Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra
ID:700741613

In [248…

```
#Importing the required libraried to perform the given tasks

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from  sklearn.model_selection  import  train_test_split
from sklearn.svm import SVC
from  sklearn.preprocessing  import  StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
from  sklearn.impute  import  SimpleImputer
from sklearn import metrics
from sklearn.metrics import accuracy_score, classification_report
from  sklearn.discriminant_analysis  import  LinearDiscriminantAnalysis
import seaborn as sns
```

1.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.
```

In [249…

```
#Loading the dataset

cc_dataset=pd.read_csv('datasets/CC.csv')
cc_dataset.head()
```

Out[249]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENT
0	C10001	40.900749	0.818182	95.40	0.00	
1	C10002	3202.467416	0.909091	0.00	0.00	
2	C10003	2495.148862	1.000000	773.17	773.17	
3	C10004	1666.670542	0.636364	1499.00	1499.00	
4	C10005	817.714335	1.000000	16.00	16.00	

Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra
ID:700741613

In [250…

```
#Applying the imputer to the dataset to fill the null values that will prevent the

X = cc_dataset.iloc[:,1:]
imputer  =  SimpleImputer(missing_values=np.nan,  strategy='mean')
imputer = imputer.fit(X)
X = imputer.transform(X)
X=pd.DataFrame(X)
```

In [251…

```
#a. Apply PCA on CC dataset

pca = PCA(2)
x_pca = pca.fit_transform(X)
df2  =  pd.DataFrame(data=x_pca)
finaldf = pd.concat([df2, X.iloc[:,-1]], axis=1)
finaldf.head()
```

Out[251]:

	0	1	16
0	-4326.383956	921.566884	12.0
1	4118.916676	-2432.846347	12.0
2	1497.907660	-1997.578692	12.0
3	1394.548556	-1488.743450	12.0
4	-3743.351874	757.342659	12.0

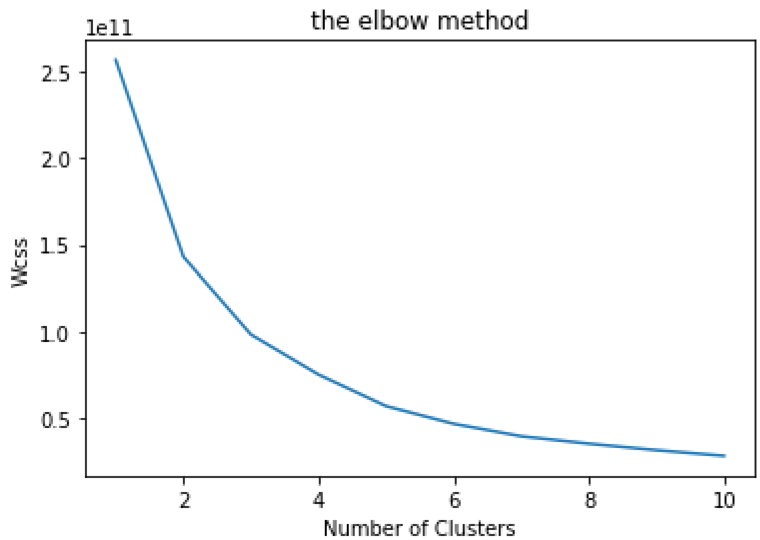
In [252…

```
#Performing the elbow method to find the best number of suitable clusters for the g
wcss = []
for i in range(1,11):
    kmeans  =  KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_sta
    kmeans.fit(finaldf)
    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()
```

Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra
ID:700741613



In [253…

```
# Apply k-means algorithm on the PCA result and report your observation if the silh

nclusters = 4
km = KMeans(n_clusters=nclusters)
km.fit(finaldf)
```

Out[253]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [254…

```
y_cluster_kmeans = km.predict(finaldf)
score = metrics.silhouette_score(finaldf, y_cluster_kmeans)
print('Silhoutte score for just PCA:',score)
```

```
Silhoutte score for just PCA: 0.504781047022562
```

In [255…

```
#Reload the dataset again
X = cc_dataset.iloc[:,1:]

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(X)

X = imputer.transform(X)

print(X)
X=pd.DataFrame(X)
```

Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra

ID:700741613

```
[[4.09007490e+01 8.18182000e-01 9.54000000e+01 ... 1.39509787e+02
  0.00000000e+00 1.20000000e+01]
 [3.20246742e+03 9.09091000e-01 0.00000000e+00 ... 1.07234022e+03
  2.22222000e-01 1.20000000e+01]
 [2.49514886e+03 1.00000000e+00 7.73170000e+02 ... 6.27284787e+02
  0.00000000e+00 1.20000000e+01]
 ...
 [2.33986730e+01 8.33333000e-01 1.44400000e+02 ... 8.24183690e+01
  2.50000000e-01 6.00000000e+00]
 [1.34575640e+01 8.33333000e-01 0.00000000e+00 ... 5.57556280e+01
  2.50000000e-01 6.00000000e+00]
 [3.72708075e+02 6.66667000e-01 1.09325000e+03 ... 8.82889560e+01
  0.00000000e+00 6.00000000e+00]]
```

In [256…

```
#Apply scaling on the dataset

scaler = StandardScaler()
scaler.fit(X)
x_scaler = scaler.transform(X)

#Apply PCA with k value as 2 again

pca = PCA(2)
x_pca  =  pca.fit_transform(x_scaler)
df2 = pd.DataFrame(data=x_pca)
finaldf  =  pd.concat([df2,cc_dataset[['TENURE']]],axis=1)
print(finaldf)
```

```
             0         1  TENURE
0    -1.682221 -1.076448      12
1    -1.138295  2.506477      12
2     0.969681 -0.383525      12
3    -0.873628  0.043166      12
4    -1.599434 -0.688578      12
...        ...       ...     ...
8945 -0.359630 -2.016142       6
8946 -0.564369 -1.639120       6
8947 -0.926204 -1.810782       6
8948 -2.336551 -0.657962       6
8949 -0.556421 -0.400473       6

[8950 rows x 3 columns]
```

In [257…

```
#Apply k-means on the scaled PCA output

nclusters = 4
km  =  KMeans(n_clusters=nclusters)
km.fit(finaldf)
```

Out[257]:

```
y_cluster_kmeans  =  km.predict(finaldf)
score  =  metrics.silhouette_score(finaldf,  y_cluster_kmeans)
print('Silhoutte score for scaled=pca=keans:',score)
```

Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra
ID:700741613

```
KMeans(algorithm='auto',copy_x=True,init='k-means++',max_iter=300, n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
random_state=None, tol=0.0001, verbose=0)
```

Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra
ID:700741613

```
Silhoutte score for scaled=pca=keans: 0.4378874170287716
```

The code loads the CC dataset and performs PCA and k-means clustering on the dataset to find the optimal number of clusters. It also applies scaling and PCA on the dataset and performs k-means clustering on the scaled PCA output to compare the results with the previous method. Finally, it prints the silhouette score for the first method and outputs the final dataset after PCA and k-means clustering.

2.Use pd_speech_features.csv

```
d. Perform Scaling
e. Apply PCA (k=3)
f. Use SVM to report performance
```

In [269…

```
#Load the dataset

speech_df=pd.read_csv('datasets/pd_speech_features.csv')
speech_df.head()
```

Out[269]:

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDev
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	

5 rows × 755 columns

In [272…

```
#Apply scaling on the dataset

x =speech_df.iloc[:,1:]
scaler = StandardScaler()
scaler.fit(x)
speech_x_scaler = scaler.transform(x)

#Apply PCA with value 3

pca = PCA(3)
speech_x_pca = pca.fit_transform(speech_x_scaler)
speech_df2 = pd.DataFrame(data=speech_x_pca)
speech_finaldf = pd.concat([speech_df2,speech_df[['class']]],axis=1)
print(speech_finaldf)
```

Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra
ID:700741613

```
              0         1          2  class
0   -10.052430  1.476819  -6.828359      1
1   -10.641066  1.590408  -6.811675      1
2   -13.520081 -1.243924  -6.794537      1
3    -9.142525  8.848870  15.300289      1
4    -6.758090  4.624220  15.645673      1
..         ...       ...        ...    ...
751  22.377449  6.470194   1.439479      0
752  13.503270  1.450496   9.344896      0
753   8.328507  2.392509  -0.911248      0
754   4.074595  5.417625  -0.847067      0
755   4.052810  6.076461  -2.022293      0

[756 rows x 4 columns]
```

In [273…

```
#Apply SVM classifier

clf = SVC(kernel='linear')
x  =speech_finaldf.iloc[:,:-1]
y =speech_finaldf.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_st
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
accuracy_score(y_test,  y_pred)
print("SVM accuracy =", accuracy_score(y_test, y_pred))
```

```
SVM accuracy = 0.768
```

In [274…

```
#Classification report for the above classifier

print(classification_report(y_test,  y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.20      0.33        69
           1       0.76      0.98      0.86       181

    accuracy                           0.77       250
   macro avg       0.79      0.59      0.59       250
weighted avg       0.78      0.77      0.71       250
```

In this code, the Parkinson's Disease dataset is loaded, and scaling and PCA are applied to reduce the dimensionality of the dataset to 3. Then, a SVM classifier is applied to classify the data, and the accuracy score and classification report are printed.

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

In [276…

```
#Load the IRIS dataset

iris_df = pd.read_csv("datasets/iris.csv")
iris_df.head()
```

Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra
ID:700741613

Out[276]:

	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 [281…

```
#apply the standard scaling

stdsc = StandardScaler()
X_train_std  =  stdsc.fit_transform(iris_df.iloc[:,:-1].values)

#Label encoding the species column
class_le = LabelEncoder()
y  =  class_le.fit_transform(iris_df['Species'].values)

#Applying LDA on the Datset

lda  =  LinearDiscriminantAnalysis(n_components=2)
X_train_lda = lda.fit_transform(X_train_std,y)

data=pd.DataFrame(X_train_lda)
data['class']=y
data.columns=["LD1","LD2","class"]
data.head()
```

Out[281]:

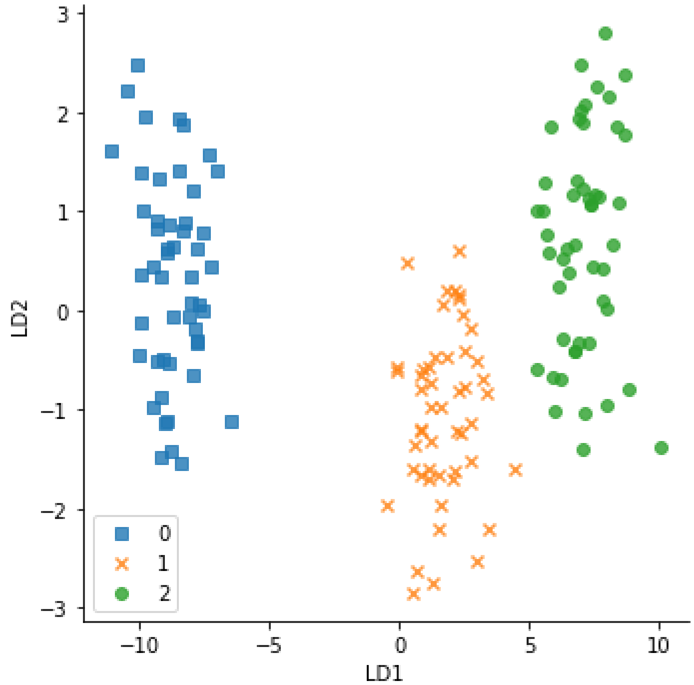
	LD1	LD2	class
0	-10.036763	-0.451330	0
1	-9.172930	-1.477234	0
2	-9.480989	-0.979693	0
3	-8.818119	-1.408602	0
4	-9.960200	-0.112546	0

In [284…

```
markers = ['s', 'x', 'o']
colors = ['y', 'b', 'g']
sns.lmplot(x="LD1", y="LD2", data=data, hue='class', markers=markers, fit_reg=False
plt.legend()
plt.show()
```

Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra
ID:700741613



In this code, the Iris dataset is loaded and standard scaling is applied to the features. Then label encoding is used to transform the target variable 'Species'. Linear Discriminant Analysis (LDA) is performed with n_components=2 and the transformed features are stored in X_train_lda. A dataframe is created with the LDA components and the target variable. Finally, a scatterplot is created with LDA components on x and y axis, and color-coded by the target variable. The plot shows the separation of the three different species in the transformed feature space.

4.Briefly identify the difference betweenPCA and LDA

PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis) are both linear transformation techniques used in machine learning and data analysis, but they have different purposes and outcomes.

PCA is primarily used for dimensionality reduction, which is the process of reducing the number of input features in a dataset while retaining most of the variance in the data. PCA finds new dimensions (or principal components) that are a linear combination of the original features, and orders them by the amount of variance they explain. PCA can be used for data visualization and to reduce the computational complexity of machine learning algorithms.

LDA, on the other hand, is a supervised learning algorithm that is primarily used for classification problems. It seeks to find a linear combination of features that maximizes the separation between classes. The goal is to create a projection that best separates the classes while retaining the maximum amount of information about the classes.

Machine Learning (Assignment # 5)

Name: Sai Priyanka Narra
ID:700741613

In summary, PCA is an unsupervised learning algorithm used for dimensionality reduction and data visualization, while LDA is a supervised learning algorithm used for classification problems.

GitHub URL:

https://github.com/SaiPriyankaNarra/ML_700741613_Assignment-5.git

Video URL:

https://drive.google.com/file/d/19PwnGlONAAvHWkQzRE9f_3a61ZPvSHIC/view?usp=share_link