SCHOOL OF COMPUTING, ENGINEERING AND DIGITAL TECHNOLOGY

MSc Data science

ADVANCED DATA ANALYTICS

Group 14

VISHWAS SIMHA SAMUDRALA(W9316602)

ARUN SAI PILLI (W93223581)

SRINIVAS YASWANTH THALLURI (W9330556)

DEVENDRA SAI MUPPALLA (W9312419)

MANIDEEP KOMMARAJU (W9311652)

SEBIN SUNNY (W9479792)

Word count: 3820 (Excluding references)

CONTENTS:

1 ABSTRACT:

In the banking sector, the credit card department is vulnerable for fraud and can cause huge losses to banks. Understanding the customer's transactional history could give more insights about customer behavior towards credit card usage. In comparison to debit card, credit card customers have a greater number of transactions which is causing complexity to differentiate valuable and default customers. Large number of customers data should be pre-processed to get some useful information, this can be achieved by using the collection of different customer transaction data. The purpose of this assignment is to segregate credit card customer into different groups by using clustering method and then these clusters are used as class labels to apply decision tree method for future prediction of customer cluster.

2. INTRODUCTION:

Credit card frauds has increased significantly in the recent decades resulting losses in billions to the banking sector, making it important to understand customer behavior in this domain. Finding valuable customers based on their credit card usage by applying clusters can result in finding valuable customers or target customers who bring profits to banks. This can be achieved by considering all the Qualitative and Quantitative attributes from a transaction by co-relating with other customer transactions to find out different patterns. Clustering is the perfect method for this type of scenario. It groups the similar characters of attributes with other transactions resulting into clusters.

When have chosen a credit card dataset from Kaggle, this includes 18 unique attributes explaining different characteristics of the transaction, full description about dataset is given in the below section. Clustering is performed using K-means algorithm, the resulting cluster values will be then appended to the dataset to implement classification using Decision Tree algorithm.

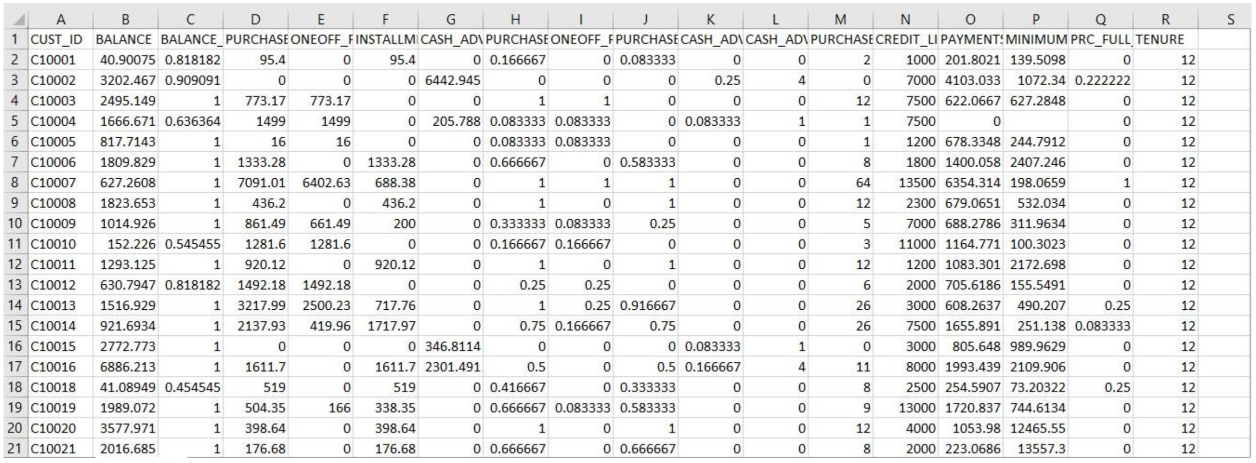
From visualizations we can see possible outliers formed due to many transactions or incorrect data, in our decision tree classification we are more focused to bring out more valuable customers not on outliers (fraudulent customers) so these are eliminated using Z-Score methodology and new dataset is created only with customer range closer to each other. By using information gain new dataset attribute values are converted between -1 to 1 and then dataset is split into train and test data set where we will be training our model on train dataset and test our test dataset for future prediction. At the end of the algorithm, we will be showing some clusters with profitable customers and banks provide them with interesting offers. The end results are compared by doing accuracy check with confusion matrix showing our model prediction rate for cluster column.

2.1 RELATED WORK

Research Article	Notes
1. “Credit Card Customer Segmentation and Target Marketing Based on Data Mining5”.	This article gives insights about how banks perform customer segmentation and marketing strategies that can be implemented and practiced by banks and marketing representatives for customer targeting.
2. “Theoretical Study of Decision Tree Algorithms to Identify Pivotal Factors for Performance Improvement: A Review3”	This paper aims to identifying and inspecting the vital criteria or the factors of the DT algorithm. It's reviewing about to provide a direction for the selection of a basic enhancement factor for the Decision Tree algorithm as per the problem or requirement
3. “Segmentation of bank customers by expected benefits and attitudes1”.	From this article we can understand how segmentation based on expected benefits and attitudes is better than demographic factors. Most banking systems use demographics factors to analyse customers in spite knowing that they have low correlation value.
4. “Using Unsupervised Machine Learning Techniques for Behavioural-based Credit Card Users Segmentation in Africa2”.	This article brings knowledge about methods to increase credit card customers across different banks in Africa based on their credit card usage stats.
5. “Predicting Breast Cancer using effective Classification with Decision Tree6”	Explaining about Predicting the class label of the data points given, and how to train the data to be identify the given as input variable that relates to the class. However, the decision tree is supervised learning which is provided with input data.

3. DATA EXPLORATION:

To understand and study Clustering and Classification algorithms we have chosen Credit Card Dataset from Kaggle. This dataset contains 9000 unique customers and their 18 different features.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	CUST_ID	BALANCE	BALANCE_	PURCHASE	ONEOFF_F	INSTALLMI	CASH_ADV	PURCHASE	ONEOFF_F	PURCHASE	CASH_ADV	CASH_ADV	PURCHASE	CREDIT_LI	PAYMENTS	MINIMUM	PRC_FULL_	TENURE	
2	C10001	40.90075	0.818182	95.4	0	95.4	0	0.166667	0	0.083333	0	0	2	1000	201.8021	139.5098	0	12	
3	C10002	3202.467	0.909091	0	0	0	6442.945	0	0	0	0.25	4	0	7000	4103.033	1072.34	0.222222	12	
4	C10003	2495.149	1	773.17	773.17	0	0	1	1	0	0	0	12	7500	622.0667	627.2848	0	12	
5	C10004	1666.671	0.636364	1499	1499	0	205.788	0.083333	0.083333	0	0.083333	1	1	7500	0		0	12	
6	C10005	817.7143	1	16	16	0	0	0.083333	0.083333	0	0	0	1	1200	678.3348	244.7912	0	12	
7	C10006	1809.829	1	1333.28	0	1333.28	0	0.666667	0	0.583333	0	0	8	1800	1400.058	2407.246	0	12	
8	C10007	627.2608	1	7091.01	6402.63	688.38	0	1	1	1	0	0	64	13500	6354.314	198.0659	1	12	
9	C10008	1823.653	1	436.2	0	436.2	0	1	0	1	0	0	12	2300	679.0651	532.034	0	12	
10	C10009	1014.926	1	861.49	661.49	200	0	0.333333	0.083333	0.25	0	0	5	7000	688.2786	311.9634	0	12	
11	C10010	152.226	0.545455	1281.6	1281.6	0	0	0.166667	0.166667	0	0	0	3	11000	1164.771	100.3023	0	12	
12	C10011	1293.125	1	920.12	0	920.12	0	1	0	1	0	0	12	1200	1083.301	2172.698	0	12	
13	C10012	630.7947	0.818182	1492.18	1492.18	0	0	0.25	0.25	0	0	0	6	2000	705.6186	155.5491	0	12	
14	C10013	1516.929	1	3217.99	2500.23	717.76	0	1	0.25	0.916667	0	0	26	3000	608.2637	490.207	0.25	12	
15	C10014	921.6934	1	2137.93	419.96	1717.97	0	0.75	0.166667	0.75	0	0	26	7500	1655.891	251.138	0.083333	12	
16	C10015	2772.773	1	0	0	0	346.8114	0	0	0	0.083333	1	0	3000	805.648	989.9629	0	12	
17	C10016	6886.213	1	1611.7	0	1611.7	2301.491	0.5	0	0.5	0.166667	4	11	8000	1993.439	2109.906	0	12	
18	C10018	41.08949	0.454545	519	0	519	0	0.416667	0	0.333333	0	0	8	2500	254.5907	73.20322	0.25	12	
19	C10019	1989.072	1	504.35	166	338.35	0	0.666667	0.083333	0.583333	0	0	9	13000	1720.837	744.6134	0	12	
20	C10020	3577.971	1	398.64	0	398.64	0	1	0	1	0	0	12	4000	1053.98	12465.55	0	12	
21	C10021	2016.685	1	176.68	0	176.68	0	0.666667	0	0.666667	0	0	8	2000	223.0686	13557.3	0	12	

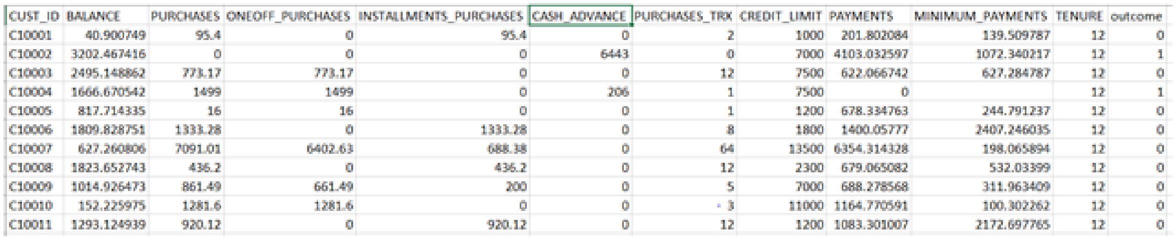
Below is the description of every column in the dataset.

CUSTID: IDENTIFICATION OF CREDIT CARD HOLDER	PURCHSESINSTALMENTFREQUENCY: HOW FREQUENTLY PURCHASES IN INSTALMENTS ARE MADE
BALANCE: BALANCE AMOUNT IN THEIR ACCOUNT	CASHADVANCEFREQUENCY: FREQUENCY OF CASH PAID IN ADVANCE
BALANCEFREQUENCY: HOW FREQUENTLY THE BALANCE IS UPDATED	PURCHASESTRX: NO OF PURCHASE TRANSACTIONS
PURCHASES: AMOUNT OF PURCHASES MADE FROM ACCOUNT	CREDITLIMIT: LIMIT OF CREDIT CARD FOR USER
ONEOFFPURCHASES: MAXIMUM PURCHASE AMOUNT IN ONE GO	PAYMENTS: AMOUNT OF PAYMENT DONE BY USER
INSTALMENTSPURCHASES: AMOUNT OF PURCHASES IN INSTALMENT	MINIMUMPAYMENTS: MINIMUM NO OF PAYMENTS MADE
CASHADVANCE: CASH IN ADVANCE GIVEN BY THE USER	PRCFULLPAYMENT: PERCENT OF FULL PAYMENT MADE BY USER
PURCHASEFREQUENCY: HOW FREQUENTLY PURCHASES ARE MADE	TENURE: TENURE YEARS OF CREDIT CARD SERVICE FOR USER
ONEOFFPURCHASESFREQUENCY: PURCHASES FREQUENCY IN ONEGO	CASHADVANCETRX: NO OF TRANSACTIONS MADE WITH CASH IN ADVANCE

Fig1: Brief description of each attribute in the credit card dataset.

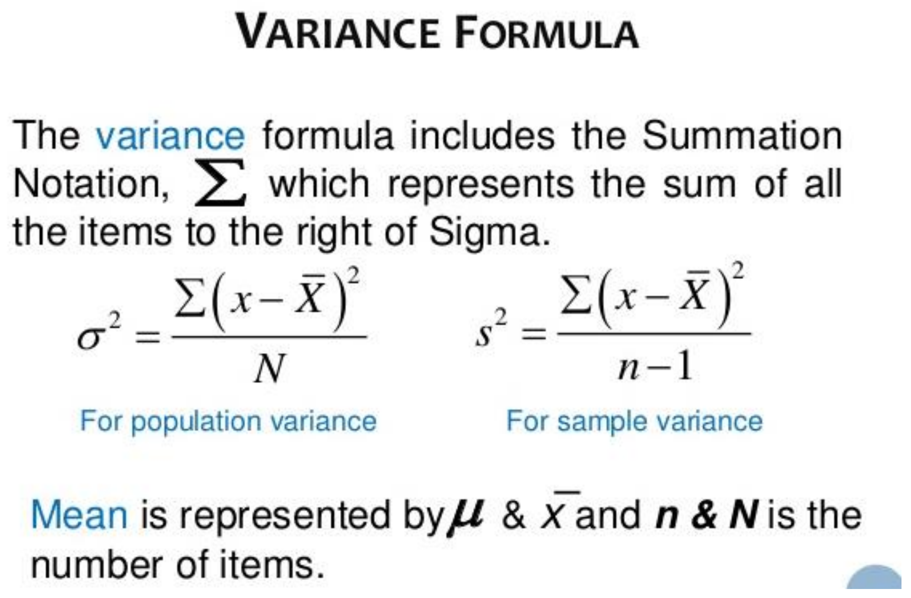
4. PRE-PROCESSING:

Credit card dataset has different types of transactional values are in various ranges, when all the attributes are considered, due to different ranges in values it very hard to understand the complete range of points.



CUST_ID	BALANCE	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	TENURE	outcome
C10001	40.900749	95.4	0	95.4	0	2	1000	201.802084	139.509787	12	0
C10002	3202.467416	0	0	0	6443	0	7000	4103.032597	1072.340217	12	1
C10003	2495.148862	773.17	773.17	0	0	12	7500	622.066742	627.284787	12	0
C10004	1666.670542	1499	1499	0	206	1	7500	0		12	1
C10005	817.714335	16	16	0	0	1	1200	678.334763	244.791237	12	0
C10006	1809.828751	1333.28	0	1333.28	0	8	1800	1400.05777	2407.246035	12	0
C10007	627.260806	7091.01	6402.63	688.38	0	64	13500	6354.314328	198.065894	12	0
C10008	1823.652743	436.2	0	436.2	0	12	2300	679.065082	532.03399	12	0
C10009	1014.926473	861.49	661.49	200	0	5	7000	688.278568	311.963409	12	0
C10010	152.225975	1281.6	1281.6	0	0	3	11000	1164.770591	100.302262	12	0
C10011	1293.124939	920.12	0	920.12	0	12	1200	1083.301007	2172.697765	12	0

When the outliers present in the data the calculated range can be incorrect and it can create problems in future output, we have a statical approach called variance to show the valid range to consider.



VARIANCE FORMULA

The variance formula includes the Summation Notation, $\sum$ which represents the sum of all the items to the right of Sigma.

$$\sigma^2 = \frac{\Sigma(x - \bar{X})^2}{N} \qquad s^2 = \frac{\Sigma(x - \bar{X})^2}{n-1}$$

For population variance — For sample variance

Mean is represented by μ & $\bar{x}$ and ***n* & *N*** is the number of items.

Note: The values of both sides of the mean are taken which is negative and positive side and then squared in the next step so that when we calculate both side of the values it should not be zeroed, and this step is very important in the statical calculation as variance will be playing a major role in the statical calculations and finally, variance is calculated.

These outliers are ignored.

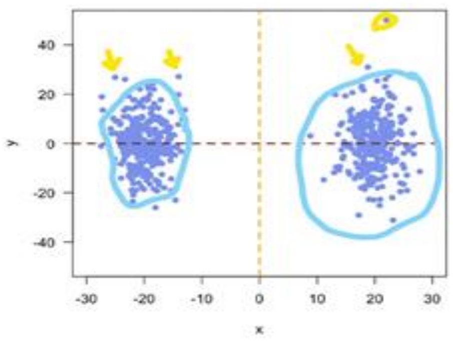
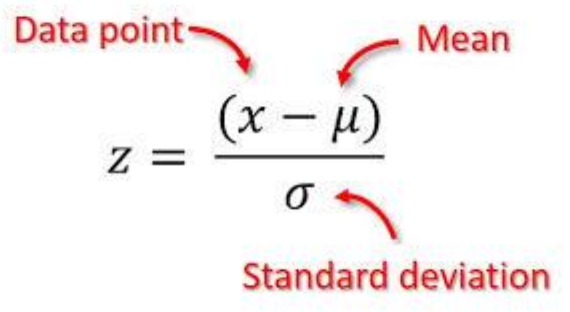


Figure 2: Clusters with outliers

To detect these outliers, we majorly have two famous techniques that are IQR and z -score. We are using z-score method in this assignment to find outliers which are not necessary for our dataset and then only consider certain values with the help of z score.

In Normal distribution (Gaussian distribution) we have a bell curve where mean is the centre element and the standard deviation on both sides to be $\pm$ 1 decreasing gradually from mean, in this concept very good empirical formula with respect to gaussian distribution is that with in the first standard deviation there will be around 68% of the data of the total distribution and similarly for second and third deviation we will have 95% and 99% of the overall data. This concept will be used to convert the data into a standard normal distribution by applying a very simple formula



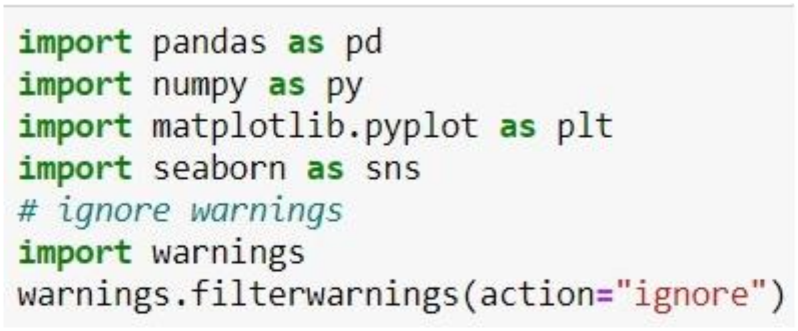
$$z = \frac{(x - \mu)}{\sigma}$$

Though this we would know maximum amount of the data lies in between 3rd standard deviation, hence z value will be taken as 3. By using these methods, we get the new dataset which has a good amount of range to apply our methodology.

These are the basic cleaning steps carried out for both models before we implement our learns.

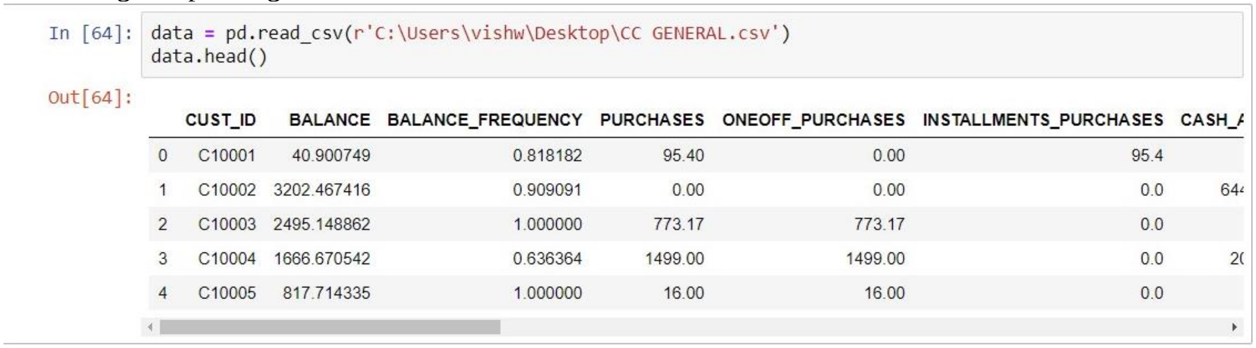
5.Data Cleansing:

1. Importing required libraries and filtering warnings.



```
import pandas as pd
import numpy as py
import matplotlib.pyplot as plt
import seaborn as sns
# ignore warnings
import warnings
warnings.filterwarnings(action="ignore")
```

2. Loading and printing data.

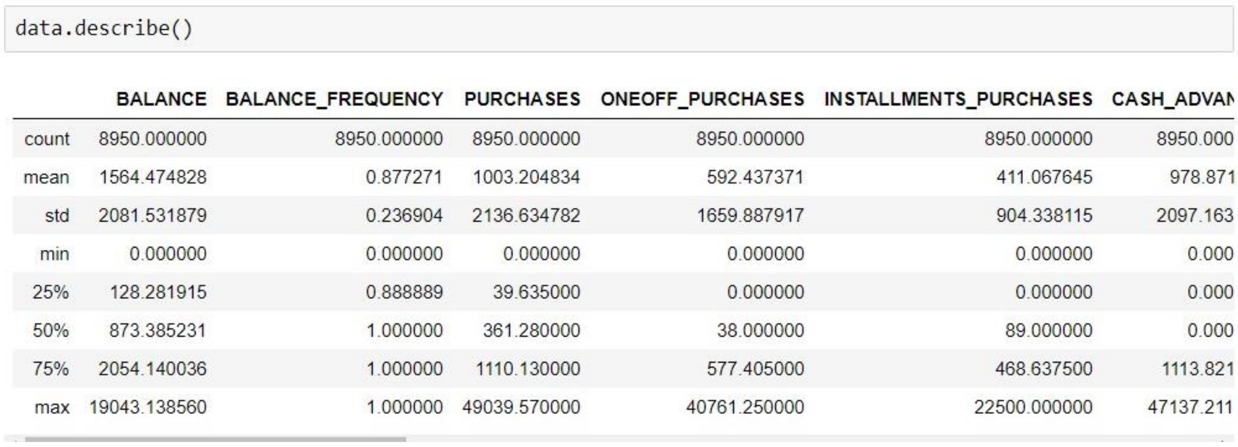


```
In [64]: data = pd.read_csv(r'C:\Users\vishw\Desktop\CC GENERAL.csv')
         data.head()
```

Out[64]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_
0	C10001	40.900749	0.818182	95.40	0.00	95.4	
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	644
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	20
4	C10005	817.714335	1.000000	16.00	16.00	0.0	

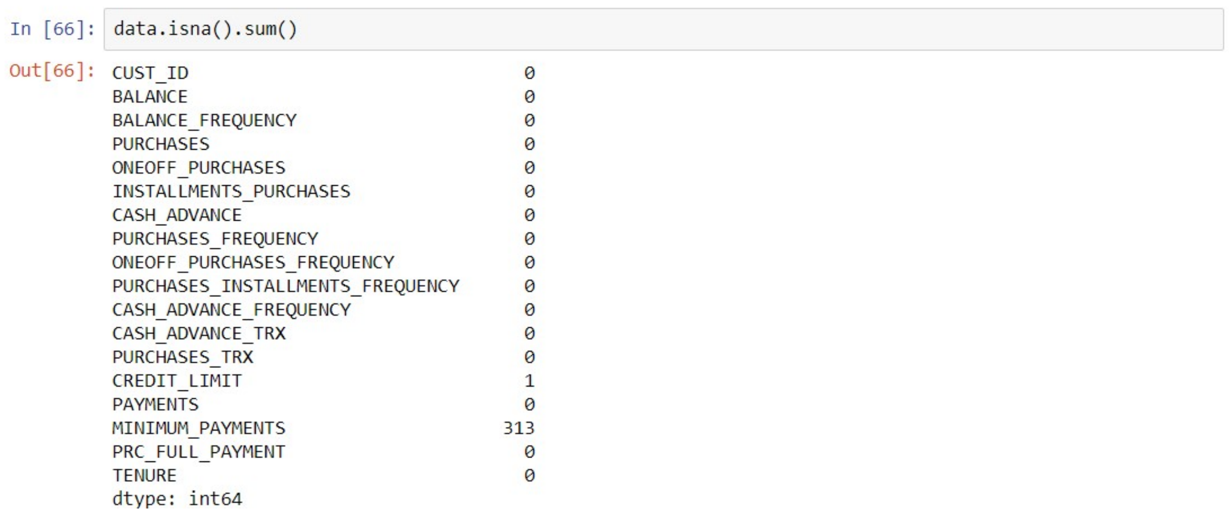
3. Describing data to get mean, standard deviation, minimum and maximum values in each column, for better understanding of data to perform cleansing and pre-processing.



```
data.describe()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVAN
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000
mean	1564.474828	0.877271	1003.204834	592.437371	411.067645	978.871
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	2097.163
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	128.281915	0.888889	39.635000	0.000000	0.000000	0.000
50%	873.385231	1.000000	361.280000	38.000000	89.000000	0.000
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	1113.821
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	47137.211

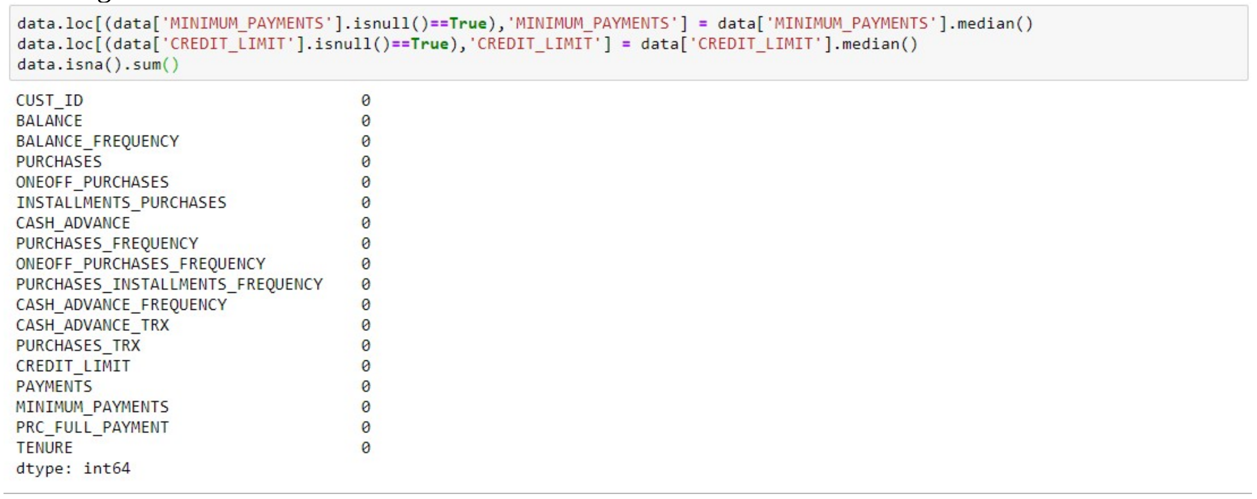
4. Finding Null values in each column.



```
In [66]: data.isna().sum()

Out[66]: 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
```

Output: We can see that there are 313 null values in minimum payments and 1 null value in credit limit. So, we will fill these null values with median value of that column so that the variance and standard deviation of the data will not change.

5. Filling null values with median values.

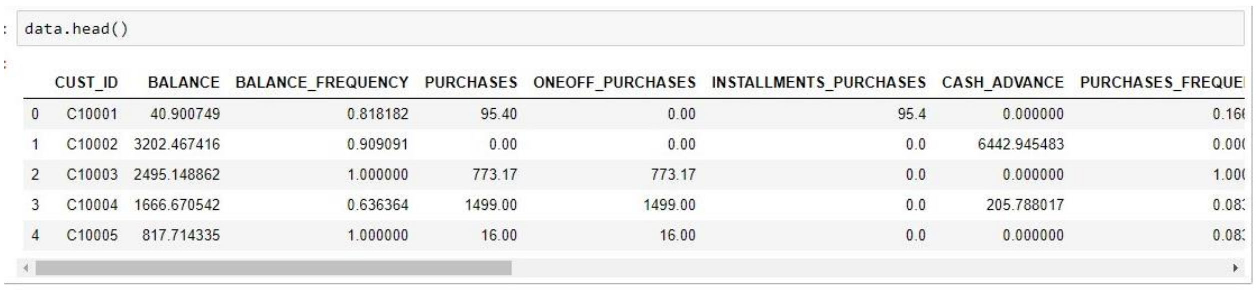


```
data.loc[(data['MINIMUM_PAYMENTS'].isnull()==True),'MINIMUM_PAYMENTS'] = data['MINIMUM_PAYMENTS'].median()
data.loc[(data['CREDIT_LIMIT'].isnull()==True),'CREDIT_LIMIT'] = data['CREDIT_LIMIT'].median()
data.isna().sum()
```

```
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
```

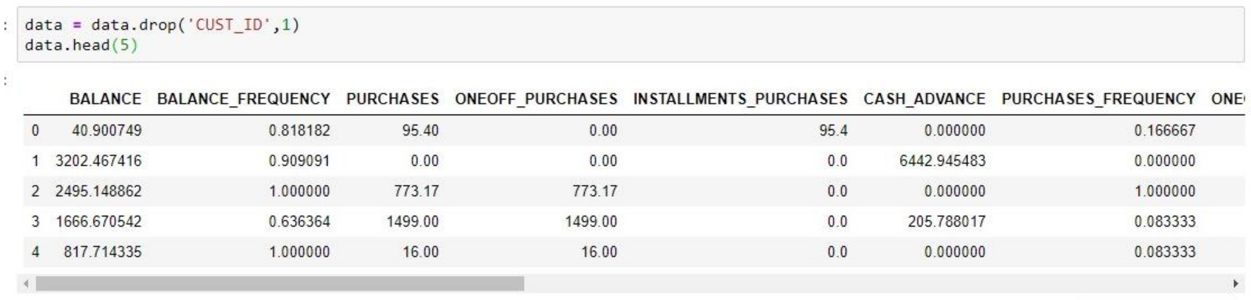
Output: From the output we can see that there are no further null values.

6. From the dataset we can see that Customer Id is a unique entity which if no use so removing the first column.



```
data.head()
```

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUE
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.16
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.00
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.00
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.08
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.08



```
data = data.drop('CUST_ID',1)
data.head(5)
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONE
0	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166667	
1	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000000	
2	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000000	
3	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083333	
4	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083333	

These are the basic cleaning steps carried out for both models before we implement our learns.

6.CLUSTERING

Clustering is a method of grouping of similar characteristics in set of objects, in such a way we can find out the objects in the same groups (cluster). Clustering is used in many fields including pattern recognition, image analysis, machine learning etc.

Clustering can be done in many ways like measuring distance between the central nodes, thickness areas of the data spaces in statistical distribution. Clustering methods can be of many types such as partitioning methods, hierarchical method, density-based method and many more.

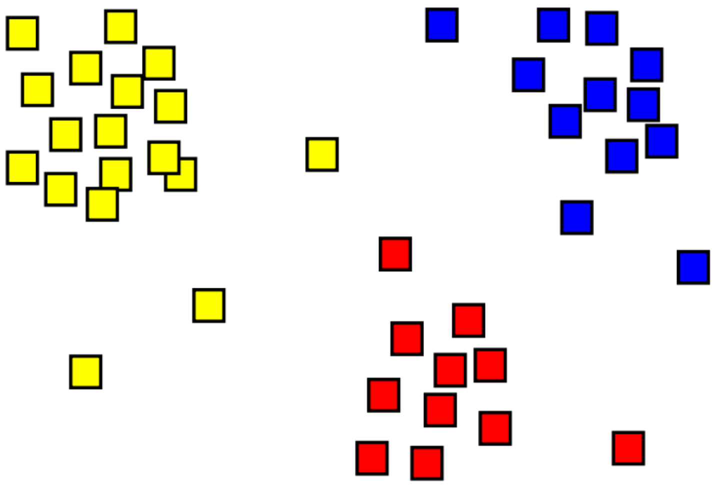


Figure 3: Example for clusters formed using nearest centroids

Centroid-based clustering

K-means clustering is an important partitioning method used in unsupervised algorithms, it tries to group similar items in the form of clusters. The number of groups are represented by K, it brings the similarity between the items and group them into the clusters.

This works in three steps

- Choose k values
- Initialize the centroids

```
Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
Repeat until convergence: {
    For every $i$, set
        $c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$
    For each $j$, set
        $\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$
}
```

- Select the group and find the clusters.

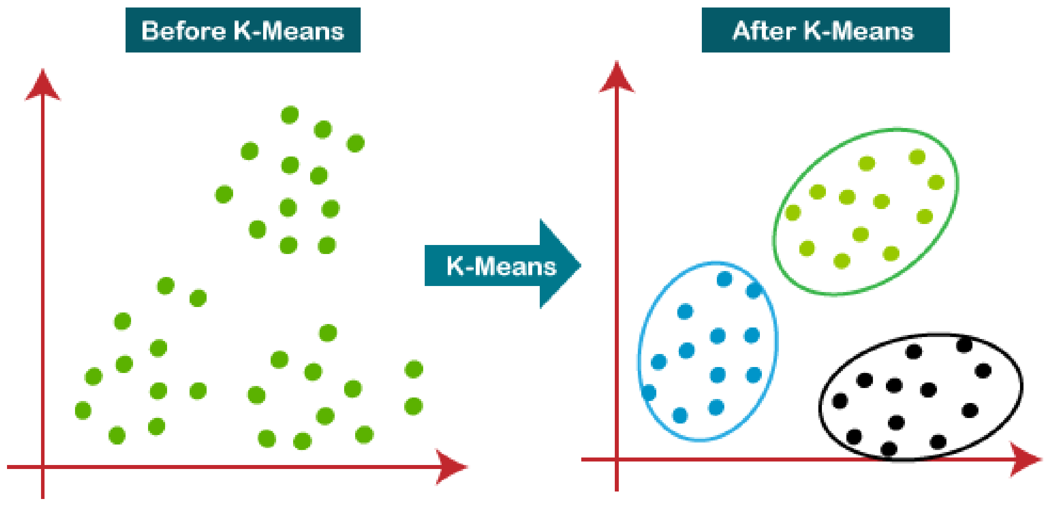


Figure 4: Example of clusters formed using K-means

Advantages of K-means

1. We can implement it easily.
2. It is faster to large datasets.
3. Abstraction of clusters for different shapes and sizes.

Disadvantages of K-means

1. It is reactive to the oddity.
2. Finding k values manually is a difficult job.
3. As the number of features increases its flexibility will downturn.

Customer Segmentation

Customer segmentation is the one of the most common technique used in clustering. It is widely used in banking, e-commerce, sports, advertising, sales etc. Customer Segmentation is the class of a market into variety customer groups that share same characteristics. Customer Segmentation is sensitive which means to find unsatisfied customer needs. Most of the business sectors needs customer information which are **Demographic information, Geographical information, Psychographics and Behavioral data.**.

Finding K-Value

First thing to do in K- means clustering is to find the ideal k-number of clusters. The maximum possibility of K in K-means clustering is total number of observations which might not be the ideal value for K. So, to determine this value we will be using a popular method known as Elbow method. To implement this method, we will plot a graph between different number of clusters and compare their within-cluster sum of squares (WCSS).

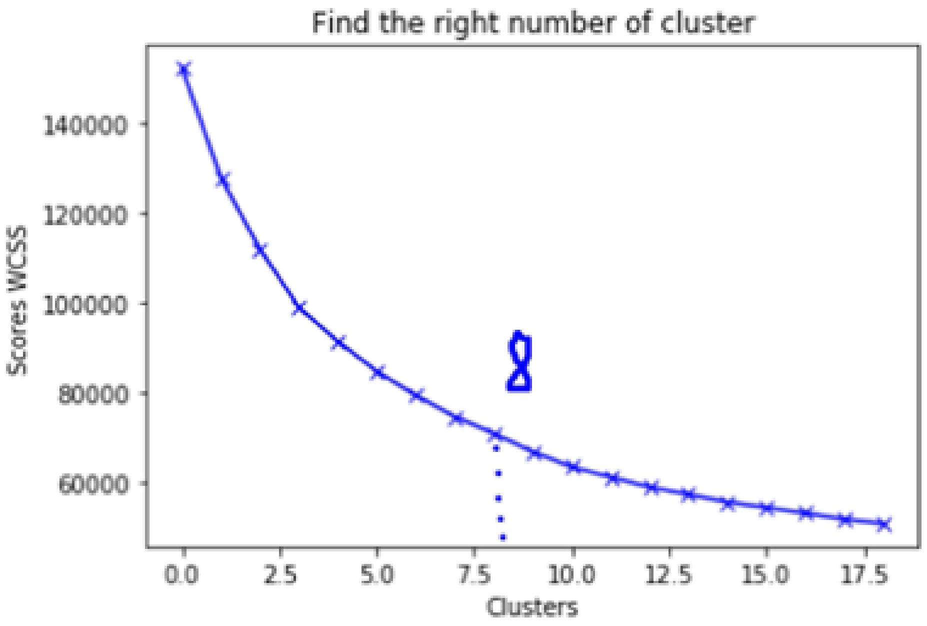


Figure 5: Finding K value using elbow method.

From the above example figure we can see WCSS vary with number of clusters. We have to select the value of k at the "elbow" to determine the optimal number of clusters, i.e., the point at which the distortion/inertia begins to decrease in a linear way. Thus, the ideal number of clusters is 8.

Principal Component Analysis (PCA), which is a **linear dimensionality reduction** method that can be exploited for separate knowledge from a high-dimensional space by pointed it into a lower-dimensional sub-space. It tries to uphold the needed parts that have more difference of the data and remove the non-essential parts with less difference. One important thing to note about PCA is that it is an **Unsupervised** dimensionality reduction technique, you can cluster the similar data points based on the feature correlation between them without any supervision, PCA is an Unsupervised dimensionality reduction technique, it can group the similar data points depending on the feature interrelationship between them outwardly in any supervision.

Steps involved in finding Principal Components:

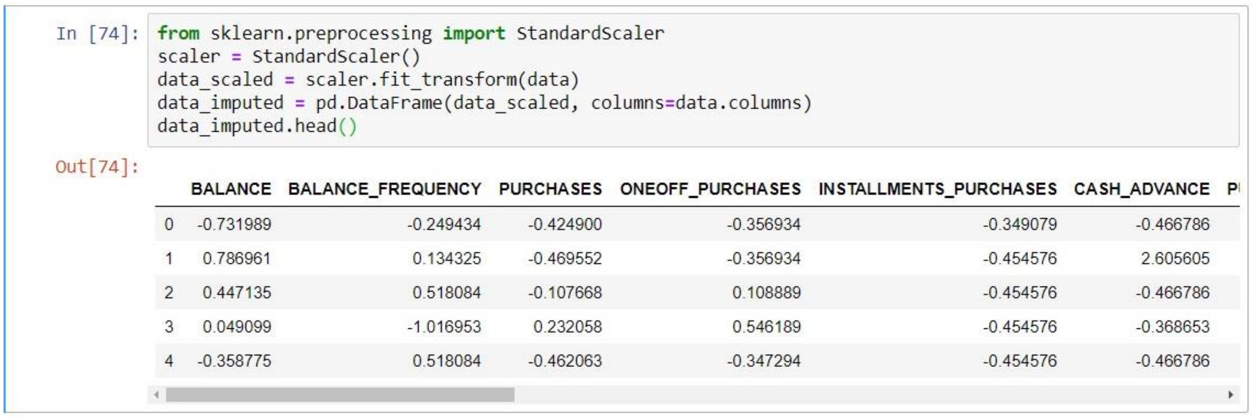
- Step 1: Standardization
- Step 2: Covariance matrix computation

- ➢ Step 3: Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components

6.1. Implementation:

As we have already pre-processed data by finding missing values and replacing them with median values. We can now implement desired algorithms.

1. The data has values ranging from 0 to 20,000 .So, for better visualization we woild like to normalize the data.

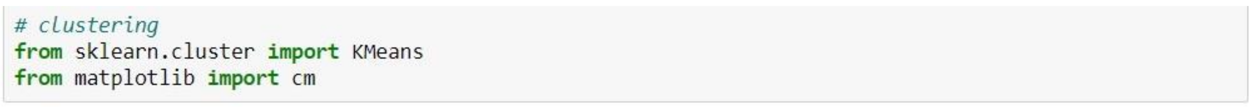


```
In [74]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         data_scaled = scaler.fit_transform(data)
         data_imputed = pd.DataFrame(data_scaled, columns=data.columns)
         data_imputed.head()
```

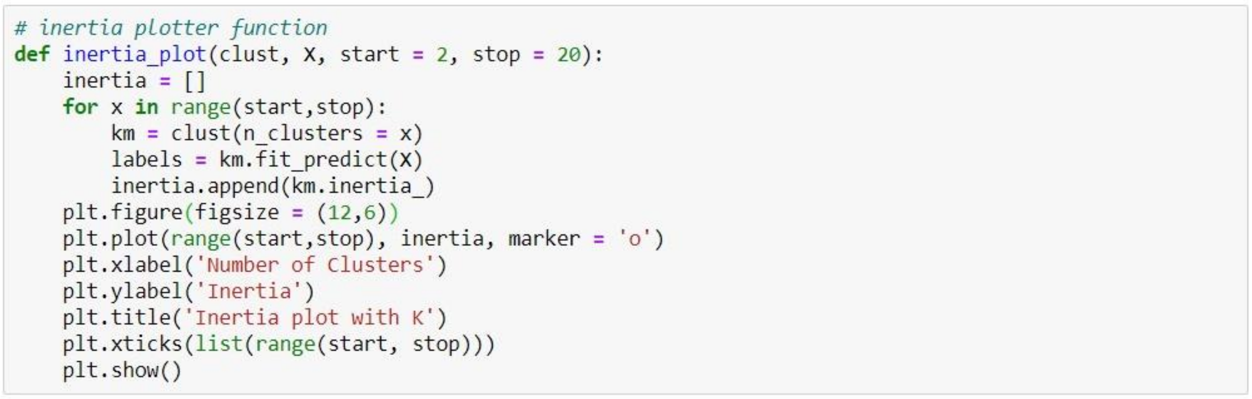
Out[74]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE
0	-0.731989	-0.249434	-0.424900	-0.356934	-0.349079	-0.466786
1	0.786961	0.134325	-0.469552	-0.356934	-0.454576	2.605605
2	0.447135	0.518084	-0.107668	0.108889	-0.454576	-0.466786
3	0.049099	-1.016953	0.232058	0.546189	-0.454576	-0.368653
4	-0.358775	0.518084	-0.462063	-0.347294	-0.454576	-0.466786

2. Importing K-Means from Sklearn and using matplotlib for visualization. In this step we would like to find the K -Value using Elbow method.



```
# clustering
from sklearn.cluster import KMeans
from matplotlib import cm
```



```
# inertia plotter function
def inertia_plot(clust, X, start = 2, stop = 20):
    inertia = []
    for x in range(start,stop):
        km = clust(n_clusters = x)
        labels = km.fit_predict(X)
        inertia.append(km.inertia_)
    plt.figure(figsize = (12,6))
    plt.plot(range(start,stop), inertia, marker = 'o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.title('Inertia plot with K')
    plt.xticks(list(range(start, stop)))
    plt.show()
```

Output: *inertia plot (KMeans, data_imputed)*

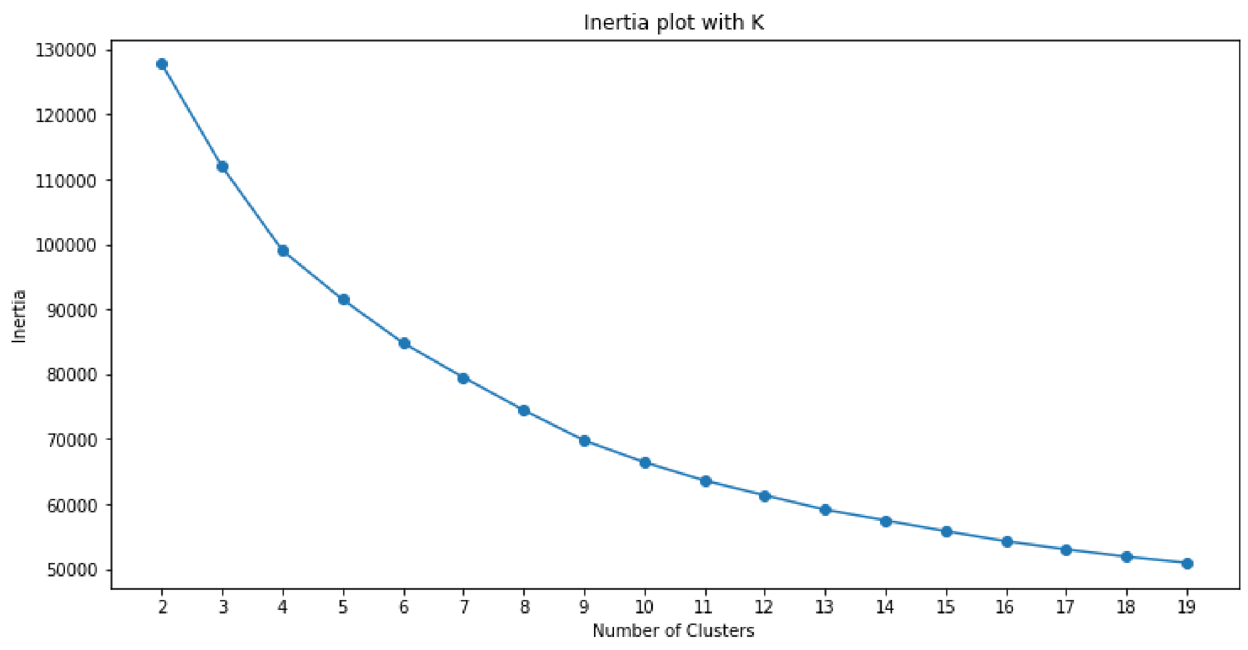
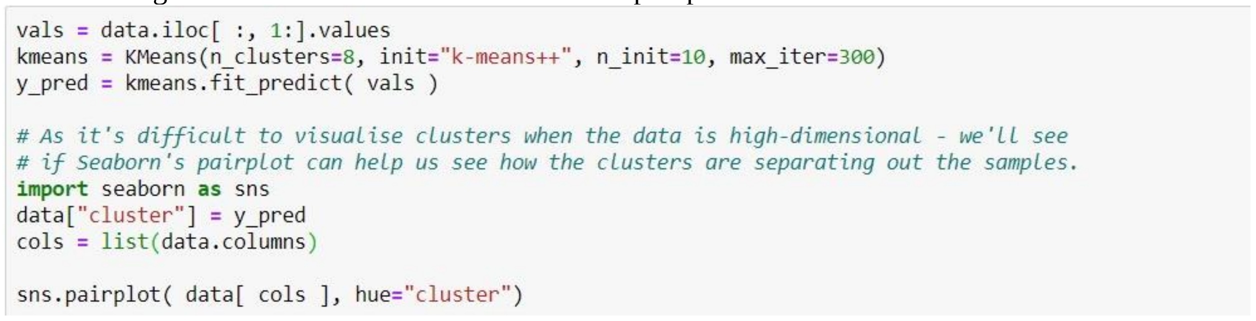


Figure 6: Finding K value using elbow method

From this inertia plot we can see that there is constant change in inertia from 6 clusters. So, we would like to continue 6 as our K value.

3. Visualizing 6 clusters vs all dimensions in seaborn pair plot.



```
vals = data.iloc[ :, 1:].values
kmeans = KMeans(n_clusters=8, init="k-means++", n_init=10, max_iter=300)
y_pred = kmeans.fit_predict( vals )

# As it's difficult to visualise clusters when the data is high-dimensional - we'll see
# if Seaborn's pairplot can help us see how the clusters are separating out the samples.
import seaborn as sns
data["cluster"] = y_pred
cols = list(data.columns)

sns.pairplot( data[ cols ], hue="cluster")
```

Now, this complete data is being segmented into 6 clusters, which means visualizing 6 clusters across all the 17 columns gives us the following output.

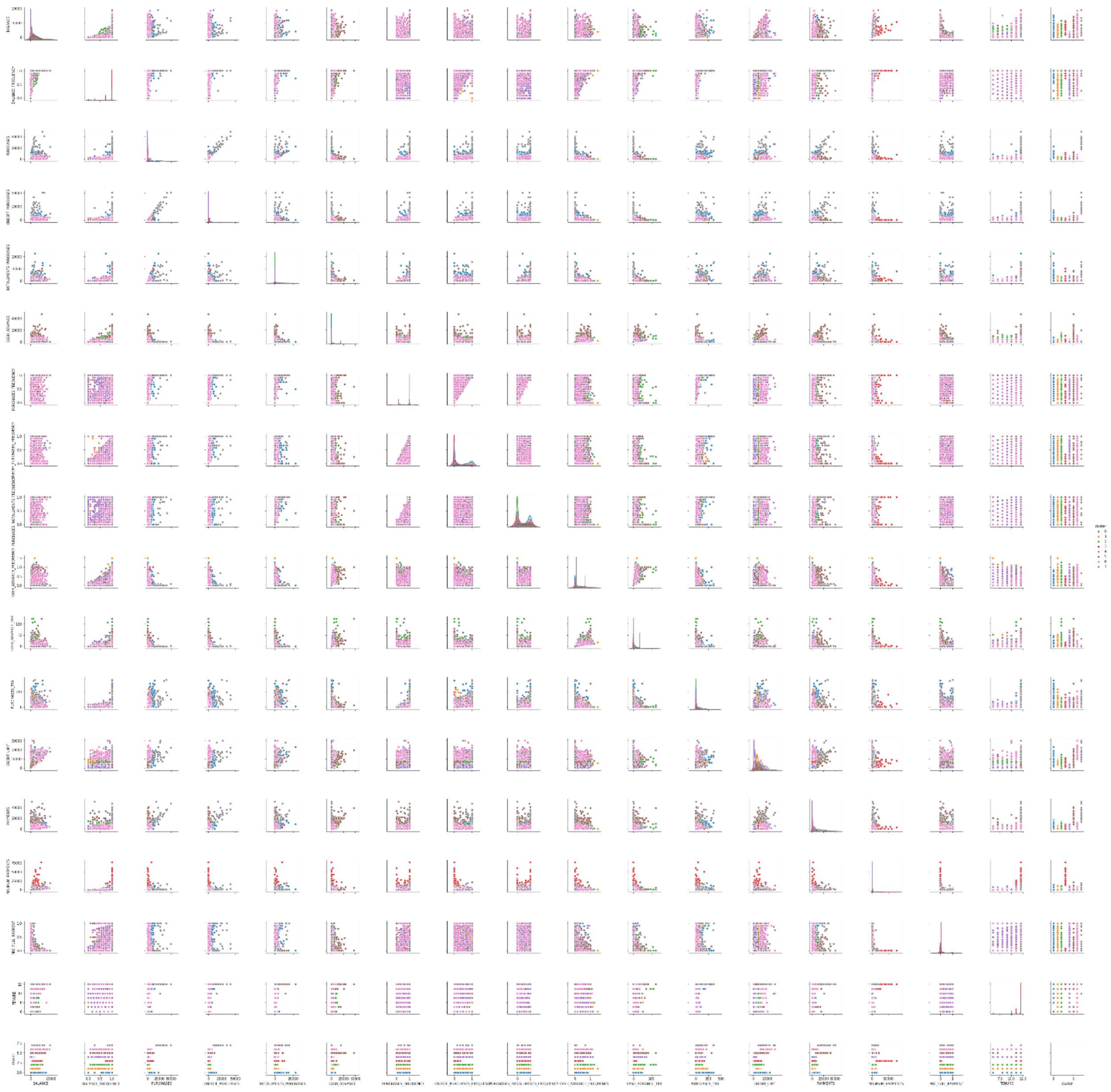
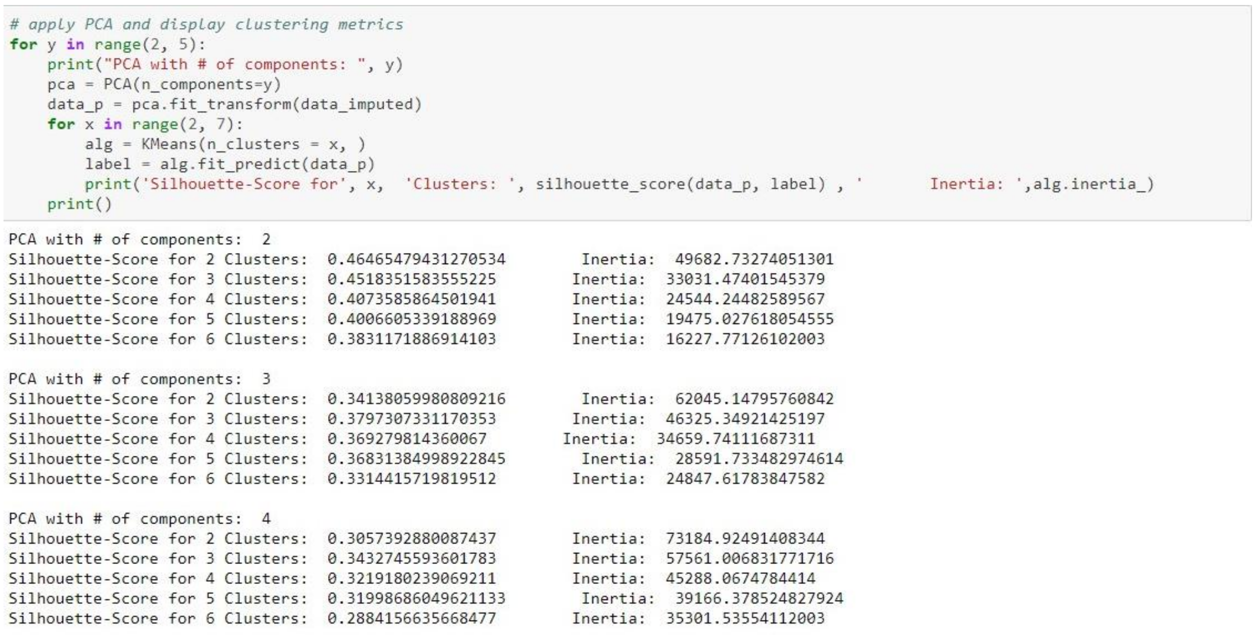


Figure 7: Visualizations of clusters formed against all attributes.

4. As we cannot draw any relations or information about clusters from the above pair-plot, we would like to perform dimensionality reduction using Principal Component Analysis.

Now, before performing PCA we need to find the ideal number of components for dimensionality reduction which can be obtained by finding silhouette scores and inertia for different number of clusters (between 2 and 6) with 2,3 and 4 components.

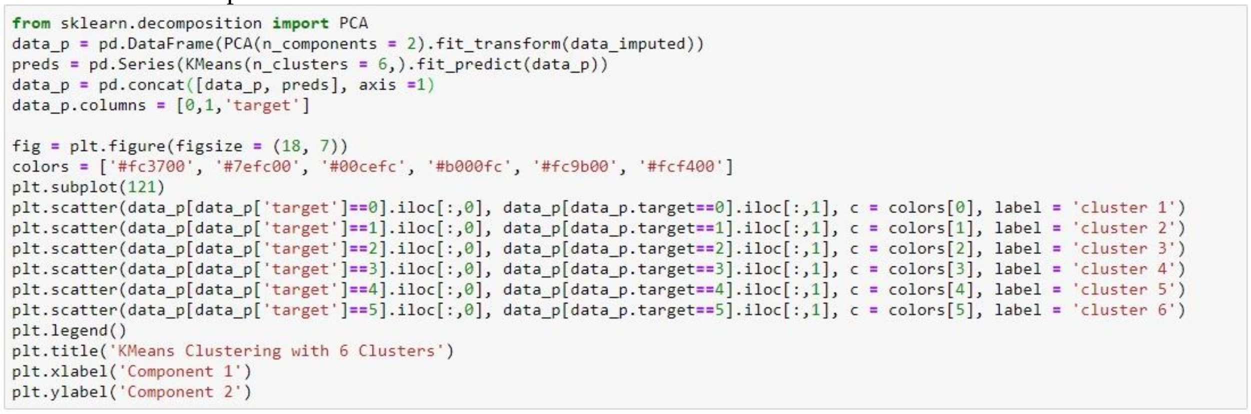


```
# apply PCA and display clustering metrics
for y in range(2, 5):
    print("PCA with # of components: ", y)
    pca = PCA(n_components=y)
    data_p = pca.fit_transform(data_imputed)
    for x in range(2, 7):
        alg = KMeans(n_clusters = x, )
        label = alg.fit_predict(data_p)
        print('Silhouette-Score for', x,  'Clusters: ', silhouette_score(data_p, label) , '       Inertia: ',alg.inertia_)
    print()
```

```
PCA with # of components:  2
Silhouette-Score for 2 Clusters:  0.46465479431270534        Inertia:  49682.73274051301
Silhouette-Score for 3 Clusters:  0.4518351583555225        Inertia:  33031.47401545379
Silhouette-Score for 4 Clusters:  0.4073585864501941        Inertia:  24544.24482589567
Silhouette-Score for 5 Clusters:  0.4006605339188969        Inertia:  19475.027618054555
Silhouette-Score for 6 Clusters:  0.3831171886914103        Inertia:  16227.77126102003

PCA with # of components:  3
Silhouette-Score for 2 Clusters:  0.34138059980809216        Inertia:  62045.14795760842
Silhouette-Score for 3 Clusters:  0.3797307331170353        Inertia:  46325.34921425197
Silhouette-Score for 4 Clusters:  0.369279814360067        Inertia:  34659.74111687311
Silhouette-Score for 5 Clusters:  0.36831384998922845        Inertia:  28591.733482974614
Silhouette-Score for 6 Clusters:  0.3314415719819512        Inertia:  24847.61783847582

PCA with # of components:  4
Silhouette-Score for 2 Clusters:  0.3057392880087437        Inertia:  73184.92491408344
Silhouette-Score for 3 Clusters:  0.3432745593601783        Inertia:  57561.006831771716
Silhouette-Score for 4 Clusters:  0.3219180239069211        Inertia:  45288.0674784414
Silhouette-Score for 5 Clusters:  0.31998686049621133        Inertia:  39166.378524827924
Silhouette-Score for 6 Clusters:  0.2884156635668477        Inertia:  35301.53554112003
```

Output: PCA with 2 components for 6 Clusters is ideal, from the above output. Because it is understood that Silhouette scores are high, and Inertia is less for this combination.

5. PCA with 2 components for 6 Clusters.



```
from sklearn.decomposition import PCA
data_p = pd.DataFrame(PCA(n_components = 2).fit_transform(data_imputed))
preds = pd.Series(KMeans(n_clusters = 6,).fit_predict(data_p))
data_p = pd.concat([data_p, preds], axis =1)
data_p.columns = [0,1,'target']

fig = plt.figure(figsize = (18, 7))
colors = ['#fc3700', '#7efc00', '#00cefc', '#b000fc', '#fc9b00', '#fcf400']
plt.subplot(121)
plt.scatter(data_p[data_p['target']==0].iloc[:,0], data_p[data_p.target==0].iloc[:,1], c = colors[0], label = 'cluster 1')
plt.scatter(data_p[data_p['target']==1].iloc[:,0], data_p[data_p.target==1].iloc[:,1], c = colors[1], label = 'cluster 2')
plt.scatter(data_p[data_p['target']==2].iloc[:,0], data_p[data_p.target==2].iloc[:,1], c = colors[2], label = 'cluster 3')
plt.scatter(data_p[data_p['target']==3].iloc[:,0], data_p[data_p.target==3].iloc[:,1], c = colors[3], label = 'cluster 4')
plt.scatter(data_p[data_p['target']==4].iloc[:,0], data_p[data_p.target==4].iloc[:,1], c = colors[4], label = 'cluster 5')
plt.scatter(data_p[data_p['target']==5].iloc[:,0], data_p[data_p.target==5].iloc[:,1], c = colors[5], label = 'cluster 6')
plt.legend()
plt.title('KMeans Clustering with 6 Clusters')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
```

Output:

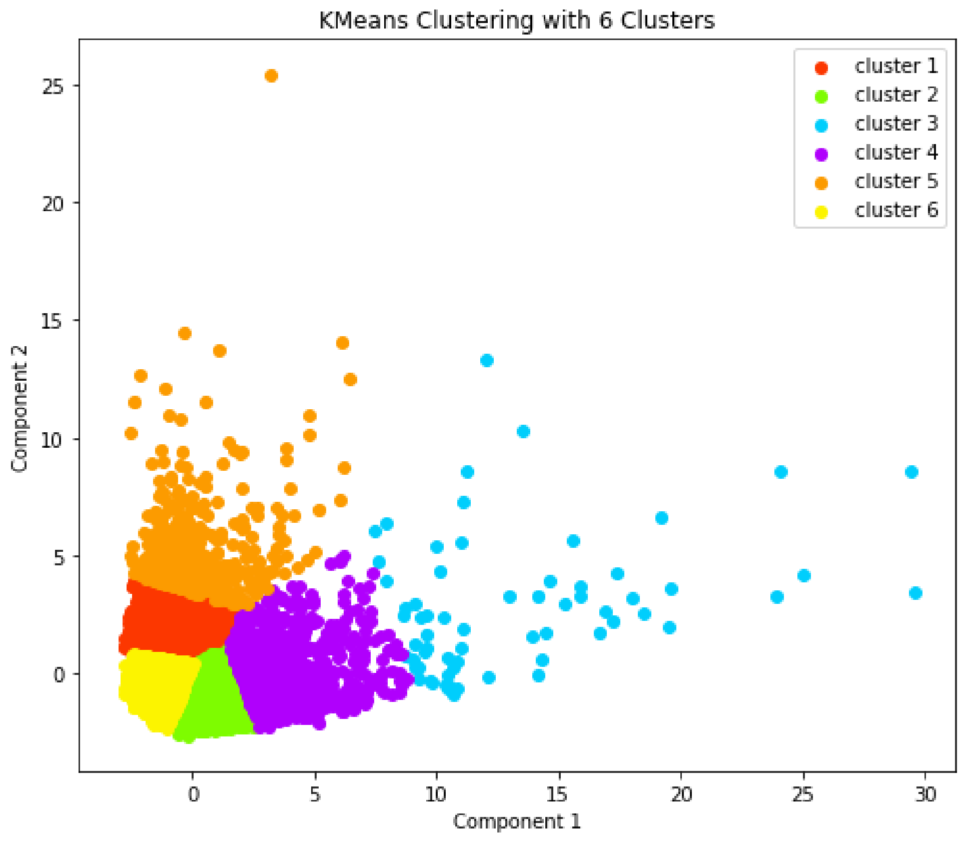
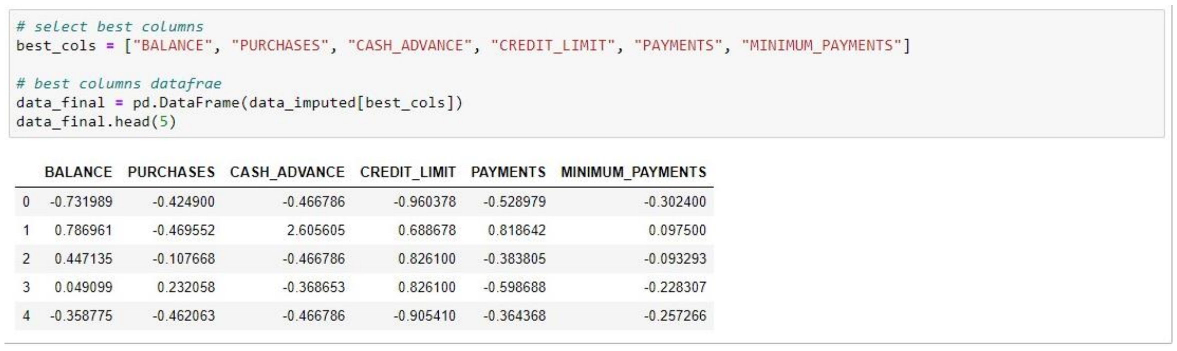


Figure 8: Visualising all 6 clusters in single graph plot using dimensionality reduction

PCA reduces all the dimensions to 2 components, using which we can visualize the 6 clusters.

6.2. Output

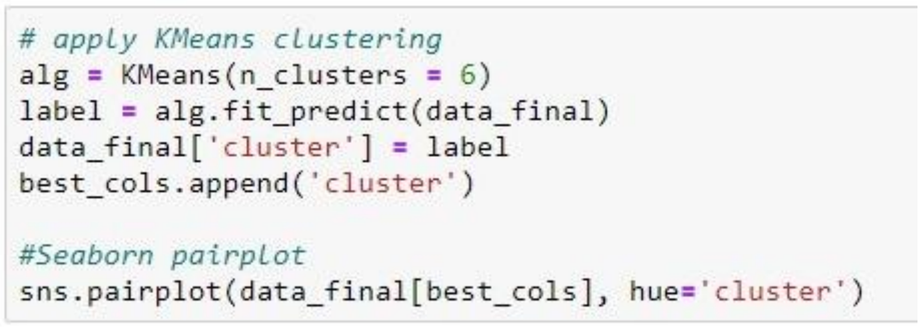
STEP 1: Firstly, we are going to consider main 6 columns to discuss the data analysis, which are "BALANCE", "PURCHASES", "CASH_ADVANCE", "CREDIT_LIMIT", "PAYMENTS", "MINIMUM_PAYMENTS" will be named as best columns. These best columns will bring out behavior of these 6 clusters.



```
# select best columns
best_cols = ["BALANCE", "PURCHASES", "CASH_ADVANCE", "CREDIT_LIMIT", "PAYMENTS", "MINIMUM_PAYMENTS"]

# best columns datafrae
data_final = pd.DataFrame(data_imputed[best_cols])
data_final.head(5)
```

	BALANCE	PURCHASES	CASH_ADVANCE	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS
0	-0.731989	-0.424900	-0.466786	-0.960378	-0.528979	-0.302400
1	0.786961	-0.469552	2.605605	0.688678	0.818642	0.097500
2	0.447135	-0.107668	-0.466786	0.826100	-0.383805	-0.093293
3	0.049099	0.232058	-0.368653	0.826100	-0.598688	-0.228307
4	-0.358775	-0.462063	-0.466786	-0.905410	-0.364368	-0.257266

STEP 2: Applying K-Means and Visualizing these 6 clusters with these best columns using seaborn pair plot.



```
# apply KMeans clustering
alg = KMeans(n_clusters = 6)
label = alg.fit_predict(data_final)
data_final['cluster'] = label
best_cols.append('cluster')

#Seaborn pairplot
sns.pairplot(data_final[best_cols], hue='cluster')
```

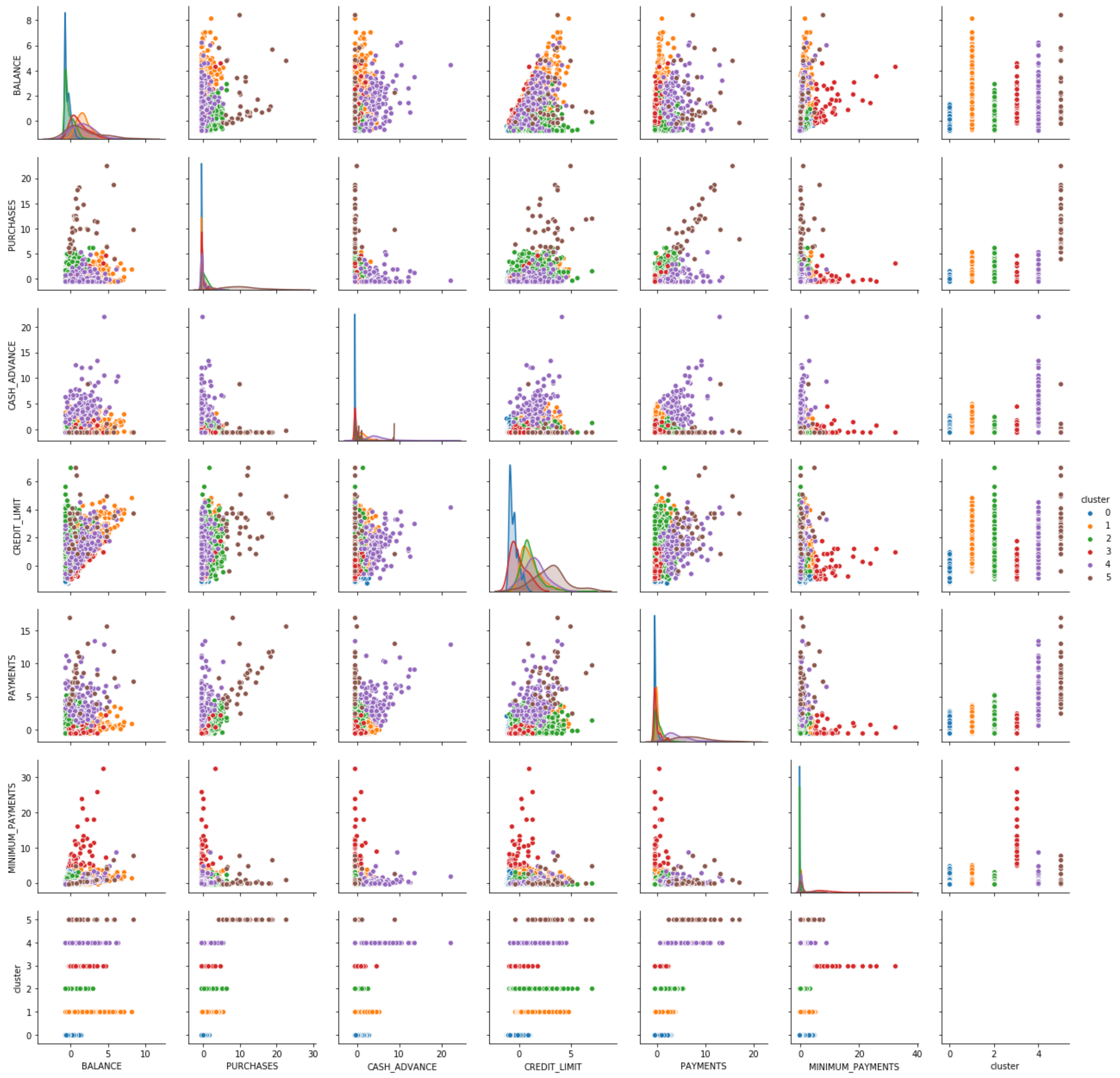


Figure 9: Visualising all 6 K-means clusters using columns, balance, purchases, cash_advance, credit_limit, payments and minimum_payments attributes.

This pair-plot can be used to compare between various columns to understand their nature. Now, let us understand about each cluster.

Step 3: Visualizing selected columns with clusters to understand their characteristics.

CLUSTER 0

```
#analyze all clusters versus Purchases, Payments and Credit Limit
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES', 'PAYMENTS', 'CREDIT_LIMIT'],
            y_vars=['cluster'],
            height=5, aspect=1)
```

Output:

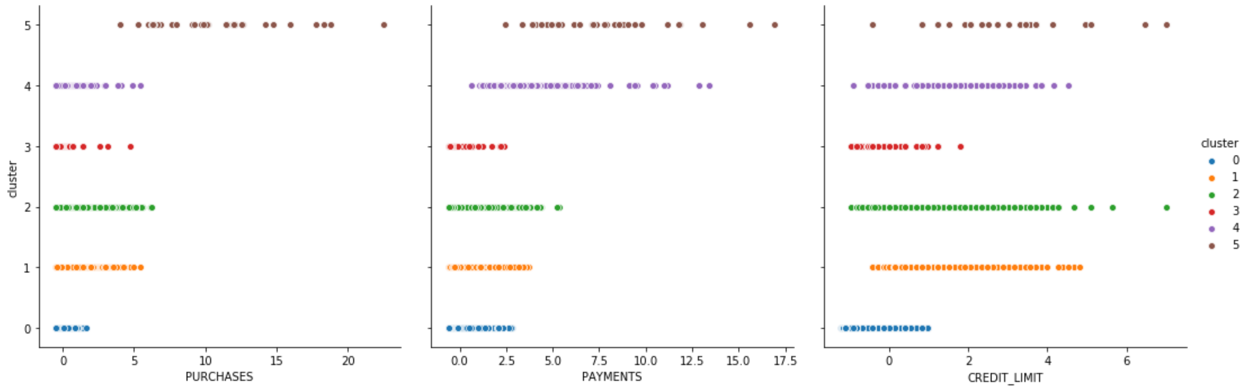


Figure 10: Visualising clusters using purchases, payments and credit_limit attributes.

From the above visualization we can see that customers belonging to cluster 0 have less purchases, low payments and low credit limits as well. This cluster is the biggest group of customers and it also evident from their behavior that they are average users of credit card.

CLUSTER 1

```
#analyzing Balance and credit limit
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['BALANCE'], y_vars=['CREDIT_LIMIT'],
            height=5, aspect=1)
```

Output

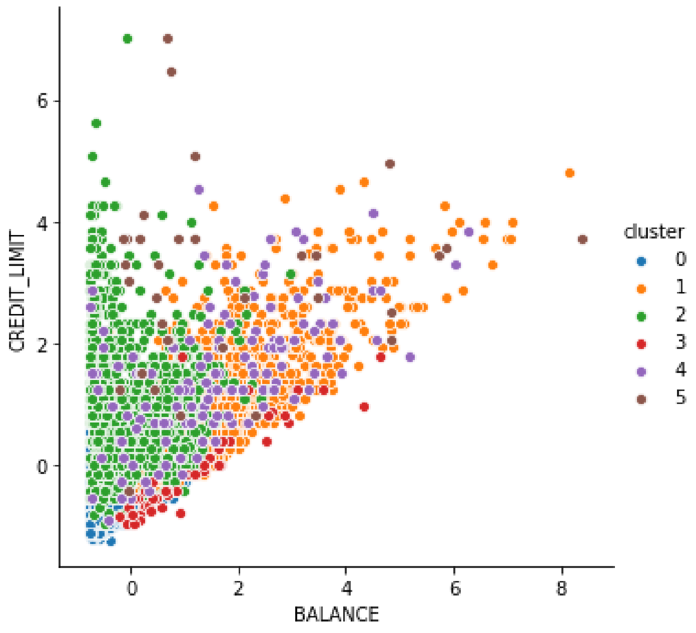


Figure 11: Visualising clusters by plotting balance vs credit_limit

This group is highly difficult to analyze as their balance and credit limit are highly varied. We can consider them as varied customers.

Cluster 2:

```
#analyze all clusters versus Purchases, Payments and Credit Limit
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES', 'PAYMENTS', 'CREDIT_LIMIT'],
             y_vars=['cluster'],
             height=5, aspect=1)
```

Output:

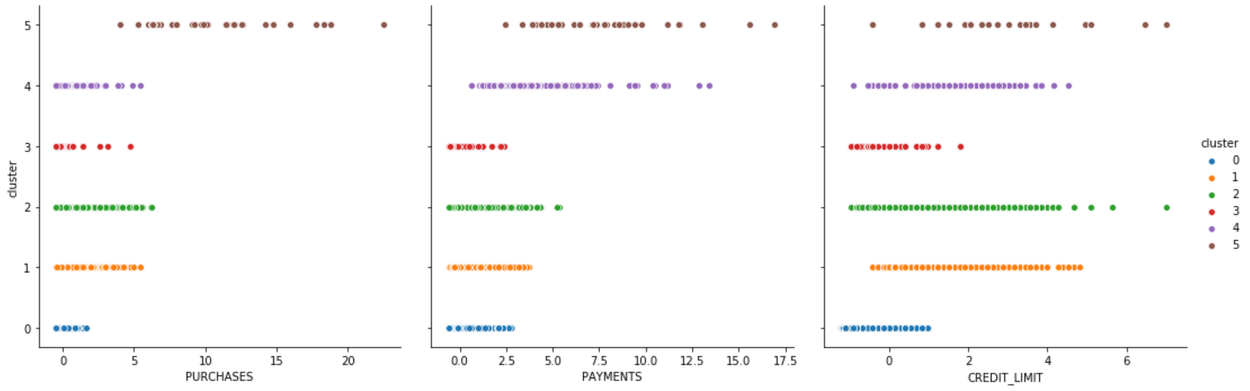


Figure 12: Visualising clusters using purchases, payments and credit_limit attributes.

This group of customers have high purchases, high payments and good credit limit, this group can be called as active users as they maintain all dimensions perfectly.

Cluster 3:

```
#Analyzing credit limit vs minimum payments.
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['MINIMUM_PAYMENTS'], y_vars=['CREDIT_LIMIT'],
             height=5, aspect=1)
```

Output:

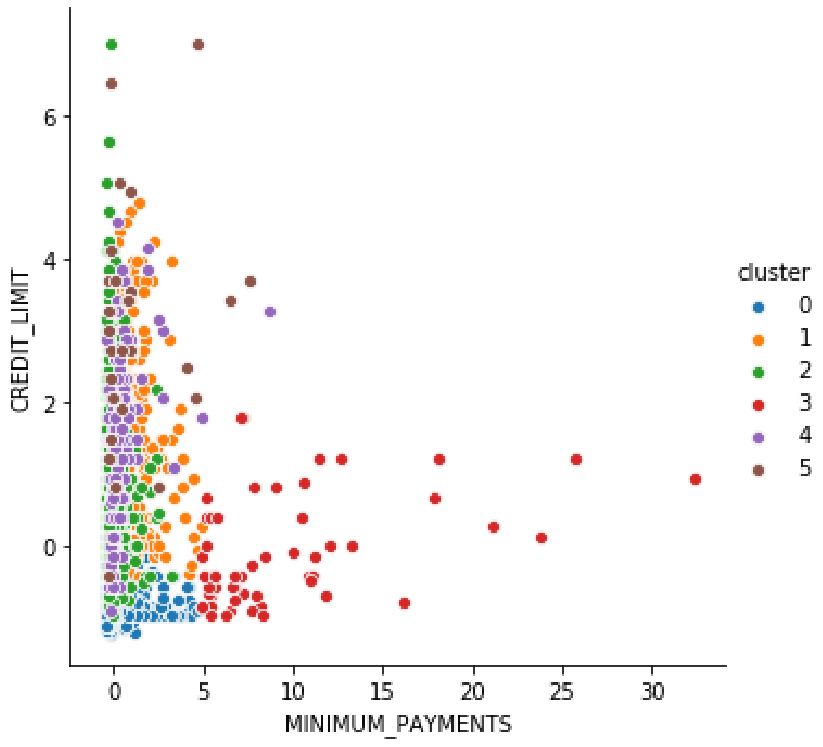


Figure 13: Visualising clusters by plotting minimun_payments vs credit_limit

Cluster 3 group of customers have high minimum payments and low credit limits, these are of high-risk category to the banks as there is a chance of not repaying the credit.

Cluster 4:

```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['BALANCE', 'CASH_ADVANCE', 'PAYMENTS'],
             y_vars=['cluster'],
             height=5, aspect=1)
```

Output:

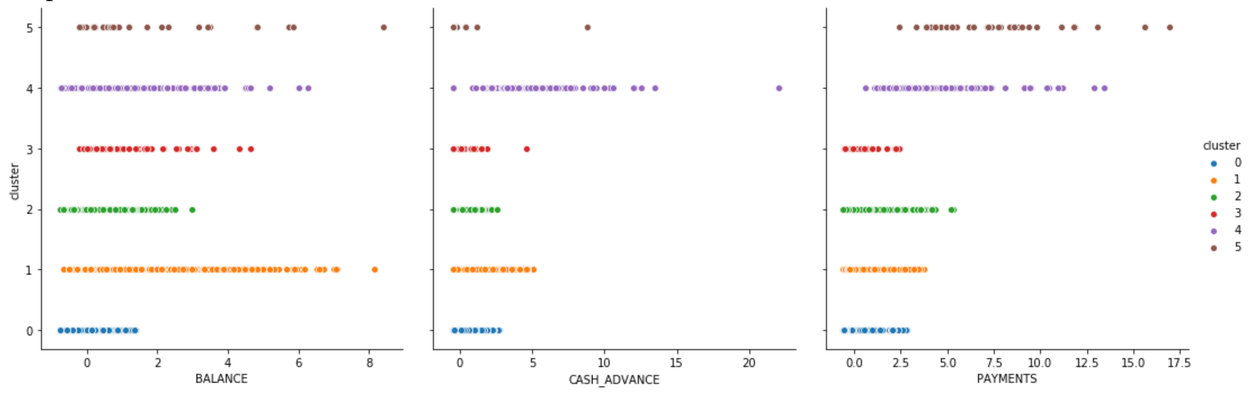


Figure 14: Visualising clusters using balance, cash_advance and payments attributes

This cluster of customers have high payments, average purchases and varied balances but a strange case is that they have very high cash advances so we can term them as Money Borrowers.

Cluster 5:

```
#analyze all clusters versus Purchases, Payments and Credit Limit
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES', 'PAYMENTS', 'CREDIT_LIMIT'],
             y_vars=['cluster'],
             height=5, aspect=1)
```

Output:

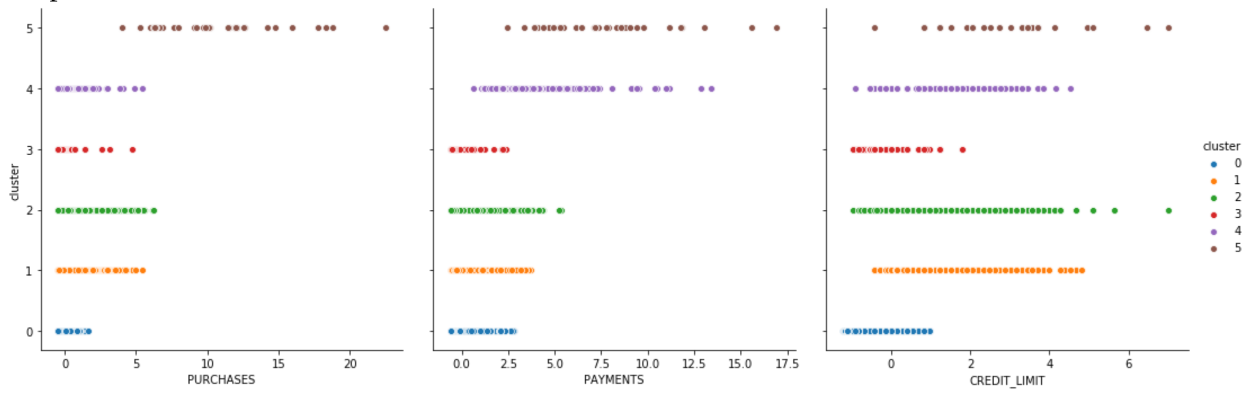


Figure 15: Visualising clusters using purchases, payments and credit_limit attributes

This group of users have highest number of purchases, highest payments, highest minimum payments saying that they can be heavy spenders on credit card. There are several other varied features also in this cluster, let us look at their behavior on payments and purchases.

```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES'], y_vars=['PAYMENTS'],
             height=5, aspect=1)
```

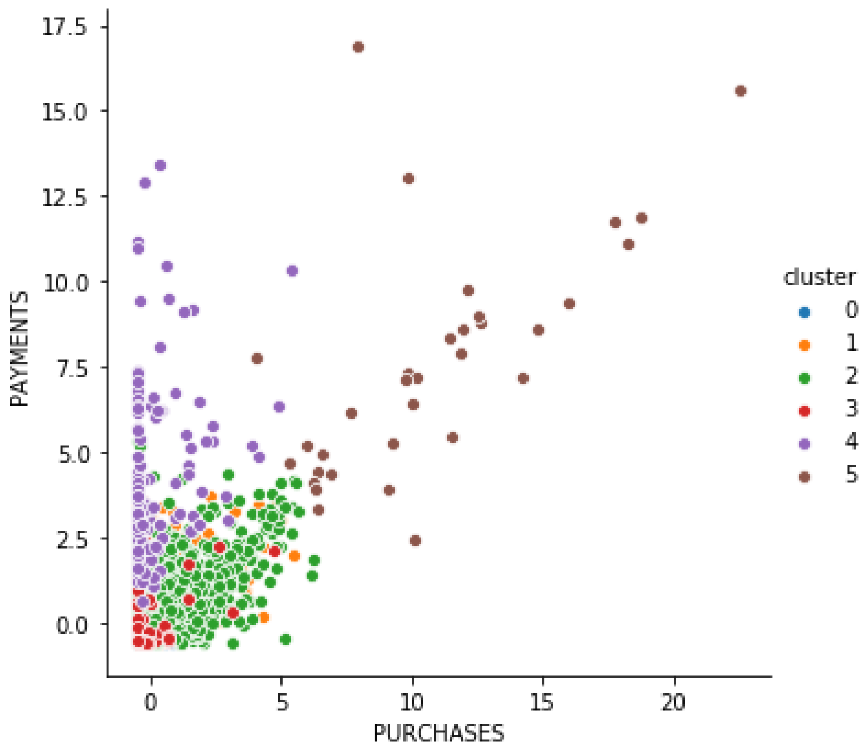


Figure 16: Visualising clusters by plotting purchases vs payments.

Naturally big spenders are users who maintain high balances, purchases, cash advances and payments. The above graph demonstrates how outlier heavy this graph is compared to the rest of the dataset.

From the above outcomes we can divide all the customers based on their usage as Average Users, Active Users, High Risk Users, Money Borrowers, Varied Customers and Big Customers.

7.CLASSIFICATION

In Classification we are using decision tree algorithm to predict the future customer cluster with all the attributes considered and without applying clustering methods again and again, to achieve this we are taking the output of clustering and appended as a class column to apply decision tree to predict future customers clusters.

Classification using decision Tree

A Decision Tree algorithm is part of a family of supervised learning algorithms. Unlike the other supervised learning algorithms, it can also be used to solve regression problems and classification problems.

Reason for using the Decision Tree is to build a training model that can predict the value or class of targeted variables from learning decision rules concluding from the previous results (training data).

The decision tree is detailed diagram-like a tree structure as where internal node represents a feature (or attribute), the branch shows us the decision rule, and each of the leaf node displays the result. The uppermost node is called as root note node in decision tree. It learns partitioning based on the value of the attributes. It divides tree recursively by calling recursive partitioning.

The flowchart design allows us to decide. It's visualization as a flowchart model easily mimics thought of the human level, which makes decision trees easier to understand.

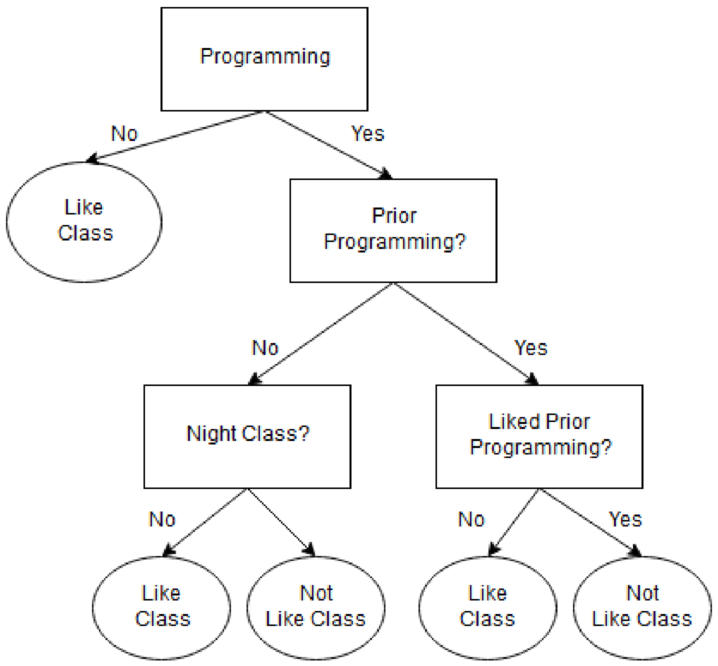


Figure 17: Decision tree flow chart

Working of a decision tree algorithm

1. Select the best attribute that breaks the records using the Attribute Selection Measures (ASM).
2. Ensure to make the decision node attribute and split into a smaller dataset.
3. Then start building a tree by repeating the same method recursively for each child until if any meets following conditions:
   - All the tuples belong to the attribute of the same value.
   - No attributes available.
   - No more cases like this.

Information gain

Entropy

A Top-down Decision Tree from a root node requires partitioning data into subsets containing instances with alike values (homogenous). In order to measure a homogeneity sample, the ID3 algorithm uses entropy. If a sample is completely homogeneous, entropy is 0, and it has the entropy of 1 if the samples are equally divided.

We must quantify two forms of entropy in order to create a decision-tree:

a) The Entropy for one attribute's frequency table:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

b) The Entropy for Multiple attribute's frequency table:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

Information Gain

The information gain is an entropy reduction. It measures the difference between entropy before division and average entropy after division of the dataset based on certain attribute values. The decision tree algorithm ID3 (Iterative Dichotomiser) uses information gain.

Information gain is represented mathematically as:

$$\text{Information Gain}(T,X) = \text{Entropy}(T) - \text{Entropy}(T, X)$$

In an easier way we conclude as:

$$Information\ Gain\ =\ Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

Information Gain:

If "before" is the pre-division dataset, K is the number that the break generates, and (j, after) sub-division j is after division.

To search the best feature which serves as the root node for info gain, each of the descriptive features is used first and the dataset is divided according to the values of these descriptive features. This gives us the residual entropy until the data set has been broken into the feature values. This value is subtracted from the initially measured data entropy to see if this separation of the function decreases the initial entropy, which provides the information gain for a feature and is calculated as:

$$InformationGain(feature) = Entropy(Dataset) - Entropy(feature)$$

The function with the greatest info gain should be used as the root node for starting to construct a decision tree.

For the construction of the decision tree, the ID3 algorithm uses the info gain

Gini Index: The sum of the squared probabilities in each class is determined by extracting from one. Its favour's bigger partitions and is simple to deploy, while info gain favor's smaller partitions with different values.

$$Gini\ Index = 1-\sum(P(x=k))^2$$

A lower Gini index function is selected to separate it.

For the decision tree construction, the classic CART algorithm uses the Gini Index.

End notes

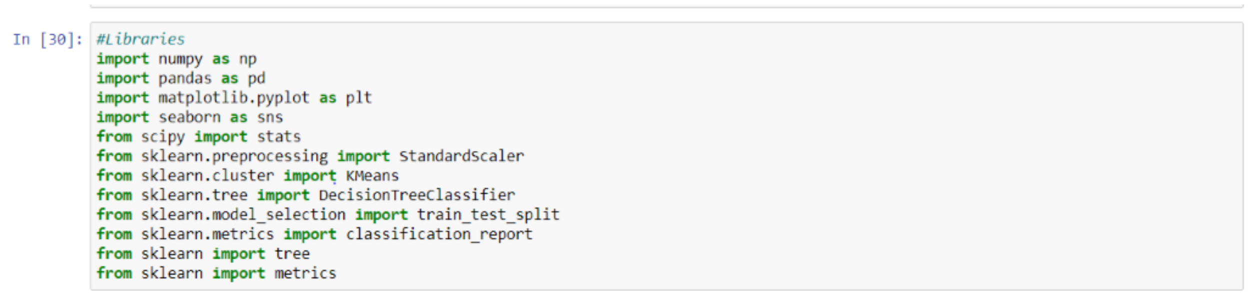
Information is a measure of uncertainty reduction. It is the estimated amount of information necessary to put a new instance in a certain class. These knowledge measurements form the basis for all algorithms of the decision tree. When we use Entropy as the basic measure for information gain, we have broader results than the Gini index caps

7.1. Implementation:

Credit card data set is purely customer transaction data which describes the customer behaviour of their account. In the data set each attribute gives the information of different segments in the customer transaction for an example highest purchased amount and its frequency describe the customer usage of credit card limit.

For this type of data sometimes attributes might not be associated with one another, and they are independent of class variable. Customers are grouped based on different attributes to form clusters then they are further classified with help of decision tree.

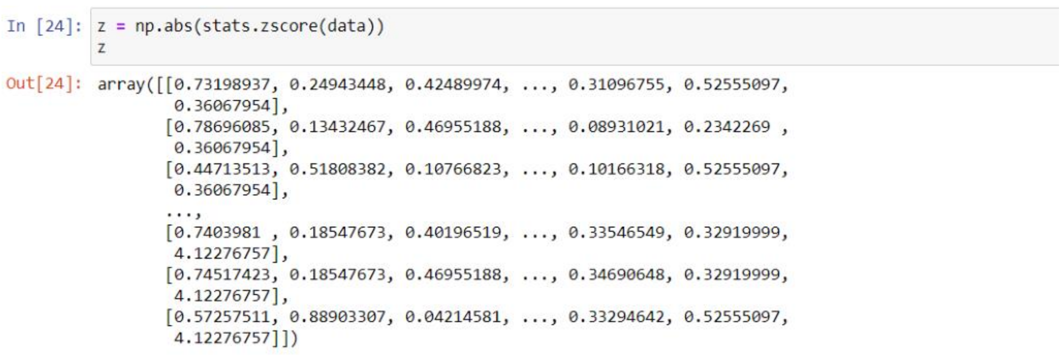
Importing all the necessary libraries - at first, we have used NumPy, pandas and Mat plot to import data, Arithmetic operations and for visualization.



```
In [30]: #Libraries
         import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         from scipy import stats
         from sklearn.preprocessing import StandardScaler
         from sklearn.cluster import KMeans
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import classification_report
         from sklearn import tree
         from sklearn import metrics
```

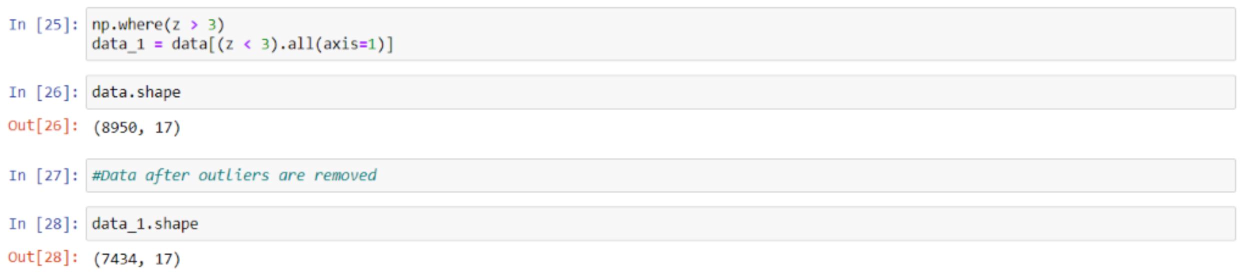
As we have already pre-processed data by replacing missing values with median values, we can now implement the desired algorithms.

Finding out z-scores for data



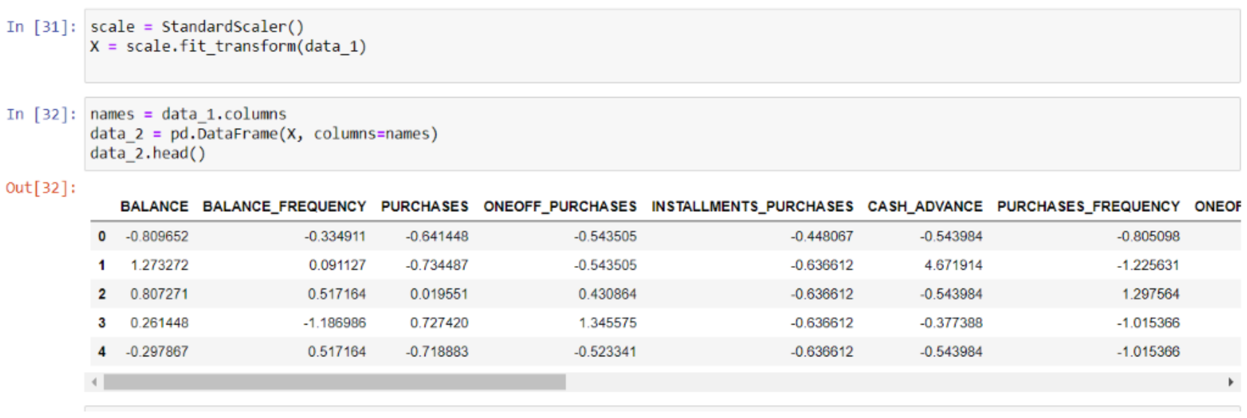
```
In [24]: z = np.abs(stats.zscore(data))
         z

Out[24]: array([[0.73198937, 0.24943448, 0.42489974, ..., 0.31096755, 0.52555097,
                 0.36067954],
                [0.78696085, 0.13432467, 0.46955188, ..., 0.08931021, 0.2342269 ,
                 0.36067954],
                [0.44713513, 0.51808382, 0.10766823, ..., 0.10166318, 0.52555097,
                 0.36067954],
                ...,
                [0.7403981 , 0.18547673, 0.40196519, ..., 0.33546549, 0.32919999,
                 4.12276757],
                [0.74517423, 0.18547673, 0.46955188, ..., 0.34690648, 0.32919999,
                 4.12276757],
                [0.57257511, 0.88903307, 0.04214581, ..., 0.33294642, 0.52555097,
                 4.12276757]])
```

Standard deviation is selected not more than 3 so z score value is taken as 3 and created a new data set and named as data_1 and below is the before and after removing outliers from the dataset.



```
In [25]: np.where(z > 3)
         data_1 = data[(z < 3).all(axis=1)]

In [26]: data.shape

Out[26]: (8950, 17)

In [27]: #Data after outliers are removed

In [28]: data_1.shape

Out[28]: (7434, 17)
```

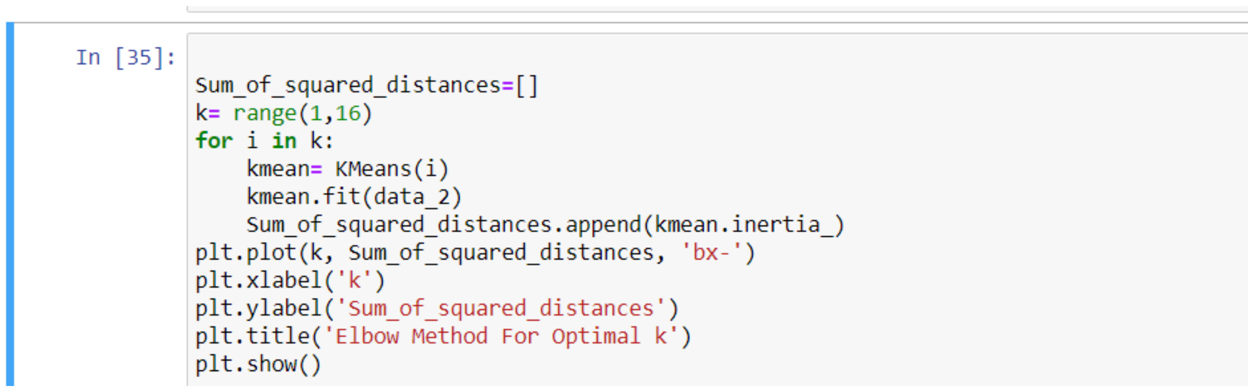
Creating a normalization for the new data set and renamed it as data_2



```
In [31]: scale = StandardScaler()
         X = scale.fit_transform(data_1)

In [32]: names = data_1.columns
         data_2 = pd.DataFrame(X, columns=names)
         data_2.head()

Out[32]:
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOF
0	-0.809652	-0.334911	-0.641448	-0.543505	-0.448067	-0.543984	-0.805098	
1	1.273272	0.091127	-0.734487	-0.543505	-0.636612	4.671914	-1.225631	
2	0.807271	0.517164	0.019551	0.430864	-0.636612	-0.543984	1.297564	
3	0.261448	-1.186986	0.727420	1.345575	-0.636612	-0.377388	-1.015366	
4	-0.297867	0.517164	-0.718883	-0.523341	-0.636612	-0.543984	-1.015366	

Applying elbow method to the dataset to know how many clusters can be formed to it,



```
In [35]:
Sum_of_squared_distances=[]
k= range(1,16)
for i in k:
    kmean= KMeans(i)
    kmean.fit(data_2)
    Sum_of_squared_distances.append(kmean.inertia_)
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()
```

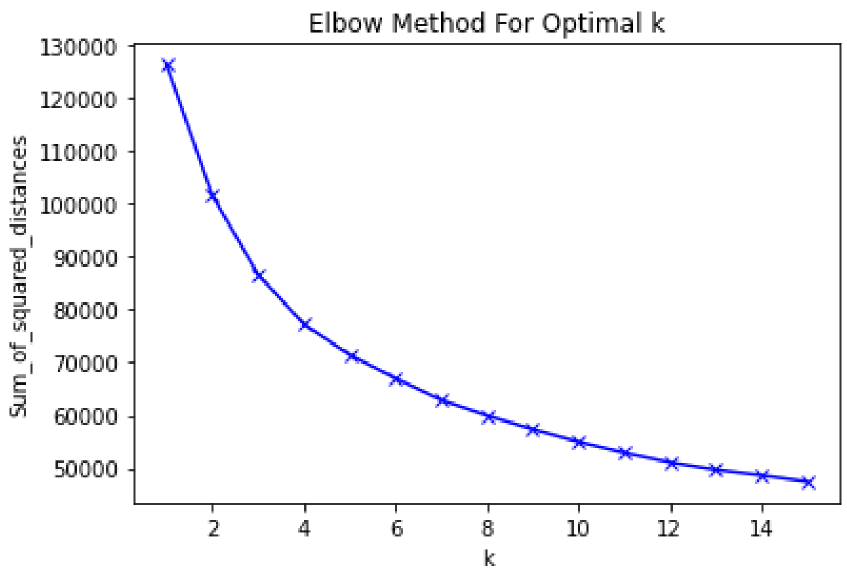
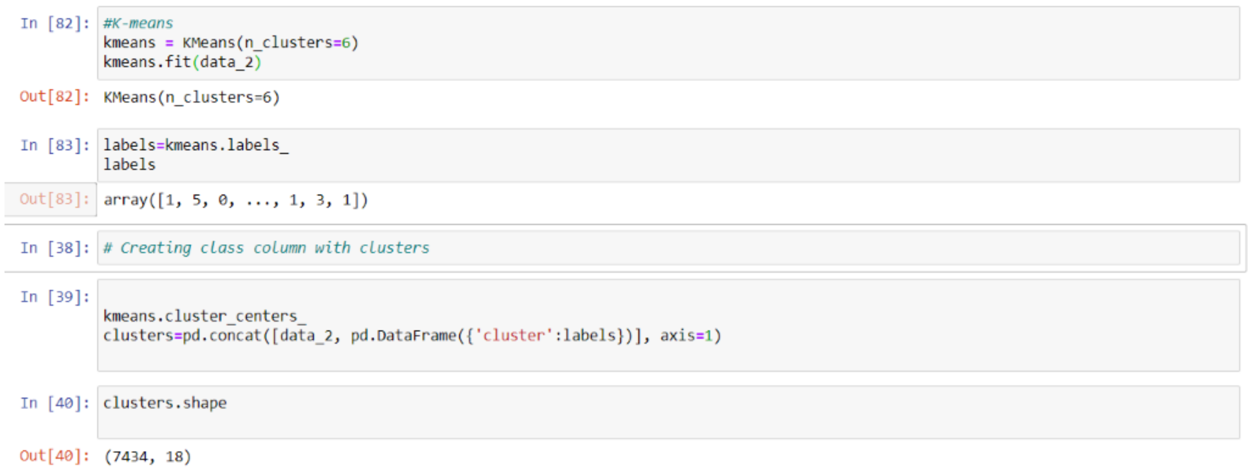


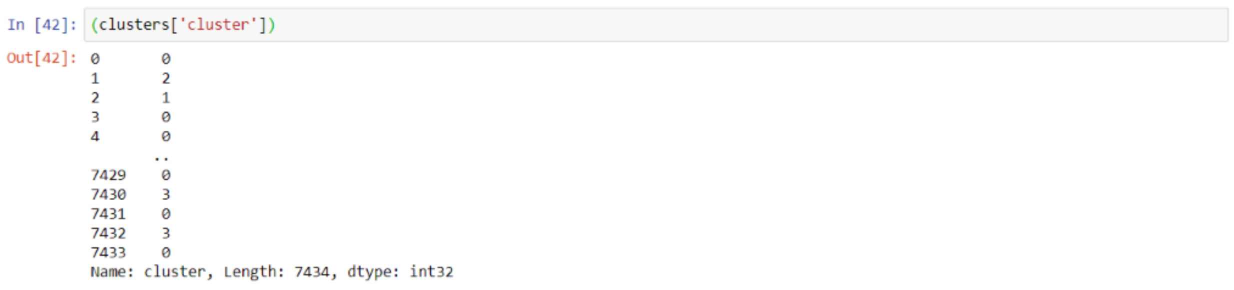
Figure 18: Elbow method showing number of clusters

Data is fit to data_2 and then creating the columns with cluster number as class column.



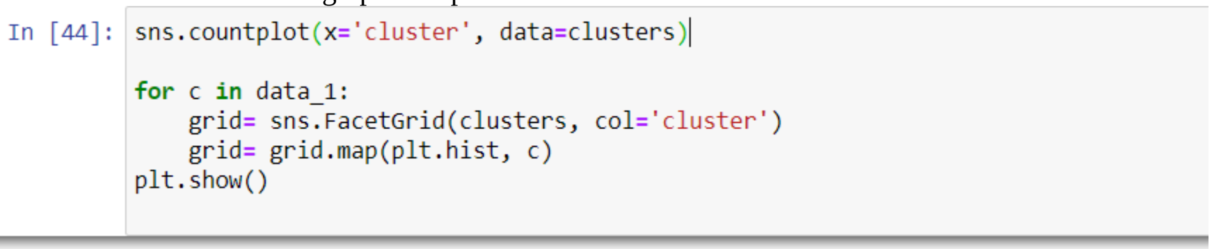
```
In [82]: #K-means
         kmeans = KMeans(n_clusters=6)
         kmeans.fit(data_2)

Out[82]: KMeans(n_clusters=6)

In [83]: labels=kmeans.labels_
         labels

Out[83]: array([1, 5, 0, ..., 1, 3, 1])

In [38]: # Creating class column with clusters

In [39]:
         kmeans.cluster_centers_
         clusters=pd.concat([data_2, pd.DataFrame({'cluster':labels})], axis=1)

In [40]: clusters.shape

Out[40]: (7434, 18)
```

Clusters are re-checked in the dataset



```
In [42]: (clusters['cluster'])
Out[42]: 0       0
         1       2
         2       1
         3       0
         4       0
                ..
         7429    0
         7430    3
         7431    0
         7432    3
         7433    0
         Name: cluster, Length: 7434, dtype: int32
```

Based on the cluster's column graphs are plotted



```
In [44]: sns.countplot(x='cluster', data=clusters)

         for c in data_1:
             grid= sns.FacetGrid(clusters, col='cluster')
             grid= grid.map(plt.hist, c)
         plt.show()
```

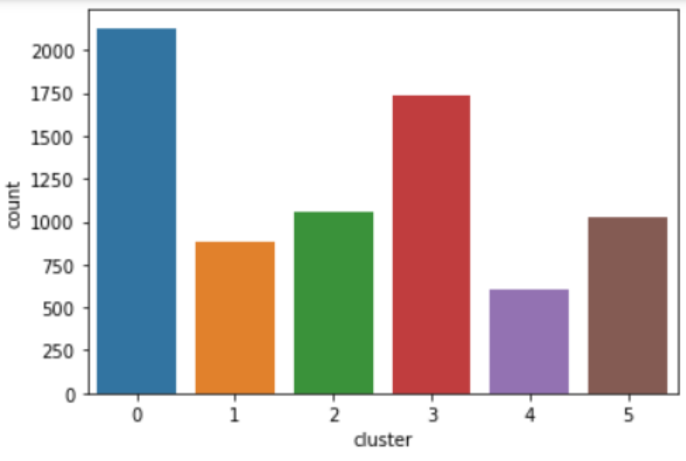
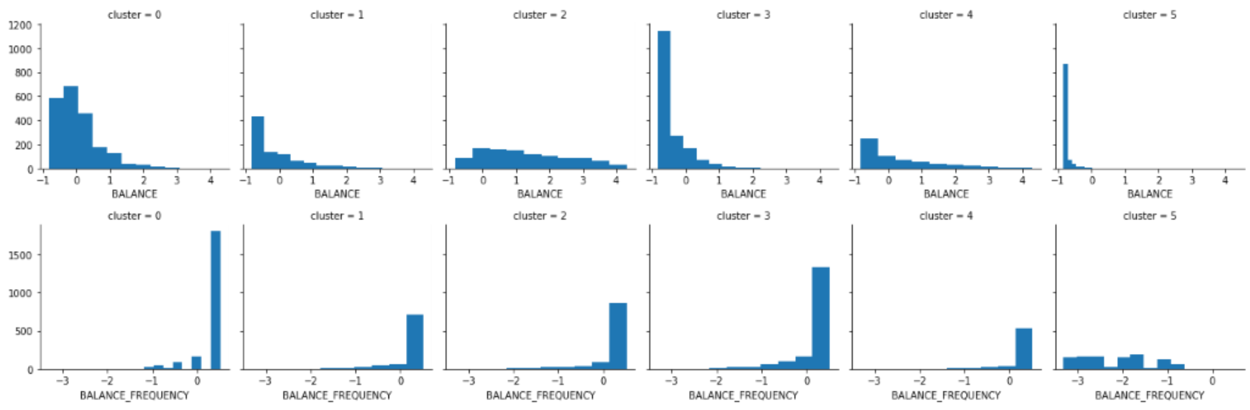


Figure 19: Count of customers in each cluster

From the above diagram we can see that the cluster "0" has the highest number of customers and some of the respective attribute are also visualized as below.



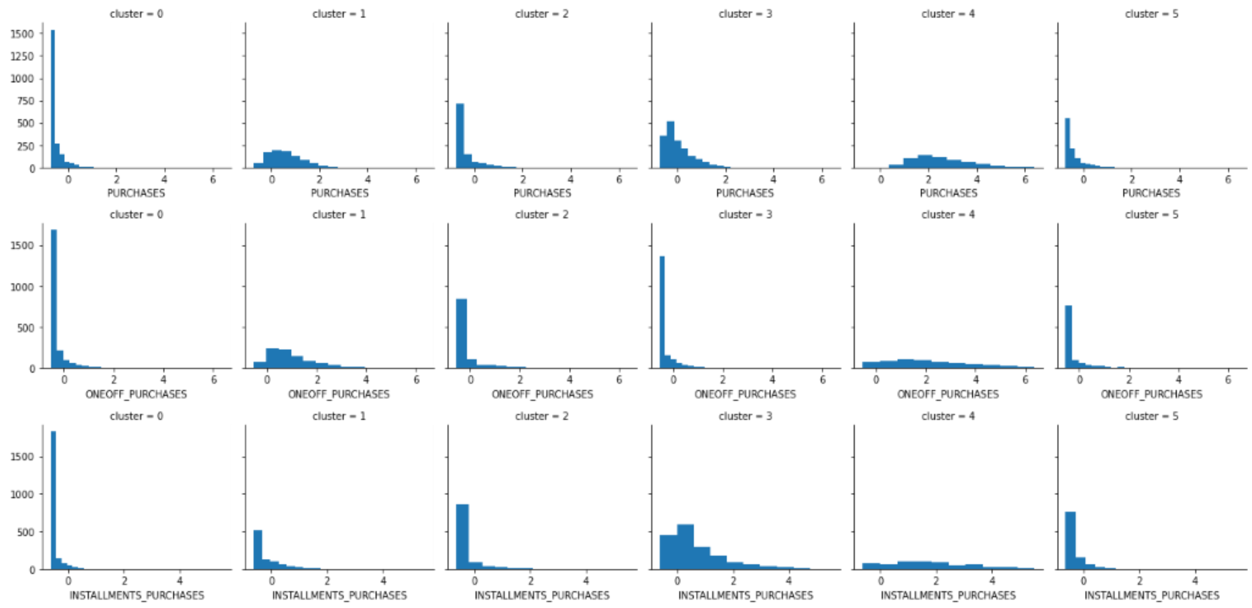


Figure 20: Number of customers in each cluster by single attribute

We can see different type of behaviours can be seen in each cluster from this we can specifically target any of the attribute and understand and compare the behaviours.

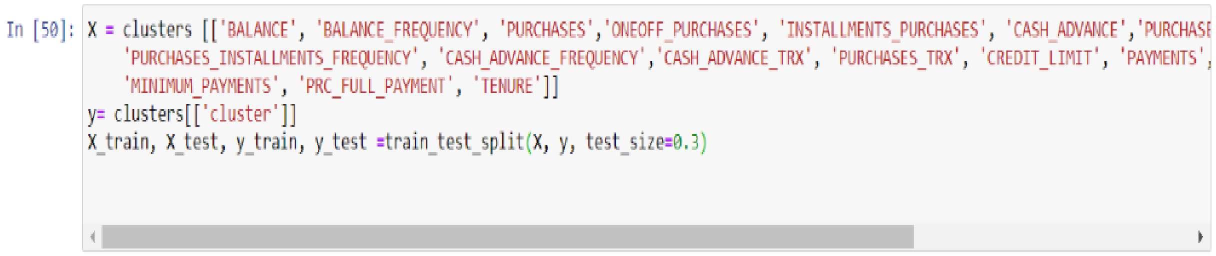
Applying decision tree algorithm to this data_2 cluster (class variable) to predict the future transaction so that the customer can be classified based on the cluster.

Checking unique clusters in the class variable and size of the clusters

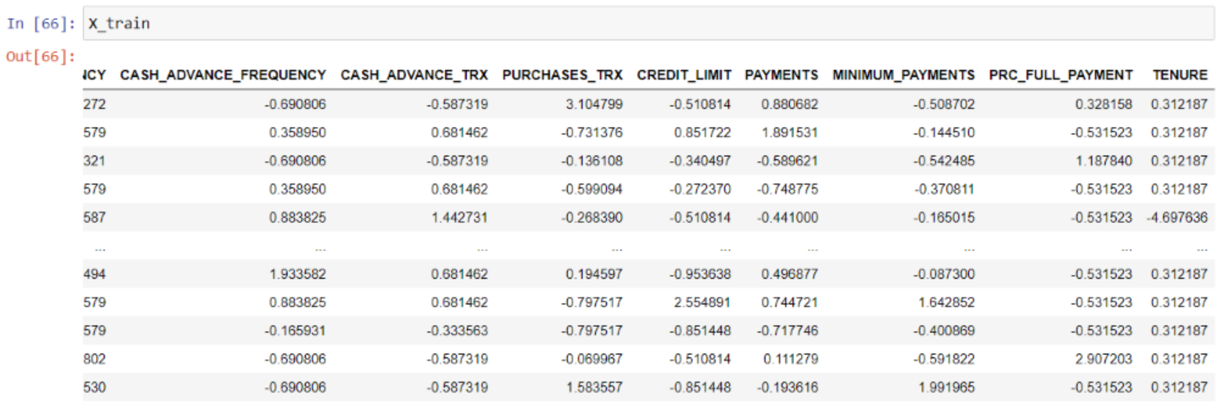
```
In [46]: clusters.shape
Out[46]: (7434, 18)

In [47]: clusters['cluster'].unique()
Out[47]: array([0, 2, 1, 3, 5, 4])

In [48]: clusters.groupby('cluster').size()
Out[48]: cluster
         0    2129
         1     883
         2    1052
         3    1739
         4     607
         5    1024
         dtype: int64
```

From above picture we can see that, how count of customers that are belonging to each cluster type.

Splitting the dataset into X and Y where X contains all the attributes information and Y contains clusters information and these further split into x_train, x_test and y_test and y_test.

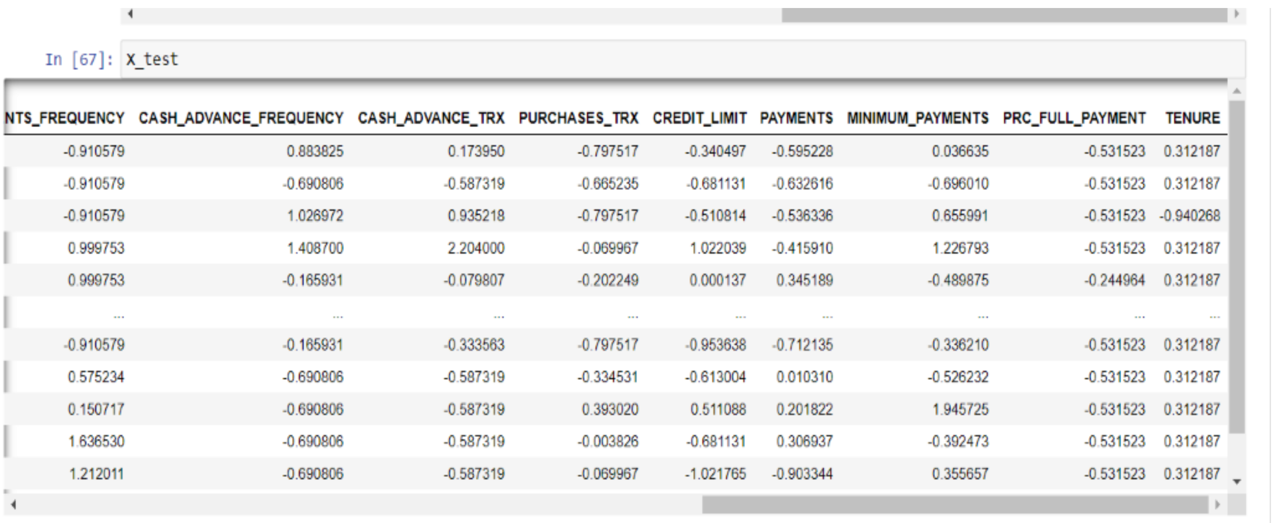


```
In [50]: X = clusters [['BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES','ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE','PURCHASE
          'PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY','CASH_ADVANCE_TRX', 'PURCHASES_TRX', 'CREDIT_LIMIT', 'PAYMENTS',
          'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT', 'TENURE']]
         y= clusters[['cluster']]
         X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=0.3)
```



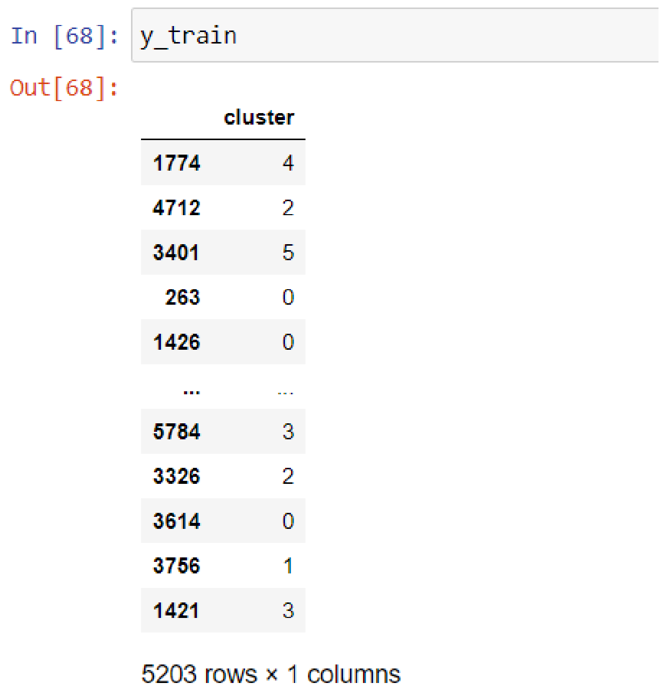
```
In [66]: X_train
Out[66]:
```

ICY	CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
272	-0.690806	-0.587319	3.104799	-0.510814	0.880682	-0.508702	0.328158	0.312187
579	0.358950	0.681462	-0.731376	0.851722	1.891531	-0.144510	-0.531523	0.312187
321	-0.690806	-0.587319	-0.136108	-0.340497	-0.589621	-0.542485	1.187840	0.312187
579	0.358950	0.681462	-0.599094	-0.272370	-0.748775	-0.370811	-0.531523	0.312187
587	0.883825	1.442731	-0.268390	-0.510814	-0.441000	-0.165015	-0.531523	-4.697636
...	...	...	...	...	...	...	...	...
494	1.933582	0.681462	0.194597	-0.953638	0.496877	-0.087300	-0.531523	0.312187
579	0.883825	0.681462	-0.797517	2.554891	0.744721	1.642852	-0.531523	0.312187
579	-0.165931	-0.333563	-0.797517	-0.851448	-0.717746	-0.400869	-0.531523	0.312187
802	-0.690806	-0.587319	-0.069967	-0.510814	0.111279	-0.591822	2.907203	0.312187
530	-0.690806	-0.587319	1.583557	-0.851448	-0.193616	1.991965	-0.531523	0.312187

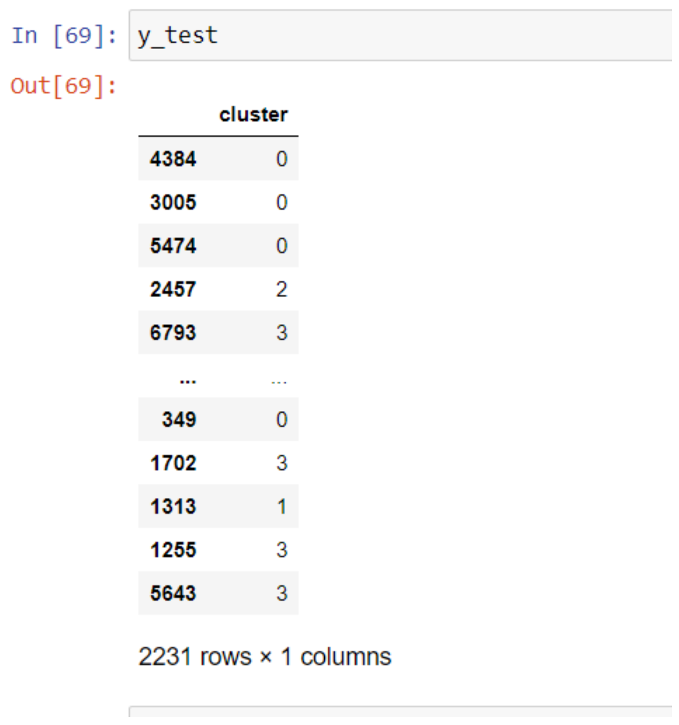


```
In [67]: X_test
```

NTS_FREQUENCY	CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
-0.910579	0.883825	0.173950	-0.797517	-0.340497	-0.595228	0.036635	-0.531523	0.312187
-0.910579	-0.690806	-0.587319	-0.665235	-0.681131	-0.632616	-0.696010	-0.531523	0.312187
-0.910579	1.026972	0.935218	-0.797517	-0.510814	-0.536336	0.655991	-0.531523	-0.940268
0.999753	1.408700	2.204000	-0.069967	1.022039	-0.415910	1.226793	-0.531523	0.312187
0.999753	-0.165931	-0.079807	-0.202249	0.000137	0.345189	-0.489875	-0.244964	0.312187
...	...	...	...	...	...	...	...	...
-0.910579	-0.165931	-0.333563	-0.797517	-0.953638	-0.712135	-0.336210	-0.531523	0.312187
0.575234	-0.690806	-0.587319	-0.334531	-0.613004	0.010310	-0.526232	-0.531523	0.312187
0.150717	-0.690806	-0.587319	0.393020	0.511088	0.201822	1.945725	-0.531523	0.312187
1.636530	-0.690806	-0.587319	-0.003826	-0.681131	0.306937	-0.392473	-0.531523	0.312187
1.212011	-0.690806	-0.587319	-0.069967	-1.021765	-0.903344	0.355657	-0.531523	0.312187

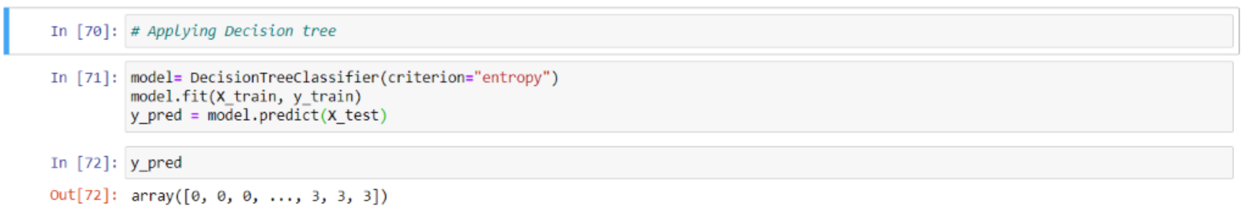


```
In [68]: y_train
Out[68]:
        cluster
1774          4
4712          2
3401          5
 263          0
1426          0
 ...        ...
5784          3
3326          2
3614          0
3756          1
1421          3

5203 rows × 1 columns
```



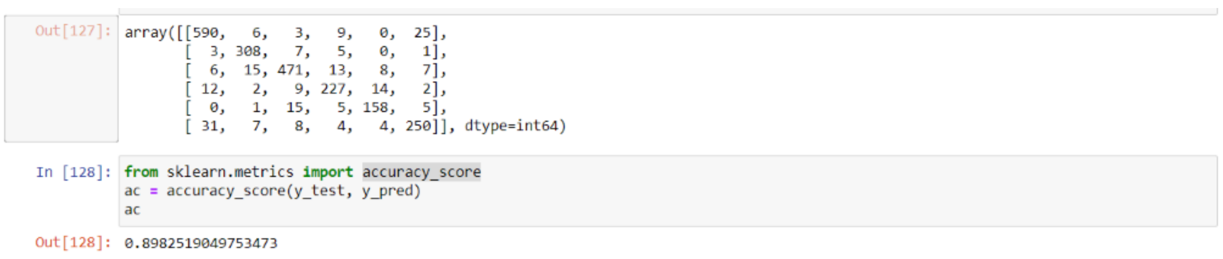
```
In [69]: y_test
Out[69]:
        cluster
4384          0
3005          0
5474          0
2457          2
6793          3
 ...        ...
 349          0
1702          3
1313          1
1255          3
5643          3

2231 rows × 1 columns
```

Applying decision tree model with criterion="entropy" to the split datasets and predicting clusters for test dataset,



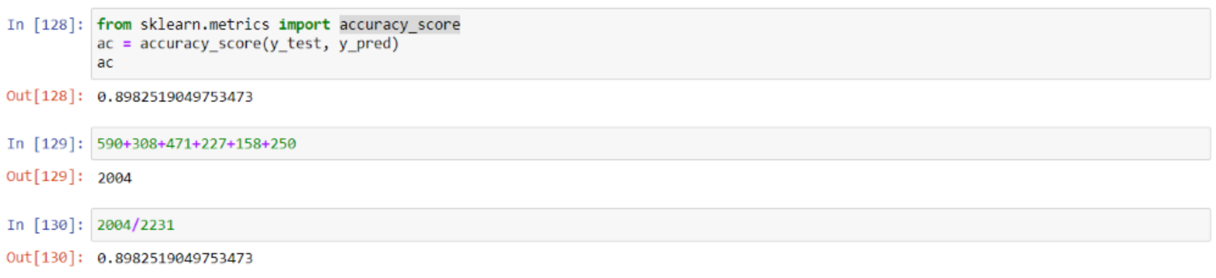
```
In [70]: # Applying Decision tree

In [71]: model= DecisionTreeClassifier(criterion="entropy")
         model.fit(X_train, y_train)
         y_pred = model.predict(X_test)

In [72]: y_pred

Out[72]: array([0, 0, 0, ..., 3, 3, 3])
```

Applying confusion matrix for y_test, y_pred



```
Out[127]: array([[590,   6,   3,   9,   0,  25],
                 [  3, 308,   7,   5,   0,   1],
                 [  6,  15, 471,  13,   8,   7],
                 [ 12,   2,   9, 227,  14,   2],
                 [  0,   1,  15,   5, 158,   5],
                 [ 31,   7,   8,   4,   4, 250]], dtype=int64)

In [128]: from sklearn.metrics import accuracy_score
          ac = accuracy_score(y_test, y_pred)
          ac

Out[128]: 0.8982519049753473
```

From accuracy score, checking accuracy

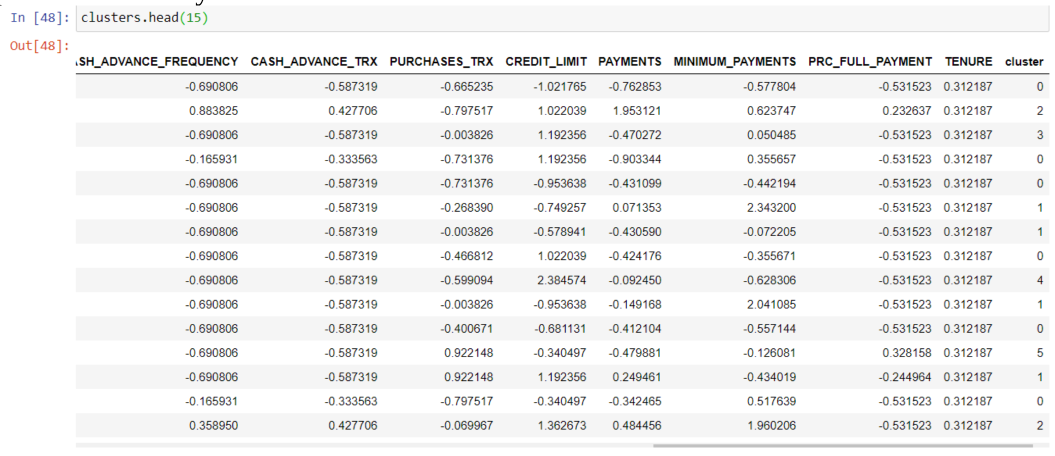


```
In [128]: from sklearn.metrics import accuracy_score
          ac = accuracy_score(y_test, y_pred)
          ac

Out[128]: 0.8982519049753473

In [129]: 590+308+471+227+158+250

Out[129]: 2004

In [130]: 2004/2231

Out[130]: 0.8982519049753473
```

Our decision tree model was able to predict the cluster from customer transaction with 89% accuracy.

7.2 Output

By applying decision tree, we can classify the customer into different clusters with all the attributes and without using the clustering method for the future customers, below is the screens shot of the predict clusters by the decision tree model.



```
In [48]: clusters.head(15)

Out[48]:
```

SH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE	cluster
-0.690806	-0.587319	-0.665235	-1.021765	-0.762853	-0.577804	-0.531523	0.312187	0
0.883825	0.427706	-0.797517	1.022039	1.953121	0.623747	0.232637	0.312187	2
-0.690806	-0.587319	-0.003826	1.192356	-0.470272	0.050485	-0.531523	0.312187	3
-0.165931	-0.333563	-0.731376	1.192356	-0.903344	0.355657	-0.531523	0.312187	0
-0.690806	-0.587319	-0.731376	-0.953638	-0.431099	-0.442194	-0.531523	0.312187	0
-0.690806	-0.587319	-0.268390	-0.749257	0.071353	2.343200	-0.531523	0.312187	1
-0.690806	-0.587319	-0.003826	-0.578941	-0.430590	-0.072205	-0.531523	0.312187	1
-0.690806	-0.587319	-0.466812	1.022039	-0.424176	-0.355671	-0.531523	0.312187	0
-0.690806	-0.587319	-0.599094	2.384574	-0.092450	-0.628306	-0.531523	0.312187	4
-0.690806	-0.587319	-0.003826	-0.953638	-0.149168	2.041085	-0.531523	0.312187	1
-0.690806	-0.587319	-0.400671	-0.681131	-0.412104	-0.557144	-0.531523	0.312187	0
-0.690806	-0.587319	0.922148	-0.340497	-0.479881	-0.126081	0.328158	0.312187	5
-0.690806	-0.587319	0.922148	1.192356	0.249461	-0.434019	-0.244964	0.312187	1
-0.165931	-0.333563	-0.797517	-0.340497	-0.342465	0.517639	-0.531523	0.312187	0
0.358950	0.427706	-0.069967	1.362673	0.484456	1.960206	-0.531523	0.312187	2

8. RESULTS, DISCUSSION AND FUTURE WORK

Credit card data has been segmented to 6 customer clusters using K-means algorithm to understand the customer behaviour. In results characteristics of each cluster has been found and has been discussed accordingly, these same insights can apply for future customer predicted by the decision tress models.

These results can be used by the banks and marketing executives to implement strategies on increasing credit card usage among average customers as of users from Cluster 0 or , also finding frauds and decreasing credit limits or revoking credit cards to users who do not payback for this belonging to cluster 3.

Better results can be obtained in this scenario, if customers data is more emphasized on attitude and benefits for customers rather than demographics. In future banks should try to grab such data to improve their market.

For Decision tree classification, in future we can consider a greater number of customer behaviour attributes and group them into several other root nodes increasing the decision tree accuracy. There are other few concepts that need to be considered with large number of attributes like balanced data fields, it can create problems to classify the attributes further down to leaf nodes.

Behavioural Noisy data is also a challenging task to remove when large number of attributes are considered, more work is needed on this topic in future.

9. CONCLUSION

Clustering on credit card data has resulted good groups of customers which are easy to analyze, strategies to increase credit card usage and market should be implemented by banks depending on the cluster characteristics. By decision tress we were able to use few more attributes to classify customers into different clusters and understand their behaviour to make better business decisions that can bring more profit to banks.

10. References

1. Achim, M. and Sebastian, M. (2001) 'Segmentation of bank customers by expected benefits and attitudes', *International Journal of Bank Marketing*, 19(1), pp. 6-18. doi: 10.1108/02652320110366472.

2. E. Umuhoza *et al.* (2020) *Using Unsupervised Machine Learning Techniques for Behavioral-based Credit Card Users Segmentation in Africa.*

3. Gulati, P., Sharma, A. and Gupta, M. (2016) 'Theoretical study of decision tree algorithms to identify pivotal factors for performance improvement: A review', *International Journal of Computer Applications*, 141(14).

4. Li, W. *et al.* (2010) 'Credit Card Customer Segmentation and Target Marketing Based on Data Mining', *2010 International Conference on Computational Intelligence and Security*, , pp. 73-76.

5. S. Marne, S. Churi and M. Marne (2020) 'Predicting Breast Cancer using effective Classification with Decision Tree and K Means Clustering technique', - *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)*. doi: 10.1109/ESCI48226.2020.9167544.

6. Li, Q. & Xie, Y. 2019, "A Behavior-cluster Based Imbalanced Classification Method for Credit Card Fraud Detection", ACM, , pp. 134.

7. https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

8. https://www.tutorialspoint.com/data_mining/dm_cluster_analysis.htm

9. K Means Clustering | K Means Clustering Algorithm in Python (analyticsvidhya.com)

10. Credit Card Customer Clustering with K-means | by Luke Sun | Towards Data Science

11. PEER MARKING

The module Advanced Data Analytics has helped me in understanding on how to implement various machine learning and mathematical algorithms, from this assessment we have particularly worked on Clustering and Classification as a group. I have contributed to working on Code implementation for Clustering and understanding clustered output and also have a complete idea of the code implementation and concepts.

S. No	Names	Marks
1	VISHWAS SIMHA SAMUDRALA (W9316602)	3
2	ARUN SAI PILLI (W9323581)	2
3	SRINIVAS YASWANTH THALLURI (W9330556)	2
4	DEVENDRA SAI MUPPALLA (W9312419)	2
5	MANIDEEP KOMMARAJU (W9311652)	2
6	SEBIN SUNNY (W9479792)	2