ML_Assignment_5

Sushmitha_Virri

#700742289

Github Link: https://github.com/Sushmitha-Virri/MLAssignments21627/blob/main/700742289_Sushmitha_Virri_ML_Assignment_5.ipynb

Drive Video Link: https://drive.google.com/file/d/10XSx5hnwqFJPNDWGBte3bJkWGiMZHB0X/view?usp=sharing

1. Principal Component Analysis

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

In [295]:
```
# importing the necessary libraries
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
sns.set(style="white", color_codes=True)
```

In [296]:
```
df= pd.read_csv("CC GENERAL.csv") # loads the dataset from csv file into the pandas dataframe

df.head() #to display few lines in the dataframe 'df'
```

Out[296]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUEN
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083

In the above python script, we have imported the required modules from the libraries, including pandas for data manipulation, StandardScaler for feature scaling, PCA for dimensionality reduction, KMeans for clustering and metrics evaluation.

We have loaded the CC GENEREAL dataset from csv file to pandas data frame.

head() function displays the first few lines of the dataframe.

Next we use df.isnull().any() function to check whether there are any missing values. If any, fill them with mean of each column using fillna() function.

‘inplace = True’ argument will make changes to the original dataframe and is displayed in the output jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: an hour a

File Edit View Insert Cell Kernel Widgets Help

In [297]:
```
df.shape # using shape attribute to get the shape of df
```

Out[297]:
```
(8950, 18)
```

In [298]:
```
df.isnull().any()
```

Out[298]:
```
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
```

In [299]:
```
df.fillna(df.mean(), inplace = True)
df.isnull().any()
```

Out[299]:
```
CUST_ID                             False
BALANCE                             False
BALANCE_FREQUENCY                   False
PURCHASES                           False
ONEOFF_PURCHASES                    False
INSTALLMENTS_PURCHASES              False
CASH_ADVANCE                        False
PURCHASES_FREQUENCY                 False
```

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [300]:
```
x = df.iloc[:,[1,2,3,4]] # extacts 1,2,3,4 columns from the dataset and assign it to 'x'
y = df.iloc[:,-1] # extracts the last column and assign it to 'y'
print(x.shape, y.shape)
```

```
(8950, 4) (8950,)
```

The above code selects a subset of columns at index 1,2,3,4 from pandas Dataframe ‘df’ and assigns it to ‘x’ and selects all rows and last column and assign it to ‘y’.

Then it prints the shape of ‘x’ and ‘y’ which represent the number of rows and colums in the output cell.

a. Apply PCA on CC dataset.

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run Code

In [301]:
```
# a. Apply PCA on CC GENERAL dataset.
# PCA is used to analyse the high dimensional dataset
#     so as to reduce the complexity of the data and extract important features.

pca = PCA(3)   #creating a PCA with 3 principal components
x_pca = pca.fit_transform(x) # fits the PCA object to the dataset 'x' then tranforming it to new dataset 'x_pca'

# creating a new pandas dataframe 'df2' with less number of dimensions.
df2 = pd.DataFrame(data = x_pca, columns = ['principal component 1',
                                            'principal component 2', 'principal component 3'])

# Concatinates principalDF with last column of 'df' and create new dataframe 'df'
df3 = pd.concat([df2, df.iloc[:,-1]], axis = 1)
df3.head()
```

Out[301]:

	principal component 1	principal component 2	principal component 3	TENURE
0	-1500.250819	-1114.178407	-64.989145	12
1	-592.910661	1914.657567	-151.542222	12
2	217.734556	905.144354	-291.615901	12
3	927.782551	-198.923616	-421.617772	12
4	-1310.548986	-359.591021	-132.892069	12

In this code we applied PCA on CC GENERAL dataset. In the first line we created a ‘pca’ object with 3 principal components.

In the second line PCA object is fit to the original dataset ‘x’ and transformed to a new dataset ‘x_pca’

The third statement creates a new pandas dataframe ‘df2’ with principal component columns.

Next statement concatenates ‘df2’ with last column of ‘df’ and assigned to a new dataframe ‘df3’

The output cell shows reduced dimensions of the dataset.

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

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: an hour ago (unsaved ch

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [302]:
```python
# b. Apply k-means algorithm on the original dataset
x = df3.iloc[:,0:-1]
y = df3.iloc[:,-1]
print(x.shape, y.shape)
```

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

In [303]:
```python
from sklearn.cluster import KMeans
# setting the number of clusters to 3
nclusters = 3 # this is k = 3 in kmeans

# Applying K-means cluster on original dataset
# creating an instance of KMeans class
k_means = KMeans(n_clusters=nclusters)
k_means.fit(x) # to fit the

# to predict the cluster label for each data point
y_cluster_kmeans = k_means.predict(x)

from sklearn import metrics
# Calculating sihouette Score
score = metrics.silhouette_score(x, y_cluster_kmeans)
print("Sihouette Score without PCA: ",score)
```

```
Sihouette Score without PCA:  0.6024089140994977
```

The above screenshot talks about the application of KMeans clustering algorithm from scikit-learn library to the original dataset.

The number of clusters is set to 3 and stored in variable nclusters. KMeans class instance is created with this value and fit() method is called to fit the data.

Predict() method is used to predict the cluster label for each datapoint.

silhouette_score() function calculates the score with input values as original dataset and predicted labels.

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: 2 hours ago (autosa

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [304]: # Applying KMeans cluster on reduced dataset
kmeans_pca = KMeans(n_clusters = nclusters)
kmeans_pca.fit(x_pca)
y_cluster_kmeans_pca = kmeans_pca.predict(x_pca)

# Calculating sihouette Score
score_pca = metrics.silhouette_score(x_pca, y_cluster_kmeans_pca)
print("Sihouette Score with PCA: ",score_pca)

Sihouette Score with PCA:  0.6024089140994978
```

The above code apples KMeans clustering to preprocessed dataset with PCA.

An instance of KMeans class is created with nclusters. fit() method is called on this instance with input as ‘x_pca’ and predict() method is then called on kmeans_pca to predict the cluster labels for PCA transformed dataset x_pca.

Silhouette_score is then calculated with PCA-transformed dataset and cluster labels as input.

Performance: From the above we can observe that the silhouette score with PCA is slightly less than the silhouette score without PCA.

Reason: Implementing PCA has reduced the number of features in the dataset.

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

To perform dimensionality reduction using PCA and clustering using K-means on the dataset.

The code for this creates an instance of StandardScaler class and fits it on the training data ‘x’ and then transforms it into fitted scaler.

The code then performs feature scaling using StandardScaler, dimensionality reduction using PCA, clustering using KMeans and then evaluates clustering performance using silhouette score.

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: 3 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [305]:
```
# StandardScaler is used to scale the features to have mean '0' and variance '1'
scaler = StandardScaler()
scaler.fit(x) # for fitting on training data 'x'
x_scale = scaler.transform(x)

pca2 = PCA(3)
x_pca2 = pca.fit_transform(x_scale)

df4 = pd.DataFrame(data = x_pca2, columns = ['principal component 1',
                                  'principal component 2', 'principal component 3'])

final_df = pd.concat([df4, df[['TENURE']]], axis = 1)
final_df.head()

from sklearn.cluster import KMeans
nclusters = 3 # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(x_scale)

# predict the cluster for each data point
y_cluster_kmeans = km.predict(x_scale)
from sklearn import metrics
score = metrics.silhouette_score(x_scale, y_cluster_kmeans)
print(score)
```

```
0.5573894918696056
```

Performance report:

We can observe from the output that the silhouette score after applying Scaling + PCA + K-means is reduced when compared to the silhouette scores without PCA and with PCA.

From this we can say that as the silhouette score is not improved that is it is not greater than the previous values and hence the performance is not improved.

2. Use pd_speech_features.csv

a. Perform Scaling

b. Apply PCA (k=3)

c. Use SVM to report performance

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: 3 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [306]:
```
# to load the datadet from csv file into the pandas dataframe
df_pd = pd.read_csv(r"pd_speech_features.csv")
```

In [307]:
```
df_pd.head()
```

Out[307]:

	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

In [308]:
```
df_pd.isnull().any()
```

Out[308]:
```
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
```

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: 3 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [309]:
```
x = df_pd.drop('class',axis=1).values
y = df_pd['class'].values
print(x.shape,y.shape)
```

```
(756, 754) (756,)
```

In [310]:
```
#a. to perform Scaling
scaler = StandardScaler()
x_scale = scaler.fit_transform(x)
#b. Applying PCA for k=3
pca = PCA(3)
x_pca = pca.fit_transform(x_scale)

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

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

Out[310]:

	principal component 1	principal component 2	Principal Component 3	class
0	-10.047372	1.471076	-6.846404	1
1	-10.637725	1.583749	-6.830976	1
2	-13.516185	-1.253543	-6.818700	1
3	-9.155084	8.833600	15.290905	1
4	-6.764470	4.611467	15.637124	1

a. Scalaing:

In the first step we scale the data using StandardScaler() function and stored in the variable 'x_scale'.

b. PCA:

The next step is to perform PCA on the scaled data using PCA() function with k=3 and store it in the variable 'x_pca'.

Next we create a dataframe 'principalDF' with three principal components. The code then concatenates 'principalDF' with 'class' column using the pd.concat() function.

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: 3 hours ago (unsaved changes

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [311]:
```
#c. to use SVM and report performace

#spliting the data into training and testing data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,random_state=0)

from sklearn.svm import SVC
svm_classifier = SVC()
svm_classifier.fit(x_train, y_train)

#predict() method of the class SVC is used to predict the target variable for the test set.
y_pred = svm_classifier.predict(x_test)

# Evaluations made by the classifier
print(classification_report(y_test, y_pred, zero_division=1))
print(confusion_matrix(y_test, y_pred))

# Accuracy score
acc_svc = accuracy_score(y_pred,y_test)
print('accuracy is',acc_svc)

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

```
              precision    recall  f1-score   support

           0       1.00      0.02      0.03        57
           1       0.75      1.00      0.86       170

    accuracy                           0.75       227
   macro avg       0.88      0.51      0.45       227
weighted avg       0.81      0.75      0.65       227

[[  1  56]
 [  0 170]]
accuracy is 0.7533039647577092
Sihouette Score:  0.8052538192732682
```

c. Support Vector Machine(SVM):

The data is split into training and testing sets using train_test_split() function. The SVM classifier is trained on the training set using the fit() method.

classification_report() and confusion_matrix() are used to determine the performance of the classifier.

silhouette_score() measures the similarity of data points within the cluster and the dissimilarity of data points among different clusters.

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: 3 hours ago (unsaved chang

File Edit View Insert Cell Kernel Widgets Help

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

In [312]:
```
import math
import numpy as np
df_iris = pd.read_csv(r"Iris.csv")
df_iris.head()
```

Out[312]:

	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

In [313]:
```
df_iris.isnull().any()
```

Out[313]:
```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

In [317]:
```
x = df_iris.iloc[:,1:-1]
y = df_iris.iloc[:,-1]
print(x.shape,y.shape)
```

```
(150, 4) (150,)
```

The above code reads the data from ‘iris.csv’ file inti the pandas dataframe ‘df_iris’. Displays the first five columns. Next checks for any missing values in the dataframe, then separates dataframe into 2 parts using ‘iloc’ method.

1. ‘x’ which contain all features of dataset except target variable.
2. ‘y’ which contain only the target variable.

jupyter 700742289_Sushmitha_Virri_ML_Assignment_5 Last Checkpoint: 3 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [320]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
from sklearn.preprocessing import StandardScaler


le = LabelEncoder()
y = le.fit_transform(y)
# StandardScaler is used to standardize the data prior to modelling
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
le = LabelEncoder()
y = le.fit_transform(y)
#fit_transform method calculates mean and standard deviatiojn of every feature
x_train_std = scaler.fit_transform(df.iloc[:,1:-1].values)
```

In [321]:
```
from sklearn.preprocessing import LabelEncoder

#labelEncoder class encodes the categorical valued to numerical values
le = LabelEncoder()
y_le = le.fit_transform(df.iloc[:,-1].values)
```

In [328]:
```
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)
```

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

The above code performs following operations:

Splitting the data into training and testing sets using train_test_split' function from the sklearn library.

Encoding the target variable using 'LabelEncoder'

Standardizing the target variable using 'StandardScaler'

Then applying (LDA) Linear Discriminant Analysis to reduce feature dimension to 2.

Thus the code does preprocessing and reduced the dimensions using LDA.

4.Explain briefly the distinction between PCA and LDA.

LDA and PCA both use linear transformations to maximize variance in a smaller dimension. The PCA method is an unsupervised learning algorithm, whereas the LDA algorithm is a supervised learning system. This means that PCA seeks maximum variance directions regardless of class labels, whereas LDA finds maximum class separability directions.

LDA and PCA both use linear transformations to maximize variance in a lower dimension. The PCA method is an unsupervised learning algorithm, whereas the LDA algorithm is a supervised learning system. This indicates that PCA finds maximum variance directions regardless of class labels, whereas LDA finds maximum class separability directions.

PCA : It condenses the characteristics into a smaller set of orthogonal variables known as principal components, which are linear combinations of the original variables. The first component captures the most variability in the data, the second the second, and so on. It reduces the features to a smaller group of orthogonal variables called principal components - linear combinations of the original variables. The first component captures the most variability in the data, the second the second most, and so on.

LDA : LDA finds linear discriminants in order to maximize the variance between the different categories while minimizing the variance within the class.