Project Report On <u>Credit Card Segmentation</u>

Submitted by:

Anusha Kulkarni.

Contents

1. Introduction

1.1 Problem Statement

This case requires trainees to develop a customer segmentation to define marketing strategy. 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. I found that there are four types of purchase behavior so I decided to cluster based on their purchase behavior.

1.2 Data

Our task is to develop customer segmentation to deploy several marketing strategies based on segment behavior.

CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES
C10001	40.900749	0.818182	95.4	0	95.4
C10002	3202.467416	0.909091	0	0	0
C10003	2495.148862	1	773.17	773.17	0
C10004	1666.670542	0.636364	1499	1499	0
C10005	817.714335	1	16	16	0
C10006	1809.828751	1	1333.28	0	1333.28

CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMENTS_FREQUENCY	CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	PURCHASES_TRX
0	0.166667	0	0.083333	0	0	2
6442.945483	0	0	0	0.25	4	0
0	1	1	0	0	0	12
205.788017	0.083333	0.083333	0	0.083333	1	1
0	0.083333	0.083333	0	0	0	1
0	0.666667	0	0.583333	0	0	8

CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
1000	201.802084	139.509787	0	12
7000	4103.032597	1072.340217	0.222222	12
7500	622.066742	627.284787	0	12
7500	0		0	12
1200	678.334763	244.791237	0	12
1800	1400.05777	2407.246035	0	12

The variable present in this data are

CUST_ID
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

The details of variable present in the dataset are as follows –

Number of attributes:

● CUST_ID -Credit card holder ID

● BALANCE Monthly average balance (based on daily balance averages)

● BALANCE_FREQUENCY Ratio of last 12 months with balance

● PURCHASES Total purchase amount spent during last 12 months

● ONEOFF_PURCHASES Total amount of one-off purchases

● INSTALLMENTS_PURCHASES Total amount of installment purchases

● CASH_ADVANCE Total cash-advance amount

● PURCHASES_ FREQUENCY-Frequency of purchases (percentage of months with at least on purchase)

● ONEOFF_PURCHASES_FREQUENCY Frequency of one-off-purchases

● PURCHASES_INSTALLMENTS_FREQUENCY Frequency of installment Purchases

● CASH_ADVANCE_ FREQUENCY Cash-Advance frequency

● AVERAGE_PURCHASE_TRX Average amount per purchase transaction

● CASH_ADVANCE_TRX Average amount per cash-advance transaction

● PURCHASES_TRX Average amount per purchase transaction

● CREDIT_LIMIT Credit limit

● PAYMENTS - Total payments (due amount paid by the customer to decrease their statement balance) in the period

● MINIMUM_PAYMENTS - Total minimum payments due in the period.

● PRC_FULL_PAYMENT - Percentage of months with full payment of the due statement balance

● TENURE - Number of months as a customer Evaluation B

2. Methodology

2.1 Pre Processing

The real-world data has many errors and more inconsistent in format. In preprocessing we transform the raw data into understandable format. Real-world data is incomplete and/or lacking in certain behaviors or trends, and it contain many errors. Data preprocessing is a proven method of resolving such Looking the data refers to exploring the data, cleaning data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

2.1.1 Exploratory Data Analysis

In exploring the data we have

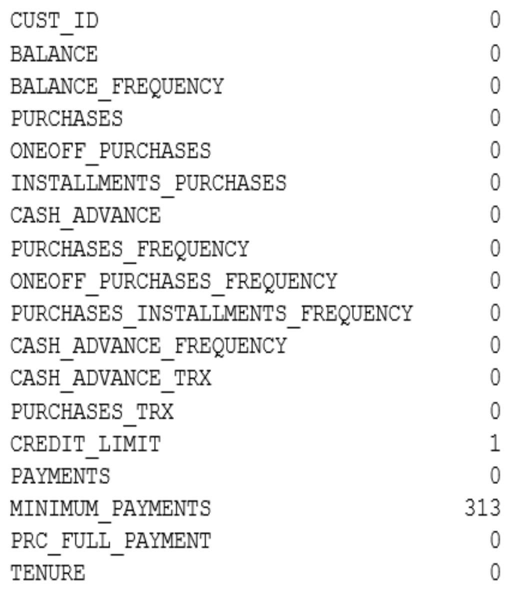
- CUST_ID which was an index to data doesn't carry much information but Later required for analysis so stored in one variable.

2.1.2 Missing Value Analysis

Missing value analysis is done to check is there any missing value present in given dataset. There missing values can be treated by replacing it by using several imputation techniques like mean, median and KNN imputation for continuous variable and mode for categorical variable. These techniques are used to impute missing value.

In R function(x){sum(is.na(x))} is the function used to check the sum of missing

In python credit_df.isnull().sum() is used to detect any missing value



```
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
```

From above we can observe that there is missing value in 2 variables in given dataset.

We replaced the given variables with median so that data don't be influenced by outliers. The two variables MINIMUM_PAYMENTS and CREDITLIMIT has missing values we replaced it with mean.

In python -

```
credit['CREDIT_LIMIT'].fillna(dat1['CREDIT_LIMIT'].median(),inplace=True)

credit['MINIMUM_PAYMENTS'].fillna(credit['MINIMUM_PAYMENTS'].median(),inplace=True)

credit.isnull().sum()
```

2.1.3 Feature Creation

In this clustering analysis for profiling we need some KPI's key performing indices so that they will help in developing profiling of customers and mainly helps in to select the suitable marketing strategy as from KPI's we can know customer behavior's

The KPIs are

1. Monthly_average_purchases
2. Monthly_cashadvance
3. Limit usage
4. Payment to min payment ratio
5. Purchase type

Purchase type: These variable was created based on the instalment and oneoff purchases we categorized data into 4 types using logic as follows

credit[(credit ['ONEOFF_PURCHASES']==0) & (credit ['INSTALLMENTS_PURCHASES']==0)].shape

credit [(dat1['ONEOFF_PURCHASES']>0) & (credit ['INSTALLMENTS_PURCHASES']>0)].shape

credit [(dat1['ONEOFF_PURCHASES']>0) & (credit ['INSTALLMENTS_PURCHASES']==0)].shape

credit [(dat1['ONEOFF_PURCHASES']==0) & (credit ['INSTALLMENTS_PURCHASES']>0)].shape

So purchase type variable defines how the data categorized as only oneoff , only instalment , both oneoff instalment and none

After categorizing we used that variable in analysis by converting it in the form of dummies

```
cr_pre['purchase_type']= credit.loc[:,'purchase_type']

pd.get_dummies(cr_pre['purchase_type']).head()
```

2.1.4 Outlier Analysis

In data there are many extreme value and even the data was in skewed format so to treat this kind of data we use logarithm transformation. It’s a kind of Extreme value analysis.

```
# log tranformation  cr_log=

credit.drop(['purchase_type'],axis=1).applymap(lambda x: np.log(x+1))
```

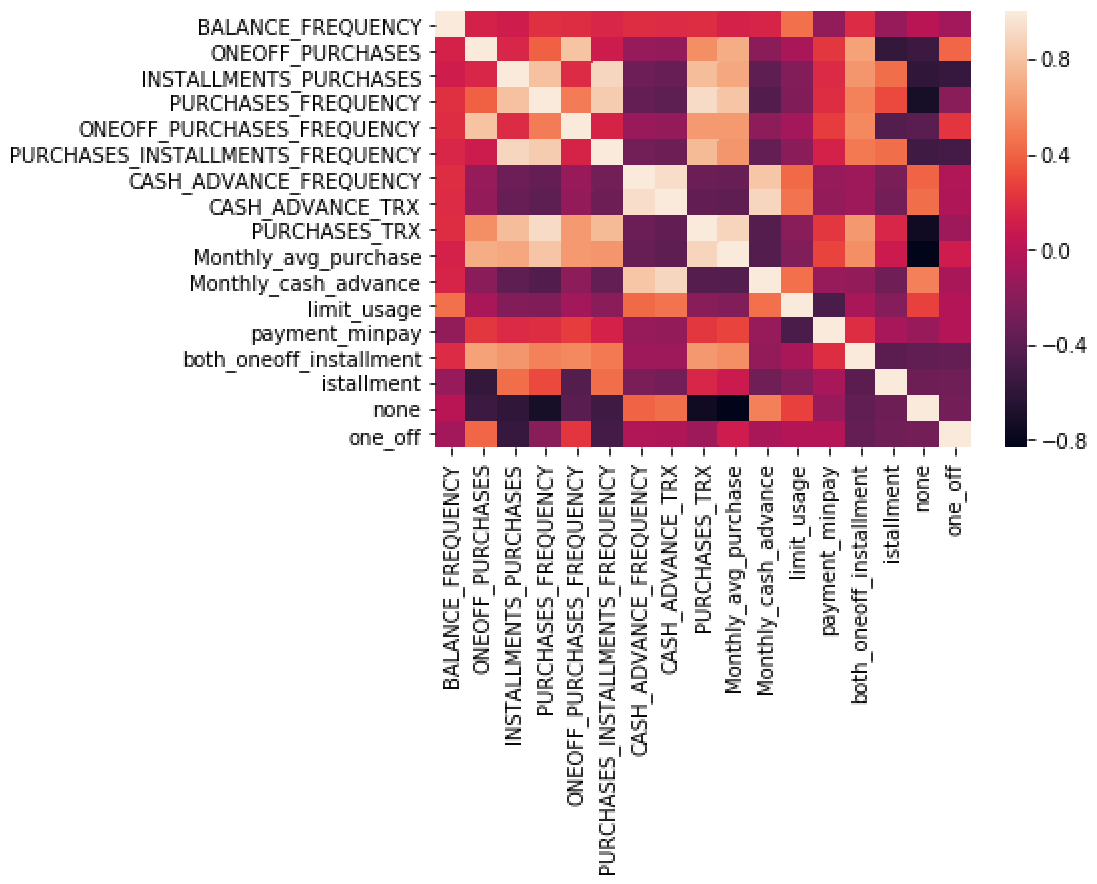
2.1.5 Feature Selection

Feature selection analysis is done to select subsets of relevant features (variables, predictors) to be in model construction.

As all variable are continuous so we can only go for correlation check. As chi-square test is only for categorical variable.

Figure show a correlation plot for all numeric variable present in dataset

In above correlation plot the two independent variables [temp, atemp] are highly correlated which are carrying same weightage of data this will leads to multicollinearity problem. They both has correlation value near to 0.99. So, I decided to remove atemp variable from dataset.



As we can see there are more number of variable which are correlated to each other so i consider PCA analysis for variable reduction

PCA is widely used to simplify high-dimensional datasets to lower dimensional ones. It lower number of variables that are not correlated, without losing information contained in the dataset

To reduce the dimensionality, the number of components are lower than the original columns Optimal ‘k’:

- Top ‘k’ components capture what proportion of the variance?

– We look at a scree plot, similar to the plot we used to choose number of clusters in k-means clustering

- Against the #principal components, we look at the cumulative proportion of variance captured - Choose k after which incremental benefit negligible

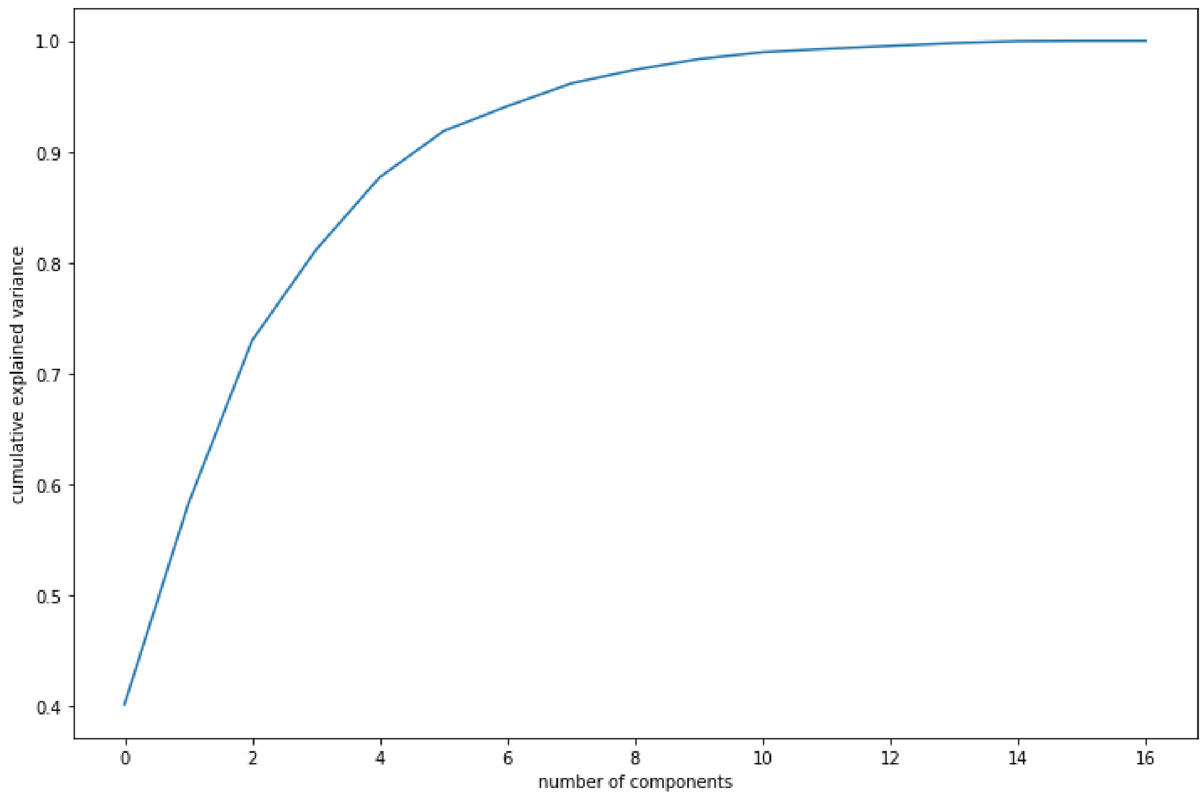
- Common choices: 80%, 90%, 95% variance

from sklearn.decomposition import PCA

```
pca = PCA(svd_solver='randomized', random_state=42)

#Performing the PCA

pca.fit(dat2)

#Let's check the variance ratios

pca.explained_variance_ratio_
```

It better to prefer n number of components which covers explained variance at least 85-95 percentage

```
#Plotting the scree plot

%matplotlib inline

fig = plt.figure(figsize = (12,8))

plt.plot(np.cumsum(pca.explained_variance_ratio_))

plt.xlabel('number of components')

plt.ylabel('cumulative explained variance')

plt.show()
```

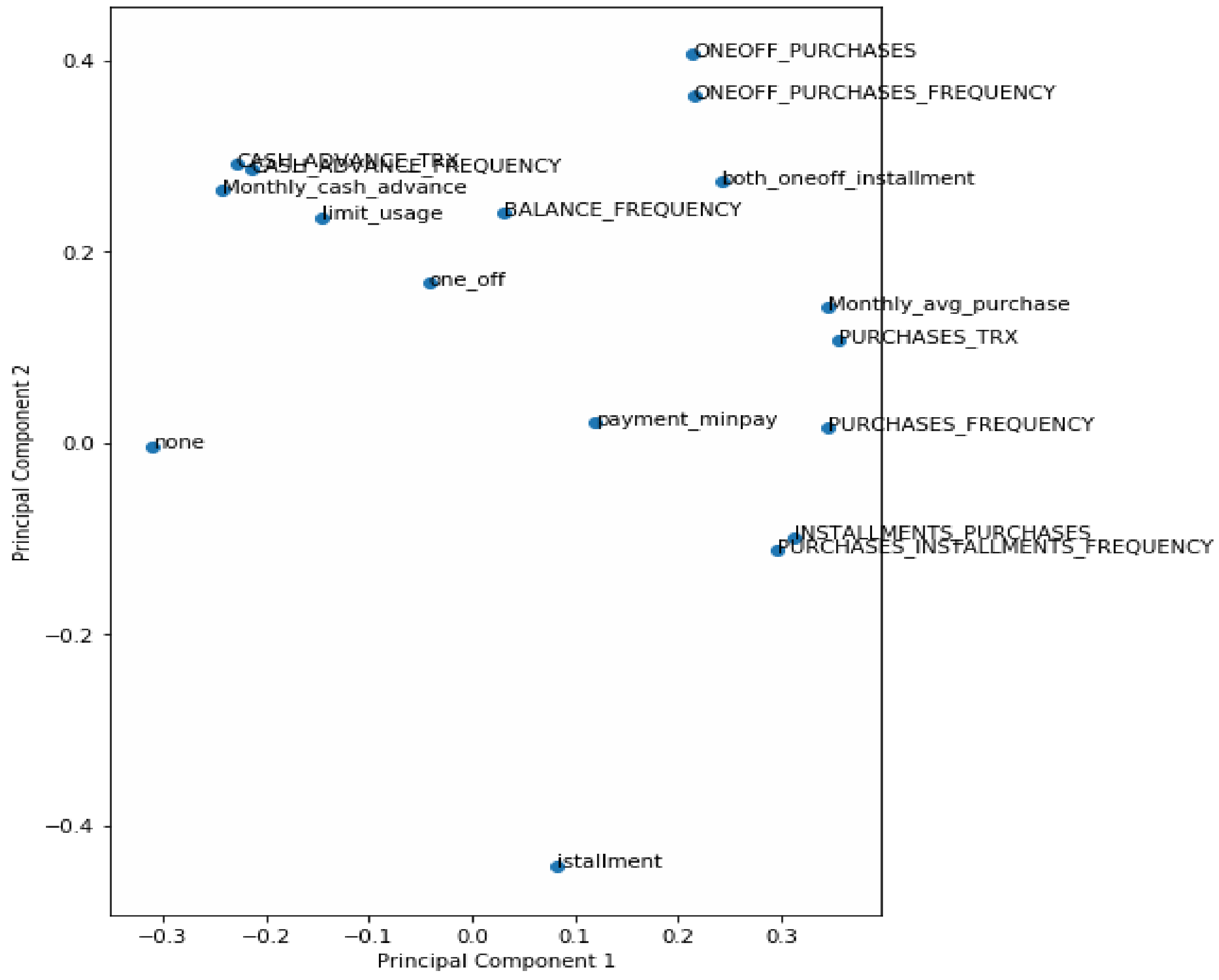


```
np.cumsum(pca.explained_variance_ratio_)
```

```
array([0.40205787, 0.58264398, 0.72993793, 0.81154428, 0.87705558,
       0.91864924, 0.94109253, 0.96161141, 0.97397871, 0.98358966,
       0.98972481, 0.992755  , 0.99539076, 0.99796169, 0.99963605,
       1.        , 1.        ])
```

As from above scree plot and cumulative summation of variance of each PC component we get to know that the 5 P components are explaining 87% of variance

By plotting graph between PC1 and PC2 components we can see how the features are loaded

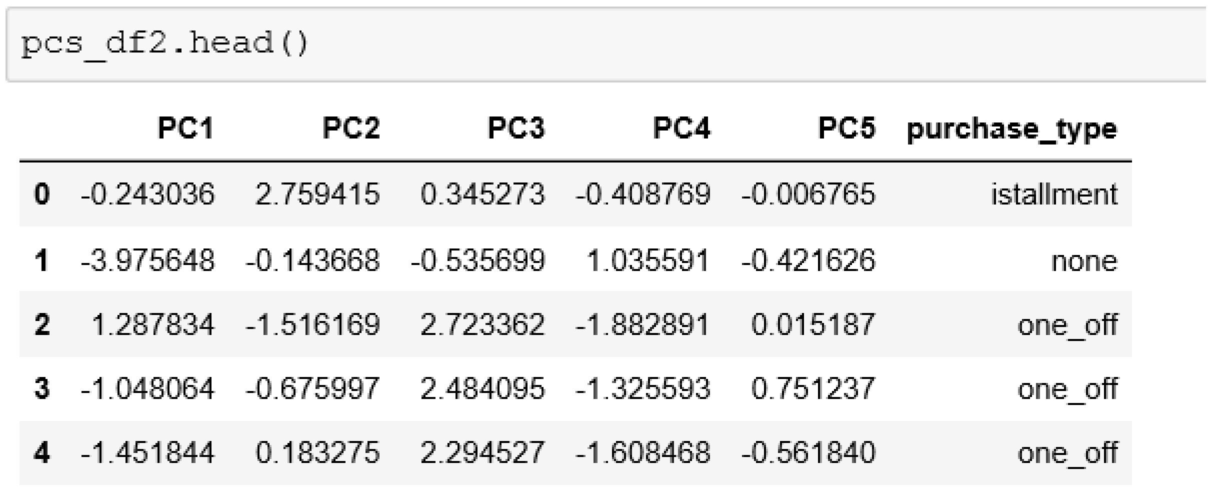


#Finally let's go ahead and do dimenstionality reduction using the five Principal Components

from sklearn.decomposition import IncrementalPCA

pca_final = IncrementalPCA(n_components=5)

From the variance we get to know 5 PC's are enough now lets we implement dimensionality reduction using principal components



```
pcs_df2.head()
```

	PC1	PC2	PC3	PC4	PC5	purchase_type
0	-0.243036	2.759415	0.345273	-0.408769	-0.006765	istallment
1	-3.975648	-0.143668	-0.535699	1.035591	-0.421626	none
2	1.287834	-1.516169	2.723362	-1.882891	0.015187	one_off
3	-1.048064	-0.675997	2.484095	-1.325593	0.751237	one_off
4	-1.451844	0.183275	2.294527	-1.608468	-0.561840	one_off

Here we able to reduce the higher dimensionality of data 18 variables to lower dimensionality of data 5 dimensions without losing much information.

2.1.6 Feature Scaling

Feature scaling includes two functions normalization and standardization. It is done reduce unwanted variation either within or between variables and to bring all of the variables into proportion with one another.

Standardrizing data

We use standardizing of data when the data was normally distributed

The main purpose of standardizing was to make the variables unit less.

```
from sklearn.preprocessing import  StandardScaler
sc=StandardScaler()

credit_scaled =sc.fit_transform(cr_dummy)
```

2.2 Clustering Modelling

2.2.1 Model Implementation

The main objective of this case was to develop a customer segmentation to define marketing strategy. Here, we can apply several clustering models. After applying many clustering models, we will do profiling of customers based on their purchasing behaviour. The two clustering models that I decided to apply are as follows.

1. K-means clustering
2. Hierarchical clustering

2.2.2 K-means clustering

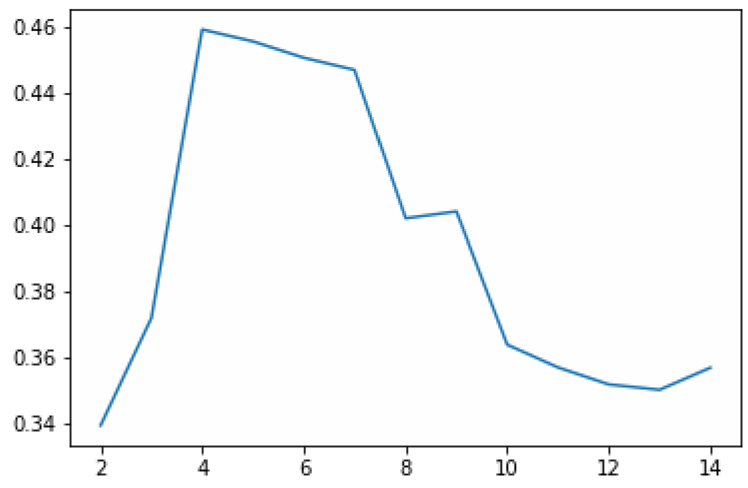
The K-Means algorithm uses the concept of the centroid to create K clusters. In simple terms, a centroid of n points on an x-y plane is another point having its own x and y coordinates and is often referred to as the geometric centre of the n points.

For example, consider three points having coordinates (x1, y1), (x2, y2) and (x3, y3). The centroid of these three points is the average of the x and y coordinates of the three points, i.e.
(x1 + x2 + x3 / 3, y1 + y2 + y3 / 3).

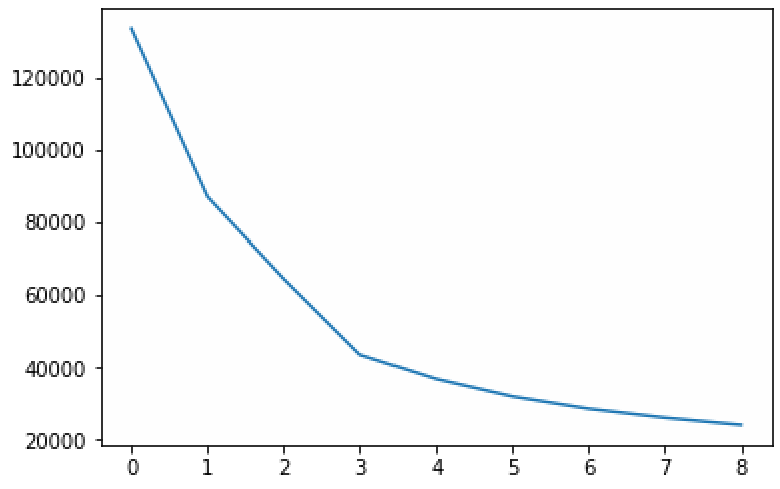
Caluclating Hopkins statistics

It’s a kind of hypothesis testing and was a way to measuring the central tendency of a data set. After checking **Hopkins** we got value as 0.93 as it is greater than 0.5 so we can conclude that the data has adequate data for clustering purpose.

After calculating **silhouette analysis score** we plotted a graph which it was suggesting us to prefer 4 clusters.



Even elbow curve suggested us to prefer cluster range from 3 to 5. Here also we're seeing a distinct bend at around 4 clusters. Hence it seems a good K to choose. From three kind of estimations we choose k value as 4.

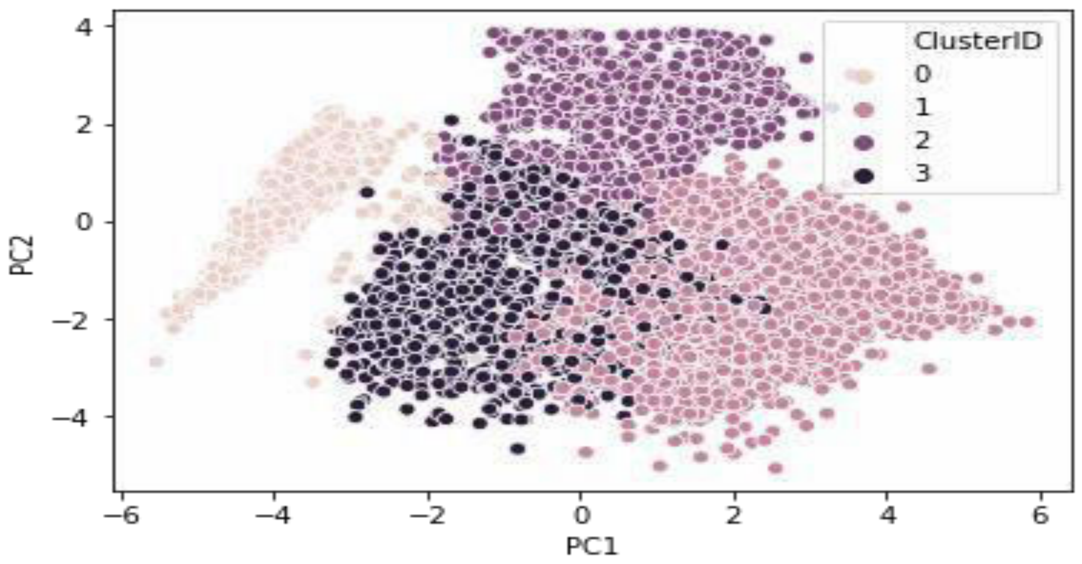


```
#Let's perform K means using K=4

model_clus2 = KMeans(n_clusters = 4, max_iter=50,random_state = 50)

model_clus2.fit(dat3_1)
```

```
1    2757
2    2229
0    2090
3    1874
Name: ClusterID, dtype: int64
```

Plotting the graph after creating clusters is as follows.

sns.scatterplot(x='PC1',y='PC2',hue='ClusterID',legend='full',data=dat_km)



From above graph we can see how accurately it clustered and we didn't found any ambiguity to identify cluster of each observation.

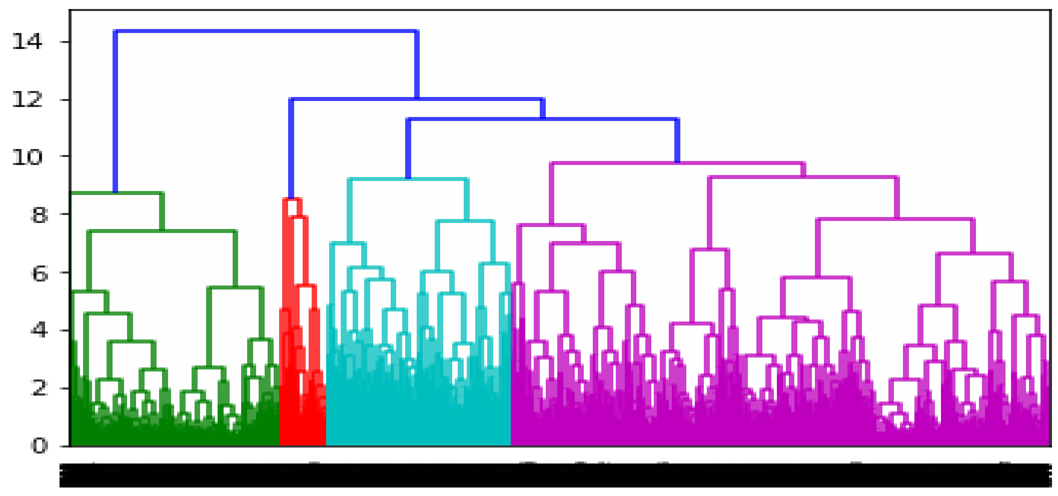
2.2.3 Hierarchical clustering

Hierarchical clustering is type of clustering analysis that groups the similar objects into groups called clusters. Finally, it creates n number of clusters where each cluster is distinct to each other and object in each cluster are broadly similar to each other.

mergings = linkage(pcs_df2, method = "complete", metric='euclidean')

dendrogram(mergings)

plt.show()



Above Figure represents the clusters in colour format as we can see it formed four clusters represented in different colours.

clusterCut = pd.Series(cut_tree(mergings, n_clusters = 4).reshape(-1,))

The above code assigns cluster ids to each observation.

.

Random forest functions in below way

i. Draws a bootstrap sample from training data.
ii. For each sample grow a decision tree and at each node of the tree
    a. Randomly draws a subset of n-try variable and p total of features that are available
    b. Picks the best variable and best split from the subset of n-try variable
    c. Continues until the tree is fully grown. Code in R and Python to

    create Random Forest model R-script:

RF_model=randomForest(cnt~.,data=train,importance=TRUE,ntree=200)

predictions_RF=predict(RF_model,test[,-12]) Python:

RFmodel = RandomForestRegressor(n_estimators = 200).fit(train.iloc[:,0:11], train.iloc[:,11])

RF_Predictions = RFmodel.predict(test.iloc[:,0:11])

3. Conclusion

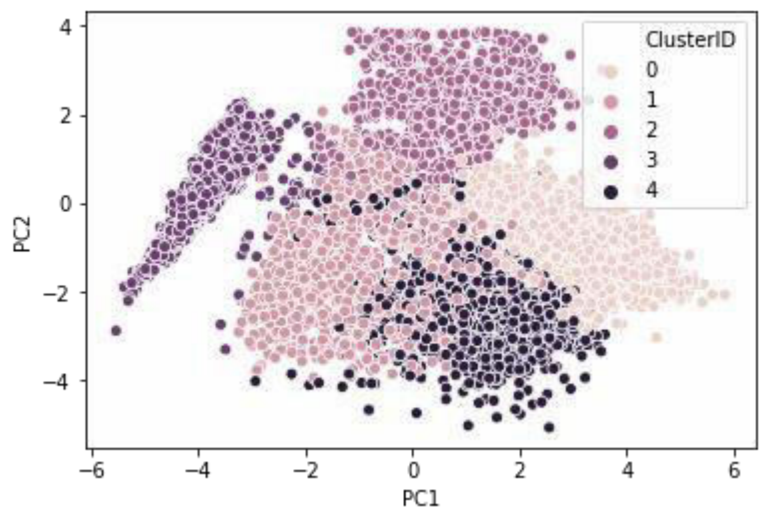
3.1 Analysis of clusters

Insights with 4 Clusters

- Cluster 3 customers are doing maximum One_Off transactions and has least payment ratio amongst all the cluster.
- Cluster 0 is the group of customers who have highest monthly cash advance and doing both installment as well as one_off purchases, have comparatively good credit score but have poor average purchase score.
- Cluster 1 customers have maximum Average Purchase and good Monthly cash advance but this cluster doesn't do installment or one_off purchases.
- cluster 2 is doing maximum installment, has maximum payment too min_payment ratio and doesn't do one-off purchases

	both	none	installme	oneoff
ClusterID	0	1	2	3
clu_balance	2177.572	1811.244	798.2671	1429.021
clu_purchases	1.859344	2280.612	543.5836	787.3537
clu_purchasefreq	0.003758	0.802265	0.703035	0.32117
clu_purchasetrx	0.045933	33.13529	12.05114	7.118997
clu_Monthlyavgpurchase	0.159337	193.7598	47.5604	69.75828
clu_Monthlycashadvance	186.298	67.64453	33.47482	77.84348
clu_limitusage	0.576217	0.354616	0.264156	0.378727
clu_cashadvtrx	6.552632	2.808125	1.018843	2.864995
clu_payminpaymentratio	9.927979	7.264788	13.40463	5.561421
clu_both	0.002392	1	0.002243	0.003735
clu_installment	0.017225	0	0.997757	0
clu_one_off	0.003349	0	0	0.996265
clu_none	0.977033	0	0	0
clu_creditlimit	4055.582	5748.837	3338.238	4512.906

But when you saw **cluster 5** two cluster was carrying same data (means they are of same purchase type) so I choose better to create only four clusters
Even if you see the plot the clustering is not properly clustered



The 4 and 1 are in ambiguity state to classify the cluster group of following observations.

BOTH THE HIERARCHICAL AND K-MEANS ARE CLUSTERED SAMELY SO I CONSIDER CLUSTERING WITH K=4 IS ADEQUATE ENOUGH FOR CUSTOMER PROFILING.

3.2 Marketing Strategy

Marketing Strategy Suggested:

a. Group 2

- They are potential target customers who are paying dues and doing purchases and maintaining comparatively good credit score)
- we can increase credit limit or can lower down interest rate
- Can be given premium card /loyalty cards to increase transactions

b. Group 1

- They have poor credit score and taking only cash on advance. We can target them by providing less interest rate on purchase transaction

c. Group 0

- This group is has minimum paying ratio and using card for just oneoff transactions (may be for utility bills only). This group seems to be risky group.

d. Group 3

- This group is performing best among all as cutomers are maintaining good credit score and paying dues on time.
- Giving rewards point will make them perform more purchases.

5. Appendix file

5.1 R-code:

```
rm(list=ls(all=T))
setwd("G:/DataScience Project ")
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies",
"e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')
lapply(x,require,character.only=TRUE)
rm(x)

## Read the data
credit <-read.csv("Credit-card-data.csv",stringsAsFactors = FALSE)

##Data Exploration
str(credit)
summary(credit)

##univariate analysis
missing_val=data.frame(apply(credit,2,function(x){sum(is.na(x))}))
missing_val$columns=row.names(missing_val)
names(missing_val)[1]="missing_percentage"
missing_val$missing_percentage=(missing_val$missing_percentage/nrow(credit))*100
missing_val=missing_val[order(-missing_val$missing_percentage),]
row.names(missing_val)=NULL
missing_val=missing_val[,c(2,1)]
write.csv(missing_val,"missing_perc.csv",row.names = F)

ggplot(data = missing_val[1:3,], aes(x=reorder(columns, -missing_percentage),y =
missing_percentage))+
  geom_bar(stat = "identity",fill = "grey")+xlab("Parameter")+
  ggtitle("Missing data percentage (Train)") + theme_bw()

##KNN IMPUTATION
CUST_ID=credit$CUST_ID
credit=knnImputation(credit[,2:18],k=3)
credit=cbind(CUST_ID,credit)
write.csv(credit,"afterknnmissing.csv",row.names = F)
#In missing value analysis KNN imputation is utilised with accurate value compared to mean and
median
#so i choosed KNN imputation method to treat with missing values

#New Variables creation#

credit$Monthly_Avg_PURCHASES <- credit$PURCHASES/credit$TENURE
credit$Monthly_CASH_ADVANCE <- credit$CASH_ADVANCE/credit$TENURE
credit$LIMIT_USAGE <- credit$BALANCE/credit$CREDIT_LIMIT
credit$MIN_PAYMENTS_RATIO <- credit$PAYMENTS/credit$MINIMUM_PAYMENTS

purchased=function(credit)
```

```
  {
    ifelse ((credit['ONEOFF_PURCHASES']==0) & (credit['INSTALLMENTS_PURCHASES']==0),
            'none',
   ifelse((credit['ONEOFF_PURCHASES']>0) & (credit['INSTALLMENTS_PURCHASES']>0),
            'both',
   ifelse ((credit['ONEOFF_PURCHASES']>0) & (credit['INSTALLMENTS_PURCHASES']==0),
            'oneoff',
            'installment')))
   }
credit$purchase_type=purchased(credit)
head(credit)
credit$purchase_type=as.factor(credit$purchase_type)

purchasetypedummy<-fastDummies::dummy_cols(credit$purchase_type)
credit1<-cbind(credit,purchasetypedummy)
credit2<-credit1[,c(-1,-23,-24)]

####OUTLIER ANALYSIS
library(openintro)
creditlog<-log1p(credit2)
summary(creditlog)
summary(credit)
col=c('BALANCE','PURCHASES','CASH_ADVANCE','TENURE','PAYMENTS','MINIMUM_PAY
MENTS','PRC_FULL_PAYMENT','CREDIT_LIMIT')
creditlog1<-creditlog[,c(-1,-3,-17,-14,-15,-16,-13)]
#library(DMwR)
# remove "Species", which is a categorical column
#outlier.scores <- lofactor(numeric_data, k=5)
#plot(density(outlier.scores))
#outlier.scores
#n=nrow(numeric_data)
#pch <- rep(".", n)
#pch[outlier.scores] <- "+"
#col <- rep("black", n)
#col[outlier.scores] <- "red"
#pairs(numeric_data[,1:7], pch=pch, col=col)


sum(is.na(creditlog1))


apply(creditlog1,2,sd)

# FeatureScaling
standardised_data <- data.frame(scale(creditlog1))


###FEATURE SELECTION
## Correlation Plot

apply(standardised_data,2,sd)
```

```
corrgram(credit1[,2:21], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

##Computing PCA
credit.pca<-prcomp(standardised_data,center = TRUE,scale. = TRUE)
summary(credit.pca)
str(credit.pca)
##the 5 PCs have explained 87% of variance in data
library(ggbiplot)
ggbiplot(credit.pca)

pca_df<-credit.pca$x[,1:5]
head(pca_df)
library(clustertend)
hopkins(pca_df,n=27)


####kmeans clustering
cluster_three <- kmeans(pca_df,3)
cluster_four <- kmeans(pca_df,4)
cluster_five <- kmeans(pca_df,5)
cluster_six <- kmeans(pca_df,6)

credit_new<-
cbind(pca_df,km_clust_3=cluster_three$cluster,km_clust_4=cluster_four$cluster,km_clust_5=cluster_
five$cluster ,km_clust_6=cluster_six$cluster   )
credit_new=as.data.frame(credit_new)
ggplot(credit_new, aes(PC1, PC2, color = credit_new$km_clust_4)) + geom_point()


credit_new=cbind(credit1$purchase_type,pca_df,km_clust_4=cluster_four$cluster)
bigdata=cbind(credit,credit_new)
bigdata=bigdata[,c(-24,-25,-26,-27,-28,-29)]
bigdata$km_clust_4=as.factor(bigdata$km_clust_4)
#####Analysis of the clusters
credit_new<-
cbind(credit,km_clust_3=cluster_three$cluster,km_clust_4=cluster_four$cluster,km_clust_5=cluster_f
ive$cluster ,km_clust_6=cluster_six$cluster   )
View(credit_new)
# Profiling

Num_Vars2 <- c(
  "CUST_ID",
  "BALANCE",
  "BALANCE_FREQUENCY",
  "PURCHASES",
  "ONEOFF_PURCHASES",
  "INSTALLMENT_PURCHASES",
  "CASH_ADVANCE",
  "PURCHASE_FREQUENCY",
  "ONEOFF_PURCHASE_FREQUENCY",
  "PURCHASES_INSTALLMENTS_FR",
  "CASH_ADVANCE_FREQUENCY",
  "CASH_ADVANCE_TRX",
  "PURCHASE_TRX",
  "CREDIT_LIMIT",
  "PAYMENTS",
  "MIN_PAYMENTS",
  "PRC_FULL_PAYMENT",
  "TENURE",
  "MONTHLY_AVG_PURCHASE",
  "MONTHLY_CASH_ADVANCE",
  "LIMIT_USAGE",
  "MIN_PAYMENT_RATIO"

)

require(tables)
tt <-cbind(tabular(1+factor(km_clust_3)+factor(km_clust_4)+factor(km_clust_5)+
                  factor(km_clust_6)~Heading()*length*All(credit[1]),
                  data=credit_new),tabular(1+factor(km_clust_3)+factor(km_clust_4)+factor(km_clust_5)+
                                 factor(km_clust_6)~Heading()*mean*All(bigdata[Num_Vars2]),
                               data=credit_new))

tt2 <- as.data.frame.matrix(tt)


rownames(tt2)<-c(
  "ALL",
  "KM3_1",
  "KM3_2",
  "KM3_3",
  "KM4_1",
  "KM4_2",
  "KM4_3",
  "KM4_4",
  "KM5_1",
  "KM5_2",
  "KM5_3",
  "KM5_4",
  "KM5_5",
  "KM6_1",
  "KM6_2",
  "KM6_3",
  "KM6_4",
  "KM6_5",
  "KM6_6")


colnames(tt2)<-c(
  "CUST_ID",
  "BALANCE",
  "BALANCE_FREQUENCY",
  "PURCHASES",
  "ONEOFF_PURCHASES",
  "INSTALLMENT_PURCHASES",
  "CASH_ADVANCE",
  "PURCHASE_FREQUENCY",
  "ONEOFF_PURCHASE_FREQUENCY",
  "PURCHASES_INSTALLMENTS_FR",
  "CASH_ADVANCE_FREQUENCY",
  "CASH_ADVANCE_TRX",
  "PURCHASE_TRX",
  "CREDIT_LIMIT", "PAYMENTS",
  "MIN_PAYMENTS",
  "PRC_FULL_PAYMENT",
  "TENURE",
  "MONTHLY_AVG_PURCHASE",
  "MONTHLY_CASH_ADVANCE",
  "LIMIT_USAGE",
  "MIN_PAYMENT_RATIO"
)


cluster_profiling2 <- t(tt2)

write.csv(cluster_profiling2,'cluster_profilingF.csv')
```

5.2 Python code

```
  #import all the necessary libraries

import pandas as pd
import numpy as np
import os

# For Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# To Scale our data
from sklearn.preprocessing import scale

# To perform KMeans clustering
from sklearn.cluster import KMeans

# To perform Hierarchical clustering
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree

#Let's read the dataset first
###########working directory################
os.chdir("G:/Data Science Project ")
credit_df= pd.read_csv("credit-card-data.csv")
credit_df.head()
credit_df.shape
CUST_ID=pd.DataFrame(CUST_ID,columns=['CUST_ID'])
# Write your code for dropping the custid columns here.
credit = credit_df.drop('CUST_ID', axis=1)
credit ['CREDIT_LIMIT'].fillna(credit ['CREDIT_LIMIT'].median(),inplace=True)
credit['MINIMUM_PAYMENTS'].fillna(credit['MINIMUM_PAYMENTS'].median(),inplace=
True)
credit.isnull().sum()
credit ['Monthly_avg_purchase']= credit['PURCHASES']/dat1['TENURE']
credit ['Monthly_cash_advance']= credit['CASH_ADVANCE']/ credit['TENURE']
credit['limit_usage']= credit.apply(lambda x: x['BALANCE']/x['CREDIT_LIMIT'], axis=1)
credit['payment_minpay']= credit.apply(lambda
x:x['PAYMENTS']/x['MINIMUM_PAYMENTS'],axis=1)
credit['payment_minpay'].describe()
def purchase(credit):
   if (credit['ONEOFF_PURCHASES']==0) & (credit['INSTALLMENTS_PURCHASES']==0):
      return 'none'
   if (credit['ONEOFF_PURCHASES']>0) & (credit ['INSTALLMENTS_PURCHASES']>0):
       return 'both_oneoff_installment'
   if (credit['ONEOFF_PURCHASES']>0) & (credit ['INSTALLMENTS_PURCHASES']==0):
        return 'one_off'
    if (credit['ONEOFF_PURCHASES']==0) & (credit ['INSTALLMENTS_PURCHASES']>0):
        return 'istallment'
credit['purchase_type']= credit.apply(purchase,axis=1)
credit['purchase_type'].value_counts()
#Extreme value analysis
# log tranformation
cr_log=credit.drop(['purchase_type'],axis=1).applymap(lambda x: np.log(x+1))
# Original dataset with categorical column converted to number type.
cre_original=pd.concat([dat1,pd.get_dummies(dat1['purchase_type'])],axis=1)
cr_dummy=pd.concat([cr_pre,pd.get_dummies(cr_pre['purchase_type'])],axis=1)
## First let us see if we can explain the dataset using fewer variables
from sklearn.preprocessing import StandardScaler
standard_scaler = StandardScaler()
credit_scaled = standard_scaler.fit_transform(cr_dummy)
#Importing the PCA module
from sklearn.decomposition import PCA
pca = PCA(svd_solver='randomized', random_state=42)
pca.fit(credit_scaled)
# Let's plot them to visualise how these features are loaded
%matplotlib inline
fig = plt.figure(figsize = (8,8))
plt.scatter(pcs_df.PC1, pcs_df.PC2)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
for i, txt in enumerate(pcs_df.Feature):
    plt.annotate(txt, (pcs_df.PC1[i],pcs_df.PC2[i]))
plt.tight_layout()
plt.show()
#Finally let's go ahead and do dimenstionality reduction using the five Principal
Components
from sklearn.decomposition import IncrementalPCA
pca_final = IncrementalPCA(n_components=5)
#Creating a transpose so that the each column is properly arranged
pc = np.transpose(df_pca)
#Calculating the Hopkins statistic
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
import numpy as np
from math import isnan

def hopkins(X):
    d = X.shape[1]
    #d = len(vars) # columns
    n = len(X) # rows
    m = int(0.1 * n)
    nbrs = NearestNeighbors(n_neighbors=1).fit(X.values)

    rand_X = sample(range(0, n, 1), m)

    ujd = []
    wjd = []
    for j in range(0, m):
        u_dist, _ =
nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).reshape(1, -1), 2,
return_distance=True)
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2,
return_distance=True)
        wjd.append(w_dist[0][1])

    H = sum(ujd) / (sum(ujd) + sum(wjd))
    if isnan(H):
        print(ujd, wjd)
        H = 0

    return H

#Let's check the Hopkins measure
hopkins(pcs_df2)
#Here also we're seeing a distinct bend at around 4 clusters. Hence it seems a good K to
choose.
#Let's perform K means using K=4
model_clus2 = KMeans(n_clusters = 4, max_iter=50,random_state = 50)
model_clus2.fit(dat3_1)
sns.scatterplot(x='PC1',y='PC2',hue='ClusterID',legend='full',data=dat_km)
####preparing profile for clustering with k=4
maindata=pd.merge(cre_original, bigdata, on='CUST_ID', how='inner')
clu_balance = pd.DataFrame(maindata1.groupby(["ClusterID"]).BALANCE.mean())
clu_purchases= pd.DataFrame(maindata1.groupby(["ClusterID"]).PURCHASES.mean())
clu_purchasefreq =
pd.DataFrame(maindata1.groupby(["ClusterID"]).PURCHASES_FREQUENCY.mean())
clu_purchasetrx =
pd.DataFrame(maindata1.groupby(["ClusterID"]).PURCHASES_TRX.mean())
clu_Monthlyavgpurchase =
pd.DataFrame(maindata1.groupby(["ClusterID"]).Monthly_avg_purchase.mean())
clu_Monthlycashadvance =
pd.DataFrame(maindata1.groupby(["ClusterID"]).Monthly_cash_advance.mean())
clu_limitusage = pd.DataFrame(maindata1.groupby(["ClusterID"]).limit_usage.mean())
clu_cashadvtrx =
pd.DataFrame(maindata1.groupby(["ClusterID"]).CASH_ADVANCE_TRX.mean())
clu_payminpaymentratio =
pd.DataFrame(maindata1.groupby(["ClusterID"]).payment_minpay.mean())
```

```
clu_both =
pd.DataFrame(maindata1.groupby(["ClusterID"]).both_oneoff_installment.mean())
clu_installment = pd.DataFrame(maindata1.groupby(["ClusterID"]).istallment.mean())
clu_one_off = pd.DataFrame(maindata1.groupby(["ClusterID"]).one_off.mean())
clu_none = pd.DataFrame(maindata1.groupby(["ClusterID"]).none.mean())
clu_creditlimit = pd.DataFrame(maindata1.groupby(["ClusterID"]).CREDIT_LIMIT.mean())
df =
pd.concat([pd.Series([0,1,2,3]),clu_balance,clu_purchases,clu_purchasefreq,clu_purchas
etrx,clu_Monthlyavgpurchase,clu_Monthlycashadvance,clu_limitusage,clu_cashadvtrx,cl
u_payminpaymentratio,clu_both,clu_installment,clu_one_off,clu_none,clu_creditlimit],
axis=1)
# heirarchical clustering
mergings = linkage(pcs_df2, method = "single", metric='euclidean')
dendrogram(mergings)
plt.show()
clusterCut = pd.Series(cut_tree(mergings, n_clusters = 4).reshape(-1,))
clusterCut.to_frame()
dat_hc_clus = pd.concat([dat_hc, clusterCut], axis=1)
df_hc.to_csv('hierarchical.csv',index=True)
df.to_csv('kmeans.csv',index=True)
```