Subject : Machine Learning
Class : III -I SEC A&B
Department : Artificial Intelligence & Data Science

Note : All the questions are designed for 6M each.

Unit – I

1. What is Machine Learning and list out its applications?

ANS

Machine Learning (ML) is a subfield of artificial intelligence (AI) that focuses on developing algorithms and statistical models that enable computer systems to learn from and make predictions or decisions based on data. The primary goal of machine learning is to enable computers to improve their performance on a specific task through experience and without being explicitly programmed.

Key characteristics of machine learning include:

1. **Learning from Data:** ML algorithms use data to discover patterns, make predictions, or optimize decisions.

2. **Adaptation and Generalization:** ML models can adapt to new data and generalize their learning to make predictions on unseen data.

3. **Automation:** Machine learning automates the process of developing models, making it suitable for a wide range of applications.

4. **Iterative Improvement:** ML models can improve their performance over time as they receive more data and feedback.

Applications of Machine Learning:

Machine learning finds applications in various domains and industries, including:

1. **Natural Language Processing (NLP):** ML is used for sentiment analysis, language translation, chatbots, and speech recognition.

2. **Computer Vision:** ML powers image and video analysis, facial recognition, object detection, and autonomous driving.

3. **Healthcare:** ML assists in disease diagnosis, drug discovery, personalized treatment plans, and patient monitoring.

4. **Finance:** ML is applied to fraud detection, credit scoring, stock market prediction, and algorithmic trading.

5. **Recommendation Systems:** ML algorithms power recommendation engines in e-commerce, streaming platforms, and content delivery.

6. **Autonomous Vehicles:** ML enables self-driving cars to perceive their environment, make decisions, and navigate safely.

7. **Predictive Maintenance:** ML predicts equipment failures and maintenance needs in manufacturing and transportation.

8. **Marketing and Advertising:** ML is used for customer segmentation, ad targeting, and marketing campaign optimization.

9. **Image and Speech Recognition:** ML recognizes patterns in images and audio data for various applications.

10. **Robotics:** ML plays a role in robotic automation, navigation, and object manipulation.

11. **Energy Management:** ML helps optimize energy consumption, grid management, and renewable energy integration.

12. **Security:** ML is used for anomaly detection, network security, and cybersecurity.

13. **Agriculture:** ML assists in crop monitoring, yield prediction, and pest control.

14. **Education:** ML supports personalized learning, student assessment, and automated grading.

15. **Entertainment:** ML is used for content recommendation, game AI, and music generation.

These applications demonstrate the versatility of machine learning across different sectors, making it a powerful tool for solving complex problems and making data-driven decisions.

2. **Define a well-posed learning problem. Discuss any 3 examples for well-posed learning problems along with its features.**

ANS

Well Posed Learning Problem – A computer program is said to learn from experience E in context to some task T and some performance measure P, if its performance on T, as was measured by P, upgrades with experience E.

Any problem can be segregated as well-posed learning problem if it has three traits –

- Task
- Performance Measure
- Experience

1.To better filter emails as spam or not

- Task – Classifying emails as spam or not
- Performance Measure – The fraction of emails accurately classified as spam or not spam
- Experience – Observing you label emails as spam or not spam

2. A checkers learning problem

- Task – Playing checkers game
- Performance Measure – percent of games won against opposer
- Experience – playing implementation games against itself

3.Face Recognition Problem

- Task – predicting different types of faces
- Performance Measure – able to predict maximum types of faces
- Experience – training machine with maximum amount of datasets of different face images

3. **Explain the steps in designing a Learning System in detail.**

ANS

The formal definition of Machine learning as discussed in the previous blogs of the Machine learning series is “A computer program is said to learn from experience E with respect to some class of tasks T

and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E''.

One of the examples discussed was learning checkers game, the parameters T, E, and P with respect to this example are,
T -> Play the checkers game.
P -> Percentage of games won against the opponent.
E -> Playing practice games against itself.

Steps to design a learning system:
To get a successful learning system we need to have a proper design, to make the design proper we'll follow certain steps. In this case, designing a learning system is a five-step process. The steps are,

1. Choosing the Training Experience
2. Choosing the Target Function
3. Choose a Representation for the Target Function
4. Choosing a Function Approximation Algorithm
5. The Final Design

Let's have a look at them briefly,

1. Choosing the Training Experience
The type of training experience chosen has a considerable amount of impact on our algorithm. The training data's characteristics need to be similar to that of the total data set's characteristics.

In order to choose the right training experience for your algorithm, consider these three attributes,

a) Type of Feedback: Check whether the training experience provides direct or indirect feedback to the algorithm based on the choices of the performance system.

In Direct feedback, you get the feedback of your choice immediately. In the case of indirect feedback, you get a sequence of moves and the final outcome of the sequence of action.

b) Degree: The degree of a training experience refers to the extent up to which the learner can control the sequence of training.
For example, the learner might rely on constant feedback about the moves played or it might itself propose a sequence of actions and only ask for help when in need.

c) The representation of the distribution of samples across which performance will be tested is the third crucial attribute.
This basically means the more diverse the set of training experience can be the better the performance can get.

2. Choosing the target function:
The next design decision is to figure out exactly what kind of knowledge will be acquired and how the performance software will put it to use.

Let's take the classic example of the checkers game to understand better. The program only needs to learn how to select the best moves out of the legal moves(Set of all possible moves is called legal moves).

The choice of the target function is a key feature in designing the entire system. The target function V: B -> R. This notation denotes that V maps any legal board state from set B to a real value.

Assigning value to target function in a checkers game,

1. V(b) = 100 if b is the final board state that is won.
2. V(b) = -100 if b is the final board state that is lost.
3. V(b) = 0 if b is the final board state that is drawn.

4. V(b) = V(b') if b is not a final state, and b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game.

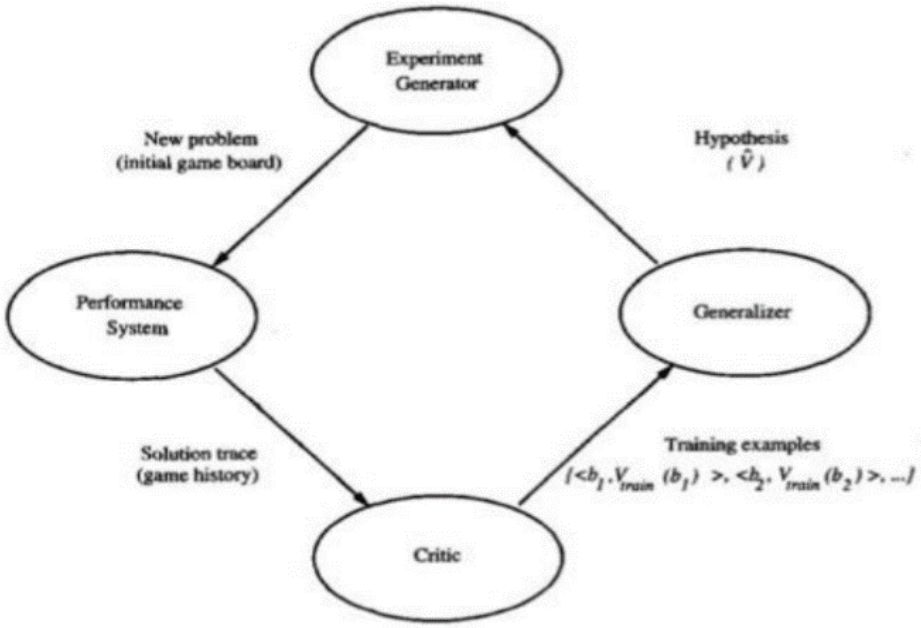
3. Choosing Representation for Target function:
Once done with choosing the target function now we have to choose a representation of this target function, When the machine algorithm has a complete list of all permitted movements, it may pick the best one using any format, such as linear equations, hierarchical graph representation, tabular form, and so on.

Out of these moves, the NextMove function will move the Target move, which will increase the success rate. For example, if a chess machine has four alternative moves, the computer will select the most optimal move that will lead to victory.

4. Choosing a Function Approximation Algorithm:
In this step, we choose a learning algorithm that can approximate the target function chosen. This step further consists of two sub-steps, a. Estimating the training value, and b. Adjusting the weights.

To estimate a training example, we consider the successor move, and in the case of adjusting the weights, one uses certain algorithms like LMS, to find weights of linear functions.

5. The Final Design:



The final design consists of four modules, as described in the picture.

1. The performance system: The performance system solves the given performance task.
2. Critic: The critic takes the history of the game and generates training examples.
3. Generalizer: It outputs the hypothesis that is its estimate of the target function.
4. Experiment Generator: It creates a new problem after taking in the hypothesis for the performance system to explore

4. Differentiate between supervised, unsupervised and reinforcement learning.

ANS

Supervised Learning, Unsupervised Learning, and Reinforcement Learning are three fundamental paradigms in machine learning, each with distinct characteristics and applications. Here's a differentiation between these three types of learning:

1. Supervised Learning:

- **Objective:** In supervised learning, the model learns to map input data to predefined output labels. It involves learning a mapping function from labeled data.

- **Labeled Data:** Supervised learning requires a labeled dataset, where each example consists of input features and corresponding output labels.

- **Learning Process:** The model is trained by minimizing the discrepancy between its predictions and the true labels (ground truth). Common algorithms include linear regression, decision trees, support vector machines, and neural networks.

- **Applications:** Supervised learning is used for tasks like classification (e.g., spam detection, image recognition) and regression (e.g., predicting house prices, stock prices).

2. Unsupervised Learning:
- **Objective:** Unsupervised learning deals with learning patterns, structures, or representations in data without the use of explicit output labels. The goal is to find hidden patterns or group similar data points.

- **Unlabeled Data:** Unsupervised learning uses unlabeled data, where only the input features are available, and there are no target labels.

- **Learning Process:** The model learns to discover inherent structures in the data through techniques like clustering, dimensionality reduction, and density estimation. Common algorithms include K-means clustering, hierarchical clustering, and principal component analysis (PCA).

- **Applications:** Unsupervised learning is used for tasks such as clustering customers for market segmentation, reducing the dimensionality of data for visualization, and anomaly detection.

3. Reinforcement Learning:
- **Objective:** Reinforcement learning is a machine learning training method based on rewarding desired behaviours and punishing undesired ones
Reinforcement learning is concerned with making sequential decisions in an environment to maximize a cumulative reward over time. It focuses on learning a policy that guides an agent's actions to achieve a goal.

- **Feedback Mechanism:** In reinforcement learning, the agent interacts with an environment and receives feedback in the form of rewards or penalties based on its actions.
- **Learning Process:** The agent learns by trial and error, exploring different actions to maximize its expected long-term rewards. Popular algorithms include Q-learning, Deep Q-Networks (DQN), and policy gradient methods.

- **Applications:** Reinforcement learning is used in applications such as game playing (e.g., AlphaGo), robotics, autonomous driving, recommendation systems, and control systems.

Key Differences:
- **Supervised learning** is characterized by the use of labeled data and aims to learn a mapping from inputs to outputs.

- **Unsupervised learning** deals with unlabeled data and focuses on discovering patterns or structures within the data.

- **Reinforcement learning** is concerned with making sequential decisions to maximize cumulative rewards in an interactive environment.

While these paradigms have distinct goals and use cases, they are not mutually exclusive, and hybrid approaches that combine elements of these learning types are also common in machine learning research and applications.

5. Explain simple linear regression..

ANS

1. Simple linear regression

Simple Linear Regression is a type of Regression algorithms that models the relationship between a dependent variable and a single independent variable. The relationship shown by a Simple Linear Regression model is linear or a sloped straight line, hence it is called Simple Linear Regression.

The key point in Simple Linear Regression is that the *dependent variable must be a continuous/real value*.However, the independent variable can be measured on continuous or categorical values.

Simple Linear regression algorithm has mainly two objectives:

- Model the relationship between the two variables. Such as the relationship between Income andexpenditure, experience and Salary, etc.
- Forecasting new observations. Such as Weather forecasting according to temperature, Revenue of acompany according to the investments in a year, etc.

Simple Linear Regression Model:

The Simple Linear Regression model can be represented using the below equation:

y= a0+a1x+ ε

Where,

a0= It is the intercept of the Regression line (can be obtained putting x=0)
a1= It is the slope of the regression line, which tells whether the line is increasing or decreasing.
ε = The error term. (For a good model it will

be negligible

6. What is regression and explain the Multivariate Linear Regression
ANS

Regression is a statistical technique used in machine learning and statistics to model the relationship between a dependent variable (or response) and one or more independent variables (or predictors). It aims to find a mathematical equation that describes how the independent variables influence the dependent variable. The goal is often to make predictions or estimate values of the dependent variable based on the values of the independent variables.

Regression can be broadly categorized into two main types:

1. **Simple Linear Regression:** This is used when there is a single independent variable that influences the dependent variable. The relationship is represented by a straight line equation.

2. **Multivariate Linear Regression:** This is used when there are multiple independent variables that collectively influence the dependent variable. The relationship is represented by a linear equation involving multiple variables.

Multivariate Linear Regression (or simply multilinear regression) is an extension of simple linear regression to handle scenarios where more than one independent variable is involved. The general equation for multivariate linear regression can be expressed as:

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \ldots + \beta_p \cdot X_p + \varepsilon$$

Where:
- Y is the dependent variable (response).
- $X_1, X_2, \ldots, X_p$ are the independent variables (predictors).
- β_0 is the intercept (the value of Y when all X variables are 0).
- $\beta_1, \beta_2, \ldots, \beta_p$ are the coefficients that represent the impact of each independent variable on Y.
- ε represents the error term, which accounts for the variability in Y that cannot be explained by the linear relationship with X variables.

The goal of multivariate linear regression is to estimate the values of $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ that minimize the sum of squared differences between the predicted values ($\hat{Y}$) and the actual values of Y. This is typically done using the method of least squares.

Here are the key steps in multivariate linear regression:

1. **Data Collection:** Gather a dataset that includes the values of the dependent variable and multiple independent variables for each observation.

2. **Model Building:** Build the multivariate linear regression model by estimating the coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ that best fit the data.

3. **Model Evaluation:** Assess the model's performance using evaluation metrics like \(R^2\) (coefficient of determination), MSE (mean squared error), or RMSE (root mean squared error).

4. **Prediction:** Use the trained model to make predictions for new data points by plugging in the values of the independent variables.

5. **Interpretation:** Interpret the estimated coefficients \(\beta_0, \beta_1, \beta_2, \ldots, \beta_p\) to understand the relationship between the independent variables and the dependent variable. Each coefficient represents the change in \(Y\) associated with a one-unit change in the corresponding \(X\) variable while holding other variables constant.

Multivariate linear regression is a versatile technique used in various fields, including economics, finance, social sciences, and machine learning, to analyze and make predictions based on multiple factors or predictors.

7. Formulate the types of regularization and explain how regularization is used to come out of overfitting

ANS

Regularization is a technique used in machine learning and statistical modelling to prevent overfitting and improve the generalization ability of a model. Overfitting occurs when a model fits the training data very closely but performs poorly on unseen data. Regularization methods introduce constraints or penalties on the model parameters during training to discourage overly complex models. There are different types of regularization, including:

1. **L1 Regularization (Lasso):**
   - **Objective:** L1 regularization adds a penalty term to the loss function that encourages the absolute values of the model's coefficients to be small.
   - **Effect:** It tends to drive some of the coefficients to exactly zero, effectively performing feature selection by eliminating less important variables.
   - **Use Cases:** L1 regularization is useful when dealing with high-dimensional data, feature selection, and situations where you suspect that only a subset of features is relevant.

2. **L2 Regularization (Ridge):**
   - **Objective:** L2 regularization adds a penalty term to the loss function that encourages the squared values of the model's coefficients to be small.
   - **Effect:** It prevents the coefficients from becoming too large and helps in reducing the impact of highly correlated features.
   - **Use Cases:** L2 regularization is widely used for general model stability and to address multicollinearity issues in linear regression.

3. **Elastic Net Regularization:**
   - **Objective:** Elastic Net combines both L1 and L2 regularization by adding a weighted sum of the absolute values (L1) and squared values (L2) of the coefficients to the loss function.
   - **Effect:** It provides a balance between feature selection and preventing large coefficients, making it more robust in situations where both L1 and L2 regularization are useful.
   - **Use Cases:** Elastic Net is beneficial when dealing with datasets containing many features and potential multicollinearity.

4. **Dropout (Neural Networks):**
   - **Objective:** In neural networks, dropout is a regularization technique that randomly deactivates (sets to zero) a fraction of neurons during each training iteration.
   - **Effect:** It reduces the reliance of the network on any specific neurons, preventing overfitting and encouraging robustness.
   - **Use Cases:** Dropout is commonly used in deep learning to regularize neural networks and improve their generalization.

How Regularization Addresses Overfitting:
- Regularization methods add penalty terms to the model's loss function, which discourage overly complex models by constraining the parameter values.
- By penalizing large coefficients, regularization discourages the model from fitting noise in the training data.
- The regularization parameter (e.g., \(\lambda\) in L1 and L2 regularization) controls the strength of regularization. Higher values of \(\lambda\) lead to stronger regularization and simpler models.

- Regularization helps the model generalize better to unseen data because it prioritizes a balance between fitting the training data and keeping the model's complexity in check.

In summary, regularization is a powerful tool to combat overfitting in machine learning models. The choice between L1, L2, or Elastic Net regularization depends on the specific problem and the desired trade-off between feature selection and coefficient control. These techniques help improve model performance on unseen data and make models more robust in real-world scenarios.

8. Differentiate between L1 and L2 regularization

ANS

L1 Regularization	L2 Regularization
The penalty term is based on the absolute values of the model's parameters.	The penalty term is based on the squares of the model's parameters.
Produces sparse solutions (some parameters are shrunk towards zero).	Produces non-sparse solutions (all parameters are used by the model).
Sensitive to outliers.	Robust to outliers.
Selects a subset of the most important features.	All features are used by the model.
Optimization is non-convex.	Optimization is convex.
The penalty term is less sensitive to correlated features.	The penalty term is more sensitive to correlated features.
Useful when dealing with high-dimensional data with many correlated features.	Useful when dealing with high-dimensional data with many correlated features and when the goal is to have a less complex model.
Also known as Lasso regularization.	Also known as Ridge regularization.

9. Derive weight update equations for simple linear regression using gradient descent algorithm.

ANS

In simple linear regression, we aim to find the best-fit line that describes the relationship between a dependent variable (Y) and an independent variable (X). The equation of a simple linear regression model is typically represented as:

$$Y = \beta_0 + \beta_1 \cdot X + \varepsilon$$

Where:
- Y is the dependent variable.
- X is the independent variable.
- β_0 is the intercept (y-intercept) of the line.
- β_1 is the slope (coefficient) of the line.
- ε represents the error term, which accounts for the variability in Y that is not explained by the linear relationship with X.

To find the best values of β_0 and β_1 that minimize the sum of squared errors (SSE), we can use the gradient descent algorithm. The goal of gradient descent is to iteratively update the coefficients until convergence to the optimal values.

The cost function (also called the loss function) for simple linear regression is often defined as the Mean Squared Error (MSE), which is given by:

$$J(\beta_0, \beta_1) = \frac{1}{2N} \sum_{i=1}^{N} (Y_i - (\beta_0 + \beta_1 \cdot X_i))^2$$

Where:
- N is the number of data points.
- Y_i is the observed value of the dependent variable for the ith data point.
- X_i is the value of the independent variable for the ith data point.
- β_0 and β_1 are the coefficients we want to optimize.

Now, let's derive the weight update equations for β_0 and β_1 using gradient descent:

1. **Initialize \(\beta_0\) and \(\beta_1\) with some values (usually 0 or random).**
2. **Define the learning rate (\(\alpha\)), which controls the step size in the gradient descent updates.**
3. **Iteratively update \(\beta_0\) and \(\beta_1\) until convergence:**

   - **Update \(\beta_0\) using the gradient of the cost function with respect to \(\beta_0\):**

     \[\beta_0 = \beta_0 - \alpha \frac{\partial J(\beta_0, \beta_1)}{\partial \beta_0}\]

     \[\beta_0 = \beta_0 + \alpha \frac{1}{N} \sum_{i=1}^{N} (Y_i - (\beta_0 + \beta_1 \cdot X_i))\]

   - **Update \(\beta_1\) using the gradient of the cost function with respect to \(\beta_1\):**

     \[\beta_1 = \beta_1 - \alpha \frac{\partial J(\beta_0, \beta_1)}{\partial \beta_1}\]

     \[\beta_1 = \beta_1 + \alpha \

10. Write the Gradient Descent algorithm and differentiate between different variants of Gradient Descent algorithm

ANS

Gradient Descent is an iterative optimization algorithm that tries to find the optimum value (Minimum/Maximum) of an objective function. It is one of the most used optimization techniques in machine learning projects for updating the parameters of a model in order to minimize a cost function.

The main aim of gradient descent is to find the best parameters of a model which gives the highest accuracy on training as well as testing datasets. In gradient descent, The gradient is a vector that points in the direction of the steepest increase of the function at a specific point. Moving in the opposite direction of the gradient allows the algorithm to gradually descend towards lower values of the function, and eventually reaching to the minimum of the function.

Steps Required in Gradient Descent Algorithm

- **Step 1** we first initialize the parameters of the model randomly
- **Step 2** Compute the gradient of the cost function with respect to each parameter. It involves making partial differentiation of cost function with respect to the parameters.
- **Step 3** Update the parameters of the model by taking steps in the opposite direction of the model. Here we choose a hyperparameter learning rate which is denoted by alpha. It helps in deciding the step size of the gradient.
- **Step 4** Repeat steps 2 and 3 iteratively to get the best parameter for the defined model
  gradient_descent – In the gradient descent function we will make the prediction on a dataset and compute the difference between the predicted and actual target value and accordingly we will update the parameter and hence it will return the updated parameter.

In the Machine Learning Regression problem, our model targets to get the best-fit regression line to predict the value y based on the given input value (x). While training the model, the model calculates the cost function like Root Mean Squared error between the predicted value (pred) and true value (y). Our model targets to minimize this cost function.
To minimize this cost function, the model needs to have the best value of θ_1 and θ_2(for Univariate linear regression problem). Initially model selects θ_1 and θ_2 values randomly and then iteratively update these value in order to minimize the cost function until it reaches the minimum. By the time model achieves the minimum cost function, it will have the best θ_1 and θ_2 values. Using these updated values of θ_1 and θ_2 in the hypothesis equation of linear equation, our model will predict the output value y.

How do θ_1 and θ_2 values get updated?

Linear Regression Cost

Function:

so our model aim is to Minimize $\frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) – y^{(i)})^2$ and store the parameters which makes it minimum.

Gradient Descent Algorithm For Linear Regression

Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y_i]^2$$

$h_\Theta(x_i)$: Predicted Value; y_i: True Value

Gradient Descent

$$\Theta_j = \Theta_j - \alpha\frac{\partial}{\partial\Theta_j}J(\Theta_0, \Theta_1)$$

α: Learning Rate

Now,

$$\frac{\partial}{\partial\Theta}J_\Theta = \frac{\partial}{\partial\Theta}\frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y]^2$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x_i) - y)\frac{\partial}{\partial\Theta_j}(\Theta x_i - y)$$

$$= \frac{1}{m}(h_\Theta(x_i) - y)x_i$$

Therefore,

$$\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]$$

Gradient descent algorithm for linear regression

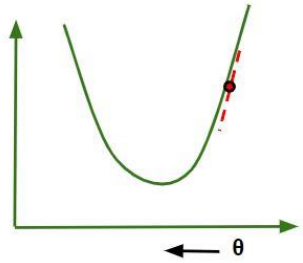
-> θ_j : Weights of the hypothesis.
-> $h_{\theta(xi)}$: predicted y value for i^{th} input.
-> **i** : Feature index number (can be 0, 1, 2, ..., n).
-> **α** : Learning Rate of Gradient Descent.

How Does Gradient Descent Work

Gradient descent works by moving downward toward the pits or valleys in the graph to find the minimum value. This is achieved by taking the

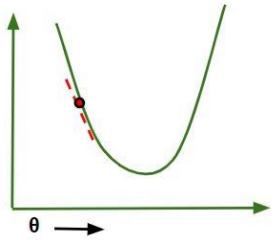
derivative of the cost function, as illustrated in the figure below. During each iteration, gradient descent step-downs the cost function in the direction of the steepest descent. By adjusting the parameters in this direction, it seeks to reach the minimum of the cost function and find the best-fit values for the parameters. The size of each step is determined by parameter **α** known as **Learning Rate**.
In the Gradient Descent algorithm, one can infer two points :

- **If slope is +ve** : $\theta_j = \theta_j -$ (+ve value). Hence the value of θ_j decreases.



If slope is +ve in Gradient Descent

- **If slope is -ve** : $\theta_j = \theta_j -$ (-ve value). Hence the value of θ_j increases.

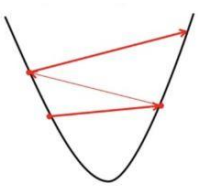


***If slope is -ve** in Gradient Descent*

How To Choose Learning Rate

The choice of correct learning rate is very important as it ensures that Gradient Descent converges in a reasonable time. :

- If we choose **α to be very large**, Gradient Descent can overshoot the minimum. It may fail to converge or even diverge.



Effect of large alpha value on Gradient Descent

- If we choose α to be very small, Gradient Descent will take small steps to reach local minima and will take a longer time to reach minima.

Effect of small alpha value on Gradient Descent

There are several variants of gradient descent that differ in the way the step size or learning rate is chosen and the way the updates are made. Here are some popular variants:

1. **Batch Gradient Descent:** In batch gradient descent, To update the model parameter values like weight and bias, the entire training dataset is used to compute the gradient and update the parameters at each iteration. This can be slow for large datasets but may lead to a more accurate model. It is effective for convex or relatively smooth error manifolds because it moves directly toward an optimal solution by taking a large step in the direction of the negative gradient of the cost function. However, it can be slow for large datasets because it computes the gradient and updates the parameters using the entire training dataset at each iteration. This can result in longer training times and higher computational costs.
2. **Stochastic Gradient Descent (SGD):** In SGD, only one training example is used to compute the gradient and update the parameters at each iteration. This can be faster than batch gradient descent but may lead to more noise in the updates.
3. **Mini-batch Gradient Descent:** In Mini-batch gradient descent a small batch of training examples is used to compute the gradient and update the parameters at each iteration. This can be a good compromise between batch gradient descent and Stochastic Gradient Descent, as it can be faster than batch gradient descent and less noisy than Stochastic Gradient Descent.
4. **Momentum-based Gradient Descent:** In momentum-based gradient descent, Momentum is a variant of gradient descent that incorporates information from the previous weight updates to help

the algorithm converge more quickly to the optimal solution. Momentum adds a term to the weight update that is proportional to the running average of the past gradients, allowing the algorithm to move more quickly in the direction of the optimal solution. The updates to the parameters are based on the current gradient and the previous updates. This can help prevent the optimization process from getting stuck in local minima and reach the global minimum faster.

5. **Nesterov Accelerated Gradient (NAG):** NAG is an extension of Momentum Gradient Descent. It evaluates the gradient at a hypothetical position ahead of the current position based on the current momentum vector, instead of evaluating the gradient at the current position. This can result in faster convergence and better performance.
6. **Adagrad:** In this variant, the learning rate is adaptively adjusted for each parameter based on the historical gradient information. This allows for larger updates for infrequent parameters and smaller updates for frequent parameters.
7. **RMSprop:** In this variant, the learning rate is adaptively adjusted for each parameter based on the moving average of the squared gradient. This helps the algorithm to converge faster in the presence of noisy gradients.
8. **Adam:** Adam stands for adaptive moment estimation, it combines the benefits of Momentum-based Gradient Descent, Adagrad, and RMSprop the learning rate is adaptively adjusted for each parameter based on the moving average of the gradient and the squared gradient, which allows for faster convergence and better performance on non-convex optimization problems. It keeps track of two exponentially decaying averages the first-moment estimate, which is the exponentially decaying average of past gradients, and the second-moment estimate, which is the exponentially decaying average of past squared gradients. The first-moment estimate is used to calculate the momentum, and the second-moment estimate is used to scale the learning rate for each parameter. This is one of the most popular optimization algorithms for deep learning.

11. Explain the steps in designing a Learning System in detail.

3RD QUESTION IS REPEATED

12. Summarize the evaluation measures for regression techniques.

ANS

Evaluation measures for regression techniques assess the performance of predictive models that aim to predict continuous numerical values. The following are key evaluation measures used in regression:

1. **Mean Absolute Error (MAE):**
   - MAE is the average of the absolute differences between predicted and actual values.
   - It measures the average magnitude of errors, with larger errors contributing proportionally more.
   - MAE is less sensitive to outliers compared to MSE.
   - Formula: $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
2. **Mean Squared Error (MSE):**
   - MSE is the average of the squared differences between predicted and actual values.
   - It penalizes larger errors more heavily than MAE, making it sensitive to outliers.
   - Formula: $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
3. **Root Mean Squared Error (RMSE):**
   - RMSE is the square root of MSE and provides a measure of the average magnitude of errors in the same units as the target variable.
   - It is more interpretable than MSE since it's in the same units as the data.
   - Formula: $\text{RMSE} = \sqrt{\text{MSE}}$
4. **R-squared (R2) Score:**
   - R2 measures the proportion of the variance in the dependent variable that is explained by the model.
   - It ranges from 0 to 1, with higher values indicating a better fit.

- It ranges from 0 to 1, with higher values indicating a better fit.
- R2 = 1 means a perfect fit, while R2 = 0 indicates that the model does no better than predicting the mean of the target variable.
- Formula: $R^2 = 1 - \frac{\text{SSE}}{\text{SST}}$, where SSE is the sum of squared errors and SST is the total sum of squares.

5. **Adjusted R-squared:**
   - Adjusted R2 adjusts the R2 score to account for the number of predictors in the model.
   - It penalizes the inclusion of irrelevant predictors and tends to increase when relevant predictors are added.
   - Formula: $\text{Adjusted } R^2 = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$, where n is the number of observations and p is the number of predictors.
6. **Mean Absolute Percentage Error (MAPE):**
   - MAPE expresses errors as a percentage of the actual values.
   - It is useful when you want to understand errors in terms of their relative size.
   - Formula: $\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{|y_i|} \times 100$

7. **Mean Bias Deviation (MBD):**
   - MBD measures the average bias of the model, indicating whether it tends to overestimate or underestimate the actual values.
   - Positive MBD indicates overestimation, and negative MBD indicates underestimation.
   - Formula: $\text{MBD} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)$
8. **Mean Percentage Error (MPE):**
   - MPE measures the average error as a percentage of the actual values.
   - It provides insights into the overall bias of the model.
   - Formula: $\text{MPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i} \times 100$

These evaluation measures help assess the accuracy, precision, and goodness of fit of regression models. The choice of which measure to use depends on the specific problem, the nature of the data, and the goals of the analysis.

13. Discuss about k-fold cross validation

ANS

K-Fold Cross-Validation

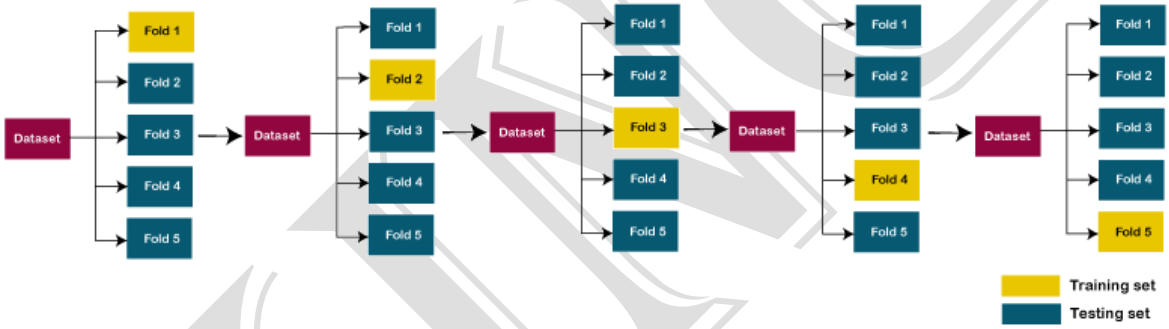
K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called **folds**. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the output is less biased than other methods.

The steps for k-fold cross-validation are:

- Split the input dataset into K groups
- For each group:
    - Take one group as the reserve or test data set.
    - Use remaining groups as the training dataset
    - Fit the model on the training set and evaluate the performance of the model using the test set.

Let's take an example of 5-folds cross-validation. So, the dataset is grouped into 5 folds. On 1st iteration, the first fold is reserved for test the model, and rest are used to train the model. On 2nd iteration, the second fold is used to test the model, and rest are used to train the model. This process will continue until each fold is not used for the test fold.

Consider the below diagram:



Stratified k-fold cross-validation

This technique is similar to k-fold cross-validation with some little changes. This approach works on stratification concept, it is a process of rearranging the data to ensure that each fold or group is a good representative of the complete dataset. To deal with the bias and variance, it is one of the best approaches.

It can be understood with an example of housing prices, such that the price of some houses can be much high than other houses. To tackle such situations, a stratified k-fold cross-validation technique is useful.

Unit – 2

1. **Write about Logistic function along with its most convenient mathematical properties**.

ANS

The logistic function, also known as the sigmoid function, is a widely used mathematical function in various fields, including machine learning, statistics, and biology. It is an essential component in logistic regression, artificial neural networks, and other models. The logistic function has several convenient mathematical properties that make it suitable for various applications. Here's an overview of the logistic function and some of its key mathematical properties:

Definition of the Logistic Function:

The logistic function is defined as follows:

\[f(x) = \frac{1}{1 + e^{-x}}\]

Here, "e" is the base of the natural logarithm (approximately equal to 2.71828), and "x" is the input variable. The logistic function takes any real number "x" as input and maps it to the range (0, 1), which is particularly useful for modeling probabilities.

Mathematical Properties of the Logistic Function:

1. *S-Shaped Curve:* The logistic function exhibits an S-shaped curve, which makes it suitable for modeling phenomena that have a natural saturation point. As "x" approaches positive infinity, \(f(x)\) approaches 1, and as "x" approaches negative infinity, \(f(x)\) approaches 0. This property is valuable for binary classification problems where we want to model probabilities of two classes.

2. *Sigmoidal Behavior:* The term "sigmoid" refers to the S-shaped curve of the logistic function. The sigmoidal behavior allows the logistic function to smoothly transition between 0 and 1, making it differentiable everywhere. This differentiability is crucial for optimization algorithms like gradient descent used in training machine learning models.

3. *Monotonicity:* The logistic function is monotonically increasing, meaning that as "x" increases, \(f(x)\) also increases. This property is essential for capturing the relationship between inputs and outputs in a continuous and consistent manner.

4. *Symmetry:* The logistic function is symmetric with respect to the y-axis (i.e., it is an even function). This symmetry simplifies mathematical calculations and is often used to ensure that logistic regression models have balanced effects for different input values.

5. *Derivative:* The derivative of the logistic function has a straightforward form:

\[\frac{d}{dx} f(x) = f(x) \cdot (1 - f(x))\]

This derivative is used in gradient-based optimization algorithms, such as gradient descent, when training machine learning models. It helps in finding the optimal model parameters efficiently.

6. *Log-Odds Transformation:* The logistic function is used to transform log-odds (logarithm of the odds) into probabilities. The log-odds is a convenient representation for modeling binary outcomes, and the logistic function maps it to probabilities between 0 and 1.

7. *Easy Interpretation:* The logistic function's output can be interpreted as the probability of an event occurring. In logistic regression, for example, the logistic function is used to model the probability of a binary outcome based on predictor variables.

8. *Flexibility:* The logistic function can be modified and extended to handle multi-class classification problems, leading to the softmax function in the context of neural networks.

In summary, the logistic function is a versatile mathematical function with valuable properties for modeling and predicting probabilities. Its S-shaped curve, differentiability, and interpretability make it a fundamental tool in various fields, particularly in binary and multi-class classification tasks.

2. **Explain the one-vs-rest approach of multi-class classification in Logistic Regression**

ANS

The one-vs-rest (OvR) approach, also known as one-vs-all (OvA) or binary relevance, is a common strategy for performing multi-class classification using binary classifiers like Logistic Regression. In the OvR approach, you transform a multi-class classification problem into multiple binary classification subproblems, where each subproblem addresses the classification of one class against the rest of the classes. Here's how the OvR approach works in the context of Logistic Regression:

Step 1: Data Preparation

1. *Dataset:* You start with a dataset that has multiple classes (more than two). Each data point in the dataset belongs to one of these classes.

2. *Label Transformation:* You transform the class labels into binary labels for each subproblem. For each class, you create a binary label where data points belonging to that class are labeled as positive (1), and data points belonging to all other classes are labeled as negative (0).

Step 2: Model Training

1. *Multiple Classifiers:* You train a separate binary classifier (e.g., Logistic Regression) for each class. Each binary classifier is trained to distinguish between its designated class (positive) and all other classes (negative).

2. *Binary Classification:* During training, each binary classifier sees a modified dataset where only one class is treated as positive, and the others are treated as negative. For example, in the classifier for Class A, Class A data points are labeled as positive, and data points from all other classes are labeled as negative.

3. *Classifier Training:* Each binary classifier is trained independently, and the model parameters (weights and biases) are learned using the logistic regression algorithm or any other binary classification algorithm.

Step 3: Prediction

1. *Prediction for Each Classifier:* To make a multi-class prediction for a new data point, you run it through all the binary classifiers. Each classifier will output a probability score indicating the likelihood of the data point belonging to its designated class (positive class).

2. *Selecting the Class:* You then select the class corresponding to the binary classifier that produces the highest probability score. In other words, you choose the class for which the binary classifier is most confident.

Advantages of the One-vs-Rest Approach:

1. *Simplicity:* The OvR approach is straightforward and easy to implement. It leverages binary classification models, which are well-understood and widely available.

2. *Interpretability:* Each binary classifier provides interpretable results, making it clear which class is being distinguished from the others.

3. *Scalability:* The OvR approach can handle a large number of classes without significant computational complexity because each binary classifier is trained independently.

Limitations of the One-vs-Rest Approach:

1. *Imbalanced Data:* If the dataset has imbalanced class distributions, where some classes have significantly fewer examples than others, this approach can lead to biased classifiers.

2. *Dependency Between Classifiers:* The OvR approach assumes that the classifiers are independent, which may not be the case in practice. Dependencies between classes can affect classification results.

3. *Lack of Margin Information:* OvR classifiers do not provide margin information or probabilities for each class, which may be useful in certain applications.

In summary, the one-vs-rest approach in Logistic Regression is a simple and effective strategy for handling multi-class classification problems by

breaking them down into multiple binary classification subproblems. It is widely used and suitable for many scenarios, but it's important to consider its limitations, especially in cases of imbalanced data and class dependencies.

3. Explain K-Nearest Neighbor learning algorithm
ANS

The abbreviation KNN stands for “K-Nearest Neighbour”. It is a supervised machine learning algorithm. Thealgorithm can be used to solve both classification and regression problem statements.

The number of nearest neighbours to a new unknown variable that has to be predicted or classified isdenoted by the symbol ‘K’.

KNN calculates the distance from all points in the proximity of the unknown data and filters out the ones
with the shortest distances to it. As a result, it’s often referred to as a distance-based algorithm.

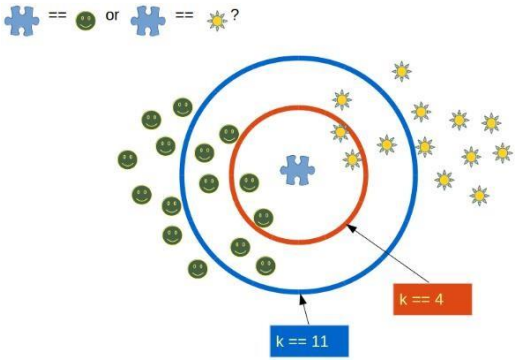
In order to correctly classify the results, we must first determine the value of K (Number of NearestNeighbours).

It is recommended to always select an odd value of K ~

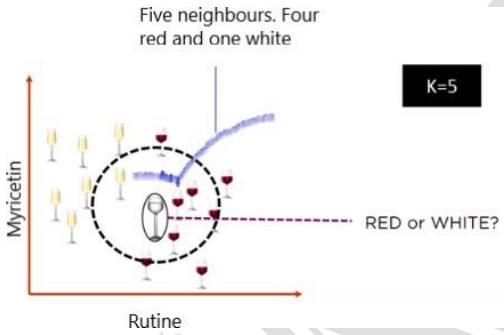
When the value of K is set to even, a situation may arise in which the elements from both groups are equal.In the diagram below, elements from both groups are equal in the internal “Red” circle (k == 4).

In this condition, the model would be unable to do the correct classification for you. Here the model willrandomly assign any of the two classes to this new unknown data.

Choosing an odd value for K is preferred because such a state of equality between the two classes would never occur here. Due to the fact that one of the two groups would still be in the majority, the value of K isselected as odd.

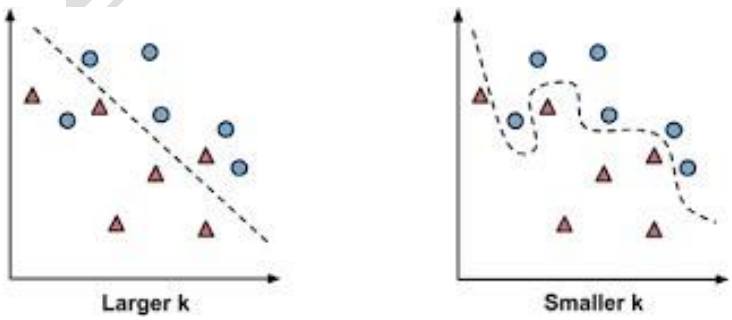


Src: https://images.app.goo.gl/Q8ZKxQ8mhP68yxqn7



The impact of selecting a smaller or larger K value on the model

- Larger K value: The case of underfitting occurs when the value of k is increased. In this case, themodel would be unable to correctly learn on the training data.
- Smaller k value: The condition of overfitting occurs when the value of k is smaller. The model will capture all of the training data, including noise. The model will perform poorly for the test data in this scenario.



Src:

https://images.app.goo.gl/vXStNS4NeEqUCDXn8

How does KNN work for ‘Classification’ and ‘Regression’ problem statements?

- ☞ Classification

When the problem statement is of ‘classification’ type, KNN tends to use the concept of “Majority Voting”.
Within the given range of K values, the class with the most votes is chosen.

Consider the following diagram, in which a circle is drawn within the radius of the five closest neighbours.Four of the five neighbours in this neighbourhood voted for ‘RED,’ while one voted for ‘WHITE.’ It will be classified as a ‘RED’ wine based on the majority votes.

Src:
https://images.app.goo.gl/Ud42nZn8Q8FpDVcs5

Real-world example:

Several parties compete in an election in a democratic country like India. Parties compete for voter supportduring election campaigns. The public votes for the candidate with whom they feel more connected.

When the votes for all of the candidates have been recorded, the candidate with the most votes is declared
as the election’s winner.

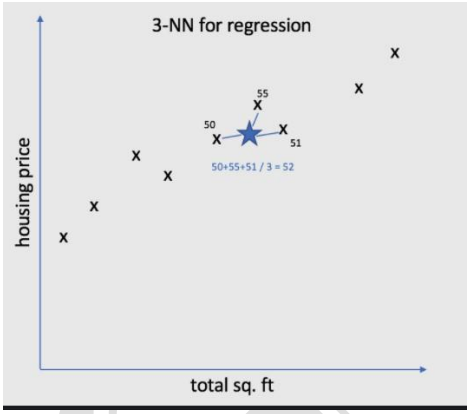
- ☞ Regression

KNN employs a mean/average method for predicting the value of new data. Based on the value of K, itwould consider all of the nearest neighbours.

The algorithm attempts to calculate the mean for all the nearest neighbours’ values until it has identified all

the nearest neighbours within a certain range of the K value.

Consider the diagram below, where the value of k is set to 3. It will now



calculate the mean (52) based onthe values of these neighbours (50, 55, and 51) and allocate this value to the unknown data.

Src: https://images.app.goo.gl/pzW97weL6vHJByni8

4. Write a short note on the following

a. Distance measures used in K-Nearest Neighbors algorithm

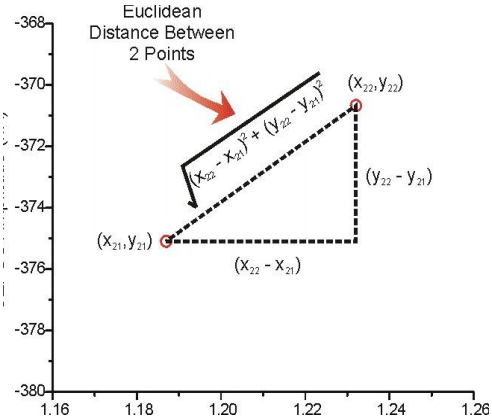
ANS

1. Distance based Methods

Distanced based algorithms are machine learning algorithms that classify queries by computing distances between these queries and a number of internally stored exemplars. Exemplars that are closets to the queryhave the largest influence on the classification assigned to the query.

Euclidean Distance

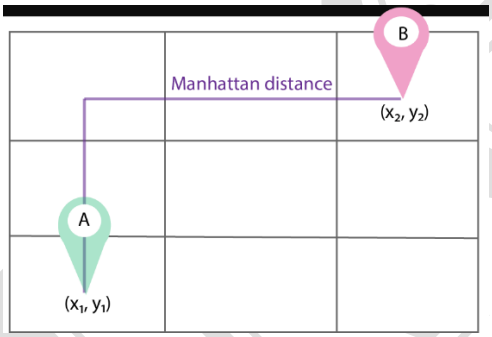
It is the most common use of distance. In most cases when people said about distance, they will refer to Euclidean distance. Euclidean distance is also known as simply distance. When data is dense or continuous, this is the



best proximity measure. The Euclidean distance between two points is the length of the path connectingthem. The Pythagorean theory gives distance between two points.

Manhattan Distance

- If you want to find Manhattan distance between two different points (x1, y1) and (x2, y2) such as thefollowing, it would look like the following:
- Manhattan distance = (x2 – x1) + (y2 – y1)
- Diagrammatically, it would look like traversing the path from point A to point B while walking onthe pink straight line.



Minkowski Distance

The generalized form of the Euclidean and Manhattan Distances is the Minkowski Distance. You can expressthe Minkowski distance as

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

The order of the norm is represented by p.

When an order(p) is 1, Manhattan Distance is represented, and when order(p) is 2 in the above formula,Euclidean Distance is represented.

b. Normalization of data.

ANS

Normalization of data is a crucial preprocessing step in various data analysis and machine learning tasks. It involves scaling the features of a dataset to a common range or distribution, typically to ensure that different features are on a level playing field and to prevent one feature from dominating the others. Normalization helps improve the stability and performance of algorithms that are sensitive to the scale of input features. Here are some key points about data normalization:

Why Normalize Data:

1. *Scale Independence:* Many machine learning algorithms are sensitive to the scale of input features. Features with larger magnitudes can dominate those with smaller magnitudes during calculations. Normalization helps ensure that all features contribute equally to the learning process.

2. *Convergence:* Normalizing data can help optimization algorithms, like gradient descent, converge faster and more reliably because it avoids oscillations and overshooting that can occur when features are on different scales.

3. *Distance-Based Algorithms:* Algorithms that rely on distance calculations, such as K-Nearest Neighbors and clustering algorithms, are highly affected by feature scales. Normalization is essential to make these algorithms work effectively.

Common Techniques for Data Normalization:

1. *Min-Max Scaling (Normalization):* This method scales features to a specified range, usually between 0 and 1.
   - Formula: \[X_normalized = \frac{X - X_min}{X_max - X_min}\]
   - Pros: Maintains the relationships between data points, useful for algorithms that require data to be in a bounded range.
   - Cons: Sensitive to outliers.

2. *Z-Score Standardization (Standardization):* This method transforms data to have a mean of 0 and a standard deviation of 1.
   - Formula: \[X_standardized = \frac{X - \mu}{\sigma}\]
   - Pros: Robust to outliers, suitable for many algorithms, maintains shape of the distribution.
   - Cons: Doesn't bound data to a specific range.

3. *Robust Scaling:* Robust scaling is similar to min-max scaling but uses the interquartile range (IQR) instead of the range (max-min). It is less affected by outliers.
   - Formula: \[X_robust = \frac{X - Q1}{Q3 - Q1}\]
   - Pros: Robust to outliers, useful when the data distribution is skewed.
   - Cons: May not be suitable for all scenarios.

4. *Log Transformation:* In cases where data is highly skewed or follows a power-law distribution, taking the logarithm of the values can normalize them and make the distribution more symmetric.

When to Normalize Data:

- Data normalization is typically performed during the data preprocessing phase, after data cleaning and before model training. It is especially important when using distance-based algorithms, neural networks, and support vector machines.

- However, not all algorithms require data normalization. Decision trees, for example, are scale-invariant, so normalization may not provide any significant benefits in such cases.

In conclusion, data normalization is a critical step in data preprocessing that ensures fair treatment of features in various machine learning and data analysis tasks. The choice of normalization method should be based on the characteristics of the data and the requirements of the specific algorithm being used.

5. Discuss the Naive Bayes Classifier.

ANS

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes,

Which can be described as:

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

principle of Bayes' Theorem.Bayes' Theorem:

- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probabilityof a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

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

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis istrue.

P(A) is Prior Probability: Probability of hypothesis

before observing the evidence.P(B) is Marginal

Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of weather conditions and corresponding target variable "Play". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.

2. Generate Likelihood table by finding the probabilities of given features.

Now, use Bayes theorem to calculate the posterior probability.

6. **Consider the training data in the following table where Play is a class attribute. In the table, the Humidity attribute has values "L" (for low) or "H" (for high), Sunny has values "Y" (for yes) or "N" (for no), Wind has values "S" (for strong) or "W" (for weak), and Play has values "Yes" or "No".**

Humidity	Sunny	Wind	Play
L	N	S	No
H	N	W	Yes
H	Y	S	Yes
H	N	W	Yes
L	Y	S	No

a. **What is class label for the following day (Humidity=L, Sunny=N, Wind=W), according to naïve Bayesian classification?**

ANS

To determine the class label for the given day (Humidity=L, Sunny=N, Wind=W) according to the Naive Bayesian classification, we will use the provided training data. We'll calculate the probabilities for both "Play=Yes" and "Play=No" and choose the class label with the higher probability.

Given training data:

1. Humidity: L, Sunny: N, Wind: S, Play: No
2. Humidity: H, Sunny: N, Wind: W, Play: Yes
3. Humidity: H, Sunny: Y, Wind: S, Play: Yes
4. Humidity: H, Sunny: N, Wind: W, Play: Yes
5. Humidity: L, Sunny: Y, Wind: S, Play: No

We'll calculate the probabilities:

1. Calculate the prior probabilities (P(Play)):
   - P(Play=Yes) = (Number of Yes) / (Total Number) = 3 / 5
   - P(Play=No) = (Number of No) / (Total Number) = 2 / 5

2. Calculate the conditional probabilities for each attribute given Play:

a. P(Humidity=L | Play):
- P(Humidity=L | Play=No) = (Number of L when Play=No) / (Total Number of Play=No) = 1 / 2
- P(Humidity=L | Play=Yes) = (Number of L when Play=Yes) / (Total Number of Play=Yes) = 1 / 3

b. P(Sunny=N | Play):
- P(Sunny=N | Play=No) = (Number of N when Play=No) / (Total Number of Play=No) = 1 / 2
- P(Sunny=N | Play=Yes) = (Number of N when Play=Yes) / (Total Number of Play=Yes) = 1 / 3

c. P(Wind=S | Play):
- P(Wind=S | Play=No) = (Number of S when Play=No) / (Total Number of Play=No) = 1 / 2
- P(Wind=S | Play=Yes) = (Number of S when Play=Yes) / (Total Number of Play=Yes) = 2 / 3

3. Calculate the marginal probabilities for each attribute:

- P(Humidity=L) = (Number of L in Humidity) / (Total Number) = 2 / 5
- P(Sunny=N) = (Number of N in Sunny) / (Total Number) = 2 / 5
- P(Wind=W) = (Number of W in Wind) / (Total Number) = 2 / 5

Now, let's calculate the probabilities for both classes:

For Play=No:
$$P(Play=No \mid Humidity=L, Sunny=N, Wind=W) \propto P(Humidity=L \mid Play=No) \cdot P(Sunny=N \mid Play=No) \cdot P(Wind=W \mid Play=No) \cdot P(Play=No)$$
$$= \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{2}{5} = \frac{1}{20}$$

For Play=Yes:
$$P(Play=Yes \mid Humidity=L, Sunny=N, Wind=W) \propto P(Humidity=L \mid Play=Yes) \cdot P(Sunny=N \mid Play=Yes) \cdot P(Wind=W \mid Play=Yes) \cdot P(Play=Yes)$$

$$= \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{3}{5} = \frac{2}{45}$$

Since $\frac{2}{45} > \frac{1}{20}$, the class label for the given day (Humidity=L, Sunny=N, Wind=W) according to Naive Bayesian classification is "Play=Yes."

7. Suppose 10000 patients get tested for flu; out of them, 9000 are actually healthy and 1000 are actually sick. For the sick people, a test was positive for 620 and negative for 380. For the healthy people, the same test was positive for 180 and negative for 8820. Construct a confusion matrix for the data and compute accuracy, precision, recall and F1 measure for the data
   To compute the confusion matrix, accuracy, precision, recall, and F1 measure, we can use the following information:
   ANS
   - True Positives (TP): The number of sick people correctly identified as sick.
   - True Negatives (TN): The number of healthy people correctly identified as healthy.
   - False Positives (FP): The number of healthy people incorrectly identified as sick.
   - False Negatives (FN): The number of sick people incorrectly identified as healthy.

   From the given information:

   - TP = 620 (positive for the sick)
   - TN = 8820 (negative for the healthy)
   - FP = 180 (positive for the healthy)
   - FN = 380 (negative for the sick)

   Now, let's construct the confusion matrix:

   ```
                    Actual Sick (Sick)   Actual Healthy (Healthy)
   Predicted Sick      620                 180
   Predicted Healthy   380                 8820
   ```

Now, let's calculate the metrics:

1. Accuracy:
   Accuracy measures the overall correctness of the test.

   Accuracy = (TP + TN) / (TP + TN + FP + FN)
   Accuracy = (620 + 8820) / (620 + 8820 + 180 + 380)
   Accuracy = 9440 / 10000
   Accuracy = 0.944

2. Precision:
   Precision measures the accuracy of positive predictions.

   Precision = TP / (TP + FP)
   Precision = 620 / (620 + 180)
   Precision = 620 / 800
   Precision = 0.775

3. Recall (Sensitivity):
   Recall measures the ability to correctly identify sick individuals.

   Recall = TP / (TP + FN)
   Recall = 620 / (620 + 380)
   Recall = 620 / 1000
   Recall = 0.62

4. F1 Measure:
   The F1 measure is the harmonic mean of precision and recall. It provides a balance between precision and recall.

   F1 = 2 * (Precision * Recall) / (Precision + Recall)
   F1 = 2 * (0.775 * 0.62) / (0.775 + 0.62)
   F1 = 2 * 0.4795 / 1.395
   F1 = 0.687

So, the metrics for the given data are:

- Accuracy: 0.944
- Precision: 0.775

- Recall: 0.62
- F1 Measure: 0.687

8. Explain the basic decision tree learning algorithm.

ANS

Decision Tree Induction Decision tree induction is a technique used for identifying unknown class labels in classification. Working of a decision tree The tree has three types of nodes.

i) A root node has no incoming edges and zero or more outgoing edges.

ii) Internal nodes, each of which has exactly one incoming edge and two or more outgoing edges.

iii) Leaf or terminal nodes, each of which has exactly one incoming edge and no outgoing edges.

Algorithm 4.1 A skeleton decision tree induction algorithm.

```
TreeGrowth (E, F)
 1: if stopping_cond(E,F) = true then
 2:    leaf = createNode().
 3:    leaf.label = Classify(E).
 4:    return leaf.
 5: else
 6:    root = createNode().
 7:    root.test_cond = find_best_split(E, F).
 8:    let V = {v|v is a possible outcome of root.test_cond }.
 9:    for each v ∈ V do
10:       E_v = {e | root.test_cond(e) = v and e ∈ E}.
11:       child = TreeGrowth(E_v, F).
12:       add child as descendent of root and label the edge (root → child) as v.
13:    end for
14: end if
15: return root.
```

i) The createNode() function extends the decision tree by creating a new node. A node in the decision tree has either a test condition, denoted as node.test_cond, or a class label, denoted as node.label.

ii) The find.best_split () function determines which attribute should be selected as the test condition for splitting the training records.

iii) The classify() function determines the class label to be assigned to a leaf node.

iv) The stopping_cond() function is used to terminate the tree-growing process by testing whether all the records are classified or not.

9. What do you mean by gain and entropy? How it is used to build the decision tree?

ANS

In the context of building decision trees, "gain" and "entropy" are two common concepts used to determine the best attribute (feature) to split the data at each node of the tree. These concepts are associated with information theory and help assess the uncertainty or impurity of a dataset.

Entropy:

- Entropy is a measure of impurity or randomness in a dataset. In the context of decision trees, it quantifies how mixed the class labels are within a node. A node with low entropy indicates that most of the samples belong to a single class, making it a pure node. A node with high entropy suggests that the samples are evenly distributed among different classes, making it an impure node.

- Mathematically, the entropy of a node S is calculated as:

$$Entropy(S) = -p_1 * \log2(p_1) - p_2 * \log2(p_2) - ... - p_c * \log2(p_c)$$

Where:
- p_i is the proportion of samples in class i within the node.
- c is the number of unique classes.

Information Gain:

- Information gain is a measure of how much information a particular attribute (feature) contributes to reducing the entropy or impurity of a node. In other words, it quantifies the reduction in uncertainty achieved by splitting the data based on a specific attribute.

- The information gain for a node S when splitting on attribute A is calculated as:

$$Information\ Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} * Entropy(S_v)$$

Where:
- $Values(A)$ represents the possible values of attribute A.

- \(S_v\) is the subset of samples in node \(S\) for which attribute \(A\) takes value \(v\).

Using Gain and Entropy to Build Decision Trees:
- The decision tree learning algorithm uses gain and entropy to determine the best attribute to split the data at each node. It calculates the information gain for each attribute and selects the attribute that maximizes the information gain.

- The attribute with the highest information gain is chosen as the splitting criterion, as it leads to the greatest reduction in uncertainty or impurity. The dataset is then split based on the values of this attribute, creating child nodes in the decision tree.

- This process is repeated recursively for each child node until a stopping condition is met, such as reaching a predefined maximum depth or achieving pure nodes (entropy equals zero).

In summary, gain and entropy are used in decision tree algorithms to evaluate the potential of different attributes for splitting the data. They help select attributes that provide the most information for making accurate predictions and result in a more effective decision tree.

10. For the following set of training samples, find which attribute can be chosen as the root for decision tree classification.

Instance	Classification	a1	a2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

ANS

To determine which attribute can be chosen as the root for decision tree classification, we need to calculate the information gain for each attribute. Information gain measures how much an attribute helps reduce the uncertainty (or impurity) in the classification.

Let's calculate the information gain for attributes a1 and a2.

Entropy before splitting (Entropy(S)):
- The dataset contains 3 instances with a positive classification (+) and 3 instances with a negative classification (-), making it a balanced dataset.
- So, Entropy(S) = -0.5 * log2(0.5) - 0.5 * log2(0.5) = 1 (since log2(0.5) = -1)

Now, let's calculate the information gain for each attribute:

Information Gain for a1:
- When we split on attribute a1:
  - For a1=T (true), there are 2 positive instances and 1 negative instance.
  - For a1=F (false), there are 1 positive instance and 2 negative instances.

- Information Gain(a1) = Entropy(S) - [(3/6) * Entropy(S|a1=T) + (3/6) * Entropy(S|a1=F)]

  - Entropy(S|a1=T) = -2/3 * log2(2/3) - 1/3 * log2(1/3) = 0.9183
  - Entropy(S|a1=F) = -1/3 * log2(1/3) - 2/3 * log2(2/3) = 0.9183

- Information Gain(a1) = 1 - [(3/6) * 0.9183 + (3/6) * 0.9183] = 1 - 0.9183 = 0.0817

Information Gain for a2:
- When we split on attribute a2:
  - For a2=T (true), there are 2 positive instances and 2 negative instances.
  - For a2=F (false), there are 1 positive instance and 1 negative instance.

- Information Gain(a2) = Entropy(S) - [(4/6) * Entropy(S|a2=T) + (2/6) * Entropy(S|a2=F)]

  - Entropy(S|a2=T) = -2/4 * log2(2/4) - 2/4 * log2(2/4) = 1
  - Entropy(S|a2=F) = -1/2 * log2(1/2) - 1/2 * log2(1/2) = 1

- Information Gain(a2) = 1 - [(4/6) * 1 + (2/6) * 1] = 1 - 0.6667 = 0.3333

Conclusion:
- The information gain for attribute a2 (0.3333) is greater than the information gain for attribute a1 (0.0817).

- Therefore, attribute a2 should be chosen as the root for the decision tree classification because it provides more information gain and better splits the dataset into classes.

11. Summarize the evaluation measures used for classification.

12. Identify the first splitting attribute for Decision Tree using Hunt's algorithm with the following dataset.

Major	Experience	Tie	Hired?
CS	programming	pretty	NO
CS	programming	pretty	NO
CS	management	pretty	YES
CS	management	ugly	YES
business	programming	pretty	YES
business	programming	ugly	YES
business	management	pretty	NO
business	management	pretty	NO

ANS

Hunt's algorithm, also known as the ID3 (Iterative Dichotomiser 3) algorithm, is used to build decision trees by selecting the best attribute to split the dataset based on information gain or entropy. To determine the first splitting attribute, we calculate the information gain or entropy for each attribute.

Let's calculate the information gain for each attribute: Major, Experience, and Tic.

Entropy before splitting (Entropy(S)):
- The dataset contains 4 instances labeled "YES" (Hired) and 4 instances labeled "NO" (Not Hired), making it a balanced dataset.
- So, Entropy(S) = -0.5 * log2(0.5) - 0.5 * log2(0.5) = 1 (since log2(0.5) = -1)

Now, let's calculate the information gain for each attribute:

Information Gain for Major:
- When we split on attribute Major:
  - For Major = cs, there are 2 "NO" and 2 "YES."
  - For Major = business, there are 2 "NO" and 2 "YES."

- Information Gain(Major) = Entropy(S) - [(4/8) * Entropy(S|Major=cs) + (4/8) * Entropy(S|Major=business)]

  - Entropy(S|Major=cs) = -2/4 * log2(2/4) - 2/4 * log2(2/4) = 1
  - Entropy(S|Major=business) = -2/4 * log2(2/4) - 2/4 * log2(2/4) = 1

- Information Gain(Major) = 1 - [(4/8) * 1 + (4/8) * 1] = 1 - 0.5 = 0.5

Information Gain for Experience:
- When we split on attribute Experience:
  - For Experience = programming, there are 2 "NO" and 3 "YES."
  - For Experience = management, there are 2 "NO" and 1 "YES."

- Information Gain(Experience) = Entropy(S) - [(5/8) * Entropy(S|Experience=programming) + (3/8) * Entropy(S|Experience=management)]

  - Entropy(S|Experience=programming) = -2/5 * log2(2/5) - 3/5 * log2(3/5) ≈ 0.971
  - Entropy(S|Experience=management) = -2/3 * log2(2/3) - 1/3 * log2(1/3) ≈ 0.918

- Information Gain(Experience) = 1 - [(5/8) * 0.971 + (3/8) * 0.918] ≈ 0.262

Information Gain for Tic:
- When we split on attribute Tic:
  - For Tic = prcny, there are 1 "NO" and 1 "YES."
  - For Tic = pretty, there are 2 "NO" and 1 "YES."
  - For Tic = ugly, there are 2 "NO" and 2 "YES."

- Information Gain(Tic) = Entropy(S) - [(2/8) * Entropy(S|Tic=prcny) + (3/8) * Entropy(S|Tic=pretty) + (3/8) * Entropy(S|Tic=ugly)]

  - Entropy(S|Tic=prcny) = -1/2 * log2(1/2) - 1/2 * log2(1/2) = 1
  - Entropy(S|Tic=pretty) = -2/3 * log2(2/3) - 1/3 * log2(1/3) ≈ 0.918
  - Entropy(S|Tic=ugly) = -2/5 * log2(2/5) - 3/5 * log2(3/5) ≈ 0.971

- Information Gain(Tic) = 1 - [(2/8) * 1 + (3/8) * 0.918 + (3/8) * 0.971] ≈ 0.256

Conclusion:
- The attribute with the highest information gain is "Major" (0.5).
- Therefore, "Major" should be chosen as the first splitting attribute for the decision tree classification.

Unit – 3

1. **Define unsupervised learning. Discuss the applications of unsupervised learning.**

ANS

Unsupervised learning is a machine learning paradigm where the algorithm is trained on a dataset without explicit supervision or labeled output. In other words, the model is not provided with the correct answers during training, and it must discover patterns and structures in the data on its own. The goal of unsupervised learning is to find inherent structures, relationships, and insights within the data.

Unsupervised learning can be categorized into two main types:

1. **Clustering:** Clustering algorithms aim to group similar data points together based on their similarities or distance metrics. The primary objective is to discover natural groupings or clusters within the data. Common clustering algorithms include K-Means, Hierarchical Clustering, and DBSCAN.

2. **Dimensionality Reduction:** Dimensionality reduction techniques are used to reduce the complexity of data by transforming it into a lower-dimensional representation while preserving essential information. This is particularly useful for visualizing high-dimensional data and reducing noise. Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are popular dimensionality reduction techniques.

Applications of Unsupervised Learning:

1. **Customer Segmentation:** Unsupervised learning is frequently used in marketing to segment customers based on their purchasing behavior, preferences, and demographics. This helps businesses tailor their marketing strategies to specific customer groups.

2. **Anomaly Detection:** Unsupervised learning can be employed to detect unusual or anomalous data points in various domains, including cybersecurity (detecting intrusions), fraud detection (identifying fraudulent transactions), and industrial equipment monitoring (identifying faulty components).

3. **Image and Video Compression:** Dimensionality reduction techniques like PCA are used in image and video compression algorithms to reduce the size of multimedia files while preserving essential visual information.

4. **Recommendation Systems:** Clustering and dimensionality reduction are essential components of recommendation systems. These systems analyze user behavior and preferences to recommend products, movies, or content tailored to individual users.

5. **Natural Language Processing (NLP):** Unsupervised learning is applied in NLP for tasks such as topic modeling, where it identifies latent topics within large text corpora without predefined categories.

6. **Biology and Genomics:** Unsupervised learning techniques are used in biological research to analyze gene expression data, classify genes, and discover hidden patterns in biological datasets.

7. **Speech and Audio Analysis:** Unsupervised learning can help classify and segment audio data based on acoustic features, making it useful in speech recognition, music genre classification, and audio source separation.

8. **Autonomous Vehicles:** Unsupervised learning is employed for environment perception tasks in autonomous vehicles, such as clustering objects in the surroundings and identifying obstacles.

9. **Healthcare:** Unsupervised learning can help identify patient clusters based on medical data, assisting in personalized medicine and disease diagnosis.

10. **Market Basket Analysis:** This technique is widely used in retail to uncover associations between products that are frequently purchased together, leading to improved inventory management and marketing strategies.

In summary, unsupervised learning plays a crucial role in discovering hidden patterns, organizing data, and making data-driven decisions across various domains, making it a valuable tool in machine learning and data analysis.

2. What is clustering? Explain K-Means clustering algorithm.

ANS

Unsupervised classification is known as cluster analysis. Clustering Grouping of similar data items together is known as clustering. This is mainly used for summarization of data.

The process of grouping data basing on the information found is known as cluster analysis. The main goal of cluster analysis is that the objects within a group be similar to one another and different from the objects in other groups.

Basics of K-means or Basic K-means Algorithm:

--K-Means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem.
-- The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters).
-- The main idea is to define k centroids, one for each cluster.
--These centroids should be placed in a intelligent way because of different location causes different result.
-- So, the better choice is to place them as much as possible far away from each other.
--The next step is to take each point belonging to a given data set and associate it to the nearest centroid.
--When, no point is pending, the first step is completed and an early group is done.
--At this point it is necessary to re-calculate k new centroids as bar centers of the clusters resulting from the previous step.
-- After obtaining these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid.
--A loop has been generated. As a result of this loop, one may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.
--Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

--Where, $||x_i^{(j)}-c_j||^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster center c_j, is an indicator of the distance of the n data points from their respective cluster centers.

--In the K-means initial centroids are often chosen randomly.
--The centroid is (typically) the mean of the points in the cluster.
--The 'Closeness' between the data points or the objects can be measured by using Euclidean distance, cosine similarity, correlation, etc.
--We can specify the stopping condition as the number of clusters that we want.
--The Complexity of K-means is O(n * K * I * d)

- n = number of points, K = number of clusters,
  I = number of iterations, d = number of attributes

3. Illustrate the K-means clustering algorithm and construct the K-means partitioning algorithm using the below dataset. Consider five points {A, B, C, D, E} with the following coordinates as a two-dimensional sample for clustering. A=(1,1); B-(1,0); C=(0,2); D=(2,4); E=(3,5)

ANS

To illustrate the K-Means clustering algorithm using the given dataset with five points {A, B, C, D, E}, we'll partition them into two clusters (K=2). Here's a step-by-step walkthrough of the algorithm:

Step 1: Initialize K centroids
- Start by randomly selecting K initial centroids. Let's choose A(1, 1) and D(2, 4) as the initial centroids for Cluster 1 and Cluster 2, respectively.

Step 2: Assign each data point to the nearest centroid
- Calculate the distance between each point and the two centroids.
- Assign each point to the cluster whose centroid is closest.

Distance calculations:
- Distance from A(1, 1) to points: A(0), B(1), C(1.41), D(3.16), E(4.24)
- Distance from D(2, 4) to points: A(3.16), B(4.47), C(2.83), D(0), E(1.41)

Cluster assignments:
Cluster 1: {A(1, 1), C(0, 2)} (Centroid: (0.5, 1.5))

Cluster 2: {B(1, 0), D(2, 4), E(3, 5)} (Centroid: (2, 3))

Step 3: Update the centroids
- Recalculate the centroids for each cluster by taking the mean of the data points in that cluster.

New Centroid for Cluster 1: (0.5, 1.5)
New Centroid for Cluster 2: (2, 3)

Step 4: Repeat steps 2 and 3 until convergence
- Continue iterating through steps 2 and 3 until the centroids no longer change significantly.

After a few iterations, you'll reach convergence, and the final cluster assignments and centroids will be stable:

Final Cluster Assignments:
Cluster 1: {A(1, 1), C(0, 2)} (Centroid: (0.5, 1.5))
Cluster 2: {B(1, 0), D(2, 4), E(3, 5)} (Centroid: (2, 3))

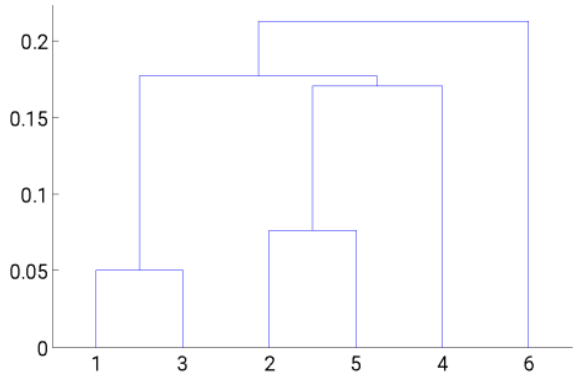
The K-Means algorithm has partitioned the dataset into two clusters based on the given coordinates. Each point is assigned to the cluster whose centroid is closest.

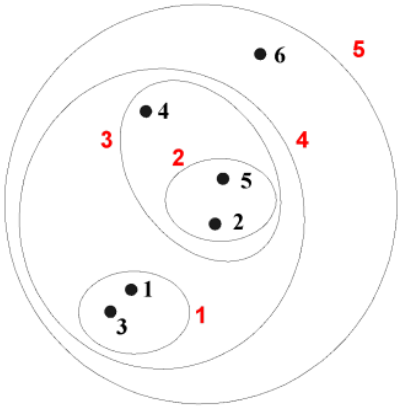
4. **Differentiate between agglomerative and divisive clustering. How the hierarchical clustering is represented?**

→Agglomerative hierarchical clustering-Basics:

Hierarchical Clustering (HC):

-- Hierarchical Clustering is a typical clustering analysis approach via partitioning data set sequentially
--It constructs nested partitions layer by layer via grouping objects into a tree of clusters (without the need to know the number of clusters in advance)
--This clustering technique uses distance matrix as clustering criteria
--HC (Hierarchical Clustering) produces a set of nested clusters organized as a hierarchical tree
--HC can be visualized as a dendrogram.
--Dendogram is a tree like diagram that records the sequences of merges or splits





--There are two main types of hierarchical clustering

Agglomerative Hierarchical Clustering (HAC):

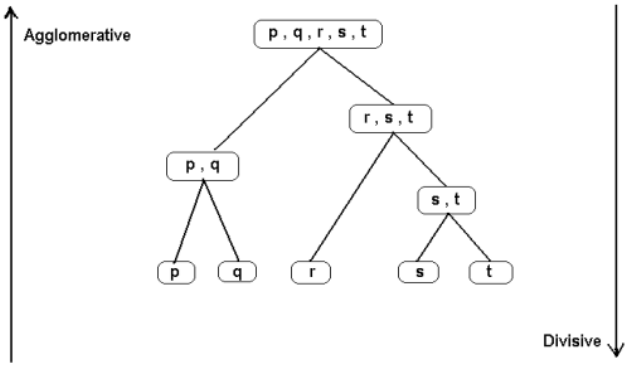
- In HAC Data objects are grouped in a bottom-up fashion.
- Initially each data object is in its own cluster.
- Then HAC merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied.

- For HAC termination condition can be specified by the user, as the desired number of clusters

Divisive Hierarchical Clustering (DHC):

- In this data objects are grouped in a top down manner
- Initially all objects are in one cluster
- Then the cluster is subdivided into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions as the desired number of clusters is obtained.

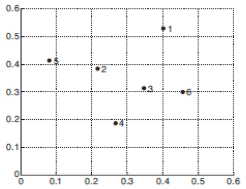
Agglomerative vs Divisive



5. Cluster the following 6 points using Single Link Hierarchical clustering.

Point	x	y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30

ANS



Set of 6 two-dimensional points.

Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

xy coordinates of 6 points.

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Euclidean distance matrix for 6 points.

Min or single link technique

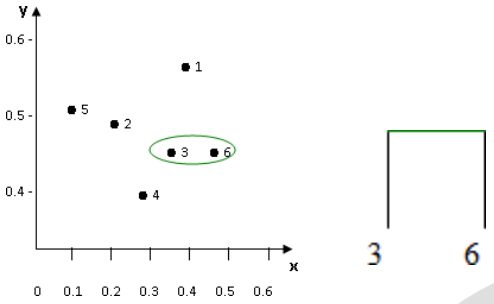
The proximity of two clusters is defined as the minimum of the distance between any two points I the two different clusters. The single link technique is good at handling non-elliptical shapes, butis sensitive to noise and outliers.

	P1	P2	P3	P4	P5	P6
P1	0	0.24	0.22	0.37	0.34	0.23
P2	0.24	0	0.15	0.20	0.14	0.25
P3	0.22	0.15	0	0.15	0.28	0.11
P4	0.37	0.20	0.15	0	0.29	0.22
P5	0.34	0.14	0.28	0.29	0	0.39
P6	0.23	0.25	0.11	0.22	0.39	0

In the above table, p3, p6 is having the lowest distance, so merge the data points into a single cluster.

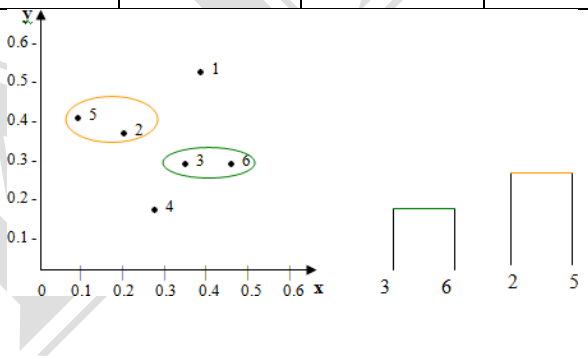
	P1	P2	P3P6	P4	P5
P1	0	0.24	0.22	0.37	0.34
P2	0.24	0	0.15	0.20	0.14

P3P6	0.22	0.15	0	0.15	0.28
P4	0.37	0.20	0.15	0	0.29
P5	0.34	0.14	0.28	0.29	0



The next lowest distance is for P2P5, so merge those two data points into a single cluster. Thematrix obtains as follows:

	P1	P2P5	P3P6	P4
P1	0	0.24	0.22	0.37
P2P5	0.24	0	0.15	0.20
P3P6	0.22	0.15	0	0.15
P4	0.37	0.20	0.15	0



The distance between (p3, p6) and (p2, p5) would be calculated as follows:

dist((p3, p6), (p2, p5)) =MIN (*dist*(p3, p2) , *dist*(p6, p2), *dist*(p3, p5),

dist(p6, p5))

=MIN (0.15, 0.25, 0.28, 0.39)

=0.15

dist((p3, p6), (p1)) =MIN (*dist*(p3, p1) , *dist*(p6, p1)

=MIN (0.22, 0.23)

=0.22

dist((p3, p6), (p4)) =MIN (*dist*(p3, p4) , *dist*(p6, p4)

=MIN (0.15, 0.22)

=0.15

dist((p2, p5), (p1)) =MIN (*dist*(p2, p1) , *dist*(p5, p1)

=MIN (0.24, 0.34)

=0.24

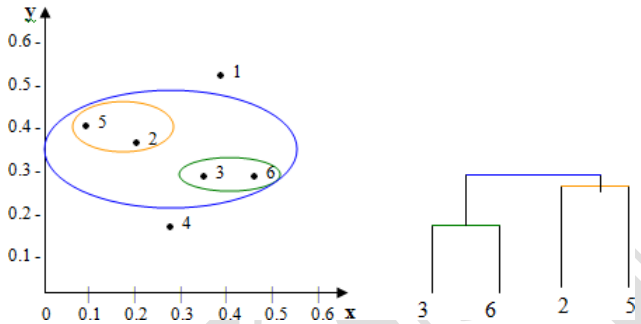
dist((p2, p5), (p4)) =MIN (*dist*(p2, p4) , *dist*(p5, p4)

=MIN (0.20, 0.29)

=0.20

So, looking at the last distance matrix above, we see that (p2, p5) and (p3, p6) have the smallestdistance from all - 0.15. We also notice that p4 and (p3, p6) have the same distance - 0.15. In that case, we can pick either one. We choose (p2, p5) and (p3, p6). So, we merge those two in asingle cluster, and re-compute the distance matrix.

	P1	P2P5P3P6	P4
P1	0	0.22	0.37
P2P5P3P6	0.22	0	0.15
P4	0.37	0.20	0



The distance between (P2, P5, P3, P6) and P1 would be calculated as follows:

dist((P2, P5, P3, P6), (p1)) =MIN (*dist*(p2, p1) , *dist*(p5, p1), *dist*(p3, p1) , *dist*(p6, p1))

=MIN (0.24, 0.34, 0.22, 0.23)

=0.22

The distance between (P2, P5, P3, P6) and P4 would be calculated as follows:

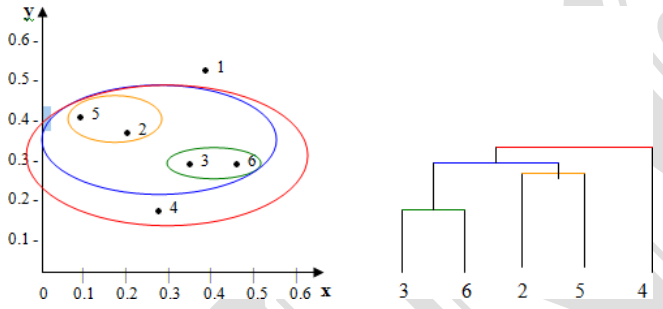
dist((P2, P5, P3, P6), (p4)) =MIN (*dist*(p2, p4) , *dist*(p5, p4), *dist*(p3, p4) , *dist*(p6, p4))

=MIN (0.20, 0.29, 0.15, 0.22)

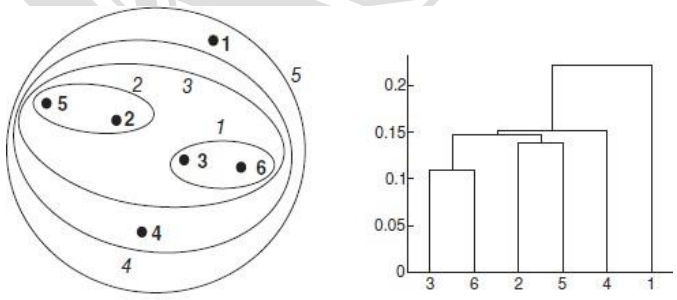
=0.15

So, looking at the last distance matrix above, we see that (p2, p5, p3, p6) and p4 have the smallestdistance from all - 0.15. So, we merge those two in a single cluster, and re-compute the distance matrix.

	P1	P2P5P3P6P4
P1	0	0.22
P2P5P3P6P4	0.22	0



Finally merge clusters (P2, P5, P3, P6, P4) and P1. The clusters and dendrogram are formed as follows:



(a) Single link clustering. (b) Single link dendrogram.