MACHINE LEARNING ASSIGNMENT-5

V.Sonalika
700742916

Video link:

https://drive.google.com/file/d/11Gqp7BhYEzekTV0UP9EMg2nmmEY0Wlbq/view?usp=sharing

Github link:

https://github.com/Sonalika2229/700742916_ML_Assign5/tree/main

Question1:

+ Code + Text

Question1: Principal Component Analysis a. Apply PCA on CC dataset. b. Apply k-means algorithm on the PCA result and report your observation if the silhouette score has improved or not? c. Perform Scaling+PCA+K-Means and report performance.

[74]
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
import numpy as np
from sklearn.datasets import load_iris
```

Imported all the required libraries

```
#Apply PCA on CC General data set

# Load the dataset
data = pd.read_csv('/content/drive/MyDrive/Mldataset/CC GENERAL.csv')
# Drop the categorical columns and ID column
data = data.drop(['CUST_ID', 'TENURE'], axis=1)
# Filling the missing values with mean of respective column
data = data.fillna(data.mean())

# Scale the data
scale = StandardScaler()
cc_scale = scale.fit_transform(data)

# Initialize PCA model
pca = PCA(n_components=2)

# Fit and transform the data using PCA
cc_pca = pca.fit_transform(cc_scale)

# Print the explained variance ratio
print('Explained variance ratio:', pca.explained_variance_ratio_)

# Create a new dataframe with the transformed data
cc_pca_df_new = pd.DataFrame(data=cc_pca, columns=['PC1', 'PC2'])

# Print the transformed data
print('Transformed data:', cc_pca_df_new.head())
```

```
Explained variance ratio: [0.28845814 0.21570572]
Transformed data:          PC1       PC2
0 -1.718893 -1.072939
1 -1.169304  2.509314
2  0.938413 -0.382598
3 -0.907502  0.045857
4 -1.637830 -0.684974
```

Initially loaded the dataset
Used the data.drop[] function that drops the categorical and ID column if they are present
data.fillna() function is used to fill the missing values with mean of each respective column
Scaled the data using the standard scaler
Initialized the PCA model with 2 components

pca.fit_transform() function is used to fit and transform the data using pca
Created a new dataframe with the transformed data
Printed the explained variance ratio and the transformed data.

```
[76] #Calculating the silhouette score without applying the PCA.

     # Load the dataset
     data = pd.read_csv('/content/drive/MyDrive/Mldataset/CC GENERAL.csv')
     # Drops the categorical columns and ID column if they are present
     data = data.drop(['CUST_ID', 'TENURE'], axis=1)

     # Fill any missing values with mean of each column
     data = data.fillna(data.mean())

     # Scale the data
     scale = StandardScaler()
     cc_scale = scale.fit_transform(data)

     # Initializing the k-means model
     k_means = KMeans(n_clusters=2)

     # Fit the k-means model to scaled data
     k_means.fit(cc_scale)

     # Calculating the silhouette value of the clustered data
     silhouette_val = silhouette_score(cc_scale, k_means.labels_)

     # Print the silhouette score
     print('Silhouette score:', silhouette_val)

     Silhouette score: 0.22590596639304975
```

Initially loaded the dataset
Used the data.drop[] function that drops the categorical and ID column if they are present
data.fillna() function is used to fill the missing values with mean of each respective column
Scaled the data using the standard scaler
Initialized the kmeans model with 2 components

Calculated the silhouette value of the clustered data

```
data = pd.read_csv('/content/drive/MyDrive/Mldataset/CC GENERAL.csv')

# Drop the categorical columns and ID column if they are present

data = data.drop(['CUST_ID', 'TENURE'], axis=1)

# Fill any missing values with mean of each column

data = data.fillna(data.mean())

# Scale the data

scale = StandardScaler()

cc_scale = scale.fit_transform(data)

# Initialize PCA model

pca = PCA(n_components=2)

# Fit and transform the data using PCA

cc_pca = pca.fit_transform(cc_scale)

# Initialize k-means model

kmeans = KMeans(n_clusters=2)

# Fit the k-means model on the PCA transformed data

kmeans.fit(cc_pca)

# Calculating the silhouette score of the clustered data

silhouette_val = silhouette_score(cc_pca, kmeans.labels_)
```

```
# Fit the k-means model on the PCA transformed data

kmeans.fit(cc_pca)

# Calculating the silhouette score of the clustered dat

silhouette_val = silhouette_score(cc_pca, kmeans.labels
# Print the silhouette score

print('Silhouette score:', silhouette_val)
```

```
Silhouette score: 0.46720635801791466
```

Initially loaded the dataset
Used the data.drop[] function that drops the categorical and ID column if they are present
data.fillna() function is used to fill the missing values with mean of each respective column
Scaled the data using the standard scaler
Initialized the PCA model with 2 components
Initialized the kmeans model with 2 components
Fit the k-means model on the PCA transformed data
Calculated the silhouette value of the clustered data

```
[78] #Perform Scaling+PCA+K-Means and report performance with 2 clusters
     # Load the dataset

     data = pd.read_csv('/content/drive/MyDrive/Mldataset/CC GENERAL.csv')

     # Drop the categorical columns and ID column

     data = data.drop(['CUST_ID', 'TENURE'], axis=1)

     # Fill any missing values with mean of each column

     data = data.fillna(data.mean())

     # Scale the data
     scale = StandardScaler()
     cc_scale = scale.fit_transform(data)

     # Apply PCA
     pca = PCA(n_components=2)
     cc_pca = pca.fit_transform(cc_scale)

     # Initialize k-means model
     kmeans = KMeans(n_clusters=2)

     # Fit the k-means model on the PCA data
     kmeans.fit(cc_pca)

     # Calculate the silhouette value of the clustered data
     silhouette_value = silhouette_score(cc_pca, kmeans.labels_)

     # Print the silhouette score
     print('Silhouette score:', silhouette_value)

     Silhouette score: 0.46318933568854204
```

Initially loaded the dataset
Used the data.drop[] function that drops the categorical and ID column if they are present
data.fillna() function is used to fill the missing values with mean of each respective column
Scaled the data using the standard scaler
Initialized the PCA model with 2 components
Initialized the kmeans model with 2 components
Fit the k-means model on the PCA transformed data
Calculated the silhouette value of the clustered data

```
#Perform Scaling+PCA+K-Means and report performance with 3 clusters in kmeans
# Load the dataset
data = pd.read_csv('/content/drive/MyDrive/Mldataset/CC GENERAL.csv')

# Drop the categorical columns and ID column

data = data.drop(['CUST_ID', 'TENURE'], axis=1)

# Fill any missing values with mean of each column
data = data.fillna(data.mean())

# Scale the data
scale = StandardScaler()

cc_scale = scale.fit_transform(data)

# Apply PCA
pca = PCA(n_components=2)

cc_pca = pca.fit_transform(cc_scale)

# Initialize k-means model

kmeans = KMeans(n_clusters=3, random_state=42)

# Fit the k-means model on the PCA data

kmeans.fit(cc_pca)

# Calculate the silhouette score of the clustered data

silhouette_value = silhouette_score(cc_pca, kmeans.labels_)

# Print the silhouette score
print('Silhouette score:', silhouette_value)
```

Question 2:

```
[80] #Using pd_speech_features.csv to perform scaling
     # Load the dataset

     data = pd.read_csv('/content/drive/MyDrive/Mldataset/pd_speech_features.csv')

     # Splitting  the data into features and target variable

     N = data.iloc[:, 1:-1]
     y = data.iloc[:, -1]

     # Standardizing the features
     scale = StandardScaler()
     N_std = scale.fit_transform(N)

     # Created the new dataframe
     std_df = pd.DataFrame(N_std, columns=N.columns)

     # Adding the target variable

     std_df['Target'] = y

     # Visualizing the standardized data

     print(std_df.head())
```

Initially loaded the dataset
Splitted the data into the features and target variable
Standardizing the features using the standard scaler
Created a new dataframe and added the target variable to it
Visualized the standard data

```python
# Apply PCA (k=3)

# Load the dataset
data = pd.read_csv('/content/drive/MyDrive/Mldataset/pd_speech_features.csv')

# Splitting the data into features and target variable

N = data.iloc[:, 1:-1]
y = data.iloc[:, -1]

# Standardize the features

scale = StandardScaler()
N_std = scale.fit_transform(N)

# Create a PCA object

pca = PCA(n_components=3)

# Fit the PCA model on the standardized data

pca.fit(N_std)

# Transform the data to the new coordinate system

N_pca = pca.transform(N_std)

# Visualize the explained variance ratio
print("Explained Variance Ratio:", pca.explained_variance_ratio_)

# Created a new dataframe with the transformed data

pca_df = pd.DataFrame(data=N_pca, columns=['PC1', 'PC2', 'PC3'])
```

```python
# Add the target variable to the new dataframe

pca_df['Target'] = y

# Visualize the transformed data

print(pca_df.head())
```

```
Explained Variance Ratio: [0.12961998 0.09390046 0.08252524]
         PC1       PC2        PC3  Target
0 -10.034309  1.473186  -6.836298       1
1 -10.624667  1.585847  -6.820881       1
2 -13.503155 -1.251541  -6.809195       1
3  -9.143503  8.834664  15.302886       1
4  -6.752753  4.612583  15.649158       1
```

Initially loaded the dataset
Splitted the data into the features and target variable
Standardizing the features using the standard scaler

Created the PCA with 3 components
pca.fit() function is used to fit the PCA model on the standardized data
Visualized the explained variance ratio and the transformed data

```
[82] #Use SVM to report performance
     # Load the dataset
     data = pd.read_csv('/content/drive/MyDrive/Mldataset/pd_speech_features.csv')

     # Split the data into features and target variable
     X = data.iloc[:, 1:-1]
     y = data.iloc[:, -1]

     # Standardize the features
     scale = StandardScaler()
     X_std = scale.fit_transform(X)

     # Creating a PCA object
     pca = PCA(n_components=3)

     # Fit the PCA model
     pca.fit(X_std)

     # Transform the data
     X_pca = pca.transform(X_std)

     # Splitting the transformed data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=42)

     # Creating an Support vector machine object with a linear kernel
     SVM = SVC(kernel='linear', random_state=42)

     # Fit the SVM model
     SVM.fit(X_train, y_train)

     # Predictting the target variable for the testing data
     y_pred = SVM.predict(X_test)

     # finding the accuracy of the model
     accuracy = accuracy_score(y_test, y_pred)
```

```
Accuracy: 0.775330396475771
```

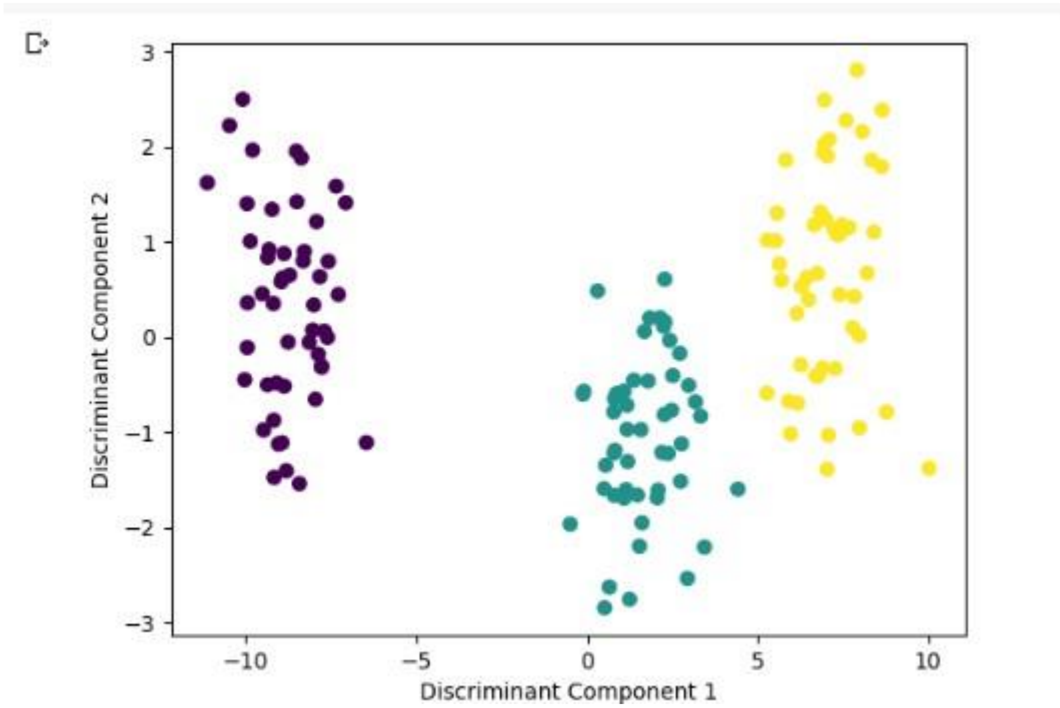
Initially loaded the dataset
Splitted the data into the features and target variable
Standardizing the features using the standard scaler
Created the PCA with 3 components

pca.fit() function is used to fit the PCA model on the standardized data
Splitted the transformed data into testing and training sets
Created a SVM with linear kernel
SVM.fit() function is used to fit the SVM model
Predicted the target variable for the testing data and found the accuracy of the model.

Question 3:

Question-3: Apply Linear Discriminant Analysis (LDA) on Iris.csv dataset to reduce dimensionality of data to k=2

```
[83] # Load the dataset
data = pd.read_csv('/content/drive/MyDrive/Mldataset/Iris.csv')

# Splitting the data into the features and target variable
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

# Converting the target variable into numeric format
y = pd.factorize(data.iloc[:, -1])[0]

# Create an LDA object
lda = LinearDiscriminantAnalysis(n_components=2)

# Fit the LDA model on the data
#and transform the data to the new coordinate system

X_lda = lda.fit_transform(X, y)

# Plot the transformed data in a scatter plot, coloring points by target variable
plt.scatter(X_lda[:, 0], X_lda[:, 1], c=y, cmap='viridis')
plt.xlabel('Discriminant Component 1')
plt.ylabel('Discriminant Component 2')
plt.show()
```

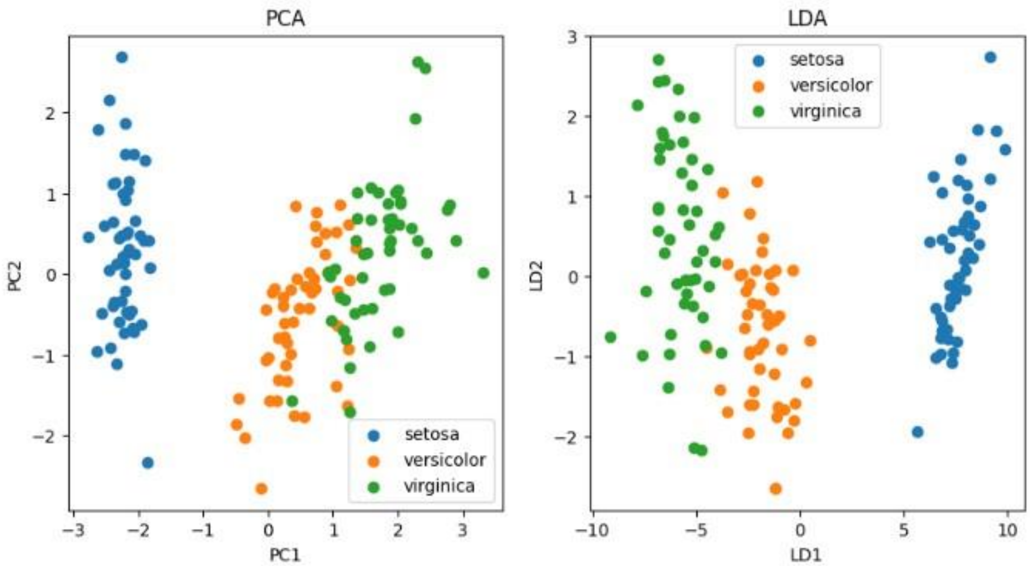


Initially loaded the dataset
Splitted the data into the features and target variable
Converted the target variable into numeric format
Created an LDA object
Fit the LDA model on the data and transform the data to the new coordinate system
Plotted the transformed data
Question 4:

Question-4:Briefly identify the difference between PCA and LDA

[84]
```
# Load the iris dataset
iris = load_iris()
X = iris.data
y = iris.target
targetnames = iris.target_names

# Scale the features
scale = StandardScaler()
X_scaled = scale.fit_transform(X)

# Applying PCA with k=2
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Applying LDA with k=2
lda = LinearDiscriminantAnalysis(n_components=2)
X_lda = lda.fit_transform(X_scaled, y)

# Plot the results
plt.figure(figsize=(10, 5))
plt.subplot(121)
for i, target_name in enumerate(targetnames):
    plt.scatter(X_pca[y == i, 0], X_pca[y == i, 1], label=target_name)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('PCA')
plt.legend()

plt.subplot(122)
for i, target_name in enumerate(targetnames):
    plt.scatter(X_lda[y == i, 0], X_lda[y == i, 1], label=target_name)
plt.xlabel('LD1')
plt.ylabel('LD2')
```

```python
for i, target_name in enumerate(targetnames):
    plt.scatter(X_lda[y == i, 0], X_lda[y == i, 1], label=target_name)
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.title('LDA')
plt.legend()

plt.show()
```



Initially loaded the dataset

Scaled the features using the standard scaler

Applied PCA with Number of components 2

Applied LDA with Number of components 2

Plotted the results.