ML SESSIONAL-2

UNIT III

1. Differentiate Linear SVM Classification and Nonlinear SVM Classification.

s.no	Linear SVM	Non-linear SVM
1.	Linear SVM is used for linearly separable data.	Non-Linear SVM is used for non-linearly separated data.
2.	if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data	if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data
3.	classifier is used called as Linear SVM classifier	classifier used is called as Non-linear SVM classifier

Linear SVM	Non-Linear SVM
It can be easily separated with a linear line.	It cannot be easily separated with a linear line.
Data is classified with the help of hyperplane.	We use Kernels to make non-separable data into separable data.
Data can be easily classified by drawing a straight line.	We map data into high dimensional space to classify.

2.Why do we need the random forest algorithm? Explain its advantages and disadvantages.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning**

Why use Random Forest?

It takes less training time as compared to other algorithms.

It predicts output with high accuracy, even for the large dataset it runs efficiently.

It can also maintain accuracy when a large proportion of data is missing.

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

3.Describe Naïve Bayes Classifiers in detail.

Naïve Bayes Classifier Algorithm

Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.

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.

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:

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

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 weatherconditions. 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
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player

should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the weather conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Applying Bayes' theorem:

P(Yes|Sunny)=

P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

Learn more

So P(Yes|Sunny) = 0.3*0.71/0.35= 0.60

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = 0.41

So as we can see from the above calculation that P(Yes|Sunny)>P(No|Sunny)

Hence on a Sunny day, Player can play the game.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It is the most popular choice for text

classification problems.

- Disadvantages of Naïve Bayes Classifier:
- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationshipbetween features.

Applications of Naïve Bayes Classifier:

- It is used in medical data classification.
- It can be used in real-time
- It is used in Text classification

UNIT IV

1.Describe how dimensionality is reduced using PCA

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning.

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.

2.Explain about K-means algorithm with an example

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning

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.

Let's illustrate the steps of the K-means algorithm with a simple example:Suppose we have the following dataset in a two-dimensional space:

Initialization: We randomly choose three points from the dataset as the initial centroids. Let's say we select [2, 10], [5, 8], and [1, 2] as the initial centroids.

Assignment Step: We calculate the Euclidean distance between each point and the centroids. For example, the distance between [2, 10] and [2, 5] is 5, while the distance between [5, 8] and [2, 5] is 3.605. Based on the distances, we assign each point to the cluster with the closest centroid.

Assignments:

Cluster 1: [2, 5], [1, 2]

Cluster 2: [8, 4], [7, 5], [6, 4]

Cluster 3: [2, 10], [5, 8], [4, 9]

Update Step: After assigning the points, we calculate the mean of each cluster's points and update the centroids accordingly.

Updated centroids:

Cluster 1: [1.5, 3.5]

Cluster 2: [7, 4.33]

Cluster 3: [3.67, 9]

Repeat Assignment and Update: We repeat the assignment and update steps iteratively until convergence. In each iteration, the points are reassigned based on the updated centroids, and the centroids are recalculated.

Iteration 2:

Assignments:

Cluster 1: [2, 5], [1, 2]

Cluster 2: [8, 4], [7, 5], [6, 4]

Cluster 3: [2, 10], [5, 8], [4, 9]

Updated centroids:

Cluster 1: [1.5, 3.5]

Cluster 2: [7, 4.33]

Cluster 3: [3.67, 9]

Iteration 3:

Assignments:

Cluster 1: [1, 2]

Cluster 2: [8, 4], [7, 5], [6, 4]

Cluster 3: [2, 10], [5, 8], [4, 9]

Updated centroids:

Cluster 1: [1, 2]

Cluster 2: [7, 4.33]

Cluster 3: [3.67, 9]

After several iterations, the assignments and centroids no longer change significantly, indicating convergence. The algorithm has clustered the points into three distinct groups based on their proximity to the centroids.

3.Explain DBSCAN algorithm with an example

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular density-based clustering algorithm used to group data points based on their density. It is particularly effective at discovering clusters of arbitrary shapes and handling outliers. Here's an explanation of the DBSCAN algorithm with an example:

Suppose we have a dataset of 100 points in a two-dimensional space. We want to cluster these points using the DBSCAN algorithm.

The DBSCAN algorithm operates based on two parameters: eps (epsilon) and min_samples. eps defines the radius within which neighboring points are considered part of the same cluster, and min_samples specifies the minimum number of points required to form a dense region (core points).

Let's illustrate the steps of the DBSCAN algorithm with a simple example:

Suppose we have the following dataset in a two-dimensional space:.

[2, 10], [2, 5], [8, 4], [5, 8], [7, 5], [6, 4], [1, 2], [4, 9]

We want to cluster these points using DBSCAN with eps = 2.5 and min_samples = 3.

Initialization: We randomly select a point from the dataset that has not been visited and has at least min_samples neighboring points within distance eps. We mark this point as visited and start a new cluster.

Expand Cluster: We expand the cluster by adding all directly-reachable points from the selected point within distance eps. If a point has at least min_samples neighboring points within distance eps, it becomes a core point, and we recursively expand the cluster from that point.

For example, let's start with point [2, 10]. Its neighbors within eps = 2.5 are [2, 5] and [1, 2]. Both points have at least min_samples = 3 neighbors within eps, so they become core points. We expand the cluster from these core points by adding their directly-reachable points.

Expanded cluster:

[2, 10]: [2, 5], [1, 2]

[2, 5]: [8, 4], [7, 5], [6, 4]

[1, 2]: None (no more directly-reachable points)

Form New Cluster or Noise: If there are no more core points or directly-reachable points within eps, we start a new cluster with an unvisited point that meets the criteria. If all points have been visited, the algorithm terminates.

In our example, we have core points [8, 4], [7, 5], and [6, 4] remaining. We start a new cluster with [8, 4] and expand it to include its directly-reachable points.

New cluster:

[8, 4]: [7, 5], [6, 4]

Repeat Expand Cluster: We repeat the expand cluster step until all points have been visited.

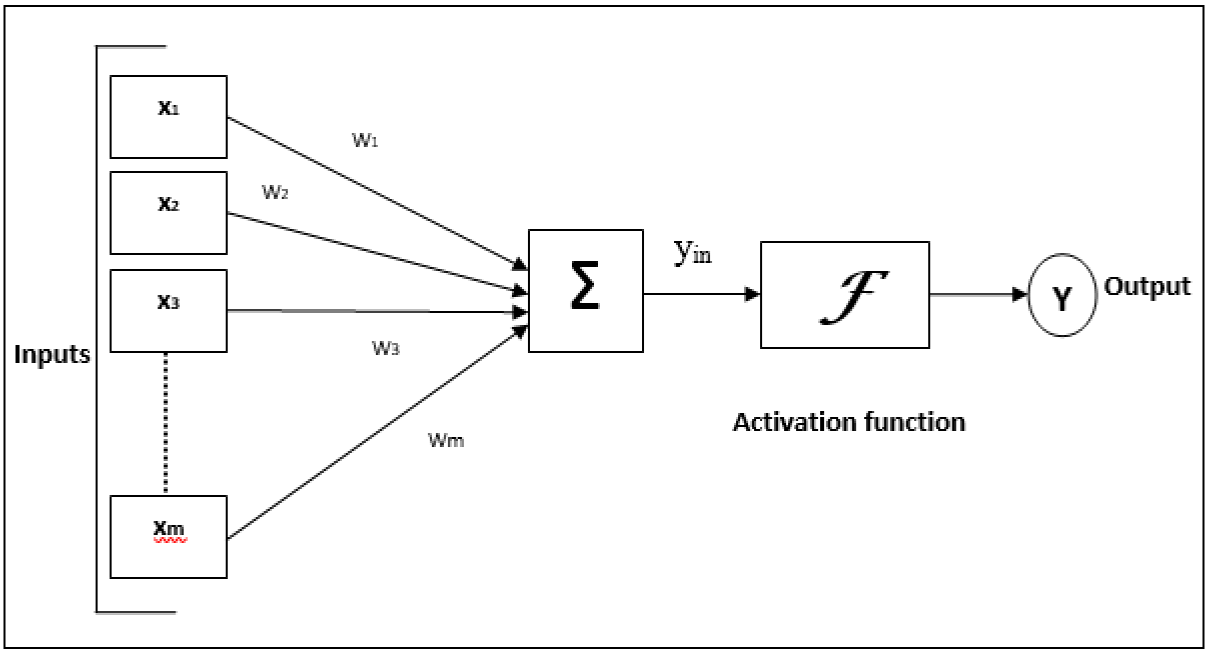
In our example, there are no more unvisited points that meet the criteria, so the algorithm terminates. We have two clusters: [2, 10], [2, 5], [1, 2] in one cluster and [8, 4], [7, 5], [6, 4] in the other cluster. The remaining points [4, 9] do not have enough neighboring points within eps and are considered noise or outliers.

UNIT V

1.Describe in detail about neural networks role in machine learning.

Neural networks play a fundamental role in machine learning, particularly in the field of deep learning. They are a type of mathematical model inspired by the structure and function of biological neural networks in the human brain.

Model of Artificial Neural Network



Processing of ANN depends upon the following three building blocks –

- Network Topology
- Adjustments of Weights or Learning
- Activation Functions

The power of neural networks lies in their ability to learn complex patterns and relationships in data. By organizing neurons into multiple layers and using non-linear activation functions, neural networks can capture and model intricate patterns that might be difficult for other machine learning algorithms to discern. Deep neural networks, which consist of many hidden layers, have demonstrated remarkable performance in various tasks, including image and speech recognition, natural language processing, and autonomous driving.

Neural networks have revolutionized machine learning by enabling the development of highly sophisticated models capable of tackling complex problems. Their ability to learn from large amounts of data, generalize patterns, and make accurate predictions has made them indispensable in many domains.

2.Explain the Implementation Process of MLPs with Keras

the implementation process of Multi-Layer Perceptrons (MLPs) with Keras involves several steps.

Keras is a high-level neural networks API written in Python that provides a user-friendly interface for building and training neural networks. MLPs are a type of feedforward neural network, where information flows in one direction, from the input layer through the hidden layers to the output layer

Install the Required Libraries: First, make sure you have Keras and its backend engine, such as TensorFlow or Theano, installed. You can install Keras using pip or conda.

Import the Required Libraries: Import the necessary libraries in your Python script or notebook. Typically, you'll need to import keras and numpy.

Prepare the Data: MLPs require input data in a numerical format. Preprocess your data accordingly, including tasks like normalization, one-hot encoding, and splitting the data into training and testing sets.

Define the Model: In Keras, you define the model architecture using the Sequential class, which allows you to stack layers on top of each other. Specify the number of layers, the number of neurons in each layer, and the activation functions for each layer.

Compile the Model: After defining the model, you need to compile it by specifying the optimizer, loss function, and any evaluation metrics.

Train the Model: Use the fit function to train the model on your training data. Specify the input data, target labels, batch size, number of epochs, and validation data if available.

Evaluate the Model: Once the training is complete, you can evaluate the model's performance on the testing data using the evaluate function.

Make Predictions: Finally, you can use the trained model to make predictions on new, unseen data using the predict function.

3.Illustrate Loading and Preprocessing Data with TensorFlow

Loading and preprocessing data with TensorFlow involves several steps to prepare the data for training or inference with machine learning models. Here's an illustration of the process:

Import the Required Libraries: Import the necessary libraries in your Python script or notebook. Typically, you'll need to import tensorflow and numpy.

Loading the Data: TensorFlow provides various methods for loading data, depending on the data format. The most common approach is to load data from files using TensorFlow's Dataset API.

Preprocessing the Data: Preprocessing steps are often required to transform the raw data into a suitable format for training. This may include tasks like normalizing the data, splitting it into features and labels, and applying any required transformations.

Creating TensorFlow Datasets: After preprocessing the data, it's often beneficial to convert it into TensorFlow Datasets for efficient training and batching.

Shuffling and Batching: To enhance the training process, it's common to shuffle the data and create mini-batches. TensorFlow provides functions to shuffle and batch the dataset.

Iterating Over the Dataset: To train or evaluate the model, you need to iterate over the dataset. TensorFlow provides an iterator or the for loop can be used.