Course Code: 20CS0906 **R20**

SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY:: PUTTUR
(AUTONOMOUS)
Siddharth Nagar, Narayanavanam Road – 517583

QUESTION BANK (DESCRIPTIVE)

Subject with Code: **Advanced Machine Learning(20CS0906)** Course & Branch: **B.Tech - CSM**

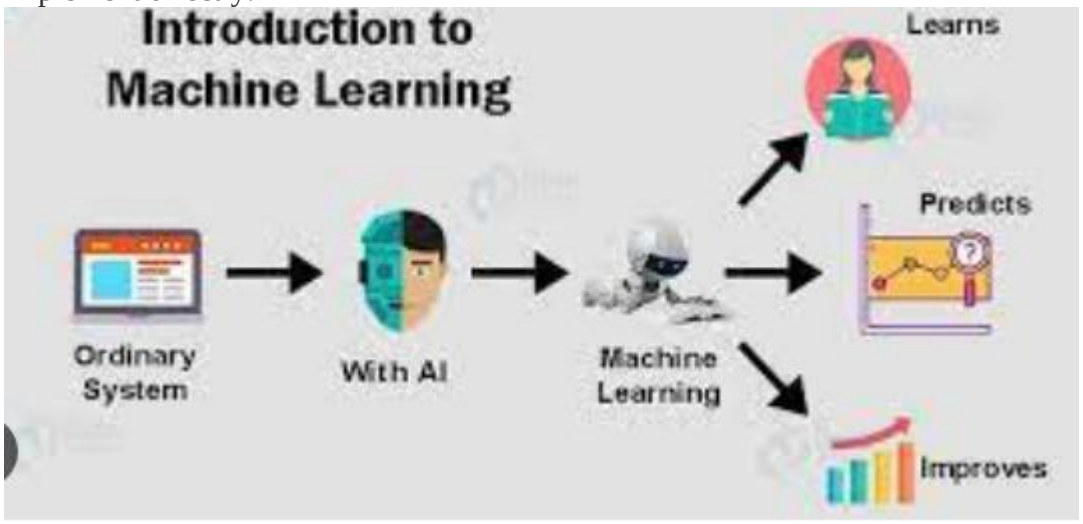
Regulation: **R20** Year &Sem: **III-B.Tech & II - Sem**

UNIT –I
INTRODUCTION

1	**a**	**What is Machine learning? Explain the need of it.**	**[L2][CO1]**	**[4M]**

Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel** in **1959. A machine has the ability to learn if it can improve its performance by gaining more data.**

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly.



Following are some key points which show the importance of Machine Learning:

- Rapid increment in the production of data
- Solving complex problems, which are difficult for a human
- Decision making in various sector including finance
- Finding hidden patterns and extracting useful information from data.

	b	**List out applications and some popular algorithms used in Machine Learning. Explain it?**	**[L2][CO1]**	**[8M]**

Applications of Machine Learning:

1. Image Recognition:

Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, **Automatic friend tagging suggestion**:

2. Speech Recognition

While using Google, we get an option of "**Search by voice**," it comes under speech recognition, and it's a popular application of machine learning.

Speech recognition is a process of converting voice instructions into text, and it is also known as "**Speech to text**", or "**Computer speech recognition**." At present, machine learning algorithms are widely used by various applications of speech recognition. **Google assistant**, **Siri**, **Cortana**, and **Alexa** are using speech recognition technology to follow the voice instructions.

3. Traffic prediction:

If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions.

It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:

- **Real Time location** of the vehicle form Google Map app and sensors
- **Average time has taken** on past days at the same time.

Everyone who is using Google Map is helping this app to make it better. It takes information from the user and sends back to its database to improve the performance.

4. Product recommendations:

Machine learning is widely used by various e-commerce and entertainment companies such as **Amazon**, **Netflix**, etc., for product recommendation to the user. Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.

Google understands the user interest using various machine learning algorithms and suggests the product as per customer interest.

As similar, when we use Netflix, we find some recommendations for entertainment series, movies, etc., and this is also done with the help of machine learning.

5. Self-driving cars:

One of the most exciting applications of machine learning is self-driving cars. Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company is working on self-driving car. It is using unsupervised learning method to train the car models to detect people and objects while driving.

6. Email Spam and Malware Filtering:

Whenever we receive a new email, it is filtered automatically as important, normal, and spam. We always receive an important mail in our inbox with the important symbol and spam emails in our spam box, and the technology behind this is Machine learning. Below are some spam filters used by Gmail:

- Content Filter
- Header filter
- General blacklists filter
- Rules-based filters
- Permission filters

Some machine learning algorithms such as **Multi-Layer Perceptron**, **Decision tree**, and **Naïve Bayes classifier** are used for email spam filtering and malware detection.

7. Virtual Personal Assistant:

We have various virtual personal assistants such as **Google assistant**, **Alexa**, **Cortana**, **Siri**. As the name suggests, they help us in finding the information using our voice instruction. These assistants can help us in various ways just by our voice instructions such as Play music, call someone, Open an email, Scheduling an appointment, etc.

These virtual assistants use machine learning algorithms as an important part.

These assistant record our voice instructions, send it over the server on a cloud, and decode it using ML algorithms and act accordingly.

8. Online Fraud Detection:

Machine learning is making our online transaction safe and secure by detecting fraud transaction. Whenever we perform some online transaction, there may be various ways that a fraudulent transaction can take place such as **fake accounts**, **fake ids**, and **steal money** in the middle of a transaction. So to detect this, **Feed Forward Neural network** helps us by checking whether it is a genuine transaction or a fraud transaction.

For each genuine transaction, the output is converted into some hash values, and these values become the input for the next round. For each genuine transaction, there is a specific pattern which gets change for the fraud transaction hence, it detects it and makes our online transactions more secure.

9. Stock Market trading:

Machine learning is widely used in stock market trading. In the stock market, there is always a risk of up and downs in shares, so for this machine learning's **long short term memory neural network** is used for the prediction of stock market trends.

10. Medical Diagnosis:

In medical science, machine learning is used for diseases diagnoses. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain. It helps in finding brain tumors and other brain-related diseases easily.

11. Automatic Language Translation:

Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages. Google's GNMT (Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation.

The technology behind the automatic translation is a sequence to sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.

Popular Algorithms used in machine learning:

There are numerous machine learning algorithms available, each with its strengths and weaknesses. The choice of algorithm depends on the nature of the problem, the type and size of the data, and the desired outcome. Here are some popular machine learning algorithms:

1. Linear Regression: A supervised learning algorithm used for regression tasks. It models the relationship between the dependent variable and one or more independent variables by fitting a

linear equation to the data.

2. Logistic Regression: A supervised learning algorithm used for classification tasks. It models the relationship between the independent variables and the probability of a binary outcome using the logistic function.

3. Decision Trees: Supervised learning algorithms that build a tree-like model of decisions and their possible consequences. They split the data based on feature values to make predictions.

4. Random Forests: An ensemble learning method that combines multiple decision trees to make predictions. It improves generalization and reduces overfitting compared to individual decision trees.

5. Support Vector Machines (SVM): A supervised learning algorithm used for both classification and regression tasks. SVM finds the best hyperplane that separates data points of different classes or predicts a continuous target variable.

6. Naive Bayes: A probabilistic supervised learning algorithm based on Bayes' theorem. It assumes independence among features and is particularly efficient for text classification and spam filtering tasks.

7. k-Nearest Neighbors (k-NN): A lazy learning algorithm that classifies new instances based on their similarity to existing labeled instances. It assigns the most frequent class label among the k nearest neighbors in the feature space.

8. Neural Networks: Deep learning algorithms that consist of interconnected layers of artificial neurons. They can learn complex patterns and relationships in data and are widely used for image recognition, natural language processing, and other tasks.

9. Gradient Boosting Methods: Ensemble learning techniques that combine weak learners, such as decision trees, in a sequential manner to create a strong predictive model. Examples include AdaBoost, Gradient Boosting Machines (GBM), and XGBoost.

10. Clustering Algorithms: Unsupervised learning algorithms used to identify groups or clusters within data. Examples include k-means clustering, hierarchical clustering, and DBSCAN.

11. Dimensionality Reduction Algorithms: Techniques used to reduce the number of features in a dataset while preserving essential information. Principal Component Analysis (PCA) and t-SNE (t-Distributed Stochastic Neighbor Embedding) are commonly used for dimensionality reduction.

12. Reinforcement Learning Algorithms: Algorithms that learn through interaction with an environment and receive rewards or penalties based on their actions. Reinforcement learning is often used in robotics, game playing, and control systems.

2		**List the Machine Learning Algorithm in testing near to excepted.**	**[L1][CO1]**	**[12M]**

1. Transformer-based Models: Transformers revolutionized natural language processing tasks, and ongoing research may have led to more advanced variants or adaptations for various domains.
2. GPT-3: This language model had already demonstrated impressive capabilities, and advancements might have resulted in even more powerful successors.
3. Deep Reinforcement Learning: Techniques like Deep Q-Networks (DQNs) and Proximal Policy Optimization (PPO) have been widely researched for their potential in complex decision-making tasks.
4. Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs): These

models were increasingly used for generating realistic data and enhancing data augmentation.
5. Meta-learning and Transfer Learning: Leveraging knowledge from previous tasks to improve performance on new, unseen tasks.
6. Neural Architecture Search (NAS): Automated methods for finding optimal neural network architectures for specific tasks.
7. Few-shot Learning: Developing models that can learn from a limited amount of labeled data.
8. Explainable AI: Techniques aimed at increasing the interpretability and transparency of machine learning models.
9. Federated Learning: Learning from decentralized data sources while maintaining data privacy.
10. Online Learning: Models that can continuously update themselves with new data as it arrives.

3	**a**	**Differentiate Machine Learning and Artificial Intelligence.**	**[L6][CO5]**	**[6M]**

Sl.No.	**ARTIFICIAL INTELLIGENCE**	**MACHINE LEARNING**
1.	1956 The terminology "Artificial Intelligence" was originally used by John McCarthy, who also hosted the first AI conference.	The terminology "Machine Learning" was first used in 1952 by IBM computer scientist Arthur Samuel, a pioneer in artificial intelligence and computer games.
2.	AI stands for Artificial intelligence, where intelligence is defined as the ability to acquire and apply knowledge.	ML stands for Machine Learning which is defined as the acquisition of knowledge or skill
3.	AI is the broader family consisting of ML and DL as its components.	Machine Learning is the subset of Artificial Intelligence.
4.	The aim is to increase the chance of success and not accuracy.	The aim is to increase accuracy, but it does not care about; the success
5.	AI is aiming to develop an intelligent system capable of performing a variety of complex jobs. decision-making	Machine learning is attempting to construct machines that can only accomplish the jobs for which they have been trained.
6.	It works as a computer program that does smart work.	Here, the tasks systems machine takes data and learns from data.
7.	The goal is to simulate natural intelligence to solve complex problems.	The goal is to learn from data on certain tasks to maximize the performance on that task.
8.	AI has a very broad variety of applications.	The scope of machine learning is constrained.

			from data.
	10.	It is developing a system that mimics humans to solve problems.	It involves creating self-learning algorithms.
	11.	AI will go for finding the optimal solution.	ML will go for a solution whether it is optimal or not.
	12.	AI leads to intelligence or wisdom.	ML leads to knowledge.
	13.	AI is a broader family consisting of ML and DL as its components.	ML is a subset of AI.
	14.	Three broad categories of AI are : 1. Artificial Narrow Intelligence (ANI) 2. Artificial General Intelligence (AGI) 3. Artificial Super Intelligence (ASI)	Three broad categories of ML are : 1. Supervised Learning 2. Unsupervised Learning 3. Reinforcement Learning
	15.	AI can work with structured, semi-structured, and unstructured data.	ML can work with only structured and semi-structured data.
	16.	AI's key uses include- • Siri, customer service via chatbots • Expert Systems • Machine Translation like Google Translate • Intelligent humanoid robots such as Sophia, and so on.	The most common uses of machine learning- • Facebook's automatic friend suggestions • Google's search algorithms • Banking fraud analysis • Stock price forecast • Online recommender systems, and so on.
	17.	AI refers to the broad field of creating machines that can simulate human intelligence and perform tasks such as understanding natural language, recognizing images and sounds, making decisions, and solving complex problems.	ML is a subset of AI that involves training algorithms on data to make predictions, decisions, and recommendations.
	18.	AI is a broad concept that includes various methods for creating intelligent machines, including rule-based systems, expert systems, and machine learning algorithms. AI systems can be programmed to follow specific rules, make logical inferences, or learn from data using ML.	focuses on teaching machines how to learn from data without being explicitly programmed, using algorithms such as neural networks, decision trees, and clustering.

19.	AI systems can be built using both structured and unstructured data, including text, images, video, and audio. AI algorithms can work with data in a variety of formats, and they can analyze and process data to extract meaningful insights.	In contrast, ML algorithms require large amounts of structured data to learn and improve their performance. The quality and quantity of the data used to train ML algorithms are critical factors in determining the accuracy and effectiveness of the system.
20.	AI is a broader concept that encompasses many different applications, including robotics, natural language processing, speech recognition, and autonomous vehicles. AI systems can be used to solve complex problems in various fields, such as healthcare, finance, and transportation.	ML, on the other hand, is primarily used for pattern recognition, predictive modeling, and decision making in fields such as marketing, fraud detection, and credit scoring.
21.	AI systems can be designed to work autonomously or with minimal human intervention, depending on the complexity of the task. AI systems can make decisions and take actions based on the data and rules provided to them.	In contrast, ML algorithms require human involvement to set up, train, and optimize the system. ML algorithms require the expertise of data scientists, engineers, and other professionals to design and implement the system.

b	**Describe classification techniques in supervised learning with an example**	**[L2][CO1]**	**[6M]**

Classification is a technique for determining which class the dependent belongs to based on one or more independent variables.

A classifier is a type of machine learning algorithm that assigns a label to a data input. Classifier algorithms use labeled data and statistical methods to produce predictions about data input classifications.

There are different types of Classification techniques used in Supervised learning.

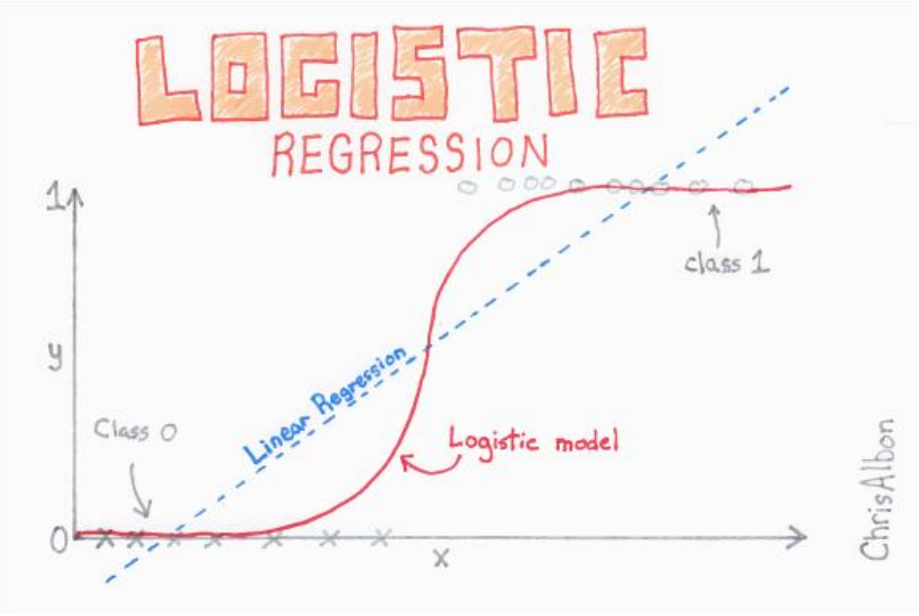
1. Logistic Regression
2. K-Nearest Neighbor
3. Support Vector Machine
    - Kernel SVM
    - Radial Basis Function
4. Naïve Bayes
5. Decision Tree Classification

Ensemble Methods for Classification:

1. Random Forest Classification
2. Gradient Boosting Classification

1. LOGISTIC REGRESSION:

Logistic regression is kind of like linear regression, but is used when the dependent variable is not a number but something else (e.g., a "yes/no" response). It's called regression but performs classification based on the regression and it classifies the dependent variable into either of the classes.



Logistic regression is used for prediction of output which is binary, as stated above. For example, if a credit card company builds a model to decide whether or not to issue a credit card to a customer, it will model for whether the customer is going to "default" or "not default" on their card.

$$y = b_0 + b_1 x$$

Linear Regression

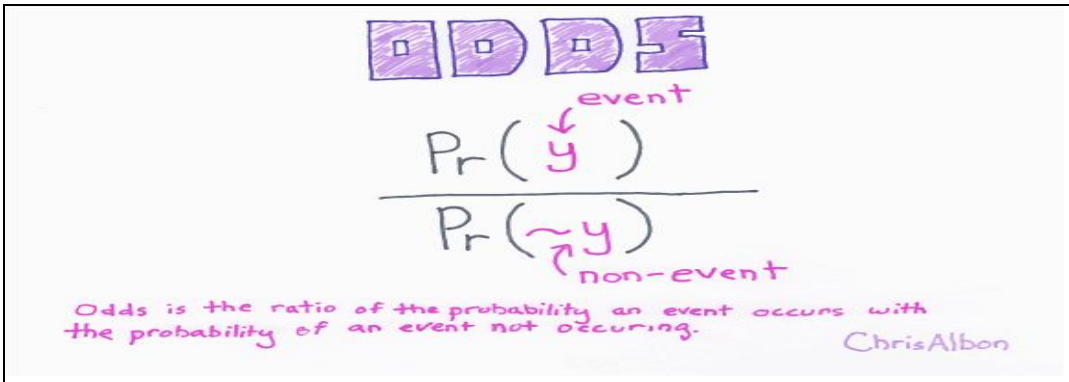
Firstly, linear regression is performed on the relationship between variables to get the model. The threshold for the classification line is assumed to be at 0.5.

$$p = \frac{1}{1 + e^{-y}}$$

Logistic Sigmoid Function

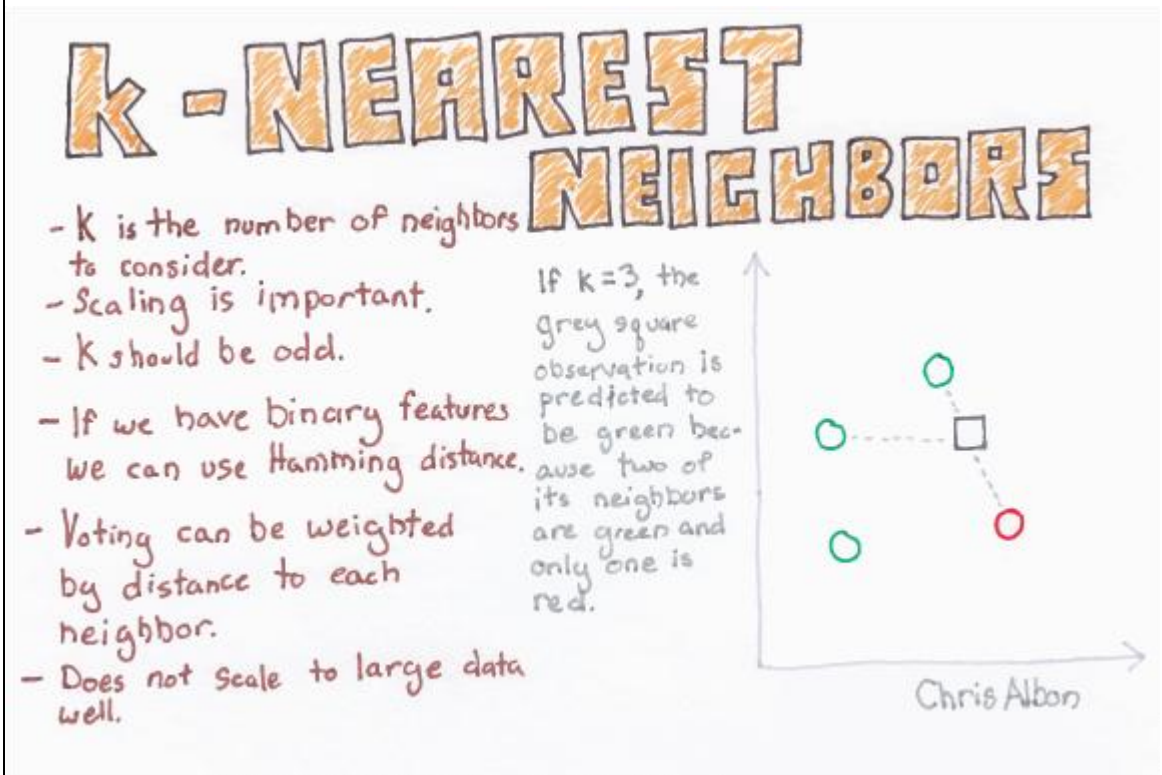
Logistic function is applied to the regression to get the probabilities of it belonging in either class.

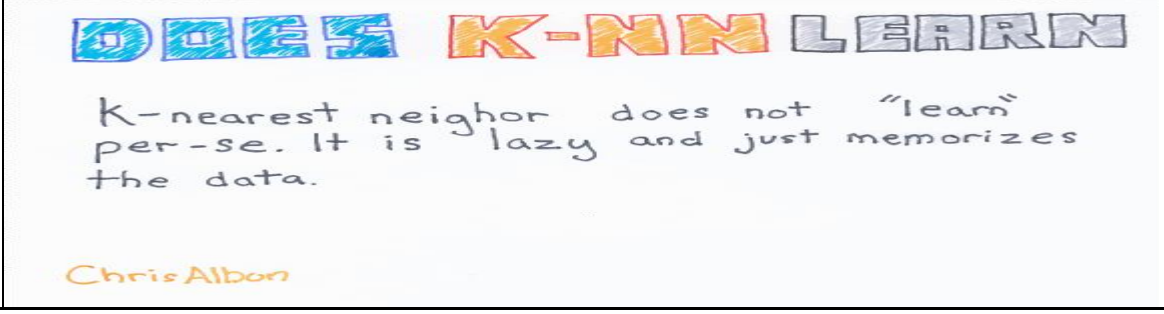
It gives the log of the probability of the event occurring to the log of the probability of it not occurring. In the end, it classifies the variable based on the higher probability of either class.

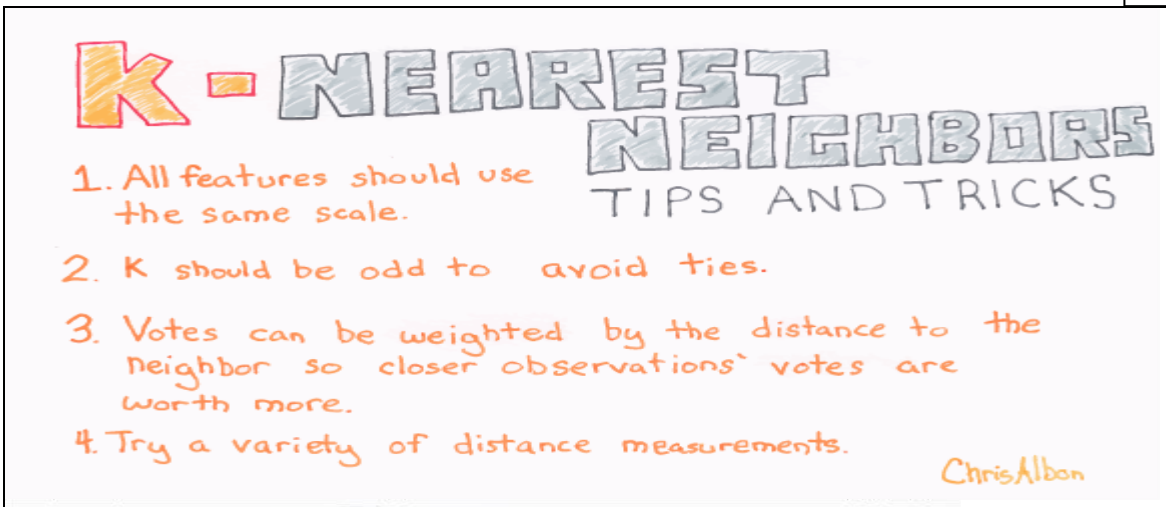


2. K-NEAREST NEIGHBORS (K-NN)

K-NN algorithm is one of the simplest classification algorithms and it is used to identify the data points that are separated into several classes to predict the classification of a new sample point. K-NN is a non-parametric**,** lazy learning algorithm. It classifies new cases based on a similarity measure (i.e., distance functions).



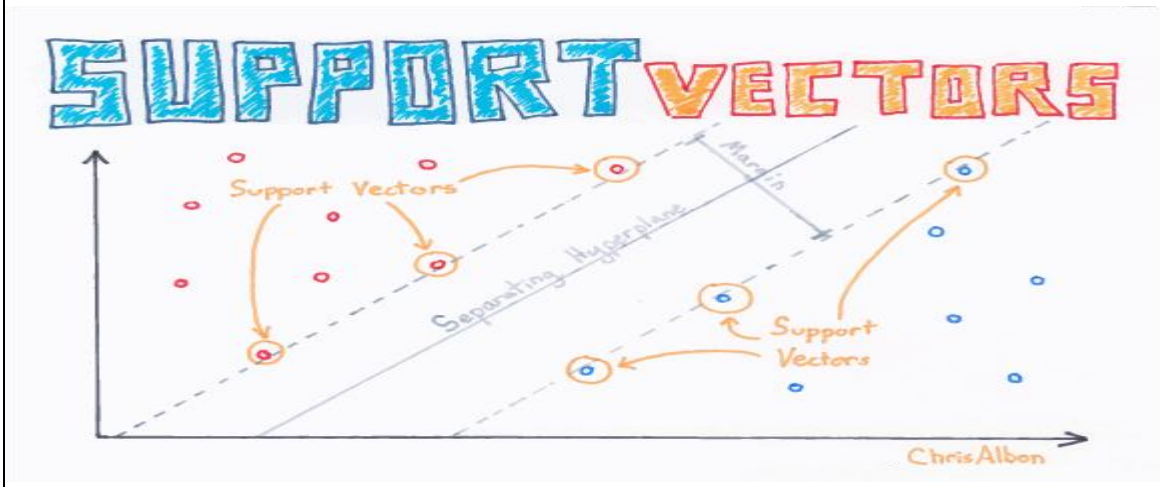




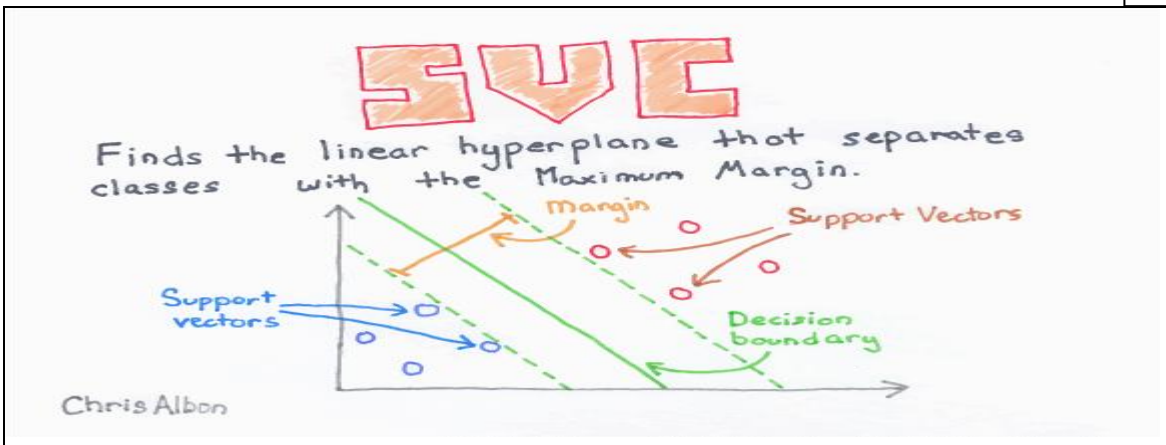
K-NN works well with a small number of input variables (*p*), but struggles when the number of inputs is very large.

3. SUPPORT VECTOR MACHINE (SVM)

Support vector is used for both regression and classification. It is based on the concept of decision planes that define decision boundaries. A decision plane (hyperplane) is one that separates between a set of objects having different class memberships.



It performs classification by finding the hyperplane that maximizes the margin between the two classes with the help of support vectors.



The learning of the hyperplane in SVM is done by transforming the problem using some linear algebra (i.e., the example above is a linear kernel which has a linear separability between each variable).

For higher dimensional data, other kernels are used as points and cannot be classified easily. They are specified in the next section.

Kernel SVM

Kernel SVM takes in a kernel function in the SVM algorithm and transforms it into the required form that maps data on a higher dimension which is separable.Types of kernel functions:

$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma |\mathbf{X}_i - \mathbf{X}_j|^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{cases}$$

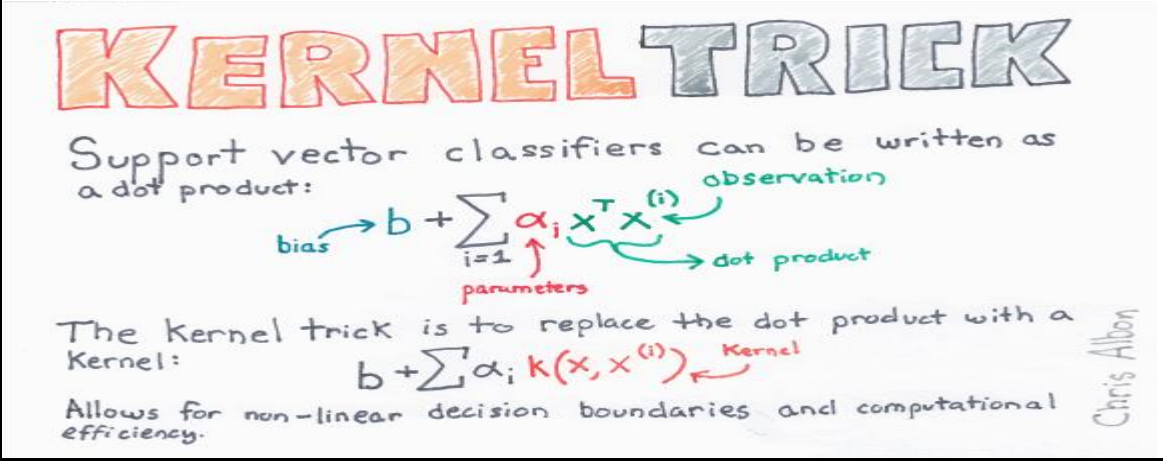
Type of kernel functions

Linear SVM is the one we discussed earlier.

In polynomial kernel, the degree of the polynomial should be specified. It allows for curved lines in the input space.

In the radial basis function (RBF) kernel, it is used for non-linearly separable variables. For distance, metric squared Euclidean distance is used. Using a typical value of the parameter can lead to overfitting our data. It is used by default in sklearn.

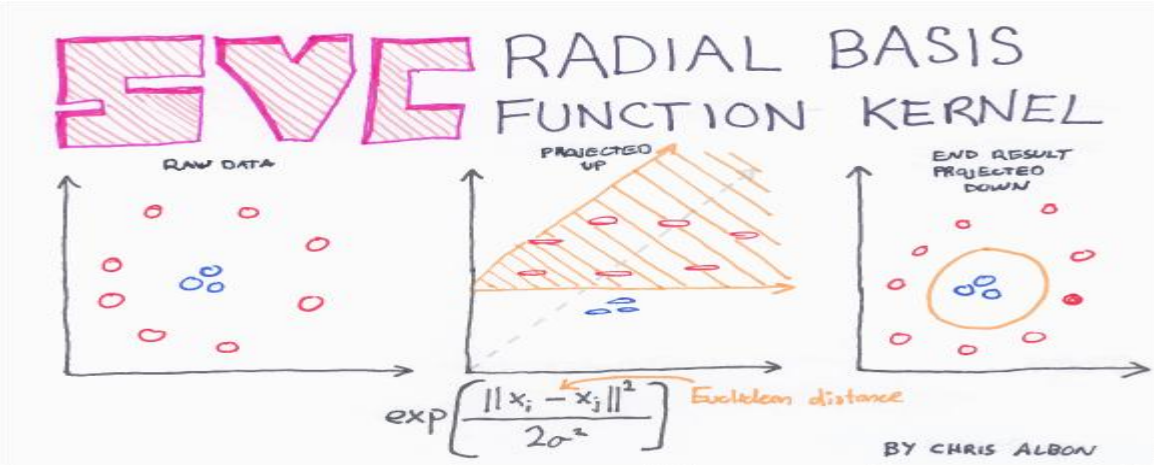
Sigmoid kernel, similar to logistic regression is used for binary classification.



Kernel trick uses the kernel function to transform data into a higher dimensional feature space and makes it possible to perform the linear separation for classification.

Radial Basis Function (RBF) Kernel

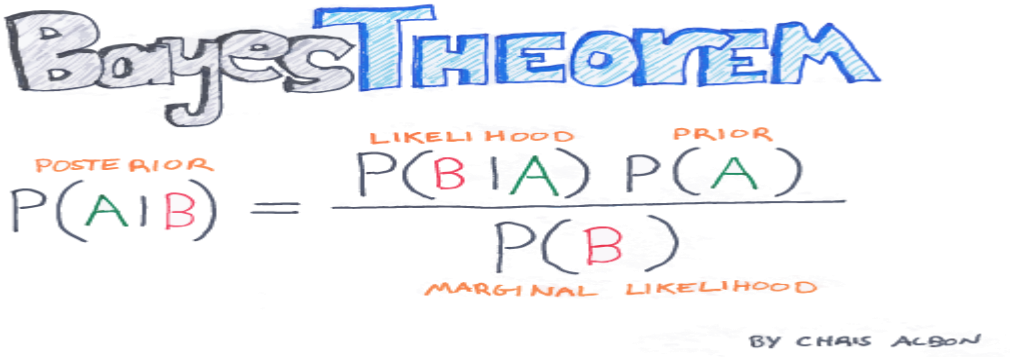
The RBF kernel SVM decision region is actually also a linear decision region. What RBF kernel SVM actually does is create non-linear combinations of features to uplift the samples onto a higher-dimensional feature space where a linear decision boundary can be used to separate classes.



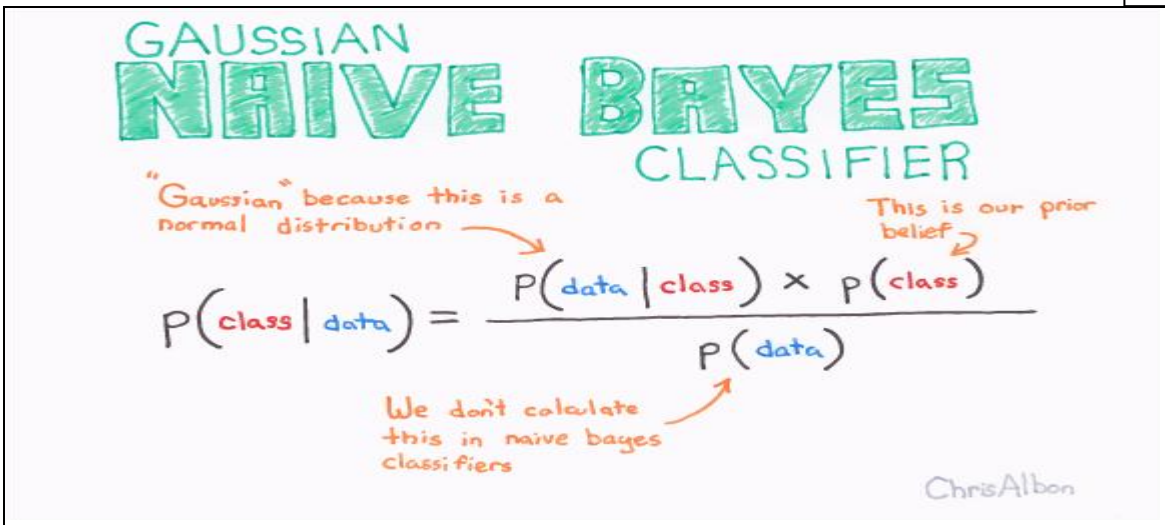
So, the rule of thumb is: use linear SVMs for linear problems, and nonlinear kernels such as the RBF kernel for non-linear problems.

4. NAIVE BAYES

The naive Bayes classifier is based on Bayes' theorem with the independence assumptions between predictors (i.e., it assumes the presence of a feature in a class is unrelated to any other feature). Even if these features depend on each other, or upon the existence of the other features, all of these properties independently. Thus, the name naive Bayes.



Based on naive Bayes, Gaussian naive Bayes is used for classification based on the binomial (normal) distribution of data.

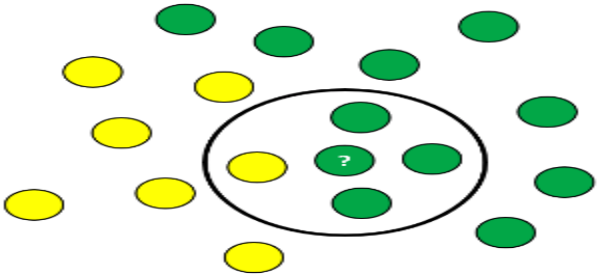


P(class|data) is the posterior probability of *class*(*target*) given *predictor*(*attribute*). The probability of a data point having either class, given the data point. This is the value that we are looking to calculate.

P(class) is the prior probability of *class*.

P(data|class) is the likelihood, which is the probability of *predictor* given *class*.

P(data) is the prior probability of *predictor or marginal likelihood*.



NB Classification Example

Naive Bayes Steps

1. Calculate Prior Probability

P(class) = Number of data points in the class/Total no. of observations

P(yellow) = 10/17

P(green) = 7/17

2. Calculate Marginal Likelihood

P(data) = Number of data points similar to observation/Total no. of observations

P(?) = 4/17

The value is present in checking both the probabilities.

3. Calculate Likelihood

P(data/class) = Number of similar observations to the class/Total no. of points in the class.

P(?/yellow) = 1/7

P(?/green) = 3/10

5. Posterior Probability for Each Class

$$p(class/data) = \frac{P(data/class) * P(class)}{P(data)}$$

$$P(yellow/?) = \frac{1/7 * 7/17}{4/17} = 0.25$$

$$P(green/?) = \frac{3/10 * 10/17}{4/17} = 0.75$$

6. Classification:

$$P(class1/data) > P(class2/data)$$

$$P(green/?) > P(yellow/?)$$

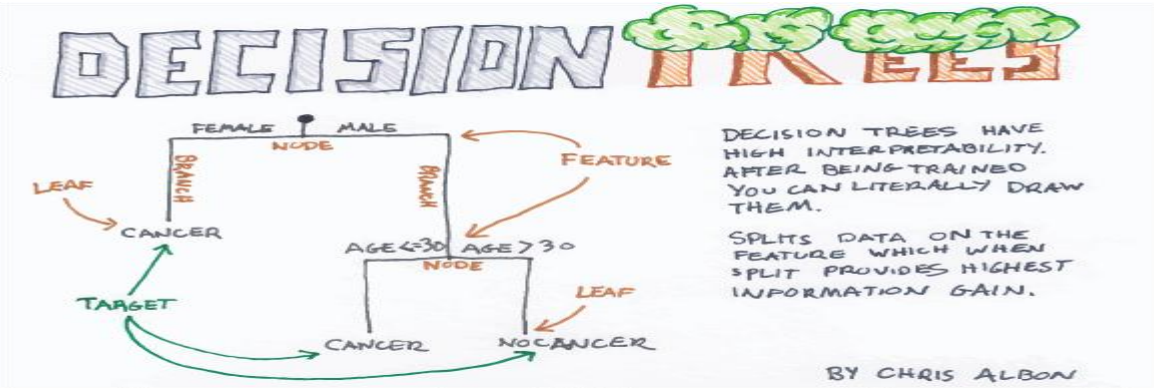
The higher probability, the class belongs to that category as from above 75% probability the point belongs to class green.

Multinomial, Bernoulli naive Bayes are the other models used in calculating probabilities. Thus, a naive Bayes model is easy to build, with no complicated iterative parameter estimation, which makes it particularly useful for very large datasets.

5. DECISION TREE CLASSIFICATION

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a

dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. It follows Iterative Dichotomiser 3 (ID3) algorithm structure for determining the split.



Entropy and information gain are used to construct a decision tree.

Entropy

Entropy is the degree or amount of uncertainty in the randomness of elements. In other words, it is a measure of impurity**.**

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Entropy

Intuitively, it tells us about the predictability of a certain event. Entropy calculates the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero, and if the sample is equally divided it has an entropy of one.

Information Gain

Information gain measures the relative change in entropy with respect to the independent attribute. It tries to estimate the information contained by each attribute. Constructing a decision tree is all about finding the attribute that returns the highest information gain (i.e., the most homogeneous branches).

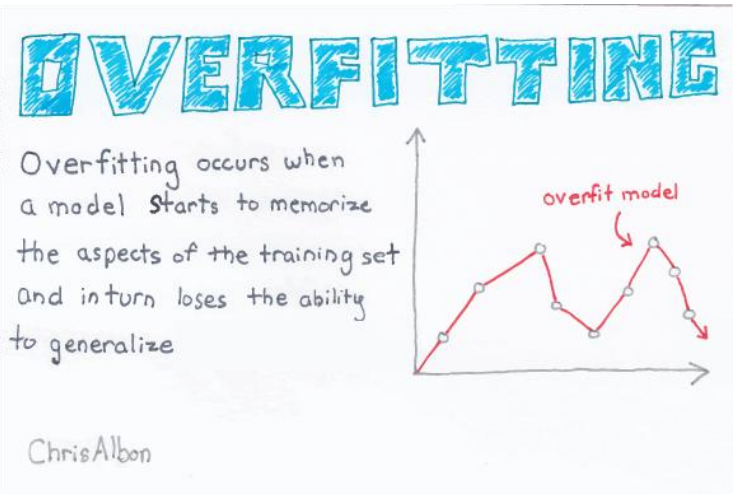
$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

Where *Gain(T, X)* is the information gain by applying feature *X*. *Entropy(T)* is the entropy of the entire set,

while the second term calculates the entropy after applying the feature *X*.

Information gain ranks attributes for filtering at a given node in the tree. The ranking is based on the highest information gain entropy in each split.

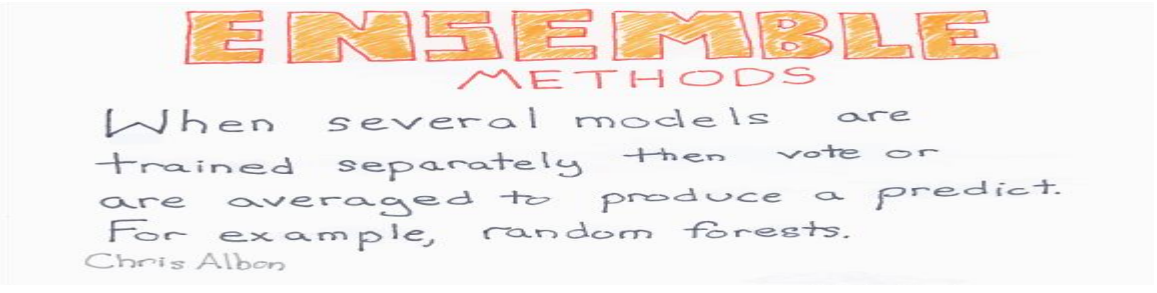
The disadvantage of a decision tree model is overfitting, as it tries to fit the model by going deeper in the training set and thereby reducing test accuracy.



Overfitting in decision trees can be minimized by pruning nodes.

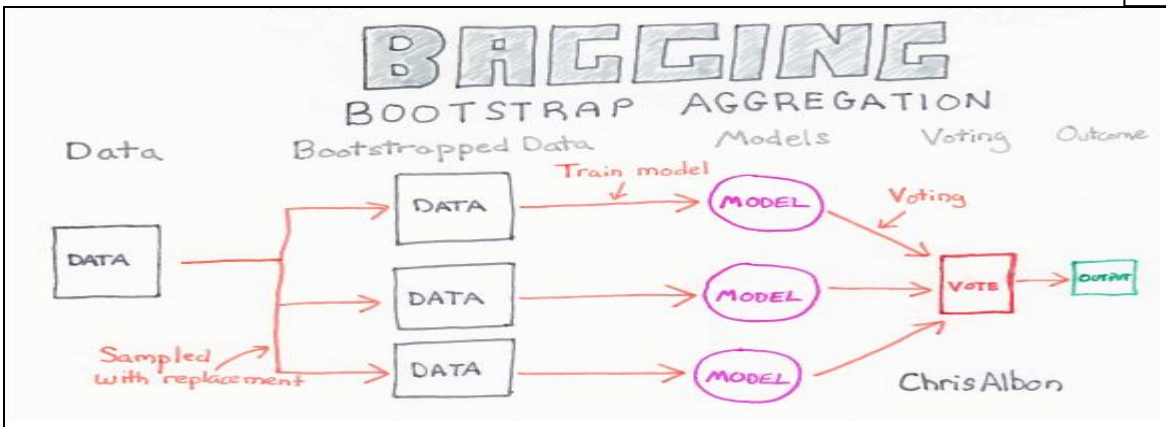
Ensemble Methods for Classification

An ensemble model is a *team of models*. Technically, ensemble models comprise several supervised learning models that are individually trained and the results merged in various ways to achieve the final prediction. This result has higher predictive power than the results of any of its constituting learning algorithms independently.

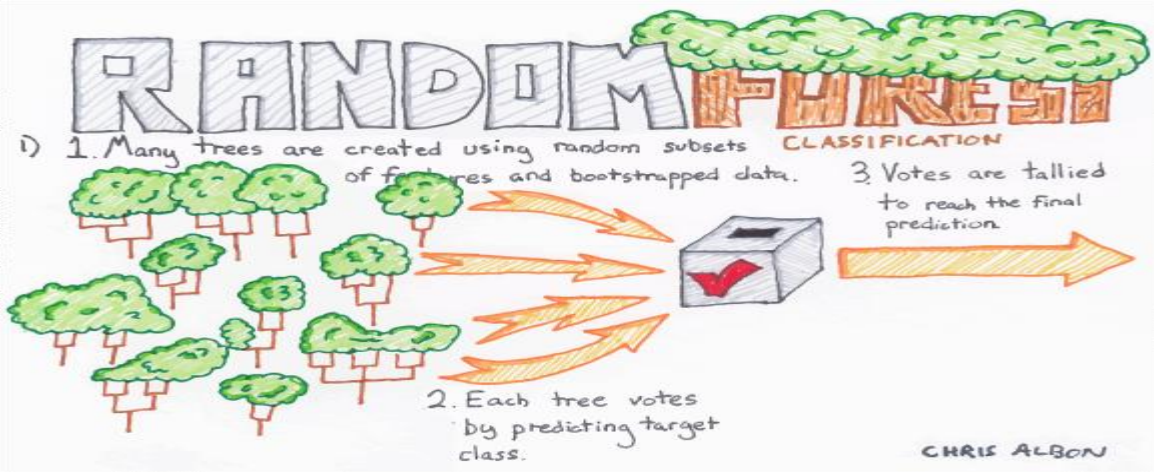


1. RANDOM FOREST CLASSIFICATION

Random forest classifier is an ensemble algorithm based on bagging i.e bootstrap aggregation. Ensemble methods combines more than one algorithm of the same or different kind for classifying objects (i.e., an ensemble of SVM, naive Bayes or decision trees, for example.)



The general idea is that a combination of learning models increases the overall result selected.

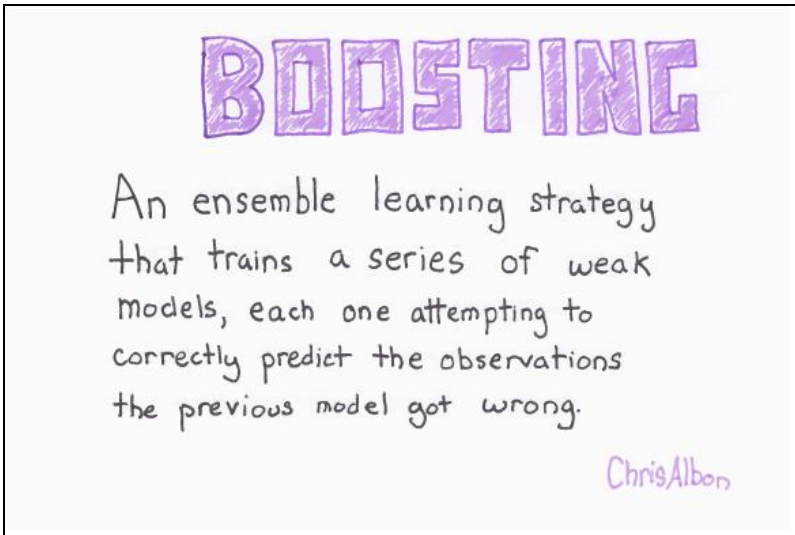


Deep decision trees may suffer from overfitting, but random forests prevent overfitting by creating trees on random subsets. The main reason is that it takes the average of all the predictions, which cancels out the biases.

Random forest adds additional randomness to the model while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

2. GRADIENT BOOSTING CLASSIFICATION

Gradient boosting classifier is a boosting ensemble method. Boosting is a way to combine (ensemble) weak learners, primarily to reduce prediction bias. Instead of creating a pool of predictors, as in bagging, boosting produces a cascade of them, where each output is the input for the following learner. Typically, in a bagging algorithm trees are grown in parallel to get the average prediction across all trees, where each tree is built on a sample of original data. Gradient boosting, on the other hand, takes a sequential approach to obtaining predictions instead of parallelizing the tree building process. In gradient boosting, each decision tree predicts the error of the previous decision tree—thereby *boosting* (improving) the error (gradient).



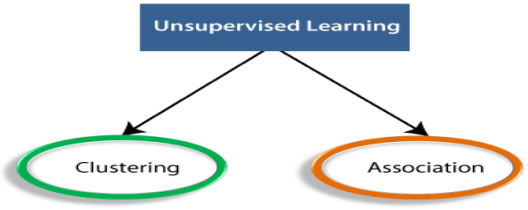
Working of Gradient Boosting

Initialize predictions with a simple decision tree.
Calculate residual (actual-prediction) value.
Build another shallow decision tree that predicts residual based on all the independent values.
Update the original prediction with the new prediction multiplied by learning rate.

Repeat steps two through four for a certain number of iterations (the number of iterations will be the number of trees).

4	a	List out various Unsupervised learning techniques used in Machine Learning.	**[L1][CO5]**	**[4M]**

Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.



- **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Types of Clustering Methods

The clustering methods are broadly divided into **Hard clustering** (datapoint belongs to only one group) and **Soft Clustering** (data points can belong to another group also). But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning:

1. **Partitioning Clustering**
2. **Density-Based Clustering**
3. **Distribution Model-Based Clustering**
4. **Hierarchical Clustering**
5. **Fuzzy Clustering**

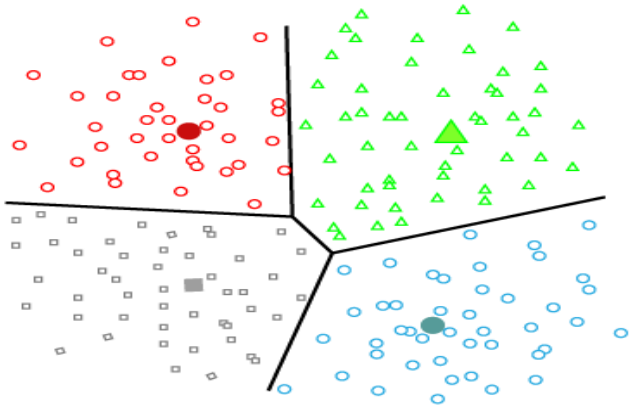
b	Analyze the clustering techniques with examples.	**[L3][CO2]**	**[8M]**

Types of Clustering Methods

Partitioning Clustering

It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the **centroid-based method**. The most common example of partitioning clustering is the **K-Means Clustering algorithm**.

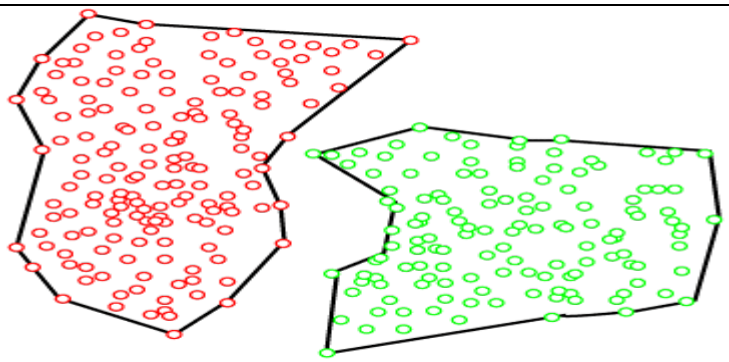
In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.



Density-Based Clustering

The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected. This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

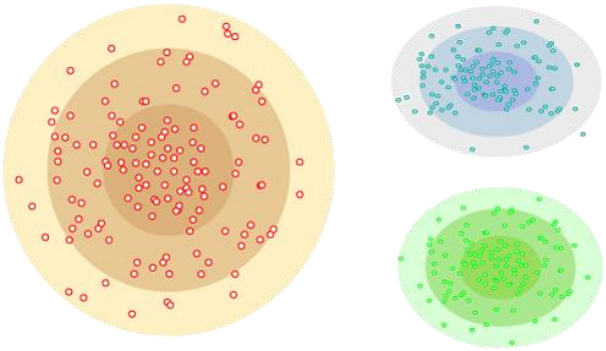
These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.



Distribution Model-Based Clustering

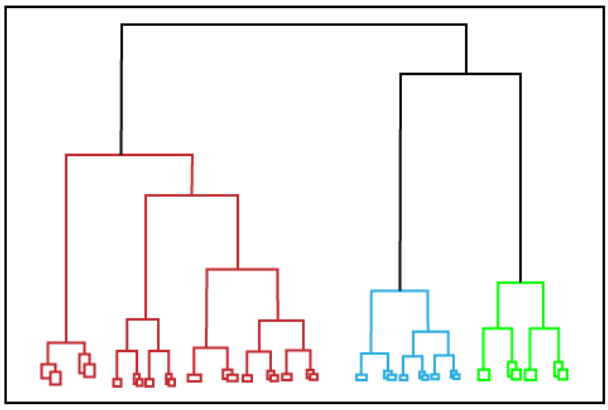
In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly **Gaussian Distribution**.

The example of this type is the **Expectation-Maximization Clustering algorithm** that uses Gaussian Mixture Models (GMM).



Hierarchical Clustering

Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a **dendrogram**. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the **Agglomerative Hierarchical algorithm**.



Fuzzy Clustering

Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster. **Fuzzy C-means algorithm** is the example of this type of clustering; it is sometimes also

known as the Fuzzy k-means algorithm.

Clustering Algorithms

The Clustering algorithms can be divided based on their models that are explained above. There are different types of clustering algorithms published, but only a few are commonly used. The clustering algorithm is based on the kind of data that we are using. Such as, some algorithms need to guess the number of clusters in the given dataset, whereas some are required to find the minimum distance between the observation of the dataset.

Here we are discussing mainly popular Clustering algorithms that are widely used in machine learning:

1. **K-Means algorithm:** The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of **O(n).**
2. **Mean-shift algorithm:** Mean-shift algorithm tries to find the dense areas in the smooth density of data points. It is an example of a centroid-based model, that works on updating the candidates for centroid to be the center of the points within a given region.
3. **DBSCAN Algorithm:** It stands **for Density-Based Spatial Clustering of Applications with Noise**. It is an example of a density-based model similar to the mean-shift, but with some remarkable advantages. In this algorithm, the areas of high density are separated by the areas of low density. Because of this, the clusters can be found in any arbitrary shape.
4. **Expectation-Maximization Clustering using GMM:** This algorithm can be used as an alternative for the k-means algorithm or for those cases where K-means can be failed. In GMM, it is assumed that the data points are Gaussian distributed.
5. **Agglomerative Hierarchical algorithm:** The Agglomerative hierarchical algorithm performs the bottom-up hierarchical clustering. In this, each data point is treated as a single cluster at the outset and then successively merged. The cluster hierarchy can be represented as a tree-structure.

1. **Affinity Propagation:** It is different from other clustering algorithms as it does not require to specify the number of clusters. In this, each data point sends a message between the pair of data points until convergence. It has $O(N^2T)$ time complexity, which is the main drawback of this algorithm.

5	Summarize the Guidelines for Machine Learning Experiments.	**[L2][CO3]**	**[12M]**

Supervised learning is one of the fundamental types of machine learning techniques. In supervised learning, the algorithm is trained on a labeled dataset, where each input data point is associated with the corresponding correct output label. The goal of the algorithm is to learn a mapping from the input data to the output labels so that it can make accurate predictions on unseen data.

The supervised learning process can be broken down into the following steps:

1. **Data Collection**: Gather a labeled dataset where each data point consists of input features (also known as independent variables) and their corresponding output labels (also known

as dependent variables). The dataset is split into two parts: a training set and a test set.

2. **Training**: During the training phase, the algorithm uses the training data to learn the relationship between the input features and the output labels. It tries to find the best possible model that can map inputs to outputs accurately.
3. **Model Selection**: Choose a specific type of supervised learning algorithm depending on the nature of the problem. Common algorithms include Linear Regression, Decision Trees, Support Vector Machines (SVM), Neural Networks, and many others. The choice of the algorithm depends on factors such as the type of data, the complexity of the problem, and the interpretability required.
4. **Feature Engineering**: Preprocess the input data to extract relevant features and potentially reduce noise or irrelevant information. Feature engineering can significantly impact the performance of the model.
5. **Training the Model**: The selected algorithm is fed with the labeled training data to optimize its internal parameters and find the best-fitting model that minimizes the error between its predictions and the true output labels.
6. **Evaluation**: Once the model is trained, it is evaluated using the test set, which the algorithm has not seen during training. The performance metrics, such as accuracy, precision, recall, F1-score, etc., are calculated to assess how well the model generalizes to new, unseen data.
7. **Prediction**: After successful training and evaluation, the model is ready to make predictions on new, unlabeled data. It takes the input features and produces the corresponding output label based on what it has learned during training.

Some key characteristics and use cases of supervised learning include:

- **Classification**: When the output labels are discrete or categorical, the problem is known as classification. For example, email spam detection (where the output can be "spam" or "not spam") is a typical classification problem.
- **Regression**: When the output labels are continuous numerical values, the problem is referred to as regression. For example, predicting house prices based on features like square footage and number of bedrooms is a regression problem.

6	a	Explain Model Selection in Machine learning.	**[L2][CO1]**	**[6M]**

Model Selection: Model selection refers to the process of choosing the best model from a set of candidate models for a specific task or problem. In machine learning, a model is a mathematical representation of the relationships between input variables (features) and the target variable (output). Model selection is crucial because different models have different complexities, assumptions, and performance characteristics, and choosing an appropriate model can greatly impact the accuracy and efficiency of the learning system.

Inductive bias

In a learning problem we only have the data. But data by itself is not sufficient to find the solution. We should make some extra assumptions to have a solution with the data we have. The set of assumptions we make to have learning possible is called the inductive bias of the learning algorithm. One way we introduce inductive bias is when we assume a hypothesis class.

Examples

- In learning the class of family car, there are infinitely many ways of separating the positive examples from the negative examples. Assuming the shape of a rectangle is an inductive bias.
- In regression, assuming a linear function is an inductive bias.

The model selection is about choosing the right inductive bias.

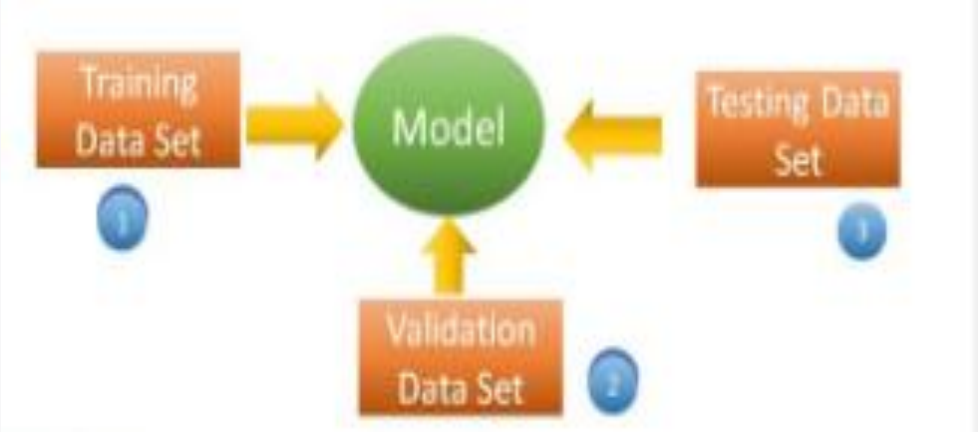
Advantages of a simple model

Even though a complex model may not be making any errors in prediction, there are certain advantages in using a simple model.

1. A simple model is easy to use.
2. A simple model is easy to train. It is likely to have fewer parameters. It is easier to find the corner values of a rectangle than the control points of an arbitrary shape.
3. A simple model is easy to explain.
4. A simple model would generalize better than a complex model. This principle is known as Occam's razor, which states that simpler explanations are more plausible and any unnecessary complexity should be shaved off.

Here are a few common considerations for model selection:

Model Complexity: Complex models can potentially capture intricate patterns in the data but may be prone to overfitting,. The trade-off between complexity and generalization is often a key factor in model selection.
Domain Knowledge: Understanding the problem domain and having prior knowledge about the data can guide the selection of an appropriate model.
Training Data Availability: The amount of available training data influences the choice of model.
Model Performance Metrics: The choice of performance metrics depends on the nature of the problem. For example, in classification tasks, metrics like accuracy, precision, recall, and F1-score are commonly used.
Computational Resources: When selecting a model, it's important to consider the available computational resources, such as processing power, memory, and time constraints.



	b	Discriminate Generalization in machine learning with examples	**[L5][CO1]**	**[6M]**

Generalization: Generalization refers to the ability of a trained model to perform well on unseen or new data that it hasn't encountered during the training phase. The ultimate goal of machine learning is to develop models that generalize well, as they can make accurate predictions or decisions on real-world, unseen instances.

To achieve good generalization, it's important to balance model complexity and simplicity. If a model is too simple, it may underfit the data, failing to capture important patterns. On the other hand, if a model is too complex, it may overfit the training data, memorizing noise or irrelevant details and performing poorly on new data.

Regularization techniques, such as L1 and L2 regularization, dropout, or early stopping, can help control the complexity of models and prevent overfitting.

Generalisation

How well a model trained on the training set predicts the right output for new instances is called generalization.

Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning. The goal of a good machine learning model is to generalize well from the training data to any data from the problem domain. This allows us to make predictions in the future on data the model has never seen. Overfitting and underfitting are the two biggest causes for poor performance of machine learning algorithms. The model should be selected having the best generalisation. This is said to be the case if these problems are avoided.

- **Underfitting**

  Underfitting is the production of a machine learning model that is not complex enough to accurately capture relationships between a datasetâA˘ Zs´ features and a target variable.

- **Overfitting**

  Overfitting is the production of an analysis which corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably.

Example 1

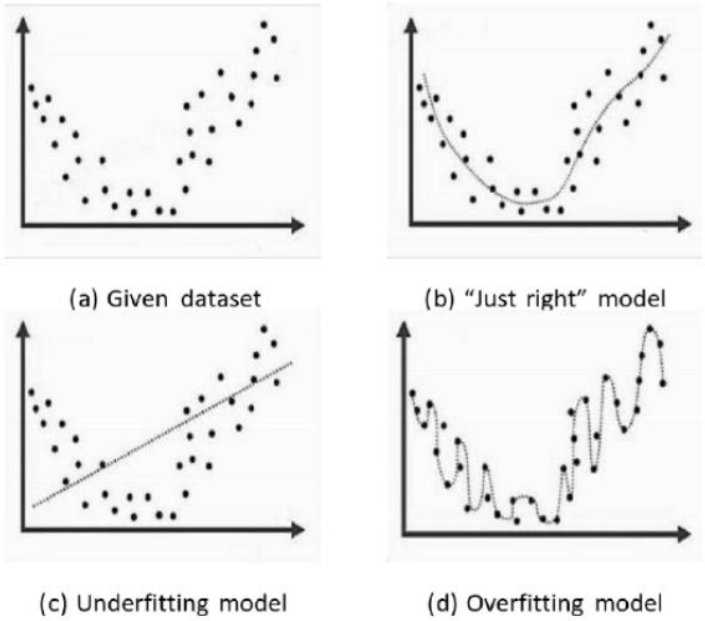


Figure: Examples for overfitting and overfitting models

7	a	Compare Supervised learning and Unsupervised learning	**[L6][CO1]**	**[6M]**

Supervised Learning	**Unsupervised Learning**
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in **Classification** and **Regression** problems.	Unsupervised Learning can be classified in **Clustering** and **Associations** problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result	Unsupervised learning model may give less accurate result as compared to supervised

	learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

b	Analyze Reinforcement Learning with neat diagram..	**[L4][CO1]**	**[6M]**

Reinforcement learning:

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

Example: The problem is as follows: We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The following problem explains the problem more easily.

The above image shows the robot, diamond, and fire. The goal of the robot is to get the reward that is the diamond and avoid the hurdles that are fired. The robot learns by trying all the possible paths and then choosing the path which gives him the reward with the least hurdles. Each right step will give the robot a reward and each wrong step will subtract the reward of the robot. The total reward will be calculated when it reaches the final reward that is the diamond.

Main points in Reinforcement learning –

- Input: The input should be an initial state from which the model will start
- Output: There are many possible outputs as there are a variety of solutions to a particular problem
- Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
- The model keeps continues to learn.
- The best solution is decided based on the maximum reward.

Types of Reinforcement: There are two types of Reinforcement:

1. **Positive –**

Positive Reinforcement is defined as when an event, occurs due to a particular behavior, increases the strength and the frequency of the behavior. In other words, it has a positive effect on behavior.
Advantages of reinforcement learning 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

2. **Negative –**

Negative Reinforcement is defined as strengthening of behavior because a negative condition is stopped or avoided.

Advantages of reinforcement learning:

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

Various Practical applications of Reinforcement Learning –

- RL can be used in robotics for industrial automation.
- RL can be used in machine learning and data processing
- RL can be used to create training systems that provide custom instruction and materials according to the requirement of students.
- RL can be used in large environments in the following situations:
  A model of the environment is known, but an analytic solution is not available;

8	Discuss clustering and association rules in unsupervised learning.	**[L2][CO2]**	**[12M]**

Clustering 4(b) answer.

ASSOCIATION RULES:

Association rule learning is a kind of unsupervised learning technique that tests for the reliance of one data element on another data element and design appropriately so that it can be more cost-effective. It tries to discover some interesting relations or associations between the variables of the dataset. It depends on various rules to find interesting relations between variables in the database.

The association rule learning is the most important approach of machine learning, and it is employed in Market Basket analysis, Web usage mining, continuous production, etc. In market basket analysis, it is an approach used by several big retailers to find the relations between items.

Types of Association Rule Learning

There are the following types of Association rule learning which are as follows −

Apriori Algorithm − This algorithm needs frequent datasets to produce association rules. It is designed to work on databases that include transactions. This algorithm needs a breadth-first search and hash tree to compute the itemset efficiently.

It is generally used for market basket analysis and support to learn the products that can be purchased together. It can be used in the healthcare area to discover drug reactions for patients.

Eclat Algorithm − The Eclat algorithm represents Equivalence Class Transformation. This algorithm needs a depth-first search method to discover frequent itemsets in a transaction database. It implements quicker execution than Apriori Algorithm.

F-P Growth Algorithm − The F-P growth algorithm represents Frequent Pattern. It is the enhanced version of the Apriori Algorithm. It describes the database in the form of a tree structure that is referred to as a frequent pattern or tree. This frequent tree aims to extract the most frequent patterns.

There are various applications of Association Rule which are as follows −

- Items purchased on a credit card, such as rental cars and hotel rooms, support insight into the following product that customer are likely to buy.
- Optional services purchased by tele-connection users (call waiting, call forwarding, DSL, speed call, etc.) support decide how to bundle these functions to maximize revenue.
- Banking services used by retail users (money industry accounts, CDs, investment services, car loans, etc.) recognize users likely to needed other services.
- Unusual group of insurance claims can be an expression of fraud and can spark higher investigation.

Medical patient histories can supports expressions of likely complications based on definite set of

	treatments.		
9	Analyze the classification and regression techniques in supervised learning.	**[L4][CO1]**	**[12M]**

Classification Techniques referred previous answers

Regression techniques:

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Logistic Regression
- Stepwise Regression
- Polynomial Regression

1. Linear Regression

It is one of the most widely known modeling techniques and the most famous regression technique in Machine Learning. Linear regression is usually among the first few topics which people pick while learning predictive modeling. In this technique, the dependent variable is continuous, the independent variable(s) can be continuous or discrete, and the nature of the regression line is linear.

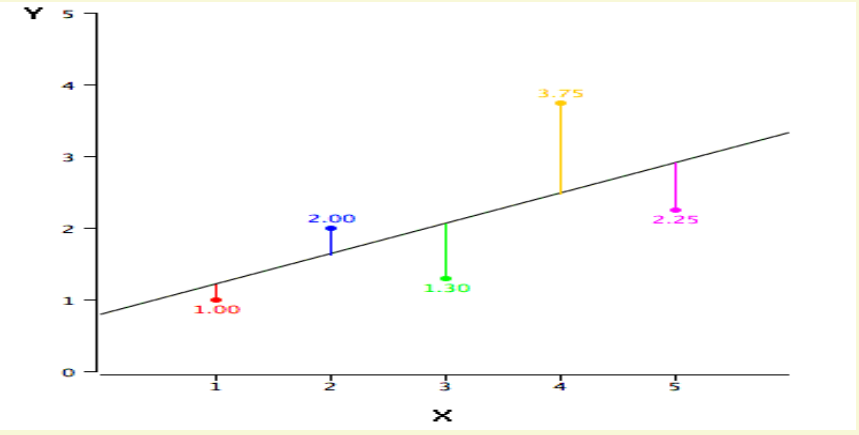
Linear Regression establishes a relationship between the dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as Regression line).

It is represented by an equation Y=a+b*X + e, where a is the intercept, b is the slope of the line and e is error term. This equation can be used to predict the value of the target variable based on the given predictor variable(s).

The difference between simple linear regression and multiple linear regression, multiple linear regression has (>1) independent variables, whereas simple linear regression has only 1 independent variable. Now, the question is "How do we obtain best-fit line?".

How to obtain the best fit line (Value of a and b)?

This task can be easily accomplished by Least Square Method. It is the most common method used for fitting a regression line. It calculates the best-fit line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line. Because the deviations are first squared, when added, there is no canceling out between positive and negative values.



We can evaluate the model performance using the metric R-square.

2. Logistic Regression

Logistic regression in Machine Learning is used to find the probability of event=Success and event=Failure. We should use logistic regression when the dependent variable is binary (0/ 1, True/ False, Yes/ No) in nature. Here the value of Y ranges from 0 to 1 and it can be represented by the following equation.

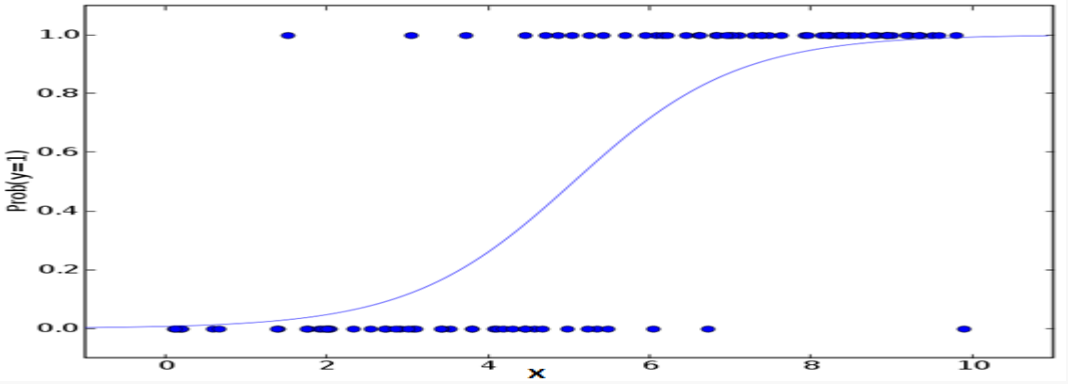
odds= p/ (1-p) = probability of event occurrence / probability of not event occurrence

ln(odds) = ln(p/(1-p))

logit(p) = ln(p/(1-p)) = b0+b1X1+b2X2+b3X3....+bkXk

Above, p is the probability of the presence of the characteristic of interest. A question that you should ask here is "why have we used to log in the equation?".

Since we are working here with a binomial distribution (dependent variable), we need to choose a link function which is best suited for this distribution. And, it is a logit function. In the equation above, the parameters are chosen to maximize the likelihood of observing the sample values rather than minimizing the sum of squared errors (like in ordinary regression).



Important Points:

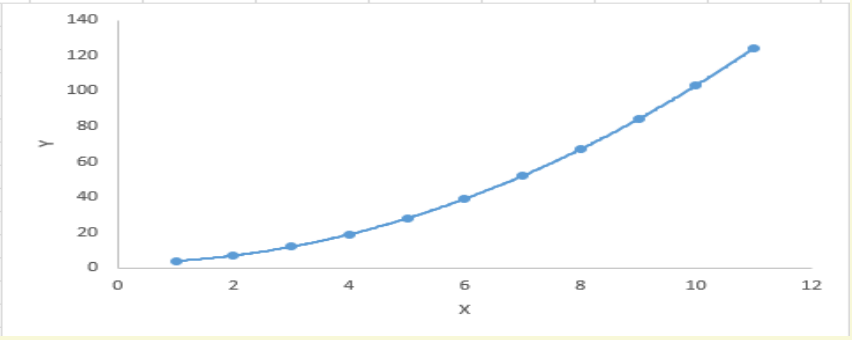
- It is widely used for classification problems
- Logistic regression doesn't require a linear relationship between dependent and independent variables. It can handle various types of relationships because it applies a non-linear log transformation to the predicted odds ratio
- To avoid over fitting and under fitting, we should include all significant variables. A good approach to ensure this practice is to use a stepwise method to estimate the logistic regression.
- It requires large sample sizes because maximum likelihood estimates are less powerful at low sample sizes than ordinary least square.
- The independent variables should not be correlated with each other i.e. no multicollinearity. However, we have the option to include interaction effects of categorical variables in the analysis and the model.
- If the values of the dependent variable are ordinal, then it is called as Ordinal logistic regression.
- If the dependent variable is multi-class then it is known as Multinomial Logistic regression.

3. Polynomial Regression

A regression equation in Machine Learning is a polynomial regression equation if the power of the independent variable is more than 1. The equation below represents a polynomial equation:

y=a+b*x^2

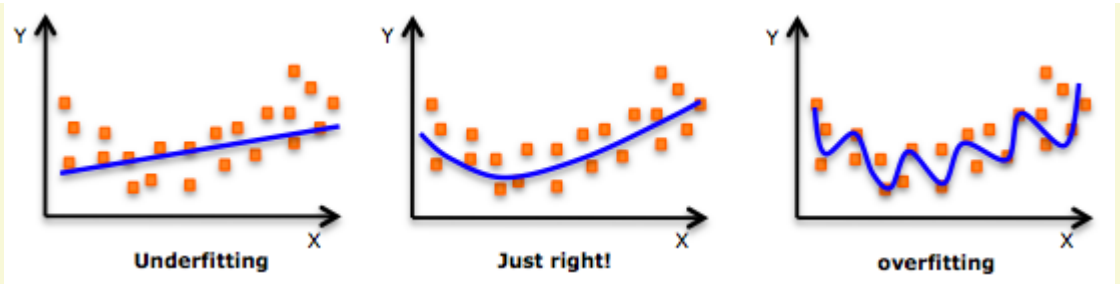
In this regression technique, the best fit line is not a straight line. It is rather a curve that fits into the data points.



Important Points:

- While there might be a temptation to fit a higher degree polynomial to get a lower error, this can result in over-fitting. Always plot the relationships to see the fit and focus on making sure that the

curve fits the nature of the problem. Here is an example of how plotting can help:



- Especially look out for curve towards the ends and see whether those shapes and trends make sense. Higher polynomials can end up producing weird results on extrapolation.

4. Stepwise Regression

This form of regression is used when we deal with multiple independent variables. In this regression technique in Machine Learning, the selection of independent variables is done with the help of an automatic process, which involves no human intervention.

This feat is achieved by observing statistical values like R-square, t-stats and AIC metric to discern significant variables. Stepwise regression fits the regression model by adding/dropping covariates one at a time based on a specified criterion. Some of the most commonly used Stepwise regression methods are listed below:

- Standard stepwise regression does two things. It adds and removes predictors as needed for each step.
- Forward selection starts with the most significant predictor in the model and adds variable for each step.
- Backward elimination starts with all predictors in the model and removes the least significant variable for each step.

This modeling technique aims to maximize the prediction power with a minimum number of predictor variables. It is one of the methods to handle higher dimensionality of data set.

10	A	Establish the Association rules in unsupervised learning.	**[L3][CO2]**	**[6M]**

ASSOCIATION RULES:

Association rule learning is a kind of unsupervised learning technique that tests for the reliance of one data element on another data element and design appropriately so that it can be more cost-effective. It tries to discover some interesting relations or associations between the variables of the dataset. It depends on various rules to find interesting relations between variables in the database.

The association rule learning is the most important approach of machine learning, and it is employed in Market Basket analysis, Web usage mining, continuous production, etc. In market basket analysis, it is an approach used by several big retailers to find the relations between items.

Types of Association Rule Learning

There are the following types of Association rule learning which are as follows −

Apriori Algorithm − This algorithm needs frequent datasets to produce association rules. It is designed to work on databases that include transactions. This algorithm needs a breadth-first search and hash tree to compute the itemset efficiently.

It is generally used for market basket analysis and support to learn the products that can be purchased together. It can be used in the healthcare area to discover drug reactions for patients.

Eclat Algorithm − The Eclat algorithm represents Equivalence Class Transformation. This algorithm needs a depth-first search method to discover frequent itemsets in a transaction database. It implements quicker execution than Apriori Algorithm.

F-P Growth Algorithm − The F-P growth algorithm represents Frequent Pattern. It is the enhanced version of the Apriori Algorithm. It describes the database in the form of a tree structure that is referred to as a frequent pattern or tree. This frequent tree aims to extract the most frequent patterns.

There are various applications of Association Rule which are as follows −

- Items purchased on a credit card, such as rental cars and hotel rooms, support insight into the following product that customer are likely to buy.
- Optional services purchased by tele-connection users (call waiting, call forwarding, DSL, speed call, etc.) support decide how to bundle these functions to maximize revenue.
- Banking services used by retail users (money industry accounts, CDs, investment services, car loans, etc.) recognize users likely to needed other services.
- Unusual group of insurance claims can be an expression of fraud and can spark higher investigation.

Medical patient histories can supports expressions of likely complications based on definite set of treatments.

b	Analyze the real world applications of ML.	**[L4][CO6]**	**[6M]**

Machine learning has found numerous applications across various industries, revolutionizing processes and enabling the development of innovative solutions. Here are some real-world applications of machine learning:

Healthcare: Machine learning is used for medical diagnosis, patient monitoring, and treatment planning. It can analyze medical records, images, and genomic data to assist in early disease detection, personalized medicine, and predicting patient outcomes. Machine learning models can also help identify patterns and anomalies in large healthcare datasets for improved decision-making.

Finance: Machine learning is widely applied in financial institutions for fraud detection, credit scoring, algorithmic trading, and risk assessment. It can analyze vast amounts of financial data to identify fraudulent transactions, predict market trends, and optimize investment strategies. Machine learning models are also used for automated trading based on historical and real-time market data.

Retail and E-commerce: Machine learning is used for personalized recommendations, demand forecasting, inventory management, and pricing optimization. By analyzing customer behavior, browsing history, and purchase patterns, machine learning models can recommend relevant products to users, optimize pricing strategies, and predict customer preferences to improve sales and customer satisfaction.

Transportation and Logistics: Machine learning is utilized for route optimization, demand forecasting, and predictive maintenance in transportation and logistics. It can analyze historical data, real-time traffic information, and weather conditions to optimize routes for delivery vehicles, forecast demand for transportation services, and detect anomalies in equipment performance to prevent breakdowns.

Manufacturing: Machine learning is used in manufacturing industries for quality control, predictive maintenance, and process optimization. It can analyze sensor data from production lines to detect anomalies and ensure product quality. Machine learning models can also predict equipment failures, enabling proactive maintenance to minimize downtime and maximize productivity.

Natural Language Processing (NLP): Machine learning techniques are applied in NLP applications such as language translation, sentiment analysis, chatbots, and voice assistants. NLP models can understand and generate human language, enabling accurate translation between languages, sentiment analysis of customer feedback, and interactive conversational experiences.

Autonomous Vehicles: Machine learning plays a crucial role in autonomous vehicles by enabling object detection and recognition, scene understanding, and decision-making. Machine learning models process sensor data from cameras, LiDAR, and radar to detect and classify objects on the road, navigate complex environments, and make real-time decisions to ensure safe driving.

Energy and Utilities: Machine learning is used for energy load forecasting, anomaly detection in power grids, and optimizing energy consumption. It can analyze historical energy consumption data, weather conditions, and other factors to predict future energy demand and optimize energy generation and distribution.

These are just a few examples of the vast range of real-world applications of machine learning. The versatility and potential of machine learning continue to expand, with ongoing research and development pushing the boundaries of what is possible in various industries and domains.

Course Code: 20CS0906 **R20**

UNIT-II

SUPERVISED LEARNING

1	a	Differentiate Supervised Learning and Unsupervised Learning	**[L4][CO5]**	**[5M]**

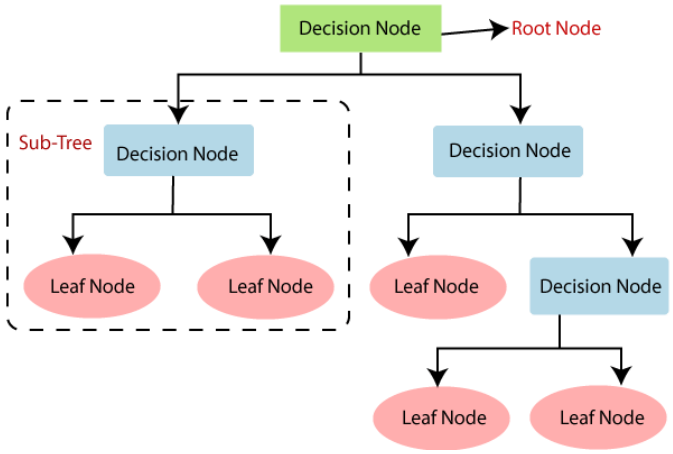
Supervised Learning	**Unsupervised Learning**
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in **Classification** and **Regression** problems.	Unsupervised Learning can be classified in **Clustering** and **Associations** problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

	b	Explain Decision Tree Classification technique with an example.	**[L2][CO6]**	**[7M]**

- 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.
- 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.

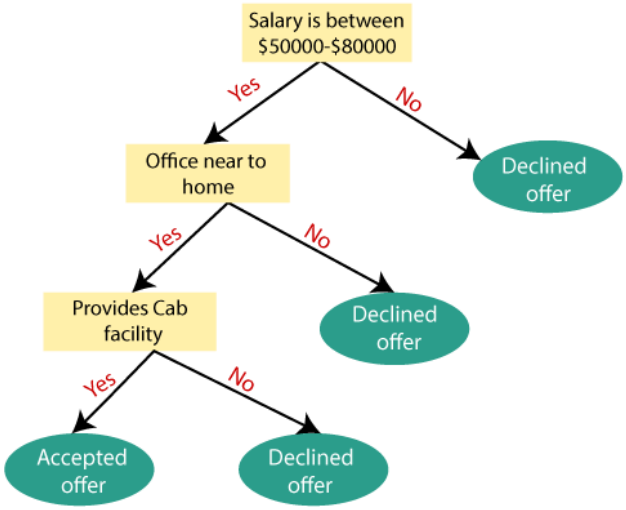


Decision Tree Terminologies

- **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.

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.



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:

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- ∑jPj2

2	a	Describe classification techniques in supervised learning.	**[L2][CO1]**	**[6M]**

Classification is a technique for determining which class the dependent belongs to based on one or

more independent variables.A classifier is a type of machine learning algorithm that assigns a label to a data input. Classifier algorithms use labeled data and statistical methods to produce predictions about data input classifications.

There are different types of Classification techniques used in Supervised learning.

- Logistic Regression
- K-Nearest Neighbor
- Support Vector Machine
- Kernel SVM

Radial Basis Function

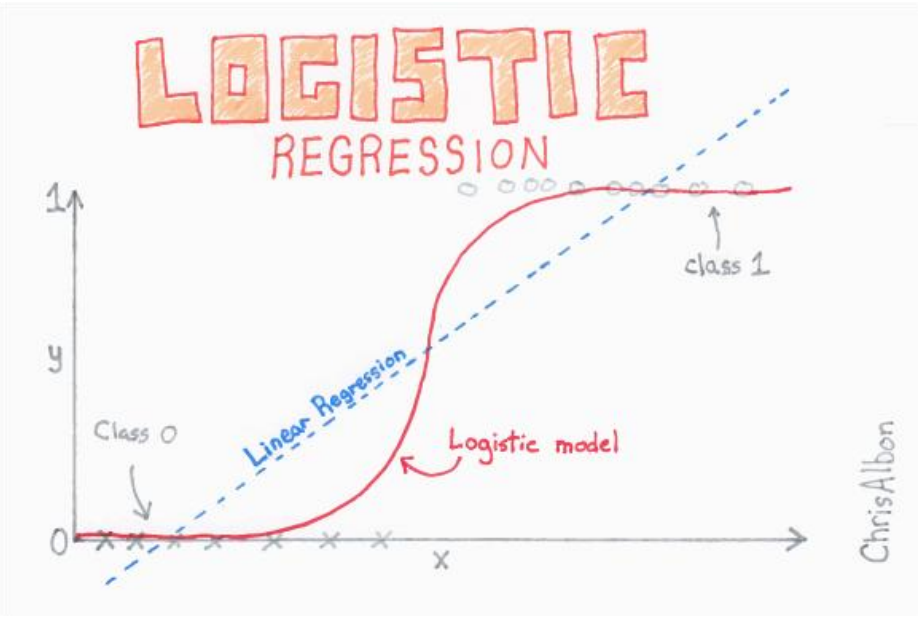
- Naïve Bayes
- Decision Tree Classification

Ensemble Methods for Classification:

3. Random Forest Classification
4. Gradient Boosting Classification

1. LOGISTIC REGRESSION:

Logistic regression is kind of like linear regression, but is used when the dependent variable is not a number but something else (e.g., a "yes/no" response). It's called regression but performs classification based on the regression and it classifies the dependent variable into either of the classes.



Logistic regression is used for prediction of output which is binary, as stated above. For example, if a credit card company builds a model to decide whether or not to issue a credit card to a customer, it will model for whether the customer is going to "default" or "not default" on their card.

$$y = b_0 + b_1 x$$

Linear Regression

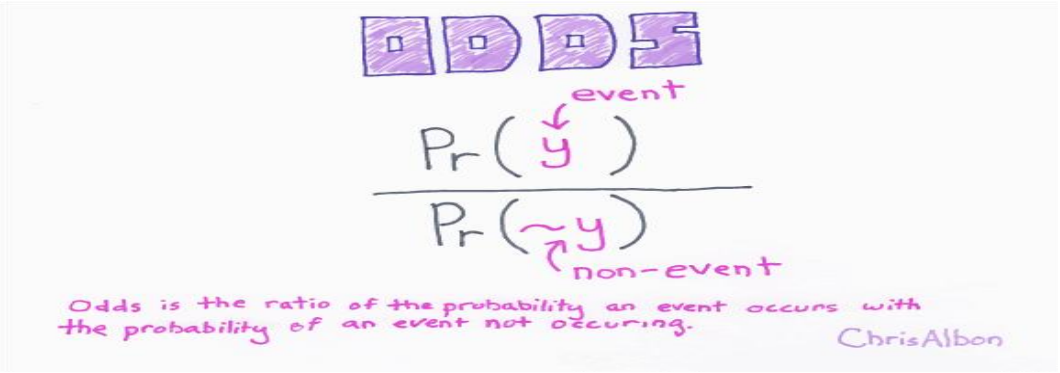
Firstly, linear regression is performed on the relationship between variables to get the model. The threshold for the classification line is assumed to be at 0.5.

$$p = \frac{1}{1 + e^{-y}}$$

Logistic Sigmoid Function

Logistic function is applied to the regression to get the probabilities of it belonging in either class.

It gives the log of the probability of the event occurring to the log of the probability of it not occurring. In the end, it classifies the variable based on the higher probability of either class.



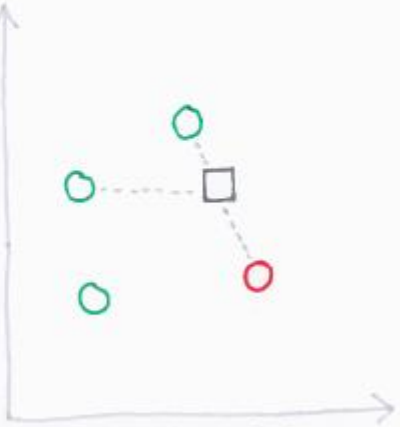
2. K-NEAREST NEIGHBORS (K-NN)

K-NN algorithm is one of the simplest classification algorithms and it is used to identify the data points that are separated into several classes to predict the classification of a new sample point. K-NN is a non-parametric**,** lazy learning algorithm. It classifies new cases based on a similarity measure (i.e., distance functions).

K-NEAREST NEIGHBORS

- K is the number of neighbors to consider.
- Scaling is important.
- K should be odd.
- If we have binary features we can use Hamming distance.
- Voting can be weighted by distance to each neighbor.
- Does not scale to large data well.

If k=3, the grey square observation is predicted to be green because two of its neighbors are green and only one is red.



Chris Albon

DOES K-NN LEARN

K-nearest neighor does not "learn" per-se. It is lazy and just memorizes the data.

Chris Albon

K-NEAREST NEIGHBORS TIPS AND TRICKS

1. All features should use the same scale.
2. K should be odd to avoid ties.
3. Votes can be weighted by the distance to the neighbor so closer observations' votes are worth more.
4. Try a variety of distance measurements.

Chris Albon

KNN NEIGHBORHOOD SIZE

Small ↓ K = Low Bias, High Variance

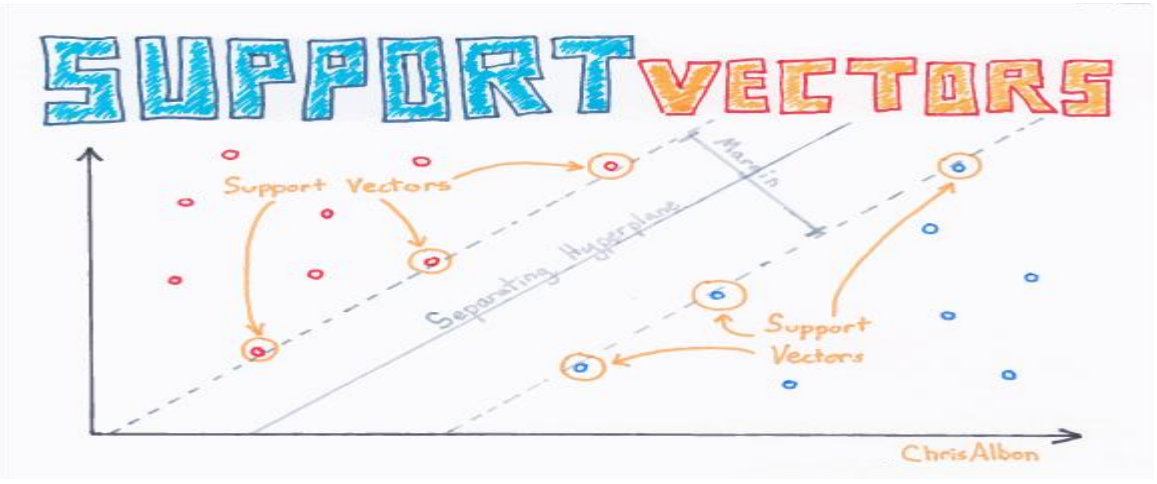
↑ LARGE K = High Bias, Low Variance

BY CHRIS ALBON

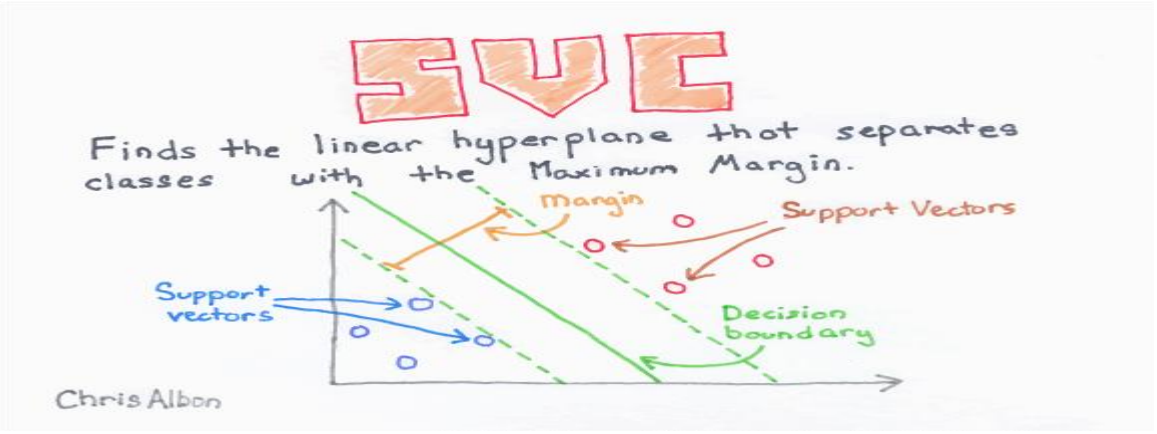
K-NN works well with a small number of input variables (*p*), but struggles when the number of inputs is very large.

3. SUPPORT VECTOR MACHINE (SVM)

Support vector is used for both regression and classification. It is based on the concept of decision planes that define decision boundaries. A decision plane (hyperplane) is one that separates between a set of objects having different class memberships.



It performs classification by finding the hyperplane that maximizes the margin between the two classes with the help of support vectors.



The learning of the hyperplane in SVM is done by transforming the problem using some linear algebra (i.e., the example above is a linear kernel which has a linear separability between each variable).

For higher dimensional data, other kernels are used as points and cannot be classified easily. They are specified in the next section.

Kernel SVM

Kernel SVM takes in a kernel function in the SVM algorithm and transforms it into the required

form that maps data on a higher dimension which is separable.Types of kernel functions:

$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma \mid \mathbf{X}_i - \mathbf{X}_j \mid^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{cases}$$

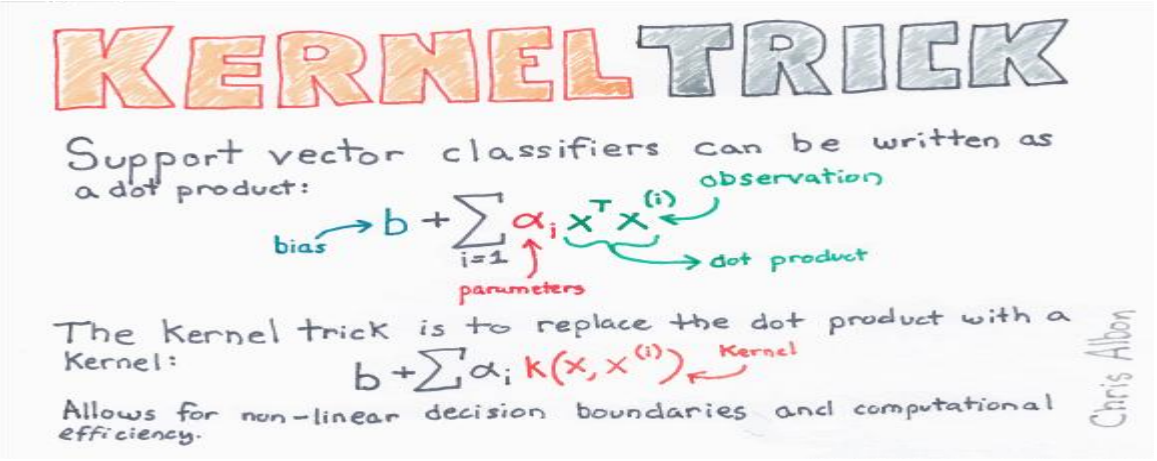
Type of kernel functions

Linear SVM is the one we discussed earlier.

In polynomial kernel, the degree of the polynomial should be specified. It allows for curved lines in the input space.

In the radial basis function (RBF) kernel, it is used for non-linearly separable variables. For distance, metric squared Euclidean distance is used. Using a typical value of the parameter can lead to overfitting our data. It is used by default in sklearn.

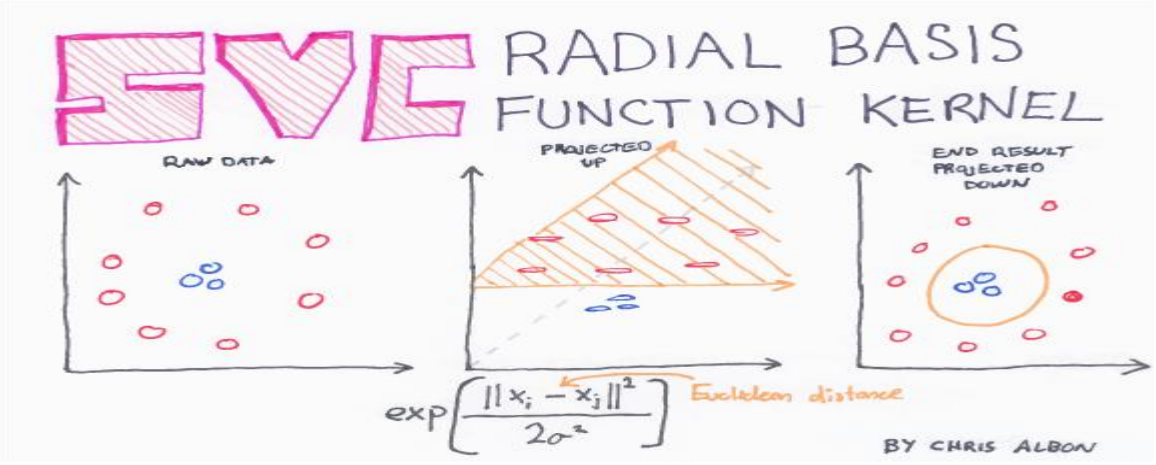
Sigmoid kernel, similar to logistic regression is used for binary classification.



Kernel trick uses the kernel function to transform data into a higher dimensional feature space and makes it possible to perform the linear separation for classification.

Radial Basis Function (RBF) Kernel

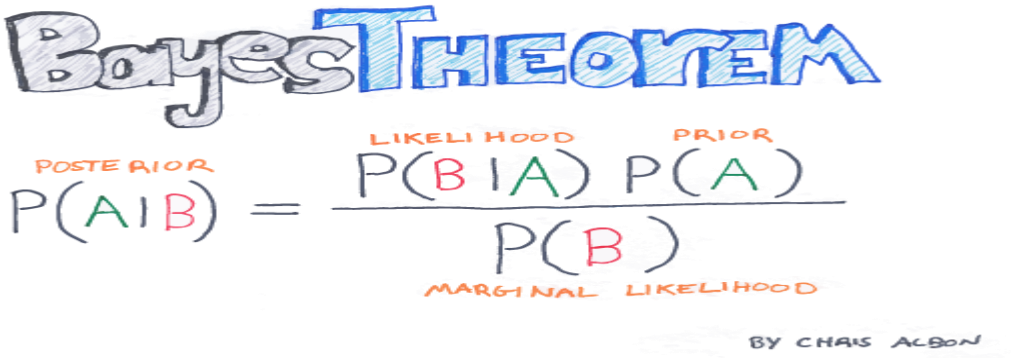
The RBF kernel SVM decision region is actually also a linear decision region. What RBF kernel SVM actually does is create non-linear combinations of features to uplift the samples onto a higher-dimensional feature space where a linear decision boundary can be used to separate classes.



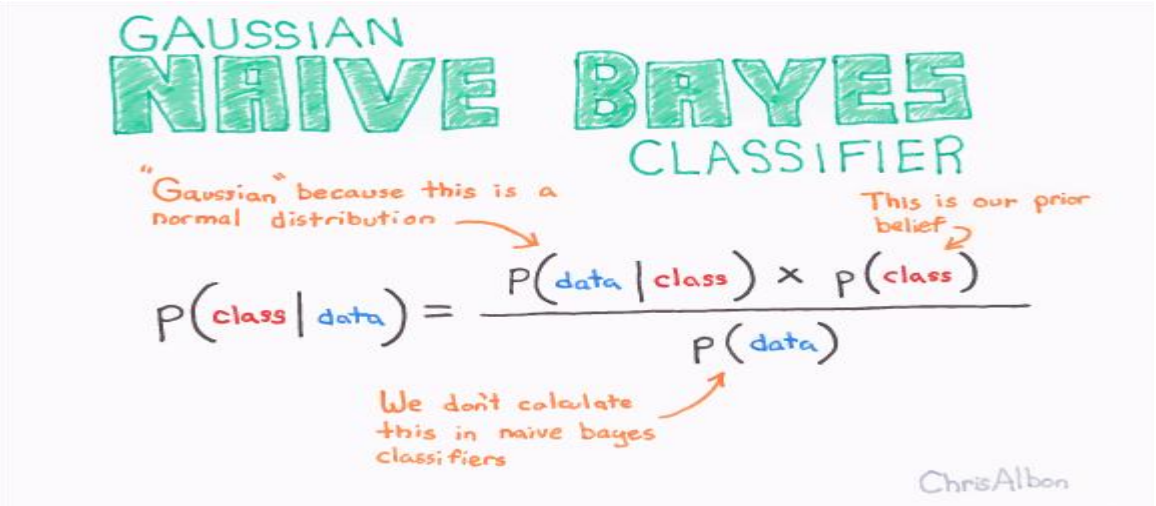
So, the rule of thumb is: use linear SVMs for linear problems, and nonlinear kernels such as the RBF kernel for non-linear problems.

4. NAIVE BAYES

The naive Bayes classifier is based on Bayes' theorem with the independence assumptions between predictors (i.e., it assumes the presence of a feature in a class is unrelated to any other feature). Even if these features depend on each other, or upon the existence of the other features, all of these properties independently. Thus, the name naive Bayes.



Based on naive Bayes, Gaussian naive Bayes is used for classification based on the binomial (normal) distribution of data.

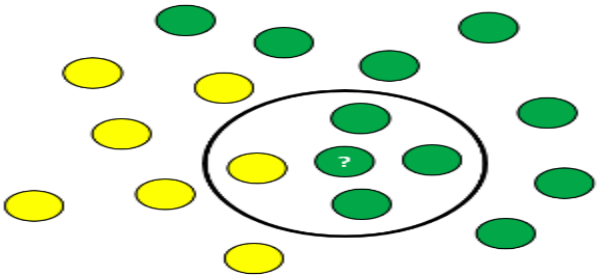


P(class|data) is the posterior probability of *class*(*target*) given *predictor*(*attribute*). The probability of a data point having either class, given the data point. This is the value that we are looking to calculate.

P(class) is the prior probability of *class*.

P(data|class) is the likelihood, which is the probability of *predictor* given *class*.

P(data) is the prior probability of *predictor or marginal likelihood*.



NB Classification Example

Naive Bayes Steps

1. Calculate Prior Probability

P(class) = Number of data points in the class/Total no. of observations

P(yellow) = 10/17

P(green) = 7/17

2. Calculate Marginal Likelihood

P(data) = Number of data points similar to observation/Total no. of observations

P(?) = 4/17

The value is present in checking both the probabilities.

3. Calculate Likelihood

P(data/class) = Number of similar observations to the class/Total no. of points in the class.

P(?/yellow) = 1/7

P(?/green) = 3/10

11. Posterior Probability for Each Class

$$p(class/data) = \frac{P(data/class) * P(class)}{P(data)}$$

$$P(yellow/?) = \frac{1/7 * 7/17}{4/17} = 0.25$$

$$P(green/?) = \frac{3/10 * 10/17}{4/17} = 0.75$$

12. Classification:

$$P(class1/data) > P(class2/data)$$

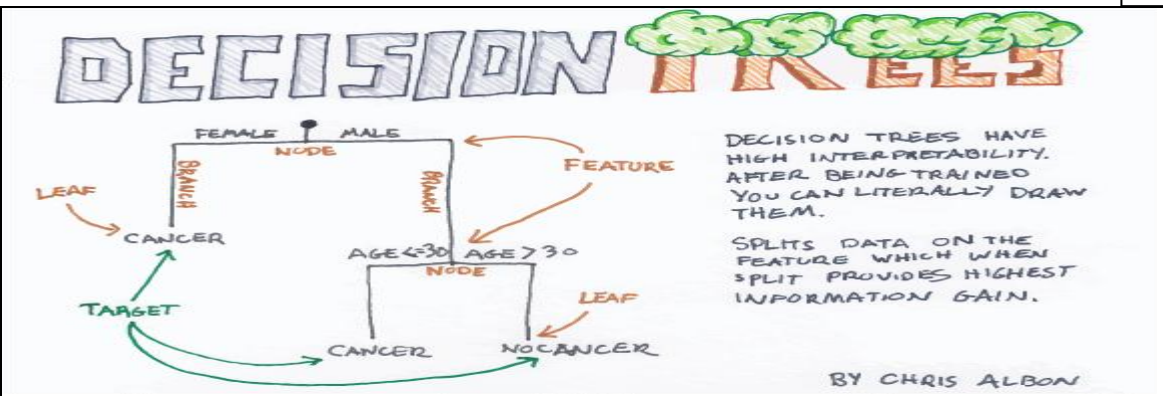
$$P(green/?) > P(yellow/?)$$

The higher probability, the class belongs to that category as from above 75% probability the point belongs to class green.

Multinomial, Bernoulli naive Bayes are the other models used in calculating probabilities. Thus, a naive Bayes model is easy to build, with no complicated iterative parameter estimation, which makes it particularly useful for very large datasets.

5. DECISION TREE CLASSIFICATION

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. It follows Iterative Dichotomiser 3 (ID3) algorithm structure for determining the split.



Entropy and information gain are used to construct a decision tree.

Entropy

Entropy is the degree or amount of uncertainty in the randomness of elements. In other words, it is a measure of impurity**.**

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Entropy

Intuitively, it tells us about the predictability of a certain event. Entropy calculates the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero, and if the sample is equally divided it has an entropy of one.

Information Gain

Information gain measures the relative change in entropy with respect to the independent attribute. It tries to estimate the information contained by each attribute. Constructing a decision tree is all about finding the attribute that returns the highest information gain (i.e., the most homogeneous branches).

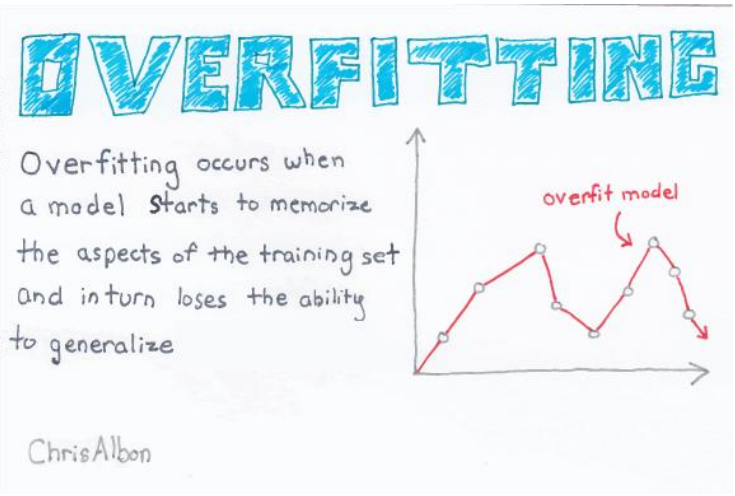
$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

Where *Gain(T, X)* is the information gain by applying feature *X*. *Entropy(T)* is the entropy of the entire set, while the second term calculates the entropy after applying the feature *X*.

Information gain ranks attributes for filtering at a given node in the tree. The ranking is based on

the highest information gain entropy in each split.

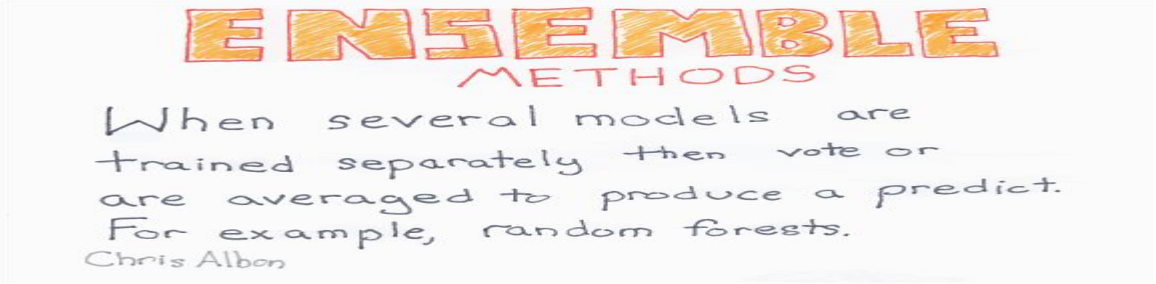
The disadvantage of a decision tree model is overfitting, as it tries to fit the model by going deeper in the training set and thereby reducing test accuracy.



Overfitting in decision trees can be minimized by pruning nodes.

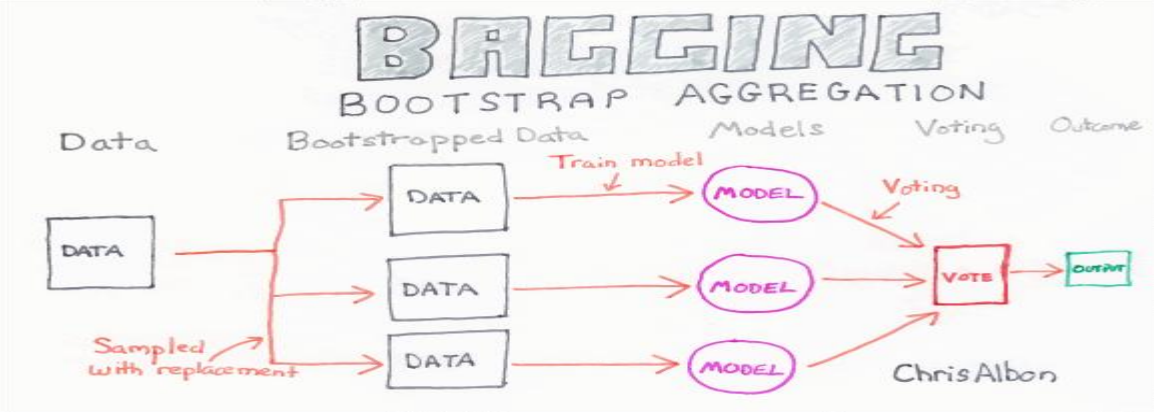
Ensemble Methods for Classification

An ensemble model is a *team of models*. Technically, ensemble models comprise several supervised learning models that are individually trained and the results merged in various ways to achieve the final prediction. This result has higher predictive power than the results of any of its constituting learning algorithms independently.

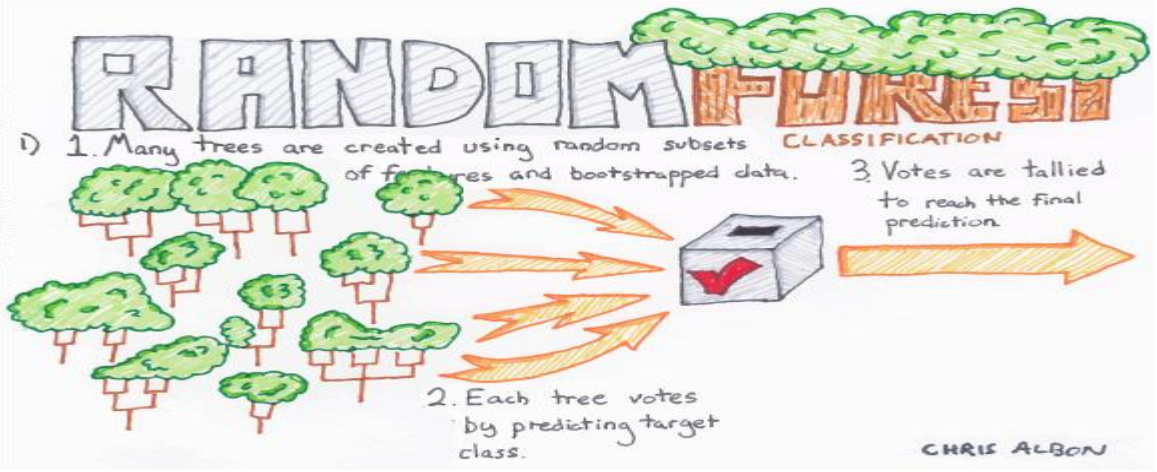


1. RANDOM FOREST CLASSIFICATION

Random forest classifier is an ensemble algorithm based on bagging i.e bootstrap aggregation. Ensemble methods combines more than one algorithm of the same or different kind for classifying objects (i.e., an ensemble of SVM, naive Bayes or decision trees, for example.)



The general idea is that a combination of learning models increases the overall result selected.



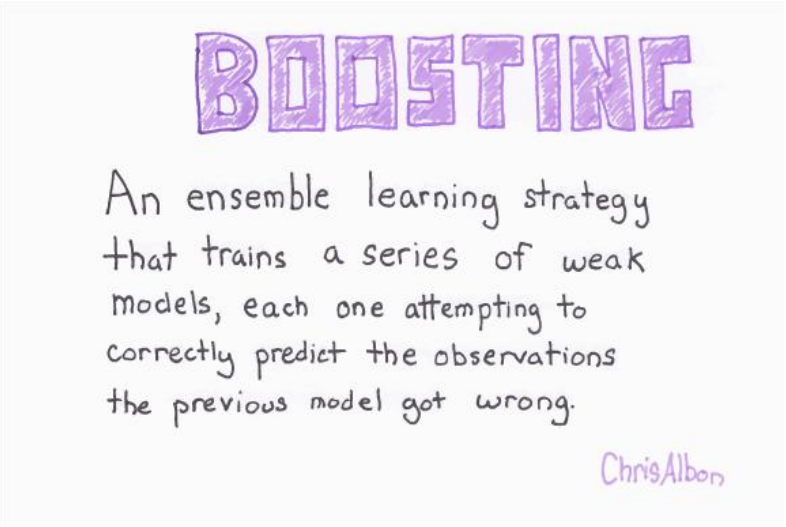
Deep decision trees may suffer from overfitting, but random forests prevent overfitting by creating trees on random subsets. The main reason is that it takes the average of all the predictions, which cancels out the biases.

Random forest adds additional randomness to the model while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

2. GRADIENT BOOSTING CLASSIFICATION

Gradient boosting classifier is a boosting ensemble method. Boosting is a way to combine (ensemble) weak learners, primarily to reduce prediction bias. Instead of creating a pool of predictors, as in bagging, boosting produces a cascade of them, where each output is the input for the following learner. Typically, in a bagging algorithm trees are grown in parallel to get the average prediction across all trees, where each tree is built on a sample of original data. Gradient boosting, on the other hand, takes a sequential approach to obtaining predictions instead of parallelizing the tree building process. In gradient boosting, each decision tree predicts the error of

the previous decision tree—thereby *boosting* (improving) the error (gradient).



Working of Gradient Boosting

Initialize predictions with a simple decision tree.

Calculate residual (actual-prediction) value.

Build another shallow decision tree that predicts residual based on all the independent values.

Update the original prediction with the new prediction multiplied by learning rate.

Repeat steps two through four for a certain number of iterations (the number of iterations will be the number of trees).

b	List out various Regression techniques in Machine Learning.	**[L1][CO1]**	**[6M]**

Regression techniques:

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Logistic regression
- Polynomial Regression
- Stepwise Regression

1. Linear Regression

It is one of the most widely known modeling techniques and the most famous regression technique in Machine Learning. Linear regression is usually among the first few topics which people pick while learning predictive modeling. In this technique, the dependent variable is continuous, the independent variable(s) can be continuous or discrete, and the nature of the regression line is linear.

Linear Regression establishes a relationship between the dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as Regression line).

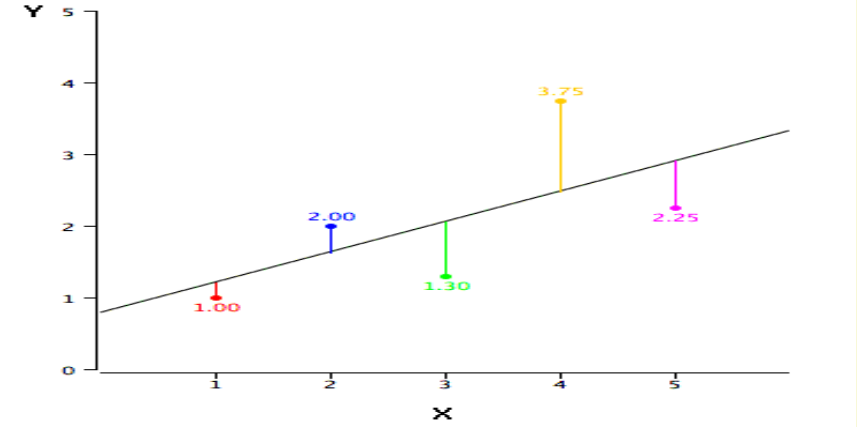
It is represented by an equation Y=a+b*X + e, where a is the intercept, b is the slope of the line

and e is error term. This equation can be used to predict the value of the target variable based on the given predictor variable(s).

The difference between simple linear regression and multiple linear regression, multiple linear regression has (>1) independent variables, whereas simple linear regression has only 1 independent variable. Now, the question is "How do we obtain best-fit line?".

How to obtain the best fit line (Value of a and b)?

This task can be easily accomplished by Least Square Method. It is the most common method used for fitting a regression line. It calculates the best-fit line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line. Because the deviations are first squared, when added, there is no canceling out between positive and negative values.



We can evaluate the model performance using the metric R-square.

2. Logistic Regression

Logistic regression in Machine Learning is used to find the probability of event=Success and event=Failure. We should use logistic regression when the dependent variable is binary (0/ 1, True/ False, Yes/ No) in nature. Here the value of Y ranges from 0 to 1 and it can be represented by the following equation.

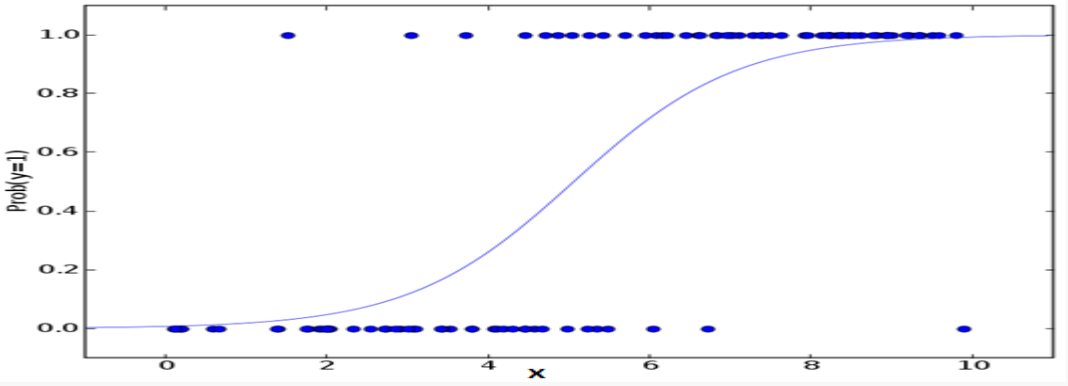
odds= p/ (1-p) = probability of event occurrence / probability of not event occurrence

ln(odds) = ln(p/(1-p))

logit(p) = ln(p/(1-p)) = b0+b1X1+b2X2+b3X3....+bkXk

Above, p is the probability of the presence of the characteristic of interest. A question that you should ask here is "why have we used to log in the equation?".

Since we are working here with a binomial distribution (dependent variable), we need to choose a link function which is best suited for this distribution. And, it is a logit function. In the equation above, the parameters are chosen to maximize the likelihood of observing the sample values rather than minimizing the sum of squared errors (like in ordinary regression).



Important Points:

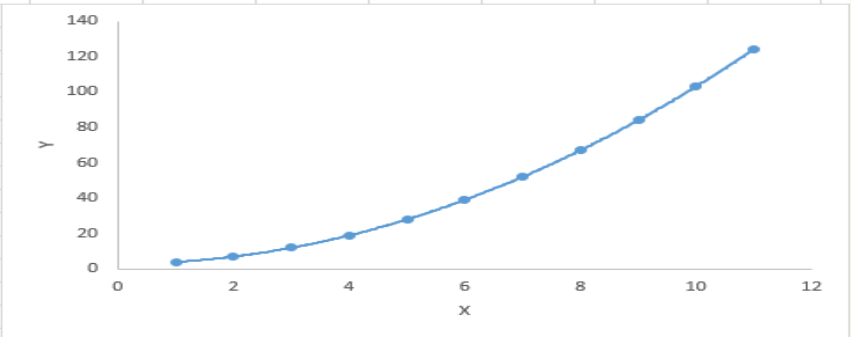
- It is widely used for classification problems
- Logistic regression doesn't require a linear relationship between dependent and independent variables. It can handle various types of relationships because it applies a non-linear log transformation to the predicted odds ratio
- To avoid over fitting and under fitting, we should include all significant variables. A good approach to ensure this practice is to use a stepwise method to estimate the logistic regression.
- It requires large sample sizes because maximum likelihood estimates are less powerful at low sample sizes than ordinary least square.
- The independent variables should not be correlated with each other i.e. no multicollinearity. However, we have the option to include interaction effects of categorical variables in the analysis and the model.
- If the values of the dependent variable are ordinal, then it is called as Ordinal logistic regression.
- If the dependent variable is multi-class then it is known as Multinomial Logistic regression.

3. Polynomial Regression

A regression equation in Machine Learning is a polynomial regression equation if the power of the independent variable is more than 1. The equation below represents a polynomial equation:

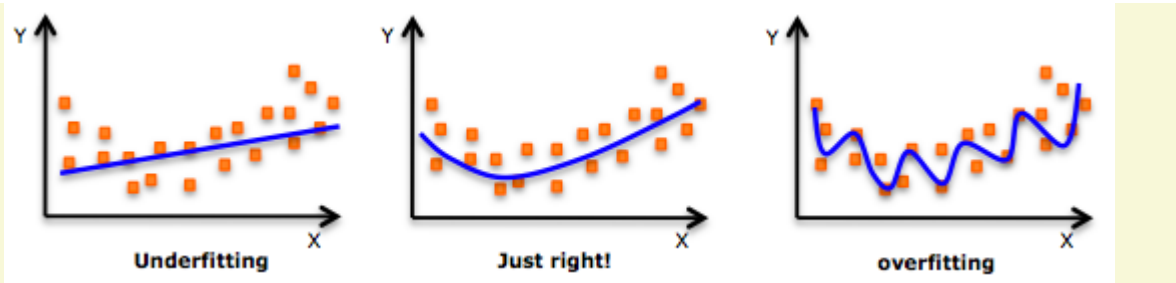
y=a+b*x^2

In this regression technique, the best fit line is not a straight line. It is rather a curve that fits into the data points.



Important Points:

- While there might be a temptation to fit a higher degree polynomial to get a lower error, this can result in over-fitting. Always plot the relationships to see the fit and focus on making sure that the curve fits the nature of the problem. Here is an example of how plotting can help:



- Especially look out for curve towards the ends and see whether those shapes and trends make sense. Higher polynomials can end up producing weird results on extrapolation.

4. Stepwise Regression

This form of regression is used when we deal with multiple independent variables. In this

regression technique in Machine Learning, the selection of independent variables is done with the help of an automatic process, which involves no human intervention.

This feat is achieved by observing statistical values like R-square, t-stats and AIC metric to discern significant variables. Stepwise regression fits the regression model by adding/dropping covariates one at a time based on a specified criterion. Some of the most commonly used Stepwise regression methods are listed below:

- Standard stepwise regression does two things. It adds and removes predictors as needed for each step.
- Forward selection starts with the most significant predictor in the model and adds variable for each step.
- Backward elimination starts with all predictors in the model and removes the least significant variable for each step.

This modeling technique aims to maximize the prediction power with a minimum number of predictor variables. It is one of the methods to handle higher dimensionality of data set.

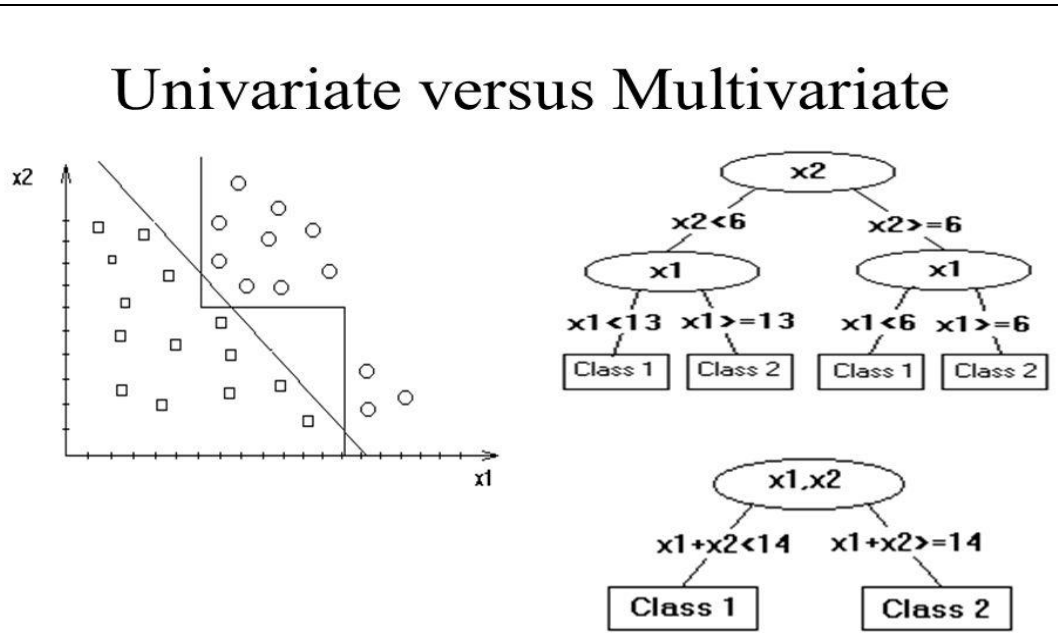
3	a	Compare Univariate and Multivariate Decision Trees.	**[L5][CO1]**	**[6M]**

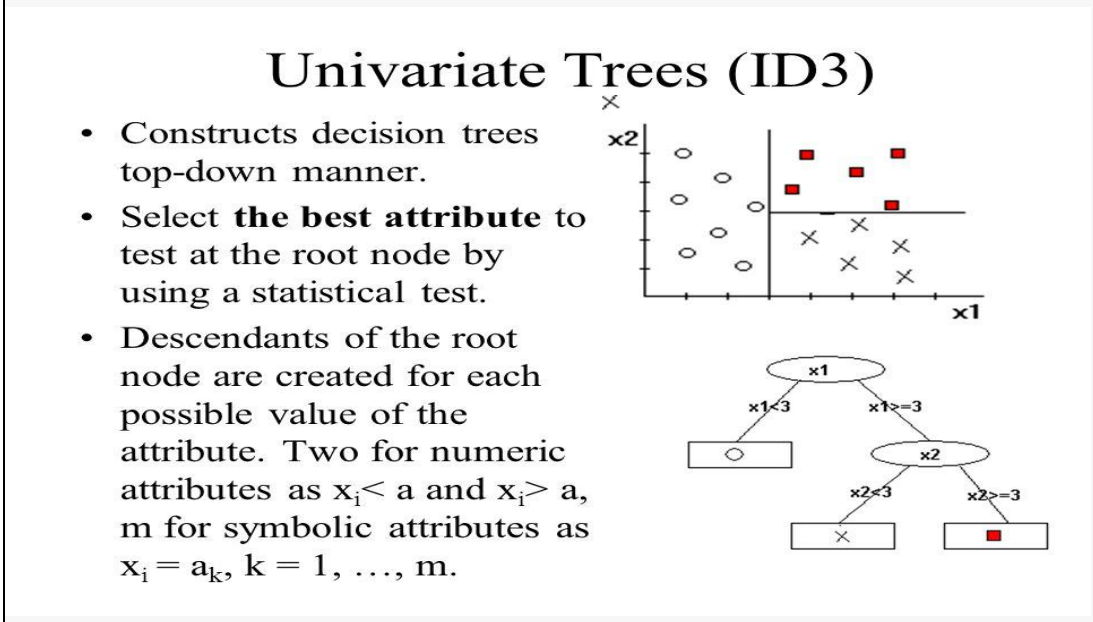
Univariate Tree: A univariate tree, also known as a decision tree, is a predictive model that uses a tree-like structure to make predictions or decisions based on a single input variable (feature). It is a supervised learning algorithm commonly used for both classification and regression tasks. In a univariate tree, the tree structure is built by recursively partitioning the data based on the values of a single feature at each internal node.

The decision tree starts with the entire dataset at the root node and selects the best feature to split the data based on certain criteria (e.g., information gain or Gini index). The data is then divided into subsets based on the feature value, and the process is repeated recursively for each subset until a stopping condition is met, such as reaching a maximum tree depth or having a minimum number of samples at a node. The leaf nodes of the tree contain the predicted outcomes or values.

Multivariate Tree: A multivariate tree, also known as a random forest or ensemble tree, is an extension of the univariate tree that uses multiple input variables (features) to make predictions. It combines the predictions of multiple univariate trees to improve the overall accuracy and robustness of the model. A multivariate tree is typically used for classification and regression tasks.

Instead of using a single feature at each internal node, a multivariate tree randomly selects a subset of features and builds univariate trees using these selected features. The number of features sampled at each node and the number of trees in the forest are hyperparameters that can be adjusted. During prediction, each tree in the forest independently makes a prediction, and the final prediction is determined by aggregating the individual tree predictions, such as taking a majority vote for classification tasks or averaging for regression tasks.





The use of multiple features and trees in a multivariate tree helps to capture more complex relationships and reduces the risk of overfitting. It can handle high-dimensional datasets and provide better generalization performance compared to a single univariate tree.

b	Explain about Pruning in supervised learning.	**[L2][CO1]**	**[6M]**

Pruning is a technique used in decision trees to reduce overfitting and improve the generalization performance of the model. Overfitting occurs when a decision tree becomes too complex and captures noise or irrelevant details from the training data, which can lead to poor performance on unseen data.

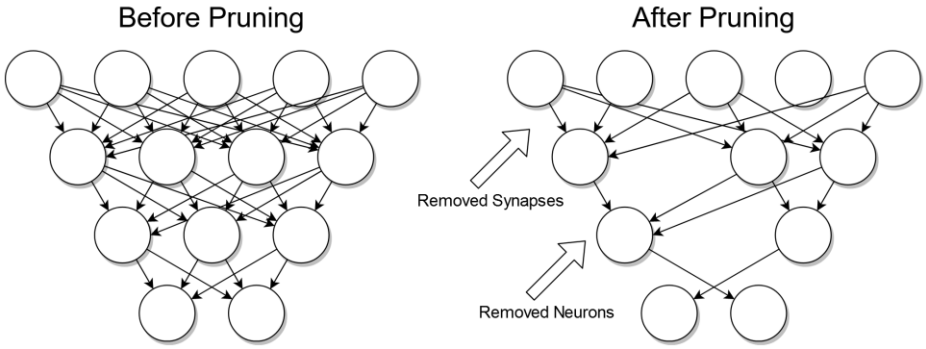
Pruning involves the process of removing branches or nodes from a decision tree to simplify its structure and make it more general. This is typically done by setting certain conditions or criteria that determine when and how to prune the tree. There are two main types of pruning techniques:

Pre-Pruning (Early Stopping): Pre-pruning involves stopping the growth of the tree before it becomes fully expanded. This is usually done by setting stopping criteria based on various measures such as maximum tree depth, minimum number of samples required at a node,

minimum improvement in impurity measures (e.g., information gain or Gini index), or other statistical significance tests. If a node does not meet these criteria, it is considered a leaf node and no further splitting is performed.

Post-Pruning (Cost Complexity Pruning): Post-pruning involves growing the tree to its full size and then selectively removing branches or nodes based on their estimated predictive ability. This is done by assigning a cost or penalty to each node based on measures like impurity or error rate. A complexity parameter, such as the cost complexity parameter or pruning parameter, is used to control the trade-off between simplicity and accuracy. By iteratively removing nodes with the highest cost, the tree is pruned to a more optimal size that balances complexity and performance.

The goal of pruning is to find the right balance between complexity and generalization. By reducing the complexity of the decision tree, pruning helps to avoid overfitting and improves the model's ability to generalize well to unseen data. Pruning is an essential step in decision tree construction, especially when dealing with complex datasets or when the decision tree grows too large.



4	a	Differentiate various Parametric and Non-Parametric Methods.	**[L4][CO1]**	**[6M]**

Parametric Methods	**Non-Parametric Methods**
Parametric Methods uses a fixed number of parameters to build the model.	Non-Parametric Methods use the flexible number of parameters to build the model.
Parametric analysis is to test group means.	A non-parametric analysis is to test medians.
It is applicable only for variables.	It is applicable for both – Variable and Attribute.
It always considers strong assumptions about data.	It generally fewer assumptions about data.
Parametric Methods require lesser data than Non-Parametric Methods.	Non-Parametric Methods requires much more data than Parametric Methods.
Parametric methods assumed to be a normal distribution.	There is no assumed distribution in non-parametric methods.

ratio data.	data.
Here when we use parametric methods then the result or outputs generated can be easily affected by outliers.	When we use non-parametric methods then the result or outputs generated cannot be seriously affected by outliers.
Parametric Methods can perform well in many situations but its performance is at peak (top) when the spread of each group is different.	Similarly, Non-Parametric Methods can perform well in many situations but its performance is at peak (top) when the spread of each group is the same.
Parametric methods have more statistical power than Non-Parametric methods.	Non-parametric methods have less statistical power than Parametric methods.
As far as the computation is considered these methods are computationally faster than the Non-Parametric methods.	As far as the computation is considered these methods are computationally slower than the Parametric methods.
Examples: Logistic Regression, Naïve Bayes Model, etc.	Examples: KNN, Decision Tree Model, etc.

b	Analyze Bayesian Decision theory in supervised learning.	**[L4][CO1]**	**[6M]**

Bayesian decision theory is a framework in supervised learning that allows us to make optimal decisions under uncertainty by incorporating prior knowledge and probabilities. It provides a principled approach for decision-making based on the Bayesian principles of probability theory. Here's an analysis of Bayesian decision theory in supervised learning:

Probabilistic Modeling:
Bayesian decision theory starts with the assumption that the underlying data distribution and the relationships between inputs and outputs are probabilistic in nature.
It involves modeling the joint probability distribution of the input features (X) and the corresponding output labels (Y) using techniques such as Bayesian networks, Gaussian processes, or probabilistic graphical models.

Prior Knowledge:
Bayesian decision theory incorporates prior knowledge or beliefs about the data before observing any new instances. This prior information is usually expressed through prior probability distributions or prior assumptions about the parameters of the model.

Likelihood Estimation:
Given the observed data, Bayesian decision theory aims to estimate the likelihood of different classes or labels given the input features.
The likelihood is computed based on the probabilistic model and the observed data, using techniques such as maximum likelihood estimation or Bayesian inference.

Bayesian Inference:
Bayesian decision theory leverages Bayesian inference to update the prior knowledge based on the observed data and compute posterior probabilities.
The posterior probabilities represent the updated belief about the class labels given the observed data and are computed using Bayes' theorem.

Decision Rule:
Once the posterior probabilities are obtained, a decision rule is applied to make predictions or decisions.
The decision rule can be based on maximizing the posterior probability (maximum a posteriori estimation), or it can consider various loss functions or utility functions to minimize the expected loss or maximize expected utility.

Decision Boundary:
Bayesian decision theory provides a framework for defining decision boundaries that separate different

classes based on the posterior probabilities.
The decision boundaries can be determined by setting thresholds on the posterior probabilities or by considering the costs associated with different misclassifications.

Optimal Decision-Making:

Bayesian decision theory aims to make decisions that minimize the expected loss or maximize the expected utility, considering the posterior probabilities and the decision rule.
This allows for optimal decision-making under uncertainty, taking into account both the prior knowledge and the observed data.

Bayesian decision theory provides a coherent and principled approach to supervised learning by incorporating probabilistic modeling, Bayesian inference, and decision theory. It allows for the integration of prior knowledge and uncertainty, leading to robust and optimal decision-making in various domains.

Bayesian Decision Theory is a fundamental statistical approach to the problem of pattern classification. It is considered as the ideal pattern classifier and often used as the benchmark for other algorithms because its decision rule automatically minimizes its loss function. It might not make much sense right now, so hold on, we'll unravel it all.

It makes the assumption that the decision problem is posed in probabilistic terms, and that all the relevant probability values are known.

Bayes' Theorem

Derivation of Bayes' Theorem:
We know from the conditional probability:
$P(A|B) = P(A, B) / P(B)$
$\Rightarrow P(A, B) = P(A|B) * P(B)$ ... (i)Similarly,
$P(A, B) = P(B|A) * P(A)$ ... (ii)From equation (i) and (ii):
$P(A|B) * P(B) = P(B|A) * P(A)$
$\Rightarrow P(A|B) = [P(B|A) * P(A)] / P(B)$
For the case of classification, let:

- $A \equiv \omega$ (state of the nature or the class of an entry)
- $B \equiv \mathbf{x}$ (input feature vector)

After substituting we get:
$P(\omega|\mathbf{x}) = [P(\mathbf{x}|\omega) * P(\omega)] / P(\mathbf{x})$which becomes:$P(\omega|\mathbf{x}) = [p(\mathbf{x}|\omega) * P(\omega)] / p(\mathbf{x})$because,

***$P(\omega|x)$ ≡ called the** posterior, it is the probability of the predicted class to be ω for a given entry of feature (x). Analogous to $P(O|\theta)$, because the class is the desired outcome to be predicted according to the data distribution (model). Capital 'P' because **ω** is a discrete random variable.

* **$p(x|\omega)$ ≡ class-conditional probability density function** for the feature. We call it ***likelihood** of* ω with respect to x, a term chosen to indicate that, other things being equal, the category (or class) for which it is large is more "likely" to be the true category. It is a function of parameters within the parameteric space that describes the probability of obtaining the observed data (x). Small 'P' because **x** is a continous random variable. We usually assume it to be following Gaussian Distribution.

* **$P(\omega)$** ≡ a **priori probability** (or simply prior) of class ω. It is usually pre-determined and depends on the external factors. It means how probable the occurence of class ω out of all the classes.

* **$p(x)$ ≡ called the** evidence, it is merely a scaling factor that guarantees that the posterior probabilities sum to one. $p(x) = sum(p(x|\omega)*P(\omega))$ over all the classes.
So finally we get the following equation to frame our decision rule

$$P(\omega_i|\overline{x}) = \frac{p(\overline{x}|\omega_i) * P(\omega_i)}{p(\overline{x})}$$

Bayes' Formula for Classification

Decision Rule

The above equation is the governing formula for our decision theory. The rule is as follows:

For each sample input, it calculates its posterior and assign it to the class corresponding to the maximum value of the posterior probability.

Mathematically it can be written as:

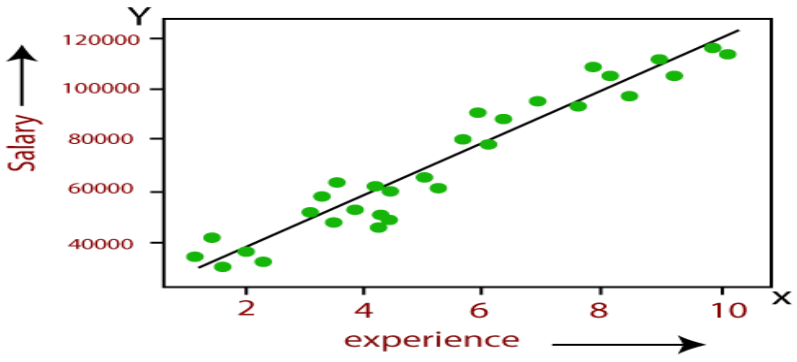
$$\omega_i = argmax_i[P(\omega_i|x)]$$

Bayes' Decision Rule

5	Summarize the following models. (i) Linear regression (ii) Logistic regression	**[L2][CO1]**	**[12M]**

Linear Regression:

- Linear Regression is one of the most simple Machine learning algorithm that comes under Supervised Learning technique and used for solving regression problems.
- It is used for predicting the continuous dependent variable with the help of independent variables.
- The goal of the Linear regression is to find the best fit line that can accurately predict the output for the continuous dependent variable.
- If single independent variable is used for prediction then it is called Simple Linear Regression and if there are more than two independent variables then such regression is called as Multiple Linear Regression.
- By finding the best fit line, algorithm establish the relationship between dependent variable and independent variable. And the relationship should be of linear nature.
- The output for Linear regression should only be the continuous values such as price, age, salary, etc. The relationship between the dependent variable and independent variable can be shown in below image:



In above image the dependent variable is on Y-axis (salary) and independent variable is on x-

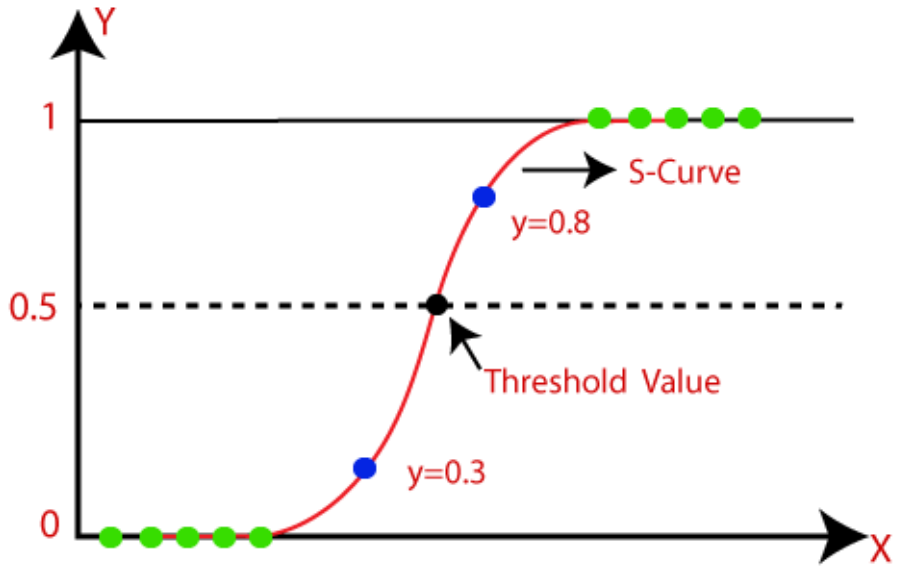
axis(experience). The regression line can be written as:

$y= a_0+a_1x+ \varepsilon$

Where, a_0 and a_1 are the coefficients and ε is the error term.

Logistic Regression:

- Logistic regression is one of the most popular Machine learning algorithm that comes under Supervised Learning techniques.
- It can be used for Classification as well as for Regression problems, but mainly used for Classification problems.
- Logistic regression is used to predict the categorical dependent variable with the help of independent variables.
- The output of Logistic Regression problem can be only between the 0 and 1.
- Logistic regression can be used where the probabilities between two classes is required. Such as whether it will rain today or not, either 0 or 1, true or false etc.
- Logistic regression is based on the concept of Maximum Likelihood estimation. According to this estimation, the observed data should be most probable.
- In logistic regression, we pass the weighted sum of inputs through an activation function that can map values in between 0 and 1. Such activation function is known as **sigmoid function** and the curve obtained is called as sigmoid curve or S-curve. Consider the below image:



- The equation for logistic regression is:

$$\log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

6	a	Organize how to Tackle Overfitting and Under fitting?	**[L4][CO3]**	**[6M]**

Underfitting: A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data, i.e., it only performs well on training data but performs poorly on testing data. *(It's just like trying to fit undersized pants!)* Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the

data well enough. It usually happens when we have fewer data to build an accurate model and also when we try to build a linear model with fewer non-linear data. In such cases, the rules of the machine learning model are too easy and flexible to be applied to such minimal data and therefore the model will probably make a lot of wrong predictions. Underfitting can be avoided by using more data and also reducing the features by feature selection.
In a nutshell, Underfitting refers to a model that can neither performs well on the training data nor generalize to new data.

Reasons for Underfitting:

- High bias and low variance
- The size of the training dataset used is not enough.
- The model is too simple.
- Training data is not cleaned and also contains noise in it.

Techniques to reduce underfitting:

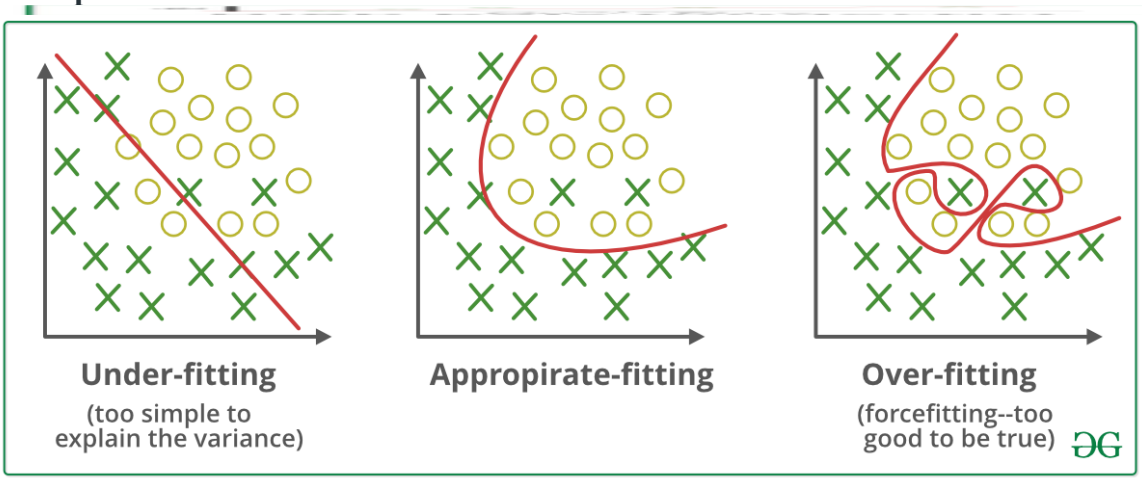
- Increase model complexity
- Increase the number of features, performing feature engineering
- Remove noise from the data.
- Increase the number of epochs or increase the duration of training to get better results.

Overfitting: A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. And when testing with test data results in High variance. Then the model does not categorize the data correctly, because of too many details and noise.
The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.
In a nutshell, Overfitting is a problem where the evaluation of machine learning algorithms on training data is different from unseen data.

Reasons for Overfitting are as follows:

1. High variance and low bias
2. The model is too complex
3. The size of the training data

Examples:



Techniques to reduce overfitting:

- Increase training data.
- Reduce model complexity.
- Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
- Ridge Regularization and Lasso Regularization Use dropout for neural networks to tackle overfitting

	b	Discriminate logistic discrimination analysis in machine learning.	**[L5][CO1]**	**[6M]**

Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of belonging to a given class. It is used for classification algorithms its name is logistic regression. it's referred to as regression because it takes the output of the linear regression function as input and uses a sigmoid function to estimate the probability for the given class. The difference between linear regression and logistic regression is that linear regression output is the continuous value that can be anything while logistic regression predicts the probability that an instance belongs to a given class or not.

Terminologies involved in Logistic Regression:

Here are some common terms involved in logistic regression:

Independent variables: The input characteristics or predictor factors applied to the dependent variable's predictions.

Dependent variable: The target variable in a logistic regression model, which we are trying to predict.

Logistic function: The formula used to represent how the independent and dependent variables relate to one another. The logistic function transforms the input variables into a probability value between 0 and 1, which represents the likelihood of the dependent variable being 1 or 0.

Odds: It is the ratio of something occurring to something not occurring. it is different from probability as the probability is the ratio of something occurring to everything that could possibly occur.

Log-odds: The log-odds, also known as the logit function, is the natural logarithm of the odds. In logistic regression, the log odds of the dependent variable are modeled as a linear combination of the independent variables and the intercept.

Coefficient: The logistic regression model's estimated parameters, show how the independent and dependent variables relate to one another.

Intercept: A constant term in the logistic regression model, which represents the log odds when all independent variables are equal to zero.

Maximum likelihood estimation: The method used to estimate the coefficients of the logistic regression model, which maximizes the likelihood of observing the data given the model.

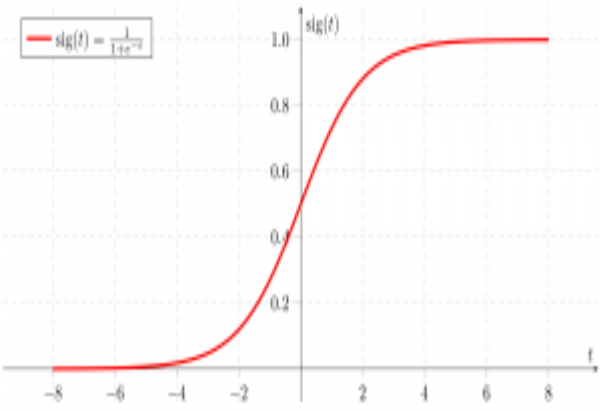
How does Logistic Regression work?

The logistic regression model transforms the linear regression function continuous value output into categorical value output using a sigmoid function, which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.

Let the independent input features be

Sigmoid Function

Now we use the sigmoid function where the input will be z and we find the probability between 0 and 1. i.e predicted y.



Sigmoid function

As shown above, the figure sigmoid function converts the continuous variable data into the probability i.e. between 0 and 1.

- Sigma(Z) tends towards 1 as

- Sigma(Z) tends towards 0 as
- Sigma(Z) is always bounded between 0 and 1

where the probability of being a class can be measured as:

$$P(y = 1) = \sigma(z)$$
$$P(y = 0) = 1 - \sigma(z)$$

Logistic Regression Equation

The odd is the ratio of something occurring to something not occurring. it is different from probability as the probability is the ratio of something occurring to everything that could possibly occur. so odd will be

$$\frac{p(x)}{1-p(x)} = e^z$$

Applying natural log on odd. then log odd will be

$$\log\left[\frac{p(x)}{1-p(x)}\right] = z$$
$$\log\left[\frac{p(x)}{1-p(x)}\right] = w \cdot X + b$$

then the final logistic regression equation will be:

$$p(X; b, w) = \frac{e^{w \cdot X + b}}{1+e^{w \cdot X + b}} = \frac{1}{1+e^{-w \cdot X + b}}$$

Likelihood function for Logistic Regression

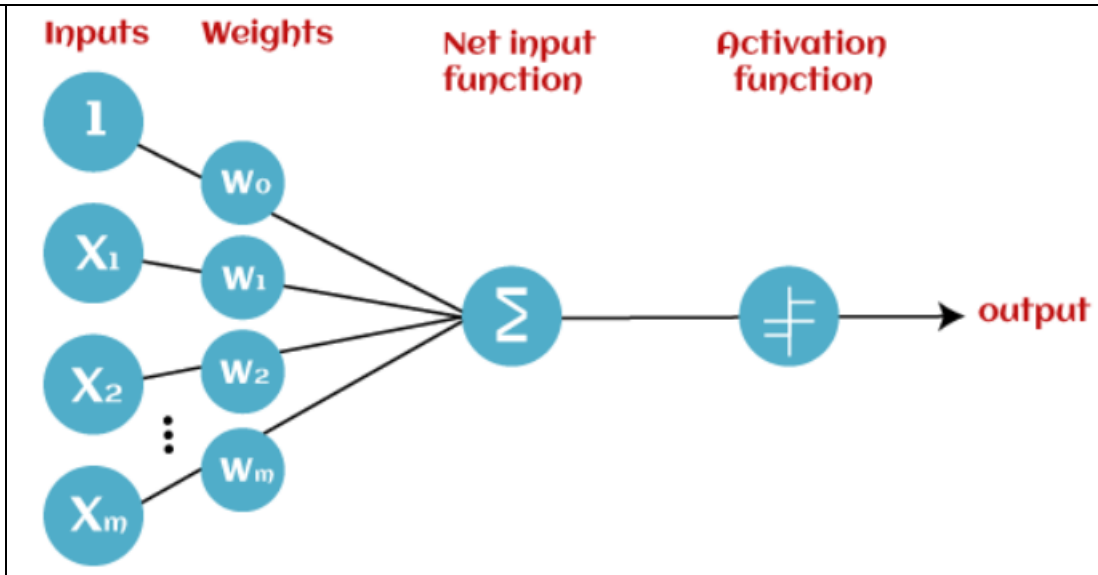
The predicted probabilities will p(X;b,w) = p(x) for y=1 and for y = 0 predicted probabilities will 1-p(X;b,w) = 1-p(x)

$$L(b, w) = \prod_{i=1} np(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

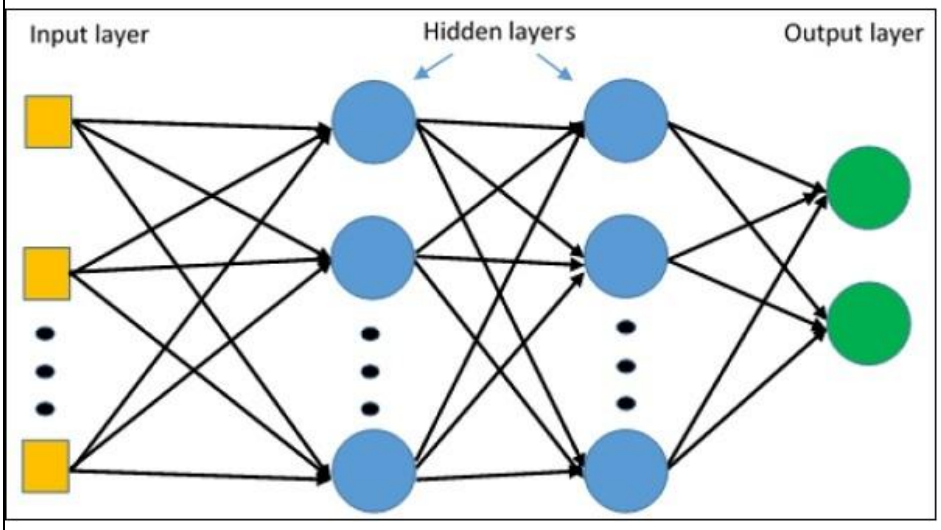
7	a	Illustrate Multi-Layer Perceptron in supervised learning.	**[L3][CO3]**	**[6M]**

Multi-Layered Perceptron Model:

Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence.



Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers. single-layer neural network with four main parameters, i.e., **input values, weights and Bias, net sum, and an activation function.**



The multi-layer perceptron model is also known as the Backpropagation algorithm, which executes in two stages as follows:

- **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.
- **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Hence, a multi-layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model. Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.

A multi-layer perceptron model has greater processing power and can process linear and non-linear patterns. Further, it can also implement logic gates such as AND, OR, XOR, NAND, NOT, XNOR, NOR.

In the multi-layer perceptron diagram above, we can see that there are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer,

and in the same way, the hidden layer processes the information and passes it to the output layer.

Every node in the multi-layer perception uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula.

$$(x) = 1/(1 + exp(-x))$$

Advantages of Multi-Layer Perceptron:

- A multi-layered perceptron model can be used to solve complex non-linear problems.
- It works well with both small and large input data.
- It helps us to obtain quick predictions after the training.
- It helps to obtain the same accuracy ratio with large as well as small data.

Disadvantages of Multi-Layer Perceptron:

- In Multi-layer perceptron, computations are difficult and time-consuming.
- In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
- The model functioning depends on the quality of the training.

b	Analyze Regression discrimination in machine learning.	**[L4][CO1]**	**[6M]**

In regression, the probabilistic model is

$$r^t = y^t + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. If $r^t \in \{0, 1\}$, y^t can be constrained to lie in this range using the sigmoid function. Assuming a linear model and two classes, we have

$$y^t = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x}^t + w_0) = \frac{1}{1 + \exp[-(\boldsymbol{w}^T \boldsymbol{x}^t + w_0)]}$$

Then the sample likelihood in regression, assuming $r|\boldsymbol{x} \sim \mathcal{N}(y, \sigma^2)$, is

$$l(\boldsymbol{w}, w_0|\mathcal{X}) = \prod_t \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(r^t - y^t)^2}{2\sigma^2}\right]$$

Maximizing the log likelihood is minimizing the sum of square errors:

$$E(\mathbf{w}, w_0|\mathcal{X}) = \frac{1}{2}\sum_t (r^t - y^t)^2$$

Using gradient descent, we get

$$\begin{aligned}\Delta \mathbf{w} &= \eta \sum_t (r^t - y^t) y^t (1 - y^t) \mathbf{x}^t \\ \Delta w_0 &= \eta \sum_t (r^t - y^t) y^t (1 - y^t)\end{aligned}$$

This method can also be used when there are $K > 2$ classes. The probabilistic model is

$$\mathbf{r}^t = \mathbf{y}^t + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}_K(0, \sigma^2 \mathbf{I}_K)$. Assuming a linear model for each class, we have

$$y_i^t = \text{sigmoid}(\mathbf{w}_i^T \mathbf{x}^t + w_{i0}) = \frac{1}{1 + \exp[-(\mathbf{w}_i^T \mathbf{x}^t + w_{i0})]}$$

Then the sample likelihood is

$$l(\{\mathbf{w}_i, w_{i0}\}_i|\mathcal{X}) = \prod_t \frac{1}{(2\pi)^{K/2}|\Sigma|^{1/2}} \exp\left[-\frac{\|\mathbf{r}^t - \mathbf{y}^t\|^2}{2\sigma^2}\right]$$

and the error function is

$$E(\{\mathbf{w}_i, w_{i0}\}_i|\mathcal{X}) = \frac{1}{2}\sum_t \|\mathbf{r}^t - \mathbf{y}^t\|^2 = \frac{1}{2}\sum_t \sum_i (r_i^t - y_i^t)^2$$

The update equations for $i = 1, \ldots, K$, are

$$\begin{aligned}\Delta \mathbf{w}_i &= \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t) \mathbf{x}^t \\ \Delta w_{i0} &= \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t)\end{aligned}$$

But note that in doing so, we do not make use of the information that only one of y_i needs to be 1 and all others are 0, or that $\sum_i y_i = 1$. The softmax function of equation 10.29 allows us to incorporate this extra information we have due to the outputs' estimating class posterior probabilities. Using sigmoid outputs in $K > 2$ case, we treat y_i as if they are independent functions.

Note also that for a given class, if we use the regression approach, there will be updates until the right output is 1 and all others are 0. This is not

in fact necessary because during testing, we are just going to choose the maximum anyway; it is enough to train only until the right output is larger than others, which is exactly what the softmax function does.

So this approach with multiple sigmoid outputs is more appropriate when the classes are *not* mutually exclusive and exhaustive. That is, for an $\boldsymbol{x}^t$, all r_i^t may be 0; namely, $\boldsymbol{x}^t$ does not belong to any of the classes, or more than one r_i^t may be 1, when classes overlap.

8	Discuss Back Propagation Algorithm in supervised learning.	**[L2][CO3]**	**[12M]**

Whatever you studied in Soft Computing the same you can read

Backpropagation is a supervised learning algorithm, for training Multi-layer Perceptrons (Artificial Neural Networks).

Back propagation (slideshare.net)

Below are the steps involved in Backpropagation:

- Step – 1: Forward Propagation
- Step – 2: Backward Propagation
- Step – 3: Putting all the values together and calculating the updated weight value

Whatever you study in soft computing you can write the same answer aslo.

9	Analyze Maximum Likelihood Estimation in supervised learning.	**[L4][CO3]**	**[12M]**

Maximum Likelihood Estimation (MLE) is a statistical method commonly used in supervised learning to estimate the parameters of a model based on observed data. In the context of supervised learning, MLE is used to find the best-fitting model that maximizes the likelihood of the observed data given the model's parameters.

Let's break down the concept of Maximum Likelihood Estimation in the context of supervised learning:

1. **Likelihood Function**: In supervised learning, we have a labeled dataset consisting of input features (X) and corresponding output labels (Y). The likelihood function (L) represents the probability of observing the given output labels (Y) given the input features (X) and the model's parameters (θ). In mathematical terms, the likelihood function is denoted as L(θ | X, Y).
2. **Assumption**: The central assumption in MLE is that the observed data (Y) is generated from a probability distribution determined by the model's parameters (θ) and the input features (X).
3. **Log-Likelihood Function**: In practice, it is common to work with the log-likelihood function (log L) rather than the likelihood function itself. Taking the logarithm of the likelihood function allows us to transform products of probabilities into sums, which simplifies calculations. The log-likelihood function is denoted as log L(θ | X, Y).
4. **MLE Objective**: The goal of MLE in supervised learning is to find the values of the model's parameters (θ) that maximize the log-likelihood function. Mathematically, this can be expressed as:

θ^ = argmax(log L(θ | X, Y))

Here, θ^ represents the estimated values of the model's parameters that maximize the likelihood of the observed data.

5. **Estimation Procedure**: To find the optimal values of θ, various optimization techniques, such as gradient descent, can be used. The optimization algorithm iteratively updates the model's parameters to maximize the log-likelihood function.
6. **Model Evaluation**: Once the MLE process is complete, the model with the estimated parameters θ^ can be evaluated on new, unseen data. The model's performance can be assessed using various evaluation metrics, such as accuracy, precision, recall, etc.
7. **Assumptions and Limitations**: MLE assumes that the data is independently and identically distributed (i.i.d.), which might not always hold in real-world scenarios. Additionally, the choice of the underlying probability distribution used to model the data can significantly impact the performance of MLE.

MLE is a powerful and widely used technique in supervised learning. It allows us to estimate the parameters of a model in a principled and statistically sound manner based on the observed data. By maximizing the likelihood of the data, MLE helps us find the best-fitting model that can make accurate predictions on new, unseen data.

10	a	Express the Evaluation of Estimator bias and variance.	**[L6][CO3]**	**[6M]**

Let $\mathcal{X}$ be a sample from a population specified up to a parameter θ, and let $d = d(\mathcal{X})$ be an estimator of θ. To evaluate the quality of this estimator, we can measure how much it is different from θ, that is, $(d(\mathcal{X}) - \theta)^2$. But since it is a random variable (it depends on the sample), we need to average this over possible $\mathcal{X}$ and consider $r(d, \theta)$, the *mean square error* of the estimator d defined as

$$r(d, \theta) = E[(d(\mathcal{X}) - \theta)^2]$$

The *bias* of an estimator is given as

$$b_\theta(d) = E[d(\mathcal{X})] - \theta$$

If $b_\theta(d) = 0$ for all θ values, then we say that d is an *unbiased estimator* of θ. For example, with x^t drawn from some density with mean μ, the sample average, m, is an unbiased estimator of the mean, μ, because

$$E[m] = E\left[\frac{\sum_t x^t}{N}\right] = \frac{1}{N}\sum_t E[x^t] = \frac{N\mu}{N} = \mu$$

This means that though on a particular sample, m may be different from μ, if we take many such samples, $\mathcal{X}_i$, and estimate many $m_i = m(\mathcal{X}_i)$, *their* average will get close to μ as the number of such samples increases. m is also a *consistent* estimator, that is, $\text{Var}(m) \to 0$ as $N \to \infty$.

$$\text{Var}(m) = \text{Var}\left(\frac{\sum_t x^t}{N}\right) = \frac{1}{N^2}\sum_t \text{Var}(x^t) = \frac{N\sigma^2}{N^2} = \frac{\sigma^2}{N}$$

As N, the number of points in the sample, gets larger, m deviates less from μ. Let us now check, s^2, the MLE of σ^2:

$$s^2 = \frac{\sum_t (x^t - m)^2}{N} = \frac{\sum_t (x^t)^2 - Nm^2}{N}$$

$$E[s^2] = \frac{\sum_t E[(x^t)^2] - N \cdot E[m^2]}{N}$$

Given that $\text{Var}(X) = E[X^2] - E[X]^2$, we get $E[X^2] = \text{Var}(X) + E[X]^2$, and we can write

$$E[(x^t)^2] = \sigma^2 + \mu^2 \text{ and } E[m^2] = \sigma^2/N + \mu^2$$

Then, plugging these in, we get

$$E[s^2] = \frac{N(\sigma^2 + \mu^2) - N(\sigma^2/N + \mu^2)}{N} = \left(\frac{N-1}{N}\right)\sigma^2 \neq \sigma^2$$

which shows that s^2 is a biased estimator of σ^2. $(N/(N-1))s^2$ is an unbiased estimator. However when N is large, the difference is negligable. This is an example of an *asymptotically unbiased estimator* whose bias goes to 0 as N goes to infinity.

The mean square error can be rewritten as follows—d is short for $d(\mathcal{X})$:

$$\begin{aligned}
r(d,\theta) &= E\left[(d-\theta)^2\right] \\
&= E\left[(d - E[d] + E[d] - \theta)^2\right] \\
&= E\left[(d-E[d])^2 + (E[d]-\theta)^2 + 2(E[d]-\theta)(d-E[d])\right] \\
&= E\left[(d-E[d])^2\right] + E\left[(E[d]-\theta)^2\right] + 2E\left[(E[d]-\theta)(d-E[d])\right] \\
&= E\left[(d-E[d])^2\right] + (E[d]-\theta)^2 + 2(E[d]-\theta)E[d-E[d]] \\
&= \underbrace{E\left[(d-E[d])^2\right]}_{variance} + \underbrace{(E[d]-\theta)^2}_{bias^2}
\end{aligned}$$

The two equalities follow because $E[d]$ is a constant and therefore $E[d] - \theta$ also is a constant, and because $E[d - E[d]] = E[d] - E[d] = 0$. In equation 4.11, the first term is the *variance* that measures how much, on average, d_i vary around the expected value (going from one dataset to another), and the second term is the *bias* that measures how much the expected value varies from the correct value θ (figure 4.1). We then write error as the sum of these two terms, the variance and the square of the bias:

$$r(d,\theta) = \text{Var}(d) + (b_\theta(d))^2$$

b	Illustrate Gradient descent algorithm and its variants.	**[L3][CO3]**	**[6M]**

Gradient descent is an optimization algorithm used to minimize the cost (or loss) function of a machine learning model. It iteratively adjusts the model's parameters in the opposite direction of the gradient of the cost function with respect to the parameters. This process continues until the algorithm converges to a local minimum, where the cost function is minimized.

The general formula for gradient descent is as follows:

$\theta i+1=\theta i-\alpha\nabla J(\theta i)$

where:

- θi represents the current parameter values at iteration i,
- α (alpha) is the learning rate, which determines the step size for each iteration,
- $\nabla J(\theta i)$ is the gradient of the cost function J with respect to the parameters θi.

Here, we illustrate the gradient descent algorithm visually for a simple 2D cost function:

1. Initialization: Start with some initial values for the parameters θ and choose a learning rate α.
2. Calculate Gradient: Compute the gradient of the cost function at the current parameter values.
3. Update Parameters: Update the parameters using the gradient and learning rate.
4. Repeat: Repeat steps 2 and 3 until convergence or after a predefined number of iterations.

There are several variants of gradient descent that aim to improve the convergence speed or overcome some of the limitations of the basic algorithm. Some popular variants include:

1. Stochastic Gradient Descent (SGD): Instead of using the entire dataset to compute the gradient, SGD updates the parameters using only one random data point at each iteration. This can lead to faster updates but may result in noisy convergence.
2. Mini-batch Gradient Descent: This approach is a compromise between batch gradient descent (using the entire dataset) and stochastic gradient descent. It updates the parameters using a small random subset (mini-batch) of the training data at each iteration.
3. Momentum: Momentum adds a fraction of the previous update vector to the current update, which helps to accelerate convergence and smooth out oscillations in the gradient descent path.
4. AdaGrad (Adaptive Gradient Algorithm): AdaGrad adapts the learning rate for each parameter based on the historical gradients, which can be beneficial for sparsely occurring features.
5. RMSprop (Root Mean Square Propagation): RMSprop is similar to AdaGrad but uses a moving average of the squared gradients to update the learning rates, which helps handle vanishing and exploding gradients.
6. Adam (Adaptive Moment Estimation): Adam combines the ideas of momentum and RMSprop, providing adaptive learning rates and momentum for each parameter.

These variants allow gradient descent to be more efficient and effective, especially when dealing with large datasets or complex optimization surfaces. The choice of which variant to use depends on the specific problem and dataset characteristics.

Course Code: 20CS0906 **R20**

UNIT –III

UNSUPERVISED LEARNING

1		Discuss the following terms in unsupervised learning i. Association rules ii . Clustering	**[L2][CO5]**	**[12M]**
	Referred from first and second units			
2	a	Explain the various Clustering algorithms.	**[L2][CO2]**	**[6M]**

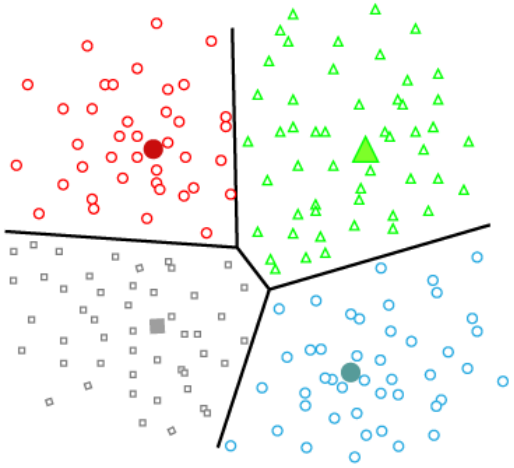
Clustering algorithms are a type of unsupervised machine learning technique used to group similar data points together based on their inherent characteristics or similarities.

Types of Clustering

Centroid-based Clustering: (Partitioning Clustering)

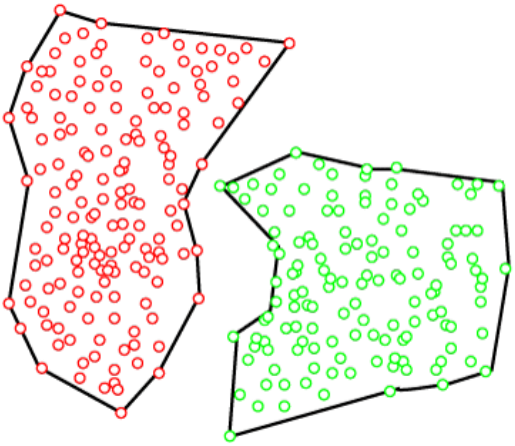
It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the **centroid-based method**. The most common example of partitioning clustering is the **K-Means Clustering algorithm**.

In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.



Density-based Clustering.: The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected. his algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.



Distribution-based Clustering: In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly **Gaussian Distribution**.

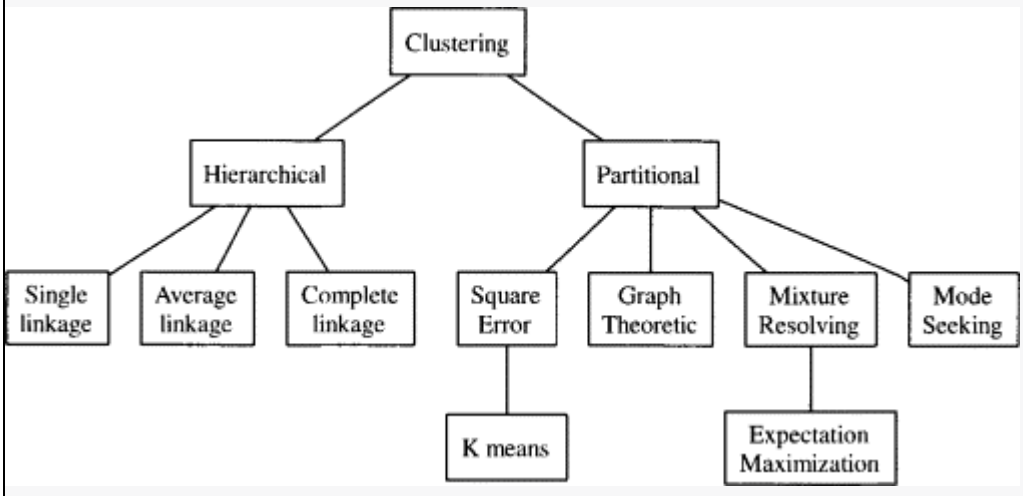
The example of this type is the **Expectation-Maximization Clustering algorithm** that uses Gaussian Mixture Models (GMM).

Hierarchical Clustering: Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created.

Fuzzy Clustering

Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster. **Fuzzy C-means algorithm** is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

These algorithms analyze the patterns and structures within the data to identify groups or clusters that share similar properties. Here are some popular clustering algorithms:



Different types of clustering algorithms are as follows:

K-means: K-means is one of the most widely used clustering algorithms. It aims to partition data into K distinct clusters based on the mean value of the data points. The algorithm iteratively assigns data points to the nearest cluster centroid and updates the centroids until convergence.
Hierarchical Clustering: Hierarchical clustering builds a hierarchy of clusters, either bottom-up

(agglomerative) or top-down (divisive). The algorithm starts with each data point as a separate cluster and then merges or splits clusters based on their similarities until a desired number of clusters is obtained.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise): DBSCAN groups data points based on their density. It defines clusters as areas of high-density separated by areas of low-density. It can discover clusters of arbitrary shape and is robust to noise and outliers.

Mean Shift: Mean Shift iteratively shifts the centroids of clusters towards the densest regions of data points. It starts with an initial set of centroids and updates them based on the mean shift of data points within a certain radius until convergence. It is effective in identifying clusters with irregular shapes and varying densities.

Gaussian Mixture Models (GMM): GMM assumes that the data points are generated from a mixture of Gaussian distributions. It models the data as a collection of Gaussian components, each representing a cluster. The algorithm estimates the parameters of the Gaussian distributions to identify the clusters.

Spectral Clustering: Spectral clustering uses the eigenvalues and eigenvectors of a similarity matrix to perform dimensionality reduction and then applies a clustering algorithm (e.g., K-means) on the reduced representation. It is particularly effective in identifying non-linear and complex structures.

Agglomerative Clustering: Agglomerative clustering is a bottom-up approach where each data point starts as a separate cluster, and then clusters are successively merged based on their similarity until a stopping criterion is met. It forms a hierarchical cluster tree or dendrogram.

These are just a few examples of clustering algorithms, and there are many other variations and specialized algorithms available depending on the specific requirements and characteristics of the data. It's important to choose the appropriate clustering algorithm based on the nature of the data and the desired outcome.

b	List out the various applications of clustering.	**[L1][CO6]**	**[6M]**

Clustering algorithms have various applications across different domains. Here are some common applications of clustering:

Customer Segmentation: Clustering is used to segment customers based on their purchasing behavior, demographics, or other attributes. This helps businesses understand customer segments and tailor marketing strategies, product recommendations, and customer support accordingly.

Image Segmentation: Clustering is employed to partition images into meaningful regions or objects based on pixel intensities, colors, textures, or other visual features. It finds applications in computer vision, object recognition, and image processing tasks.

Anomaly Detection: Clustering algorithms can be used to identify anomalies or outliers in datasets. By clustering normal data points together, any data point that does not belong to any cluster can be considered as an anomaly. This is useful in fraud detection, network intrusion detection, and detecting anomalies in sensor data.

Document Clustering: Clustering is utilized to group documents or texts based on their content or similarity. It aids in tasks like information retrieval, topic modeling, sentiment analysis, and document organization.

Recommender Systems: Clustering is used in collaborative filtering-based recommender systems to group users or items with similar preferences. This helps in making personalized recommendations by identifying clusters of users with similar tastes or clusters of items with similar characteristics.

Market Segmentation: Clustering assists in market research by segmenting markets based on customer preferences, behaviors, or demographics. This enables businesses to target specific market segments with tailored marketing campaigns and product offerings.

Gene Expression Analysis: Clustering is applied to gene expression data to identify groups of genes with similar expression patterns. This aids in understanding genetic relationships, gene function discovery, and studying diseases at a molecular level.

Image Compression: Clustering algorithms, such as vector quantization, are used in image compression techniques to group similar image patches and represent them with fewer bits. This helps

in reducing the storage space required for images.

Social Network Analysis: Clustering can be used to identify communities or clusters of individuals with similar interests or social connections in social network data. It helps in understanding social relationships, influence analysis, and targeted advertising.

Traffic Pattern Analysis: Clustering algorithms can be used to analyze traffic patterns and identify groups of similar traffic flow patterns in transportation data. This aids in traffic management, route planning, and optimizing transportation systems.

These are just a few examples of the wide range of applications where clustering algorithms can be employed. The suitability of clustering depends on the specific problem and the nature of the data being analyzed.

3	a	Illustrate the mixtures of latent variable models.	**[L3][CO3]**	**[6M]**

Latent Variable models

Latent variable models aim to model the probability distribution with latent variables.

> Latent variables are a transformation of the data points into a **continuous lower-dimensional space**.

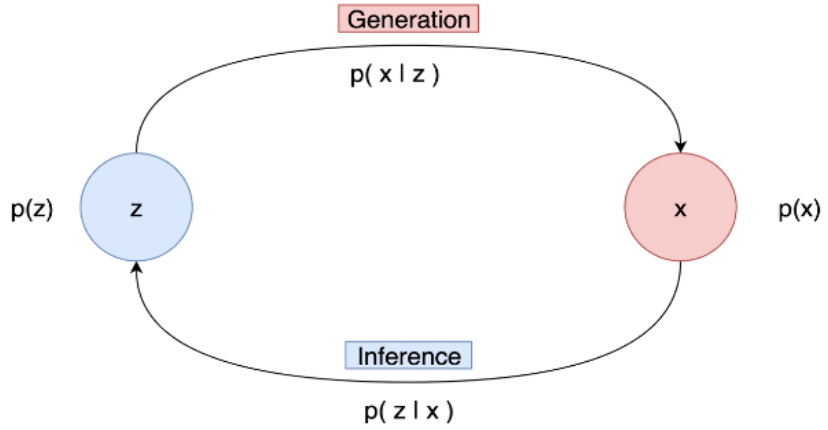
Intuitively, the latent variables will describe or "explain" the data in a simpler way.

In a stricter mathematical form, data points x that follow a probability distribution $p(x)$, are mapped into latent variables z that follow a distribution $p(z)$.

Given that idea, we can now define five basic terms:

- The **prior distribution** $p(z)$ that models the behaviour of the latent variables
- The **likelihood** $p(x|z)$ that defines how to map latent variables to the data points
- The **joint distribution** $p(x, z) = p(x|z)p(z)$, which is the multiplication of the likelihood and the prior and essentially describes our model.
- The **marginal distribution** $p(x)$ is the distribution of the original data and it is the ultimate goal of the model. The marginal distribution tells us how possible it is to generate a data point.
- The **posterior distribution** $p(z|x)$ which describes the latent variables that can be produced by a specific data point

Visually we can keep in mind the following diagram.



And here is the point where everything clicks together. If we assume that we somehow know the likelihood $p(x|z)$, the posterior $p(z|x)$, the marginal $p(x)$, and the prior $p(z)$ we can do the following:

In machine learning, mixture models are a class of latent variable models that are used to represent complex distributions by combining simpler component distributions. Latent variable models involve unobserved variables (latent variables) that are used to capture hidden patterns or structure in the data.

Let's consider an example of a mixture of Gaussian distributions, which is one of the most commonly used types of mixture models. In this case, the observed data is assumed to come from a combination of several Gaussian distributions.

Model Representation:
Latent Variables: We introduce a set of latent variables, often called "mixture indicators" or "cluster assignments," denoted as z. Each latent variable z corresponds to a specific component of the mixture.
Parameters: We have a set of parameters for the mixture model, including the mixing proportions π and the parameters (mean and covariance) of each Gaussian component.
Data Generation:
Sample Cluster: For each data point, we first sample a latent variable z from a categorical distribution according to the mixing proportions π. This determines the component from which the data point will be generated.
Generate Data: Given the selected component, we sample the data point x from the corresponding Gaussian distribution.
Model Inference:
Given observed data points x, the goal is to infer the latent variables z and the model parameters. Inference can be done using various techniques such as Expectation-Maximization (EM) algorithm, variational inference, or Markov chain Monte Carlo (MCMC) methods.
Model Learning:
The model parameters, including the mixing proportions π and the Gaussian parameters, are learned from the observed data using the chosen inference algorithm.
The learning process involves iteratively updating the model parameters until convergence, maximizing the likelihood or posterior probability of the observed data.
Model Utilization:
Once the model is learned, it can be used for various tasks such as clustering, density estimation, anomaly detection, or generating new data points from the learned distribution.

Mixture models are powerful tools in machine learning as they can capture complex data distributions by combining simpler components. They are widely used in various domains, including image analysis, natural language processing, recommendation systems, and many more.

	b	How mixture density is calculated in unsupervised learning?	**[L1][CO2]**	**[6M]**
		In unsupervised learning, the calculation of the mixture density involves estimating the parameters of a mixture model from the observed data. The mixture density represents the probability density function (PDF) of the observed data, which is a combination of multiple component densities. Here's a general overview of how the mixture density is calculated in unsupervised learning: **Choose the Mixture Model**: Select the type of mixture model that best suits the data distribution. Common choices include Gaussian Mixture Models (GMMs) or other types of mixture models like Dirichlet Process Mixtures. **Specify the Number of Components**: Determine the number of components (clusters) in the mixture model. This can be done based on prior knowledge or using techniques such as model selection criteria (e.g., AIC, BIC) or cross-validation. **Initialize the Model Parameters**: Initialize the parameters of the mixture model, including the mixing proportions and the parameters of each component distribution (e.g., mean, covariance for Gaussian components). **E-step: Expectation Step**: Given the current parameter estimates, calculate the posterior probabilities or responsibilities of each component for each data point. This step is often computed using the Bayes' theorem or the posterior probability of the latent variables given the observed data. **M-step: Maximization Step:** Update the model parameters based on the responsibilities obtained in the E-step. This typically involves maximizing the likelihood or maximizing the expected complete-data log-likelihood. **Iterative Optimization**: Iterate between the E-step and M-step until convergence. The convergence criteria can be based on the change in log-likelihood or a predetermined number of iterations. **Compute the Mixture Density**: Once the mixture model parameters have converged, the mixture density can be computed by combining the densities of each component, weighted by the corresponding mixing proportions. **Utilize the Mixture Density**: The calculated mixture density can be used for various purposes, such as clustering, density estimation, anomaly detection, or generating new samples from the learned distribution. It's important to note that the specific algorithms and techniques used for the estimation and calculation of the mixture density may vary depending on the chosen mixture model and the inference method employed (e.g., EM algorithm, variational inference, etc.).		
4	a	Analyze the working principle of K-means Clustering.	**[L4][CO2]**	**[7M]**
		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.

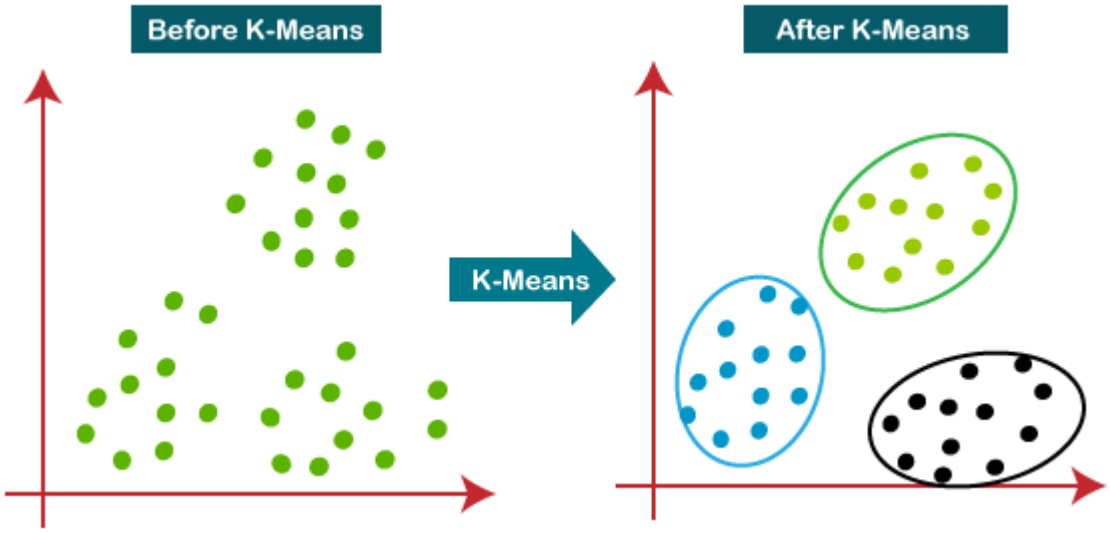
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

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:



How does the K-Means Algorithm Work?

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.

Consider any example for the explanation.

	b	Give the different types of Partitional algorithms used in clustering.	**[L2][CO2]**	**[5M]**

Partitional clustering algorithms are a class of clustering algorithms that partition the dataset into non-overlapping clusters. Here are some commonly used types of partitional clustering algorithms:

K-means: K-means is a widely used partitional clustering algorithm. It aims to partition the data into K clusters, where K is pre-specified by the user. The algorithm iteratively assigns data points to the nearest cluster centroid and updates the centroids until convergence.

K-medoids: K-medoids is a variation of K-means that uses actual data points, known as medoids, as cluster centers. It is robust to outliers compared to K-means, as medoids can be any data point in the cluster rather than the mean of the cluster.

Fuzzy C-means: Fuzzy C-means extends K-means by allowing data points to belong to multiple clusters with different degrees of membership. It assigns membership weights to data points indicating their degree of belongingness to each cluster. This algorithm is useful when data points exhibit partial membership to different clusters.

Partitioning Around Medoids (PAM): PAM is a partitional clustering algorithm that, similar to K-medoids, uses medoids as cluster centers. It differs from K-medoids in the way it selects initial medoids and updates them during the iterative process. PAM aims to minimize the total dissimilarity between data points and their closest medoid.

CLARA (Clustering Large Applications): CLARA is an algorithm that extends PAM to handle large datasets. It samples subsets of the data and applies PAM to each subset, providing an approximate clustering solution. The final clustering is obtained by merging the results of multiple runs.

CLARANS (Clustering Large Applications based on RANdomized Search): CLARANS is another partitional clustering algorithm suitable for large datasets. It randomly explores the search space to find the best medoids and avoid exhaustive search. It offers a trade-off between efficiency and accuracy.

X-means: X-means is an extension of K-means that automatically determines the optimal number of clusters. It starts with a single cluster and recursively splits clusters based on a statistical criterion until the optimal number of clusters is found.

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies): BIRCH is a partitional clustering algorithm that constructs a tree-like structure called the Clustering Feature Tree (CFT) to perform clustering. It performs hierarchical clustering on the CFT, resulting in a set of subclusters that represent the final clustering solution.

5	a	List out the various types of Cluster methods in unsupervised learning.	**[L1][CO6]**	**[6M]**

Types of Clustering

Several approaches to clustering exist. For an exhaustive list, see A Comprehensive Survey of Clustering Algorithms Xu, D. & Tian, Y. Ann. Data. Sci. (2015) 2: 165. Each approach is best suited to a particular data distribution. Below is a short discussion of four common approaches, focusing on centroid-based clustering using k-means.

Centroid-based Clustering

Centroid-based clustering organizes the data into non-hierarchical clusters, in contrast to hierarchical clustering defined below. k-means is the most widely-used centroid-based clustering algorithm. Centroid-based algorithms are efficient but sensitive to initial conditions and outliers. This course focuses on k-means because it is an efficient, effective, and simple clustering algorithm.

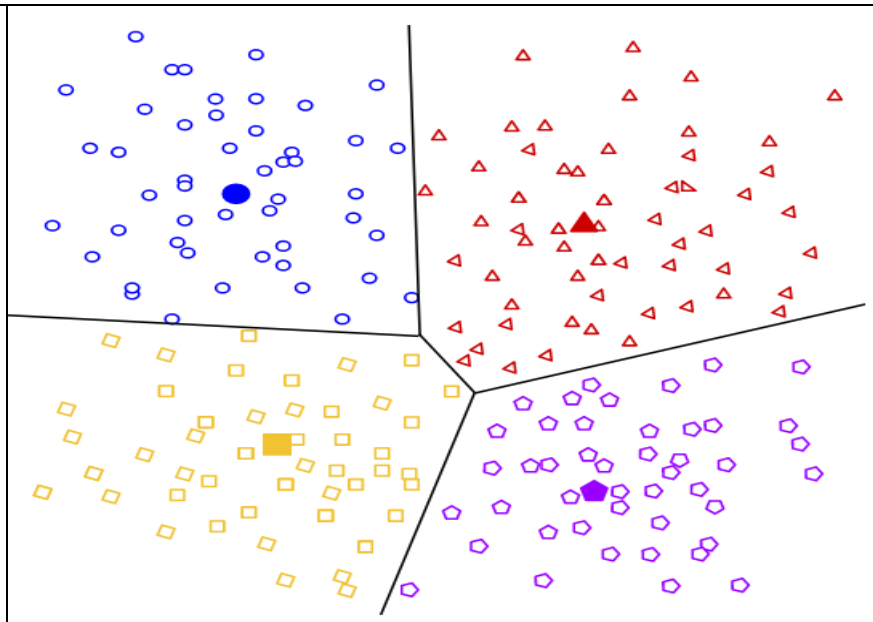


Figure 1: Example of centroid-based clustering.

Density-based Clustering

Density-based clustering connects areas of high example density into clusters. This allows for arbitrary-shaped distributions as long as dense areas can be connected. These algorithms have difficulty with data of varying densities and high dimensions. Further, by design, these algorithms do not assign outliers to clusters.

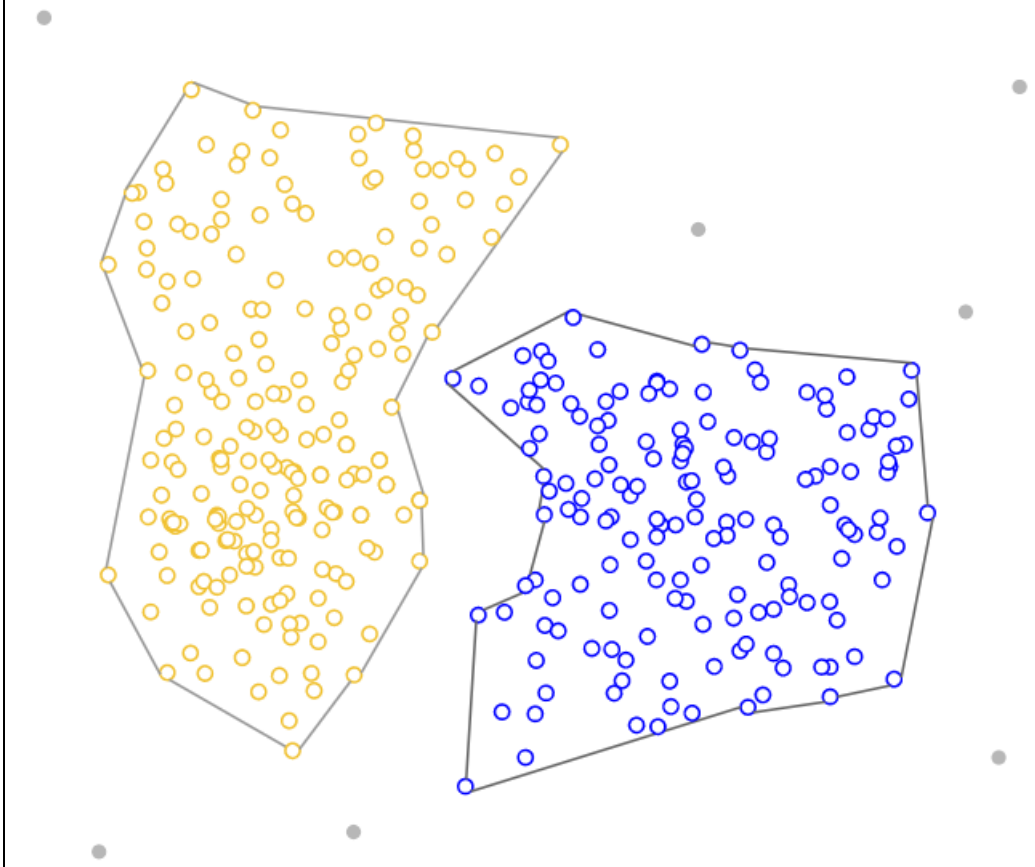


Figure 2: Example of density-based clustering.

Distribution-based Clustering

This clustering approach assumes data is composed of distributions, such as **Gaussian distributions**. In Figure 3, the distribution-based algorithm clusters data into three Gaussian distributions. As distance from the distribution's center increases, the probability that a point belongs to the distribution

decreases. The bands show that decrease in probability. When you do not know the type of distribution in your data, you should use a different algorithm.

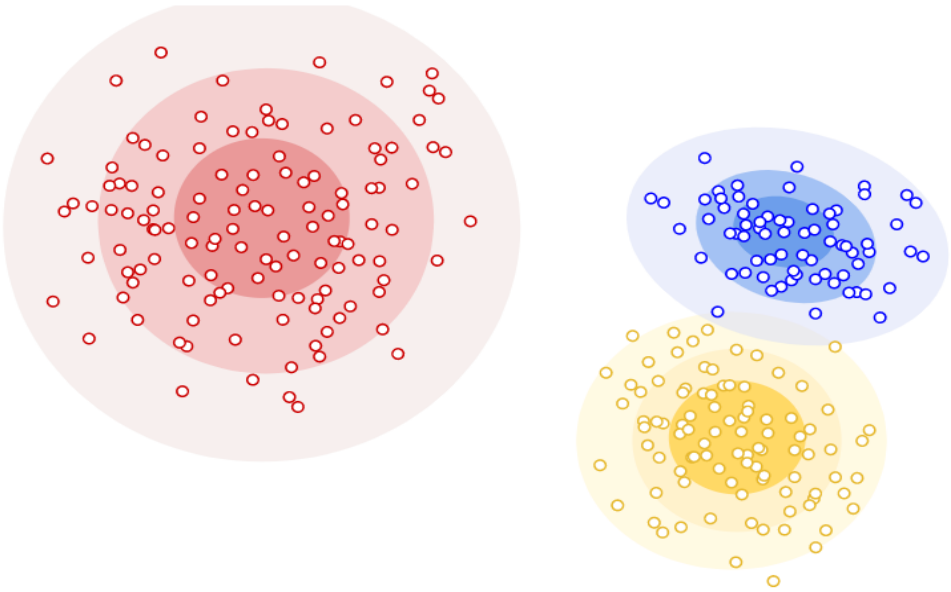


Figure 3: Example of distribution-based clustering.

Hierarchical Clustering

Hierarchical clustering creates a tree of clusters. Hierarchical clustering, not surprisingly, is well suited to hierarchical data, such as taxonomies. See *Comparison of 61 Sequenced Escherichia coli Genomes* by Oksana Lukjancenko, Trudy Wassenaar & Dave Ussery for an example. In addition, another advantage is that any number of clusters can be chosen by cutting the tree at the right level.

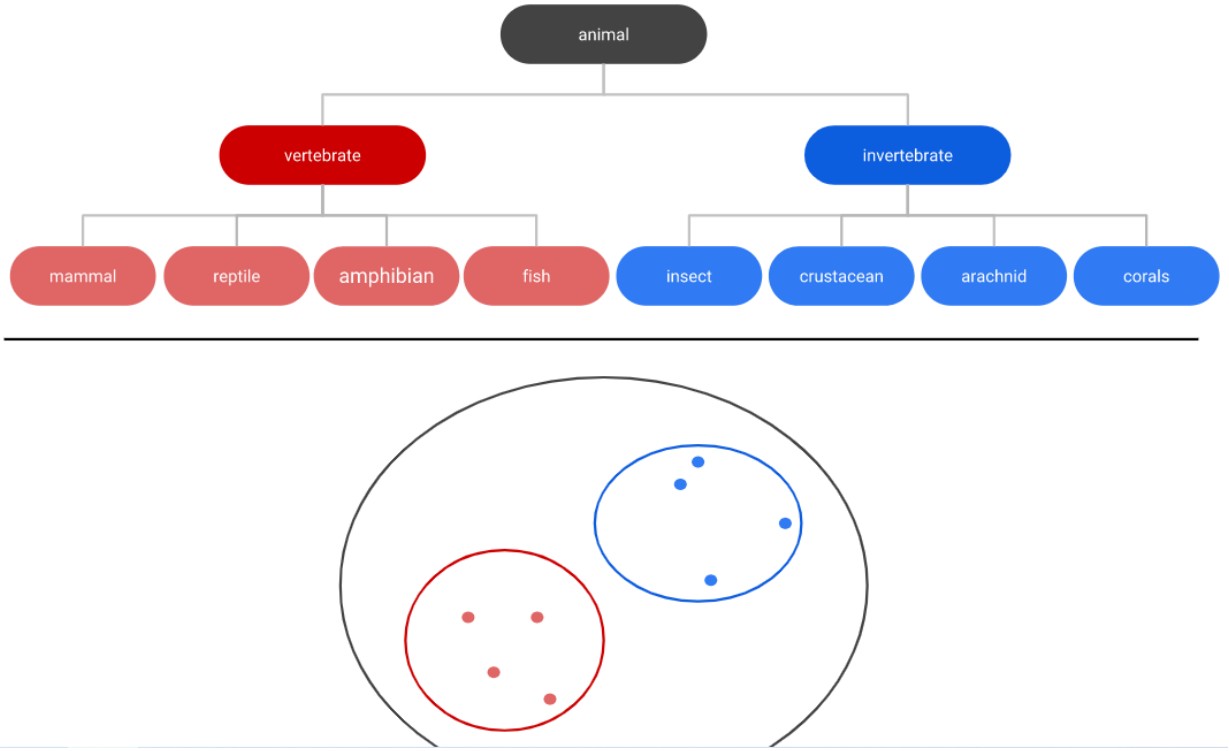
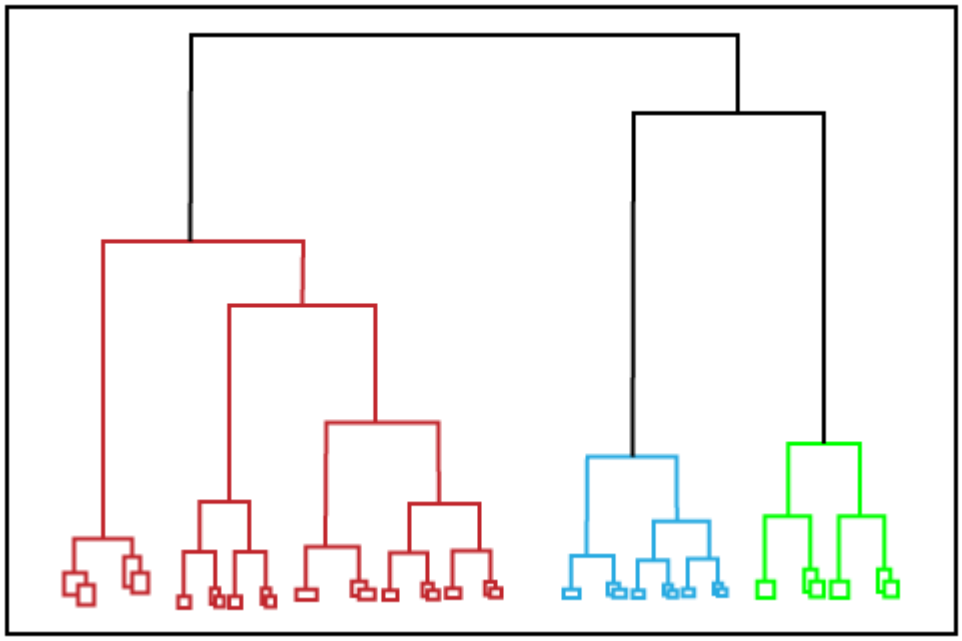


Figure 4: Example of a hierarchical tree clustering animals.

Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a **dendrogram**. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the **Agglomerative Hierarchical algorithm**.



b	Infer the similarities and differences between average-link clustering and k-means?	**[L4][CO5]**	**[6M]**

Both average-link clustering and k-means are popular clustering algorithms, but they have some similarities and differences in terms of their approach and characteristics. Here's a comparison between the two:

Similarities:

Unsupervised Learning: Both average-link clustering and k-means are unsupervised learning algorithms, meaning they do not require labeled data for training. They discover patterns and groupings in the data without prior knowledge of the class labels.

Iterative Process: Both algorithms use an iterative process to refine their cluster assignments. They repeatedly update the cluster centroids or merge clusters until convergence or a stopping criterion is met.

Differences:

Algorithm Type: Average-link clustering is a hierarchical clustering algorithm, whereas k-means is a centroid-based clustering algorithm. This fundamental difference affects how the clusters are formed and the overall approach to clustering.

Cluster Representation: Average-link clustering produces a hierarchical structure of clusters, often represented as a dendrogram. It captures the nested relationships between clusters and allows for fdifferent levels of granularity. In contrast, k-means produces non-overlapping, flat clusters, with each data point assigned to a single cluster.

Distance Metric: Average-link clustering typically uses a distance or dissimilarity metric, such as Euclidean distance or cosine similarity, to measure the similarity between clusters during the merging process. K-means, on the other hand, uses the distance between data points and the cluster centroids to assign points to the nearest centroid.

Number of Clusters: Average-link clustering does not require specifying the number of clusters in advance. The hierarchy can be cut at different levels to obtain different numbers of clusters. In contrast, k-means requires the user to specify the desired number of clusters (K) before running the algorithm.

Complexity: Average-link clustering can have higher computational complexity compared to k-means, especially for large datasets, as it needs to compute and update the pairwise distances between clusters in each iteration. K-means, on the other hand, has a lower computational complexity, making it more efficient for larger datasets.

Sensitivity to Initialization: K-means is sensitive to the initial placement of cluster centroids.

Different initializations can result in different final cluster assignments and centroids. Average-link clustering is less sensitive to initialization because it operates on a hierarchical structure and merges clusters based on similarity.

6	a	Generalize K-Means Clustering algorithm in Unsupervised Learning.	**[L6][CO2]**	**[6M]**

Generalized k-means clustering is an extension of the traditional k-means clustering algorithm that allows for more flexible and customizable clustering. While the standard k-means algorithm assigns data points to clusters based on their proximity to cluster centroids, generalized k-means clustering introduces additional parameters and distance metrics to accommodate various data types and cluster shapes.

In traditional k-means clustering, each data point is assigned to the cluster with the nearest centroid, where the centroid is the mean vector of the data points in that cluster. The algorithm aims to minimize the sum of squared distances between the data points and their assigned centroids. However, this approach assumes that the clusters are spherical and that the data features are continuous and normally distributed.

Generalized k-means clustering relaxes these assumptions and offers more flexibility. Here are a few key elements that can be customized in generalized k-means clustering:

Distance metrics: Instead of relying solely on the Euclidean distance, generalized k-means allows for the use of other distance metrics that are more suitable for specific data types. For example, for categorical data, Hamming distance or Jaccard distance can be used.

Cluster shape: Traditional k-means assumes that clusters are spherical and have equal variance. Generalized k-means allows for different cluster shapes, such as elliptical or arbitrary-shaped clusters. This is achieved by using a covariance matrix for each cluster and considering the Mahalanobis distance to measure the dissimilarity between data points and cluster centroids.

Weighting: Generalized k-means allows for assigning different weights to different dimensions or features of the data. By assigning appropriate weights, certain dimensions can be emphasized or de-emphasized in the clustering process.

Constraints: Generalized k-means can incorporate additional constraints into the clustering process. For example, constraints can be applied to enforce that certain data points must belong to specific clusters or that clusters must have a minimum number of data points.

Overall, generalized k-means clustering offers more flexibility and adaptability to different data types and clustering scenarios. By customizing the distance metric, cluster shape, weighting, and constraints, it becomes possible to better model and analyze complex data sets in a way that suits the specific requirements of the problem at hand.

	b	Estimate the problems associated with clustering large data.	**[L5][CO6]**	**[6M]**

Clustering large data sets can pose several challenges and problems. **Here are some common issues associated with clustering large data:**

Scalability: As the data size increases, clustering algorithms may struggle to handle the computational and memory requirements. The time complexity of clustering algorithms can be quite high, and the computational cost grows exponentially with the number of data points. Efficient algorithms and distributed computing techniques are required to tackle scalability issues.

High Dimensionality: Large data sets often have a high number of dimensions or features, which can lead to the curse of dimensionality. In high-dimensional spaces, the distance between points becomes less meaningful, and the clustering algorithms may struggle to find meaningful clusters. Dimensionality reduction techniques or feature selection methods can be employed to mitigate this problem.

Computational Complexity: Many clustering algorithms have computational complexities

that are quadratic or higher, such as hierarchical clustering algorithms or k-means clustering. With large data sets, these algorithms can become prohibitively slow or impractical to execute. Approximation techniques, parallelization, or sampling methods may be used to address this challenge.

Noise and Outliers: Large data sets often contain noise, outliers, or irrelevant data points. These outliers can have a significant impact on clustering results, as they may form their own clusters or disrupt the clustering of other data points. Preprocessing steps, such as outlier detection and data cleaning, are important to handle noisy data effectively.

Cluster Interpretability: Interpreting and understanding clusters in large data sets can be challenging. Visualizing high-dimensional data becomes more difficult, and it may be hard to discern meaningful patterns or extract insights from the clustering results. Advanced visualization techniques and dimensionality reduction methods can help in improving interpretability.

Cluster Validity and Evaluation: Assessing the quality and validity of clustering results becomes more complex with large data sets. Traditional clustering evaluation metrics may not be suitable, and it may be difficult to define ground truth or expert-labeled clusters for comparison. Developing appropriate evaluation measures for large-scale clustering is an ongoing research area.

Storage and Memory Constraints: Large data sets require significant storage space and memory to process and store intermediate results during clustering. Managing storage and memory constraints can be challenging, particularly when dealing with distributed computing or limited resources.

Problems associated with clustering

There are a number of problems with clustering. Among them:

- dealing with large number of dimensions and large number of data items can be problematic because of time complexity;
- the effectiveness of the method depends on the definition of "distance" (for distance-based clustering). If an *obvious* distance measure doesn't exist we must "define" it, which is not always easy, especially in multidimensional spaces;
- the result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways.

7		Describe the various types of Hierarchal Clustering techniques.	**[L2][CO3]**	**[12M]**

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

1. **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
2. **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach.**

Agglomerative Hierarchical clustering

The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters

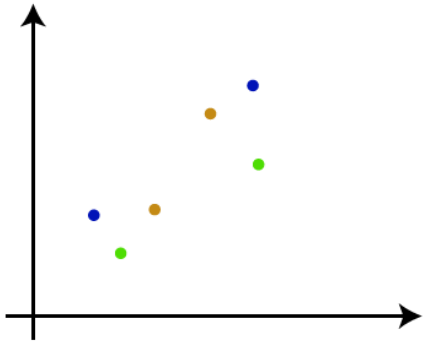
together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

This hierarchy of clusters is represented in the form of the dendrogram.

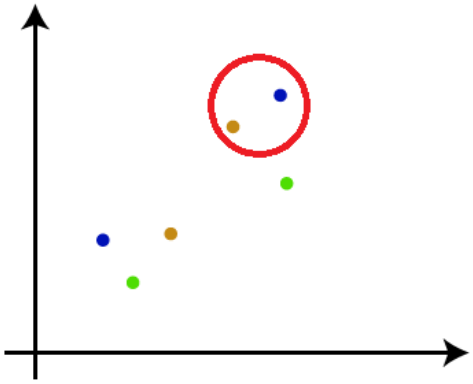
How the Agglomerative Hierarchical clustering Work?

The working of the AHC algorithm can be explained using the below steps:

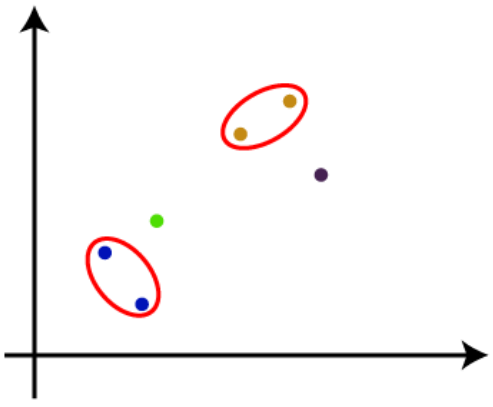
- **Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



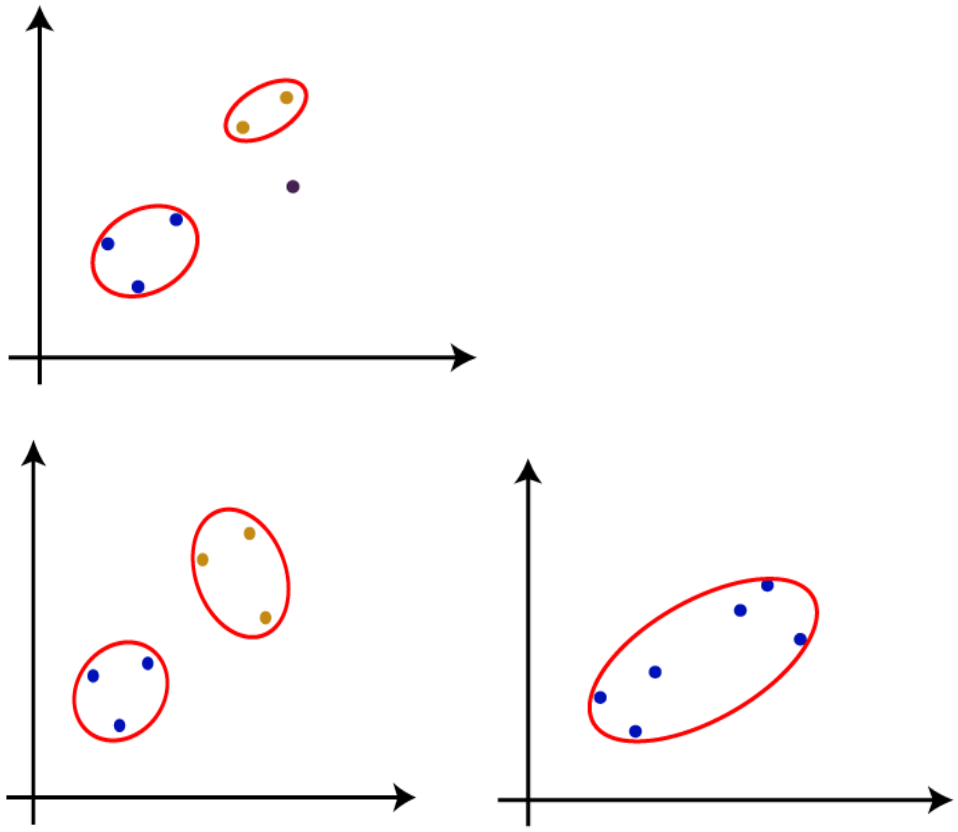
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.



- **Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.



- **Step-4:** Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:



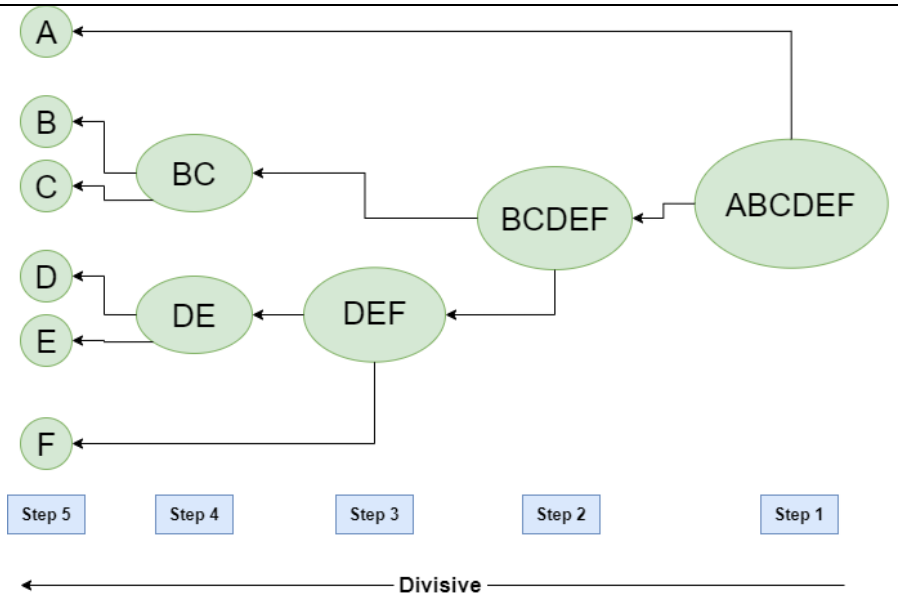
- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

Hierarchical Divisive clustering

It is also known as a top-down approach. This algorithm also does not require to prespecify the number of clusters. Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton clusters.

Algorithm :

given a dataset (d_1, d_2, d_3, ...d_N) of size N
at the top we have all data in one cluster
the cluster is split using a flat clustering method eg. K-Means etc
repeat
choose the best cluster among all the clusters to split
split that cluster by the flat clustering algorithm
until each data is in its own singleton cluster

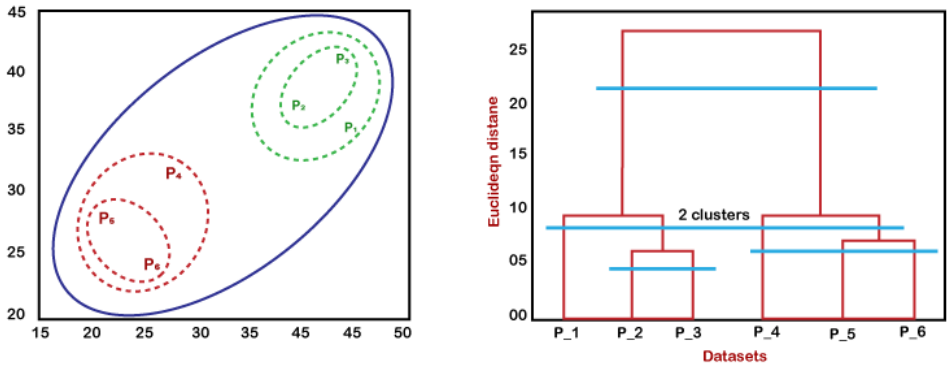


Hierarchical Divisive clustering

Woking of Dendrogram in Hierarchical clustering

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

The working of the dendrogram can be explained using the below diagram:



In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

- As we have discussed above, firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created, which connects P2 and P3 with a rectangular shape. The hight is decided according to the Euclidean distance between the data points.
- In the next step, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than of previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.
- Again, two new dendrograms are created that combine P1, P2, and P3 in one dendrogram, and P4, P5, and P6, in another dendrogram.
- At last, the final dendrogram is created that combines all the data points together.

8		Analyze the Expectation-Maximization algorithm with simple Example.	**[L4][CO3]**	**[12M]**

EM ALGORITHM

- In the real-world applications of machine learning, it is very common that there are many relevant features available for learning but only a small subset of them are observable.
- The ***Expectation-Maximization algorithm*** can be used for the latent variables (variables that are not directly observable and are actually inferred from the values of the other observed variables).
- This algorithm is actually the base for many unsupervised clustering algorithms in the field of machine learning.

EM ALGORITHM

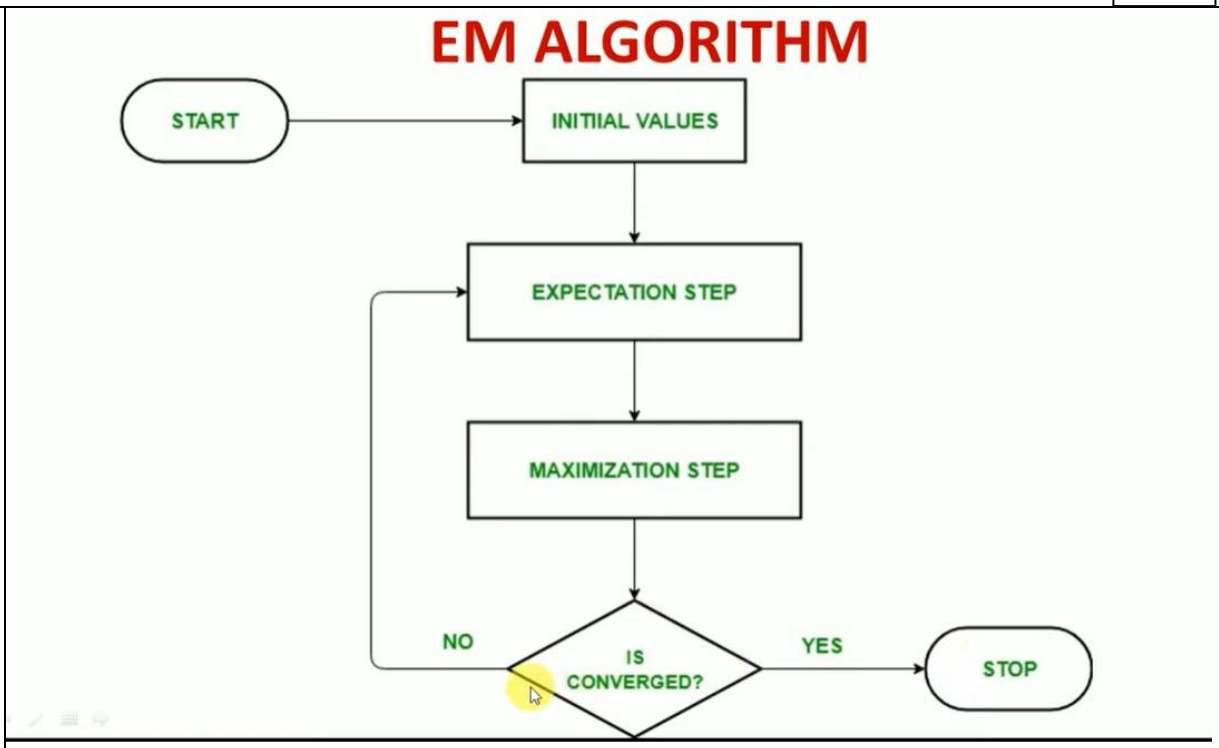
Let us understand the EM algorithm in detail.

- Initially, a set of initial values of the parameters are considered. A set of incomplete observed data is given to the system with the assumption that the observed data comes from a specific model.
- The next step is known as "Expectation" – step or *E-step*. In this step, we use the observed data in order to estimate or guess the values of the missing or incomplete data. It is basically used to update the variables.
- The next step is known as "Maximization"-step or *M-step*. In this step, we use the complete data generated in the preceding "Expectation" – step in order to update the values of the parameters. It is basically used to update the hypothesis.
- Now, in the fourth step, it is checked whether the values are converging or not, if yes, then stop otherwise repeat *step-2* and *step-3* i.e. "Expectation" – step and "Maximization" – step until the convergence occurs.

EM ALGORITHM

Algorithm:

1. Given a set of incomplete data, consider a set of starting parameters.
2. **Expectation step (E – step):** Using the observed available data of the dataset, estimate (guess) the values of the missing data.
3. **Maximization step (M – step):** Complete data generated after the expectation (E) step is used in order to update the parameters.
4. Repeat step 2 and step 3 until convergence.



Usage of EM algorithm –

- It can be used to fill the missing data in a sample.
- It can be used as the basis of unsupervised learning of clusters.
- It can be used for the purpose of estimating the parameters of Hidden Markov Model (HMM).
- It can be used for discovering the values of latent variables.

Advantages of EM algorithm –

- It is always guaranteed that likelihood will increase with each iteration.
- The E-step and M-step are often pretty easy for many problems in terms of implementation.
- Solutions to the M-steps often exist in the closed form.

Disadvantages of EM algorithm –

- It has slow convergence.
- It makes convergence to the local optima only.
- It requires both the probabilities, forward and backward (numerical optimization requires only forward probability).

Example

Let C1 and C2 be two coins.

Θ_1 be probability of getting head with C1

Θ_2 be probability of getting head with C2

Find value of Θ_1 and Θ_2 by tossing C1 and C2 for multiple times..

Toss for 5 times Choosing any of the coin randomly.

Each selected coin has to toss for 10 times.

B	H	T	T	T	H	H	T	H	T	H
A	H	H	H	H	T	H	H	H	H	H
A	H	T	H	H	H	H	H	T	H	H
B	H	T	H	T	T	T	H	H	T	T
A	T	H	H	H	T	H	H	H	T	H

Θ_1 =no of heads with C1 / Total no of flips using C1

Θ_2 =no of heads with C2 / Total no of flips using C2

If we know the coin labels the probability will be as follows:

Coin A	Coin B
	5 H, 5 T
9 H, 1 T	
8 H, 2 T	
	4 H, 6 T
7 H, 3 T	

$\Theta_1 = 24/(24+6) = 0.8$

$\Theta_2 = 9/(9+11) = 0.45$

If we don't know the identity of coin labels then we will assume or estimate the probabilities.

$\Theta_1 = 0.6$

$\Theta_2 = 0.5$

We have to use binomial distribution to find likelihood.

$$L(C)=\Theta^k (1-\Theta)^{n-k}$$

$$L(C)=\Theta^{k}(1-\Theta)^{n-k}$$

Likelihood For first coin Flips

$L(A)= 0.6^{5}(1- 0.6)^{10-5} = 0.0007963$

$L(B)= 0.5^{5}(1- 0.5)^{10-5} = 0.0009766$

$P(A)=L(A)/L(A)+L(B) = 0.0007963/(0.0007963+0.0009766)=0.45$

$P(B)=L(B)/L(A)+L(B) = 0.0009766/(0.0007963+0.0009766) = 0.55$

In similar fashion find probability of all coins with all flips. It will be as follows:

L(H): Likely no of heads L(T): Likely no of tails

											Iteration 1->:		Coin A		Coin B	
											P(A)	P(B)	L(H)	L(T)	L(H)	L(T)
B	H	T	T	T	H	H	T	H	T	H	0.45	0.55	2.2	2.2	2.8	2.8
A	H	H	H	H	T	H	H	H	H	H	0.80	0.20	7.2	0.8	1.8	0.2
A	H	T	H	H	H	H	H	T	H	H	0.73	0.27	5.9	1.5	2.1	0.5
B	H	T	H	T	T	T	H	H	T	T	0.35	0.65	1.4	2.1	2.6	3.9
											0.65	0.35	4.5	1.9	2.5	1.1

For Coin A:

$\sum L(H) = 21.3$

$\sum L(T) = 8.6$

$\Theta_1 = 21.3/(21.3+8.6)$

$= 0.71$

For Coin B:

$\sum L(H) = 11.7$

$\sum L(T) = 8.4$

$\Theta_2 = 11.7/(11.7+8.4)$

$= 0.58$

These values of Θ_1 and Θ_2 will be sent to next iteration.

The process will be continued until you get stable value of Θ_1 and Θ_2.

After 10 iteration:

$\Theta = 0.80$ and $\Theta = 0.52$

In similar fashion find probability of all coins with all flips. It will be as follows:

L(H): Likely no of heads L(T): Likely no of tails

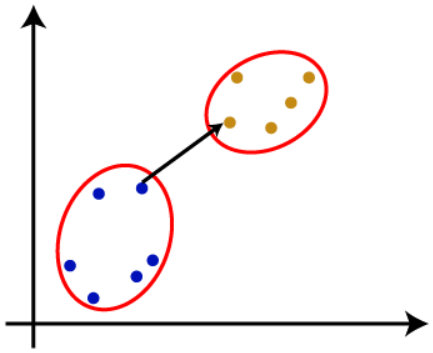
											Iteration 1->:		Coin A		Coin B	
											P(A)	P(B)	L(H)	L(T)	L(H)	L(T)
B	H	T	T	T	H	H	T	H	T	H	0.45	0.55	2.2	2.2	2.8	2.8
A	H	H	H	H	T	H	H	H	H	H	0.80	0.20	7.2	0.8	1.8	0.2
A	H	T	H	H	H	H	H	T	H	H	0.73	0.27	5.9	1.5	2.1	0.5
B	H	T	H	T	T	T	H	H	T	T	0.35	0.65	1.4	2.1	2.6	3.9
A	H	T	H	H	H	H	H	T	H	H	0.65	0.35	4.5	1.9	2.5	1.1

9 a Demonstrate linkage methods in Hierarchical Clustering **[L2][CO3]** **[6M]**

Hierarchical clustering is a clustering algorithm that builds a hierarchy of clusters. Linkage methods are used in hierarchical clustering to determine how the distance between clusters is measured and how clusters are merged. Here, I will demonstrate three commonly used linkage methods: Single Linkage, Complete Linkage, and Average Linkage.

The **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters. Consider the below image:



Single linkage, also known as the nearest-neighbor linkage, measures the distance between two clusters as the shortest distance between any two points in the two clusters.

Let's say we have the following data points and their pairwise distances:

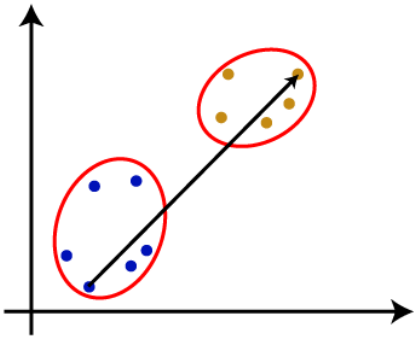
A: (1, 1)
B: (2, 2)
C: (4, 4)
D: (6, 6)

Initially, each data point is considered as a separate cluster.

Calculate the pairwise distances between all clusters:
Distance between AB: $\sqrt{((2-1)^2 + (2-1)^2)} = \sqrt{2}$
Distance between AC: $\sqrt{((4-1)^2 + (4-1)^2)} = \sqrt{18}$
Distance between AD: $\sqrt{((6-1)^2 + (6-1)^2)} = \sqrt{50}$
Merge the two closest clusters (A and B) to form a new cluster AB.
Update the pairwise distances:
Distance between AB and C: $\sqrt{((4-2)^2 + (4-2)^2)} = \sqrt{8}$
Distance between AB and D: $\sqrt{((6-2)^2 + (6-2)^2)} = \sqrt{32}$
Merge the closest clusters (AB and C) to form a new cluster ABC.
Merge the last two remaining clusters (ABC and D) to obtain the final cluster ABCD.

The dendrogram representation of the clustering process would show the steps of merging clusters based on single linkage.

2. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



Complete linkage, also known as the farthest-neighbor linkage, measures the distance between two clusters as the maximum distance between any two points in the two clusters.

Using the same data points as before:

Calculate the pairwise distances between all clusters.
Merge the two clusters with the maximum distance between their points.
Update the pairwise distances.
Repeat steps 2 and 3 until all clusters are merged into a single cluster.

The dendrogram representation of the clustering process using complete linkage would show the steps of merging clusters based on the maximum distance.

3. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.

Average linkage measures the distance between two clusters as the average distance between all pairs of points from the two clusters.

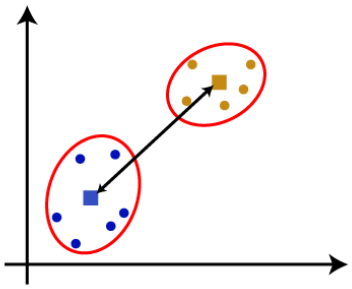
Using the same data points as before:

Calculate the pairwise distances between all clusters.

Merge the two clusters with the minimum average distance between their points.
Update the pairwise distances.
Repeat steps 2 and 3 until all clusters are merged into a single cluster.

The dendrogram representation of the clustering process using average linkage would show the steps of merging clusters based on the average distance.

4. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

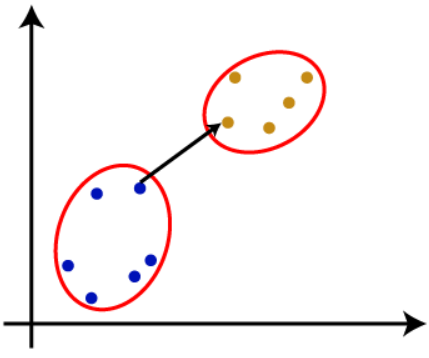
These are just examples to demonstrate the basic concepts of single linkage, complete linkage, and average linkage in hierarchical clustering. In practice, various other linkage methods and distance metrics can be used based on the specific requirements of the data and the clustering task.

b	How can we measure the distance between two clusters?	**[L1][CO3]**	**[6M]**

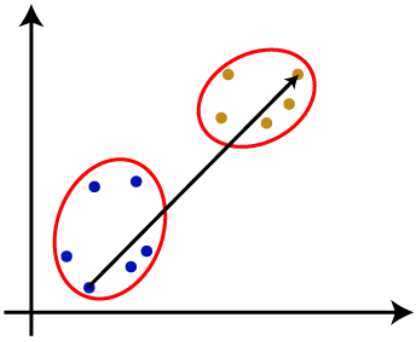
Measure for the distance between two clusters

The **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

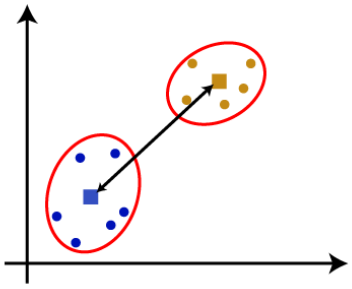
5. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters. Consider the below image:



6. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



7. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.
8. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

Here are several ways to measure the distance between two clusters in cluster analysis. Here are a few commonly used distance measures:

Euclidean Distance: It is the straight-line distance between two points in the Euclidean space. In the context of clustering, the distance between two clusters is computed as the Euclidean distance between their centroid points. The centroid of a cluster is the mean of the feature values of all the points in that cluster.
Manhattan Distance: Also known as the city block distance or L1 norm, it is the sum of the absolute differences between the coordinates of two points. In clustering, the Manhattan distance between two clusters can be calculated as the average Manhattan distance between all pairs of points from the two clusters.
Minkowski Distance: It is a generalization of Euclidean and Manhattan distances. The Minkowski distance between two points is defined as the nth root of the sum of the absolute values raised to the power of n of the differences of their coordinates. When n=1, it reduces to Manhattan distance, and when n=2, it reduces to Euclidean distance.
Mahalanobis Distance: It takes into account the covariance structure of the data and is used when the data has correlated features. The Mahalanobis distance between two clusters is calculated based on the Mahalanobis distance between their centroid points, which considers the covariance matrix of the data.
Linkage-based Distances: In hierarchical clustering, distances between clusters can be measured using different linkage methods, such as single linkage, complete linkage, and average linkage. These methods define the distance between two clusters based on the distances between their individual points.

The choice of distance measure depends on the nature of the data and the specific requirements of the

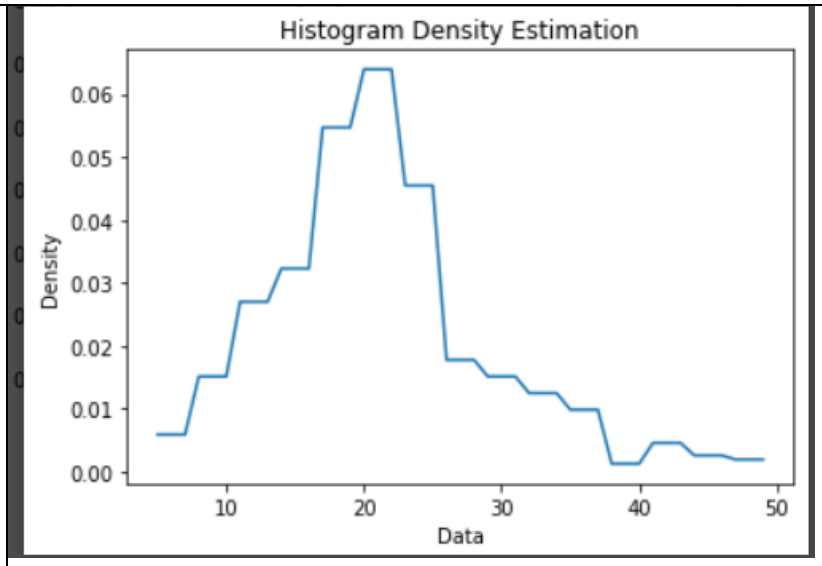
	clustering problem. Different distance measures may yield different cluster structures and interpretations, so it is important to consider the characteristics of your data and the goals of your analysis when selecting a distance measure.		
10	Summarize the following terms briefly i.K-means Clustering ii. Hierarchal Clustering	**[L2][CO5]**	**[12M]**
	Refer Q. No 4 and 7 Answers		

Course Code: 20CS0906 **R20**

UNIT-IV
NON PARAMETRIC METHODS
&
DIMENTIONALITY REDUCTION

1	a	**Define and Explain about non-parametric methods?** Algorithms that do not make strong assumptions about the form of the mapping function are called nonparametric machine learning algorithms. By not making assumptions, they are free to learn any functional form from the training data. Nonparametric methods are good when you have a lot of data and no prior knowledge, and when you don't want to worry too much about choosing just the right features . Nonparametric methods seek to best fit the training data in constructing the mapping function, whilst maintaining some ability to generalize to unseen data. As such, they are able to fit a large number of functional forms. Some more examples of popular nonparametric machine learning algorithms are: • k-Nearest Neighbors • Decision Trees like CART and C4.5 • Support Vector Machines Benefits of Nonparametric Machine Learning Algorithms: **Flexibility**: Capable of fitting a large number of functional forms. **Power**: No assumptions (or weak assumptions) about the underlying function. **Performance**: Can result in higher performance models for prediction. Limitations of Nonparametric Machine Learning Algorithms: **More data**: Require a lot more training data to estimate the mapping function. **Slower**: A lot slower to train as they often have far more parameters to train. **Overfitting**: More of a risk to overfit the training data and it is harder to explain why specific predictions are made.	**[L1][CO3]**	**[4M]**
	b	**List out advantages and limitations of non-parametric methods in ML** **Advantages for using nonparametric methods:** • They can be used to test population parameters when the variable is not normally distributed. • They can be used when the data are nominal or ordinal. • They can be used to test hypotheses that do not involve population parameters. • In some cases, the computations are easier than those for the	**[L2][CO3]**	**[8M]**

		parametric counterparts. • They are easy to understand. **Disadvantages for using nonparametric methods:** • They are ***less sensitive*** than their parametric counterparts when the assumptions of the parametric methods are met. Therefore, larger differences are needed before the null hypothesis can be rejected. • They tend to use ***less information*** than the parametric tests. For example, the sign test requires the researcher to determine only whether the data values are above or below the median , not how much above or below the Median each value is. • They are ***less efficient*** than their parametric counterparts when the assumptions of the parametric methods are met. That is, larger sample sizes are needed to overcome the loss of information. For example, the nonparametric sign test is about 60% as efficient as its parametric counterpart, the t-test. Thus, a sample size of 100 is needed for use of the sign test, compared with a sample size of 60 for use of the t-test to obtain the same results.		
2	a	State and explain various non-parametric estimation techniques? Non-parametric Density Estimations: Similar inputs have similar outputs. These are also called instance-based or memory-based learning algorithms. There are 4 Non – parametric density estimation methods: • Histogram Estimator • Naive Estimator • Kernel Density Estimator (KDE) • KNN estimator (K – Nearest Neighbor Estimator) **Histogram Estimator** It is the oldest and the most popular method used to estimate the density, where the input space is divided into equal-sized intervals called **bins**. Given the training set X = {xt}N t=1 an origin x0 and the bin width h, the histogram density estimator function is: $$\hat{p}(x) = \frac{\#\{x^t \text{ in the same bin as } x\}}{Nh}$$ Histogram estimator The density of a sample is dependent on the number of training samples present in that bin. In constructing the histogram of densities we choose the origin and the bin width, the position of origin affects the estimation near the boundaries.	**[L1][CO3]**	**[6M]**



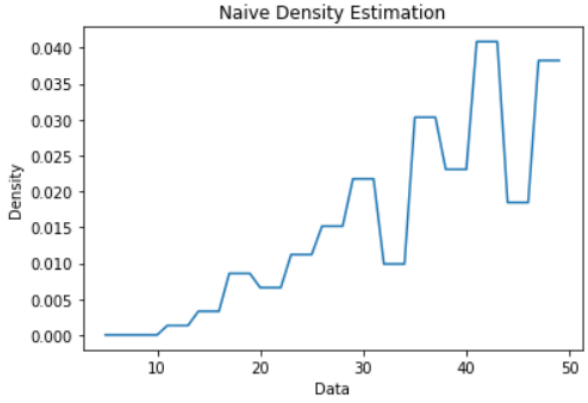
Naive Estimator

Unlike the Histogram estimator, the Naive estimator does not use the concept of origin. There is no assumption of choosing the origin. The density of the sample depends on the neighboring training samples. Given the training set X = {xt}Nt=1 and the bin width h, the Naive density estimator function is:

$$\hat{p}(x) = \frac{\#\{x - h/2 < x^t \leq x + h/2\}}{Nh}$$

Naive estimator

The values in the range of h/2 to the left and right of the sample involve the density contribution.



Naive Density Estimation plot

Kernel Density Estimator (KDE)

Kernel estimator is used to smoothen the probability distribution function (pdf) and cumulative distribution function (CDF) graphics. The kernel is nothing but a weight. Gaussian Kernel is the most popular kernel:

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right]$$

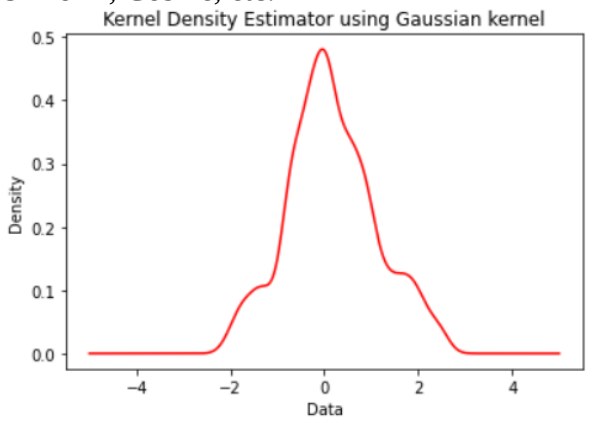
Gaussian kernel

The kernel estimator is also called Parzen Window:

$$\hat{p}(x) = \frac{1}{Nh}\sum_{t=1}^{N} K\left(\frac{x - x^t}{h}\right)$$

Kernel density estimator

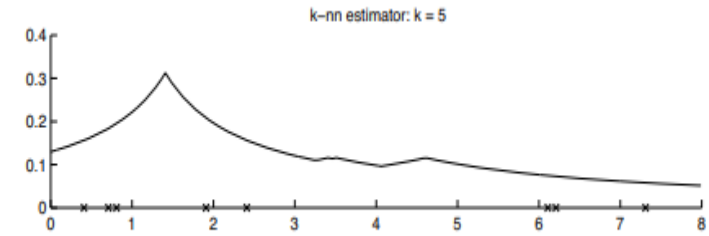
As you can observe, as |x – xt| increases that means, the training sample is far away from the given sample, and the kernel value decreases. Hence we can say that the contribution of a farther sample is less when compared to the nearest training samples. There are many more kernels: Gaussian, Rectangular, Triangular, Biweight, Uniform, Cosine, etc.



KDE plot using Gaussian Kernel

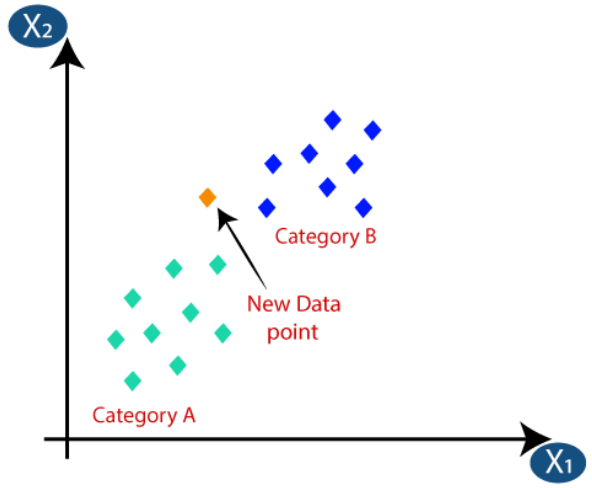
K – Nearest Neighbor Estimator (KNN Estimator)

Unlike the previous methods of fixing the bin width h, in this estimation, we fix the value of nearest neighbors k. The density of a sample depends on the value of k and the distance of the kth nearest n he Kernel e re dk(x) is t eighbor.

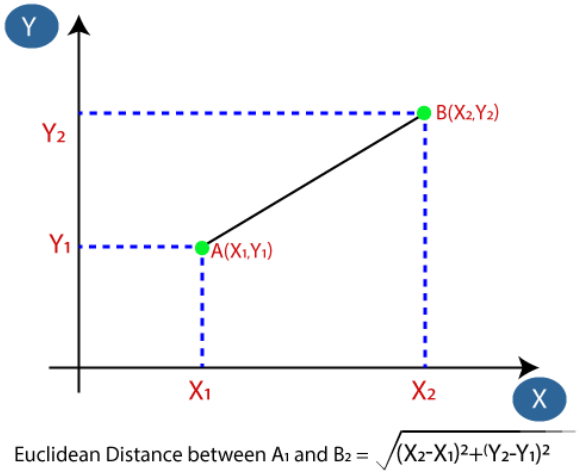


KNN Estimator

3	a	Analyze the K-Nearest Neighbor estimator? 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 neighbor 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:	**[L4][CO6]**	**[6M]**

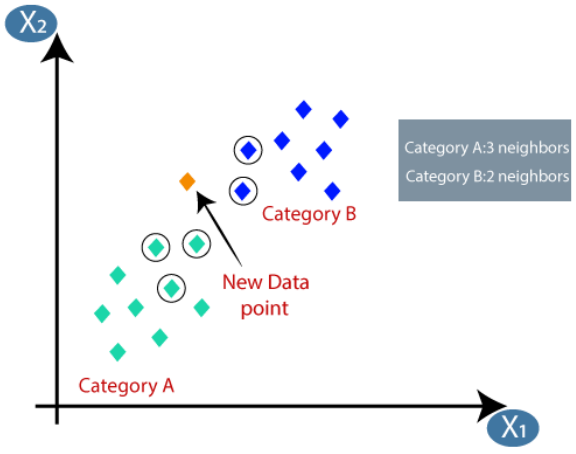


- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



- 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.



How to select the value of K in the K-NN Algorithm?
Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for

		"K", so we need to try some values to find the best out of them. The most preferred value for K is 5. o A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model. o Large values for K are good, but it may find some difficulties. Advantages of KNN Algorithm: o It is simple to implement. o It is robust to the noisy training data o It can be more effective if the training data is large. Disadvantages of KNN Algorithm: o Always needs to determine the value of K which may be complex some time. The computation cost is high because of calculating the distance between the data points for all the training samples.		
	b	**Express the non-parametric classification techniques?** Nonparametric classification techniques in machine learning are algorithms that do not make explicit assumptions about the functional form or distribution of the underlying data. These methods are flexible and can be powerful in situations where the data may not adhere to specific parametric assumptions. Here are some common nonparametric classification techniques: **K-Nearest Neighbors (KNN):** KNN is a simple and intuitive algorithm that classifies a data point based on the majority class of its nearest neighbors. It does not assume any specific form for the decision boundary and can handle complex decision boundaries. KNN is often used for both binary and multiclass classification problems. **Decision Trees**: Decision trees recursively partition the data based on different features to create a hierarchical structure of if-else rules. They can handle both categorical and numerical data and are capable of capturing non-linear relationships and interactions between features. **Random Forests**: Random forests are an ensemble learning method that combines multiple decision trees. Each tree is trained on a different subset of the data using bootstrap sampling, and the final prediction is determined by aggregating the predictions of individual trees. Random forests can handle high-dimensional data and are robust against overfitting. **Support Vector Machines (SVM):** SVMs find an optimal hyperplane that separates the data points of different classes with the largest margin. They can utilize kernel functions to implicitly map the data into higher-dimensional feature spaces, allowing them to capture complex decision boundaries. SVMs work well for both linear and non-linear classification problems. **Neural Networks**: While neural networks are often associated with parametric models, they can also be considered nonparametric depending on their architecture. Deep neural networks with multiple hidden layers have the capacity to learn complex decision boundaries and patterns in the data, making them powerful nonparametric classifiers. **Gaussian Processes (GPs):** GPs are a probabilistic approach to nonparametric classification. They model the underlying data distribution as a Gaussian process, which provides a flexible and expressive framework to make predictions. GPs can handle small to	**[L6][CO3]**	**[6M]**

		moderate-sized datasets and can provide uncertainty estimates for predictions. These are just a few examples of nonparametric classification techniques in machine learning. Each method has its own strengths and weaknesses, and the choice of algorithm depends on the specific problem and the characteristics of the dataset at hand. Regenerate response		
4	a	**Illustrate Condensed Nearest Neighbour(CNN)?** Condensed Nearest Neighbour (CNN) is a nonparametric classification technique that aims to reduce the size of the training dataset while maintaining its representativeness. It is a type of **instance-based learning algorithm** that focuses on selecting a subset of informative instances (prototypes) from the original training set to create a condensed set that can be used for classification. The CNN algorithm follows these main steps: **Initialization**: The algorithm starts with an empty set of prototypes. Iterative process: The algorithm iteratively selects instances from the original training set and adds them to the prototype set if they are misclassified. Initially, the first misclassified instance is added to the prototype set. **Nearest Neighbor Classification**: At each iteration, the misclassified instances are tested against the prototypes using a nearest neighbor classification rule. If an instance is misclassified, it is added to the prototype set. **Termination**: The iterative process continues until no more misclassified instances are found or until a convergence criterion is met. The CNN algorithm has several advantages: **Reduction of computational complexity**: By selecting a condensed set of prototypes, the algorithm reduces the computational burden of classification since it only requires comparing new instances to a smaller set of prototypes instead of the entire training set. **Improved generalization**: The condensed set of prototypes represents the most informative instances from the original training set. By focusing on these instances, CNN can potentially improve generalization performance and reduce overfitting. **Interpretability**: The condensed set of prototypes can provide insights into the characteristics of the underlying data, as they represent the most relevant instances for classification. However, CNN also has some limitations: **Sensitivity to initial selection**: The algorithm's performance can depend on the initial selection of the first misclassified instance. Different initial instances may lead to different prototype sets and, consequently, different classification results. **Sensitivity to noisy or irrelevant instances**: CNN may select noisy or irrelevant instances as prototypes, which can negatively impact	**[L3][CO4]**	**[6M]**

		classification performance. **Computational overhead during training**: While CNN reduces the computational complexity during classification, the process of selecting prototypes can be computationally expensive, especially for large datasets. Overall, Condensed Nearest Neighbor is a useful technique for reducing the size of the training dataset while preserving classification accuracy, particularly in situations where computational efficiency and interpretability are important factors.		
4	b	**Differentiate Exploratory and Confirmatory factor analysis.** **Exploratory Factor Analysis (EFA)** and Confirmatory Factor Analysis (CFA) are both techniques used in psychometrics and statistics to analyze the underlying factor structure of a set of observed variables. However, they differ in their objectives and approaches: **Exploratory Factor Analysis (EFA):** o **Objective**: EFA is used to explore and discover the latent factors that explain the relationships among a set of observed variables. It aims to identify the underlying structure and dimensions of the data. o **Hypotheses:** EFA does not rely on predefined hypotheses about the number of factors or their relationships. It allows for an open exploration of the data to uncover patterns and identify the most interpretable factor structure. o **Model Specification**: EFA is more flexible in terms of model specification. It does not require a priori specification of the factor structure and allows for the estimation of cross-loadings (variables that load on multiple factors). o **Model Fit**: EFA does not provide formal measures of model fit since it is an exploratory technique. Instead, researchers typically rely on subjective judgments, such as the interpretability of the factors and the amount of variance explained. o **Data Usage**: EFA can be used as an initial step to understand the structure of the data, generate hypotheses, and guide the development of measurement instruments or further research. **Confirmatory Factor Analysis (CFA):** o **Objective**: CFA is used to test and confirm a specific hypothesized factor structure that is derived from theory or prior research. It aims to assess how well the observed data fit the predefined factor model. o **Hypotheses**: CFA relies on specific a priori hypotheses about the number of factors, their relationships, and the loading patterns of variables on those factors. o **Model Specification**: CFA requires researchers to specify the factor structure in advance. It involves specifying the factor loadings, factor correlations, and potential measurement errors. o **Model Fit**: CFA provides formal measures of model fit, such as chi-square test, comparative fit index (CFI), root mean square error of approximation (RMSEA), etc. These measures evaluate how well the observed data fit the	**[L5][C04]**	**6M**

		hypothesized factor model. ○ **Data Usage**: CFA is typically used to assess the adequacy of a hypothesized factor structure, validate measurement instruments, and test specific theoretical constructs. In summary, EFA is an exploratory technique used for data exploration and **hypothesis generation**, whereas CFA is a confirmatory technique used for **hypothesis testing** and **model validation**. EFA allows for more flexibility in model specification and does not require predefined hypotheses, while CFA relies on specific hypotheses and requires prior specification of the factor structure.		
5	a	**List out the applications of PCA?** Applications of PCA in Machine Learning  • PCA is used to visualize multidimensional data. • It is used to reduce the number of dimensions in healthcare data. • PCA can help resize an image. • It can be used in finance to analyze stock data and forecast returns. • PCA helps to find patterns in the high-dimensional datasets. PCA is a widely used technique in data analysis and has a variety of applications, including: • Data compression: PCA can be used to reduce the dimensionality of high-dimensional datasets, making them easier to store and analyze. • Feature extraction: PCA can be used to identify the most important features in a dataset, which can be used to build predictive models. • Visualization: PCA can be used to visualize high-dimensional data in two or three dimensions, making it easier to understand and interpret. Data pre-processing: PCA can be used as a pre-processing step for other machine learning algorithms, such as clustering and classification.	**[L1][CO6]**	**[6M]**
	b	**Distinguish between parametric and non-parametric classifications?** Difference between Parametric and Non-Parametric Methods are as follows: Parametric Methods | Non-Parametric Methods Parametric Methods uses a fixed number of parameters to build the model. | Non-Parametric Methods use the flexible number of parameters to build the model. Parametric analysis is to test | A non-parametric analysis is to	**[L4][CO5]**	**[6M]**

		group means.	test medians.		
		It is applicable only for variables.	It is applicable for both – Variable and Attribute.		
		It always considers strong assumptions about data.	It generally fewer assumptions about data.		
		Parametric Methods require lesser data than Non-Parametric Methods.	Non-Parametric Methods requires much more data than Parametric Methods.		
		Parametric methods assumed to be a normal distribution.	There is no assumed distribution in non-parametric methods.		
		Parametric data handles – Intervals data or ratio data.	But non-parametric methods handle original data.		
		Here when we use parametric methods then the result or outputs generated can be easily affected by outliers.	When we use non-parametric methods then the result or outputs generated cannot be seriously affected by outliers.		
		Parametric Methods can perform well in many situations but its performance is at peak (top) when the spread of each group is different.	Similarly, Non-Parametric Methods can perform well in many situations but its performance is at peak (top) when the spread of each group is the same.		
		Parametric methods have more statistical power than Non-Parametric methods.	Non-parametric methods have less statistical power than Parametric methods.		
		As far as the computation is considered these methods are computationally faster than the Non-Parametric methods.	As far as the computation is considered these methods are computationally slower than the Parametric methods.		
		Examples: Logistic Regression, Naïve Bayes Model, etc.	Examples: KNN, Decision Tree Model, etc.		
6	a	**Discuss the Principal Component Analysis?** 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		**[L2][CO5]**	**[6M]**

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.

The PCA algorithm is based on some mathematical concepts such as:

- ariance and Covariance
- Eigenvalues and Eigen factors

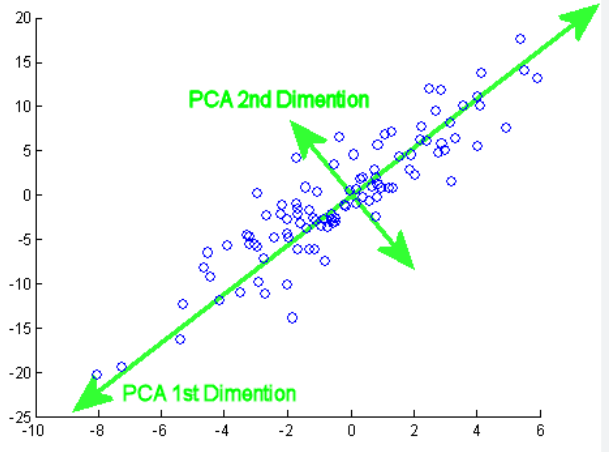
Some common terms used in PCA algorithm:

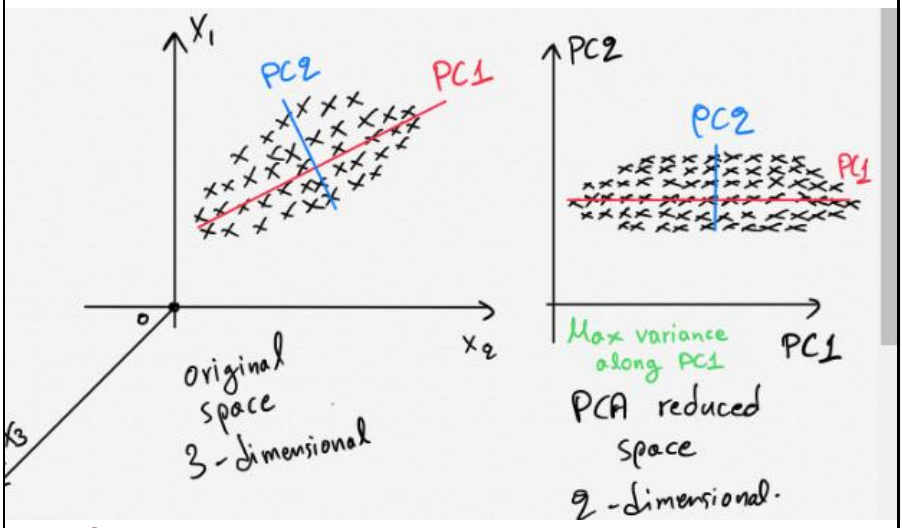
- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets 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.

Principal Components in PCA

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

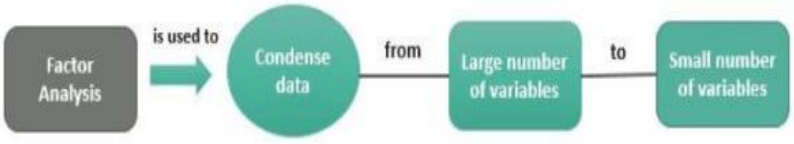
- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will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

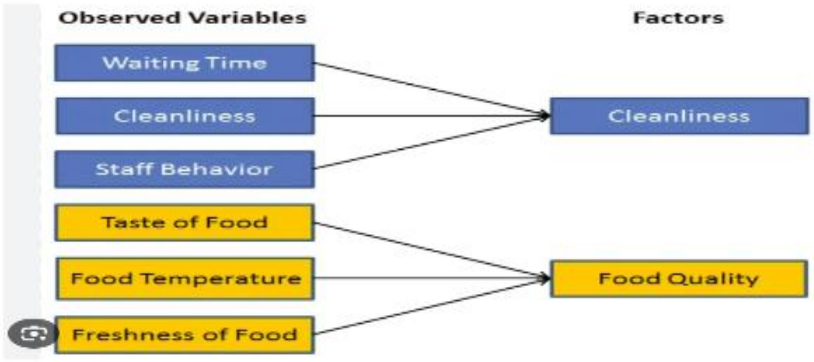


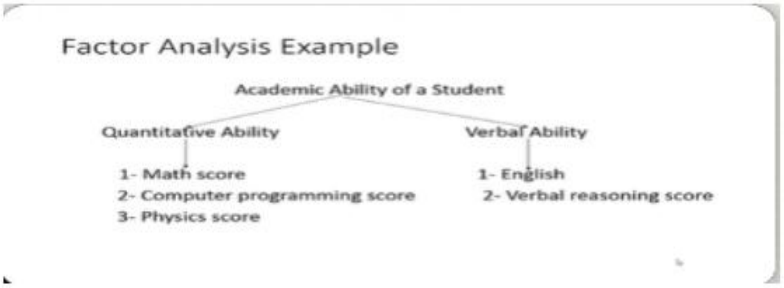


Steps for PCA algorithm

1. **Getting the dataset**
   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure**
   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data**
   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance. If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**
   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors**
   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.

		6. **Sorting the Eigen Vectors** In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*. 7. **Calculating the new features Or Principal Components** Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other. 8. **Remove less or unimportant features from the new dataset.** The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out. **Applications of Principal Component Analysis** o PCA is mainly used as the dimensionality reduction technique in various AI applications such **as computer vision, image compression, etc.** It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.		
6	b	Discuss about Factor Analysis? Factor Analytics is a special technique reducing the huge number of variables into a few numbers of factors is known as factoring of the data, and managing which data is to be present in sheet comes under factor analysis. It is completely a statistical approach that is also used to describe fluctuations among the observed and correlated variables in terms of a potentially lower number of unobserved variables called *factors.*  Factor analysis is a very effective tool for inspecting changeable relationships for complex concepts such as social status, economic status, dietary patterns, psychological scales, biology, psychometrics, personality theories, marketing, product management, operations research, finance, etc. For example:	**[L2][CO5]**	**[6M]**





The main goal of factor analysis is to reduce the dimensionality of the data by identifying a smaller number of unobservable factors that account for the common variance among the observed variables. It assumes that each observed variable is influenced by a combination of these latent factors.

The process of factor analysis involves the following steps:

1. **Data Collection**: Collect a dataset consisting of a set of observed variables (also known as indicators) that are believed to be related to the underlying factors.
2. **Factor Extraction**: Use a statistical method, such as Principal Component Analysis (PCA) or Maximum Likelihood Estimation (MLE), to extract the initial set of factors. These factors are linear combinations of the observed variables that capture the most significant sources of variance in the data.
3. **Factor Rotation**: After extracting the factors, it is often useful to rotate them to achieve a more interpretable and meaningful solution. Orthogonal rotation methods (e.g., Varimax, Quartimax) or oblique rotation methods (e.g., Promax, Oblimin) can be applied to achieve different goals, such as maximizing factor loadings or allowing for correlations among factors.
4. **Factor Interpretation**: Once the factors are extracted and rotated, the researcher interprets the meaning of each factor based on the pattern of loadings. Loadings represent the strength and direction of the relationship between each observed variable and the underlying factor. Variables with high loadings on a factor are considered to be strongly associated with that factor.

Types of factor analysis:

Exploratory factor analysis (EFA) :

It is used to identify composite inter-relationships among items and group items that are the part of uniting concepts. The Analyst can't make any prior assumptions about the relationships among factors. It is also used to find the fundamental structure of a huge set of variables. It lessens the large data to a much smaller set of summary variables. It is almost similar to the Confirmatory Factor Analysis(CFA).

Similarities are:

- Evaluate the internal reliability of an amount.
- Examine the factors represented by item sets. They presume that the factors aren't correlated.
- Investigate the grade/class of each item.

Confirmatory factor analysis (CFA) :
It is a more complex(composite) approach that tests the theory that the items are associated with specific factors. Confirmatory Factor Analysis uses a properly structured equation model to test a measurement model whereby loading on the factors allows for the evaluation of relationships between observed variables and unobserved variables.

It is similar to the Exploratory Factor Analysis.
The main difference between the two is:

- Simply use Exploratory Factor Analysis to explore the pattern.
- Use Confirmatory Factor Analysis to perform hypothesis testing.

Multiple Factor Analysis :

This type of Factor Analysis is used when your variables are structured in changeable groups. For example, you may have a teenager's health questionnaire with several points like sleeping patterns, wrong addictions, psychological health, mobile phone addiction, or learning disabilities.
The Multiple Factor Analysis is performed in two steps which are:-

- Firstly, the Principal Component Analysis will perform on each and every section of the data. Further, this can give a useful eigenvalue, which is actually used to normalize the data sets for further use.
- The newly formed data sets are going to merge into a distinctive matrix and then global PCA is performed.

Generalized Procrustes Analysis (GPA) :
The Procrustes analysis is actually a suggested way to compare then the two approximate sets of configurations and shapes, which were originally developed to equivalent to the two solutions from Factor Analysis, this technique was actually used to extend the GP Analysis so that more than two shapes could be compared in many ways. The shapes are properly aligned to achieve the target shape. Mainly GPA (Generalized Procrustes Analysis) uses geometric transformations.
Geometric progressions are :

- Isotropic rescaling,
- Reflection,
- Rotation,
- Translation of matrices to compare the sets of data.

Eigenvalues
When factor analysis going to generate the factors, each and every factor has ab associated eigenvalue which will give the total variance explained by each factor.
Usually, the factors having eigenvalues greater than 1 are useful :
Percentage of variation explained by

F1 = Eigenvalue of Factor 1/No. of Variables

Percentage of variation explained by

F2 = Eigenvalue of Factor 2/No. of Variables

FactorLoadings

In addition, factors are created with equality; some factors have more weights some have low. In a simple example, imagine your car company says Maruti Suzuki is conducting a survey includes, using – telephonic survey, physical survey, google forms, etc. for customer satisfaction and the results show the following factor loadings:

VARIABLE	F1	F2	F3
Problem 1	0.985	0.111	-0.032
Problem 2	0.724	0.008	0.167
Problem 3	0.798	0.180	0.345

Here –

F1 – Factor 1

F2 – Factor 2

F3 – Factor 3

The factors that affect the question the most (and therefore have the highest factor loadings) are bolded. Factor loadings are similar to correlation coefficients in that they can vary from -1 to 1. The closer factors are to -1 or 1, the more they affect the variable.

7		**List out and explain the various dimensionality reduction techniques**	**[L2][CO3]**	**[12M]**

The number of input features, variables, or columns present in a given dataset is known as dimensionality, and the process to reduce these features is called dimensionality reduction.

A dataset contains a huge number of input features in various cases, which makes the predictive modeling task more complicated. Because it is very difficult to visualize or make predictions for the training dataset with a high number of features, for such cases, dimensionality reduction techniques are required to use.

Dimensionality reduction technique can be defined as, *"It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information."* These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems.

It is commonly used in the fields that deal with high-dimensional data, such as **speech recognition, signal processing, bioinformatics, etc. It can also be used for data visualization, noise reduction, cluster analysis**, etc.

Approaches of Dimension Reduction

There are two ways to apply the dimension reduction technique, which are given below:

Feature Selection

Feature selection is the process of selecting the subset of the relevant features and leaving out the irrelevant features present in a dataset to build a model of high accuracy. In other words, it is a way of

selecting the optimal features from the input dataset.

Three methods are used for the feature selection:

1. Filters Methods

In this method, the dataset is filtered, and a subset that contains only the relevant features is taken. Some common techniques of filters method are:

- **Correlation**
- **Chi-Square Test**
- **ANOVA**
- **Information Gain, etc.**

2. Wrappers Methods

The wrapper method has the same goal as the filter method, but it takes a machine learning model for its evaluation. In this method, some features are fed to the ML model, and evaluate the performance. The performance decides whether to add those features or remove to increase the accuracy of the model. This method is more accurate than the filtering method but complex to work. Some common techniques of wrapper methods are:

- Forward Selection
- Backward Selection
- Bi-directional Elimination

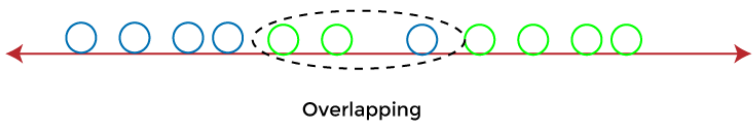
3. Embedded Methods: Embedded methods check the different training iterations of the machine learning model and evaluate the importance of each feature. Some common techniques of Embedded methods are:

- **LASSO**
- **Elastic Net**
- **Ridge Regression, etc.**

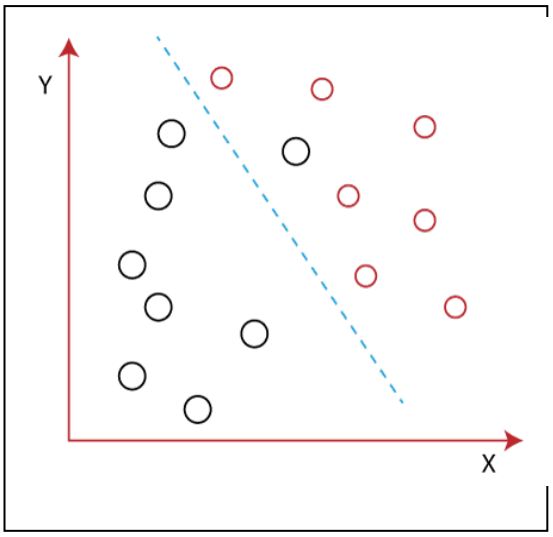
Feature Extraction:
Feature extraction is the process of transforming the space containing many dimensions into space with fewer dimensions. This approach is useful when we want to keep the whole information but use fewer resources while processing the information.

Some common feature extraction techniques are:

- Principal Component Analysis
- Linear Discriminant Analysis
- Kernel PCA
- Quadratic Discriminant Analysis

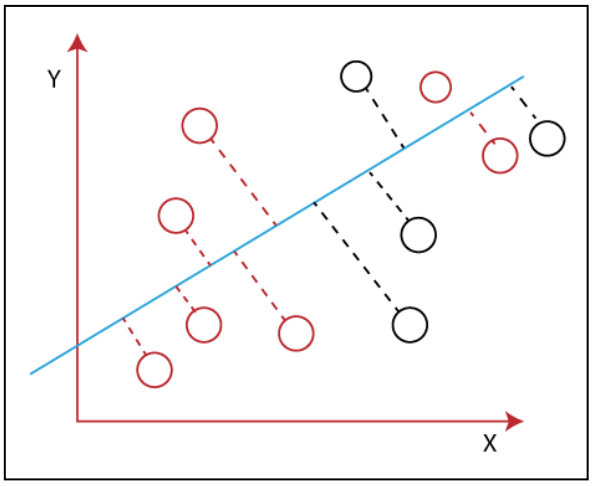
		Factor Analysis Factor analysis is a technique in which each variable is kept within a group according to the correlation with other variables, it means variables within a group can have a high correlation between themselves, but they have a low correlation with variables of other groups. We can understand it by an example, such as if we have two variables Income and spend. These two variables have a high correlation, which means people with high income spends more, and vice versa. So, such variables are put into a group, and that group is known as the **factor**. The number of these factors will be reduced as compared to the original dimension of the dataset. Auto-encoders One of the popular methods of dimensionality reduction is auto-encoder, which is a type of ANN or artificial neural network, and its main aim is to copy the inputs to their outputs. In this, the input is compressed into latent-space representation, and output is occurred using this representation. It has mainly two parts: ○ **Encoder:** The function of the encoder is to compress the input to form the latent-space representation. **Decoder:** The function of the decoder is to recreate the output from the latent-space representation.		
8	a	**Explain Linear Discriminant Analysis?** Linear Discriminant Analysis (LDA) is one of the commonly used dimensionality reduction techniques in machine learning to solve more than two-class classification problems. It is also known as Normal Discriminant Analysis (NDA) or Discriminant Function Analysis (DFA). This can be used to project the features of higher dimensional space into lower-dimensional space in order to reduce resources and dimensional costs. In this topic, "Linear Discriminant Analysis (LDA) in machine learning" Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification problems in machine learning. It is also considered a pre-processing step for modeling differences in ML and applications of pattern classification. Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems. For e.g., if we have two classes with multiple features and need to separate them efficiently. When we classify them using a single feature, then it may show overlapping.  Overlapping To overcome the overlapping issue in the classification process, we must increase the number of features regularly. Example:	**[L2][CO4]**	**[8M]**

Let's assume we have to classify two different classes having two sets of data points in a 2-dimensional plane as shown below image:



Linear Discriminant analysis is used as a dimensionality reduction technique in machine learning, using which we can easily transform a 2-D and 3-D graph into a 1-dimensional plane.

Let's consider an example where we have two classes in a 2-D plane having an X-Y axis, and we need to classify them efficiently. As we have already seen in the above example that LDA enables us to draw a straight line that can completely separate the two classes of the data points. Here, LDA uses an X-Y axis to create a new axis by separating them using a straight line and projecting data onto a new axis.



To create a new axis, Linear Discriminant Analysis uses the following criteria:

- It maximizes the distance between means of two classes.
- It minimizes the variance within the individual class.

Using the above two conditions, LDA generates a new axis in such a way that it can maximize the distance between the means of the two classes and minimizes the variation within each class.

In other words, we can say that the new axis will increase the separation between the data points of the two classes and plot them onto the new axis.

Extension to Linear Discriminant Analysis (LDA)

Linear Discriminant analysis is one of the most simple and effective methods to solve classification problems in machine learning. It has so many extensions and variations as follows:

1. Quadratic Discriminant Analysis (QDA): For multiple input

		variables, each class deploys its own estimate of variance. 2. Flexible Discriminant Analysis (FDA): it is used when there are non-linear groups of inputs are used, such as splines. Flexible Discriminant Analysis (FDA): This uses regularization in the estimate of the variance (actually covariance) and hence moderates the influence of different variables on LDA.		
	b	**Outline the various applications of Linear Discriminant Analysis?** **Applications of LDA** Some of the common real-world applications of Linear discriminant Analysis are given below: **Face Recognition** Face recognition is the popular application of computer vision, where each face is represented as the combination of a number of pixel values. In this case, LDA is used to minimize the number of features to a manageable number before going through the classification process. It generates a new template in which each dimension consists of a linear combination of pixel values. If a linear combination is generated using Fisher's linear discriminant, then it is called Fisher's face. **Medical** In the medical field, LDA has a great application in classifying the patient disease on the basis of various parameters of patient health and the medical treatment which is going on. On such parameters, it classifies disease as mild, moderate, or severe. This classification helps the doctors in either increasing or decreasing the pace of the treatment. **Customer Identification** In customer identification, LDA is currently being applied. It means with the help of LDA; we can easily identify and select the features that can specify the group of customers who are likely to purchase a specific product in a shopping mall. This can be helpful when we want to identify a group of customers who mostly purchase a product in a shopping mall. **For Predictions** LDA can also be used for making predictions and so in decision making. For example, "will you buy this product" will give a predicted result of either one or two possible classes as a buying or not. **In Learning** Nowadays, robots are being trained for learning and talking to simulate human work, and it can also be considered a classification problem. In this case, LDA builds similar groups on the basis of different parameters, including pitches, frequencies, sound, tunes, etc.	**[L1][CO6]**	**[4M]**
9	a	**Compare Multidimensionality scaling and Metric dimensionality scaling.** Multidimensional scaling (MDS) and Metric multidimensional scaling (MMDS) are both techniques used in data analysis to visualize and analyze the relationships between objects or entities based on their similarities or dissimilarities. However, there are some key differences between these two methods. Conceptual Difference:	**[L5][CO5]**	**[6M]**

		MDS: Multidimensional scaling is a general term that refers to a family of methods aimed at representing the structure of similarity or dissimilarity data in a lower-dimensional space. MDS attempts to preserve the original distances or dissimilarities between objects in the data. MMDS: Metric multidimensional scaling is a specific form of MDS that assumes the underlying distances or dissimilarities between objects are metric (i.e., satisfy the triangle inequality). It aims to find a low-dimensional representation that not only preserves the ordinal relationships between objects but also satisfies the triangle inequality. Mathematical Difference: MDS: MDS techniques, such as classical MDS or non-metric MDS, focus on finding a configuration of points in a lower-dimensional space that best approximates the pairwise dissimilarities between objects. It uses optimization algorithms to minimize the discrepancy between observed dissimilarities and distances in the reduced space. MMDS: MMDS, on the other hand, specifically deals with metric dissimilarities. It constructs a Euclidean distance matrix based on the dissimilarities and then applies classical MDS to obtain a low-dimensional representation that respects the metric properties of the data. Data Requirements: MDS: MDS can handle various types of dissimilarity measures, including ordinal, interval, or even non-metric dissimilarities. It is more flexible in terms of data requirements and can be applied to both metric and non-metric data. MMDS: MMDS assumes that the dissimilarity measures are metric, meaning they obey the triangle inequality. This assumption restricts its applicability to situations where the data can be represented by a metric space. Preserved Relationships: MDS: In MDS, the goal is to preserve the original pairwise dissimilarities or similarities as closely as possible in the lower-dimensional space. The emphasis is on preserving the ordinal relationships between objects. MMDS: MMDS aims to preserve the metric relationships between objects, in addition to the ordinal relationships. It ensures that the distances between objects in the reduced space conform to the triangle inequality.		
	b	**List out the applications of MDS.** Multidimensional scaling (MDS) has various applications across different fields. Some of the common applications of MDS include: Psychology and Cognitive Science: MDS is widely used in psychology and cognitive science to understand and visualize how individuals perceive and organize information. It can be used to study mental representations of concepts, semantic relationships, and similarity judgments. Marketing and Consumer Research: MDS is used to analyze consumer preferences, brand positioning, and product mapping. By representing consumer perceptions in a lower-dimensional space, MDS helps identify market segments, understand product similarities, and optimize marketing strategies. Social Sciences: MDS is applied in social sciences, such as sociology and political science, to explore and map social structures and relationships. It helps understand social networks, analyze intergroup relations, and visualize social distance or similarity between individuals or groups. Geographic Information Systems (GIS): MDS is utilized in GIS	**[L1][CO6]**	**[6M]**

applications to visualize and analyze spatial relationships. It can be used to create maps or visualizations of geographic data based on the perceived similarities or dissimilarities between locations, such as in crime mapping or transportation planning.

Image and Pattern Recognition: MDS is employed in computer vision and pattern recognition tasks. It helps visualize and analyze similarities or dissimilarities between images or patterns, facilitating tasks like image retrieval, object recognition, and clustering.

Marketing Research: MDS is used in marketing research to understand and visualize consumer preferences and perceptions. It helps businesses identify market segments, study brand associations, and analyze customer satisfaction.

Environmental Science: MDS is applied in environmental science to analyze and visualize similarities or dissimilarities between ecological communities, habitats, or species. It aids in studying biodiversity, species distributions, and ecological relationships.

Human-Computer Interaction: MDS is utilized in human-computer interaction (HCI) research to understand user preferences, usability evaluations, and interface design. It helps designers and researchers map user perceptions and preferences in a lower-dimensional space.

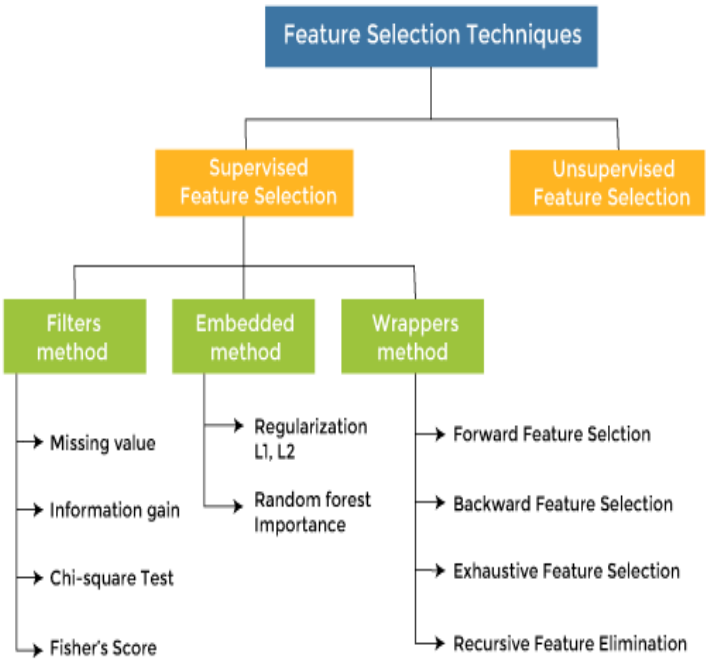
10 a Differentiate Feature selection and Feature Extraction. **[L2][CO3]** **[6M]**

Sl.No	Feature Selection	Feature Extraction
1.	Selects a subset of relevant features from the original set of features.	Extracts a new set of features that are more informative and compact.
2.	Reduces the dimensionality of the feature space and simplifies the model.	Captures the essential information from the original features and represents it in a lower-dimensional feature space.
3.	Can be categorized into filter, wrapper, and embedded methods.	Can be categorized into linear and nonlinear methods.
4.	Requires domain knowledge and feature engineering.	Can be applied to raw data without feature engineering.

		5.	Can improve the model's interpretability and reduce overfitting.	Can improve the model performance and handle nonlinear relationships.		
		6.	May lose some information and introduce bias if the wrong features are selected.	May introduce some noise and redundancy if the extracted features are not informative.		
10	b	**Explain about Subset Selection Techniques.**			**[L4][CO4]**	**[6M]**

Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features

feature Selection as, "***It is a process of automatically or manually selecting the subset of most appropriate and relevant features to be used in model building***." Feature selection is performed by either including the important features or excluding the irrelevant features in the dataset without changing them.



Wrapper Methods

In wrapper methodology, selection of features is done by considering it as a search problem, in which different combinations are made, evaluated, and compared with other combinations. It trains the algorithm by using the subset of features iteratively.

Types of Wrapper methods

- Forward selection
- Backward Selection
- Exhaustive selection

- Recursive Selection

Filter Methods

In Filter Method, features are selected on the basis of statistics measures. This method does not depend on the learning algorithm and chooses the features as a pre-processing step.

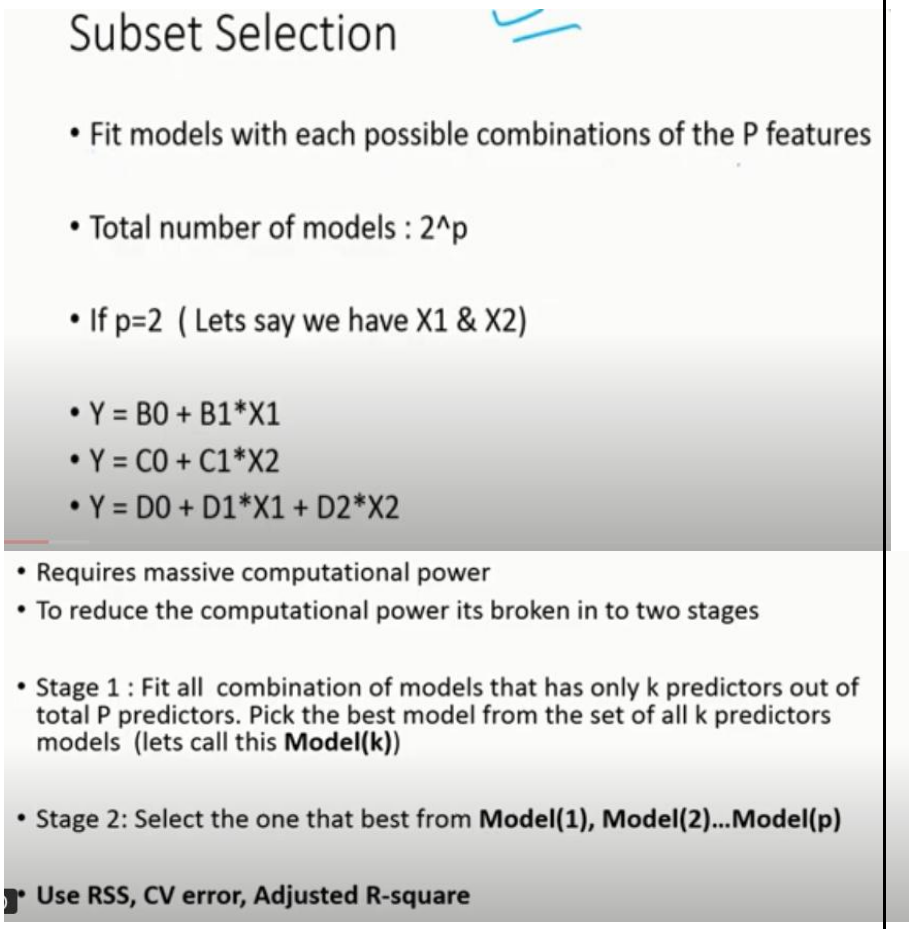
Tyeps filter methods are

- Missing value
- Information gain
- Chi-square Test
- Fisher's Score

Embedded Methods

Embedded methods combined the advantages of both filter and wrapper methods by considering the interaction of features along with low computational cost. These are fast processing methods similar to the filter method but more accurate than the filter method.

- Regularization
- Random Forest Importance



- P >20 it is almost impossible to use best subset selection

  - **Forward Stepwise Approach**
  - **Backward Stepwise Approach**
  - **Hybrid**

Forward Stepwise

- Start with a null model
- Add predictors to the model **one at a time**. Choose the best model among the bests for each k based on RSS
- **If a variable is retained it never drops from the model**

	Subset Selection	Forward Stepwise
One variable	X1	**X1**
Two Variables	X1 X3	**X1** X2
Three Variables	X1 X3 X4	**X1 X2** X4
Four Variables	X1 X2 X3 X5	**X1 X2 X4** X5

Backward

- It is the reverse of forward : Start with all predictors and then drop one at a time and then select the best model

	Backward Stepwise	Forward Stepwise
	X1 **X2** X3 X4 X5	**X1**
	X1 X3 **X4** X5	**X1** X2
	X1 **X3** X5	**X1 X2** X4
	X1 **X 5**	**X1 X2 X4** X5
	X1	**X1 X2 X4 X3 X5**

Course Code: 20CS0906 **R20**

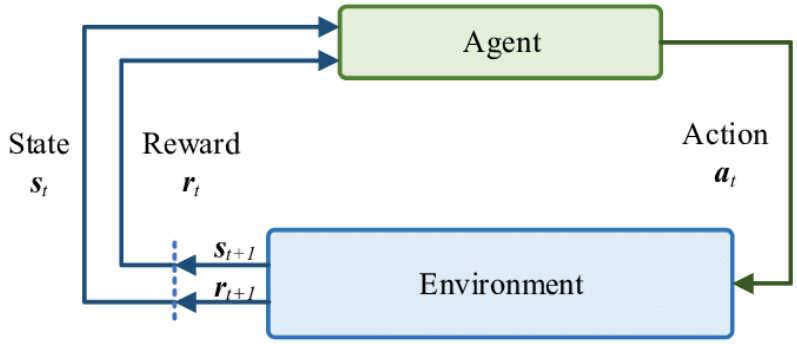
UNIT –V

REINFORCEMENT LEARNING

1	**a**	**Define and explain about the Reinforcement learning.**	**[L2][CO4]**	**[6M]**

- 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 labelled data, unlike supervised learning.
- Since there is no labelled data, so the agent is bound to learn by its experience only.

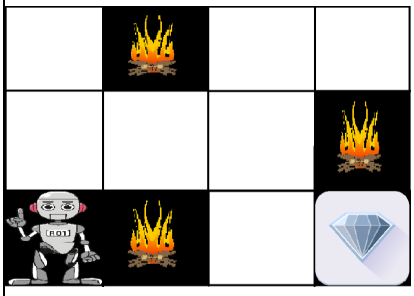
- "Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."



Reinforcement learning uses algorithms that learn from outcomes and decide which action to take next. After each action, the algorithm receives feedback that helps it determine whether the choice it made was correct, neutral or incorrect. It is a good technique to use for automated systems that have to make a lot of small decisions without human guidance.

Example:

The problem is as follows: We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The following problem explains the problem more easily.



The above image shows the robot, diamond, and fire. The goal of the robot is to get the reward that is the diamond and avoid the hurdles that are fired. The robot learns by trying all the possible

paths and then choosing the path which gives him the reward with the least hurdles. Each right step will give the robot a reward and each wrong step will subtract the reward of the robot. The total reward will be calculated when it reaches the final reward that is the diamond.

Main points in Reinforcement learning –

- Input: The input should be an initial state from which the model will start
- Output: There are many possible outputs as there are a variety of solutions to a particular problem
- Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
- The model keeps continues to learn.
- The best solution is decided based on the maximum reward.

	b	**Compare unsupervised learning and Reinforcement learning.**	**[L4][CO5]**	**[6M]**

Criteria	**Unsupervised ML**	**Reinforcement ML**
Definition	Trained using unlabelled data without any guidance.	Works on interacting with the environment
Type of data	Unlabelled data	No – predefined data
Type of problems	Association and Clustering	Exploitation or Exploration
Supervision	No supervision	No supervision
Algorithms	K – Means, C – Means, Apriori	Q – Learning, SARSA
Aim	Discover underlying patterns	Learn a series of action
Application	Recommendation System, Anomaly Detection	Self Driving Cars, Gaming, Healthcare

2	**a**	**Explain various types of reinforcement learning techniques.**	**[L2][CO4]**	**[6M]**
		Types of Reinforcement: There are two types of Reinforcement: 1. **Positive:** Positive Reinforcement is defined as when an event, occurs due to a particular behavior, increases the strength and the frequency of the behavior. In other words, it has a positive effect on behavior. Advantages of reinforcement learning 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 2. **Negative:** Negative Reinforcement is defined as strengthening of behavior because a negative condition is stopped or avoided. Advantages of reinforcement learning: • Increases Behavior • Provide defiance to a minimum standard of performance • It Only provides enough to meet up the minimum behavior		
	b	**List out the advantages and disadvantages of Reinforcement Learning.**	**[L1][CO1]**	**[6M]**
		Advantages of Reinforcement Learning: 1. Flexibility and Adaptability: Reinforcement learning allows agents to adapt to changing environments and learn optimal strategies without explicitly programmed rules. It can handle complex and dynamic scenarios where traditional rule-based approaches may fail. 2. Learning from Experience: Reinforcement learning agents learn by interacting with the environment and receiving feedback in the form of rewards or punishments. This experiential learning enables agents to discover optimal policies by exploring different actions and observing their consequences. 3. Handling Uncertainty: Reinforcement learning is capable of dealing with uncertain and partially observable environments. Agents can learn to make decisions based on probabilistic models, effectively managing uncertainty and making near-optimal decisions. 4. Generalization: Reinforcement learning algorithms can generalize knowledge learned from one task or environment to new, unseen situations. This ability to transfer knowledge allows agents to apply learned policies to similar problems, reducing the need for retraining from scratch. 5. Autonomous Decision Making: Reinforcement learning enables autonomous decision making without the need for human intervention. This is particularly useful in domains where human expertise is limited or costly to acquire. **Disadvantages of Reinforcement Learning**: 1.**High Sample Complexity**: Reinforcement learning often requires a large number of interactions with the environment to achieve good performance. The agent must explore and gather sufficient data to learn effective policies, which can be time-consuming and inefficient in domains with lengthy feedback cycles or high-dimensional state spaces. 2.**Exploration-Exploitation Trade-off**: Finding an optimal policy requires a balance between exploration (trying new actions to learn) and exploitation (taking the best-known actions to maximize rewards). Striking the right balance can be challenging, as excessive exploration can hinder performance, while exploitation alone may lead to suboptimal solutions. 3.**Reward Design**: Designing suitable reward functions that guide		

		the learning process is a crucial aspect of reinforcement learning. The reward signal should effectively capture the desired behaviour and provide clear guidance to the agent. However, designing appropriate reward functions can be complex and subjective, leading to biases or unintended consequences. 4**. Lack of Safety**: Reinforcement learning agents typically optimize for a specific objective without considering potential risks or safety concerns. If the reward signal is not carefully defined, agents may discover unintended ways to achieve high rewards that are not aligned with human values or safety requirements. 5.**Limited Explain ability**: Reinforcement learning models often lack interpretability, making it challenging to understand and explain the decision-making process. This limitation can hinder trust and acceptance, especially in critical applications where explanations are crucial, such as healthcare or finance		
3	a	**List the applications of Reinforcement Learning and explain it.**	**[L2][CO6]**	**[6M]**
		RL has numerous applications across various domains. Here are some notable applications of reinforcement learning: 1. **Game Playing**: RL has been highly successful in game-playing scenarios. For instance, AlphaGo, developed by Deep Mind, used RL to defeat world champions in the board game Go. RL has also been applied to games like chess, poker, and video games, achieving remarkable results. 2. **Robotics**: RL enables robots to learn tasks and behaviours autonomously. Robots can learn to grasp objects, walk, navigate through environments, and perform complex tasks using reinforcement learning algorithms. 3. **Autonomous Vehicles**: Reinforcement learning can be employed to train autonomous vehicles to make decisions in dynamic and uncertain environments. RL helps in tasks like lane following, collision avoidance, and efficient route planning. 4. **Resource Management**: RL can optimize resource allocation in various domains, such as energy management, traffic signal control, and inventory management. It learns to make decisions that maximize efficiency, minimize costs, or optimize performance based on feedback and rewards. 5. **Recommendation Systems**: Reinforcement learning can enhance recommendation systems by learning user preferences and making personalized recommendations. By incorporating user feedback and reinforcement signals, RL algorithms can adapt and improve the recommendations over time. 6. **Healthcare:** RL can assist in optimizing treatment plans and personalized medicine. It can learn from patient data and clinical trials to suggest appropriate interventions, drug dosages, and treatment schedules. 7. **Finance**: RL can be applied to algorithmic trading, portfolio management, and risk analysis. RL algorithms can learn to make trading decisions by analysing market data, optimizing portfolios, and adapting to changing market conditions. 8. **Industrial Control Systems**: Reinforcement learning can optimize complex industrial processes by learning control policies that maximize efficiency, reduce downtime, and		

minimize resource consumption. It has applications in areas like manufacturing, power systems, and chemical processes.

9. **Natural Language Processing**: RL algorithms have been used in natural language processing tasks such as dialogue systems, machine translation, and text generation. RL can improve the performance of language models by learning to generate coherent and contextually appropriate responses.
10. **Education**: Reinforcement learning can be employed in adaptive learning systems and intelligent tutoring systems. It can adapt the learning experience based on the student's progress, providing personalized feedback and optimizing learning outcomes.

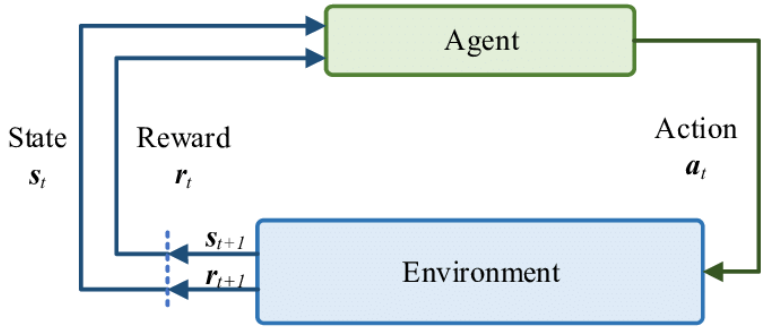
	b	**Differentiate the Reinforcement learning and Supervised learning.**	**[L4][CO5]**	**[6M]**

Criteria	**Supervised ML**	**Reinforcement ML**
Definition	Learns by using labelled data	Works on interacting with the environment
Type of data	Labelled data	No – predefined data
Type of problems	Regression and classification	Exploitation or Exploration
Supervision	Extra Supervision	No supervision
Algorithms	Linear Regression, Logistic Regression, SVM, KNN etc.	Q – Learning, SARSA
Aim	Calculate outcomes	Learn a series of action
Application	Risk Evaluation, Forecast Sales	Self Driving Cars, Gaming, Healthcare

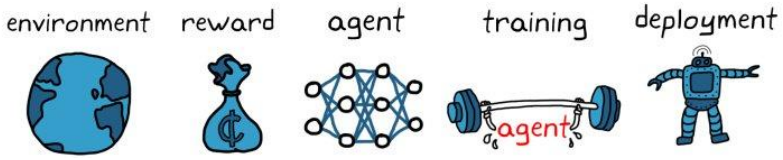
4		**Analyze the working process of Reinforcement learning.**	**[L4][CO3]**	**[12M]**

- 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 labelled data, unlike supervised learning.

- Since there is no labelled data, so the agent is bound to learn by its experience only.
- "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."



Agent – is the sole decision-maker and learner
Environment – a physical world where an agent learns and decides the actions to be performed
Action – a list of action which an agent can perform
State – the current situation of the agent in the environment
Reward – For each selected action by agent, the environment gives a reward. It's usually a scalar value and nothing but feedback from the environment
Policy – the agent prepares strategy (decision-making) to map situations to actions.
Value Function – The value of state shows up the reward achieved starting from the state until the policy is executed
Model – Every RL agent doesn't use a model of its environment. The agent's view maps state-action pairs probability distributions over the states



Elements of Reinforcement Learning

Reinforcement learning elements are as follows:

1. Policy
2. Reward function
3. Value function
4. Model of the environment

Policy: Policy defines the learning agent behavior for given time period. It is a mapping from perceived states of the environment to actions to be taken when in those states.
Reward function: Reward function is used to define a goal in a reinforcement learning problem.A reward function is a function that provides a numerical score based on the state of the environment
Value function: Value functions specify what is good in the long run. The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state.

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:** 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.

represent the agent state:

We can represent the agent state using the **Markov State** that contains all the required information from the history. The State St is Markov state if it follows the given condition:

$P[S_t+1 \mid S_t] = P[S_t+1 \mid S_1, ..., S_t]$

Markov Decision Process or MDP, is used to **formalize the reinforcement learning problems**. If the environment is completely observable, then its dynamic can be modeled as a **Markov Process**

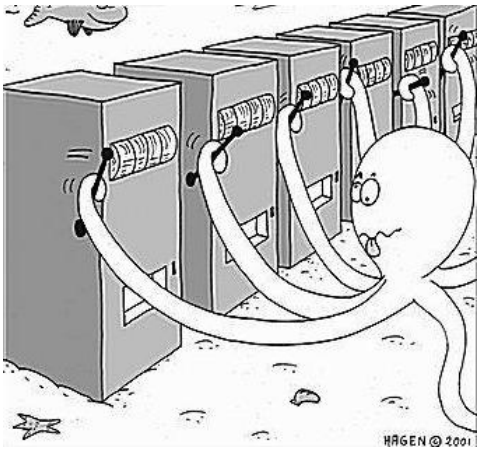
Markov Property:

It says that "If the agent is present in the current state S1, performs an action a1 and move to the state s2, then the state transition from s1 to s2 only depends on the current state and future action and states do not depend on past actions, rewards, or states."

5	a	**Explain in detail about Single State Case: K-Armed Bandit problem**	**[L2][CO4]**	**[6M]**

A bandit is defined as someone who steals your money. A one-armed bandit is a simple slot machine wherein you insert a coin into the machine, pull a lever, and get an immediate reward. But why is it called a bandit? It turns out all casinos configure these slot machines in such a way that all gamblers end up losing money!

A multi-armed bandit is a complicated slot machine wherein instead of 1 , there are several levers which a gambler can pull, with each lever giving a different return. The probability distribution for the reward corresponding to each lever is different and is unknown to the gambler.



The task is to identify which lever to pull in order to get maximum reward after a given set of trials. This problem statement is like a single step Markov decision process. Each arm chosen is equivalent to an action, which then leads to an immediate reward.

There are infinite ways to build multi-armed bandit agents. Pure-exploration agents are completely random. They focus on exploration and never exploit any of the data they have gathered.

As the name suggests, pure-exploitation agents would always choose the best possible solution since they already have all the data to exploit. Being paradoxical by nature, this makes them possible in theory only and equally bad as the random agents.

There are three most popular MAB agents that are neither completely random nor impossible to deploy in practice.

Epsilon-greedy

Epsilon-greedy multi-armed bandits take care of the balance between exploration and exploitation by adding the exploration value (epsilon) to the formula. In case epsilon equals 0.3, the agent will explore random possibilities 30% of the time and focus on exploiting the best average outcome the other 70% of time.

A decay parameter is also included and it reduces epsilon over time. When constructing the agent, you may decide to remove epsilon from the equation after a certain amount of time or actions taken. This will cause the agent to focus solely on exploitation of the data it already gathered and remove random tests from the equation.

Upper confidence bound

These multi-armed bandits are quite similar to the epsilon-greedy agents. However, the key difference between the two is an additional parameter included when building upper confidence bound bandits. A variable is included in the equation that forces the bandit to focus on the least-explored possibilities from time to time. For example, if you have options A, B, C, and D, and option D has only been chosen ten times, while the rest have been selected hundreds of times, the bandit will purposefully select D to explore the outcomes.

In essence, upper confidence bound agents sacrifice some of the resources to avoid a huge yet quite improbable mistake of never

exploring the best possible outcome.

$$A_t = \text{argmax}_a \left(Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}}\right)$$

Exploit Explore

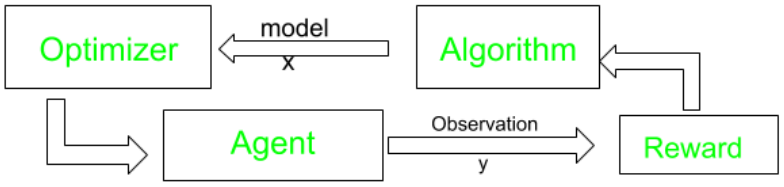
t = timesteps $N_t(a)$ = no. of times action (a) is taken

Thompson Sampling (Bayesian)

This agent is built quite disparately from the two we explored above. Being by far the most advanced bandit solution on the list, an essay-length article would be required to explain how it works with sufficient detail. However, we can opt for a less intricate analysis instead.

The Thompson bandit is able to trust certain choices more or less based on how often they were picked in the past. For example, we have option A that the agent chose a hundred times with an average reward ratio of 0.71. We also have option B that was chosen a total of twenty times with the same average reward radio as option A.

In this case, the Thompson sampling agent would go for option A a bit more often. This is because a higher frequency of choosing a path tends to yield lower average rewards. The agent assumes option A is more trustworthy and option B would have lower average outcomes if it was chosen more frequently.



	b	**What are the Elements involved in Reinforcement Learning using Markov Decision Process (MDP)?**	**[L1][CO4]**	**[6M]**

1. **Agent:** A reinforcement learning agent is the entity which we are training to make correct decisions. For example, a robot that is being trained to move around a house without crashing.
2. **Environment:** The environment is the surroundings with which the agent interacts. For example, the house where the robot moves. The agent cannot manipulate the environment; it can only control its own actions. In other words, the robot can't control where a table is in the house, but it can walk around it.
3. **State:** The state defines the current situation of the agent This can be the exact position of the robot in the house, the alignment of its two legs or its current posture. It all depends on how you address the problem.
4. **Action:** The choice that the agent makes at the current time step. For example, the robot can move its right or left leg, raise its arm, lift an object or turn right/left, etc. We know the set of actions (decisions) that the agent can perform in advance.
5. **Policy:** A policy is the thought process behind picking an action. In practice, it's a probability distribution assigned to the set of actions. Highly rewarding actions will have a high probability and vice versa. If an action has a low probability, it doesn't mean it won't be picked at all. It's just less likely to

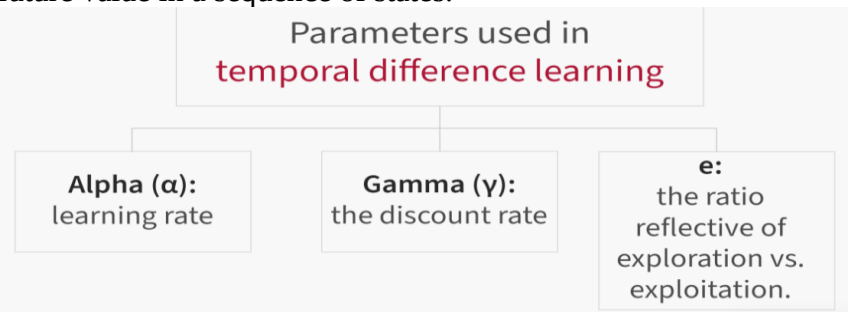
		be picked. $$\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$$ $$\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$$ **Discount (γ)** The variable $\gamma \in [0, 1]$ in the figure is the discount factor. The intuition behind using a discount is that there is no certainty about the future rewards. While it is important to consider future rewards to increase the Return, it's also equally important to limit the contribution of the future rewards to the Return (since you can't be 100 percent certain of the future.). **Return (G_t)** The return G_t is the total discounted reward from time-step t $$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$		
6	a	**Explain Model-Based Learning with an example.**	**[L2][CO4]**	**[6M]**
		"Model" is one of those terms that gets thrown around a lot in machine learning (and in scientific disciplines more generally), often with a relatively vague explanation of what we mean. Fortunately, in reinforcement learning, a model has a very specific meaning: it refers to the different dynamic states of an environment and how these states lead to a reward. Model-based RL entails constructing such a model. Model-free RL, conversely, forgoes this environmental information and only concerns itself with determining what action to take given a specific state. As a result, model-based RL tends to emphasize planning, whereas model-free RL tends to emphasize learning (that said, a lot of learning also , In general, the core function of RL algorithms is to determine a policy that maximizes this long-term return, though there are a variety of different methods and algorithms to accomplish this. And again, the major difference between model-based and model-free RL is simply that the former incorporates a model of the agent's environment, specifically one that influences how the agent's overall policy is determined goes on in model-based RL).  A flow diagram of model-based RL		
	b	**Distinguish between model based learning and temporal difference learning.**	**[L5][CO5]**	**[6M]**
		Model-based learning and temporal difference (TD) learning are two approaches to reinforcement learning, which is a branch of machine learning concerned with learning optimal behavior through		

interaction with an environment. Here are the key differences between these two approaches:

1. Learning Approach:
    - Model-Based Learning: In model-based learning, the agent learns a model of the environment, including its dynamics and transition probabilities. It then uses this model to plan and make decisions about its actions.
    - Temporal Difference Learning: TD learning is a model-free learning approach. Instead of explicitly learning the dynamics of the environment, the agent directly estimates the value or utility of states or state-action pairs through trial-and-error experience.
2. Planning vs. Direct Learning:
    - Model-Based Learning: With a learned model of the environment, model-based learning algorithms can perform planning, which involves simulating different sequences of actions and estimating their outcomes to make decisions.
    - Temporal Difference Learning: TD learning algorithms do not perform explicit planning. They learn from direct experience by updating value estimates based on the observed rewards and the estimated values of subsequent states.
3. Exploration vs. Exploitation:
    - Model-Based Learning: Model-based learning can incorporate explicit exploration strategies based on uncertainty about the model. By actively exploring the environment, the agent can improve its model and make more informed decisions.
    - Temporal Difference Learning: TD learning algorithms typically use exploration strategies to balance exploration and exploitation but do not rely on a learned model to guide their exploration. Common approaches include epsilon-greedy or softmax exploration.
4. Sample Efficiency:
    - Model-Based Learning: Model-based learning algorithms can achieve higher sample efficiency since they can leverage their learned model to plan and simulate potential outcomes before executing actions.
    - Temporal Difference Learning: TD learning algorithms might require more samples to converge to an optimal policy since they rely on direct interaction with the environment to estimate values.
5. Computational Complexity:
    - Model-Based Learning: Model-based learning can be computationally more demanding because it involves learning and maintaining a model of the environment, as well as performing planning and simulation.
    - Temporal Difference Learning: TD learning algorithms are often computationally simpler since they do not require explicit modeling or planning.

		They update value estimates based on observed rewards and subsequent state values. In practice, the choice between model-based learning and TD learning depends on the specific problem, available computational resources, and the trade-off between sample efficiency and computational complexity.		
7	**a**	**Illustrate in detail about K-Armed Bandit in reinforcementlearning.**	**[L3][CO4]**	**[6M]**
		In reinforcement learning, the K-Armed Bandit problem is a classic and fundamental setup that helps us understand the trade-off between exploration and exploitation. The problem is named after a hypothetical scenario in which an agent faces K different slot machines (arms) in a casino, each with an unknown probability of winning (payoff). The agent's goal is to maximize its total reward over a series of plays. Formally, the K-Armed Bandit problem is defined as follows: 1. There are K arms, indexed from 1 to K. 2. Each arm i has an associated probability distribution of the payoff qi, which is unknown to the agent. 3. At each time step t, the agent selects an arm At to pull based on its current knowledge. 4. After selecting an arm, the agent receives a reward Rt from a distribution that corresponds to the arm At. 5. The agent's goal is to maximize the expected total reward over a fixed number of time steps or an infinite time horizon. The main challenge in the K-Armed Bandit problem is to balance the exploration of potentially better arms (exploration) with exploiting the arms that are currently believed to be the best (exploitation). Initially, the agent lacks information about the arms' true values, so it needs to explore different arms to gather data and refine its estimates. As it gains more knowledge, it should start exploiting the arms that seem to have higher expected rewards. One popular algorithm for solving the K-Armed Bandit problem is the epsilon-greedy algorithm: 1. Initialize action-value estimates for each arm $Q(a)$ to 0 and initialize a counter $N(a)$ to 0 (denoting the number of times arm a has been pulled). 2. Set a small positive value for ϵ, typically between 0 and 1, which represents the exploration rate. 3. At each time step t: a. With probability ϵ, choose a random arm (explore). b. Otherwise, choose the arm with the highest estimated value $Q(a)$ (exploit). c. Pull the chosen arm and receive the reward Rt. d. Update the action-value estimate for the chosen arm using the formula: ←sum of rewards from arm number of times arm has been pulled$Q(a)$←number of times arm a has been pulledsum of rewards from arm a e. Increment the counter $N(a)$ for the chosen arm.		

		Over time, as the agent gathers more data through exploration, the action-value estimates become more accurate, and the agent can transition from exploration to exploitation, favoring the arms with higher estimated values. The epsilon-greedy algorithm is simple and effective for the K-Armed Bandit problem, but there are more sophisticated algorithms, such as UCB (Upper Confidence Bound) and Thompson sampling, which can provide better exploration-exploitation trade-offs and faster convergence rates, especially in more complex reinforcement learning scenarios.		
	b	**Describe Exploration and Exploitation strategies in temporal difference learning.**	**[L1][CO4]**	**[6M]**
		In the context of temporal difference (TD) learning, exploration and exploitation refer to two fundamental aspects of the learning process. TD learning is a type of reinforcement learning algorithm used to learn value functions or policies in Markov Decision Processes (MDPs). Exploration: Exploration involves actively seeking new and unfamiliar states or actions to gather more information about the environment. The goal of exploration is to discover the underlying structure of the MDP, identify optimal or near-optimal policies, and avoid premature convergence to suboptimal solutions. In TD learning, exploration is typically achieved by taking random or stochastic actions, allowing the agent to visit different states and observe their outcomes. Exploitation: Exploitation refers to utilizing the knowledge or information gained from past experiences to make decisions that maximize the expected reward. The exploitation phase exploits the learned value estimates or policies to choose actions that are expected to yield the highest immediate or long-term rewards. Exploitation is crucial for the agent to leverage its learned knowledge and make efficient decisions based on what it has already learned. Balancing Exploration and Exploitation: A critical challenge in TD learning is finding the right balance between exploration and exploitation. If an agent focuses too much on exploration, it may waste time in unproductive states or take suboptimal actions, leading to slow learning progress. On the other hand, excessive exploitation can lead to the agent being stuck in local optima and missing out on better solutions in other parts of the environment. To address this challenge, various exploration-exploitation strategies can be employed, such as: Epsilon-Greedy: The agent chooses the action with the highest estimated value most of the time (exploitation), but occasionally selects a random action with a small probability epsilon (exploration). Upper Confidence Bound (UCB): Actions are chosen based on an exploration bonus calculated using the estimated value and a measure of uncertainty. This encourages the agent to explore actions that have uncertain or potentially high rewards. Thompson Sampling: The agent maintains a distribution over		

		possible value functions and samples from it to choose actions. This balances exploration and exploitation by considering the uncertainty in value estimates. Softmax Action Selection: Actions are chosen probabilistically based on their estimated values, with the probabilities being proportional to the exponential of the estimated values. This allows for a controlled exploration based on the relative values of different actions. These are just a few examples of exploration-exploitation trade-offs in TD learning. The choice of strategy depends on the specific problem, environment, and desired behavior of the learning agent. Researchers continue to explore and develop new techniques to effectively balance exploration and exploitation in reinforcement learning algorithms..		
8	**a**	**Describe various parameters used in Temporal Difference Learning.**	**[L2][CO4]**	**[6M]**
		Temporal Difference Learning is an unsupervised learning technique that is very commonly used in reinforcement learning for the purpose of predicting the total reward expected over the future. Temporal Difference Learning (TD Learning) focuses on predicting a variable's future value in a sequence of states.  Gamma (γ): the discount rate. A value between 0 and 1. The higher the value the less you are discounting. Lambda (λ): the credit assignment variable. A value between 0 and 1. The higher the value the more credit you can assign to further back states and actions. Alpha (α): the learning rate. How much of the error should we accept and therefore adjust our estimates towards. A value between 0 and 1. A higher value adjusts aggressively, accepting more of the error while a smaller one adjusts conservatively but may make more conservative moves towards the actual values. Delta (δ): a change or difference in value.		
	b	**List out the advantages, disadvantages of Temporal difference learning.**	**[L2][CO5]**	**[6M]**
		Advantages: • It can learn in every step online or offline. • It can learn from a sequence which is not complete as well. • It can work in continuous environments. • It has lower variance compared to MC method and is more efficient than MC method. Limitations : • It is a biased estimation. • It is more sensitive to initialization.		

9	**a**	**Explain the Nonparametric rewards and actions in temporal difference learning.**	**[L2][CO5]**	**[6M]**
		In temporal difference (TD) learning, the goal is to learn an optimal policy or value function by estimating the value of states or state-action pairs based on observed rewards and transitions between states. In traditional TD learning, parametric representations such as linear function approximators or neural networks are often used to estimate the values. However, it is also possible to use non-parametric approaches for rewards and actions in TD learning. Non-parametric rewards refer to the use of explicit reward values for individual states or state-action pairs, without assuming a specific functional form or parameterization. Instead of using a function approximator to estimate the value, the rewards are directly observed and used to update the value estimates. This can be useful when the reward structure is complex or difficult to model parametrically. Non-parametric actions, on the other hand, involve explicitly considering the available actions in a state without assuming a specific functional form. In TD learning, this can be done by maintaining a table or lookup structure that stores the values for each state-action pair. The table is updated based on observed rewards and transitions, and the action selection is performed by looking up the values in the table for each action and choosing the one with the highest value. This approach is often referred to as tabular Q-learning. Tabular Q-learning is a popular non-parametric TD learning algorithm. It maintains a table (often called a Q-table) that stores the estimated values for each state-action pair. The Q-table is updated based on the observed rewards and transitions using an update rule such as the Bellman equation. The action selection is performed by choosing the action with the highest value in the Q-table for a given state. Non-parametric approaches like tabular Q-learning have some advantages. They can provide accurate value estimates when the state or action space is relatively small and discrete. They also have strong convergence guarantees and are easy to interpret. However, they suffer from the curse of dimensionality when the state or action space becomes large, as maintaining a table for every state-action pair becomes infeasible. In practice, a trade-off between parametric and non-parametric approaches is often made depending on the characteristics of the problem at hand. Parametric approaches, such as using function approximators, allow for more generalization but may require more data and computational resources. Non-parametric approaches, like tabular Q-learning, are more data-efficient and have better theoretical guarantees but are limited to small and discrete state or action spaces.		
	b	**Assess in detail about partially observables states in Reinforcement learning.**	**[L5][CO5]**	**[6M]**
		A partially observable system is one in which the entire state of the system is not fully visible to an external sensor. In a partially observable system the observer may utilize a memory system in order to add information to the observer's understanding of the system. **A fully observed state** means that there is no hidden information. Clear examples of this are chess and Go because both players have all the information. The fact that both these games are deterministic		

		doesn't matter. A game where the state changes are stochastic can still be fully observable. Games like poker, where both players can observe their own hand but not their opponents' are called partially observable. Other examples of this can be real time strategy games like Starcraft where you can only see in the line of sight of your units. An example of a partially observable system would be a card game in which some of the cards are discarded into a pile face down. In this case the observer is only able to view their own cards and potentially those of the dealer. They are not able to view the face-down (used) cards, nor the cards that will be dealt at some stage in the future. A memory system can be used to remember the previously dealt cards that are now on the used pile. This adds to the total sum of knowledge that the observer can use to make decisions A **partially observable Markov decision process (POMDP)** is a combination of a regular Markov Decision Process to model system dynamics with a hidden Markov model that connects unobservable system states probabilistically to observations. $P=(S,A,T,R,\Omega,O,\gamma)$, Where $S=\{s1,s2,\ldots,sn\}$ is a set of partially observable states, $A=\{a1,a2,\ldots,am\}$ is a set of actions, T a set of conditional transition probabilities $T(s'\|s,a)$ for the state transition $s\rightarrow s'$ conditioned on the taken action. $R:S\times A\rightarrow R$ is the reward function, $\Omega=\{o1,o2,\ldots,ok\}$ is a set of observations, O is a set of observation probabilities $O(o\|s',a)$ conditioned on the reached state and the taken action, and $\gamma\in[0,1]$ is the discount factor.		
10	**a**	**Explain Generalization process in Model Based Learning.**	**[L2][CO5]**	**[6M]**
		In model-based learning, the generalization process refers to the ability of the learned model to make accurate predictions or simulate the behavior of the environment beyond the specific experiences it has encountered during training. Generalization allows the model to make informed decisions in novel situations and generalize its knowledge to unseen states and actions. The generalization process in model-based learning typically involves the following steps: 1. Training the Model: During the training phase, the model-based learning algorithm collects data by interacting with the environment. It observes the states, actions, and resulting rewards and uses this data to learn the dynamics of the environment. The learned model captures the transition probabilities and the expected rewards associated with different state-action pairs. 2. Model Evaluation: Once the model is trained, it needs to be evaluated to assess its predictive accuracy and generalization capabilities. The model can be tested by comparing its predictions against actual observations from the environment. This evaluation helps identify the areas where the model may require further improvement. 3. Generalization Testing: To assess the generalization		

		capabilities of the learned model, it is exposed to novel situations or unseen states and actions that were not encountered during training. The model is used to simulate the environment's dynamics and predict the outcomes of actions in these new situations. 4. Assessing Performance: The performance of the learned model in generalization testing is evaluated by comparing its predictions or simulated outcomes with the actual observed outcomes. Metrics such as prediction accuracy, error rates, or reward accumulation can be used to quantify the model's generalization performance. 5. Iterative Refinement: If the model's generalization performance is not satisfactory, iterative refinement techniques can be applied. These techniques involve updating the model parameters, adjusting the learning algorithm, or collecting additional training data to improve the model's accuracy and generalization capabilities. By going through the generalization process, a model-based learning algorithm aims to develop a learned model that can accurately simulate the environment's dynamics, predict outcomes, and make informed decisions in novel situations. Generalization is crucial for the model to effectively transfer its learned knowledge to real-world scenarios beyond the specific training experiences.		
	b	**Difference between Model based learning and Model free learning**	**[L1][CO4]**	**[6M]**
		Model-based Algorithm Model-based algorithm updates Q-table of the next state S and greedy action A. Based on the highest reward, it chooses the next action. At last, it tries to maximize the rewards of all episodes this way. It is also known as off-policy model as it's primary job is to understand the environment and then create a state-action table. This table is used for getting the prediction of rewards in every state. In off-policy methods, the policy used to generate behavior, called the behavior policy, maybe unrelated to the policy that is evaluated and improved called estimation policy. DQN is an example of a model-based algorithm. Suppose you are learning to swim in swimming pool. You'll learn it by failing and getting experience from your failures. Your swimming model will be trained based on the conditions of the swimming pool. Now, if you are told to swim in flowing water than it will be a challenging task for your model. **Model-free Algorithm** Model-free algorithms update Q-table of the next state S and current policy's action A'. It won't try to understand the whole environment but instead, it follows policy approach. Policy could be some algorithm like the actor-critic. DDPG is an example of a model-free algorithm which is based on the actor-critic approach. **The difference** Here, the main difference is that model-based algorithm tries to get familiar with its environment. The model-free algorithm tries to optimize its policy gradient. If the environment is changed completely then, the model-free algorithm has a higher chance of		

		success than a model-based algorithm. There are several methods that could be used to differentiate between a model-based and a model-free algorithm. Those methods are as follows: 1. If reward is estimated before the action is taken then this is a model-based algorithm. 2. If the accuracy of the model decreased with change in the environment then it could be a model-based algorithm. In the real world, we don't have a fixed environment in every situation. So, most of the use cases could be solved using a model-free algorithm. Self-driving cars, robots, big games like AlphaGo		

Prepared by:**Dr.R.M.Mallika , K.Sirisha, B.Rajakumar**