bank-cluster

September 15, 2024

[ ]:
```
# this data is to develop a customer segmentation to define marketing strategy .
# the dataset summarizes the customer about 9000 active credit card holder␣
↪during the lasr 6 months,
# it contain 18 behavioural variables
```

[ ]:
```
# 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
```

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

[ ]:
```
df=pd.read_csv(r'C:\Users\USER\Documents\CC GENERAL.csv')
```

1 Data exporation and cleaning

[48]:
```
df.head()
```

[48]:
```
       BALANCE  BALANCE_FREQUENCY  PURCHASES  ONEOFF_PURCHASES  \
0    40.900749           0.818182      95.40              0.00
1  3202.467416           0.909091       0.00              0.00
2  2495.148862           1.000000     773.17            773.17
4   817.714335           1.000000      16.00             16.00
5  1809.828751           1.000000    1333.28              0.00

   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
4                    0.00      0.000000             0.083333
5                 1333.28      0.000000             0.666667

   ONEOFF_PURCHASES_FREQUENCY  PURCHASES_INSTALLMENTS_FREQUENCY  \
0                    0.000000                          0.083333
1                    0.000000                          0.000000
2                    1.000000                          0.000000
4                    0.083333                          0.000000
5                    0.000000                          0.583333

   CASH_ADVANCE_FREQUENCY  CASH_ADVANCE_TRX  PURCHASES_TRX  CREDIT_LIMIT  \
0                    0.00                 0              2        1000.0
1                    0.25                 4              0        7000.0
2                    0.00                 0             12        7500.0
4                    0.00                 0              1        1200.0
5                    0.00                 0              8        1800.0

      PAYMENTS  MINIMUM_PAYMENTS  PRC_FULL_PAYMENT  TENURE  cluster
0   201.802084        139.509787          0.000000      12        3
1  4103.032597       1072.340217          0.222222      12        0
2   622.066742        627.284787          0.000000      12        1
4   678.334763        244.791237          0.000000      12        3
5  1400.057770       2407.246035          0.000000      12        2
```

[49]:
```
df.isnull().sum() # checking for missing values
```

[49]:
```
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
cluster                             0
dtype: int64
```

[ ]:
```
df.drop(['CUST_ID'],axis=1,inplace=True)
```

[ ]:
```
df.info()
```

[ ]:
```
dfr=df.dropna(inplace =True) # handling missing values
```

[50]:
```
df.isnull().sum() #reconfirming missing value well handled
```

[50]:
```
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
cluster                             0
dtype: int64
```

[52]:
```
df.duplicated().sum() # checking for duplicates
```

[52]:
```
0
```

[51]:
```
df.shape
```

[51]: (8636, 18)

[53]:
```
df.columns
```

[53]:
```
Index(['BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES', 'ONEOFF_PURCHASES',
       'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE', 'PURCHASES_FREQUENCY',
       'ONEOFF_PURCHASES_FREQUENCY', 'PURCHASES_INSTALLMENTS_FREQUENCY',
       'CASH_ADVANCE_FREQUENCY', 'CASH_ADVANCE_TRX', 'PURCHASES_TRX',
       'CREDIT_LIMIT', 'PAYMENTS', 'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT',
       'TENURE', 'cluster'],
      dtype='object')
```

[ ]:
```
# statistical analysis
```

[85]:
```
df.describe()
```

[85]:
```
            BALANCE  BALANCE_FREQUENCY     PURCHASES  ONEOFF_PURCHASES  \
count   8636.000000        8636.000000   8636.000000       8636.000000
mean    1601.224893           0.895035   1025.433874        604.901438
std     2095.571300           0.207697   2167.107984       1684.307803
min        0.000000           0.000000      0.000000          0.000000
25%      148.095189           0.909091     43.367500          0.000000
50%      916.855459           1.000000    375.405000         44.995000
75%     2105.195853           1.000000   1145.980000        599.100000
max    19043.138560           1.000000  49039.570000      40761.250000

       INSTALLMENTS_PURCHASES  CASH_ADVANCE  PURCHASES_FREQUENCY  \
count             8636.000000   8636.000000          8636.000000
mean               420.843533    994.175523             0.496000
std                917.245182   2121.458303             0.401273
min                  0.000000      0.000000             0.000000
25%                  0.000000      0.000000             0.083333
50%                 94.785000      0.000000             0.500000
75%                484.147500   1132.385490             0.916667
max              22500.000000  47137.211760             1.000000

       ONEOFF_PURCHASES_FREQUENCY  PURCHASES_INSTALLMENTS_FREQUENCY  \
count                 8636.000000                       8636.000000
mean                     0.205909                          0.368820
std                      0.300054                          0.398093
min                      0.000000                          0.000000
25%                      0.000000                          0.000000
50%                      0.083333                          0.166667
75%                      0.333333                          0.750000
max                      1.000000                          1.000000

       CASH_ADVANCE_FREQUENCY  CASH_ADVANCE_TRX  PURCHASES_TRX  CREDIT_LIMIT  \
```

```
count          8636.000000      8636.000000    8636.000000   8636.000000
mean              0.137604         3.313918      15.033233   4522.091030
std               0.201791         6.912506      25.180468   3659.240379
min               0.000000         0.000000       0.000000     50.000000
25%               0.000000         0.000000       1.000000   1600.000000
50%               0.000000         0.000000       7.000000   3000.000000
75%               0.250000         4.000000      18.000000   6500.000000
max               1.500000       123.000000     358.000000  30000.000000

           PAYMENTS  MINIMUM_PAYMENTS  PRC_FULL_PAYMENT       TENURE  \
count   8636.000000       8636.000000       8636.000000  8636.000000
mean    1784.478099        864.304943          0.159304    11.534391
std     2909.810090       2372.566350          0.296271     1.310984
min        0.049513          0.019163          0.000000     6.000000
25%      418.559237        169.163545          0.000000    12.000000
50%      896.675701        312.452292          0.000000    12.000000
75%     1951.142090        825.496463          0.166667    12.000000
max    50721.483360      76406.207520          1.000000    12.000000

           cluster
count  8636.000000
mean      1.712019
std       1.014589
min       0.000000
25%       1.000000
50%       2.000000
75%       3.000000
max       3.000000
```

2 select target variable

[ ]:
```python
x=df[['BALANCE', 'PURCHASES', 'ONEOFF_PURCHASES', 'CASH_ADVANCE',␣
↪'PURCHASES_FREQUENCY', 'CASH_ADVANCE_FREQUENCY', 'PURCHASES_TRX',␣
↪'CREDIT_LIMIT', 'PAYMENTS']]
```

3 normalization

[54]:
```python
x.shape
```

[54]:
```
(8636, 9)
```

[ ]:
```python
from sklearn .preprocessing import MinMaxScaler
```

[ ]:
```python
scaler=MinMaxScaler()
x_scaled=scaler.fit_transform(x)
```

[ ]:
```
x_scaled
```

4 model building

[55]:
```
from sklearn.cluster import KMeans
```

[56]:
```
# to determine kmeans am using elbow method
```

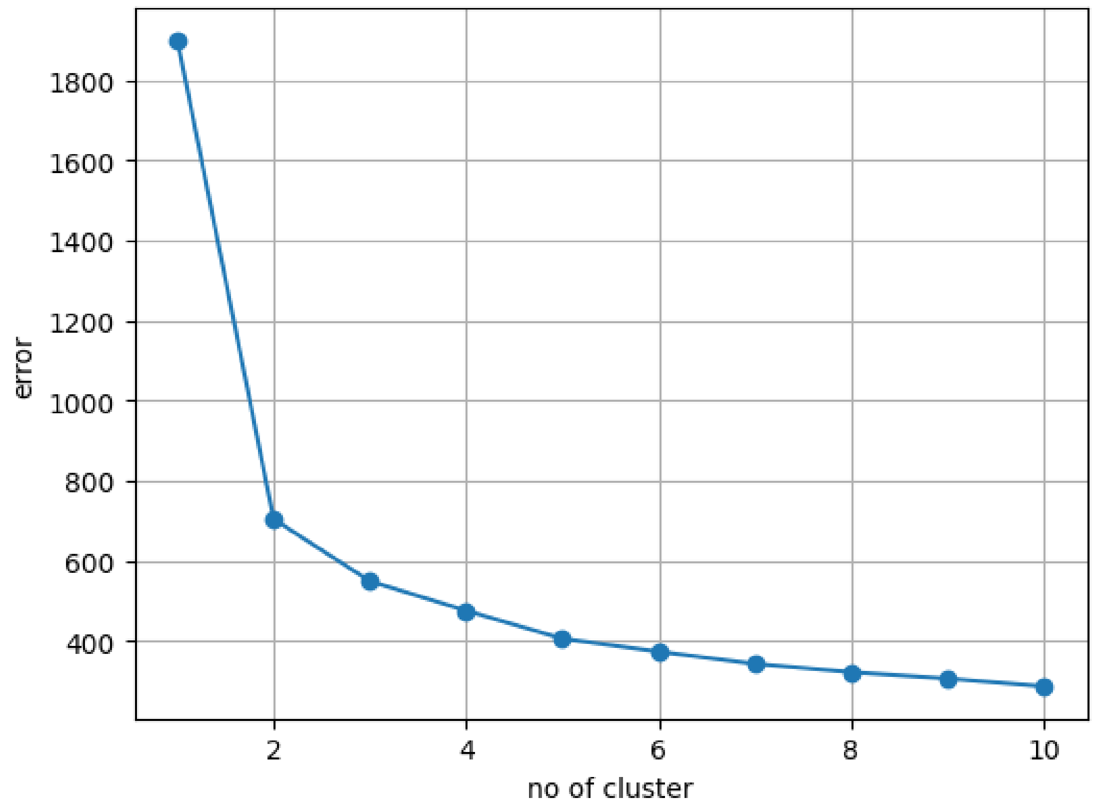
[57]:
```
import warnings
warnings.filterwarnings('ignore')
```

[58]:
```
error=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i). fit(x_scaled)
    error.append(kmeans.inertia_)
```

[ ]:
```
# plot the elbow method
```

[59]:
```
plt.plot(range(1,11),error,marker='o')
plt.grid(True)
plt.xlabel('no of cluster')
plt.ylabel('error')
# the elbow hand is btw 2 and 4 , i will chose 4 , it closer to point
```

[59]:
```
Text(0, 0.5, 'error')
```



[60]:
```
cluster=KMeans(4,random_state=0)
```

[61]:
```
cluster.fit(x_scaled)
```

[61]:
```
KMeans(n_clusters=4, random_state=0)
```

[62]:
```
label=cluster.labels_
```

[63]:
```
centroid=cluster.cluster_centers_
```

[64]:
```
centroid
```

[64]:
```
array([[0.23706451, 0.00493122, 0.00517144, 0.0880425 , 0.07656704,
        0.32208619, 0.00443459, 0.24761352, 0.06356372],
       [0.07825055, 0.04133436, 0.02751268, 0.01234673, 0.94301962,
        0.05341141, 0.08968989, 0.16827653, 0.04397727],
       [0.05847284, 0.01672681, 0.01268414, 0.01002344, 0.50457268,
        0.05181498, 0.02871611, 0.13483992, 0.02401614],
       [0.0562265 , 0.00339573, 0.00360175, 0.01658999, 0.07085595,
        0.08781136, 0.00346297, 0.10158343, 0.02178704]])
```

[ ]:

```python

```

[65]:

```python
df['cluster']=label
```

[66]:

```python
df['cluster']
```

[66]:

```
0       3
1       0
2       1
4       3
5       2
       ..
8943    3
8945    1
8947    1
8948    3
8949    2
Name: cluster, Length: 8636, dtype: int32
```

[ ]:

```python
# viewing each clusters
```

[ ]:

```python
first=df[df.cluster.isin([0])]
```

[ ]:

```python
second=df[df.cluster.isin([1])]
```

[ ]:

```python
third=df[df.cluster.isin([2])]
```

[ ]:

```python
fourth=df[df.cluster.isin([3])]
```

[ ]:

```python
# size of the clusters
```

[ ]:

```python
first.shape,second.shape,third.shape,fourth.shape
```

[ ]:

```python
import seaborn as sns
```

[67]:

```python
# plotting the clusters
```

[68]:

```python
tf=pd.DataFrame(x_scaled,columns=['BALANCE', 'PURCHASES', 'ONEOFF_PURCHASES',␣
↪'CASH_ADVANCE',
       'PURCHASES_FREQUENCY', 'CASH_ADVANCE_FREQUENCY', 'PURCHASES_TRX',
       'CREDIT_LIMIT', 'PAYMENTS'])
```

[79]:

```python
tf
```

[79]:

```
      BALANCE  PURCHASES  ONEOFF_PURCHASES  CASH_ADVANCE  \
0    0.002148   0.001945          0.000000      0.000000
1    0.168169   0.000000          0.000000      0.136685
```

```
2     0.131026  0.015766          0.018968      0.000000
3     0.042940  0.000326          0.000393      0.000000
4     0.095038  0.027188          0.000000      0.000000
…        …         …                 …              …
8631  0.000308  0.000426          0.000513      0.000000
8632  0.001496  0.005936          0.000000      0.000000
8633  0.001229  0.002945          0.000000      0.000000
8634  0.000707  0.000000          0.000000      0.000776
8635  0.019572  0.022293          0.026821      0.002695

      PURCHASES_FREQUENCY  CASH_ADVANCE_FREQUENCY  PURCHASES_TRX  \
0                0.166667                0.000000       0.005587
1                0.000000                0.166667       0.000000
2                1.000000                0.000000       0.033520
3                0.083333                0.000000       0.002793
4                0.666667                0.000000       0.022346
…                   …                       …              …
8631             0.166667                0.000000       0.002793
8632             1.000000                0.000000       0.016760
8633             0.833333                0.000000       0.013966
8634             0.000000                0.111111       0.000000
8635             0.666667                0.222222       0.064246

      CREDIT_LIMIT  PAYMENTS  clusters  cluster
0         0.031720  0.003978         3        3
1         0.232053  0.080892         0        0
2         0.248748  0.012263         1        1
3         0.038397  0.013373         3        3
4         0.058431  0.027602         2        2
…            …         …         …        …
8631      0.015025  0.001155         3        3
8632      0.031720  0.006418         1        1
8633      0.031720  0.001601         1        1
8634      0.015025  0.001035         3        3
8635      0.038397  0.001244         2        2

[8636 rows x 11 columns]
```

[75]:
```
tf['clusters']=label
```

[76]:
```
tf_group=tf.groupby('clusters').mean()
```

[81]:
```
x['cl']=label
```

[82]:
```
D=x.groupby('cl').mean()
```

[83]:
```
D
```

[83]:

```
        BALANCE    PURCHASES  ONEOFF_PURCHASES  CASH_ADVANCE  \
cl
0   4514.452353   241.824667        210.794335   4150.077810
1   1490.136151  2027.019482       1121.451302    581.990267
2   1113.506361   820.275463        517.021393    472.477044
3   1070.728965   166.525231        146.811750    782.005839

    PURCHASES_FREQUENCY  CASH_ADVANCE_FREQUENCY  PURCHASES_TRX  CREDIT_LIMIT  \
cl
0              0.076567                0.483129       1.587583   7466.024995
1              0.943020                0.080117      32.108982   5089.882145
2              0.504573                0.077722      10.280368   4088.455511
3              0.070856                0.131717       1.239744   3092.423786

       PAYMENTS
cl
0   3224.092469
1   2230.639485
2   1218.182488
3   1105.119531
```

[84]:

```
tf_group
```

[84]:

```
           BALANCE  PURCHASES  ONEOFF_PURCHASES  CASH_ADVANCE  \
clusters
0         0.237065   0.004931          0.005171      0.088042
1         0.078251   0.041334          0.027513      0.012347
2         0.058473   0.016727          0.012684      0.010023
3         0.056226   0.003396          0.003602      0.016590

          PURCHASES_FREQUENCY  CASH_ADVANCE_FREQUENCY  PURCHASES_TRX  \
clusters
0                    0.076567                0.322086       0.004435
1                    0.943020                0.053411       0.089690
2                    0.504573                0.051815       0.028716
3                    0.070856                0.087811       0.003463

          CREDIT_LIMIT  PAYMENTS
clusters
0             0.247614  0.063564
1             0.168277  0.043977
2             0.134840  0.024016
3             0.101583  0.021787
```

5 cluster Analysis

6 cluster 0:

7 the group have the highest balance of 4514 and updates frequently but makes low purchases and infrequent purchases,they pay up in advance

8 they contribute only 7.4% to company sales, in other words, the contribute very less to the bank.

9 cluster 1:

10 they makes the highest purchases by 62.3% to bank sales and frequently, they are the main source of company sales, they dont pay in advance but makes thier payment.

11 cluster2:

12 this group makes purchases and frequently, contribute 25.2% of company sales, they have low credit limit of 4086 and makes payment, they dont make advance payment also.

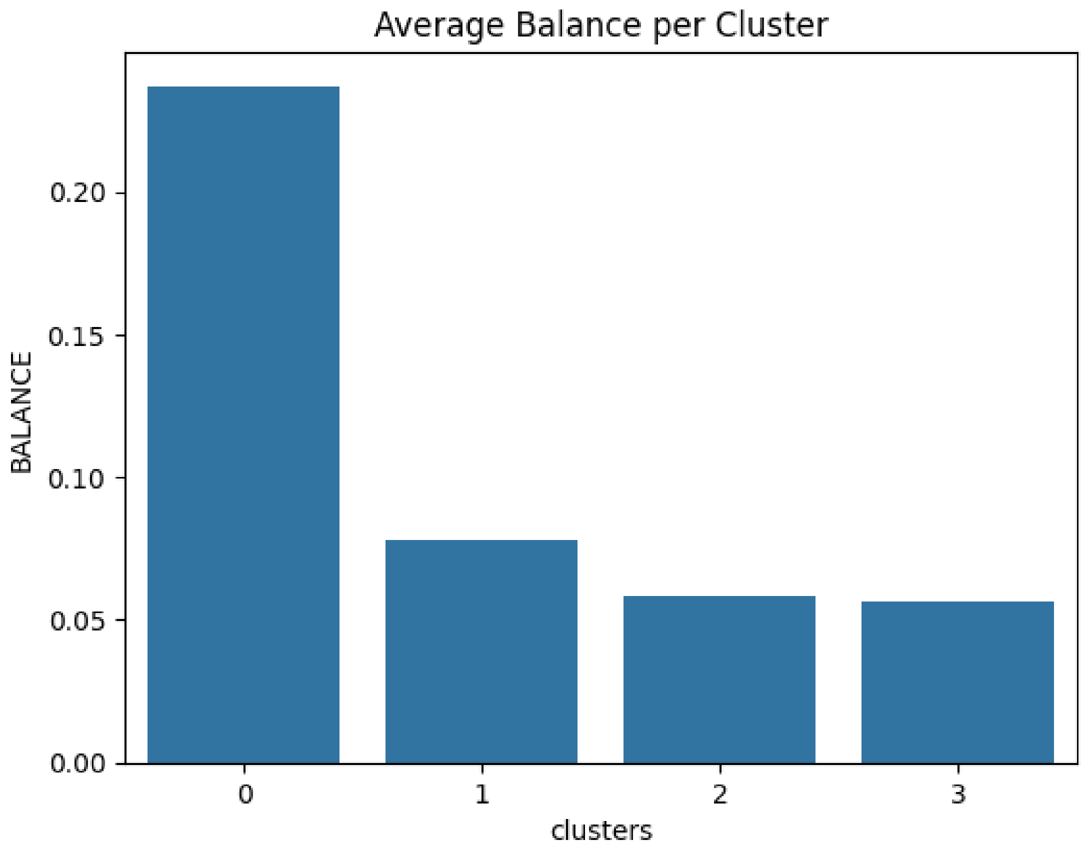
13 cluster 3:

14 this group has the lowest balance , makes the lowest purchases and not frequently, they contribute 5.1% to company sales. therefore they contribute very little to the company.

15 Interpret visulaization

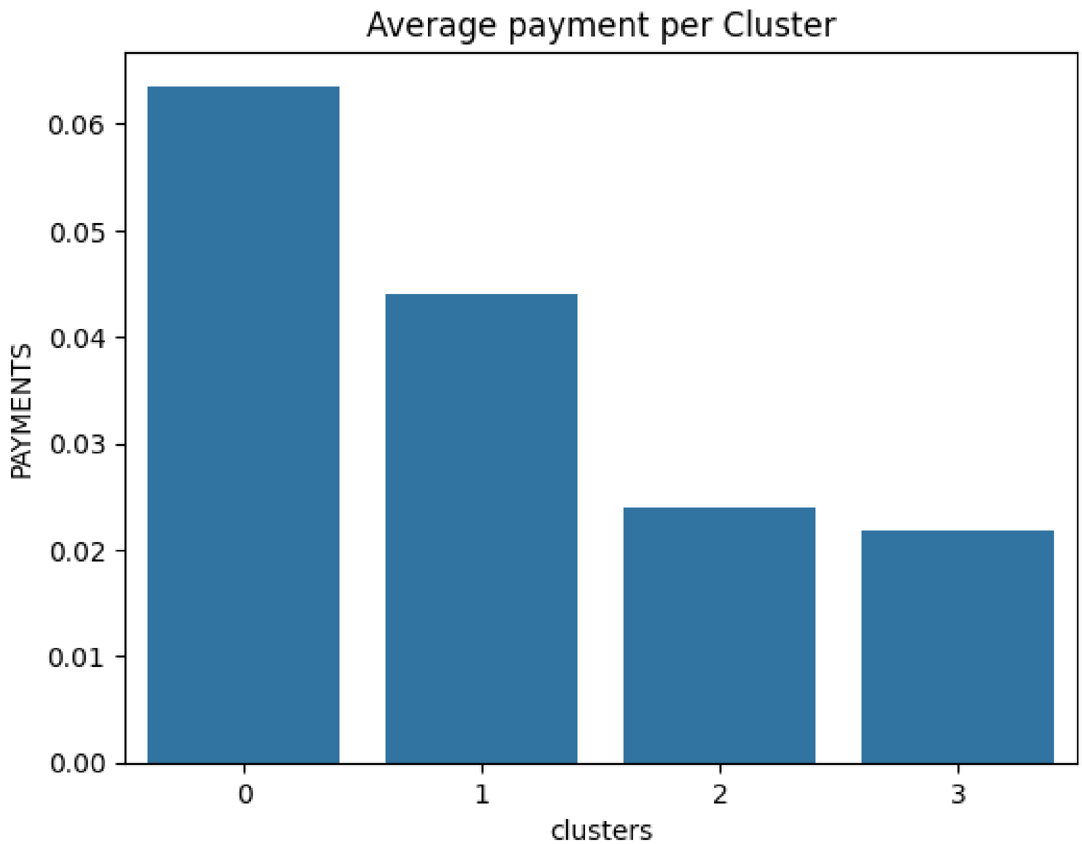
[86]:
```
sns.barplot(x='clusters', y='BALANCE', data=tf_group)
plt.title('Average Balance per Cluster')

# this shows that cluster 0 has the highest balance with lots of differences␣
 ↪compare to the other clusters
```

[86]:
```
Text(0.5, 1.0, 'Average Balance per Cluster')
```



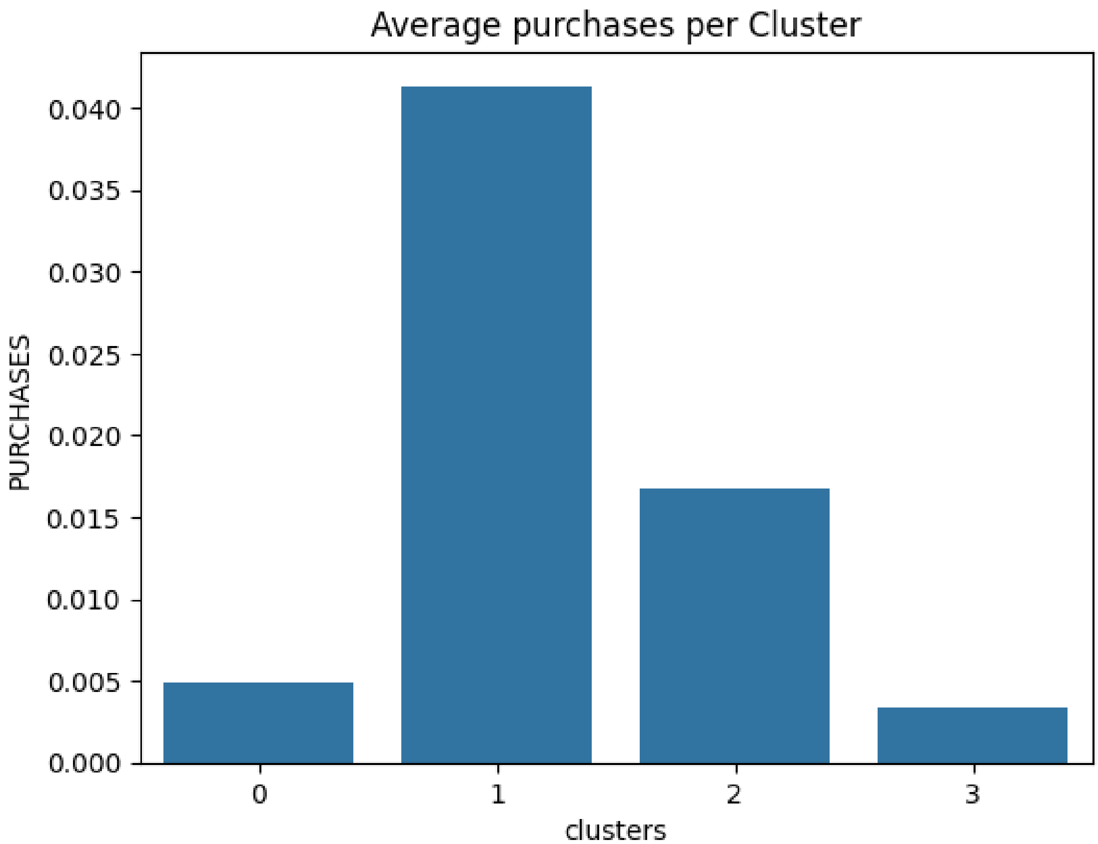
[87]:
```
sns.barplot(x='clusters', y='PAYMENTS', data=tf_group)
plt.title('Average payment per Cluster')

# this also shows that cluster 0 makes payment time
```

[87]:
```
Text(0.5, 1.0, 'Average payment per Cluster')
```



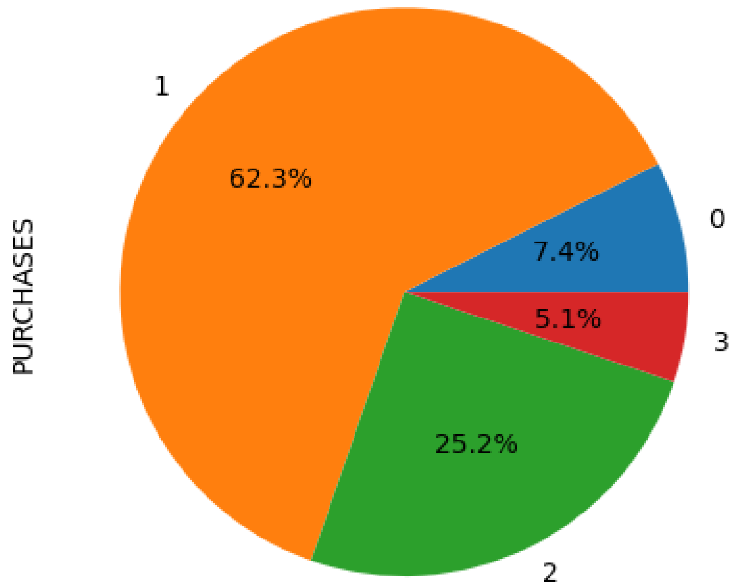
[88]:
```
sns.barplot(x='clusters', y='PURCHASES', data=tf_group)
plt.title('Average purchases per Cluster')

# this shows that cluster 1 and 2 makes the highest puchases in the money by 62.
 ↪3%
```

[88]:
```
Text(0.5, 1.0, 'Average purchases per Cluster')
```



[89]:
```
tf.groupby(['clusters'])['PURCHASES'] .mean().plot(kind='pie',autopct= '%1.1f%%␣
↪' )
```

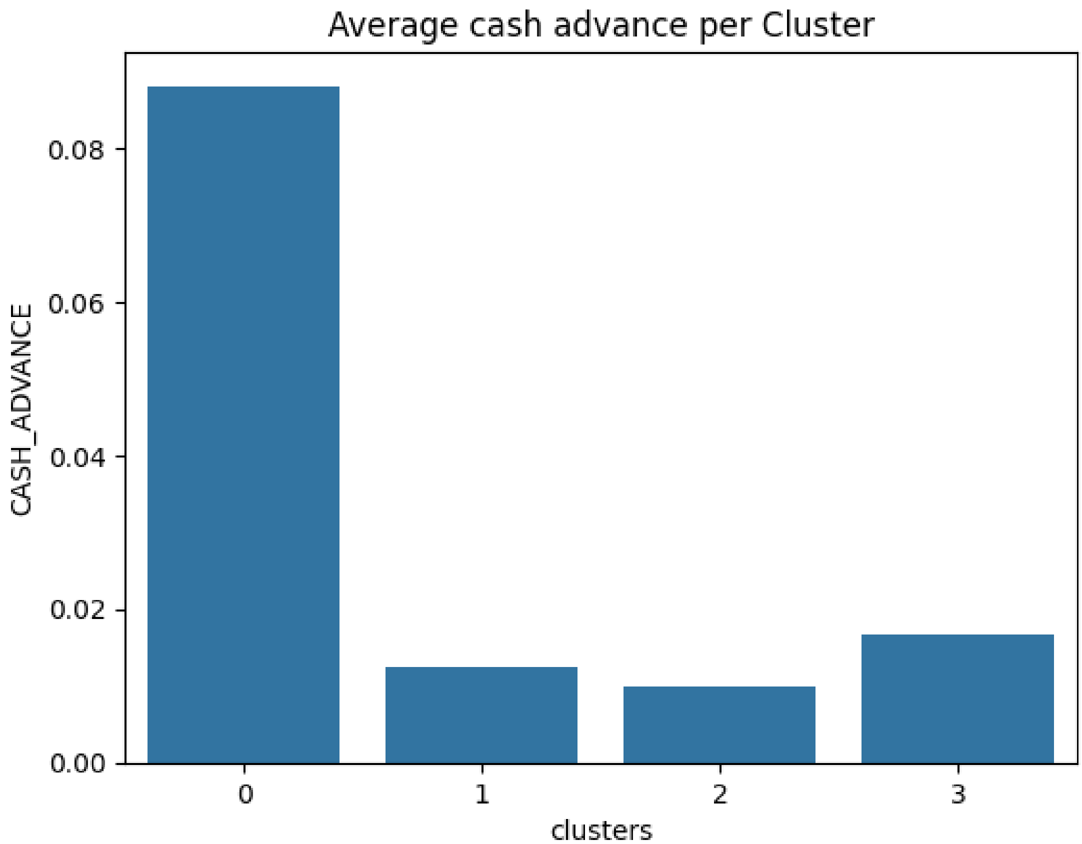
[89]: <Axes: ylabel='PURCHASES'>



[ ]:
```
# centroids is the same values as centroids
```

[90]:
```
sns.barplot(x='clusters', y='CASH_ADVANCE', data=tf_group)
plt.title('Average cash advance per Cluster')

# cluster pays up on time but makes no purchases
```

[90]: Text(0.5, 1.0, 'Average cash advance per Cluster')


Average cash advance per Cluster
CASH_ADVANCE
0.00
0.02
0.04
0.06
0.08
0
1
2
3
clusters

16 Recommendations and strategy

17 for cluster 0 and 3:

18 the bank should advertise and encourage them to make purchases by implementing discounts, promotions e.g, buy one and get one free,

19 free gifts,incentives and other promotional means. send advert email of thier bonuses, communicate the importance of the company product

20 to the clusters through emails,create combo packages. Also hand out questionaires to understand the interest of this clusters.

21 that way the company can know what to advertise and how to advertise it to interest the clusters.

22 cluster 1 and 2 :

23 they bank should increase thier credit limit and put up bonuses to encourage more spending

[ ]:

```
```