Spring 2023 5710 Machine Learning: Assignment 5

Singam Manasvi

Course Id: CS 5710

CRN :23921

Student Id:700742501

GitHub Link: https://github.com/singammanasvi9440/Assignment_5

1.

```
[1]  # import the libraries
     import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
     import seaborn as sns
     from sklearn import preprocessing, metrics
     from sklearn.preprocessing import StandardScaler, LabelEncoder
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
     from sklearn.decomposition import PCA
     from sklearn.cluster import KMeans
     sns.set(style="white", color_codes=True)
     import warnings
     warnings.filterwarnings("ignore")
```

Question-1: Principal Component Analysis

```
df = pd.read_csv("/content/CC GENERAL.csv")
df.head()

#Read the CC General file
```

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FRE
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.

Check the features of the CC dataset.

```
df.isnull().any()
# check null values in the dataset using isnull() function
```

```
CUST_ID                              False
BALANCE                              False
BALANCE_FREQUENCY                    False
PURCHASES                            False
ONEOFF_PURCHASES                     False
INSTALLMENTS_PURCHASES               False
CASH_ADVANCE                         False
PURCHASES_FREQUENCY                  False
ONEOFF_PURCHASES_FREQUENCY           False
PURCHASES_INSTALLMENTS_FREQUENCY     False
CASH_ADVANCE_FREQUENCY               False
CASH_ADVANCE_TRX                     False
PURCHASES_TRX                        False
CREDIT_LIMIT                          True
PAYMENTS                             False
MINIMUM_PAYMENTS                      True
PRC_FULL_PAYMENT                     False
TENURE                               False
dtype: bool
```

```
df.fillna(df.mean(), inplace=True)
df.isnull().any()

#replace the null data with the mean
```

```
CUST_ID                              False
BALANCE                              False
BALANCE_FREQUENCY                    False
PURCHASES                            False
ONEOFF_PURCHASES                     False
INSTALLMENTS_PURCHASES               False
CASH_ADVANCE                         False
PURCHASES_FREQUENCY                  False
```

EDA on the dataset.

a.

```
[5] x = df.iloc[:,1:-1]
    y = df.iloc[:,-1]
    print(x.shape,y.shape)

    (8950, 16) (8950,)
```

a. Apply PCA on CC dataset

```
# a. Apply PCA on CC dataset.
#Datasets can be analyzed with PCA so that redundant features can be removed without losing too much information.
pca = PCA(3)   #Instantiate PCA
x_pca = pca.fit_transform(x)
principalDf = pd.DataFrame(data = x_pca, columns = ['principal component 1', 'principal component 2', 'principal compone
finalDf = pd.concat([principalDf, df.iloc[:,-1]], axis = 1)
finalDf.head()

#PCA(3)- performs principal component analysis (PCA) on dataset x, reducing the dimensionality of the data from the orig
#fit_transform()- method of the PCA object is called on the data x to obtain a transformed version of the data, where ea
#principalDf- represents the transformed data x_pca and three principal components
#finalDf- concatenating principalDf with the last column of the original DataFrame df using pd.concat(). This is likely
```

	principal component 1	principal component 2	principal component 3	TENURE
0	-4326.383979	921.566882	183.708383	12
1	4118.916665	-2432.846346	2369.969289	12
2	1497.907641	-1997.578694	-2125.631328	12
3	1394.548536	-1488.743453	-2431.799649	12

Applying PCA on CC dataset.

b.

```
[7] # b. Apply k-means algorithm on the PCA result and report your observation if the silhouette score has improved or not?
X = finalDf.iloc[:,0:-1]
y = finalDf.iloc[:,-1]
print(X.shape,y.shape)

#X- predictor variable- contains all rows of finalDf except for the last column, representing the principal components g
#y- target variable- contains only the last column of finalDf, representing the target variable.
```

```
(8950, 3) (8950,)
```

```
nclusters = 3 # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X)

# predict the cluster for each data point
y_cluster_kmeans = km.predict(X)

# Summary of the predictions made by the classifier
print(classification_report(y, y_cluster_kmeans, zero_division=1))
print(confusion_matrix(y, y_cluster_kmeans))

#finding the accuracy
train_accuracy = accuracy_score(y, y_cluster_kmeans)
print("\nAccuracy for our Training dataset with PCA:", train_accuracy)

#Calculating sihouette Score
score = metrics.silhouette_score(X, y_cluster_kmeans)
print("Sihouette Score: ",score)   #ranges from -1 to +1, high value shows that it is matched more
```

Applying k-means algorithm on the PCA result.

c.

```
[9]
x = df.iloc[:,1:-1]
y = df.iloc[:,-1]
print(x.shape,y.shape)
```

```
(8950, 16) (8950,)
```

```
## Scale the dataset; This is very important before you apply PCA
scaler = StandardScaler()
scaler.fit(x)
X_scaled_array = scaler.transform(x)

# Instantiate PCA
pca = PCA(3)

# Determine transformed features
x_pca = pca.fit_transform(X_scaled_array)
principalDf = pd.DataFrame(data = x_pca, columns = ['principal component 1', 'principal component 2','principal componer
finalDf = pd.concat([principalDf, df.iloc[:,-1]], axis = 1)
finalDf.head()
```

	principal component 1	principal component 2	principal component 3	TENURE
0	-1.718892	-1.072939	0.535728	12
1	-1.169312	2.509307	0.627441	12
2	0.938416	-0.382596	0.161391	12
3	-0.907504	0.045855	1.521540	12
4	-1.637828	-0.684972	0.425833	12

Perform Scaling + PCA + K-Means.

```
[11] x = finalDf.iloc[:,0:-1]
     y = finalDf["TENURE"]
     print(X.shape,y.shape)

     (8950, 3) (8950,)
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.34,random_state=0)
nclusters = 3
# this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X_train,y_train)


# predict the cluster for each training data point
y_clus_train = km.predict(X_train)

# Summary of the predictions made by the classifier
print(classification_report(y_train, y_clus_train, zero_division=1))
print(confusion_matrix(y_train, y_clus_train))

train_accuracy = accuracy_score(y_train, y_clus_train)
print("Accuracy for our Training dataset with PCA:", train_accuracy)

#Calculating sihouette Score
score = metrics.silhouette_score(X_train, y_clus_train)
print("Sihouette Score: ",score)   #ranges from -1 to +1, high value shows that it is matched more
```

```
              precision    recall  f1-score   support

           0       0.00      1.00      0.00       0.0
           1       0.00      1.00      0.00       0.0
           2       0.00      1.00      0.00       0.0
           6       1.00      0.00      0.00     139.0
           7       1.00      0.00      0.00     135.0
```

```
# predict the cluster for each testing data point
y_clus_test = km.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_clus_test, zero_division=1))
print(confusion_matrix(y_test, y_clus_test))

train_accuracy = accuracy_score(y_test, y_clus_test)
print("\nAccuracy for our Testing dataset with PCA:", train_accuracy)

#Calculating sihouette Score
score = metrics.silhouette_score(X_test, y_clus_test)
print("Sihouette Score: ",score)   #ranges from -1 to +1, high value shows that it is matched more


#First scale the data Applies the fit_transform() method of the StandardScaler instance to the feature matrix X to perfc
#This method first computes the mean and standard deviation of each feature in X, and then scales the features such that
#Then apply PCA to reduce the dimensionality to 3 components.
#Then split the data into training and testing sets using the train_test_split() function.
#Perform K-means clustering on the training set and test set and predict the cluster for each training data point.
#Finally, evaluate the performance of the clustering on the training & training set using classification_report(), confu
```

```
              precision    recall  f1-score   support

           0       0.00      1.00      0.00       0.0
           1       0.00      1.00      0.00       0.0
           2       0.00      1.00      0.00       0.0
           6       1.00      0.00      0.00      65.0
           7       1.00      0.00      0.00      55.0
           8       1.00      0.00      0.00      68.0
           9       1.00      0.00      0.00      57.0
          10       1.00      0.00      0.00      85.0
          11       1.00      0.00      0.00     103.0
          12       1.00      0.00      0.00    2610.0
```

2.

a.

```
df_pd = pd.read_csv("/content/pd_speech_features.csv")
df_pd.head()
```

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...

5 rows × 755 columns

```
df_pd.isnull().any()
```

```
id                         False
gender                     False
PPE                        False
DFA                        False
RPDE                       False
                           ...
tqwt_kurtosisValue_dec_33  False
tqwt_kurtosisValue_dec_34  False
tqwt_kurtosisValue_dec_35  False
```

```
df_pd.isnull().any()
```

```
id                         False
gender                     False
PPE                        False
DFA                        False
RPDE                       False
                           ...
tqwt_kurtosisValue_dec_33  False
tqwt_kurtosisValue_dec_34  False
tqwt_kurtosisValue_dec_35  False
tqwt_kurtosisValue_dec_36  False
class                      False
Length: 755, dtype: bool
```

```
X = df_pd.drop('class',axis=1).values
Y = df_pd['class'].values

# this codes represents dropping the target variable class from main data frame and creates a new data fram X
# Y returns the class column from the main data frame
```

```
# a. Perform Scaling

#Scaling Data
scaler = StandardScaler()
X_Scale = scaler.fit_transform(X)

#StandardScaler to scale the input X, this is important as it ensures that all the features are on the same scale and pr
#Applies the fit_transform() method of the StandardScaler instance to the feature matrix X to perform feature scaling
```

StandardScaler to scale the input X, this is important as it ensures that all the features are on the same scale and prevents features with larger magnitude from dominating the distance calculations.

b.

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

# Apply PCA with k =3
pca3 = PCA(n_components=3)
principalComponents = pca3.fit_transform(X_Scale)

principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2','Prin

finalDf = pd.concat([principalDf, df_pd[['class']]], axis = 1)
finalDf.head()
```

	principal component 1	principal component 2	Principal Component 3	class
0	-10.047372	1.471076	-6.846404	1
1	-10.637725	1.583749	-6.830979	1
2	-13.516185	-1.253542	-6.818698	1
3	-9.155084	8.833600	15.290904	1
4	-6.764470	4.611466	15.637123	1

```python
X = finalDf.drop('class',axis=1).values
Y = finalDf['class'].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,random_state=0)
```

Apply PCA.

c.

```python
# c. Use SVM to report performance

from sklearn.svm import SVC

svmClassifier = SVC()
svmClassifier.fit(X_train, Y_train)

y_pred = svmClassifier.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(Y_test, y_pred, zero_division=1))
print(confusion_matrix(Y_test, y_pred))

# Accuracy score
glass_acc_svc = accuracy_score(y_pred,Y_test)
print('accuracy is',glass_acc_svc)

#Calculate sihouette Score
score = metrics.silhouette_score(X_test, y_pred)
print("Sihouette Score: ",score)

#It then trains an SVM classifier on the training set, predicts the classes for the test set using the trained classifie
```

```
              precision    recall  f1-score   support

           0       0.67      0.42      0.52        57
           1       0.83      0.93      0.88       170

    accuracy                           0.80       227
   macro avg       0.75      0.68      0.70       227
weighted avg       0.79      0.80      0.79       227

[[ 24  33]
 [ 12 158]]
```

Apply SVM to check the performance.

3.

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

A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
df_iris = pd.read_csv("/content/Iris.csv")
df_iris.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
[ ] df_iris.isnull().any()
```

```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

```
[ ] x = df_iris.iloc[:,1:-1]
    y = df_iris.iloc[:,-1]
    print(x.shape,y.shape)
```

Apply Linear Discriminant analysis on iris data set to reduce dimensionality of data to k=2

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
le = LabelEncoder()
y = le.fit_transform(y)
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components=2)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
print(X_train.shape,X_test.shape)

#fit and transform the scaler object on our training data and only transform our test data.
#LabelEncoder to encode our target variable y into numerical values.
#(LDA) to perform dimensionality reduction on our input features x. Here, we are reducing the number of input features
#we transform our training and test data using the fit_transform and transform methods of the LDA object respectively
```

```
(105, 2) (45, 2)
```

4.

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

PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis) are both popular techniques in machine learning for dimensionality reduction. However, they have different purposes and methods:

Purpose: PCA is used for unsupervised learning and finds the directions of maximum variance in a dataset. It reduces the number of features by transforming the original dataset into a new coordinate system, where the features are uncorrelated and sorted by their variance. PCA is commonly used for data compression, visualization, and noise reduction. LDA, on the other hand, is used for supervised learning and aims to find the linear combinations of features that best separate the classes. It reduces the number of features by projecting the original dataset onto a lower-dimensional space while maximizing the class separability. LDA is commonly used for feature extraction, pattern recognition, and classification.

Method: PCA operates by finding the eigenvectors and eigenvalues of the covariance matrix of the data. The eigenvectors represent the directions of maximum variance, and the eigenvalues represent the amount of variance explained by each eigenvector. PCA selects the top k eigenvectors, where k is the desired dimensionality of the reduced dataset. LDA, on the other hand, maximizes the between-class scatter and minimizes the within-class scatter of the data. It involves finding the eigenvectors and eigenvalues of the product of two matrices: the between-class scatter matrix and the within-class scatter matrix. LDA selects the top k eigenvectors that correspond to the largest eigenvalues.