**Assignment (Session 21-24)**

**Problem Statement**

Task 1:

1. Use the below-given data set

Data Set

2. Perform the below-given activities:

a. Apply PCA to the dataset and show the proportion of variance

|  |
| --- |
| Variance of the PCA features |
|  |  |
|  | The fish dataset is 6-dimensional. But what is its intrinsic dimension? Make a plot of the variances of the PCA features to find out. As before, samples is a 2D array, where each row represents a fish. You'll need to standardize the features first. |
|  |  |
|  | INSTRUCTIONS |
|  | 100XP |
|  | Create an instance of StandardScaler called scaler. |
|  | Create a PCA instance called pca. |
|  | Use the make\_pipeline() function to create a pipeline chaining scaler and pca. |
|  | Use the .fit() method of pipeline to fit it to the fish samples samples. |
|  | Extract the number of components used using the .n\_components\_ attribute of pca. Place this inside a range() function and store the result as features. |
|  | Use the plt.bar() function to plot the explained variances, with features on the x-axis and pca.explained\_variance\_ on the y-axis. |
|  | ''' |
|  | # Perform the necessary imports |
|  | from sklearn.decomposition import PCA |
|  | from sklearn.preprocessing import StandardScaler |
|  | from sklearn.pipeline import make\_pipeline |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | # Create scaler: scaler |
|  | scaler = StandardScaler() |
|  |  |
|  | # Create a PCA instance: pca |
|  | pca = PCA() |
|  |  |
|  | # Create pipeline: pipeline |
|  | pipeline = make\_pipeline(scaler, pca) |
|  |  |
|  | # Fit the pipeline to 'samples' |
|  | pipeline.fit(samples) |
|  |  |
|  | # Plot the explained variances |
|  | features = range(pca.n\_components\_) |
|  | plt.bar(features, pca.explained\_variance\_) |
|  | plt.xlabel('PCA feature') |
|  | plt.ylabel('variance') |
|  | plt.xticks(features) |
|  | plt.show() |

b. Perform PCA using SVD approach

A simple K-Means Clustering model implemented in python. The class KMeans is imported from sklearn.cluster library. In order to find the optimal number of cluster for the dataset, the model was provided with different numbers of cluster ranging from 1 to 10. The 'k-means++' method to passed to the init argument to avoid the Random Initialization Trap. The max\_iter and the n\_init were passed with their default values.

The WCSS ( or Within Cluster Sum of Squares ) was caluated and plotted to find the optimal number of clusters. The "Elbow Method" was used to find the optimal number of clusters.

Once the optimal number of clusters were found the model was reinitalised with the n\_cluster arguments begin passed with the optimal number of clusters found using the "Elbow Method".

Finally, the clusters were visualised using scatter plot.