**Assignment 1**

**Q1. Differentiate Supervised and Unsupervised Machine Learning?**

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

**Q2. What is Clustering? List out various major clustering methods?**

* **Clustering:** Clustering or Cluster analysis is the method of grouping the entities based on similarities. A cluster is a collection of objects which are “similar” amongst themselves and are “dissimilar” to the objects belonging to a different cluster. It is defined as an unsupervised learning problem that aims to make training data with a given set of inputs but without any target values, or labels In clustering, the machine learns the attributes and trends by itself without any provided input-output mapping. The clustering algorithms extract patterns and inferences from the type of data objects and then make discrete classes of clustering them suitably.
* **Clustering Methods: 1.** Partitioning Methods, **2.** Hierarchical Methods, **3.** Density-based Methods**, 4.** Grid Based Methods.

**Q3. Write a Short Note on:**

**A) Partitioning based Clustering Methods.**

* Given a set of n objects, a partitioning method constructs k partitionof the data, Where each partition represents a cluster and k <= n
* That is, it divides the data into k groups such that each group must contain at least one object.
* The basic partitioning methods typically adopt exclusive cluster separation. That is the object must belongs to exactly one group
* **Most popular partitioning methods are:**
* 1. K-Means Clustering
* 2. K-Medoids algorithm

**B) Hierarchical Clustering Methods.**

* A hierarchical method creates a hierarchical decomposition of the given set of data objects.
* A hierarchical method can be classified into agglomerative or divisive Approach.
* The agglomerative approach also called bottom up approach, starts with each
* object forming a separate group.
* It then successively merges the objects or groups close to one another, until all groups are merged into one. The divisive approach, also called top-down approach, starts with all the
* objects in the same cluster.
* It then split the cluster into smaller clusters, until each object is one cluster.

**Q4. With suitable example explain K-Means Algorithm?**

|  |  |
| --- | --- |
| import numpy as np  from sklearn.cluster import KMeans  from sklearn.datasets import make\_blobs  import matplotlib.pyplot as plt  **# Generate sample data**  X, y = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60, random\_state=0)   |  | | --- | |  |   **# Instantiate KMeans object with number of clusters**  kmeans = KMeans(n\_clusters=4)  **# Fit the KMeans object to the data**  kmeans.fit(X)  **# Get the cluster labels for each data point**  labels = kmeans.predict(X)  **# Get the coordinates of the cluster centers**  centroids = kmeans.cluster\_centers\_  **# Visualize the clusters and centroids**  plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=200, alpha=0.5)  plt.show() |

* In this example, we first generate sample data using the make\_blobs() function from sklearn.datasets. We create 300 data points with 4 clusters, a standard deviation of 0.60, and a random state of 0.
* Next, we instantiate a KMeans object with 4 clusters, fit it to the data using the fit() method, and get the predicted cluster labels for each data point using the predict() method.
* Finally, we get the coordinates of the cluster centers using the cluster\_centers\_ attribute and visualize the clusters and centroids using matplotlib.
* The KMeans algorithm works by iteratively assigning data points to the nearest cluster center and updating the cluster centers to minimize the sum of squared distances between each data point and its assigned cluster center. This process continues until convergence, when the cluster assignments no longer change.

**Q5. Explain Silhouette algorithm to choose K ?**

* Silhouette algorithm is one of the many algorithms to determine the optimal number of clusters for an unsupervised learning technique.
* In the Silhouette algorithm, we assume that the data has already been clustered into k clusters by a clustering technique.
* Silhouette analysis can be used to study the separation distance between the resulting clusters.
* The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like number of clusters visually. This measure has a range of [-1, 1].
* Silhouette coefficients near +1 indicate that the sample is far away from the Neighboring clusters.
* A value of 0 indicates that the sample is on or very close to the decision boundary between two Neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.

**Q6. What is Dimensionality Reduction, How PCA can be used to reduce Dimensions?**

* **Dimensionality Reduction:** In Machine Learning, dimension refers to the number of features in a particular dataset. Dimensionality Reduction refers to reducing dimensions or features so that we can get a more interpretable model and improves the performance of the model. There are basically three reasons for Dimensionality reduction: Visualization, Interpretability, Time and Space Complexity.
* **How PCA can be used to reduce Dimensions:** The principal component analysis is an unsupervised machine learning algorithm used for feature selection using dimensionality reduction techniques. PCA transforms and fits the data from a higher-dimensional space to a new, lower-dimensional subspace. PCA finds out the principal components from the data. This results into an entirely new coordinate system of the points where the first axis corresponds to the first principal component that explains the most variance in the data.

**Q7. With Suitable example explain Math Behind PCA?**

|  |  |
| --- | --- |
| x1 | x2 |
| 2 | 4 |
| 3 | 5 |
| 5 | 6 |
| 6 | 8 |

* Let's take a simple example to explain the math behind PCA. Suppose we have a dataset consisting of two features, x1 and x2, with the following data points:

|  |  |
| --- | --- |
| x1 | x2 |
| -2 | -1.75 |
| -1 | -0.75 |
| 1 | 0.25 |
| 2 | 2.75 |

* To perform PCA on this dataset, we first need to calculate the covariance matrix. The covariance matrix is a measure of how much two variables are linearly related to each other. It is a square matrix that has the same number of rows and columns as the number of features in the dataset.
* To calculate the covariance matrix, we first need to center the data by subtracting the mean of each feature from each data point. **The mean of x1 is (2+3+5+6)/4=4** and **the mean of x2 is (4+5+6+8)/4=5.75.** Subtracting these means from each data point, we get:
* Next, we calculate the covariance matrix by taking the dot product of the centered data matrix with its transpose:

|  |  |
| --- | --- |
| 2.5 | 3.5 |
| 3.5 | 4.5 |

* The covariance matrix tells us how much each feature varies and how much they are related to each other. In this case, the covariance between x1 and x2 is positive, which means that they are positively correlated.
* The next step in PCA is to calculate the eigenvectors and eigenvalues of the covariance matrix. Eigenvectors are special vectors that do not change direction when a linear transformation is applied to them. Eigenvalues are scalars that represent the amount of variance in the data that is explained by each eigenvector.
* To calculate the eigenvectors and eigenvalues, we solve the following equation: **C \* v = λ \* v**
* Where C is the covariance matrix, v is the eigenvector, and λ is the eigenvalue. The eigenvectors and eigenvalues can be calculated using linear algebra libraries such as NumPy or MATLAB.
* In our example, the eigenvectors and eigenvalues are:

**λ1 = 0.0322, v1 = [0.8208, 0.5714]**

**λ2 = 7.9678, v2 = [-0.5714, 0.8208]**

* The eigenvectors are unit vectors, which means that their magnitude is equal to 1. The eigenvalues represent the amount of variance explained by each eigenvector. In this case, the eigenvector v2 explains most of the variance in the data.

|  |  |
| --- | --- |
| -0.2271 | 2.5089 |
| -0.0605 | 0.4248 |
| 0.2059 | -0.1263 |
| 0.0817 | -2.8074 |

* The final step in PCA is to project the data onto the new feature space defined by the eigenvectors. To do this, we take the dot product of the centered data matrix with the eigenvectors:
* The new feature space has two dimensions, one for each eigenvector. We can see that the data is now

**Q8. How PCA is applied on Olevitti Images, demonstrate using suitable Python code?**

|  |
| --- |
| from sklearn.datasets import fetch\_olivetti\_faces  # Load the Olivetti dataset  olivetti = fetch\_olivetti\_faces()  X = olivetti.data |

* PCA (Principal Component Analysis) is a widely used technique for dimensionality reduction in machine learning and image processing. It is often applied to images to extract important features and reduce the dimensionality of the image data. To demonstrate how PCA can be applied to Olivetti images in Python, we first need to load the Olivetti dataset from the scikit-learn library:

|  |
| --- |
| from sklearn.decomposition import PCA  # Instantiate a PCA object with n\_components=100  pca = PCA(n\_components=100)  # Fit the PCA object to the Olivetti dataset  pca.fit(X)  # Transform the Olivetti dataset using the fitted PCA object  X\_pca = pca.transform(X) |

* Next, we can apply PCA to the dataset using the PCA class from scikit-learn:
* In this example, we have instantiated a PCA object with n\_components=100, which means that the PCA algorithm will extract the 100 most important components from the Olivetti dataset. We then fit the PCA object to the dataset using the fit method and transform the dataset using the transform method. The transformed dataset, X\_pca, will have 100 columns, one for each of the extracted components.
* We can also visualize the extracted components using matplotlib:

|  |
| --- |
| import matplotlib.pyplot as plt  # Plot the first 25 extracted components  fig, axes = plt.subplots(5, 5, figsize=(8, 8),  subplot\_kw={'xticks':[], 'yticks':[]},  gridspec\_kw=dict(hspace=0.3, wspace=0.3))  for i, ax in enumerate(axes.flat):  ax.imshow(pca.components\_[i].reshape(64, 64), cmap='gray')  plt.show() |

* This code will plot the first 25 extracted components as grayscale images. Each component represents a pattern that is commonly found in the Olivetti dataset. These patterns can be thought of as the building blocks of the images in the dataset.
* PCA can be applied to Olivetti images in Python using the PCA class from scikit-learn. The transformed dataset will have fewer dimensions, making it easier to analyze and process. The extracted components can also be visualized to gain insight into the patterns that are present in the dataset.