Assignment 3 - ML - Navya Renduchintala - 2306AML112

Using Credit Card Dataset develop a customer segmentation using KMeans to define marketing strategy.

The Dataset summarizes the usage behavior of about 9000 active credit card holders during the last 6 months.

The file is at a customer level with 18 behavioral variables.

Following is the Data Dictionary for Credit Card dataset :-

CUST_ID : Identification of Credit Card holder (Categorical) BALANCE : Balance amount left in their account to make purchases (BALANCE_FREQUENCY : How frequently the Balance is updated, score between 0 and 1 (1 = frequently updated, 0 = not frequently updated) PURCHASES : Amount of purchases made from account ONEOFF_PURCHASES : Maximum purchase amount done in one-go INSTALLMENTS_PURCHASES : Amount of purchase done in installment CASH_ADVANCE : Cash in advance given by the user PURCHASES_FREQUENCY : How frequently the Purchases are being made, score between 0 and 1 (1 = frequently purchased, 0 = not frequently purchased) ONEOFFPURCHASESFREQUENCY : How frequently Purchases are happening in one-go (1 = frequently purchased, 0 = not frequently purchased) PURCHASESINSTALLMENTSFREQUENCY : How frequently purchases in installments are being done (1 = frequently done, 0 = not frequently done) CASHADVANCEFREQUENCY : How frequently the cash in advance being paid CASHADVANCETRX : Number of Transactions made with "Cash in Advanced" PURCHASES_TRX : Numbe of purchase transactions made CREDIT_LIMIT : Limit of Credit Card for user PAYMENTS : Amount of Payment done by user MINIMUM_PAYMENTS : Minimum amount of payments made by user PRCFULLPAYMENT : Percent of full payment paid by user TENURE : Tenure of credit card service for user

In [1]:
```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv('CC_GENERAL.csv')
data
```

Out[2]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	F
0	C10001	40.900749	0.818182	95.40	0.00	95.40	0.000000	
1	C10002	3202.467416	0.909091	0.00	0.00	0.00	6442.945483	
2	C10003	2495.148862	1.000000	773.17	773.17	0.00	0.000000	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.00	205.788017	
4	C10005	817.714335	1.000000	16.00	16.00	0.00	0.000000	
...	...	...	...	...	...	...	...	...
8945	C19186	28.493517	1.000000	291.12	0.00	291.12	0.000000	
8946	C19187	19.183215	1.000000	300.00	0.00	300.00	0.000000	
8947	C19188	23.398673	0.833333	144.40	0.00	144.40	0.000000	
8948	C19189	13.457564	0.833333	0.00	0.00	0.00	36.558778	
8949	C19190	372.708075	0.666667	1093.25	1093.25	0.00	127.040008	

8950 rows × 18 columns

In [3]:
```
print(data.isnull().sum())
```

```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

In [4]:
```
data['CREDIT_LIMIT'].fillna(data['CREDIT_LIMIT'].mean(),inplace=True)
data['MINIMUM_PAYMENTS'].fillna(data['MINIMUM_PAYMENTS'].mean(),inplace=True)
print(data)
```

```
     CUST_ID      BALANCE  BALANCE_FREQUENCY  PURCHASES  ONEOFF_PURCHASES  \
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
...      ...          ...                ...        ...               ...
8945  C19186    28.493517           1.000000     291.12              0.00
8946  C19187    19.183215           1.000000     300.00              0.00
8947  C19188    23.398673           0.833333     144.40              0.00
8948  C19189    13.457564           0.833333       0.00              0.00
8949  C19190   372.708075           0.666667    1093.25           1093.25

      INSTALLMENTS_PURCHASES  CASH_ADVANCE  PURCHASES_FREQUENCY  \
0                      95.40      0.000000             0.166667
1                       0.00   6442.945483             0.000000
2                       0.00      0.000000             1.000000
3                       0.00    205.788017             0.083333
4                       0.00      0.000000             0.083333
...                      ...           ...                  ...
8945                  291.12      0.000000             1.000000
8946                  300.00      0.000000             1.000000
8947                  144.40      0.000000             0.833333
8948                    0.00     36.558778             0.000000
8949                    0.00    127.040008             0.666667

      ONEOFF_PURCHASES_FREQUENCY  PURCHASES_INSTALLMENTS_FREQUENCY  \
0                       0.000000                          0.083333
1                       0.000000                          0.000000
2                       1.000000                          0.000000
3                       0.083333                          0.000000
4                       0.083333                          0.000000
...                          ...                               ...
8945                    0.000000                          0.833333
8946                    0.000000                          0.833333
8947                    0.000000                          0.666667
8948                    0.000000                          0.000000
8949                    0.666667                          0.000000

      CASH_ADVANCE_FREQUENCY  CASH_ADVANCE_TRX  PURCHASES_TRX  CREDIT_LIMIT  \
0                   0.000000                 0              2        1000.0
1                   0.250000                 4              0        7000.0
2                   0.000000                 0             12        7500.0
3                   0.083333                 1              1        7500.0
4                   0.000000                 0              1        1200.0
...                      ...               ...            ...           ...
8945                0.000000                 0              6        1000.0
8946                0.000000                 0              6        1000.0
8947                0.000000                 0              5        1000.0
8948                0.166667                 2              0         500.0
8949                0.333333                 2             23        1200.0

         PAYMENTS  MINIMUM_PAYMENTS  PRC_FULL_PAYMENT  TENURE
0      201.802084        139.509787          0.000000      12
1     4103.032597       1072.340217          0.222222      12
2      622.066742        627.284787          0.000000      12
3        0.000000        864.206542          0.000000      12
4      678.334763        244.791237          0.000000      12
...           ...               ...               ...     ...
8945   325.594462         48.886365          0.500000       6
8946   275.861322        864.206542          0.000000       6
8947    81.270775         82.418369          0.250000       6
8948    52.549959         55.755628          0.250000       6
8949    63.165404         88.288956          0.000000       6

[8950 rows x 18 columns]
```

In [5]:
```
print(data.isnull().sum())
```

```
CUST_ID                           0
BALANCE                           0
BALANCE_FREQUENCY                 0
PURCHASES                         0
ONEOFF_PURCHASES                  0
INSTALLMENTS_PURCHASES            0
CASH_ADVANCE                      0
PURCHASES_FREQUENCY               0
ONEOFF_PURCHASES_FREQUENCY        0
PURCHASES_INSTALLMENTS_FREQUENCY  0
CASH_ADVANCE_FREQUENCY            0
CASH_ADVANCE_TRX                  0
PURCHASES_TRX                     0
CREDIT_LIMIT                      0
PAYMENTS                          0
MINIMUM_PAYMENTS                  0
PRC_FULL_PAYMENT                  0
TENURE                            0
dtype: int64
```

In [6]:
```
data.dtypes
```

Out[6]:
```
CUST_ID                            object
BALANCE                           float64
BALANCE_FREQUENCY                 float64
PURCHASES                         float64
ONEOFF_PURCHASES                  float64
INSTALLMENTS_PURCHASES            float64
CASH_ADVANCE                      float64
PURCHASES_FREQUENCY               float64
ONEOFF_PURCHASES_FREQUENCY        float64
PURCHASES_INSTALLMENTS_FREQUENCY  float64
CASH_ADVANCE_FREQUENCY            float64
CASH_ADVANCE_TRX                    int64
PURCHASES_TRX                       int64
CREDIT_LIMIT                      float64
PAYMENTS                          float64
MINIMUM_PAYMENTS                  float64
PRC_FULL_PAYMENT                  float64
TENURE                              int64
dtype: object
```

In [7]:
```
categorical_features = ['CUST_ID']
```

In [8]:
```
for col in categorical_features:
    dummies = pd.get_dummies(data[col], prefix=col)
    data = pd.concat([data, dummies], axis=1)
    data.drop(col, axis=1, inplace=True)
data.head()
```

Out[8]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_F
0	40.900749	0.818182	95.40	0.00	95.4	0.000000	
1	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	
2	2495.148862	1.000000	773.17	773.17	0.0	0.000000	
3	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	
4	817.714335	1.000000	16.00	16.00	0.0	0.000000	

5 rows × 8967 columns

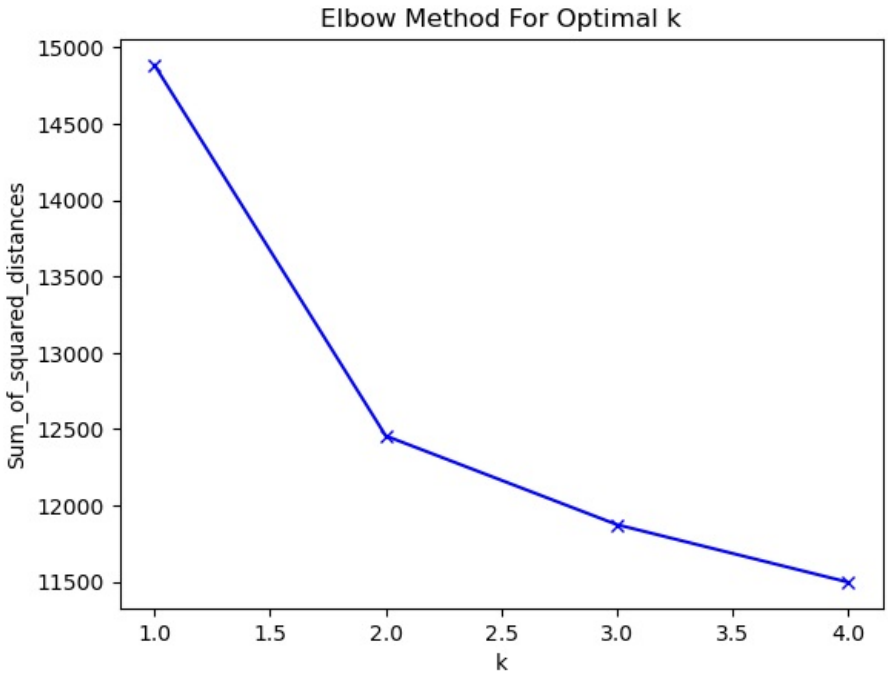
In [9]:
```
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
mms.fit(data)
data_transformed = mms.transform(data)
data_transformed
```

Out[9]:
```
array([[2.14779454e-03, 8.18182000e-01, 1.94536779e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.68169097e-01, 9.09091000e-01, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.31026136e-01, 1.00000000e+00, 1.57662475e-02, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [1.22871936e-03, 8.33333000e-01, 2.94456089e-03, ...,
        1.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [7.06688341e-04, 8.33333000e-01, 0.00000000e+00, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
       [1.95717777e-02, 6.66667000e-01, 2.22932216e-02, ...,
        0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
```

In [10]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [11]:
```
Sum_of_squared_distances = []
K = range(1,5)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(data_transformed)
    Sum_of_squared_distances.append(km.inertia_)
```

In [12]:
```
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



In [13]:
```
kmeans = KMeans(n_clusters =4, init='k-means++')
kmeans.fit(data_transformed)
pred = kmeans.predict(data_transformed)
pred
```

Out[13]:
```
array([0, 0, 2, ..., 1, 0, 2])
```

In [14]:
```
frame = pd.DataFrame(data_transformed)
frame['cluster'] = pred
frame['cluster'].value_counts()
```

Out[14]:
```
0    3649
1    2687
2    1402
3    1212
Name: cluster, dtype: int64
```