Feature_Engineering

April 29, 2022

1 Project No. 4 -

A retail company "ABC Private Limited" wants to understand the customer purchase behaviour (specifically, purchase amount) against various products of different categories. They have shared purchase summaries of various customers for selected high volume products from last month.

[1]:
```
#Importing Libraries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
#Reading the dataset
data = pd.read_csv('ABC_Retail_Dataset.csv')
```

[3]:
```
#Checking the first 5 rows of the dataset
data.head()
```

[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969

[4]:
```
#Checking the last 5 rows of the dataset
data.tail()
```

[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
550063	1006033	P00372445	M	51-55	13	B	1	1	20	NaN	NaN	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	NaN	NaN	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	NaN	NaN	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	NaN	NaN	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	NaN	NaN	490

[5]:
```
#Finding the number of rows and columns of the dataset
data.shape
```

[5]: (550068, 12)

[6]:
```
#Checking types of variables of the data
data.dtypes
```

[6]:
```
User_ID                       int64
Product_ID                   object
Gender                       object
Age                          object
Occupation                    int64
City_Category                object
Stay_In_Current_City_Years   object
Marital_Status                int64
Product_Category_1            int64
Product_Category_2          float64
Product_Category_3          float64
Purchase                      int64
dtype: object
```

[7]:
```
#Checking out the missing values in the dataset
data.isnull().sum()
```

[7]:
```
User_ID                          0
Product_ID                       0
Gender                           0
Age                              0
Occupation                       0
City_Category                    0
Stay_In_Current_City_Years       0
Marital_Status                   0
Product_Category_1               0
Product_Category_2          173638
Product_Category_3          383247
Purchase                         0
dtype: int64
```

[8]:
```
data.isna().sum() / data.shape[0]
```

[8]:
```
User_ID                     0.000000
Product_ID                  0.000000
Gender                      0.000000
Age                         0.000000
Occupation                  0.000000
City_Category               0.000000
Stay_In_Current_City_Years  0.000000
Marital_Status              0.000000
Product_Category_1          0.000000
Product_Category_2          0.315666
Product_Category_3          0.696727
Purchase                    0.000000
dtype: float64
```

In this dataset, Product Category 1 contains 31% of missing values whereas Product Category 3 contains almost 70% of missing values.

[9]:
```
#Lets impute these missing values with median

data['Product_Category_2'].fillna(data['Product_Category_2'].median(), inplace
 ↪= True)
data['Product_Category_3'].fillna(data['Product_Category_3'].median(), inplace
 ↪= True)
```

[10]:
```
#Checking the number of missing values again
data.isna().sum()
```

[10]:
```
User_ID                     0
Product_ID                  0
Gender                      0
Age                         0
Occupation                  0
```

```
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
Purchase                      0
dtype: int64
```

[11]:
```
#Checking the number of unique categories in each columns

data.nunique()
```

[11]:
```
User_ID                       5891
Product_ID                    3631
Gender                           2
Age                              7
Occupation                      21
City_Category                    3
Stay_In_Current_City_Years       5
Marital_Status                   2
Product_Category_1              20
Product_Category_2              17
Product_Category_3              15
Purchase                     18105
dtype: int64
```

1.1 Calculating Correlation Matrix

[12]:
```
#Dropping the target variable

data = data.drop(['Purchase'], axis = 1)
```

[13]:
```
#Dropping the Categorical Variable
data = data.drop(['Product_ID'], axis = 1)
```

[14]:
```
#Encoding the Age variable using map function
data['Age'] = data['Age'].map({'0-17': 1 , '18-25': 2 , '26-35': 3 , '36-45': 4␣
 ↪, '46-50': 5 , '51-55': 6 , '55+': 6})
```

[15]:
```
#Encoding the city category variable

data['City_Category'] = data['City_Category'].map({'A': 1 , 'B': 2 , 'C': 3})
```

[16]:
```
#Encoding the Stay in city variable as it is an object datatype
```

```
data['Stay_In_Current_City_Years'] = data['Stay_In_Current_City_Years'].
↪map({'0': 0 , '1': 1 , '2': 2 , '3': 3 , '4+': 4})
```

[17]:
```
#Encoding the Gender Variable

data = pd.get_dummies(data, drop_first = True)
```

[18]:
```
data.head()
```

[18]:
```
   User_ID  Age  Occupation  City_Category  Stay_In_Current_City_Years  \
0  1000001    1          10              1                           2
1  1000001    1          10              1                           2
2  1000001    1          10              1                           2
3  1000001    1          10              1                           2
4  1000002    6          16              3                           4

   Marital_Status  Product_Category_1  Product_Category_2  Product_Category_3  \
0               0                   3                 9.0                14.0
1               0                   1                 6.0                14.0
2               0                  12                 9.0                14.0
3               0                  12                14.0                14.0
4               0                   8                 9.0                14.0

   Gender_M
0         0
1         0
2         0
3         0
4         1
```

[19]:
```
# Correlation matrix for independent variables
data.corr()
```

[19]:
```
                             User_ID       Age  Occupation  City_Category  \
User_ID                     1.000000  0.035797   -0.023971       0.022859
Age                         0.035797  1.000000    0.091237       0.116207
Occupation                 -0.023971  0.091237    1.000000       0.034479
City_Category               0.022859  0.116207    0.034479       1.000000
Stay_In_Current_City_Years -0.030737 -0.006269    0.030005       0.019946
Marital_Status              0.020443  0.319946    0.024280       0.039790
Product_Category_1          0.003825  0.060368   -0.007618      -0.014364
Product_Category_2          0.001644  0.043811    0.000557      -0.006888
Product_Category_3          0.001008  0.035566    0.004325      -0.011300
Gender_M                   -0.033474 -0.005322    0.117291      -0.004515

                            Stay_In_Current_City_Years  Marital_Status  \
User_ID                                      -0.030737        0.020443
```

```
Age                                  -0.006269        0.319946
Occupation                            0.030005        0.024280
City_Category                         0.019946        0.039790
Stay_In_Current_City_Years            1.000000       -0.012819
Marital_Status                       -0.012819        1.000000
Product_Category_1                   -0.004213        0.019888
Product_Category_2                   -0.001087        0.011526
Product_Category_3                    0.000673        0.012705
Gender_M                              0.014660       -0.011603

                           Product_Category_1  Product_Category_2  \
User_ID                              0.003825            0.001644
Age                                  0.060368            0.043811
Occupation                          -0.007618            0.000557
City_Category                       -0.014364           -0.006888
Stay_In_Current_City_Years          -0.004213           -0.001087
Marital_Status                       0.019888            0.011526
Product_Category_1                   1.000000            0.331691
Product_Category_2                   0.331691            1.000000
Product_Category_3                   0.195930            0.416680
Gender_M                            -0.045594           -0.014051

                           Product_Category_3  Gender_M
User_ID                              0.001008 -0.033474
Age                                  0.035566 -0.005322
Occupation                           0.004325  0.117291
City_Category                       -0.011300 -0.004515
Stay_In_Current_City_Years           0.000673  0.014660
Marital_Status                       0.012705 -0.011603
Product_Category_1                   0.195930 -0.045594
Product_Category_2                   0.416680 -0.014051
Product_Category_3                   1.000000  0.005887
Gender_M                             0.005887  1.000000
```

[20]:
```
#Finding the absolute values and only the upper diagonal of the correlation␣
↪matrix

matrix = data.corr().abs()
upper_diagonal = matrix.where(np.triu(np.ones(matrix.shape), k=1).astype(np.
↪bool))
upper_diagonal
```

[20]:
```
                           User_ID       Age  Occupation  City_Category  \
User_ID                        NaN  0.035797    0.023971       0.022859
Age                            NaN       NaN    0.091237       0.116207
Occupation                     NaN       NaN         NaN       0.034479
City_Category                  NaN       NaN         NaN            NaN
```

Stay_In_Current_City_Years	NaN	NaN	NaN	NaN
Marital_Status	NaN	NaN	NaN	NaN
Product_Category_1	NaN	NaN	NaN	NaN
Product_Category_2	NaN	NaN	NaN	NaN
Product_Category_3	NaN	NaN	NaN	NaN
Gender_M	NaN	NaN	NaN	NaN

	Stay_In_Current_City_Years	Marital_Status	\
User_ID	0.030737	0.020443	
Age	0.006269	0.319946	
Occupation	0.030005	0.024280	
City_Category	0.019946	0.039790	
Stay_In_Current_City_Years	NaN	0.012819	
Marital_Status	NaN	NaN	
Product_Category_1	NaN	NaN	
Product_Category_2	NaN	NaN	
Product_Category_3	NaN	NaN	
Gender_M	NaN	NaN	

	Product_Category_1	Product_Category_2	\
User_ID	0.003825	0.001644	
Age	0.060368	0.043811	
Occupation	0.007618	0.000557	
City_Category	0.014364	0.006888	
Stay_In_Current_City_Years	0.004213	0.001087	
Marital_Status	0.019888	0.011526	
Product_Category_1	NaN	0.331691	
Product_Category_2	NaN	NaN	
Product_Category_3	NaN	NaN	
Gender_M	NaN	NaN	

	Product_Category_3	Gender_M
User_ID	0.001008	0.033474
Age	0.035566	0.005322
Occupation	0.004325	0.117291
City_Category	0.011300	0.004515
Stay_In_Current_City_Years	0.000673	0.014660
Marital_Status	0.012705	0.011603
Product_Category_1	0.195930	0.045594
Product_Category_2	0.416680	0.014051
Product_Category_3	NaN	0.005887
Gender_M	NaN	NaN

Looking at the above values, we can say that Product_Category_2, Product_Category_3 are highly correlated with correlation value 0.416680 & Product_Category_1, Product_Category_2 are highly correlated with correlation value 0.331691.

1.2 Covariance Matrix

[21]:
```
data.head()
```

[21]:
```
   User_ID  Age  Occupation  City_Category  Stay_In_Current_City_Years  \
0  1000001    1          10              1                           2
1  1000001    1          10              1                           2
2  1000001    1          10              1                           2
3  1000001    1          10              1                           2
4  1000002    6          16              3                           4

   Marital_Status  Product_Category_1  Product_Category_2  Product_Category_3  \
0               0                   3                 9.0                14.0
1               0                   1                 6.0                14.0
2               0                  12                 9.0                14.0
3               0                  12                14.0                14.0
4               0                   8                 9.0                14.0

   Gender_M
0         0
1         0
2         0
3         0
4         1
```

[22]:
```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
```

[23]:
```
data_scale = scale.fit_transform(data)
```

[25]:
```
data_scale = pd.DataFrame(data_scale, columns = data.columns)
```

[26]:
```
data_scale.head()
```

[26]:
```
    User_ID       Age  Occupation  City_Category  Stay_In_Current_City_Years  \
0 -1.752639 -1.945160    0.294864      -1.371516                    0.109801
1 -1.752639 -1.945160    0.294864      -1.371516                    0.109801
2 -1.752639 -1.945160    0.294864      -1.371516                    0.109801
3 -1.752639 -1.945160    0.294864      -1.371516                    0.109801
4 -1.752061  2.012703    1.214734       1.259336                    1.660861

   Marital_Status  Product_Category_1  Product_Category_2  Product_Category_3  \
0       -0.833018           -0.610809           -0.136401            0.171658
1       -0.833018           -1.118912           -0.846289            0.171658
2       -0.833018            1.675656           -0.136401            0.171658
3       -0.833018            1.675656            1.046745            0.171658
4       -0.833018            0.659449           -0.136401            0.171658
```

```
   Gender_M
0 -1.746513
1 -1.746513
2 -1.746513
3 -1.746513
4  0.572570
```

[27]:
```
data_scale.cov()
```

[27]:
```
                             User_ID       Age  Occupation  City_Category  \
User_ID                     1.000002  0.035797   -0.023971       0.022859
Age                         0.035797  1.000002    0.091237       0.116207
Occupation                 -0.023971  0.091237    1.000002       0.034479
City_Category               0.022859  0.116207    0.034479       1.000002
Stay_In_Current_City_Years -0.030737 -0.006269    0.030005       0.019946
Marital_Status              0.020443  0.319947    0.024280       0.039791
Product_Category_1          0.003825  0.060368   -0.007618      -0.014364
Product_Category_2          0.001644  0.043811    0.000557      -0.006888
Product_Category_3          0.001008  0.035566    0.004325      -0.011300
Gender_M                   -0.033475 -0.005322    0.117291      -0.004515

                            Stay_In_Current_City_Years  Marital_Status  \
User_ID                                      -0.030737        0.020443
Age                                          -0.006269        0.319947
Occupation                                    0.030005        0.024280
City_Category                                 0.019946        0.039791
Stay_In_Current_City_Years                    1.000002       -0.012819
Marital_Status                               -0.012819        1.000002
Product_Category_1                           -0.004213        0.019888
Product_Category_2                           -0.001087        0.011526
Product_Category_3                            0.000673        0.012705
Gender_M                                      0.014660       -0.011603

                            Product_Category_1  Product_Category_2  \
User_ID                               0.003825            0.001644
Age                                   0.060368            0.043811
Occupation                           -0.007618            0.000557
City_Category                        -0.014364           -0.006888
Stay_In_Current_City_Years           -0.004213           -0.001087
Marital_Status                        0.019888            0.011526
Product_Category_1                    1.000002            0.331692
Product_Category_2                    0.331692            1.000002
Product_Category_3                    0.195930            0.416680
Gender_M                             -0.045594           -0.014051

                            Product_Category_3  Gender_M
User_ID                               0.001008 -0.033475
```

```
Age                                0.035566 -0.005322
Occupation                         0.004325  0.117291
City_Category                     -0.011300 -0.004515
Stay_In_Current_City_Years         0.000673  0.014660
Marital_Status                     0.012705 -0.011603
Product_Category_1                 0.195930 -0.045594
Product_Category_2                 0.416680 -0.014051
Product_Category_3                 1.000002  0.005887
Gender_M                           0.005887  1.000002
```

[29]:
```
# Checking the upper diagonal of the covariance matrix

covar_matrix = data_scale.cov()
upper_dia = covar_matrix.where(np.triu(np.ones(covar_matrix.shape), k=1).
 ↪astype(np.bool))
upper_dia
```

[29]:
```
                           User_ID       Age  Occupation  City_Category  \
User_ID                        NaN  0.035797   -0.023971       0.022859
Age                            NaN       NaN    0.091237       0.116207
Occupation                     NaN       NaN         NaN       0.034479
City_Category                  NaN       NaN         NaN            NaN
Stay_In_Current_City_Years     NaN       NaN         NaN            NaN
Marital_Status                 NaN       NaN         NaN            NaN
Product_Category_1             NaN       NaN         NaN            NaN
Product_Category_2             NaN       NaN         NaN            NaN
Product_Category_3             NaN       NaN         NaN            NaN
Gender_M                       NaN       NaN         NaN            NaN

                           Stay_In_Current_City_Years  Marital_Status  \
User_ID                                    -0.030737        0.020443
Age                                        -0.006269        0.319947
Occupation                                  0.030005        0.024280
City_Category                               0.019946        0.039791
Stay_In_Current_City_Years                       NaN       -0.012819
Marital_Status                                   NaN            NaN
Product_Category_1                               NaN            NaN
Product_Category_2                               NaN            NaN
Product_Category_3                               NaN            NaN
Gender_M                                         NaN            NaN

                           Product_Category_1  Product_Category_2  \
User_ID                              0.003825            0.001644
Age                                  0.060368            0.043811
Occupation                          -0.007618            0.000557
City_Category                       -0.014364           -0.006888
Stay_In_Current_City_Years          -0.004213           -0.001087
```

```
Marital_Status                        0.019888           0.011526
Product_Category_1                         NaN           0.331692
Product_Category_2                         NaN                NaN
Product_Category_3                         NaN                NaN
Gender_M                                   NaN                NaN

                            Product_Category_3  Gender_M
User_ID                           0.001008 -0.033475
Age                               0.035566 -0.005322
Occupation                        0.004325  0.117291
City_Category                    -0.011300 -0.004515
Stay_In_Current_City_Years        0.000673  0.014660
Marital_Status                    0.012705 -0.011603
Product_Category_1                0.195930 -0.045594
Product_Category_2                0.416680 -0.014051
Product_Category_3                     NaN  0.005887
Gender_M                               NaN       NaN
```

1.3 Eigen Values and Eigen Vectors

[30]:
```
# Calculating Eigen values and Eigen vectors from the Covariance Matrix
from numpy.linalg import eig

eigen_value, eigen_vector = eig(covar_matrix)
```

[31]:
```
print('Eigen values are - ', eigen_value)
print('\nEigen vectors are - ', eigen_vector)
```

```
Eigen values are -  [1.65781065 1.36665469 0.55109579 0.66117784 1.14119194
1.00708257
 0.99079999 0.94462233 0.8015423  0.87804008]

Eigen vectors are -  [[ 2.21770836e-02  9.40454735e-02  4.56305063e-04
-3.34499983e-02
  -3.48896413e-01  2.05126051e-01  7.37161412e-01 -5.31110652e-01
  -1.59214206e-02 -3.84130118e-04]
 [ 1.89037969e-01  6.48489573e-01  2.01069899e-02  7.24007447e-01
  -5.89809927e-02  1.01609990e-02 -1.14413496e-01 -3.80337483e-02
  -3.18623793e-02 -7.85324930e-04]
 [ 2.11612910e-02  2.40545267e-01 -3.50836412e-03 -1.44634674e-01
   5.92750549e-01  1.55383202e-01  1.73691045e-01 -3.89551497e-02
   1.25077388e-01 -7.05642323e-01]
 [ 1.63375158e-02  3.09519204e-01 -1.76025398e-02 -1.58559967e-01
   1.26794389e-02 -4.12668583e-01  5.30519449e-01  6.35167439e-01
  -3.94465561e-02  1.46898315e-01]
 [-8.61568068e-03 -3.58255102e-04  9.95806322e-04  5.91959854e-03
   2.87687401e-01 -7.95950522e-01 -2.01953103e-02 -5.24907873e-01
  -1.26808628e-02  8.65528955e-02]
```

```
[ 1.32099663e-01  6.04805549e-01 -1.97149048e-03 -6.46933409e-01
 -1.51849046e-01  7.72195896e-02 -3.25342293e-01 -1.70713932e-01
  6.52504030e-02  1.73100369e-01]
[ 5.01771561e-01 -8.95889868e-02 -3.15892008e-01 -8.51501607e-02
 -5.28510324e-02 -3.66273079e-02 -3.54196342e-02  4.75938350e-03
 -7.68169947e-01 -1.94071488e-01]
[ 6.19982395e-01 -1.47467019e-01  7.59041457e-01 -3.39766580e-02
  5.15975070e-02 -2.47218596e-03  4.39340222e-02  2.57250364e-02
  9.81201846e-02  4.10974466e-02]
[ 5.54430735e-01 -1.37230563e-01 -5.68627089e-01  3.55802829e-02
  9.29177141e-02  2.77623757e-02  5.84812315e-02  1.01668818e-02
  5.56756741e-01  1.62066868e-01]
[-4.45785562e-02  5.05741411e-02  5.93947843e-05  2.01262339e-02
  6.35111916e-01  3.48944462e-01  1.37923451e-01 -7.78486559e-02
 -2.59657057e-01  6.14341500e-01]]
```

1.3.1 Selecting Principal Components

[32]:
```
#Arranging the magnitude of eigen values in decreasing order

sort_index = np.argsort(eigen_value)[::-1]
sort_eigen_value = eigen_value[sort_index]
sort_eigen_value
```

[32]:
```
array([1.65781065, 1.36665469, 1.14119194, 1.00708257, 0.99079999,
       0.94462233, 0.87804008, 0.8015423 , 0.66117784, 0.55109579])
```

[33]:
```
#Arranging the eigen vectors

sort_eigen_vector = eigen_vector[:, sort_index]
sort_eigen_vector
```

[33]:
```
array([[ 2.21770836e-02,  9.40454735e-02, -3.48896413e-01,
         2.05126051e-01,  7.37161412e-01, -5.31110652e-01,
        -3.84130118e-04, -1.59214206e-02, -3.34499983e-02,
         4.56305063e-04],
       [ 1.89037969e-01,  6.48489573e-01, -5.89809927e-02,
         1.01609990e-02, -1.14413496e-01, -3.80337483e-02,
        -7.85324930e-04, -3.18623793e-02,  7.24007447e-01,
         2.01069899e-02],
       [ 2.11612910e-02,  2.40545267e-01,  5.92750549e-01,
         1.55383202e-01,  1.73691045e-01, -3.89551497e-02,
        -7.05642323e-01,  1.25077388e-01, -1.44634674e-01,
        -3.50836412e-03],
       [ 1.63375158e-02,  3.09519204e-01,  1.26794389e-02,
        -4.12668583e-01,  5.30519449e-01,  6.35167439e-01,
         1.46898315e-01, -3.94465561e-02, -1.58559967e-01,
        -1.76025398e-02],
```

```
[-8.61568068e-03, -3.58255102e-04,  2.87687401e-01,
 -7.95950522e-01, -2.01953103e-02, -5.24907873e-01,
  8.65528955e-02, -1.26808628e-02,  5.91959854e-03,
  9.95806322e-04],
 [ 1.32099663e-01,  6.04805549e-01, -1.51849046e-01,
  7.72195896e-02, -3.25342293e-01, -1.70713932e-01,
  1.73100369e-01,  6.52504030e-02, -6.46933409e-01,
 -1.97149048e-03],
 [ 5.01771561e-01, -8.95889868e-02, -5.28510324e-02,
 -3.66273079e-02, -3.54196342e-02,  4.75938350e-03,
 -1.94071488e-01, -7.68169947e-01, -8.51501607e-02,
 -3.15892008e-01],
 [ 6.19982395e-01, -1.47467019e-01,  5.15975070e-02,
 -2.47218596e-03,  4.39340222e-02,  2.57250364e-02,
  4.10974466e-02,  9.81201846e-02, -3.39766580e-02,
  7.59041457e-01],
 [ 5.54430735e-01, -1.37230563e-01,  9.29177141e-02,
  2.77623757e-02,  5.84812315e-02,  1.01668818e-02,
  1.62066868e-01,  5.56756741e-01,  3.55802829e-02,
 -5.68627089e-01],
 [-4.45785562e-02,  5.05741411e-02,  6.35111916e-01,
  3.48944462e-01,  1.37923451e-01, -7.78486559e-02,
  6.14341500e-01, -2.59657057e-01,  2.01262339e-02,
  5.93947843e-05]])
```

[34]:
```
#Calculating the percentage of variance of each eigen vector

for j in sort_eigen_value:
    print(j / sum(sort_eigen_value))
```

```
0.1657807631992945
0.13666522090954863
0.11411898693986047
0.1007080742866183
0.09907981888390606
0.09446206119320333
0.08780384796459795
0.08015408412234185
0.06611766338054524
0.05510947912008357
```

[35]:
```
#Selecting the most 2 important Principal Components
k_components = 2
eigen_vec_subset = sort_eigen_vector[:,0:k_components]
eigen_vec_subset
```

[35]:
```
array([[ 2.21770836e-02,  9.40454735e-02],
       [ 1.89037969e-01,  6.48489573e-01],
       [ 2.11612910e-02,  2.40545267e-01],
       [ 1.63375158e-02,  3.09519204e-01],
       [-8.61568068e-03, -3.58255102e-04],
       [ 1.32099663e-01,  6.04805549e-01],
       [ 5.01771561e-01, -8.95889868e-02],
       [ 6.19982395e-01, -1.47467019e-01],
       [ 5.54430735e-01, -1.37230563e-01],
       [-4.45785562e-02,  5.05741411e-02]])
```

1.3.2 Selecting 8 features using Principal Component Analysis

[36]:
```
#Selecting 8 principal components

k_components = 8
eigen_vec_subset = sort_eigen_vector[:,0:k_components]
eigen_vec_subset
```

[36]:
```
array([[ 2.21770836e-02,  9.40454735e-02, -3.48896413e-01,
         2.05126051e-01,  7.37161412e-01, -5.31110652e-01,
        -3.84130118e-04, -1.59214206e-02],
       [ 1.89037969e-01,  6.48489573e-01, -5.89809927e-02,
         1.01609990e-02, -1.14413496e-01, -3.80337483e-02,
        -7.85324930e-04, -3.18623793e-02],
       [ 2.11612910e-02,  2.40545267e-01,  5.92750549e-01,
         1.55383202e-01,  1.73691045e-01, -3.89551497e-02,
        -7.05642323e-01,  1.25077388e-01],
       [ 1.63375158e-02,  3.09519204e-01,  1.26794389e-02,
        -4.12668583e-01,  5.30519449e-01,  6.35167439e-01,
         1.46898315e-01, -3.94465561e-02],
       [-8.61568068e-03, -3.58255102e-04,  2.87687401e-01,
        -7.95950522e-01, -2.01953103e-02, -5.24907873e-01,
         8.65528955e-02, -1.26808628e-02],
       [ 1.32099663e-01,  6.04805549e-01, -1.51849046e-01,
         7.72195896e-02, -3.25342293e-01, -1.70713932e-01,
         1.73100369e-01,  6.52504030e-02],
       [ 5.01771561e-01, -8.95889868e-02, -5.28510324e-02,
        -3.66273079e-02, -3.54196342e-02,  4.75938350e-03,
        -1.94071488e-01, -7.68169947e-01],
       [ 6.19982395e-01, -1.47467019e-01,  5.15975070e-02,
        -2.47218596e-03,  4.39340222e-02,  2.57250364e-02,
         4.10974466e-02,  9.81201846e-02],
       [ 5.54430735e-01, -1.37230563e-01,  9.29177141e-02,
         2.77623757e-02,  5.84812315e-02,  1.01668818e-02,
         1.62066868e-01,  5.56756741e-01],
       [-4.45785562e-02,  5.05741411e-02,  6.35111916e-01,
```

```
       3.48944462e-01,  1.37923451e-01, -7.78486559e-02,
       6.14341500e-01, -2.59657057e-01]])
```

[ ]: