Student ID: 700747817

Name: Manikanta Venkata Kondasai Kalyan Badeti Parameswara

Machine Learning - Assignment 5

**Video Link:**

<https://drive.google.com/file/d/1qnUrUBnViDmG9D9QhOXEG9AStsqhGuVz/view?usp=sharing>

GitHub Link: <https://github.com/Kalyansai6/ML---Assignment-5>

1. Principal Component Analysis

a. Apply PCA on CC dataset.

To do data analysis and apply machine learning algorithms on data, first I imported a few python libraries.

Using read\_csv method imported “CC“ data set. The head() method of pandas library results topmost rows of a data set.

*from sklearn.decomposition import PCA*

*from sklearn.preprocessing import StandardScaler*

*from sklearn.model\_selection import train\_test\_split*

*import pandas as pd*

*from sklearn.metrics import accuracy\_score*

*import matplotlib.pyplot as plt*

*import warnings*

*warnings.filterwarnings('ignore')*

*df= pd.read\_csv(r"CC.csv") # reading cc data set*

*df.head() # results top most rows in a data set*

Graphical user interface, text, application, email

Description automatically generated

Isnull() method of pandas library checks for any values present in data set.

*df.isnull().sum() #checking any null values are present*

Graphical user interface, text

Description automatically generated

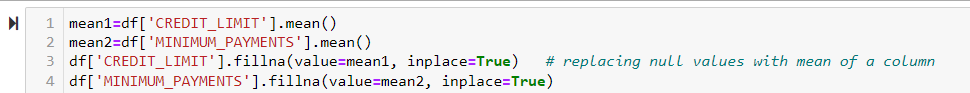
In this data set there are a few null values present in minimum payments and credit limit columns. So, these null values are replaced with their column mean value using fillna() method.

*mean1=df['CREDIT\_LIMIT'].mean()*

*mean2=df['MINIMUM\_PAYMENTS'].mean()*

*df['CREDIT\_LIMIT'].fillna(value=mean1, inplace=True) # replacing null values with mean of a column*

*df['MINIMUM\_PAYMENTS'].fillna(value=mean2, inplace=True)*

**

To perform PCA on this data set we don’t need the output labels because PCA does not rely on the output labels. Using the drop method, we removed a few columns which are unnecessary.

*X = df.drop(['TENURE','CUST\_ID'],axis=1).values # preprocessing the data by removing the columns*

*y = df['TENURE'].values*

**

From sklearn python library we imported PCA method to perform PCA on the data set. PCA results in a data frame with features having maximum variance with other features by ignoring the duplicate features. Here we reduced the dimensionality of data into two components by keeping k value is equal to 2.

*# performing PCA*

*pca2 = PCA(n\_components=2)*

*principalComponents = pca2.fit\_transform(X) # pca is applied on the data set without output labels*

*# creating a data frame for the pca results*

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

*# adding a new column to the data frame*

*finalDf = pd.concat([principalDf, df[['TENURE']]], axis = 1)*

*finalDf # printing the results*

*Graphical user interface, text, application

Description automatically generated*

b. Apply k-means algorithm on the PCA result and report your observation if the silhouette score has improved or not?

To perform k-means algorithm on a data set first we need to find the number of clusters required to fit our data together into clusters by using elbow method. This elbow method results in a graph from where we need to find the number of clusters value i.e., k value.

*# Use the elbow method to find a good number of clusters with the K-Means algorithm*

*from sklearn.cluster import KMeans*

*wcss = []*

*for i in range(1,11):*

*kmeans = KMeans(n\_clusters=i,init='k-means++',max\_iter=300,n\_init=10,random\_state=0)*

*kmeans.fit(X)*

*wcss.append(kmeans.inertia\_)*

*plt.plot(range(1,11),wcss)*

*plt.title('the elbow method')*

*plt.xlabel('Number of Clusters')*

*plt.ylabel('Wcss')*

*plt.show()*

The elbow method results in a graph. From graph, the next point to point where the wcss value starts decreasing linearly will be the k value. From the graph below, from number of clusters is 2 the wcss value starts decreasing linearly. So, the number of clusters required to fit our data is 3 i.e., k value is 3. In k-means algorithm k is the number of clusters.

***Chart

Description automatically generated***

Using KMeans method of sklearn library, I applied K- Means algorithm on data set we got after performing PCA. After performing k-means on the PCA data we got a silhouette score of 57% which is higher than the silhouette score of raw data without performing PCA.

The silhouette score has been improved when we perform PCA on the data set. when we applied kmeans on the data set without performing PCA we got a silhouette score of 46.5%. After performing PCA we got a silhouette score of 57%. The silhouette score has been improved by more than 10%.

*# Calculate the silhouette score for the above clustering*

*nclusters = 3 # this is the k in kmeans*

*km = KMeans(n\_clusters=nclusters)*

*km.fit(finalDf) # fitting out kmeans model with our data set*

*y\_cluster\_kmeans = km.predict(finalDf)*

*from sklearn import metrics*

*score = metrics.silhouette\_score(finalDf, y\_cluster\_kmeans)*

*print(score)*

*Text

Description automatically generated*

c. Perform Scaling + PCA + K-Means and report performance.

Using StandardScalar method we performed feature scaling on the data set. Feature scaling is used to normalize the range of all features.

*scaler = StandardScaler() # feature scaling using standard scaler*

*X\_Scale = scaler.fit\_transform(X)*

**

We are performing PCA on the feature scaled data set using the PCA method.

*# performing pca*

*pca3 = PCA(n\_components=2)*

*principalComponents1 = pca3.fit\_transform(X\_Scale)*

*principalDf1 = pd.DataFrame(data = principalComponents1, columns = ['principal component 1', 'principal component 2'])*

*finalDf2 = pd.concat([principalDf1, df[['TENURE']]], axis = 1)*

*finalDf2*

*Graphical user interface, text

Description automatically generated*

To perform k-means algorithm on a data set first we need to find the number of clusters required to fit our data together into clusters by using elbow method.

*# Use the elbow method to find a good number of clusters with the K-Means algorithm*

*from sklearn.cluster import KMeans*

*wcss = []*

*for i in range(1,11):*

*kmeans = KMeans(n\_clusters=i,init='k-means++',max\_iter=300,n\_init=10,random\_state=0)*

*kmeans.fit(finalDf2)*

*wcss.append(kmeans.inertia\_)*

*plt.plot(range(1,11),wcss)*

*plt.title('the elbow method')*

*plt.xlabel('Number of Clusters')*

*plt.ylabel('Wcss')*

*plt.show()*

The elbow method results in a graph. From graph, the next point to point where the wcss value starts decreasing linearly will be the k value. From the graph below, from number of clusters is 2 the wcss value starts decreasing linearly. So, the number of clusters required to fit our data is 3 i.e., k value is 3.

***Graphical user interface

Description automatically generated with low confidence***

Using KMeans method of sklearn library, I applied K- Means algorithm by taking k value as 3 on data set, we got after performing feature scaling and PCA. After performing k-means on this data we got a silhouette score of 38%.

*#Calculate the silhouette score for the above clustering*

*nclusters = 3 # this is the k in kmeans*

*km = KMeans(n\_clusters=nclusters)*

*km.fit(finalDf2)*

*y\_cluster\_kmeans = km.predict(finalDf2)*

*from sklearn import metrics*

*score = metrics.silhouette\_score(finalDf2, y\_cluster\_kmeans)*

*print(score)*

*Graphical user interface, text, application

Description automatically generated*

2. Use pd\_speech\_features.csv

Using read\_csv method imported a csv file. The head() method of pandas library results top most rows of a data set.

*df1= pd.read\_csv(r"pd\_speech\_features.csv") # reading pd\_speech\_features csv file*

*df1.head()*

*A picture containing graphical user interface

Description automatically generated*

1. Perform Scaling

Using StandardScalar method we performed feature scaling on the data set. Feature scaling is used to normalize the range of all features

*scaler = StandardScaler() #performing feature selection*

*X\_Scale = scaler.fit\_transform(X)*

*Text

Description automatically generated*

1. Apply PCA (k=3)

To perform PCA on this data set we don’t need the output labels because PCA does not rely on the output labels. Using the drop method, we removed a class column which is unnecessary.

*X = df1.drop('class',axis=1).values # preprocessing the data*

*y = df1['class'].values*

**

To run PCA on the data set, we imported the PCA method from the Sklearn Python package. By discarding the duplicate features, PCA produces a data frame with features that have the greatest variation with other characteristics. Here, we kept the k value at 3, which reduced the data's dimension to three components.

*# performing pca*

*pca4 = PCA(n\_components=3)*

*principalComponents2 = pca4.fit\_transform(X\_Scale)*

*principalDf2 = pd.DataFrame(data = principalComponents2, columns = ['principal component 1', 'principal component 2',*

*'principal components 3'])*

*finalDf3 = pd.concat([principalDf2, df1[['class']]], axis = 1)*

*finalDf3*

*Table

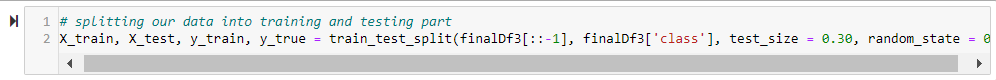
Description automatically generated with low confidence*

1. Use SVM to report performance

sklearn module contains train\_test\_split method to split our data set into training and testing data sets. In this method, test\_size defines how much proportion of data to be in the test data set. When we change test\_size value whole analysis results will change.

*# splitting our data into training and testing part*

*X\_train, X\_test, y\_train, y\_true = train\_test\_split(finalDf3[::-1], finalDf3['class'], test\_size = 0.30, random\_state = 0)*

**

Support vector machine algorithm is applied to the data set we got after performing PCA using sklearn module. We got an accuracy of 74.8% when we trained SVM on our data set.

*# training and predcting svm model on our data set*

*from sklearn.metrics import confusion\_matrix*

*from sklearn.metrics import classification\_report*

*# Support Vector Machine's*

*from sklearn.svm import SVC*

*classifier = SVC()*

*classifier.fit(X\_train, y\_train)*

*y\_pred = classifier.predict(X\_test)*

*# Summary of the predictions made by the classifier*

*print(classification\_report(y\_true, y\_pred))*

*print(confusion\_matrix(y\_true, y\_pred))*

*# Accuracy score*

*from sklearn.metrics import accuracy\_score*

*print('accuracy is',accuracy\_score(y\_pred,y\_true))*

The Machine learning algorithms are being compared with each other based on different metrices like F1-score, accuracy, recall and precision. Based on those values we are going to determine which algorithm is the most accurate and complete classifier for the dataset.

Text

Description automatically generated

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

A csv file was imported using the read\_csv method. The top rows of a data set are returned by the pandas library's head() method.

*df2= pd.read\_csv("Iris.csv") # reading iris csv file*

*df2.head()*

*Graphical user interface

Description automatically generated with medium confidence*

Isnull() method of pandas library checks for any values present in data set. In this iris dataset there are no null values.

*df2.isnull().any() # checking null values*

*Graphical user interface, text

Description automatically generated*

To perform LDA on this data set we need the output labels because LDA rely on these output labels to reduce the dimensionality of data based on output classes.

*X = df2.iloc[:, 1:5].values # preprocessing the data*

*y = df2.iloc[:, 5].values*

*A picture containing icon

Description automatically generated*

The LinearDiscriminantAnalysis class of the sklearn.discriminant\_analysis library can be used to Perform LDA in Python. By setting n\_components value as 2 we will get the results in two linear discriminates. We execute the fit and transform methods to retrieve our results.

*# performing lda on the data set*

*from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA*

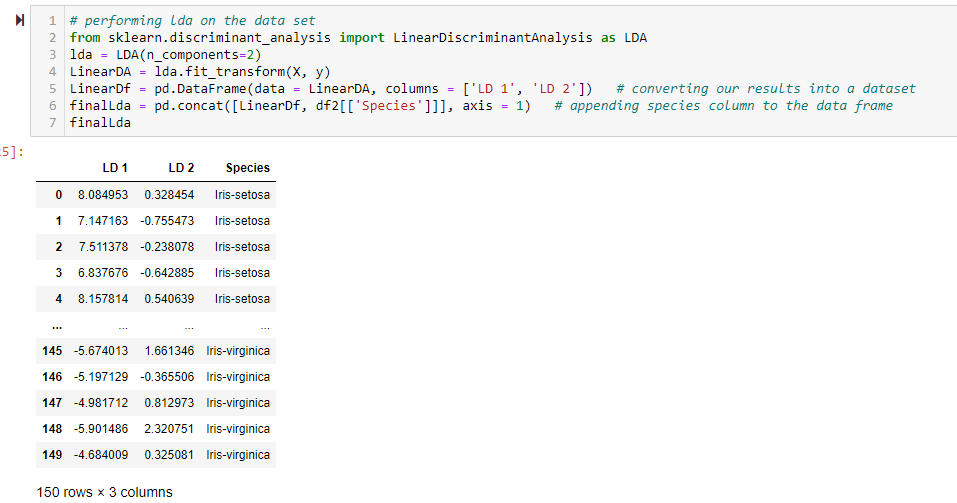
*lda = LDA(n\_components=2)*

*LinearDA = lda.fit\_transform(X, y)*

*LinearDf = pd.DataFrame(data = LinearDA, columns = ['LD 1', 'LD 2']) # converting our results into a dataset*

*finalLda = pd.concat([LinearDf, df2[['Species']]], axis = 1) # appending species column to the data frame*

*finalLda*

**

4. Briefly identify the difference between PCA and LDA

Dimensionality reduction in machine learning refers to the process of collecting a collection of major variables to reduce the number of random variables being considered. Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two main algorithms in dimensionality reduction.

PCA is an unsupervised while LDA is a supervised dimensionality reduction technique.

PCA gets the results without depending on the output labels. PCA results a data set with maximum variance between the features by ignoring the duplicates of other features. Since the variance between the features is independent of the outcome, PCA does not consider the output labels.

LDA depends on the output labels. Based on the output labels information LDA reduces the feature set dimensions and finds a decision boundary. The data points are then projected to new dimensions so that the clusters are as distinct from one another as possible, and the individual components of a cluster are as near the cluster centroid as possible.