GREAT LEARNING

PROJECT
DATA MINING

PREPARED BY : NIKHIL RAMPURIA

PROBLEM 1: CLUSTERING

PROBLEM 2: CART-RF-ANN

Problem 1: Clustering

A leading bank wants to develop a customer segmentation to give promotional offers to its customers. They collected a sample that summarizes the activities of users during the past few months. You are given the task to identify the segments based on credit card usage.
1.1 Read the data and do exploratory data analysis. Describe the data briefly.

1.2 Do you think scaling is necessary for clustering in this case? Justify

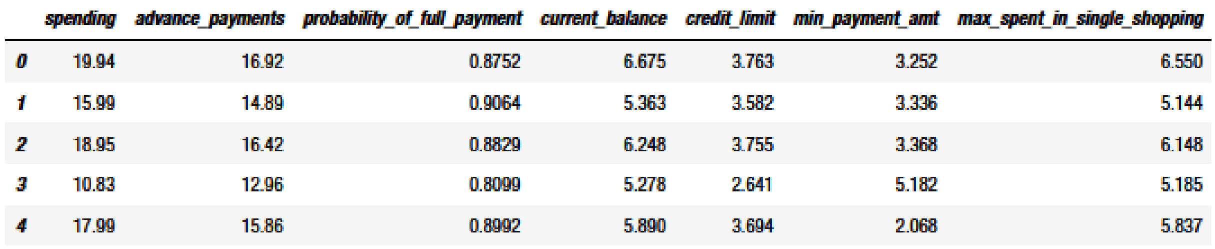
1.3 Apply hierarchical clustering to scaled data. Identify the number of optimum clusters using Dendrogram and briefly describe them

1.4 Apply K-Means clustering on scaled data and determine optimum clusters. Apply elbow curve and silhouette score.

1.5 Describe cluster profiles for the clusters defined. Recommend different promotional strategies for different clusters.

1.1 Read the data and do exploratory data analysis. Describe the data briefly.

Import Dataset:

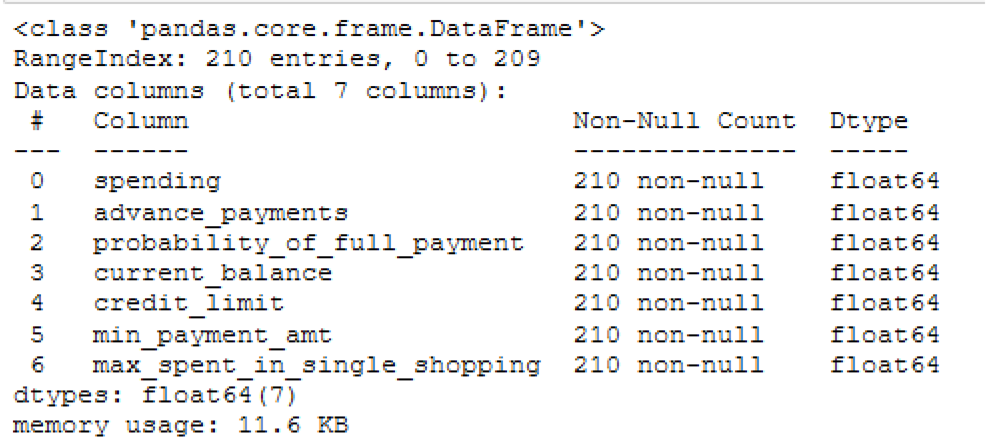


	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.550
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144
2	18.95	16.42	0.8829	6.248	3.755	3.368	6.148
3	10.83	12.96	0.8099	5.278	2.641	5.182	5.185
4	17.99	15.86	0.8992	5.890	3.694	2.068	5.837

Shape :

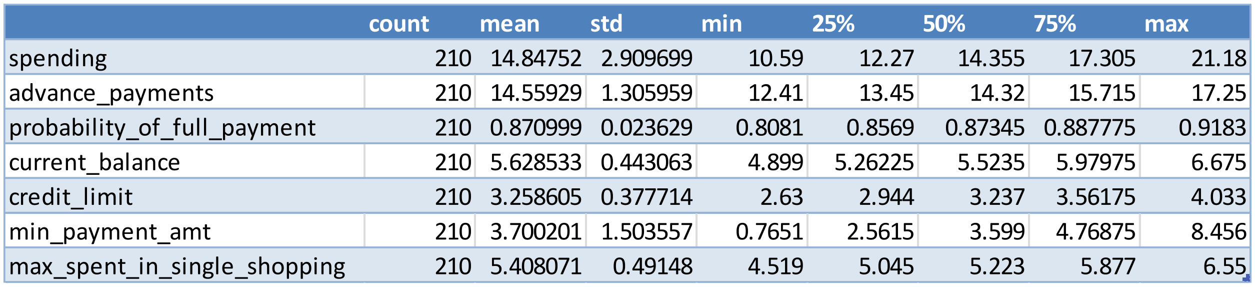
(210, 7)

Information of the dataframe:



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210 entries, 0 to 209
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   spending                      210 non-null    float64
 1   advance_payments              210 non-null    float64
 2   probability_of_full_payment   210 non-null    float64
 3   current_balance               210 non-null    float64
 4   credit_limit                  210 non-null    float64
 5   min_payment_amt               210 non-null    float64
 6   max_spent_in_single_shopping  210 non-null    float64
dtypes: float64(7)
memory usage: 11.6 KB
```

Summary of the dataframe:



	count	mean	std	min	25%	50%	75%	max
spending	210	14.84752	2.909699	10.59	12.27	14.355	17.305	21.18
advance_payments	210	14.55929	1.305959	12.41	13.45	14.32	15.715	17.25
probability_of_full_payment	210	0.870999	0.023629	0.8081	0.8569	0.87345	0.887775	0.9183
current_balance	210	5.628533	0.443063	4.899	5.26225	5.5235	5.97975	6.675
credit_limit	210	3.258605	0.377714	2.63	2.944	3.237	3.56175	4.033
min_payment_amt	210	3.700201	1.503557	0.7651	2.5615	3.599	4.76875	8.456
max_spent_in_single_shopping	210	5.408071	0.49148	4.519	5.045	5.223	5.877	6.55

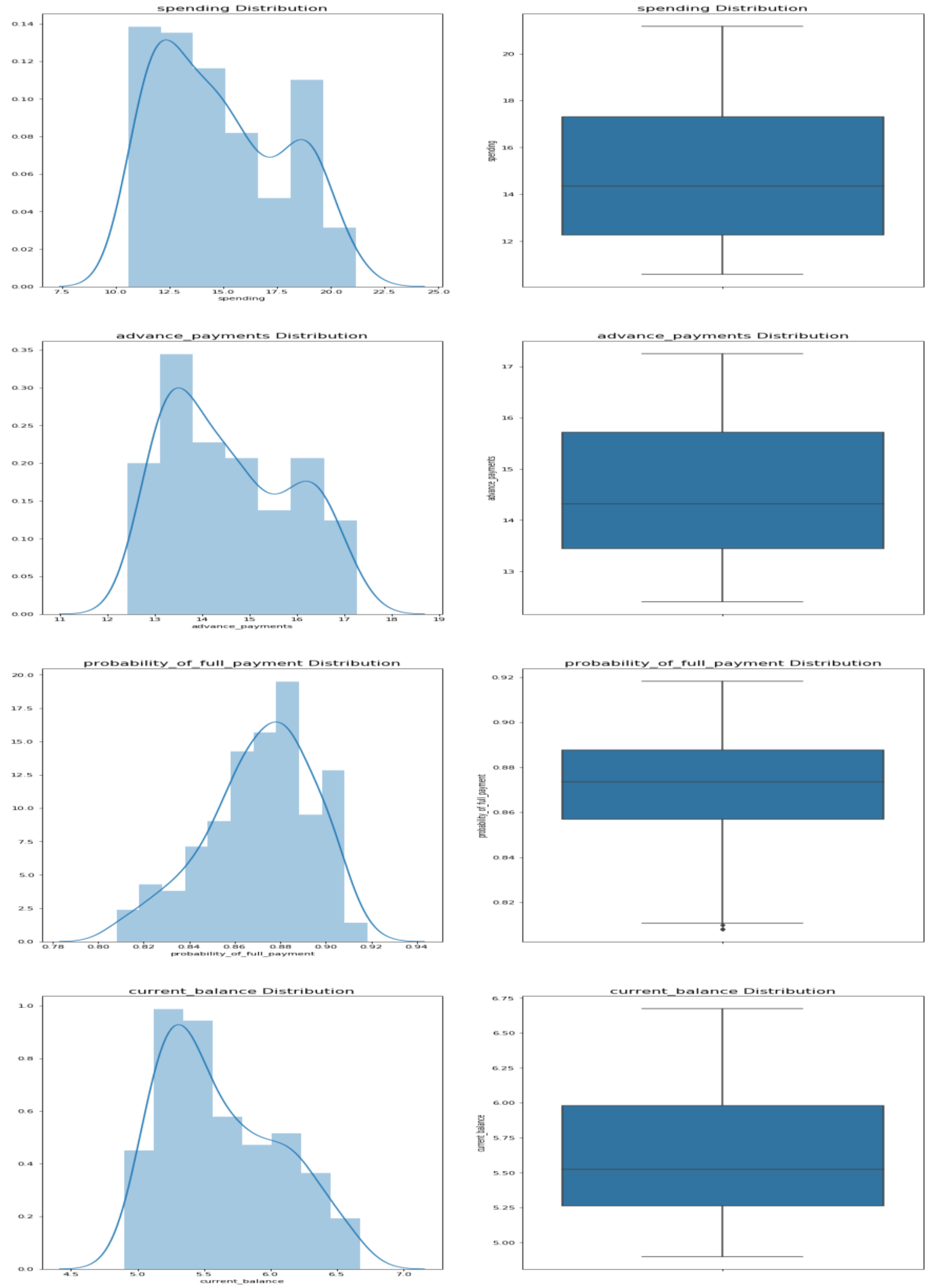
Checking for missing value:

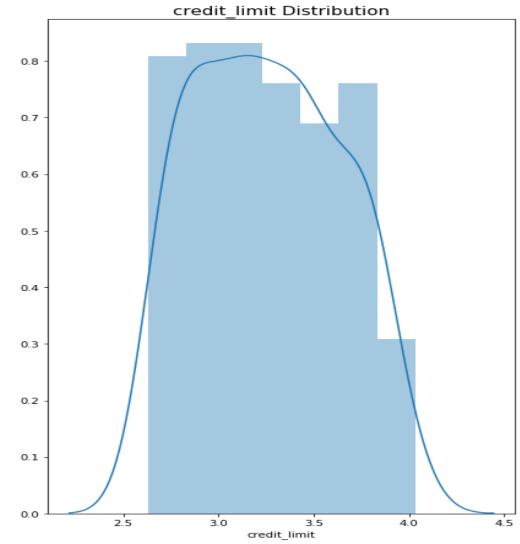
```
spending                        0
advance_payments                0
probability_of_full_payment     0
current_balance                 0
credit_limit                    0
min_payment_amt                 0
max_spent_in_single_shopping    0
dtype: int64
```

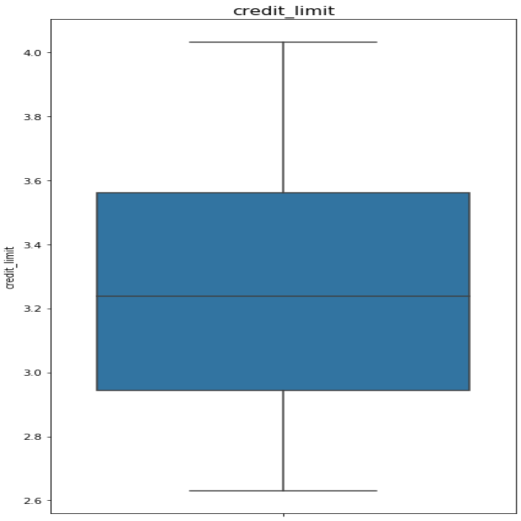
There are no missing values present

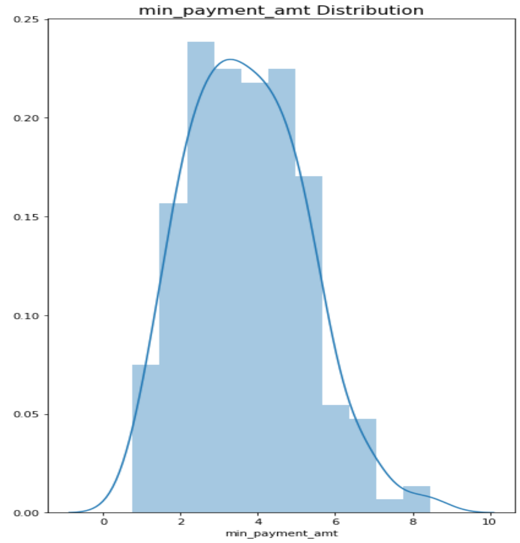
Univariate Analysis : BOX PLOT AND DIST PLOT

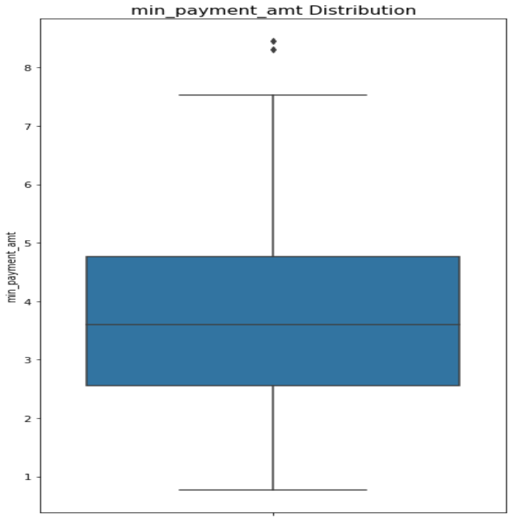
Boxplot gives us a good indication of how the values in the data are spread out and also tells us if any outlier is present.
Displot shows us univariant set of observations .

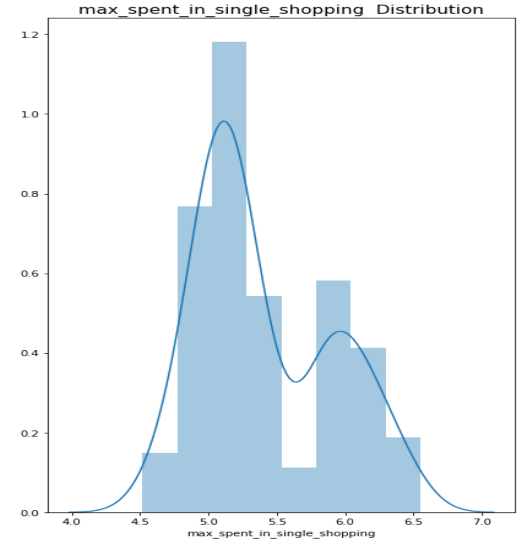


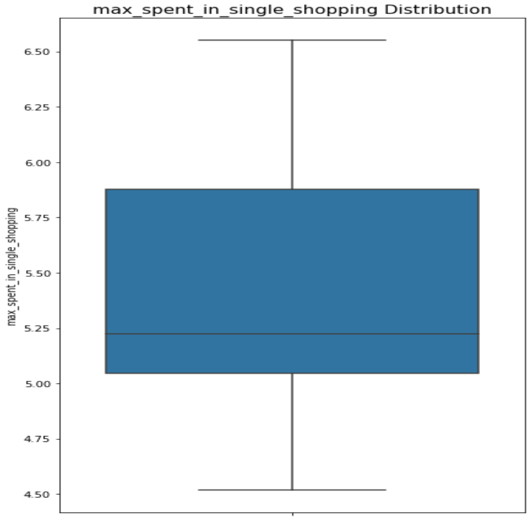

credit_limit Distribution
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
2.5
3.0
3.5
4.0
4.5
credit_limit


credit_limit
credit_limit
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0


min_payment_amt Distribution
0.00
0.05
0.10
0.15
0.20
0.25
0
2
4
6
8
10
min_payment_amt


min_payment_amt Distribution
min_payment_amt
1
2
3
4
5
6
7
8


max_spent_in_single_shopping Distribution
0.0
0.2
0.4
0.6
0.8
1.0
1.2
4.0
4.5
5.0
5.5
6.0
6.5
7.0
max_spent_in_single_shopping


max_spent_in_single_shopping Distribution
max_spent_in_single_shopping
4.50
4.75
5.00
5.25
5.50
5.75
6.00
6.25
6.50

SKEWNESS:

```
spending                    0.399889
advance_payments            0.386573
probability_of_full_payment   -0.537954
current_balance             0.525482
credit_limit                0.134378
min_payment_amt             0.401667
max_spent_in_single_shopping   0.561897
dtype: float64
```

Univariate analysis refer to the analysis of a single variable.

The purpose of univariate analysis is to summarize and find patterns in the data.

No null value found

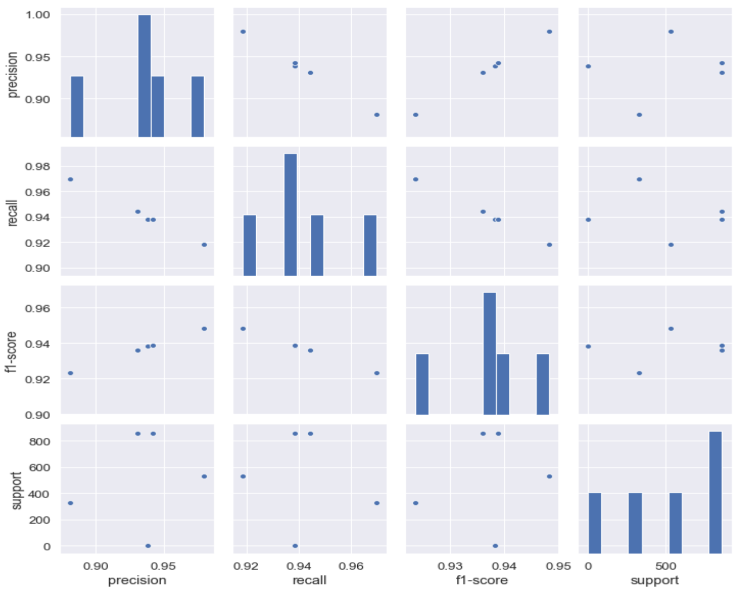
No missing value found

Outlier is present only in 1 variable i.e min_payment_amt which means that there are only a few customers whose minimum payment amount falls on the higher side on an average and probability_of_full_payment is in decimal which means there are few customers whose probability to pay full to the bank is on the lower side of average.

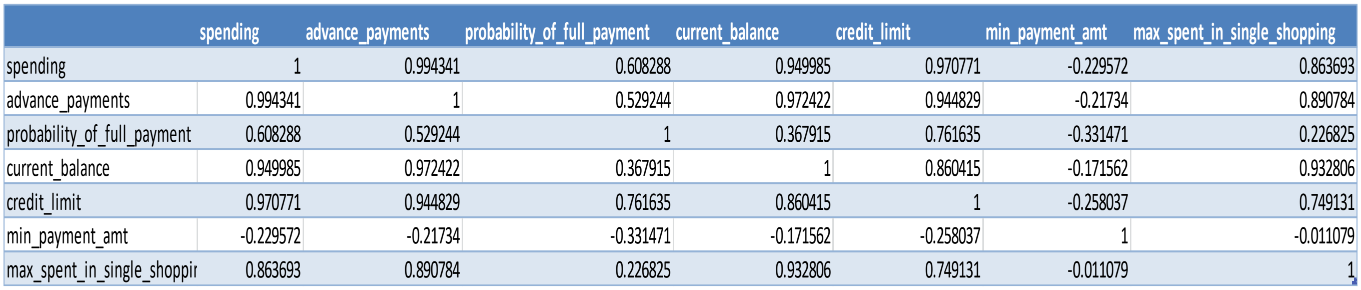
Since only these variable have a very small outlier value, hence there is no need to treat the outliers.

We may conclude that most of the customers have a higher spending capacity, and most of the customers have a higher probability to make full payment to the bank.

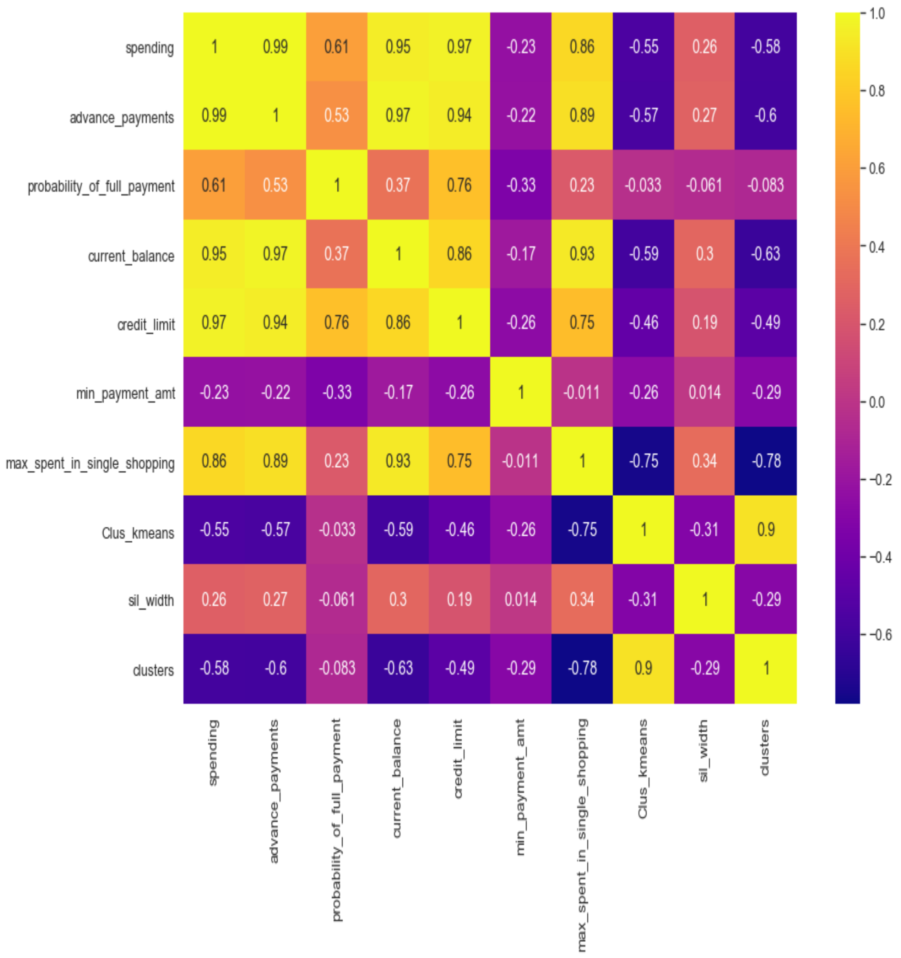
Multivariate Analysis:



Correlation



	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
spending	1	0.994341	0.608288	0.949985	0.970771	-0.229572	0.863693
advance_payments	0.994341	1	0.529244	0.972422	0.944829	-0.21734	0.890784
probability_of_full_payment	0.608288	0.529244	1	0.367915	0.761635	-0.331471	0.226825
current_balance	0.949985	0.972422	0.367915	1	0.860415	-0.171562	0.932806
credit_limit	0.970771	0.944829	0.761635	0.860415	1	-0.258037	0.749131
min_payment_amt	-0.229572	-0.21734	-0.331471	-0.171562	-0.258037	1	-0.011079
max_spent_in_single_shoppir	0.863693	0.890784	0.226825	0.932806	0.749131	-0.011079	1



From the above Heatmap we can infer that the:
spending and advance_payments are highly correlated spending
current_balance are highly correlated spending and credit_limit are highly correlated
Advance_payment and current_balance are highly correlated
Advance_payment and credit_limit are highly correlated
Advance_payment and max_spent_in_single_shopping are highly correlated
current_balance and max_spent_in_single_shopping are highly correlated

So by this we may infer that the customers who have high credit limit spends more.
Also the customers who pays the money in cash also spends more and the customer who have high Balance amount left in the account to make purchases spends more. min_payment_amt is not correlated with any of the variable so it will not affect by any changes in spending, current_balance or credit limit of the account
Probability of full payments is higher for those customers who have a higher credit limit.

1.2 Do you think scaling is necessary for clustering in this case? Justify

Yes, the Data Scaling is necessary in this case.

For this data given to us ,Clearly these variables are measuring very different from each other and all the variables are expressed in different units such as spending in 1000's, advance payments in 100's and credit limit in 10000's and probability_of_full_payment is expressed as fraction and thus have very different scales.
If we perform cluster analysis on this data, difference in one variable will most likely dominate the other 2 variables simply because of the scale.
All these different variables need to be converted to one scale in order to perform meaningful analysis.

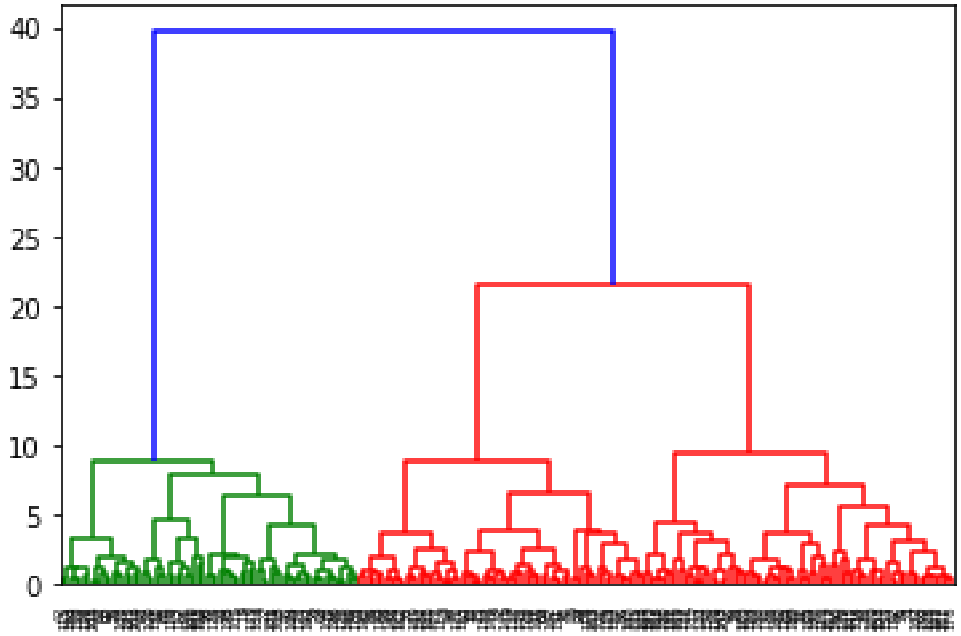
Scaled data:

```
array([[ 1.75435461,  1.81196782,  0.17822987, ...,  1.33857863,
        -0.29880602,  2.3289982 ],
       [ 0.39358228,  0.25383997,  1.501773  , ...,  0.85823561,
        -0.24280501, -0.53858174],
       [ 1.41330028,  1.42819249,  0.50487353, ...,  1.317348  ,
        -0.22147129,  1.50910692],
       ...,
       [-0.2816364 , -0.30647202,  0.36488339, ..., -0.15287318,
        -1.3221578 , -0.83023461],
       [ 0.43836719,  0.33827054,  1.23027698, ...,  0.60081421,
        -0.95348449,  0.07123789],
       [ 0.24889256,  0.45340314, -0.77624835, ..., -0.07325831,
        -0.70681338,  0.96047321]])
```

1.3 Apply hierarchical clustering to scaled data. Identify the number of optimum clusters using Dendrogram and briefly describe them

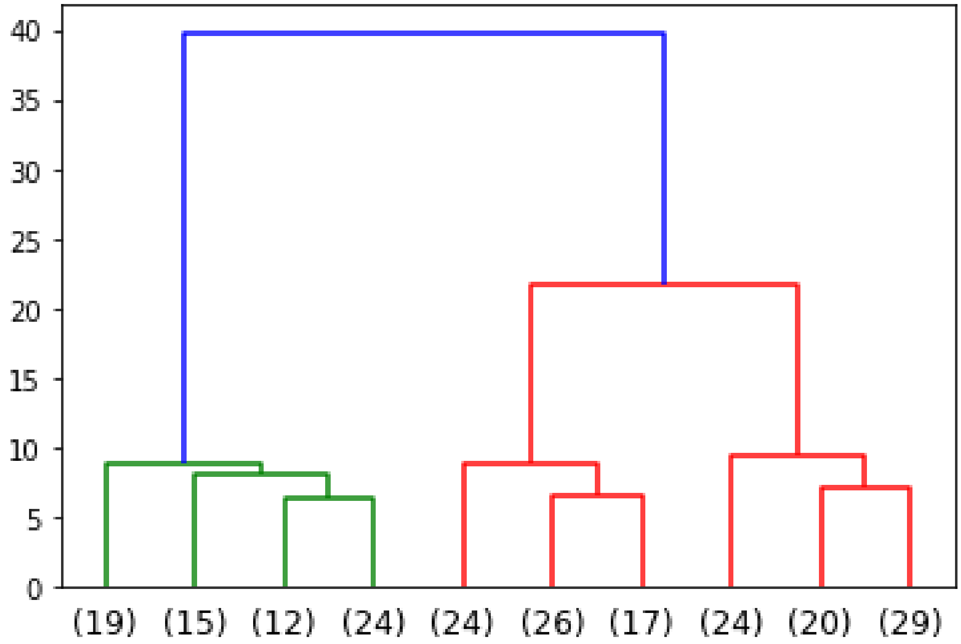
Clustering,is the extraction of natural groupings of similar data objects.
Hierarchical clustering relies using clustering techniques to find a hierarchy of clusters, where this hierarchy resembles a dendrogram.
Agglomerative clustering uses a bottom-up approach.
For the dataset in question we will be using Hierarchical Clustering method to create optimum clusters and categorising the dataset on the basis of these clusters.

```
dend = dendrogram(wardlink)
```



Cutting the Dendrogram with suitable clusters:

```
dend = dendrogram(wardlink,truncate_mode='lastp',p=10,)
```



We choose truncate mode to select the last p clusters. In green cluster we have 19+15+12+24 = 70 products In red we have 24+26+17+24+20+29 = 140 products.

From above Dendrogram we may see that optimum number of Clusters may be taken as 3 and maximum number of customer falls in red cluster.

There are two methods to find out the Clustering of the dataset

Method 1: criterion = 'maxclust' where a cut is defined based on the number of clusters.

```
Cluster_1 = fcluster(wardlink, 2, criterion='maxclust')
```

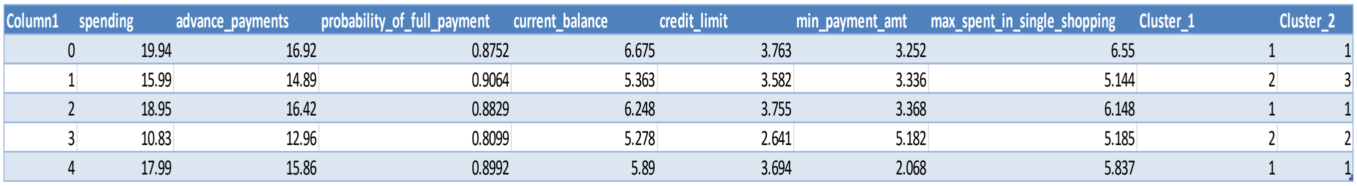
```
array([1, 2, 1, 2, 1, 2, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2,
       1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1,
       2, 2, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1,
       1, 2, 1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1,
       1, 2, 2, 1, 2, 2, 2, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1,
       2, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2,
       2, 1, 2, 1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1,
       2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 2,
       1, 2, 2, 1, 2, 2, 1, 2, 1, 2, 1, 2], dtype=int32)
```

Method 2: criterion = 'distance' where a cut is defined based on distance in the y-axis.

```
Cluster_2 = fcluster(wardlink,10, criterion='distance')
array([1, 3, 1, 2, 1, 2, 2, 3, 1, 2, 1, 3, 2, 1, 3, 2, 3, 2, 3, 2, 2, 2,
       1, 2, 3, 1, 3, 2, 2, 2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 1, 1, 3, 1, 1,
       2, 2, 3, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 3, 2, 2, 3, 3, 1,
       1, 3, 1, 2, 3, 2, 1, 1, 2, 1, 3, 2, 1, 3, 3, 3, 3, 1, 2, 3, 3, 1,
       1, 2, 3, 1, 3, 2, 2, 1, 1, 1, 2, 1, 2, 1, 3, 1, 3, 1, 1, 2, 2, 1,
       3, 3, 1, 2, 2, 1, 3, 3, 2, 1, 3, 2, 2, 2, 3, 3, 1, 2, 3, 3, 2, 3,
       3, 1, 2, 1, 1, 2, 1, 3, 3, 3, 2, 2, 3, 2, 1, 2, 3, 2, 3, 2, 3, 3,
       3, 3, 3, 2, 3, 1, 1, 2, 1, 1, 1, 2, 1, 3, 3, 3, 3, 2, 3, 1, 1, 1,
       3, 3, 1, 2, 3, 3, 3, 3, 1, 1, 3, 3, 3, 2, 3, 3, 2, 1, 3, 1, 1, 2,
       1, 2, 3, 1, 3, 2, 1, 3, 1, 3, 1, 3], dtype=int32)
```

We have successfully divided observations into two clusters .
We can see that both the criterion have resulted in the same output.
With Wardlink method we used criterion as maxclust and distance to find the optimal number of clusters. When you look at the dendrogram, it seems that 2 clusters would be optimal. But with distance criterion and value as 10, we see that that the optimal number of clusters shall be 3.
By cutting a horizontal line at the value 10 mentioned gives the optimal number of clusters.

Appending clusters to original dataset for further analysis

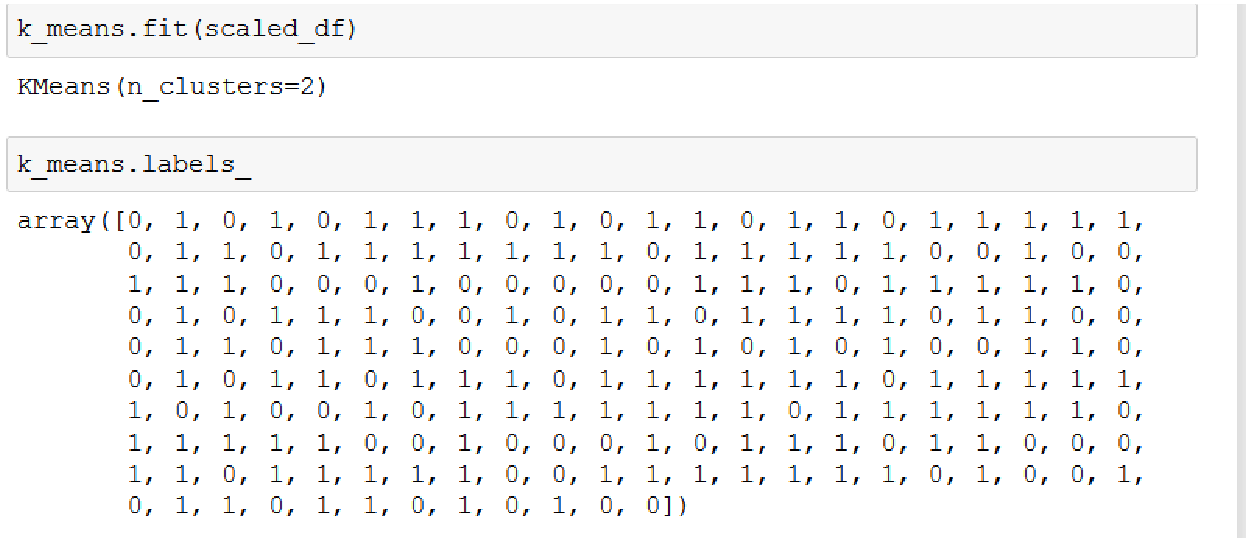


Column1	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Cluster_1	Cluster_2
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.55	1	1
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144	2	3
2	18.95	16.42	0.8829	6.248	3.755	3.368	6.148	1	1
3	10.83	12.96	0.8099	5.278	2.641	5.182	5.185	2	2
4	17.99	15.86	0.8992	5.89	3.694	2.068	5.837	1	1

1.4 Apply K-Means clustering on scaled data and determine optimum clusters. Apply elbow curve and silhouette score.

K-means is a centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid.
The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid

For the dataset we will be using K-means clustering on scaled data and identify the clusters formed .
Then we will calculate the value of inertia and store it in WSS

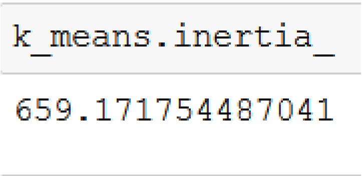


```
k_means.fit(scaled_df)
```

```
KMeans(n_clusters=2)
```

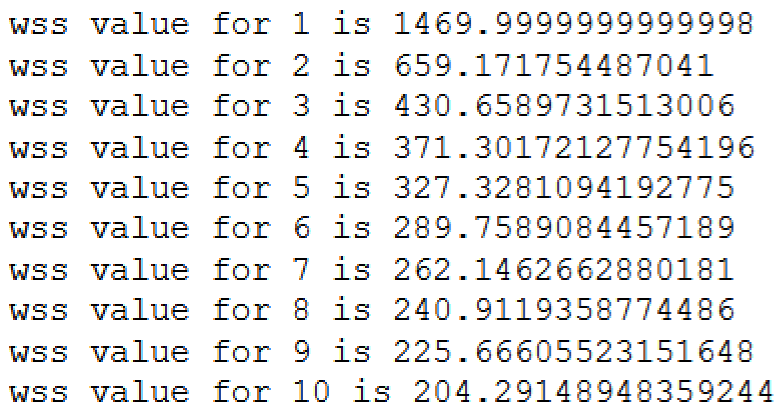
```
k_means.labels_
```

```
array([0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0])
```



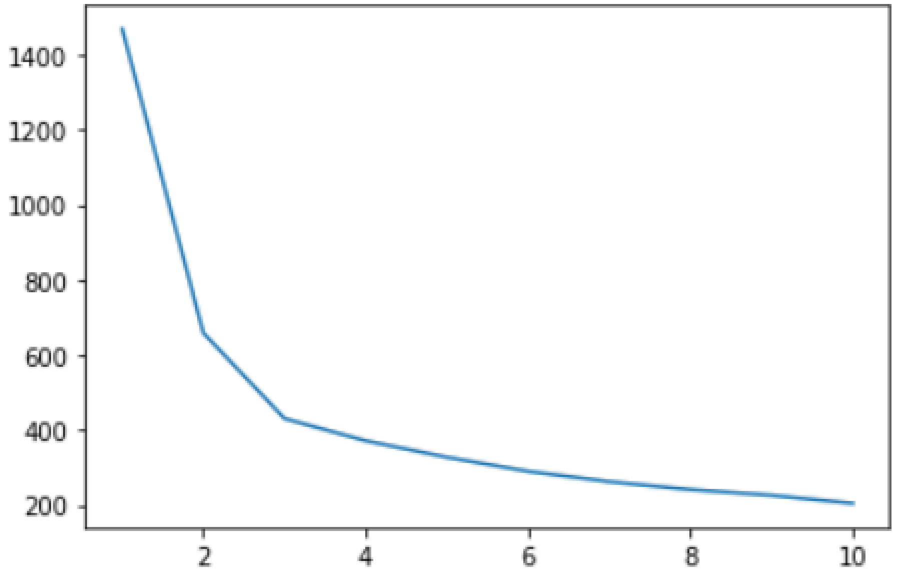
```
k_means.inertia_
```

```
659.171754487041
```



```
wss value for 1 is 1469.9999999999998
wss value for 2 is 659.171754487041
wss value for 3 is 430.6589731513006
wss value for 4 is 371.30172127754196
wss value for 5 is 327.3281094192775
wss value for 6 is 289.7589084457189
wss value for 7 is 262.1462662880181
wss value for 8 is 240.9119358774486
wss value for 9 is 225.66605523151648
wss value for 10 is 204.29148948359244
```

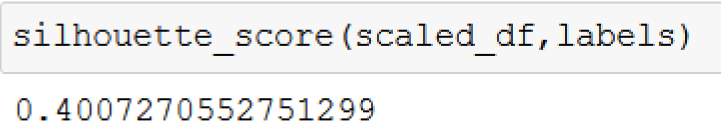
We will now plot two curves to determine the optimal number of clusters to be used for our clustering. The two methods to determine the optimal number of clusters are within sum of squares(wss) method and average silhouette scores method.



As per the above wss plot we can conclude that the optimal number of clusters to be taken for k-means clustering is 3 since as per the elbow it can be easily seen in the curve that after 3 the curve gets flat.

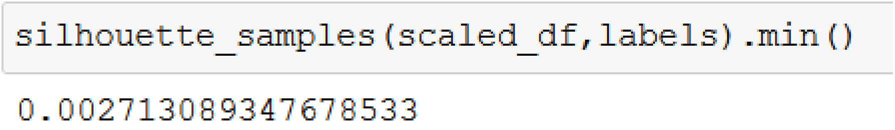
Calculating the silhouette scores and silhouette width :

When we compared silhouette score for n_clusters = 3, n_clusters = 4 and n_clusters = 5 . n_clusters = 3 gives the highest value so we can conclude that the optimal number of clusters to be taken for k-means clustering is 3.



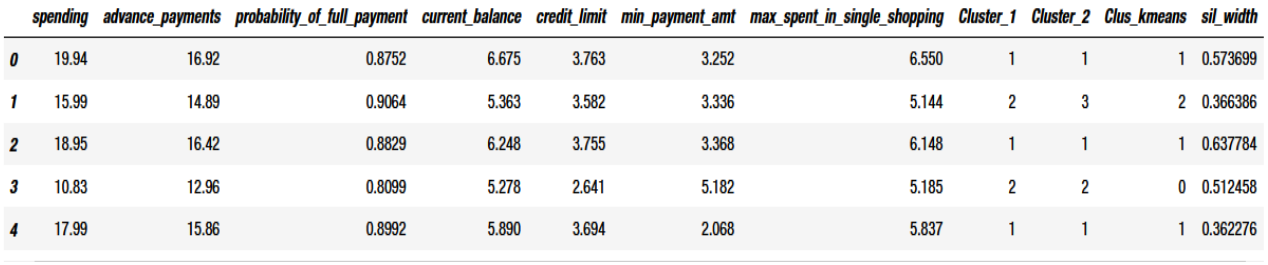
```
silhouette_score(scaled_df,labels)
```

```
0.4007270552751299
```



```
silhouette_samples(scaled_df,labels).min()
```

```
0.002713089347678533
```



	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Cluster_1	Cluster_2	Clus_kmeans	sil_width
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.550	1	1	1	0.573699
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144	2	3	2	0.366386
2	18.95	16.42	0.8829	6.248	3.755	3.368	6.148	1	1	1	0.637784
3	10.83	12.96	0.8099	5.278	2.641	5.182	5.185	2	2	0	0.512458
4	17.99	15.86	0.8992	5.890	3.694	2.068	5.837	1	1	1	0.362276

As per wss method and silhouette score we can conclude that the optimal number of k or clusters that needs to be taken for k-means clustering is 3.

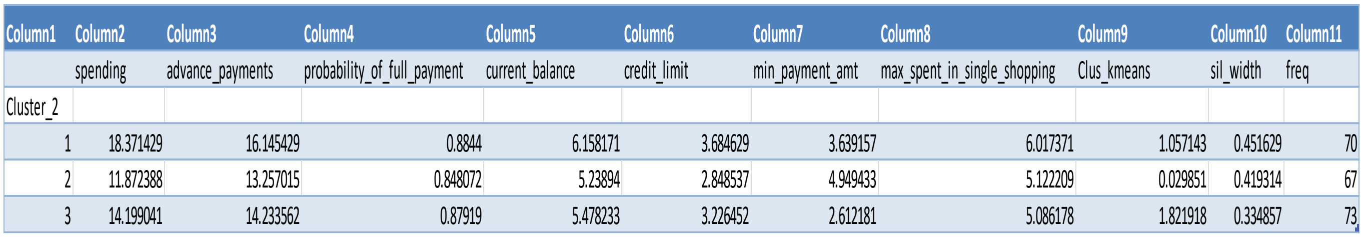
The silhouette scores is calculated as 0.400727 and silhouette widths is calculated using silhouette_samples The minimum silhouette score is 0.0027. The silhouette score ranges from -1 to +1 and higher the silhouette score better the clustering.

The silhouette score 0.4007 indicates that the object is well matched to its own cluster.

1.5 Describe cluster profiles for the clusters defined. Recommend different promotional strategies for different clusters.

Observations for Hierarchical Clustering:

Cluster profile:



Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11
	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Clus_kmeans	sil_width	freq
Cluster_2										
1	18.371429	16.145429	0.8844	6.158171	3.684629	3.639157	6.017371	1.057143	0.451629	70
2	11.872388	13.257015	0.848072	5.23894	2.848537	4.949433	5.122209	0.029851	0.419314	67
3	14.199041	14.233562	0.87919	5.478233	3.226452	2.612181	5.086178	1.821918	0.334857	73

For Cluster 1,

Spending is highest, averaging 18371 which is highest among three clusters.

Highest advance payments around 1614 which is highest among three clusters

Probability of Full Payment is very high, averaging around 0.8844 which is highest among three clusters.

Current Balance is around 6158 which is highest among three clusters.

Credit Limit is also high 36846 which is highest among three clusters.

min_payment_amt is 363

And also, max_spent_in_single_shopping is around 6017, which is higher comparatively to other clusters.

For Cluster 2,

The average Spending of this cluster is on lower side, averaging 11872.

Highest advance payments around 1325 which is lowest among the three clusters.

Probability of Full Payment is the least amongst other clusters, averaging around 0.848.

Current Balance is around 5238 which is least among three clusters.

Credit Limit is least for this cluster ranging around 28485.

min_payment_amt is 494 which is max in this cluster.

max_spent_in_single_shopping is around 5122.

For Cluster 3,

Average Spending of this cluster is is 14199.

Highest advance payments around 1423

Probability of Full Payment is on the higher side, averaging around 0.879.

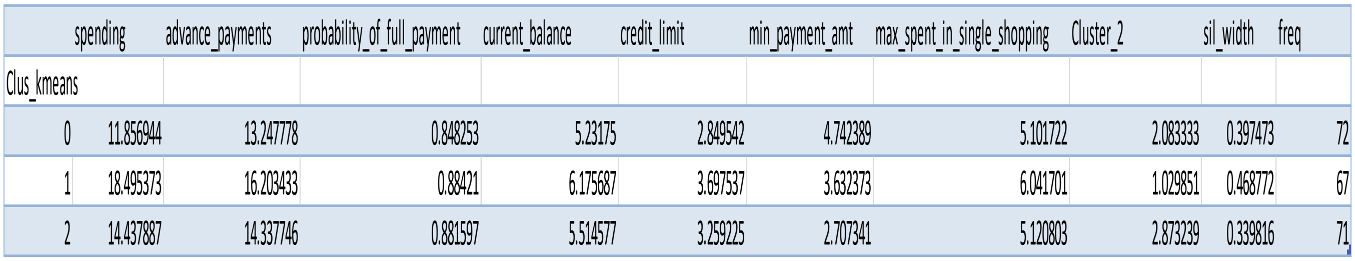
Current Balance is around 5478 which is average among three clusters.

Credit Limit is around 32264 which is average among three clusters.

min_payment_amt is 261 which is least among three cluster.

Max_spent_in_single_shopping is the least around 5086.

Observations for K-means Clustering:



	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Cluster_2	sil_width	freq
Clus_kmeans										
0	11.856944	13.247778	0.848253	5.23175	2.849542	4.742389	5.101722	2.083333	0.397473	72
1	18.495373	16.203433	0.88421	6.175687	3.697537	3.632373	6.041701	1.029851	0.468772	67
2	14.437887	14.337746	0.881597	5.514577	3.259225	2.707341	5.120803	2.873239	0.339816	71

For Cluster 1 (Clus_kmeans =0),

Spending for this cluster is least among all the three clusters, averaging 11856.

advance payments around 1324 which is lowest among the three clusters.

Probability of Full Payment is the least amongst other clusters, averaging around 0.848.

Current Balance is around 5231 which is least among three clusters.

Credit Limit is least for this cluster ranging around 28495.

min_payment_amt is found max in this cluster, around 474.

Max_spent_in_single_shopping is around 5101.

For Cluster 2(Clus_kmeans =1),

Spending is highest amongst other clusters, averaging 18495.

Highest advance payments around 1620 which is highest among three clusters

Probability of Full Payment is on the higher side, averaging around 0.884.

Current Balance is around 6175 which is highest among three clusters.

Credit Limit is ranging in between for this cluster is around 36975 which is highest amongst three clusters.

min_payment_amt is found least in this cluster, it is around 363.

max_spent_in_single_shopping is the highest around 6041.

For Cluster 3 (Clus_kmeans =2),

Average Spending of this cluster is 14437.

advance payments around 1433.

Probability of Full Payment is least among other Clusters, averaging around 0.8815.

Current Balance is around 5514,which is average as compared to other two clusters.

Credit Limit for this cluster ranging around 32592.

min_payment_amt is averaging 270,which is the least amongst others.

max_spent_in_single_shopping is around 5120.

Promotional strategy :

For Hierarical Clusters:

Cluster 1 are premium customers (Top Spending Customers).Bank should focus on Cluster 1 as the customers in this cluster have higher spending,highest advance payments,highest Current Balance and highest credit limit.

This segment appears to be upper class and can be targeted using various offers such as cards with rewards and loyalty points for every spent . They can also provide credits for flight tickets.

Cluster 2 are low spending customers (Poor spending customers has the least Credit limit and so may be they spend least and also they have least current balance.
There are probabilities that these customers may increase spending if the bank increases their Credit limit as their averages of all parameters are relatively same compared to others.Bank can also think of providing them offers for shopping at various websites that may increase their max_spent_in_single_shopping.

Cluster 3 medium customers (Medium spending customers) has average spending. Bank should give customers in this Cluster more promotional offers because there are more chances that these customers may move to Cluster 1 of high spending.

Hence Bank should provide more promotional offers for customers in Cluster 2 & 3 .

For K-means Clusters:

Bank should focus on Cluster 2 are premium customers (Top Spending Customers) as the customers in this cluster have higher spending.

This segment appears to be upper class and can be targeted using various offers such as cards with rewards and loyalty points for every spent.They can also provide credits for flight tickets.

Cluster 1 are low spending customers (Poor spending customers) and has the least Credit limit and so may be they spend least and also they have least current balance.
There are probabilities that these customers may increase spending if the bank increases their Credit limit as their averages of all parameters are relatively same compared to others.

Cluster 3 are medium customers (Medium spending customers) has average spending. Bank should give customers in this Cluster more promotional offers because there are more chances that these customers may move to Cluster 1 of high spending.

Hence Bank should provide more promotional offers for customers in Cluster 2 & 3

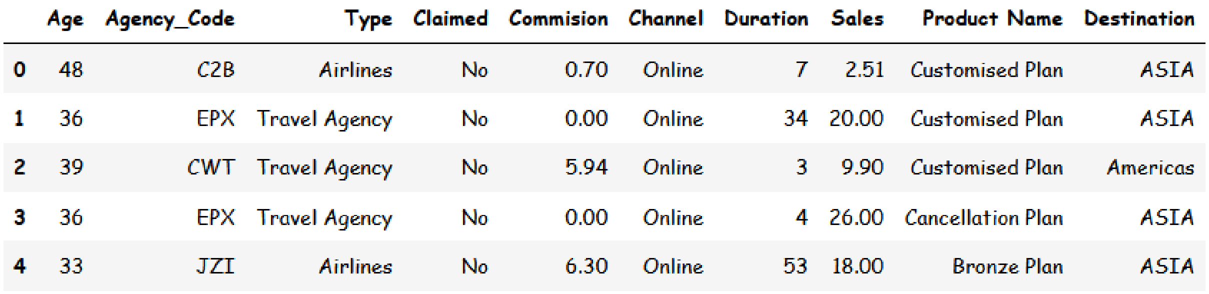
Problem 2: CART-RF-ANN

An Insurance firm providing tour insurance is facing higher claim frequency. The management decides to collect data from the past few years. You are assigned the task to make a model which predicts the claim status and provide recommendations to management. Use CART, RF & ANN and compare the models' performances in train and test sets.

2.1 Data Ingestion: Read the dataset. Do the descriptive statistics and do null value condition check, write an inference on it. 2.2 Data Split: Split the data into test and train, build classification model CART, Random Forest, Artificial Neural Network 2.3 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model 2.4 Final Model: Compare all the model and write an inference which model is best/optimized. 2.5 Inference: Basis on these predictions, what are the business insights and recommendations

2.1 Data Ingestion: Read the dataset. Do the descriptive statistics and do null value condition check, write an inference on it.

Import Dataset:

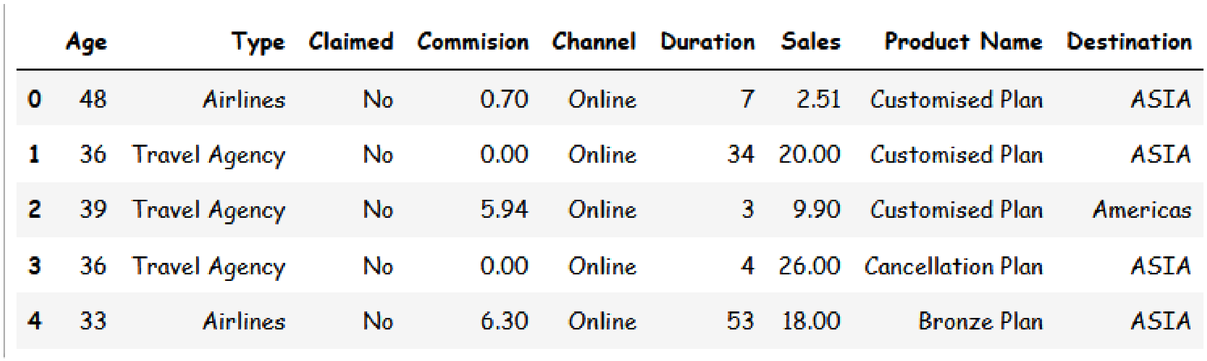


	Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
0	48	C2B	Airlines	No	0.70	Online	7	2.51	Customised Plan	ASIA
1	36	EPX	Travel Agency	No	0.00	Online	34	20.00	Customised Plan	ASIA
2	39	CWT	Travel Agency	No	5.94	Online	3	9.90	Customised Plan	Americas
3	36	EPX	Travel Agency	No	0.00	Online	4	26.00	Cancellation Plan	ASIA
4	33	JZI	Airlines	No	6.30	Online	53	18.00	Bronze Plan	ASIA

Shape :

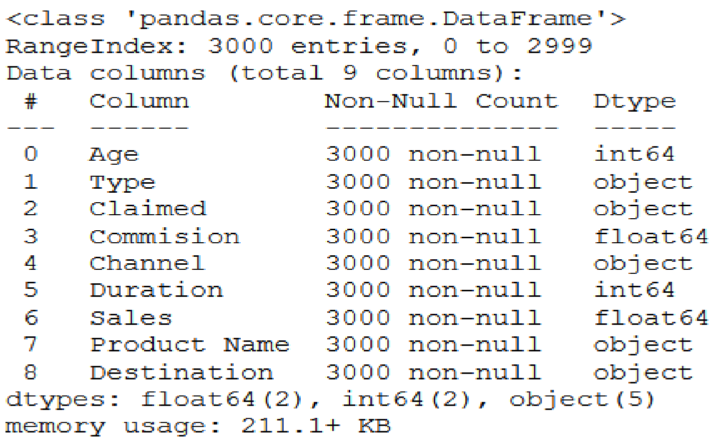
(3000, 9)

Removed the unwanted column Agency_Code:



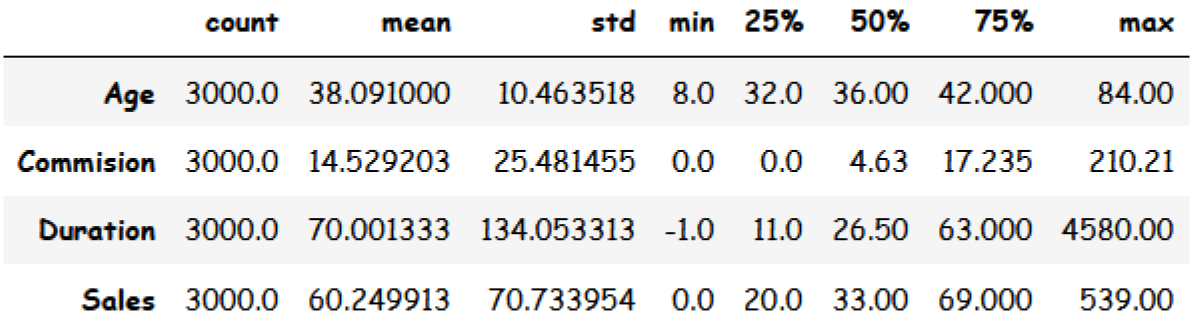
	Age	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
0	48	Airlines	No	0.70	Online	7	2.51	Customised Plan	ASIA
1	36	Travel Agency	No	0.00	Online	34	20.00	Customised Plan	ASIA
2	39	Travel Agency	No	5.94	Online	3	9.90	Customised Plan	Americas
3	36	Travel Agency	No	0.00	Online	4	26.00	Cancellation Plan	ASIA
4	33	Airlines	No	6.30	Online	53	18.00	Bronze Plan	ASIA

Information of the dataframe:



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Age           3000 non-null   int64  
 1   Type          3000 non-null   object 
 2   Claimed       3000 non-null   object 
 3   Commision     3000 non-null   float64
 4   Channel       3000 non-null   object 
 5   Duration      3000 non-null   int64  
 6   Sales         3000 non-null   float64
 7   Product Name  3000 non-null   object 
 8   Destination   3000 non-null   object 
dtypes: float64(2), int64(2), object(5)
memory usage: 211.1+ KB
```

Summary of the dataframe:

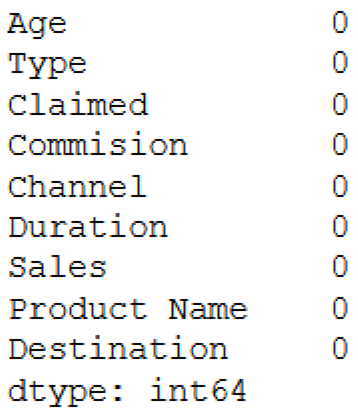


	count	mean	std	min	25%	50%	75%	max
Age	3000.0	38.091000	10.463518	8.0	32.0	36.00	42.000	84.00
Commision	3000.0	14.529203	25.481455	0.0	0.0	4.63	17.235	210.21
Duration	3000.0	70.001333	134.053313	-1.0	11.0	26.50	63.000	4580.00
Sales	3000.0	60.249913	70.733954	0.0	20.0	33.00	69.000	539.00

As there are only 4 continuous variables Age,Commision,Duration and Sales, the result is shown for them only.

Now lets take variable Duration - we can say that there are outliers as the difference between 75% (63) and max value(4580) is very high and similar case is with other variables also. Hence outlier detection and treatment is necessary for analysis on the dataset.

Checking for missing value:



```
Age             0
Type            0
Claimed         0
Commision       0
Channel         0
Duration        0
Sales           0
Product Name    0
Destination     0
dtype: int64
```

There are no missing values present

'Claimed' status is the target variable while all others are the predictors.

Out of the 9 columns,5 are object type,2 are int and 2 are float type.

Object -Type , Claimed ,Channel ,Product Name and Destination

Int -Age,Duration
float-Commision,Sales
There are no missing values in the data.

Check for duplicate data:

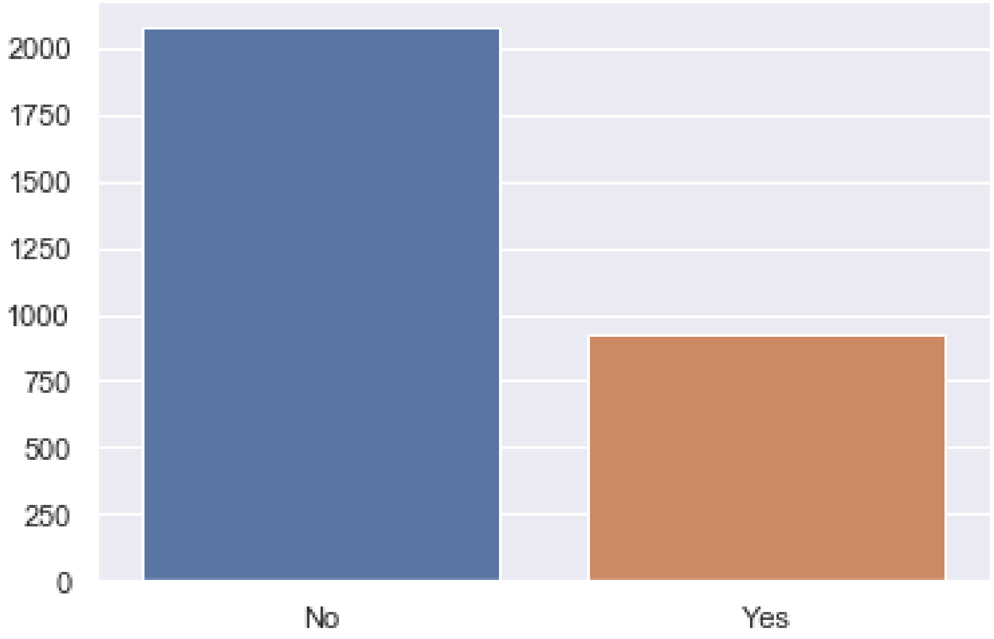
Number of duplicate rows = 139

After Removing Duplicates:

Number of duplicate rows = 0
(2861, 9)

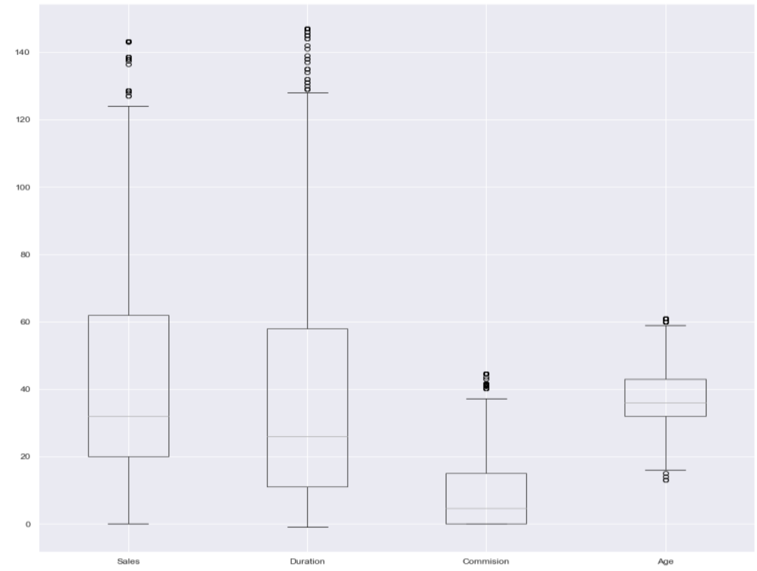
Checking the object variables to see if they are useful for the model building or we can drop the columns if the values are like serial numbers/ customer id etc by using value count function.
It is observed that all the variables are required to build the model.

As our Target Variable is Claimed, we can Plot the countplot for "Claimed"

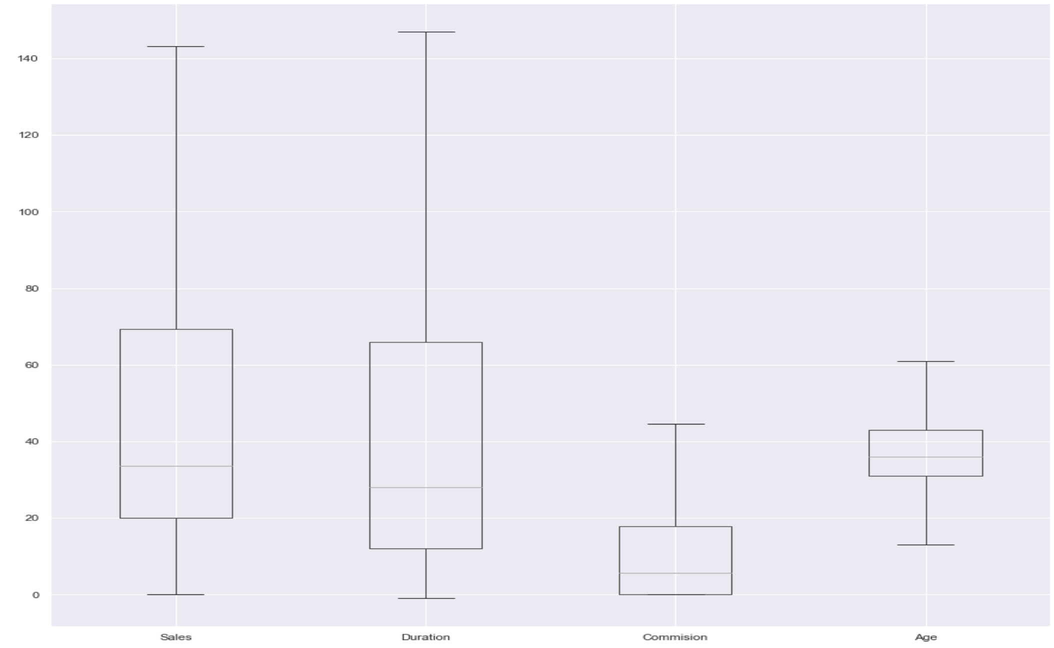


From the above value count for Claimed status No- 69.2% and Claimed status Yes- 30.8% . It shows there are a large number of people who have claimed for insurance. The data is well balanced and is fit for model building.

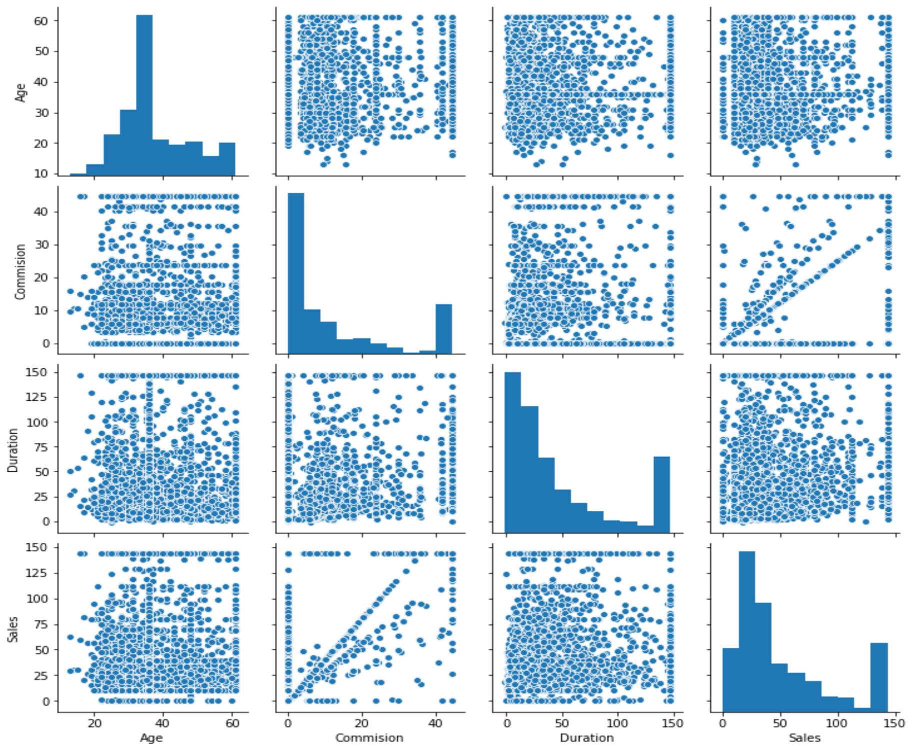
Check for outliers:



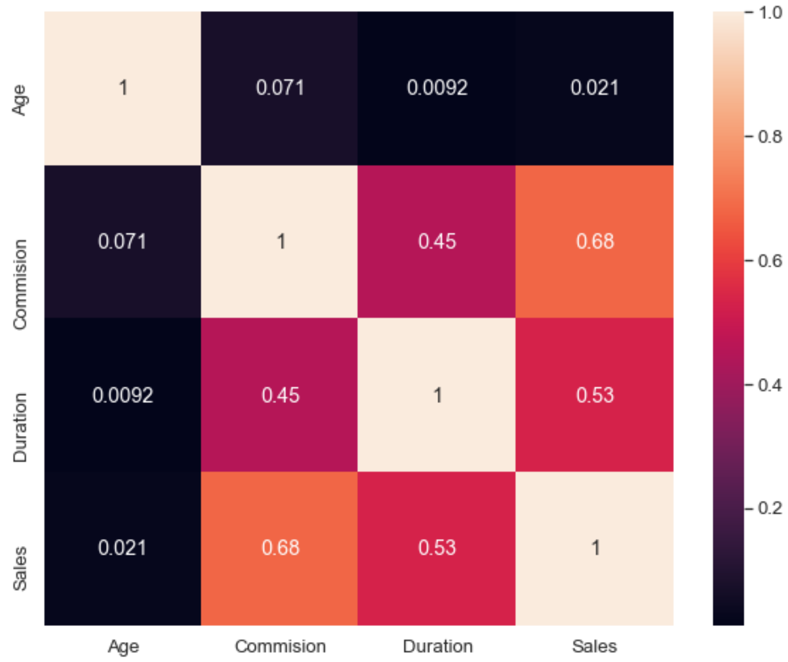
After Treating Outliers:



Checking pairwise distribution of the continuous variables:



HEATMAP:



Inference:

By checking the Boxplots for all the continous variables we can conclude that a very high number of outliers are present in all the continous variables namely, Age, Commision, Duration and Sales which means that we need to treat these outlier values so as to proceed further with our model building and analysis as these values can create errors and can deviate from the actual results.

We can conclude from the that the majority of the customers doing a claim in our data belong to age group of 30-50 years and it is observed that age group between 30-36 contribute to the highest number of claims , with the type of Tour Agency being Travel Agency, Channel being Online, Product name being Customised Plan and Destination being Asia.

2.2 Data Split: Split the data into test and train, build classification model CART, Random Forest, Artificial Neural Network

Decision tree , Random forest and ANN can take only numerical columns. It cannot take object .So we will converts all the columns with data type as object into categorical codes.

Converting all objects to categorical codes using code

```
for feature in ins_df.columns:

  if ins_df[feature].dtype == 'object':

    print('\n')

    print('feature:',feature)

    print(pd.Categorical(ins_df[feature].unique()))

    print(pd.Categorical(ins_df[feature].unique()).codes)

    ins_df[feature] = pd.Categorical(ins_df[feature]).codes
```

Proportion of 1s and 0s

```
0    0.680531
1    0.319469
```

Where 0 represents not claimed and 1 represents claimed

We will separate target variable from other variables.

Extracting the target column into separate vectors for training set and test set

```
X = ins_df.drop("Claimed", axis=1)

y = ins_df.pop("Claimed")
```

Splitting the data into Train and Test set:

from sklearn model_selection we will import train_test_split

```
X_train, X_test, train_labels, test_labels = train_test_split(X, y, test_size=.30, random_state=1)
```

Test size we have given as 0.30 as we want 30% of the data is to be tested. Random state we have given as 1,as it will give similar results when we run the same on different computers with the same random state.

Checking the dimensions of the training and test data

```
X_train (2002, 9)
X_test (859, 9)
train_labels (2002,)
test_labels (859,)
```

We may observe that the samples are almost equally distributed between
the train and test datasets with reference to target variable. So we can build the model now.

Building a Decision Tree Classifier

We will use “gini” as the criterion.

Grid search is essentially an optimization algorithm which lets you select the best parameters for your optimization problem from a list of parameter options that you provide, hence automating the 'trial-and-error' method.
Now we will give different values for each parameter to this and apply the same to our data model.
The command used for the same is

```
param_grid = {

   'criterion': ['gini'],

   'max_depth': [10,20,30,50],

   'min_samples_leaf': [50,100,150],

   'min_samples_split': [150,300,450],

}
```

```
dtcl = DecisionTreeClassifier(random_state=1)
grid_search = GridSearchCV(estimator = dtcl, param_grid = param_grid, cv = 10)
```

Now , we will fit our train dataset to the grid search.

After fitting the values, we will get the best parameters using the command:

```
best_grid = grid_search.best_estimator_
```

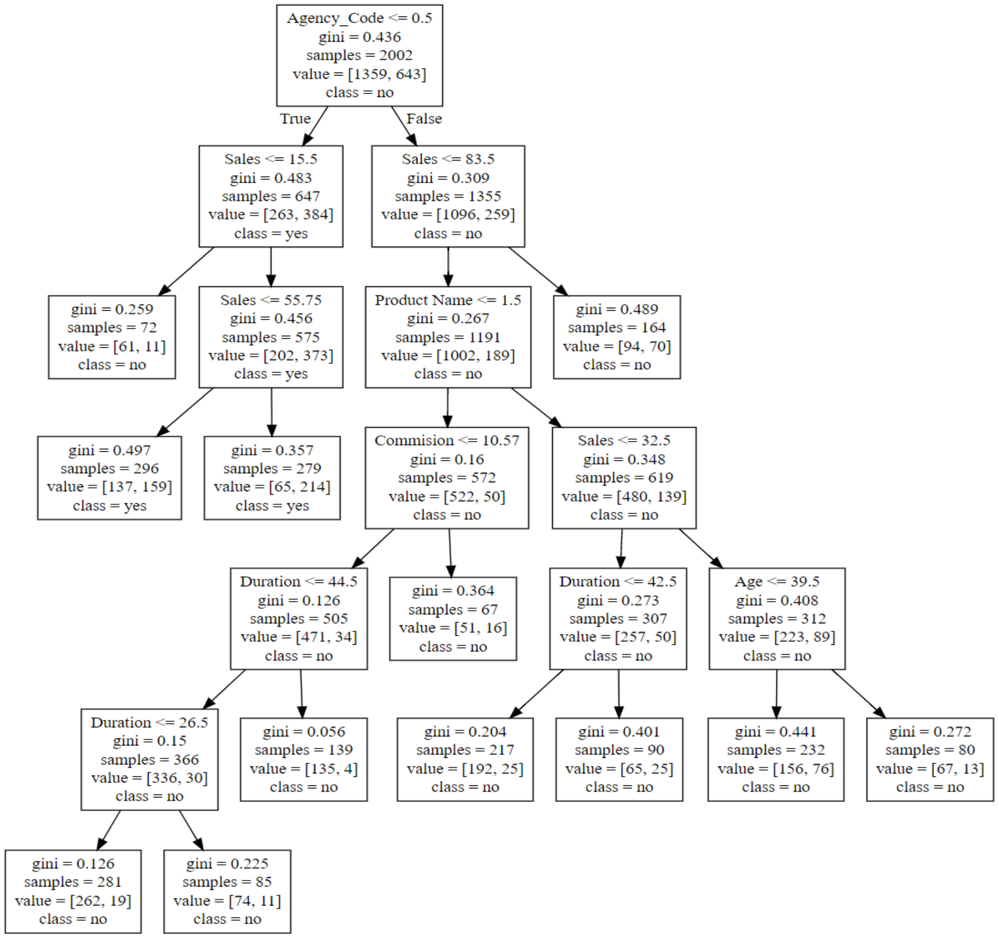
We will get the following best parameters for our CART model:

```
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 300}
```

Now, will add the best parameters and build the tree again. On running the code and saving

the output to .dot file and view the chart online by pasting the code on http://webgraphviz.com/

, which will give the following tree:



The above tree looks good ,we can go ahead for conducting the test and train data analysis.

Variable Importance:

```
Agency_Code   0.600450
Sales         0.304966
Product Name  0.047357
Duration      0.018764
Commision     0.014732
Age           0.013731
Type          0.000000
Channel       0.000000
Destination   0.000000
```

We will use our train and test data with the best Parameters to predict:

```
ytrain_predict = best_grid.predict(X_train)
ytest_predict = best_grid.predict(X_test)
```

We will get following results :

	0	1
0	0.573171	0.426829
1	0.971223	0.028777
2	0.232975	0.767025
3	0.837500	0.162500
4	0.837500	0.162500

Building a Random Forest Classifier:

Due to large volume of data, trying for different parameter values in the gridsearch with higher cv value will have higher execution time, so the best values that came after the search are directly put in Param_grid.

As Splitting of the data into Train and Test set is already done
We can start building the RF Model.
We can get the best parameters with GridSearchCV function.
We can give different values to the parameters and fit the data and get the best parameters.

```
param_grid = {
   'max_depth': [10],
   'max_features': [6],
   'min_samples_leaf': [10],
   'min_samples_split': [50],
   'n_estimators': [300]
}

rfcl = RandomForestClassifier(random_state=1)

grid_search = GridSearchCV(estimator = rfcl, param_grid = param_grid, cv = 10)
```

Best parameters:

```
{'max_depth': 10,
 'max_features': 6,
 'min_samples_leaf': 10,
 'min_samples_split': 50,
 'n_estimators': 300}
```

Best grid:

```
RandomForestClassifier(max_depth=10, max_features=6, min_samples_leaf=10,
                       min_samples_split=50, n_estimators=300, random_state=1)
```

We will be predicting our train and test data with the best Parameters using:

```
ytrain_predict = best_grid.predict(X_train)
ytest_predict = best_grid.predict(X_test)
```

Building a Neural Network Classifier:

For ANN model, scaling is mandatory.
Hence we will use StandardScalar on the dataset.
The Data before scaling looks like this:

Hidden layer sizes: 100
Activation: The Activation method we use is RELU
Solver: We use is adam
Tolerance:0.1
Max iterations : 10000

We will fit the train data to estimate the best parameters:

```
'activation': 'relu',
 'hidden_layer_sizes': 100,
 'max_iter': 10000,
 'solver': 'adam',
 'tol': 0.1}
```

# Predicting the Training and Testing data

```
ytrain_predict = best_grid.predict(X_train)

ytest_predict = best_grid.predict(X_test)
```

2.3 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model

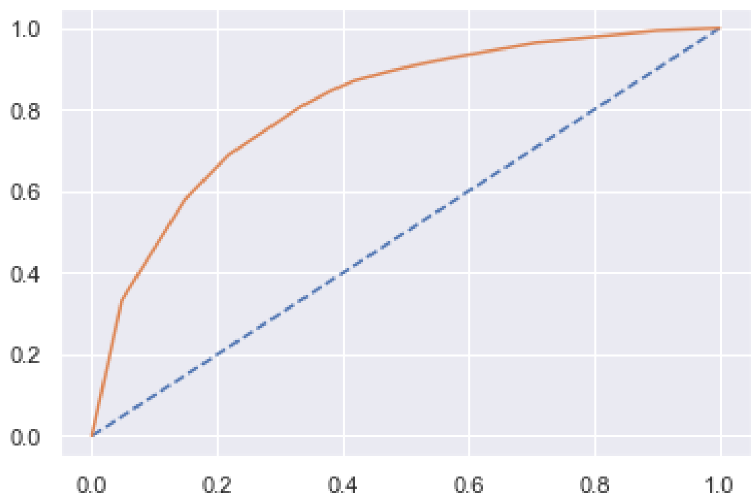
Now we will do Model Evaluation for Decision tree:

We will predict the probabilities of train data,
calculate the AUC, calculating the ROC curve and plotting the same.

AUC and ROC for the training data

AUC value =0.810

ROC plot for train data is given below:



Confusion Matrix for the training data

```
array([[1157,  202],
       [ 270,  373]], dtype=int64)
```

Train Data Accuracy, we get the result of :

0.7642357642357642

Classification report for train data as follows:

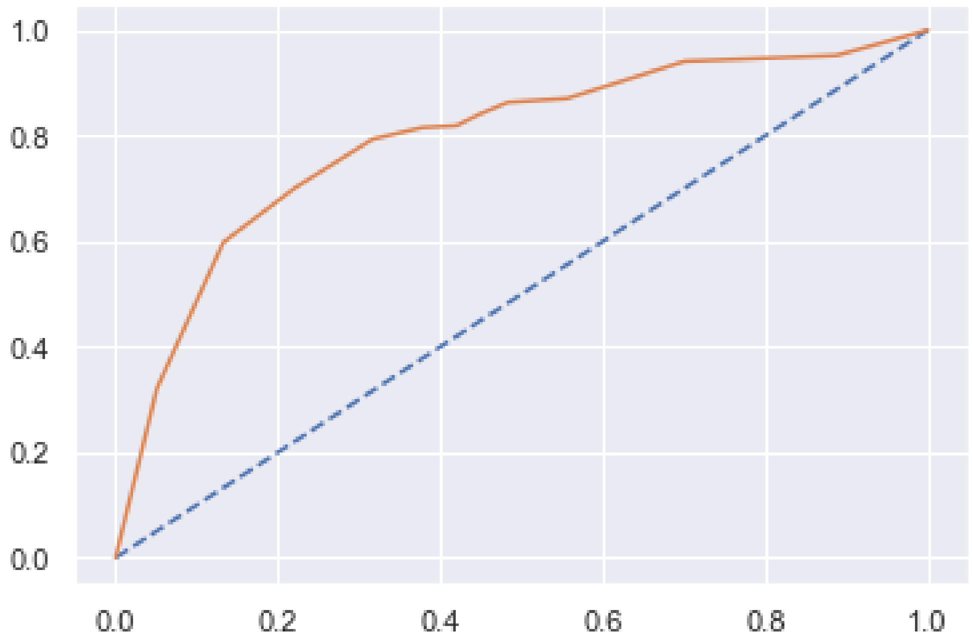
```
              precision    recall  f1-score   support

           0       0.81      0.85      0.83      1359
           1       0.65      0.58      0.61       643

    accuracy                           0.76      2002
   macro avg       0.73      0.72      0.72      2002
weighted avg       0.76      0.76      0.76      2002
```

```
cart_train_precision  0.65
cart_train_recall  0.58
cart_train_f1  0.61
```

AUC and ROC for the test data

AUC: 0.792

ROC plot for test data is given below:



# Confusion Matrix for the test data:

```
array([[510,  78],
       [109, 162]], dtype=int64)
```

#Test Data Accuracy ,we get the result of :

0.7823050058207218

cart_test_precision 0.68
cart_test_recall 0.6
cart_test_f1 0.63

Classification report for test data as follows

```
              precision    recall  f1-score   support

           0       0.82      0.87      0.85       588
           1       0.68      0.60      0.63       271

    accuracy                           0.78       859
   macro avg       0.75      0.73      0.74       859
weighted avg       0.78      0.78      0.78       859
```

Cart Conclusion:

Train Data: AUC: 81.0% Accuracy: 76.4% Precision: 65% f1-Score: 61%

Test Data: AUC: 79.2% Accuracy: 78.2% Precision: 68% f1-Score: 63%

Training and Test set results are almost similar, and with the overall measures good, the model is a good model.

Agency_Code is the most important variable for predicting Claim status.

RF Model Performance Evaluation on Training data

Confusion Matrix for the training data

```
array([[1222,  137],
      [ 255,  388]], dtype=int64)
```

Rf train accuracy :

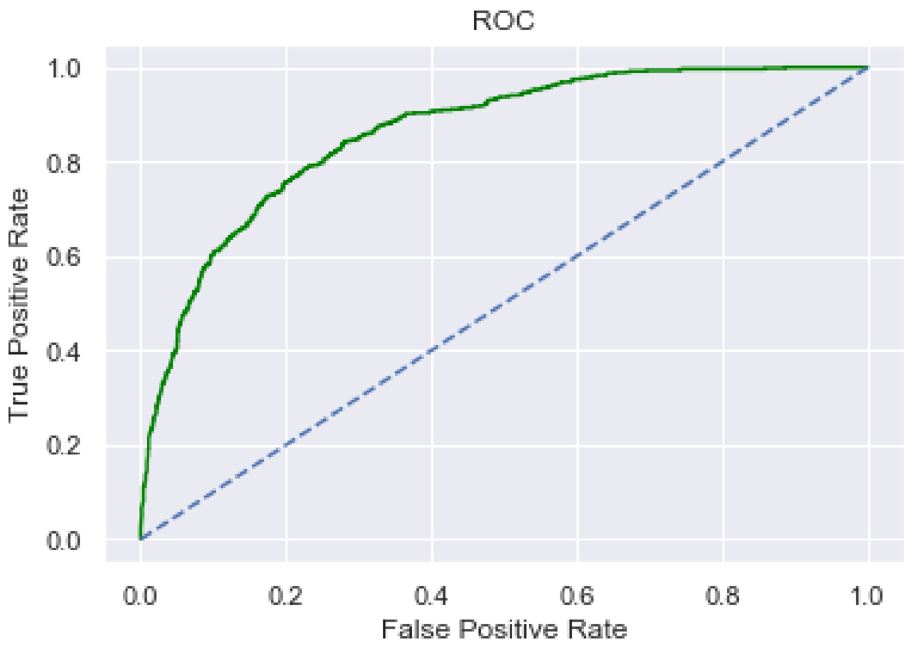
```
0.8041958041958042
```

Classification report for train data as follows:

```
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1359
           1       0.74      0.60      0.66       643

    accuracy                           0.80      2002
   macro avg       0.78      0.75      0.76      2002
weighted avg       0.80      0.80      0.80      2002

rf_train_precision  0.74
rf_train_recall  0.6
rf_train_f1  0.66
```

Area under Curve for train data is 0.8621487760303123

ROC plot for train data is given below:



Confusion Matrix for the test data:

```
array([[521,  67],
      [114, 157]], dtype=int64)
```

Test accuracy:

0.789289871944121

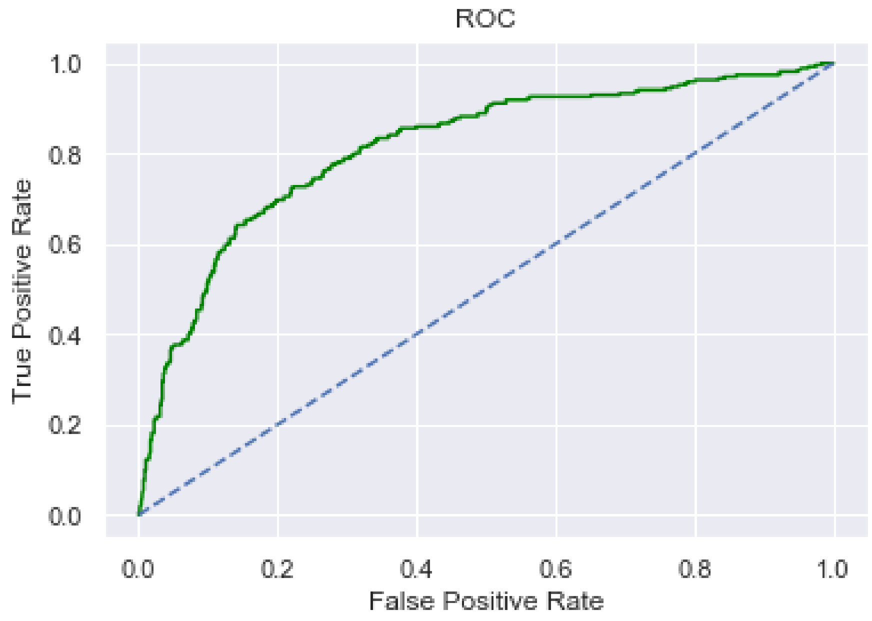
Classification report for test data as follows:

```
              precision    recall  f1-score   support

         0       0.82      0.89      0.85       588
         1       0.70      0.58      0.63       271

   accuracy                           0.79       859
  macro avg       0.76      0.73      0.74       859
weighted avg       0.78      0.79      0.78       859
```

```
rf_test_precision  0.7
rf_test_recall  0.58
rf_test_f1  0.63
```

Area under Curve for test data is 0.813402113612973

ROC plot for test data is given below:



# Variable Importance

```
                 Imp
Agency_Code   0.329660
Sales         0.203813
Product Name  0.176773
Duration      0.095796
Commision     0.088820
Age           0.074503
Type          0.016091
Destination   0.012847
Channel       0.001698
```

Random Forest Conclusion

Train Data: AUC: 86.2% Accuracy: 80.4% Precision: 0.74% f1-Score: 0.66%

Test Data: AUC: 78.9% Accuracy: 81.3% Precision: 0.7% f1-Score: 0.63%

Training and Test set results are good, and with the overall measures high, the model is a good model. Agency code is again the most important variable for predicting claim status.

Artificial Neural Network Model Performance Evaluation on Training data

Now we have to assign the best grid parameters to predict the training data.
After doing the same we will start to analyse model performance of train data.

Confusion matrix for the train data:

```
array([[1057,  302],
       [ 280,  363]], dtype=int64)
```

Neural network Accuracy

```
0.7092907092907093
```

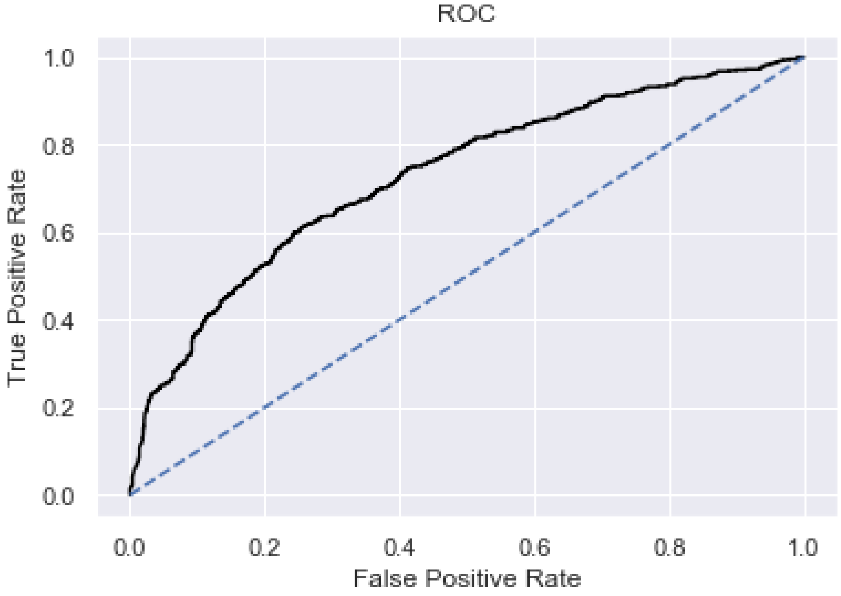
Classification report for train data as follows:

```
              precision    recall  f1-score   support

           0       0.79      0.78      0.78      1359
           1       0.55      0.56      0.56       643

    accuracy                           0.71      2002
   macro avg       0.67      0.67      0.67      2002
weighted avg       0.71      0.71      0.71      2002
```

```
nn_train_precision  0.55
nn_train_recall  0.56
nn_train_f1  0.56
```

Area under Curve for the train data is 0.7294066284673228

Roc curve for the train data:



NN Model Performance Evaluation on Test data:

Confusion matrix for the test data:

```
array ([[458, 130],
      [123, 148]], dtype=int64)
```

Neural network Accuracy

0.7054714784633295

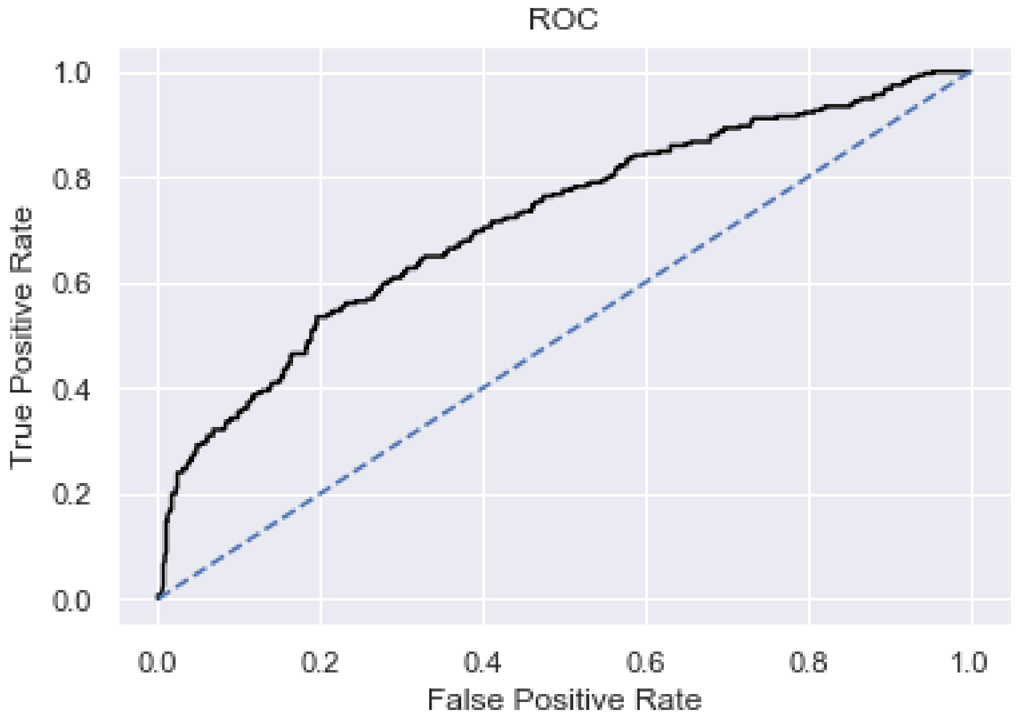
Classification report for train data as follows:

```
             precision    recall  f1-score   support

        0      0.79      0.78      0.78       588
        1      0.53      0.55      0.54       271

   accuracy                        0.71       859
  macro avg      0.66      0.66      0.66       859
weighted avg      0.71      0.71      0.71       859
```

```
nn_test_precision  0.53
nn_test_recall  0.55
nn_test_f1  0.54
```

Area under Curve for the train data is 0.7155345533047168

Roc curve for the test data is:



Neural Network Conclusion:

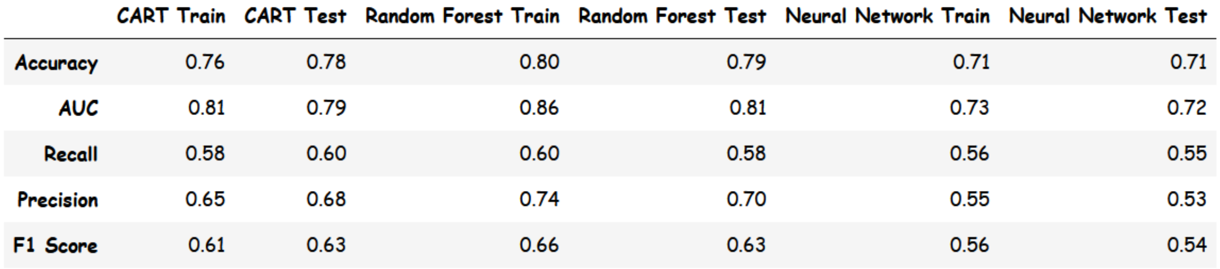
Train Data: AUC: 78.7% Accuracy: 75.8% Precision: 64% f1-Score: 59%

Test Data: AUC: 78.6% Accuracy: 75.4% Precision: 63% f1-Score: 57%

Training and Test set results are almost similar, and with the overall measures high, the model is a good model.

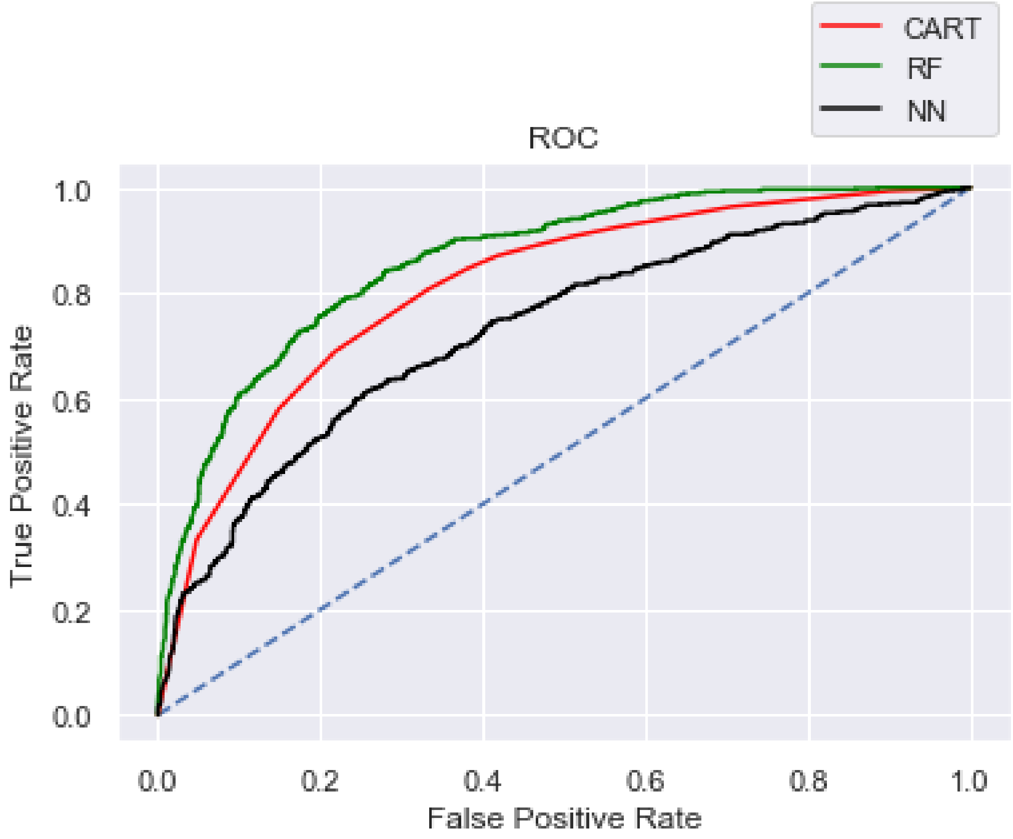
# 2.4 Final Model: Compare all the model and write an inference which model is best/optimized.

Comparison of the performance metrics from the 3 models:

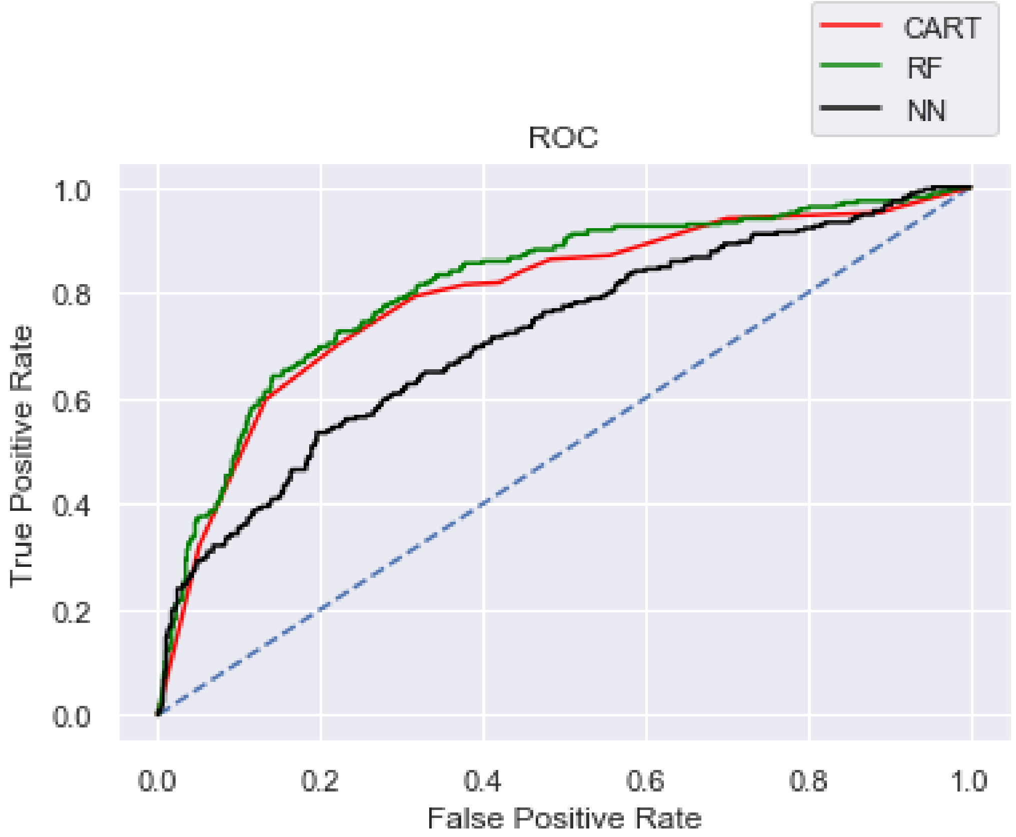


	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.76	0.78	0.80	0.79	0.71	0.71
AUC	0.81	0.79	0.86	0.81	0.73	0.72
Recall	0.58	0.60	0.60	0.58	0.56	0.55
Precision	0.65	0.68	0.74	0.70	0.55	0.53
F1 Score	0.61	0.63	0.66	0.63	0.56	0.54

ROC Curve for the 3 models on the Training data:



ROC Curve for the 3 models on the Test data :



Out of the 3 models, Random Forest has slightly better performance than the Cart and Neural network model

Overall all the 3 models are reasonaly stable enough to be used for making any future predictions.
From Cart and Random Forest Model, Instead of creating a single Decision Tree it can create a multiple decision trees as Random Forest have much less variance than a single decision tree and hence can provide the best claim status from the data.

The Agency_Code is found to be the most useful feature amongst all other features for predicting if a person will claim for insurance or not.

2.5) Inference: Basis on these predictions, what are the business insights and recommendations ?

The Agency code has significant importance
Data contains significant number of outliers.
Out of the 3 models, Random Forest has slightly better performance than the Cart and Neural network models.
Instead of creating a single Decision Tree it can create a multiple decision trees as Random Forest have much less variance than a single decision tree and hence can provide the best claim status from the data.
Overall all the three models are reasonably stable enough to be used for making any future predictions as there is no overfitting.
By performing the 3 models, we can conclude that the data set is well balanced to conduct the modelling.

Claims are higher for Destination- ASIA. May be people are negligentto follow the terms and conditions or the terms and conditions present in the policy are less for them so the company can revise the Terms and Conditions for ASIA. Company should take necessary procedures before alloting policy in ASIA. Company can also increase the premium on the policy to recover the claim cost .

Claims are Higher for Online Distribution channel and very low for offline channel. Reason may be the online procedure is easy and saves time. So the insurance company should promote the offline purchase by providing some offers ,less premium cost or any other additional offer.

Management can make the policy purchase procedure very simple and increase the complexity of the Claim Procedure.
Reductions in claims payments minimizes the need to raise insurance premiums. This makes insurance more efficient and affordable.