Name : Garimella Suresh Babu
Student Id: 700741367
User Id :SXG13670@ucmo.edu

GitHub Link:-
https://github.com/Suresh-Garimella/Principal_Component_Analysis-Linear_Discriminent_Analysis

Video Link:-

https://github.com/Suresh-Garimella/Linear-Regression-Clustering/blob/main/ScreenRecording.mkv

1. Principal Component Analysis

a. Apply PCA on CC dataset.

Importing all the essential libraries and packages to perform PCA on the given Dataset.

Namely Packages from Sklearn - to perform PCA.
Pandas - to Maintain Dataframes.
Matplotlib - to visualize the data.

```
from sklearn.model_selection import train_test_split
import pandas as pd
# from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [86]:
```
df= pd.read_csv(r"CC.csv")

df.head()
```

Out[86]:

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

In [65]:
```
df.shape
```

Out[65]: (8950, 18)

1. Apply Linear Regression to the provided dataset using underlying steps.
a. Import the given “Salary_Data.csv”

Imports : pandas , numpy and matplotlib libraries
Using the pandas library we read the Salary_Data.csv file.

After that we are reading the CC.csv file using pandas

In [85]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import pandas as pd
# from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [86]:
```
df= pd.read_csv(r"CC.csv")

df.head()
```

Out[86]:

	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 [65]:
```
df.shape
```

Out[65]: (8950, 18)

Then we Found that the data contains null values. And we Replace the Null the null values with the mean of their column.

Then by preprocessing we deleted the CUST_ID column.

```
In [68]: mean1=df['CREDIT_LIMIT'].mean()
         mean2=df['MINIMUM_PAYMENTS'].mean()
         df['CREDIT_LIMIT'].fillna(value=mean1, inplace=True)
         df['MINIMUM_PAYMENTS'].fillna(value=mean2, inplace=True)
```

```
In [ ]:
```

```
In [69]: df.values
```

```
Out[69]: array([['C10001', 40.900749, 0.818182, ..., 139.50978700000002, 0.0, 12],
                ['C10002', 3202.467416, 0.909091, ..., 1072.340217,
                 0.22222199999999998, 12],
                ['C10003', 2495.148862, 1.0, ..., 627.284787, 0.0, 12],
                ...,
                ['C19188', 23.398673000000002, 0.833333, ..., 82.418369, 0.25, 6],
                ['C19189', 13.457564000000001, 0.833333, ..., 55.755628, 0.25, 6],
                ['C19190', 372.708075, 0.666667, ..., 88.288956, 0.0, 6]],
               dtype=object)
```

```
In [70]: df['TENURE'].value_counts()
```

```
Out[70]: 12    7584
         11     365
         10     236
         6      204
         8      196
         7      190
         9      175
         Name: TENURE, dtype: int64
```

```
In [71]: del df['CUST_ID']

         df.head()
```

Then we need to Drop the Decision Attribute “TENURE” and store the rest of columns in X.And also store the decision attribute column in Y.

n [72]:
```python
X = df.drop('TENURE',axis=1).values
print(X)
y = df['TENURE'].values
print(y)
```

```
[[4.09007490e+01 8.18182000e-01 9.54000000e+01 ... 2.01802084e+02
  1.39509787e+02 0.00000000e+00]
 [3.20246742e+03 9.09091000e-01 0.00000000e+00 ... 4.10303260e+03
  1.07234022e+03 2.22222000e-01]
 [2.49514886e+03 1.00000000e+00 7.73170000e+02 ... 6.22066742e+02
  6.27284787e+02 0.00000000e+00]
 ...
 [2.33986730e+01 8.33333000e-01 1.44400000e+02 ... 8.12707750e+01
  8.24183690e+01 2.50000000e-01]
 [1.34575640e+01 8.33333000e-01 0.00000000e+00 ... 5.25499590e+01
  5.57556280e+01 2.50000000e-01]
 [3.72708075e+02 6.66667000e-01 1.09325000e+03 ... 6.31654040e+01
  8.82889560e+01 0.00000000e+00]]
[12 12 12 ...  6  6  6]
```

Then transformed the data using PCA with 2 components that means the final dataset has only 2 columns excluding the final attribute.Then we added the Tenure attribute to the dataframe.

In [73]:
```python
%%time

pca2 = PCA(n_components=2)
principalComponents = pca2.fit_transform(X)

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

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

```
Wall time: 58.4 ms
```

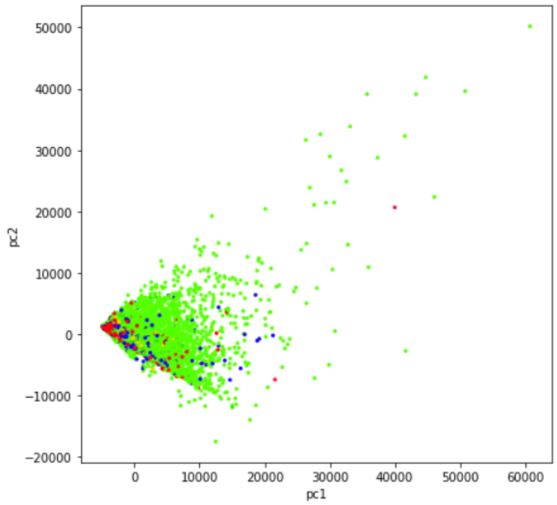
Out[73]:

	principal component 1	principal component 2	TENURE
0	-4326.383979	921.566882	12
1	4118.916665	-2432.846346	12
2	1497.907641	-1997.578694	12
3	1394.548536	-1488.743453	12
4	-3743.351896	757.342657	12

Visualization of the final dataset after PCA.

In [74]:
```python
plt.figure(figsize=(7,7))
plt.scatter(finalDf['principal component 1'],finalDf['principal component 2'],c=df['TENURE'],cmap='prism', s =5)
plt.xlabel('pc1')
plt.ylabel('pc2')
```

Out[74]: Text(0, 0.5, 'pc2')

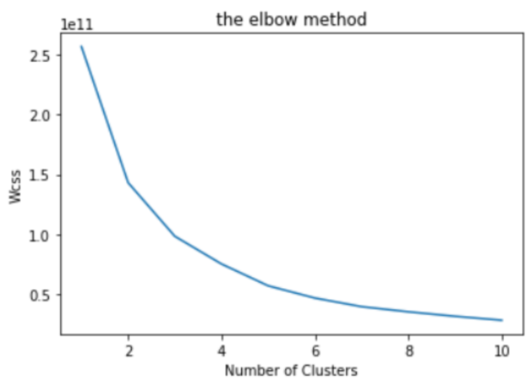


In [75]:
```python
%%time

##elbow method to know the number of clusters

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(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()
```



Wall time: 9.27 s

Scalling + PCA + K-Means

In [77]:
```
scaler = StandardScaler()
X_Scale = scaler.fit_transform(X)
```

In [78]:
```
%%time

pca2 = PCA(n_components=2)
principalComponents = pca2.fit_transform(X_Scale)

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

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

```
Wall time: 44.9 ms
```

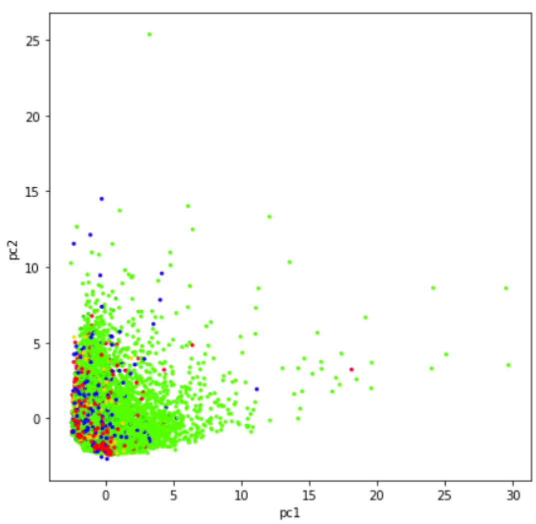
Out[78]:

	principal component 1	principal component 2	TENURE
0	-1.718890	-1.072934	12
1	-1.169302	2.509332	12
2	0.938413	-0.382604	12
3	-0.907501	0.045864	12
4	-1.637828	-0.684970	12

In [79]:
```
plt.figure(figsize=(7,7))
plt.scatter(finalDf['principal component 1'],finalDf['principal component 2'],c=df['TENURE'],cmap='prism', s =5)
plt.xlabel('pc1')

plt.ylabel('pc2')
```

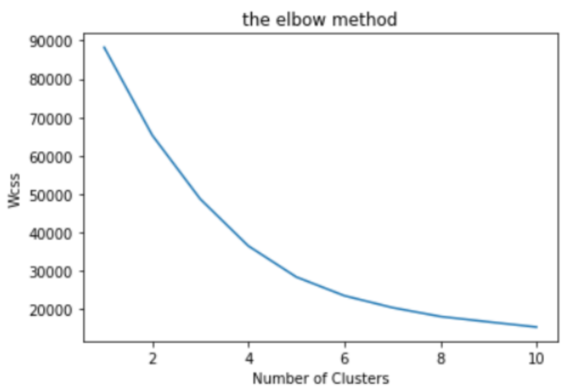
Out[79]: `Text(0, 0.5, 'pc2')`



In [80]:
```
%%time
##elbow method to know the number of clusters


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(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()
```




```
Wall time: 12.5 s
```

In [81]:
```
#  Calculate the silhouette score for the above clustering
#since the elbow point is at 3
nclusters = 3  # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(finalDf)

y_cluster_kmeans = km.predict(finalDf)
from sklearn import metrics
score = metrics.silhouette_score(finalDf, y_cluster_kmeans)
print(score)
```

0.38366768475212315

ı [43]:
```
df= pd.read_csv(r"pd_speech_features.csv")

df.head()
```

ıt[43]:

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_kurtosisValue_dec_28	tq
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...	1.5620	
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...	1.5589	
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...	1.5643	
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...	3.7805	
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...	6.1727	

5 rows × 755 columns

n [44]:
```
df.isnull().any()
```

ut[44]:
```
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
```

n [45]:
```
X = df.drop('class',axis=1).values
y = df['class'].values
```

n [46]:
```
print(X)
```

```
[[  0.         1.         0.85247 ...   2.6202     3.0004    18.9405 ]
 [  0.         1.         0.76686 ...   6.5245     6.3431    45.178  ]
 [  0.         1.         0.85083 ...   2.9199     3.1495     4.7666 ]
 ...
 [251.         0.         0.88389 ...   3.5377     3.3545     5.0424 ]
 [251.         0.         0.83782 ...   2.6801     2.8332     3.7131 ]
 [251.         0.         0.81304 ...   4.0116     2.6217     3.1527 ]]
```

In [47]:
```
scaler = StandardScaler()
X_Scale = scaler.fit_transform(X)
X_Scale
```

Out[47]:
```
array([[-1.72519117,  0.96874225,  0.62764391, ..., -0.775137  ,
        -0.81472704, -0.36659507],
       [-1.72519117,  0.96874225,  0.12161952, ..., -0.52664699,
        -0.58297219,  0.40039616],
       [-1.72519117,  0.96874225,  0.61795018, ..., -0.75606253,
        -0.8043897 , -0.7809355 ],
       ...,
       [ 1.72519117, -1.03226633,  0.81336154, ..., -0.71674252,
        -0.79017671, -0.77287314],
       [ 1.72519117, -1.03226633,  0.54105055, ..., -0.77132466,
        -0.82631929, -0.81173208],
       [ 1.72519117, -1.03226633,  0.3945807 , ..., -0.68658105,
        -0.84098293, -0.82811405]])
```

In [48]:
```
X_Scale.shape
```

Out[48]:
```
(756, 754)
```

n [49]:
```
pca3 = PCA(n_components=3)
principalComponents = pca3.fit_transform(X_Scale)

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

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

ut[49]:

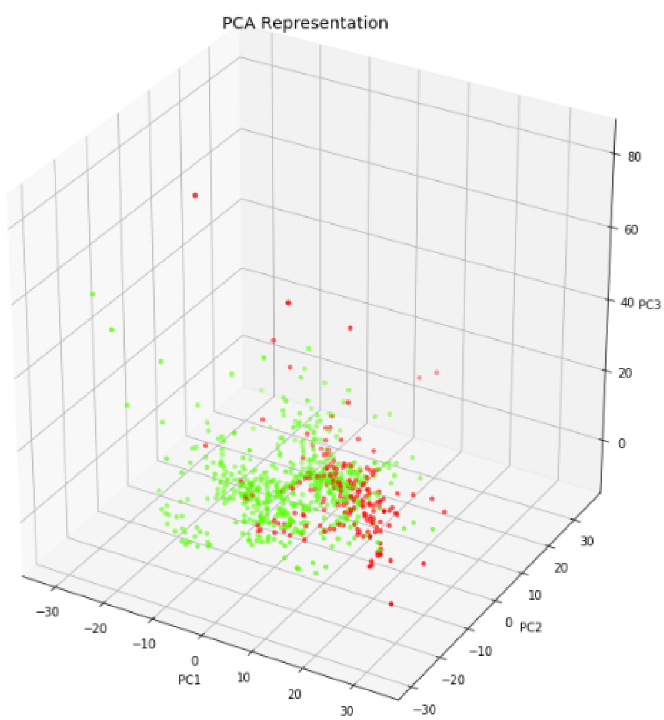
	principal component 1	principal component 2	principal component 3	class
0	-10.047372	1.471076	-6.846404	1
1	-10.637725	1.583749	-6.830977	1
2	-13.516185	-1.253542	-6.818697	1
3	-9.155083	8.833600	15.290898	1
4	-6.764470	4.611465	15.637113	1

[50]:
```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(9,9))
axes = Axes3D(fig)
axes.set_title('PCA Representation', size=14)
axes.set_xlabel('PC1')
axes.set_ylabel('PC2')
axes.set_zlabel('PC3')

axes.scatter(finalDf['principal component 1'],finalDf['principal component 2'],finalDf['principal component 3'],c=finalDf['class'
```

[50]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x24100d589c8>



Apply SVM

In [51]:
```
from sklearn.model_selection import train_test_split, cross_validate
training_set = finalDf[::-1]
labels = finalDf["class"]


X_train, X_val, Y_train, Y_val = train_test_split(training_set, labels, test_size=0.3, random_state=0)
```

In [52]:
```
X_train.head()
```

Out[52]:

	principal component 1	principal component 2	principal component 3	class
175	-1.888843	-2.410033	-6.888701	1
64	-9.681345	-2.826047	-5.915025	1
440	-4.096262	9.699011	-0.957693	1
555	-0.576323	10.768601	1.271319	1
436	9.454904	8.833950	0.601404	1

In [53]:
```
from sklearn.svm import SVC

classifier = LinearSVC()

classifier.fit(X_train, Y_train)

y_pred = classifier.predict(X_val)

from sklearn.metrics import classification_report
# Summary of the predictions made by the classifier
print(classification_report(Y_val, y_pred))

from sklearn.metrics import confusion_matrix
print(confusion_matrix(Y_val, y_pred))
# Package used to get the Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        57
           1       0.75      0.99      0.85       170

    accuracy                           0.74       227
   macro avg       0.37      0.50      0.43       227
weighted avg       0.56      0.74      0.64       227

[[  0  57]
 [  1 169]]
accuracy is 0.7444933920704846
```

n [76]:

```python
import pandas as pd
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
```

n [77]:

```python
df= pd.read_csv(r"Iris.csv")

df.head()
```

ut[77]:

	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 [79]:

```python
df.isnull().any()
```

Out[79]:

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

In [82]:
```python
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components=2)
output = lda.fit_transform(X, Y)
outputdf= pd.DataFrame(output,columns=["Linear Discriminant 1","Linear Discriminant 2"])
finalDf = pd.concat([outputdf, df["Species"]], axis = 1)
```

In [83]:
```python
finalDf.head()
```

Out[83]:

	Linear Discriminant 1	Linear Discriminant 2	Species
0	-8.084953	0.328454	Iris-setosa
1	-7.147163	-0.755473	Iris-setosa
2	-7.511378	-0.238078	Iris-setosa
3	-6.837676	-0.642885	Iris-setosa
4	-8.157814	0.540639	Iris-setosa

In [84]:
```python
finalDf.shape
```

Out[84]: (150, 3)

In [128]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [129]:
```python
# Importing the datasets

datasets = pd.read_csv('Salary_Data.csv')
```

In [130]:
```python
datasets.head()
```

Out[130]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

4. Briefly identify the difference between PCA and LDA

Both LDA and PCA rely on linear transformations and aim to maximize the variance in a lower dimension. PCA is an unsupervised learning algorithm while LDA is a supervised learning algorithm. This means that PCA finds directions of maximum variance regardless of class labels while LDA finds directions of maximum class separability.

#PCA

It reduces the features into a smaller subset of orthogonal variables, called principal components – linear combinations of the original variables. The first component

captures the largest variability of the data, while the second captures the second largest, and so on.

#LDA

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