling appointments, sending messages, and setting reminders. Examples of intelligent personal assistants include Siri, Alexa, and Google Assistant.
- **Autonomous robots:** These are agents that are designed to operate autonomously in the physical world. They can perform tasks such as cleaning, sorting, and delivering goods. Examples of autonomous robots include the Roomba vacuum cleaner and the Amazon delivery robot.
- **Gaming agents:** These are agents that are designed to play games, either against human opponents or other agents. Examples of gaming agents include chess-playing agents and poker-playing agents.
- **Fraud detection agents:** These are agents that are designed to detect fraudulent behavior in financial transactions. They can analyze patterns of behavior to identify suspicious activity and alert authorities. Examples of

fraud detection agents include those used by banks and credit card companies.

- **Traffic management agents:** These are agents that are designed to manage traffic flow in cities. They can monitor traffic patterns, adjust traffic lights, and reroute vehicles to minimize congestion. Examples of traffic management agents include those used in smart cities around the world.
- A **software agent** has Keystrokes, file contents, received network packages that act as sensors and displays on the screen, files, and sent network packets acting as actuators.
- A Human-agent has eyes, ears, and other organs which act as sensors, and hands, legs, mouth, and other body parts act as actuators.
- A **Robotic agent** has Cameras and infrared range finders which act as sensors and various motors act as actuators.



Characteristics of an Agent

Types of Agents

Agents can be grouped into five classes based on their degree of perceived intelligence and capability :

- Simple Reflex Agents
- Model-Based Reflex Agents

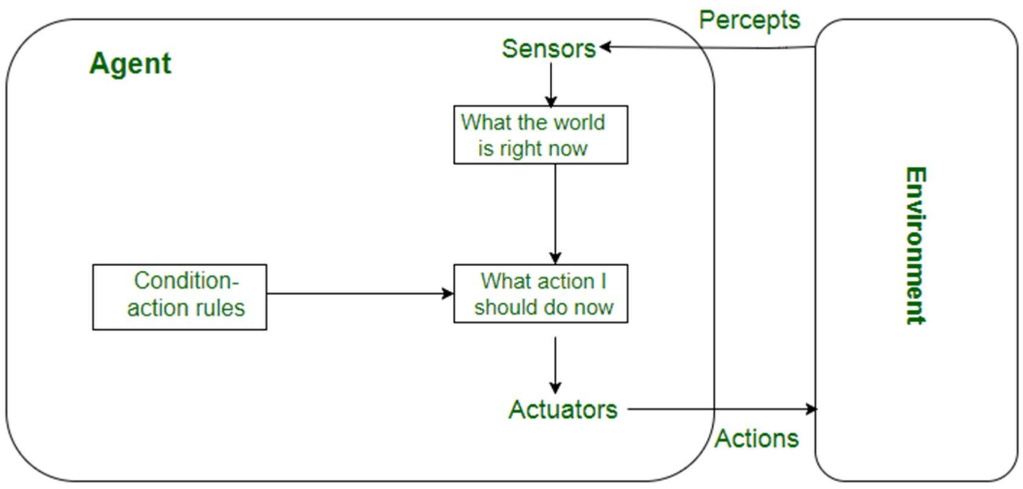
- Goal-Based Agents
- Utility-Based Agents
- Learning Agent
- Multi-agent systems
- Hierarchical agents

Simple Reflex Agents

Simple reflex agents ignore the rest of the percept history and act only on the basis of the **current percept**. Percept history is the history of all that an agent has perceived to date. The agent function is based on the **condition-action rule**. A condition-action rule is a rule that maps a state i.e., a condition to an action. If the condition is true, then the action is taken, else not. This agent function only succeeds when the environment is fully observable. For simple reflex agents operating in partially observable environments, infinite loops are often unavoidable. It may be possible to escape from infinite loops if the agent can randomize its actions.

Problems with Simple reflex agents are :

- Very limited intelligence.
- No knowledge of non-perceptual parts of the state.
- Usually too big to generate and store.
- If there occurs any change in the environment, then the collection of rules needs to be updated.

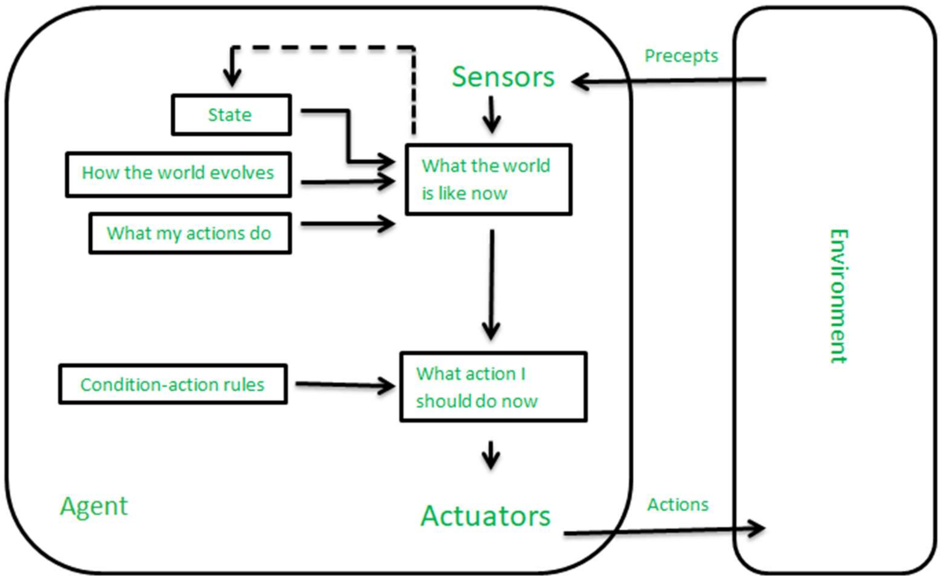


Model-Based Reflex Agents

It works by finding a rule whose condition matches the current situation. A model-based agent can handle **partially observable environments** by the use of a model about the world. The agent has to keep track of the **internal state** which is adjusted by each percept and that depends on the percept history. The current state is stored inside the agent which maintains some kind of structure describing the part of the world which cannot be seen.

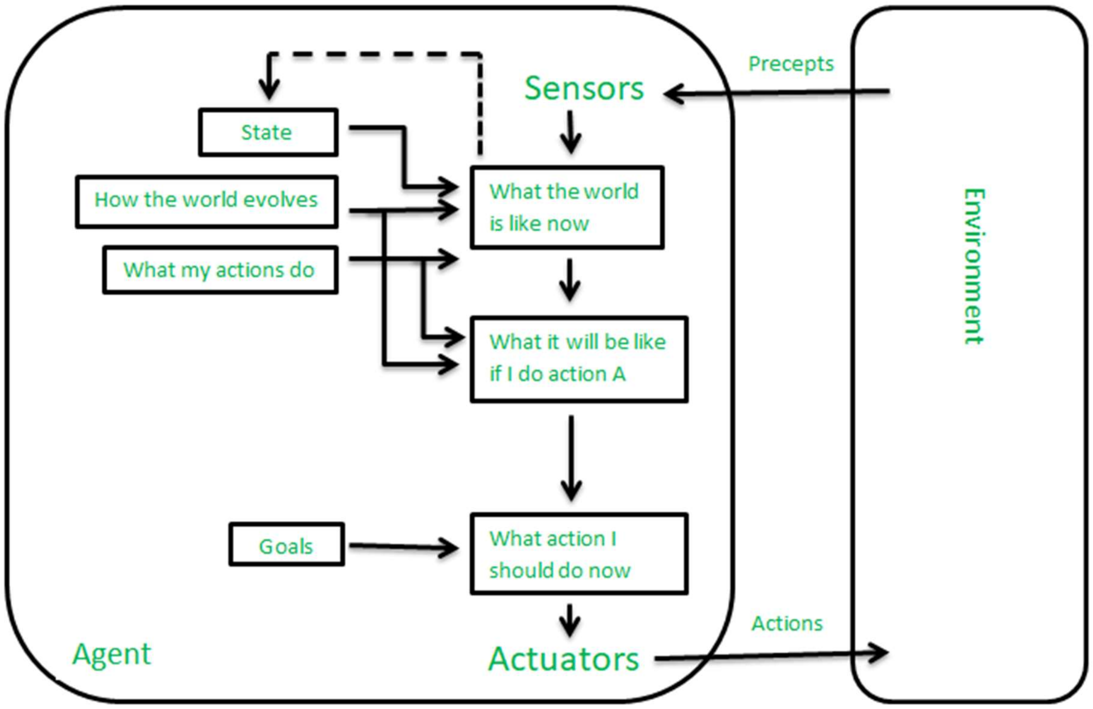
Updating the state requires information about:

- How the world evolves independently from the agent?
- How do the agent’s actions affect the world?



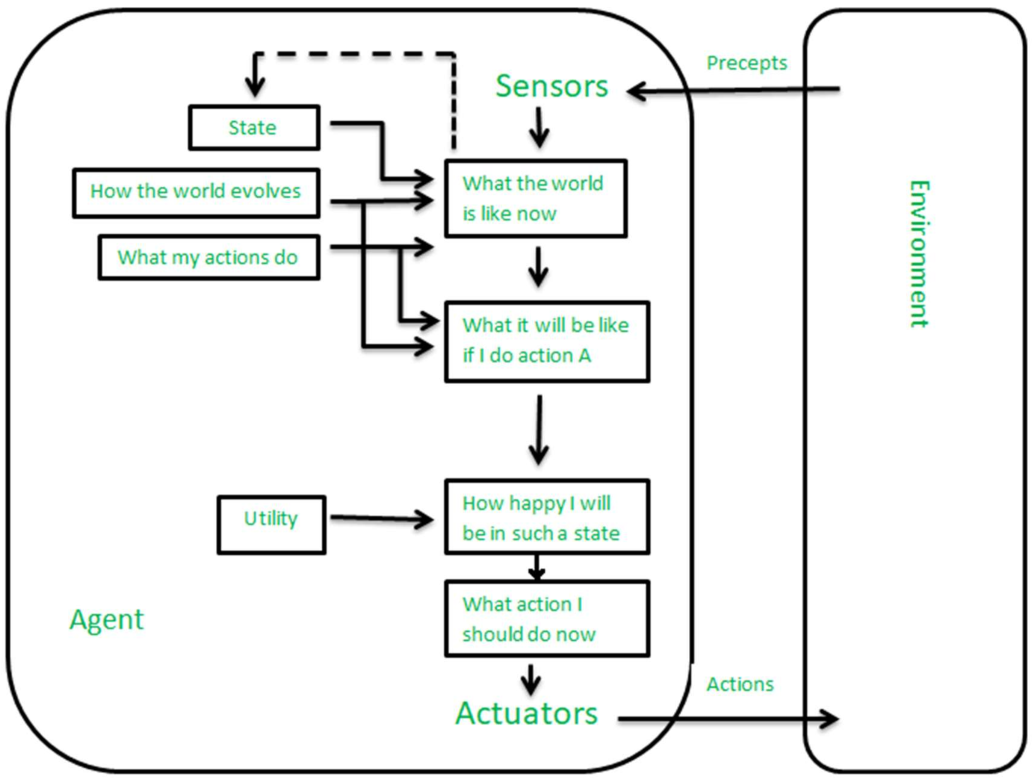
Goal-Based Agents

These kinds of agents take decisions based on how far they are currently from their **goal**(description of desirable situations). Their every action is intended to reduce their distance from the goal. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state. The knowledge that supports its decisions is represented explicitly and can be modified, which makes these agents more flexible. They usually require search and planning. The goal-based agent’s behavior can easily be changed.



Utility-Based Agents

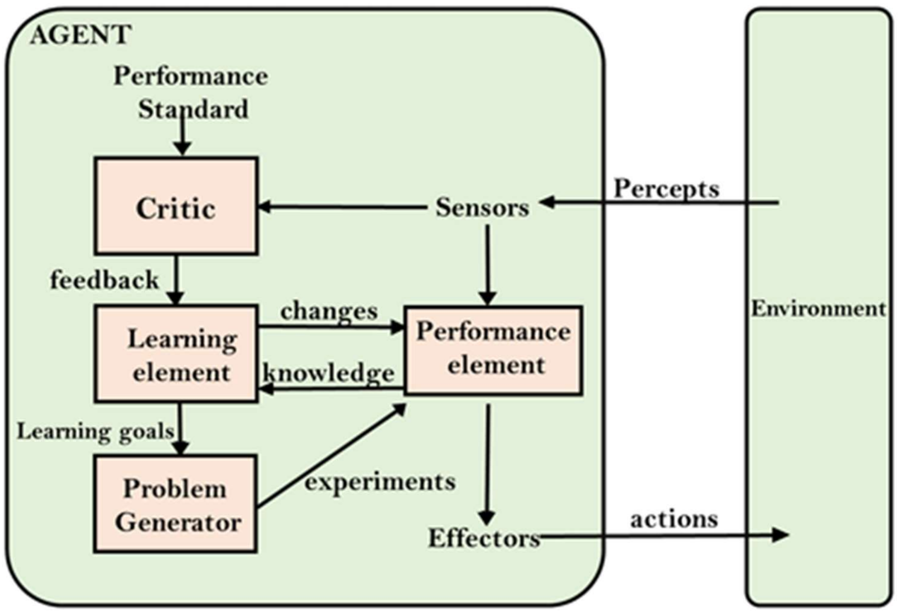
The agents which are developed having their end uses as building blocks are called utility-based agents. When there are multiple possible alternatives, then to decide which one is best, utility-based agents are used. They choose actions based on a **preference (utility)** for each state. Sometimes achieving the desired goal is not enough. We may look for a quicker, safer, cheaper trip to reach a destination. Agent happiness should be taken into consideration. Utility describes how **"happy"** the agent is. Because of the uncertainty in the world, a utility agent chooses the action that maximizes the expected utility. A utility function maps a state onto a real number which describes the associated degree of happiness.


Precepts
Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Actions
Environment

Learning Agent

A learning agent in AI is the type of agent that can learn from its past experiences or it has learning capabilities. It starts to act with basic knowledge and then is able to act and adapt automatically through learning. A learning agent has mainly four conceptual components, which are:

1. **Learning element:** It is responsible for making improvements by learning from the environment.
2. **Critic:** The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standard.
3. **Performance element:** It is responsible for selecting external action.
4. **Problem Generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.



Learning Agent

Multi-Agent Systems

These agents interact with other agents to achieve a common goal. They may have to coordinate their actions and communicate with each other to achieve their objective.

A multi-agent system (MAS) is a system composed of multiple interacting agents that are designed to work together to achieve a common goal. These agents may be autonomous or semi-autonomous and are capable of perceiving their environment, making decisions, and taking action to achieve the common objective.

MAS can be used in a variety of applications, including transportation systems, robotics, and social networks. They can help improve efficiency, reduce costs, and increase flexibility in complex systems. MAS can be classified into different types based on their characteristics, such as whether the agents have the same or different goals, whether the agents are cooperative or competitive, and whether the agents are homogeneous or heterogeneous.

- In a homogeneous MAS, all the agents have the same capabilities, goals, and behaviors.
- In contrast, in a heterogeneous MAS, the agents have different capabilities, goals, and behaviors.

This can make coordination more challenging but can also lead to more flexible and robust systems.

Cooperative MAS involves agents working together to achieve a common goal, while competitive MAS involves agents working against each other to achieve their own goals. In some cases, MAS can also involve both cooperative and competitive behavior, where agents must balance their own interests with the interests of the group.

MAS can be implemented using different techniques, such as game theory, machine learning, and agent-based modeling. Game theory is used to analyze strategic interactions between agents and predict their behavior. Machine learning is used to train agents to improve their decision-making capabilities over time. Agent-based modeling is used to simulate complex systems and study the interactions between agents.

Overall, multi-agent systems are a powerful tool in artificial intelligence that can help solve complex problems and improve efficiency in a variety of applications.

Hierarchical Agents

These agents are organized into a hierarchy, with high-level agents overseeing the behavior of lower-level agents. The high-level agents provide goals and constraints, while the low-level agents carry out specific tasks. Hierarchical agents are useful in complex environments with many tasks and sub-tasks.

- Hierarchical agents are agents that are organized into a hierarchy, with high-level agents overseeing the behavior of lower-level agents. The high-level agents provide goals and constraints, while the low-level agents carry out specific tasks. This structure allows for more efficient and organized decision-making in complex environments.

- Hierarchical agents can be implemented in a variety of applications, including robotics, manufacturing, and transportation systems. They are particularly useful in environments where there are many tasks and sub-tasks that need to be coordinated and prioritized.

- In a hierarchical agent system, the high-level agents are responsible for setting goals and constraints for the lower-level agents. These goals and constraints are typically based on the overall objective of the system. For example, in a manufacturing system, the high-level agents might set production targets for the lower-level agents based on customer demand.

- The low-level agents are responsible for carrying out specific tasks to achieve the goals set by the high-level agents. These tasks may be relatively simple or more complex, depending on the specific application. For example, in a transportation system, low-level agents might be responsible for managing traffic flow at specific intersections.

- Hierarchical agents can be organized into different levels, depending on the complexity of the system. In a simple system, there may be only two levels: high-level agents and low-level agents. In a more complex system, there may be multiple levels, with intermediate-level agents responsible for coordinating the activities of lower-level agents.
- One advantage of hierarchical agents is that they allow for more efficient use of resources. By organizing agents into a hierarchy, it is possible to allocate tasks to the agents that are best suited to carry them out, while avoiding duplication of effort. This can lead to faster, more efficient decision-making and better overall performance of the system.

Overall, hierarchical agents are a powerful tool in artificial intelligence that can help solve complex problems and improve efficiency in a variety of applications.

Uses of Agents

Agents are used in a wide range of applications in artificial intelligence, including:

- **Robotics:** Agents can be used to control robots and automate tasks in manufacturing, transportation, and other industries.
- **Smart homes and buildings:** Agents can be used to control heating, lighting, and other systems in smart homes and buildings, optimizing energy use and improving comfort.
- **Transportation systems:** Agents can be used to manage traffic flow, optimize routes for autonomous vehicles, and improve logistics and supply chain management.
- **Healthcare:** Agents can be used to monitor patients, provide personalized treatment plans, and optimize healthcare resource allocation.
- **Finance:** Agents can be used for automated trading, fraud detection, and risk management in the financial industry.
- **Games:** Agents can be used to create intelligent opponents in games and simulations, providing a more challenging and realistic experience for players.

- **Natural language processing:** Agents can be used for language translation, question answering, and chatbots that can communicate with users in natural language.
- **Cybersecurity:** Agents can be used for intrusion detection, malware analysis, and network security.
- **Environmental monitoring:** Agents can be used to monitor and manage natural resources, track climate change, and improve environmental sustainability.
- **Social media:** Agents can be used to analyze social media data, identify trends and patterns, and provide personalized recommendations to users.

Inductive Learning in ML

Machine learning, a subset of artificial intelligence, has revolutionized various domains by enabling systems to learn from data and make intelligent decisions. One of the fundamental approaches in machine learning is inductive learning, which allows machines to generalize patterns from a given set of examples.

Inductive learning is a powerful technique that enables machines to learn from experience, adapt to new situations, and make accurate predictions.

A common method in machine learning is known as inductive learning, which trains a model to generalise from specific examples to make predictions on brand-new, unexplored data. We will examine inductive learning in more detail in this post, along with some of its uses in machine learning.

What does inductive learning mean?

A kind of artificial learning called inductive learning allows a model to generalise from a set of precise instances to make forecasts on brand-new, unexplored data. Inductive learning aims to develop a function that converts inputs to outputs so that it may be applied to new data to predict future outcomes.

In supervised learning, when the model is trained using labelled data, inductive learning is frequently utilised. Each training sample in supervised learning is connected to a predetermined label or output, and the system learns can predict the output according to its input attributes.

A technique to machine learning known as inductive learning uses a model to learn from a collection of training examples to generate predictions on brand-new, unexplored data. Deductive learning, which includes drawing logical conclusions from the set of premises, is sometimes compared with inductive learning.

The model can learn by inductive learning by extrapolating from particular examples. This indicates that the model makes an effort to find links or trends in the information that may be applied to fresh, untainted data. An inductive learning system, for instance, may be taught to distinguish between photographs of cats and dogs based on characteristics like fur colour, ear form, and tail length.

When a model is trained using labelled data, which means so each video sequence is connected to a specific output or label, inductive learning is frequently utilised. In order to generate predictions on brand-new, unforeseen data, the model must first learn a functional that converts inputs to outputs.

Using a machine learning model to learn a theory, like a tree structure, neural network, or a support vector machine, is one popular method of inductive learning. A loss function that calculates the variance between the anticipated and actual results will normally be minimised by the algorithm using an optimization method, such as gradient descent with stochastic gradient descent.

Inductive learning is an effective methodology that has been used to create a wide variety of models based on machine learning for tasks like fraud detection, natural language, and picture identification.

What is the process of inductive learning?

A technique to machine learning called inductive learning teacher and students from examples to create predictions on brand-new, untainted data. This is how it usually goes:

Data gathering: Gathering data is the initial stage in inductive learning. Usually, the data is labelled, which means that each instance is connected to a certain output or label.

The following step is to choose a model for machine learning that is appropriate for the given task. Models of many kinds, including choice trees, artificial neural, and support vector machines, can be applied to inductive learning.

Training: Using an optimization method like gradient descent and stochastic gradient descent, the algorithm is taught on the labelled data. A loss function that calculates the discrepancy between the predicted and actual outputs is minimised using the optimization technique.

A different data set that wasn't utilised for training is used to evaluate the model after it has been trained. The degree to which the model generalises to fresh, untested data is measured in the evaluation.

Lastly, predictions on fresh, unobserved data are made using the trained model.

Inductive learning's central tenet is the ability to generalise from particular examples to brand-new, unexplored material. This entails finding correlations and trends in the information that can be applied to forecast the outcome of fresh data. To put it another way, the model seeks to understand the fundamental structure of the information to generate precise predictions about brand-new, untainted data.

Overfitting, which happens when the system is too complicated and fits the data for training too closely, is one of the fundamental problems with inductive learning. Poor generalisation to fresh, unforeseen facts may result from this. This has led to the development of a few approaches, including regularisation, cross-validation, and premature stopping, which serve to reduce overfitting and enhance the generalisation capabilities of the model.

A variety of deep learning models have been developed using a variety of inductive learning techniques for a variety of applications.

Applications of inductive learning:

An approach to machine learning known as inductive learning uses learning from examples to predict outcomes for brand-new datasets. This method has been applied to a variety of machine learning applications, including image identification and natural language processing. We will examine a few inductive learning applications in more detail in this article.

Image Identification

Image recognition is one of the most widely used inductive learning applications. Models that can identify items in photographs, such as automobiles, people, and animals, can be trained via inductive learning. The models employ the learnt patterns and correlations to generate predictions on new, unlabeled images after being trained on a sizable dataset of labelled images.

Automatic Language Recognition

Natural language processing (NLP), which includes teaching machines to comprehend and produce human language, such as text and speech, also employs inductive learning. The subject of a document can be ascertained, the emotion of a line can be identified, or even writing can be generated in response to a prompt using inductive learning.

Detecting fraud

Models that can identify shady dealings, such credit card fraud, can be trained via inductive learning. The models employ the learned correlations and patterns to detect fraudulent activity in real-time after being trained on a dataset of labelled transactions.

Systems of Recommendations

Moreover, models that can provide individualised suggestions, such as recommending books or products to clients, can be trained via inductive learning. The models employ the learnt patterns and correlations to recommend new items that are probably going to appeal to the user after being trained on such a dataset of customer preferences and behaviours.

Language Recognition

Another usage of inductive learning is in speech recognition, in which the systems are trained on a huge dataset of labelled speech samples and then apply the relationships and patterns discovered to transcribe fresh, unheard speech. Applications for speech recognition span from virtual personal assistants to voice-activated gadgets.

Medical Conclusion

The algorithms are taught using a dataset of labelled medical records, and they may also be used to identify new patients by applying the patterns and links they have discovered. When there is a lot of data accessible, like when analysing medical imaging data, this can be quite helpful.

Conclusion:

Machine learning's effective inductive learning method has been applied to a variety of tasks, including image identification and medical diagnosis. Inductive learning enables algorithms to generate predictions on fresh, unforeseen data by learning from particular examples. This makes it a useful tool in many machine learning domains, such as speech recognition, fraud detection, recommender systems, natural language processing, and medical diagnosis.

Types of Machine Learning

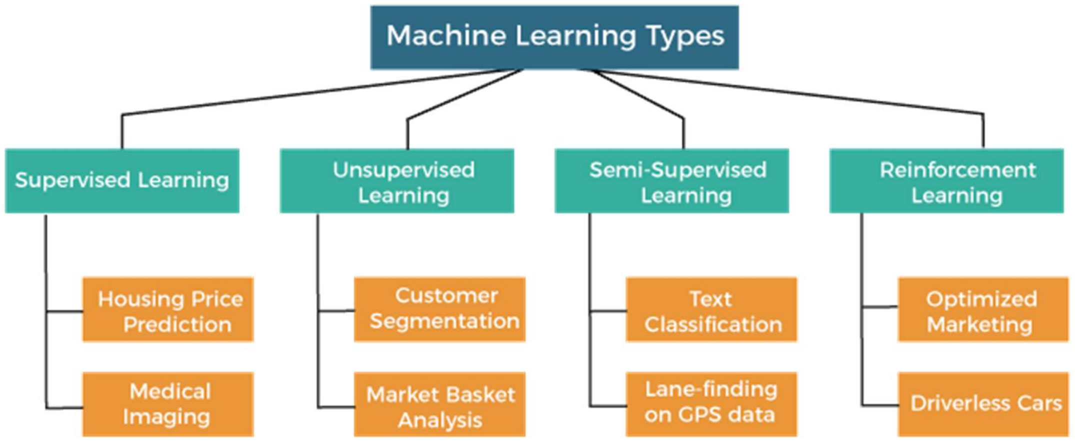
Machine learning is a subset of AI, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions.

Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and based on training, they build the model & perform a specific task.

These ML algorithms help to solve different business problems like Regression, Classification, Forecasting, Clustering, and Associations, etc.

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning



1. Supervised Machine Learning

As its name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are smaller), etc. After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning.

The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are **Risk Assessment, Fraud Detection, Spam filtering,** etc.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

- **Classification**
- **Regression**

a) Classification

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "**Yes" or No, Male or Female, Red or Blue, etc**. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.**

Some popular classification algorithms are given below:

- **Random Forest Algorithm**
- **Decision Tree Algorithm**
- **Logistic Regression Algorithm**
- **Support Vector Machine Algorithm**

b) Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

Some popular Regression algorithms are given below:

- **Simple Linear Regression Algorithm**
- **Multivariate Regression Algorithm**
- **Decision Tree Algorithm**
- **Lasso Regression**

Advantages and Disadvantages of Supervised Learning

Advantages:

- Since supervised learning work with the labelled dataset so we can have an exact idea about the classes of objects.
- These algorithms are helpful in predicting the output on the basis of prior experience.

Disadvantages:

- These algorithms are not able to solve complex tasks.
- It may predict the wrong output if the test data is different from the training data.
- It requires lots of computational time to train the algorithm.

Applications of Supervised Learning

Some common applications of Supervised Learning are given below:

- **ImageSegmentation:**
  Supervised Learning algorithms are used in image segmentation. In this process, image classification is performed on different image data with pre-defined labels.
- **MedicalDiagnosis:**
  Supervised algorithms are also used in the medical field for diagnosis purposes. It is done by using medical images and past labelled data with labels for disease conditions. With such a process, the machine can identify a disease for the new patients.
- **Fraud Detection -** Supervised Learning classification algorithms are used for identifying fraud transactions, fraud customers, etc. It is done by using historic data to identify the patterns that can lead to possible fraud.
- **Spam detection -** In spam detection & filtering, classification algorithms are used. These algorithms classify an email as spam or not spam. The spam emails are sent to the spam folder.
- **Speech Recognition -** Supervised learning algorithms are also used in speech recognition. The algorithm is trained with voice data, and various identifications can be done using the same, such as voice-activated passwords, voice commands, etc.

2. Unsupervised Machine Learning

Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output without any supervision.

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences. Machines are instructed to find the hidden patterns from the input dataset.

Let's take an example to understand it more preciously; suppose there is a basket of fruit images, and we input it into the machine learning model. The images are totally unknown to the model, and the task of the machine is to find the patterns and categories of the objects.

So, now the machine will discover its patterns and differences, such as colour difference, shape difference, and predict the output when it is tested with the test dataset.

Categories of Unsupervised Machine Learning

Unsupervised Learning can be further classified into two types, which are given below:

- **Clustering**
- **Association**

1) Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is grouping the customers by their purchasing behaviour.

Some of the popular clustering algorithms are given below:

- **K-Means Clustering algorithm**
- **Mean-shift algorithm**
- **DBSCAN Algorithm**

- **Principal Component Analysis**
- **Independent Component Analysis**

2) Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in **Market Basket analysis, Web usage mining, continuous production**, etc.

Some popular algorithms of Association rule learning are **Apriori Algorithm, Eclat, FP-growth algorithm.**

Advantages and Disadvantages of Unsupervised Learning Algorithm

Advantages:

- These algorithms can be used for complicated tasks compared to the supervised ones because these algorithms work on the unlabeled dataset.
- Unsupervised algorithms are preferable for various tasks as getting the unlabeled dataset is easier as compared to the labelled dataset.

Disadvantages:

- The output of an unsupervised algorithm can be less accurate as the dataset is not labelled, and algorithms are not trained with the exact output in prior.
- Working with Unsupervised learning is more difficult as it works with the unlabelled dataset that does not map with the output.

Applications of Unsupervised Learning

- **Network Analysis:** Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.
- **Recommendation Systems:** Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.
- **Anomaly Detection:** Anomaly detection is a popular application of unsupervised learning, which can identify unusual data points within the dataset. It is used to discover fraudulent transactions.

- **Singular Value Decomposition:** Singular Value Decomposition or SVD is used to extract particular information from the database. For example, extracting information of each user located at a particular location.

3. Semi-Supervised Learning

Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period.

Although Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabeled data. As labels are costly, but for corporate purposes, they may have few labels. It is completely different from supervised and unsupervised learning as they are based on the presence & absence of labels.

To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced. The main aim of semi-supervised learning is to effectively use all the available data, rather than only labelled data like in supervised learning. Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabeled data into labelled data. It is because labelled data is a comparatively more expensive acquisition than unlabeled data.

We can imagine these algorithms with an example. Supervised learning is where a student is under the supervision of an instructor at home and college. Further, if that student is self-analysing the same concept without any help from the instructor, it comes under unsupervised learning. Under semi-supervised learning, the student must revise himself after analyzing the same concept under the guidance of an instructor at college.

Advantages and disadvantages of Semi-supervised Learning

Advantages:

- It is simple and easy to understand the algorithm.
- It is highly efficient.
- It is used to solve drawbacks of Supervised and Unsupervised Learning algorithms.

Disadvantages:

- Iterations results may not be stable.
- We cannot apply these algorithms to network-level data.
- Accuracy is low.

4. Reinforcement Learning

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance.

Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.

The reinforcement learning process is like a human being; for example, a child learns various things by experiences in his day-to-day life. An example of reinforcement learning is to play a game, where the Game is the environment, moves of an agent at each step define states, and the goal of the agent is to get a high score. Agent receives feedback in terms of punishment and rewards.

Due to its way of working, reinforcement learning is employed in different fields such as **Game theory, Operation Research, Information theory, multi-agent systems.**

A reinforcement learning problem can be formalized using **Markov Decision Process(MDP).** In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.

Categories of Reinforcement Learning

Reinforcement learning is categorized mainly into two types of methods/algorithms:

- **Positive Reinforcement Learning:** Positive reinforcement learning specifies increasing the tendency that the required behaviour would occur again by adding something. It enhances the strength of the behaviour of the agent and positively impacts it.

- **Negative Reinforcement Learning:** Negative reinforcement learning works exactly opposite to the positive RL. It increases the tendency that the specific behaviour would occur again by avoiding the negative condition.

Real-world Use cases of Reinforcement Learning

- **Video Games:**
  RL algorithms are much popular in gaming applications. It is used to gain super-human performance. Some popular games that use RL algorithms are **AlphaGO** and **AlphaGO Zero**.
- **Resource Management:**
  The "Resource Management with Deep Reinforcement Learning" paper showed that how to use RL in computer to automatically learn and schedule resources to wait for different jobs in order to minimize average job slowdown.
- **Robotics:**
  RL is widely being used in Robotics applications. Robots are used in the industrial and manufacturing area, and these robots are made more powerful with reinforcement learning. There are different industries that have their vision of building intelligent robots using AI and Machine learning technology.
- **Text Mining**
  Text-mining, one of the great applications of NLP, is now being implemented with the help of Reinforcement Learning by Salesforce company.

Advantages and Disadvantages of Reinforcement Learning

Advantages

- It helps in solving complex real-world problems which are difficult to be solved by general techniques.
- The learning model of RL is similar to the learning of human beings; hence most accurate results can be found.
- Helps in achieving long term results.

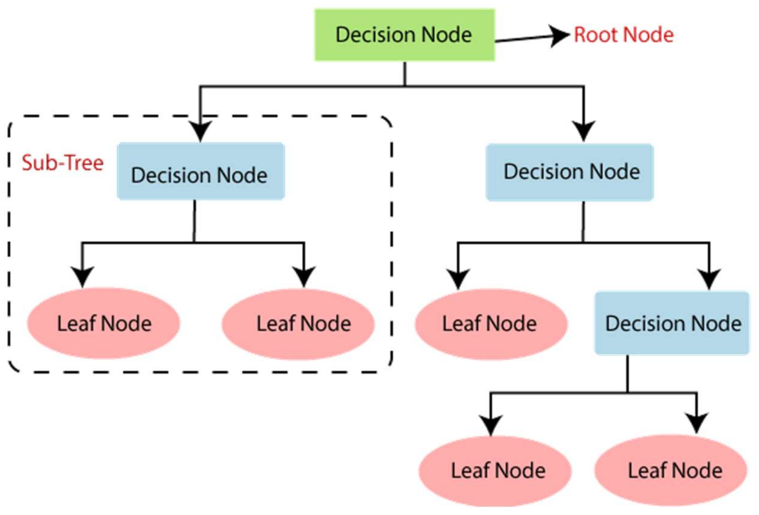
Disadvantage

- RL algorithms are not preferred for simple problems.
- RL algorithms require huge data and computations.

- Too much reinforcement learning can lead to an overload of states which can weaken the results.

What is Learning decision trees in machine learning?

- 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 based on 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.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:



Why use Decision Trees?

- There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:
- 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 Terminologies

- **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.

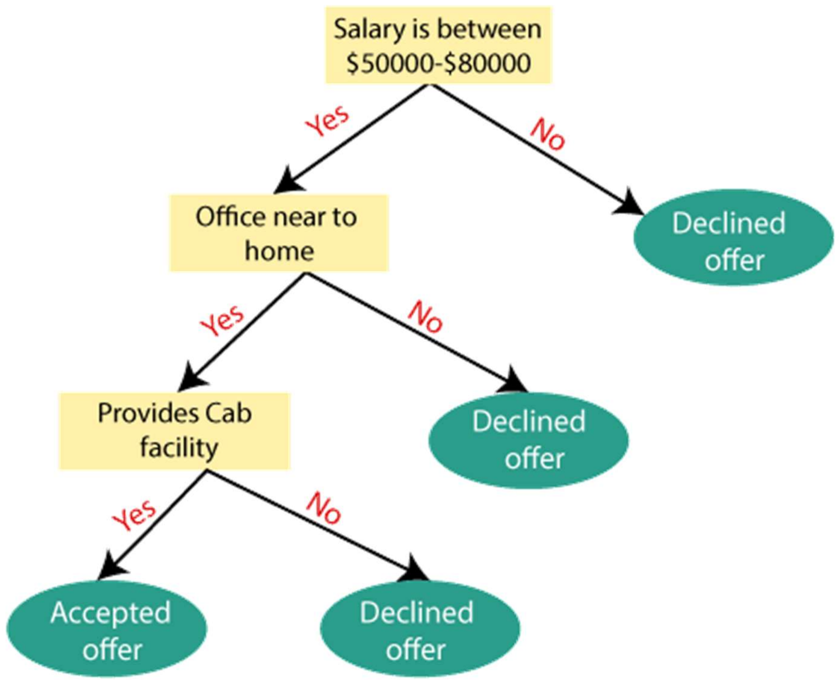
How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

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.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. 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:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

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**

2. 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.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

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

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

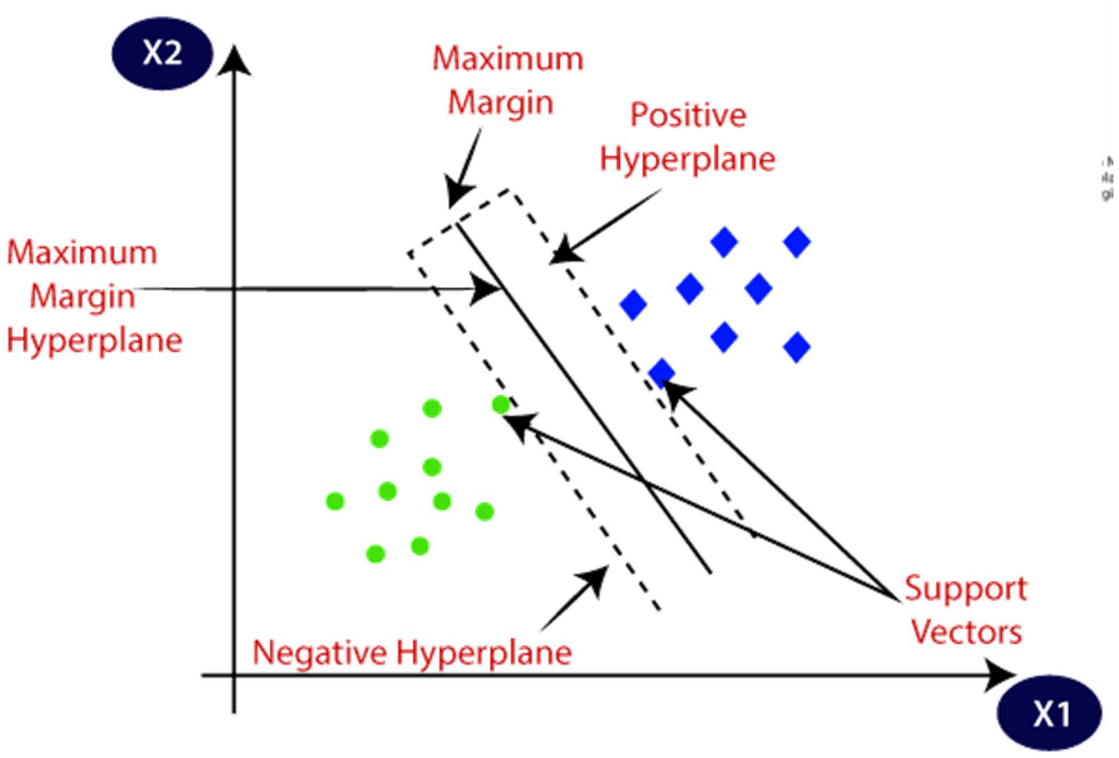
- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

Support Vector Machine (SVM)

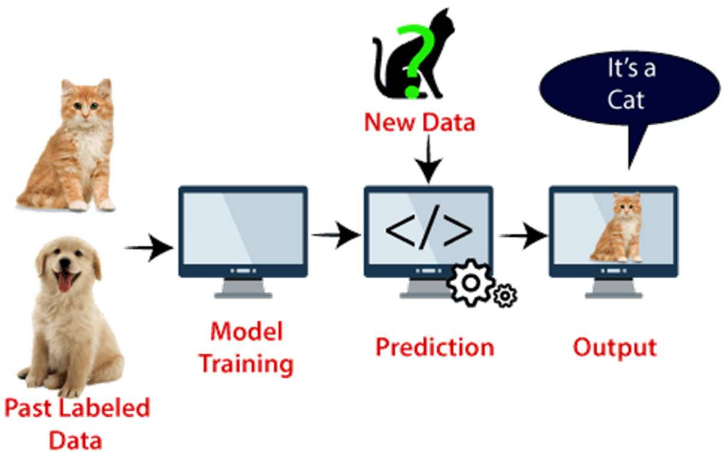
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

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:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. Based on the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

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.

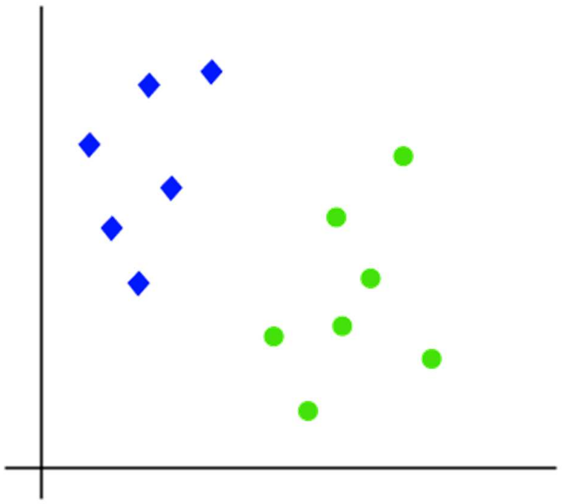
Support Vectors:

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.

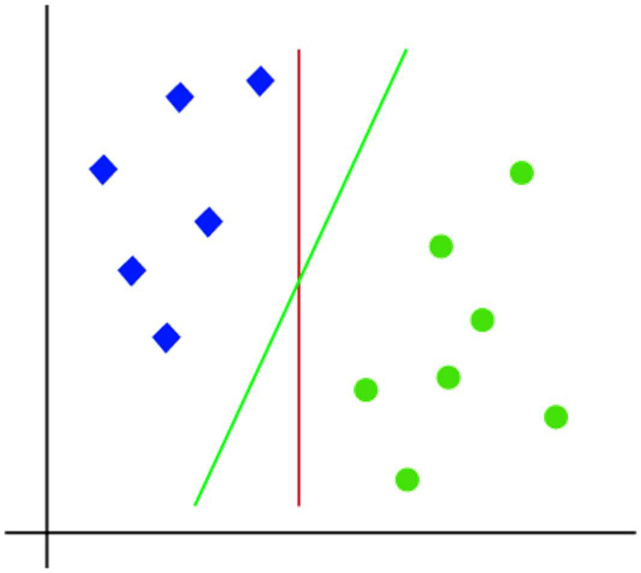
How does SVM works?

Linear SVM:

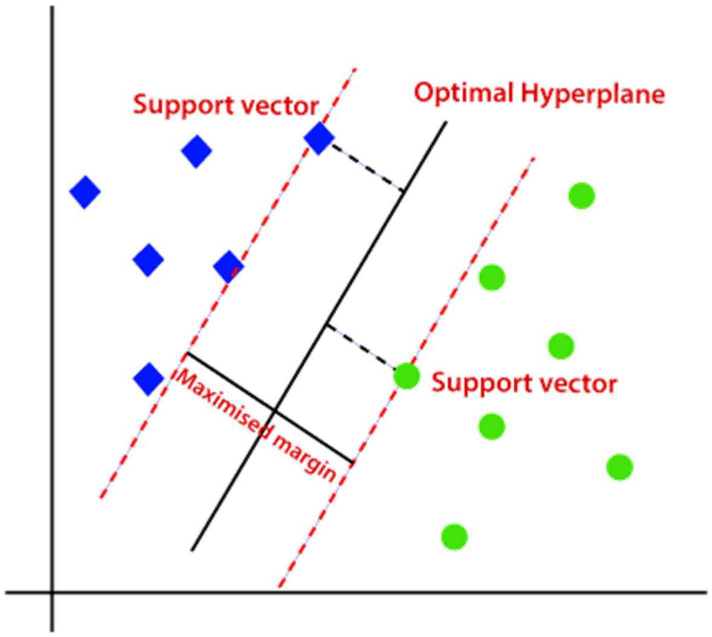
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



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:

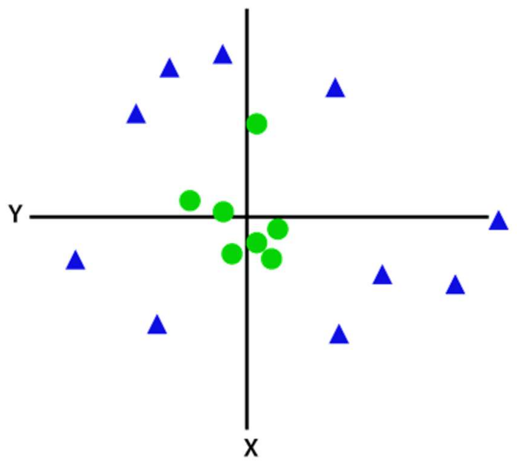


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane 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:

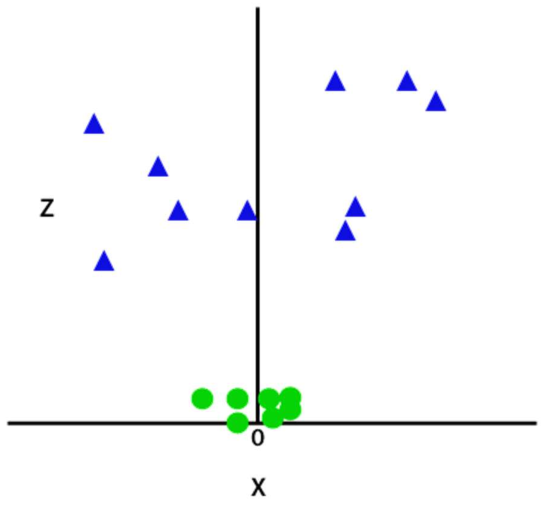
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



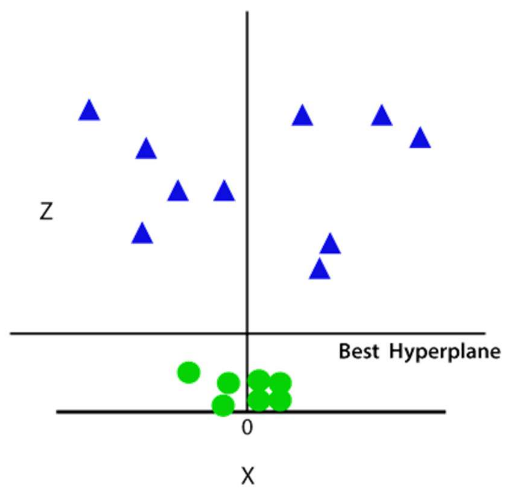
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 +y^2$

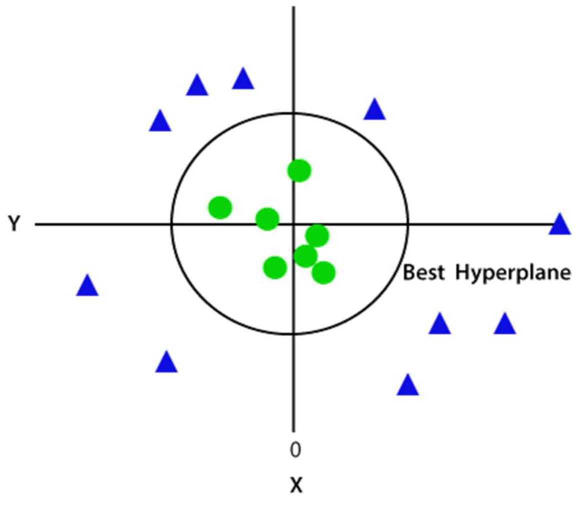
By adding the third dimension, the sample space will become as below image:



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



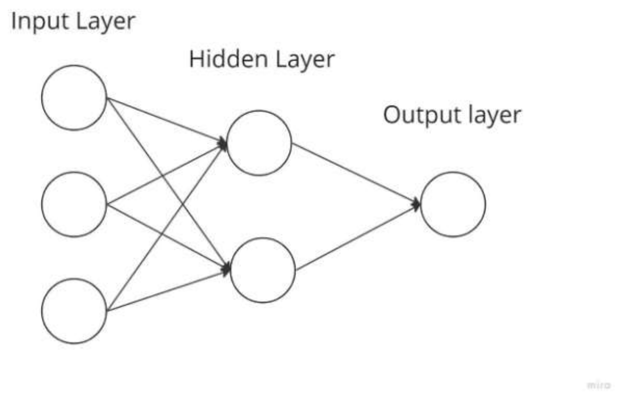
Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



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

What is Feed Forward Neural Network?

The most fundamental kind of neural network, in which input data travels only in one way before leaving through output nodes and passing through artificial neural nodes. Input and output layers are present in locations where hidden layers may or may not be present. Based on this, they are further divided into single-layered and multi-layered feed-forward neural network

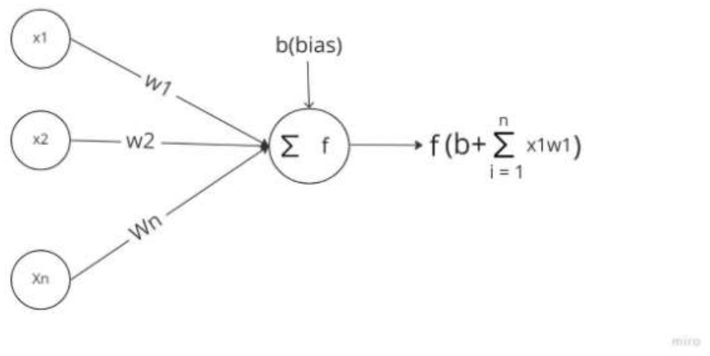


The complexity of the function is inversely correlated with the number of layers. It cannot spread backward; it can only go forward. In this scenario, the weights are unchanged. Weights are added to the inputs before being passed to an activation function.

Feed Forward Neural Network components

Neurons

The fundamental component of a neural network is an artificial neuron. The following is a schematic illustration of a neuron.



It operates in two parts, as can be seen above: first, it computes the weighted sum of its inputs, and then it uses an activation function to normalize the total. There are both linear and nonlinear activation functions. Additionally, each input to a neuron has a corresponding weight. The network must learn these parameters during the training phase.

Activation Function

A neuron's activation function serves as a decision-making body at the output. The activation function determines whether the neuron learns linear or non-linear decision limits. Additionally, it has a leveling impact on neuron output, preventing the cascade effect from making neurons' output after multiple layers become exceedingly enormous. The three most popular activation mechanisms are as follows −

- Sigmoid − Mapped to the output values are input values between 0 and 1.
- Tanh − The input values are mapped to a value in the range of -1 and 1.
- ReLu − This function only allows positive numbers to pass through it. Inverse values are assigned to 0.

Input layer

This layer's neurons take in information and send it to the network's other levels. The number of neurons in the input layer must equal the number of features or attributes in the dataset.

Output Layer

This layer is the one that provides the predictions. For various issues, a distinct activation function should be used in this layer. We want the result to be either 0 or 1 for a binary classification task. As a result, the sigmoid activation function is employed. A Softmax (think of it as a sigmoid applied to several classes) is used for multiclass classification problems. We can utilize a linear unit to solve regression problems where the result does not fall into a predetermined category.

Hidden layer

Layers concealed between the input and output are used to segregate them. The number of hidden layers will depend on the type of model. In order to actually transfer the input to the next layer, numerous neurons in hidden layers first modify it. To improving predictability, this network's weights are adjusted continuously.

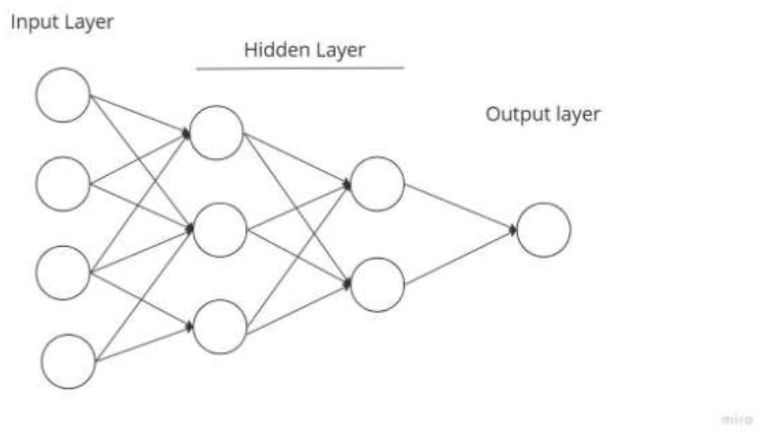
Working principle of Feed Forward Neural Network

A single-layer perceptron could represent how the feed-forward neural network looks when it is simplified. As inputs enter the layer, this model multiplies them with weights. The total is then obtained by adding the weighted input values collectively. In general, the output value is 1, and if the total of the values falls below the threshold, it is typically -1, as long as the sum is above a predetermined threshold, which is set at zero.

The single-layer perceptron is a feed-forward neural network model that is frequently used for classification. Single-layer perceptrons can also benefit from machine learning. Neural networks can modify their weights during training based on a characteristic known as the delta rule, which allows them to compare their outputs to the expected values. Gradient descent is the consequence of training and learning. Multi-layered perceptrons update their weights in the same way. However, this is known as back-propagation. In this situation, the network's hidden layers will be changed based on the final layer's output values.

Multi-layer Feed Forward Neural Network

An entrance points into sophisticated neural networks, where incoming data is routed through several layers of artificial neurons. Every node is linked to every neuron in the following layer, resulting in a fully connected neural network. There are input and output layers, as well as several hidden levels, for a total of at least three or more layers. It possesses bidirectional propagation, which means it can propagate both forward and backward.



Inputs are multiplied by weights and supplied into the activation function, where they are adjusted to minimize loss during backpropagation. Weights are just machine-learned values from Neural Networks. They modify themselves based on the gap between projected and training outcomes. Softmax is used as an output layer activation function after nonlinear activation functions.

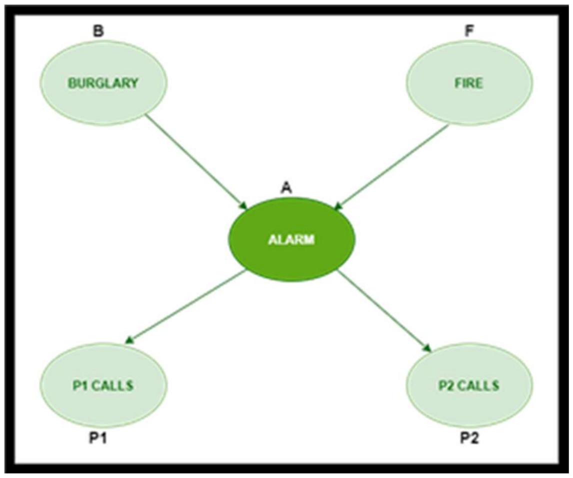
Conclusion

Finally, feedforward neural networks are sometimes referred to as Multi-layered Networks of Neurons (MLN). The information only flows forward in the neural network, first through the input nodes, then through the hidden layers (single or many layers), and ultimately through the output nodes, which is why this network of models is termed feedforward.

Bayesian Belief Networks

Bayesian Belief Network is a graphical representation of different probabilistic relationships among random variables in a particular set. It is a classifier with no dependency on attributes i.e it is condition independent. Due to its feature of joint probability, the probability in Bayesian Belief Network is derived, based on a condition — P(attribute/parent) i.e probability of an attribute, true over parent attribute.

- Consider this example:



- In the above figure, we have an alarm 'A' – a node, say installed in a house of a person 'gfg', which rings upon two probabilities i.e burglary

'B' and fire 'F', which are – parent nodes of the alarm node. The alarm is the parent node of two probabilities P1 calls 'P1' & P2 calls 'P2' person nodes.

- Upon the instance of burglary and fire, 'P1' and 'P2' call person 'gfg', respectively. But, there are few drawbacks in this case, as sometimes 'P1' may forget to call the person 'gfg', even after hearing the alarm, as he has a tendency to forget things, quick. Similarly, 'P2', sometimes fails to call the person 'gfg', as he is only able to hear the alarm, from a certain distance.

Q) Find the probability that 'P1' is true (P1 has called 'gfg'), 'P2' is true (P2 has called 'gfg') when the alarm 'A' rang, but no burglary 'B' and fire 'F' has occurred.

=> **P (P1, P2, A, ~B, ~F)** [where- P1, P2 & A are 'true' events and '~B' & '~F' are 'false' events]

Burglary 'B' –

- **P (B=T) = 0.001** ('B' is true i.e burglary has occurred)
- **P (B=F) = 0.999** ('B' is false i.e burglary has not occurred)

Fire 'F' –

- **P (F=T) = 0.002** ('F' is true i.e fire has occurred)
- **P (F=F) = 0.998** ('F' is false i.e fire has not occurred)

Alarm 'A' –

B	**F**	**P (A=T)**	**P (A=F)**
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71

F	F	0.001	**0.999**

- The alarm ‘A’ node can be ‘true’ or ‘false’ (i.e may have rung or may not have rung). It has two parent nodes burglary ‘B’ and fire ‘F’ which can be ‘true’ or ‘false’ (i.e may have occurred or may not have occurred) depending upon different conditions.

Person ‘P1’ –

A	**P (P1=T)**	**P (P1=F)**
T	**0.95**	0.05
F	0.05	0.95

- The person ‘P1’ node can be ‘true’ or ‘false’ (i.e may have called the person ‘gfg’ or not) . It has a parent node, the alarm ‘A’, which can be ‘true’ or ‘false’ (i.e may have rung or may not have rung ,upon burglary ‘B’ or fire ‘F’).

Person ‘P2’ –

A	**P (P2=T)**	**P (P2=F)**
T	**0.80**	0.20
F	0.01	0.99

- The person ‘P2’ node can be ‘true’ or false’ (i.e may have called the person ‘gfg’ or not). It has a parent node, the alarm ‘A’, which can be ‘true’ or ‘false’ (i.e may have rung or may not have rung, upon burglary ‘B’ or fire ‘F’).

Solution: Considering the observed probabilistic scan –

With respect to the question — **P (P1, P2, A, ~B, ~F)** , we need to get the probability of 'P1'. We find it with regard to its parent node – alarm 'A'. To get the probability of 'P2', we find it with regard to its parent node — alarm 'A'.

We find the probability of alarm 'A' node with regard to '~B' & '~F' since burglary 'B' and fire 'F' are parent nodes of alarm 'A'.

From the observed probabilistic scan, we can deduce –

P (P1, P2, A, ~B, ~F)

= P (P1/A) * P (P2/A) * P (A/~B~F) * P (~B) * P (~F)

= 0.95 * 0.80 * 0.001 * 0.999 * 0.998

= 0.00075
