Data Science Problem

(Associate Data Scientist)

Aditya Gupta
aditya17325@iiitd.ac.in
9560903351

Data Description

The sample 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.

Dealing with Missing Entries

- Finding the columns with missing entries:
  - data.isnull().sum()
- Filling the missing entries with median of the respective columns
  - credit_limit = data['CREDIT_LIMIT']
  - credit_limit.fillna(credit_limit.median(), inplace=True)
  - min_payments = data['MINIMUM_PAYMENTS']
  - min_payments.fillna(min_payments.median(), inplace=True)

Deriving Intelligent KPIs

Averages of Purchases and Cash Advances

1. Monthly Average Purchases -

   data['MonthlyAvgPurchases'] = data['PURCHASES']/data['TENURE']

2. Monthly Cash Advance -

   data['MonthlyCashAdvance'] =

   data['CASH_ADVANCE']/data['TENURE']

Analysing the Purchasing Behaviour

Python Code:

```
print(data[(data['ONEOFF_PURCHASES']==0) & (data['INSTALLMENTS_PURCHASES']==0)].shape)
print(data[(data['ONEOFF_PURCHASES']>0) & (data['INSTALLMENTS_PURCHASES']>0)].shape)
print(data[(data['ONEOFF_PURCHASES']>0) & (data['INSTALLMENTS_PURCHASES']==0)].shape)
print(data[(data['ONEOFF_PURCHASES']==0) & (data['INSTALLMENTS_PURCHASES']>0)].shape)
```

Output:

```
(2042, 20)
(2774, 20)
(1874, 20)
(2260, 20)
```

We can clearly see that there are four different types of customers, i.e., customers with:

1. One-off purchases
2. Installment Purchases
3. Both
4. Neither

Dividing the Customers into Categories

Python Code:

```
def PurchasingBehaviour(data):
        if (data['ONEOFF_PURCHASES']>0) & (data['INSTALLMENTS_PURCHASES']==0):
                return 'OneOff'
        if (data['ONEOFF_PURCHASES']==0) & (data['INSTALLMENTS_PURCHASES']>0):
                return 'Installment'
        if (data['ONEOFF_PURCHASES']>0) & (data['INSTALLMENTS_PURCHASES']>0):
                return 'Both'
        if (data['ONEOFF_PURCHASES']==0) & (data['INSTALLMENTS_PURCHASES']==0):
                return 'Neither'
data['PurchaseBehaviour']=data.apply(PurchasingBehaviour,axis=1)
```

Deriving Categorical KPI based on Purchasing Behaviour

Python Code:

```
data['PurchaseBehaviour']=data.apply(PurchasingBehaviour,axis=1)
```

So now we have a column defining that particular customers purchasing behaviour from credit card.

Limiting the Credit Usage of Customers

Python Code:

```
data['LimitUsage'] = data.apply(lambda x: x['BALANCE']/x['CREDIT_LIMIT'], axis=1)
```

This helps us to find the ratio between the customer’s balance and credit limit.

Lower the ratio, the better the customer is in paying the dues.

Payments to Minimum Payments Ratio

Python Code:

```
data['MinPayment']=data.apply(lambda x:x['PAYMENTS']/x['MINIMUM_PAYMENTS'],axis=1)
```

This makes another column of payments to minimum payments ratios for every customer.

Treating Extreme Values

There exist extreme values in the data which can cloud the end result and inferences. So to get rid of the haziness in the data, we will do log transformation of the dataset.

Python Code:

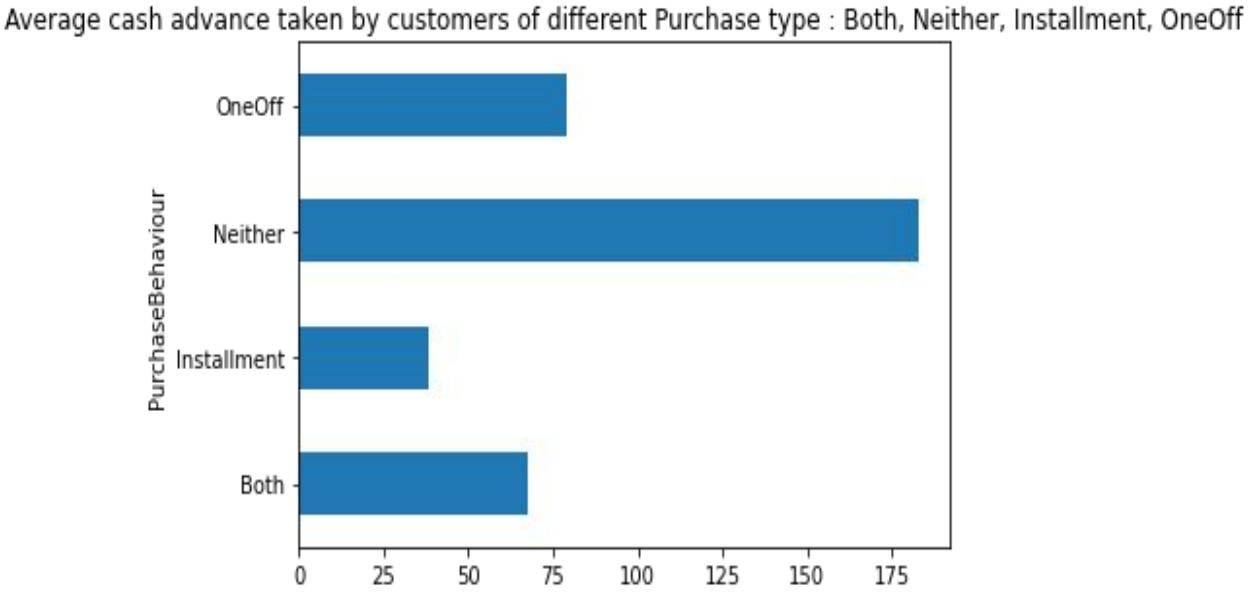
```
DataLog=data.drop(['CUST_ID','PurchaseBehaviour'],axis=1).applymap(lambda x: numpy.log(x+1))
```

Average Payments to Minimum Payments Ratio

Insight - Customers with Installment purchases are paying their dues.

Analysing the Cash Advance and Purchasing behaviours

Customers in neither of the two purchasing types, take more cash in advance



Devising the ML Algorithm

Making dummy data from the 4 categories of purchasing habits.

Python Code:

```
PreData['PurchaseBehaviour'] = data.loc[:,'PurchaseBehaviour']
pandas.get_dummies(PreData['PurchaseBehaviour'])
```

Merging the Dummy data with the Original Dataset

Python Code for concatenation:

```
DataDummy = pandas.concat([PreData,pandas.get_dummies(PreData['PurchaseBehaviour'])], axis=1)
```

Now we don't need the column of Purchase Behaviour so we will drop it from the dummy..

Heat Map of the Dummy Data

Heat Map of the Dummy Data

The heatmap does show correlation between several features of the dataset. Implementing dimensionality reduction will help in detering the multicollinearity of the data.

Applying PCA on the dataset

Standardizing the data before PCA, to get rid of scaling effects.

Python Code:

```
scaler = preprocessing.StandardScaler()

ScaledData = scaler.fit_transform(DataDummy)
pca = decomposition.PCA(n_components=17)
DataPCA = pca.fit(ScaledData)
```

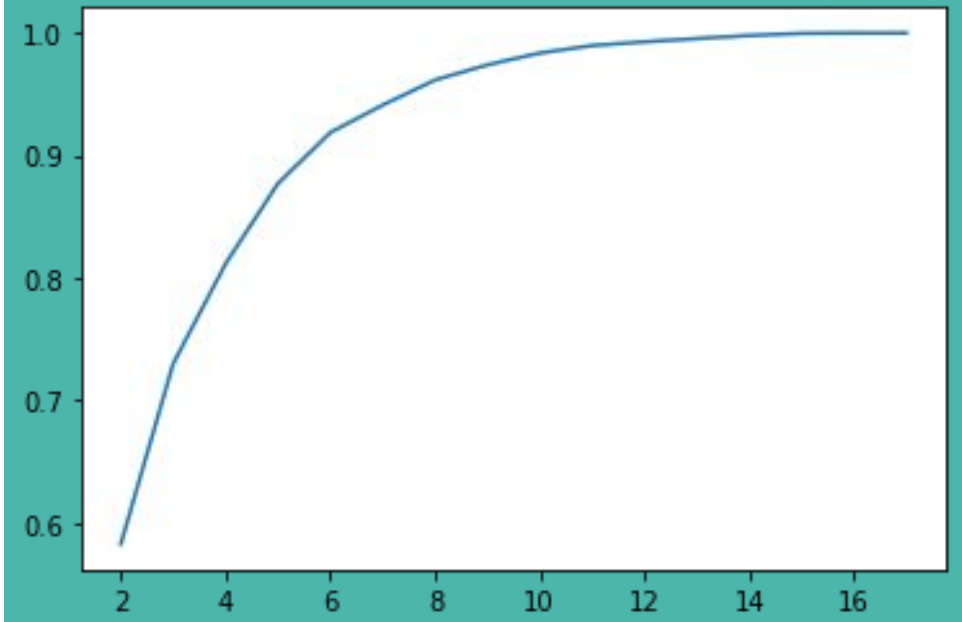
Checking the Variance Ratios of the various KPIs

Python Code:

```
VarianceRatios = {}
for xyz in range(2,18):
    pca = decomposition.PCA(n_components=xyz)
    DataPCA = pca.fit(ScaledData)
    VarianceRatios[xyz]=sum(DataPCA.explained_variance_ratio_)
```

Variance Ratios

5 components are painting approximately 87% of the variance of the dataset, so we select only those, to narrow our field of view to something substantive.



Selecting 5 Principal Components

Python Code:

```
PCs = decomposition.PCA(n_components=6).fit(ScaledData)

ReducedData = PCs.fit_transform(ScaledData)
df = pandas.DataFrame(ReducedData)
pandas.DataFrame(PCs.components_.T,
                 columns = ['PC: ' + str(s) for s in range(6)],
                 index = DataDummy.columns)
```

Now we have eigenvectors for all the principal components,

Clustering

There are 4 different types of customers, so we should start with 4 cluster., based on the distinct purchasing behaviours.

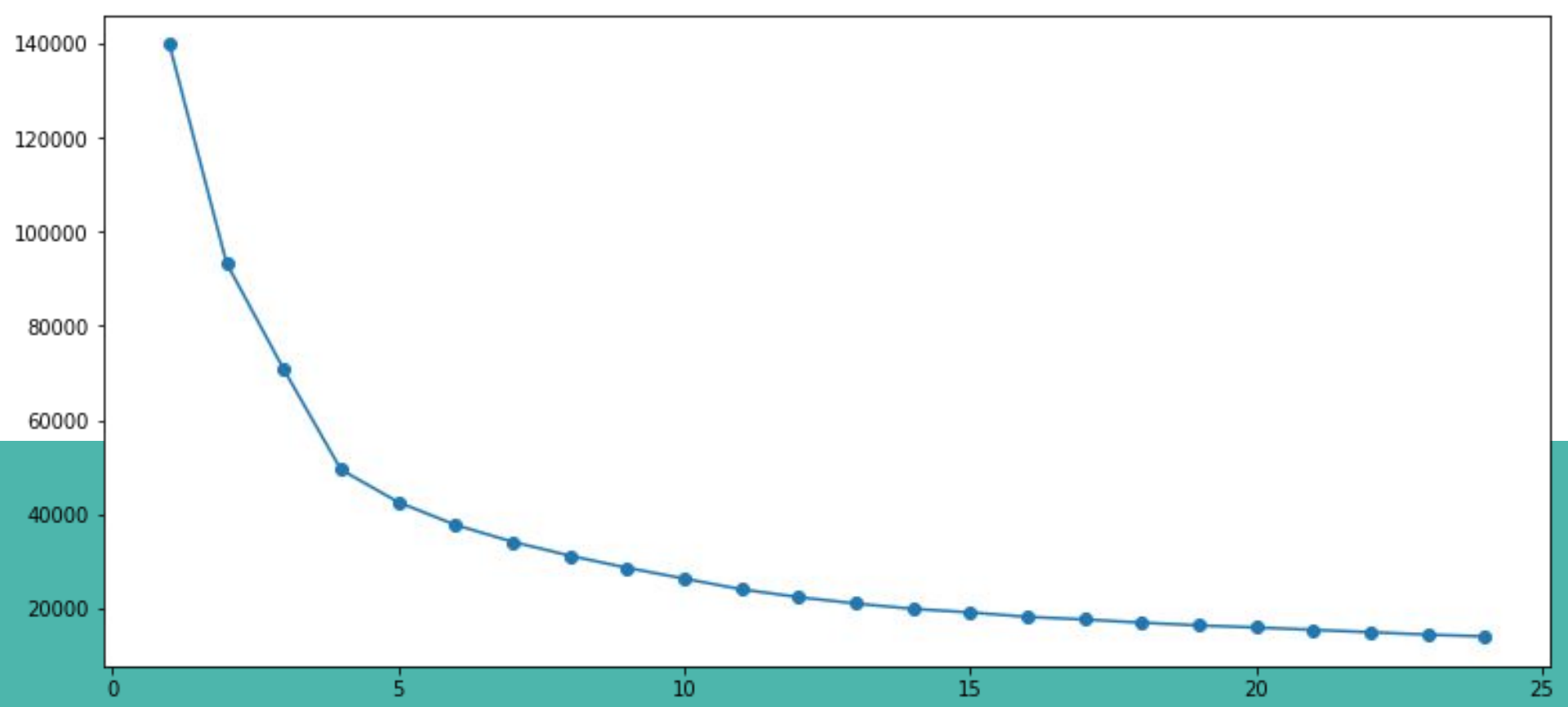
Python Code:

```
KM4 = cluster.KMeans(n_clusters = 4, random_state = 123)
KM4.fit(ReducedData)
KM4.labels_
pandas.Series(KM4.labels_).value_counts()
```

Identifying Cluster Error

Python Code:

```
ClusterRange = range( 1, 25 )
ClusterErrors = []
for n in ClusterRange:
    clusters = cluster.KMeans(n)
    clusters.fit(ReducedData)
    ClusterErrors.append(clusters.inertia_)
```



This is the graph of the cluster errors. It is clear that the elbow range is between 4 to 6.

Silhouette Coefficient

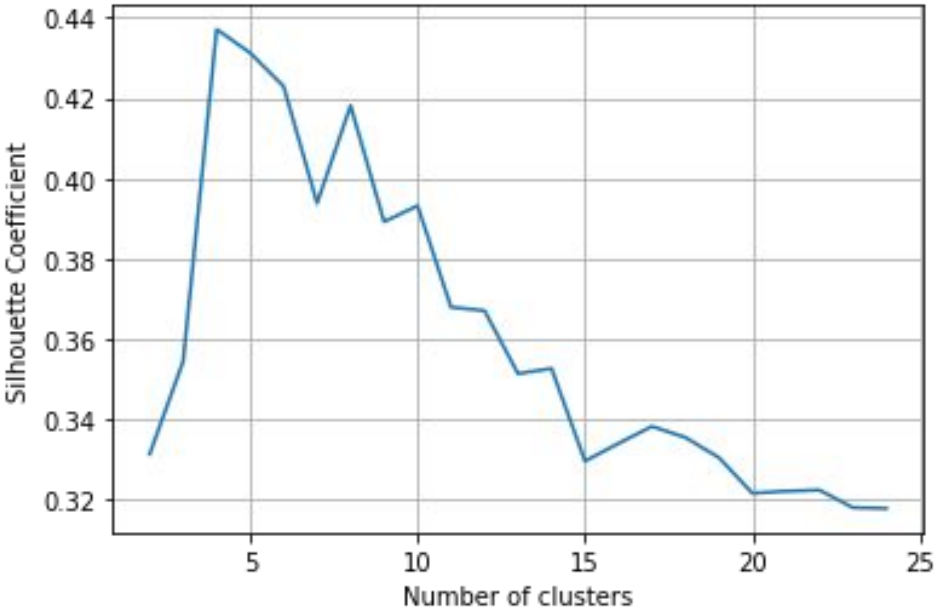
In [127]:

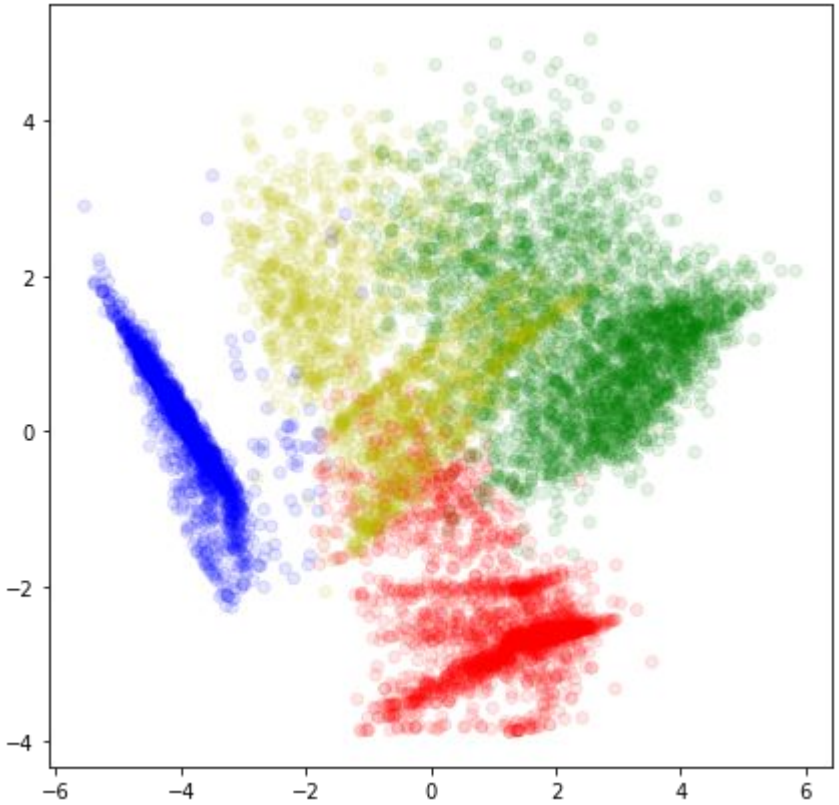
Python Code:

```
KRange = range(2, 25)

Scores = []
for K in KRange:
    KM = cluster.KMeans(n_clusters=K, random_state=1)
    KM.fit(ReducedData)
    Scores.append(metrics.silhouette_score(ReducedData, KM.labels_))
```

Plotting the Results





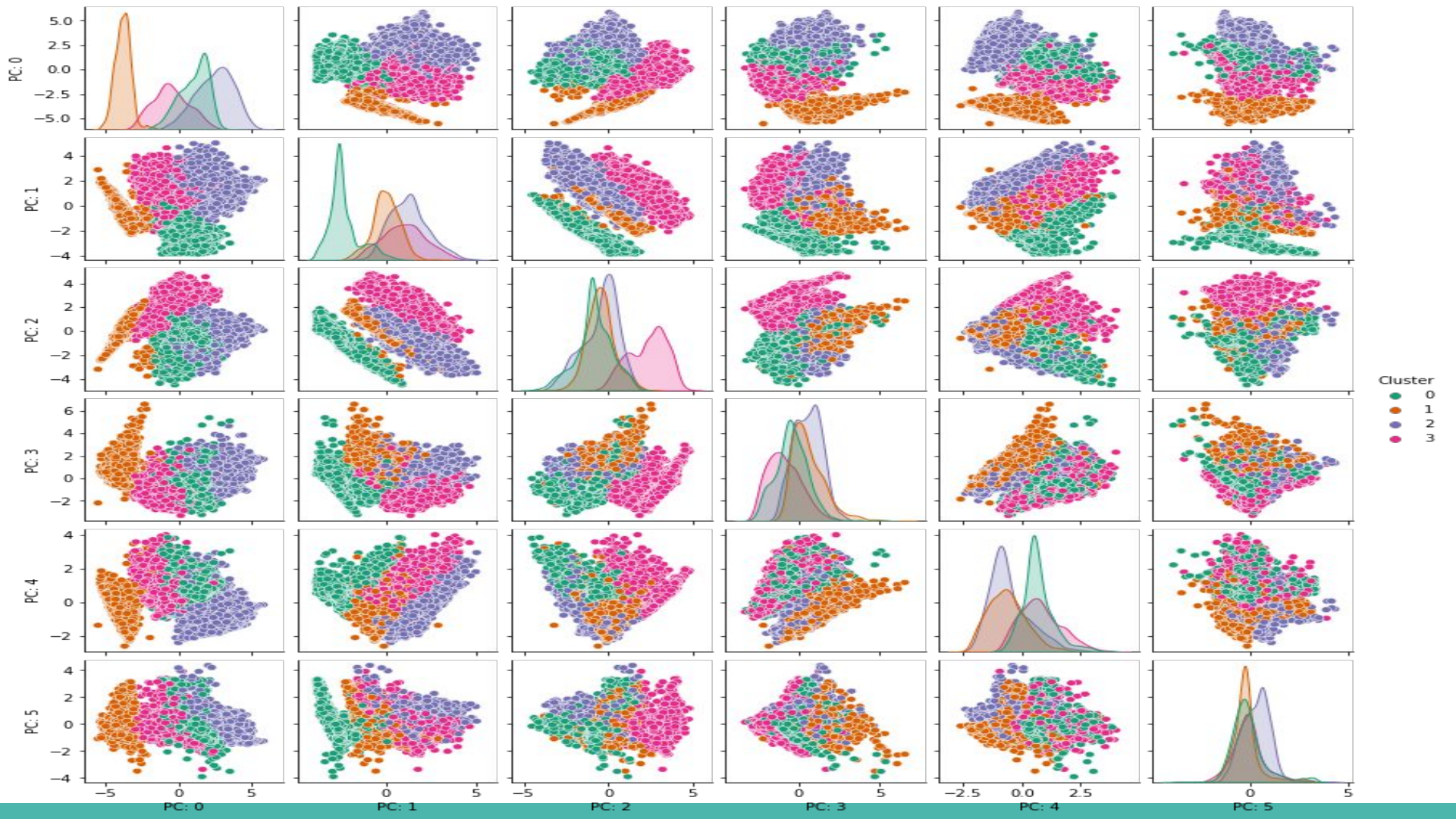
It was very difficult to separate plots for each cluster. So I used pair plot, to get all the graphs in one go.

Pair Plots

Python Code:

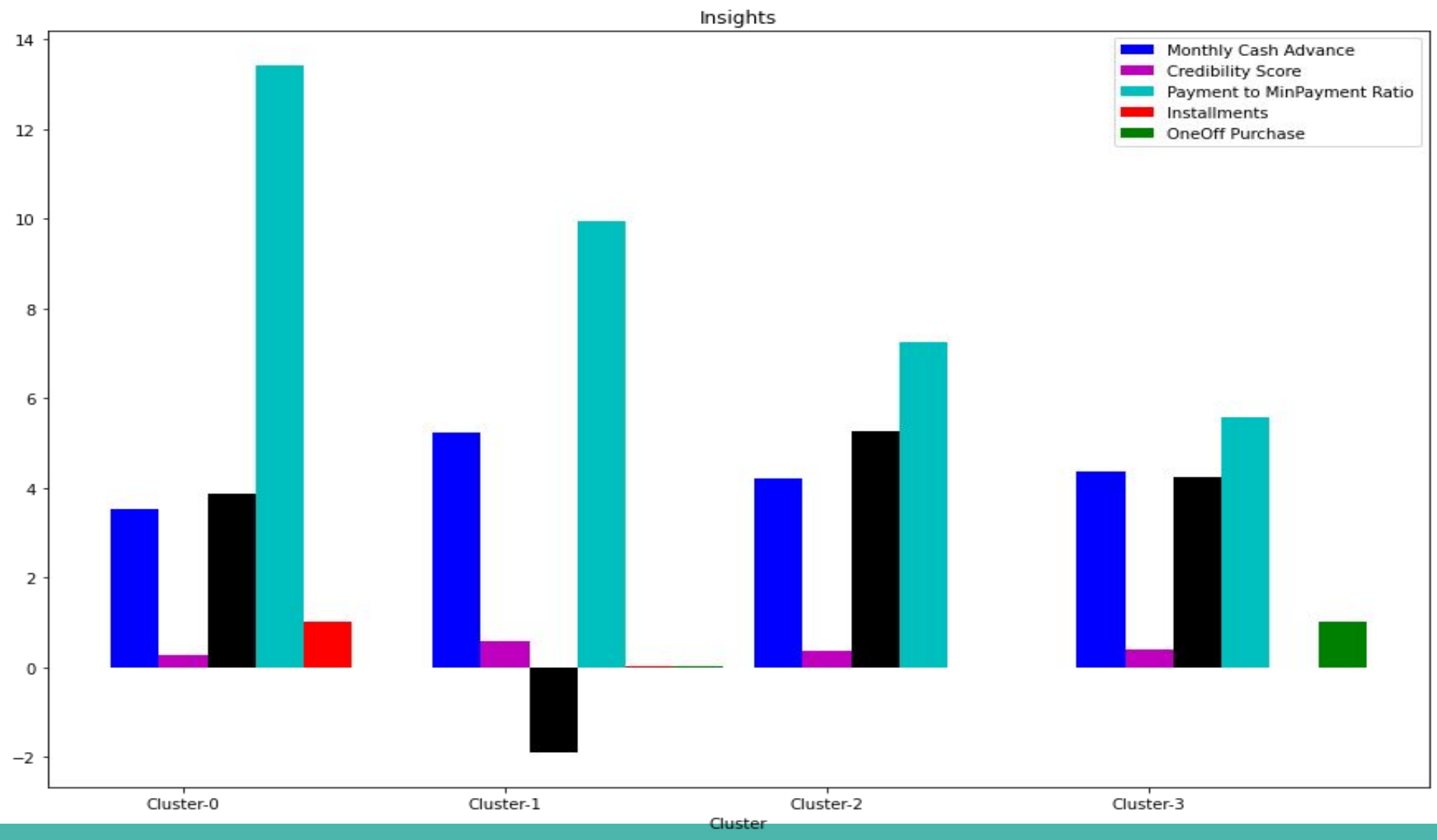
```
DFPairPlot=pandas.DataFrame(ReducedData,columns=['PC: ' +str(i) for i in range(6)])

DFPairPlot['Cluster']=KM4.labels_
seaborn.pairplot(DFPairPlot,
                 hue='Cluster',
                 palette= 'Dark2',
                 diag_kind='kde',
                 height = 2)
```


PC: 0
PC: 1
PC: 2
PC: 3
PC: 4
PC: 5
Cluster
0
1
2
3

Merging the Clusters with the Original Data Frame

```
ColsKPI=['PURCHASES_TRX','MonthlyAvgPurchases','MonthlyCashAdvance','Li
mitUsage','CASH_ADVANCE_TRX','MinPayment','Both','Installment','Neither','On
eOff','CREDIT_LIMIT']
ClusterDF4 = pandas.concat([DataOg[ColsKPI],
                            pandas.Series(KM4.labels_,name='Cluster4')],axis=1)
Clus4 = ClusterDF4.groupby('Cluster4')\.apply(lambda x: x[ColsKPI].mean()).T
```


Insights
Monthly Cash Advance
Credibility Score
Payment to MinPayment Ratio
Installments
OneOff Purchase
14
12
10
8
6
4
2
0
−2
Cluster-0
Cluster-1
Cluster-2
Cluster-3
Cluster

Insights

Cluster 0 - highest monthly average purchases

- incur both purchases (installments and one-off)
- good credibility score
- 31% of the customer base

Cluster 1 - maximum cash in advance taken

- less minimum payment
- weak credibility score
- least purchases
- 23% of the customer base

Insights (contd.)

Cluster 2 - best credibility score

- highest installment purchases
- 25% of the customer base

Cluster 3 - maximum one-off purchases

- least payment to minimum payment ratio
- 21% if the customer base

Strategies based on Cluster's Characteristics

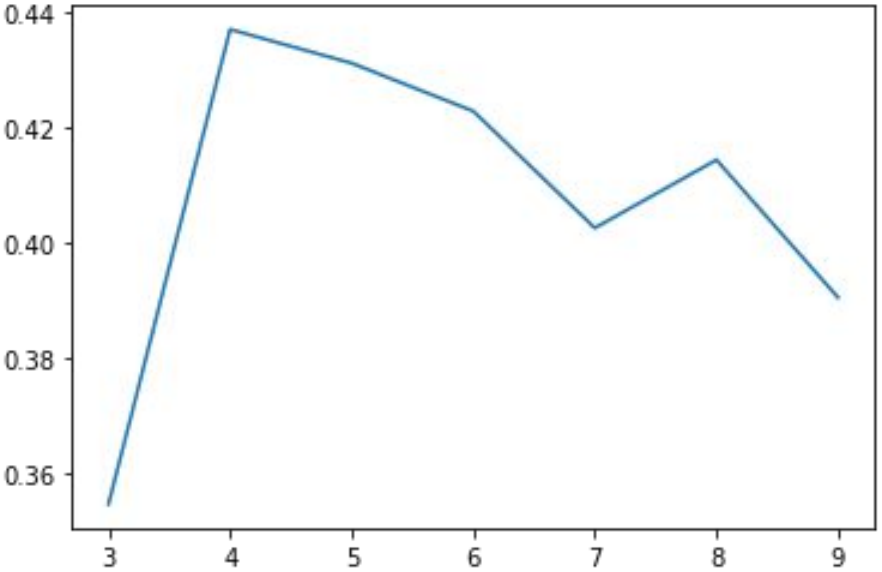
Cluster 0 - should be given platinum, or special edition cards with added benefits lie loyalty bonus and progressive bonuses on subsequent purchases to increase the number of transactions

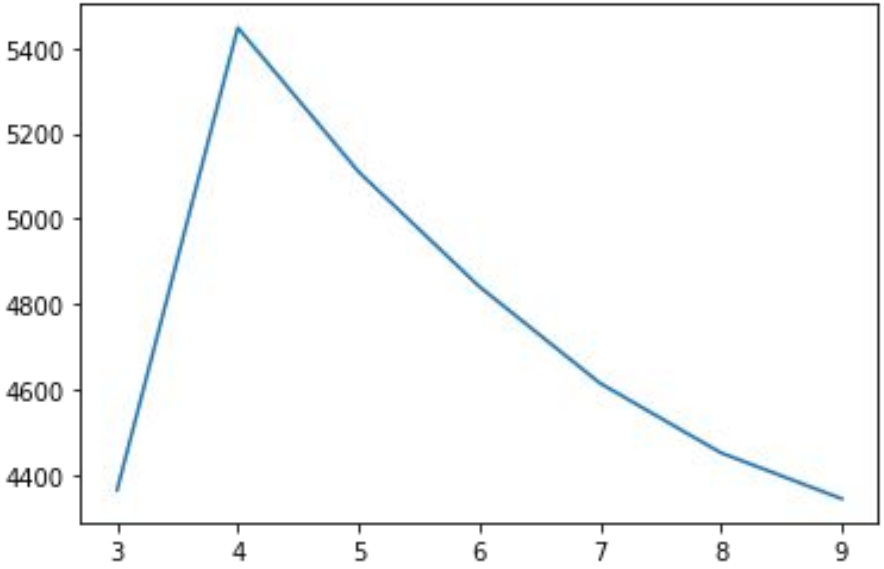
Cluster 1 - provide less interest on purchase transactions

Cluster 2 - provide purchase incentives to raise the number of purchases

Cluster 3 - can't understand this group of customers, as it makes one-off purchases and also has a low payment to mimpayment ratio.

Silhouette and Calinski Harabasz Scores





Conclusion

Clustering with 4 principal components was the right thing to do, as making more clusters would just have resulted in overlapping characteristics of one or more clusters.

Furthermore, the Silhouette and Calinski performance metrics also depict a similar picture. It can be clearly seen in the graphs that K-means with 4 clusters would bring out the distinct characteristics of all credit card customers and divide them into 4 different heterogeneous groups.