PROBLEM 1:

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.

SOLUTION:

Loading the basic libraries and checking the head and tail of the Data Frame: -

	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

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
205	13.89	14.02	0.8880	5.439	3.199	3.986	4.738
206	16.77	15.62	0.8638	5.927	3.438	4.920	5.795
207	14.03	14.16	0.8796	5.438	3.201	1.717	5.001
208	16.12	15.00	0.9000	5.709	3.485	2.270	5.443
209	15.57	15.15	0.8527	5.920	3.231	2.640	5.879

Checking Info: -

```
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
```

As can be seen all the columns are numeric, to be specific contains float type variables, with 210 Non-null values.

There are 7 columns namely, Spending, Advance payments, probability of full payment, current balance, credit limit, min payment amount and max spent in single shopping. Following is the description of the 7 columns (as mentioned in the data dictionary): -

- **spending**: Amount spent by the customer per month (in 1000s)

- **advance_payments**: Amount paid by the customer in advance by cash (in 100s)
- **probability_of_full_payment**: Probability of payment done in full by the customer to the bank
- **current_balance**: Balance amount left in the account to make purchases (in 1000s)
- **credit_limit**: Limit of the amount in credit card (10000s)
- **min_payment_amt**: minimum paid by the customer while making payments for purchases made monthly (in 100s)
- **max_spent_in_single_shopping**: Maximum amount spent in one purchase (in 1000s)

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

Checking the data summary: -

	count	mean	std	min	25%	50%	75%	max	IQR	COV
spending	210.0	14.847524	2.909699	10.5900	12.27000	14.35500	17.305000	21.1800	5.035000	0.195972
advance_payments	210.0	14.559286	1.305959	12.4100	13.45000	14.32000	15.715000	17.2500	2.265000	0.089699
probability_of_full_payment	210.0	0.870999	0.023629	0.8081	0.85690	0.87345	0.887775	0.9183	0.030875	0.027129
current_balance	210.0	5.628533	0.443063	4.8990	5.26225	5.52350	5.979750	6.6750	0.717500	0.078717
credit_limit	210.0	3.258605	0.377714	2.6300	2.94400	3.23700	3.561750	4.0330	0.617750	0.115913
min_payment_amt	210.0	3.700201	1.503557	0.7651	2.56150	3.59900	4.768750	8.4560	2.207250	0.406345
max_spent_in_single_shopping	210.0	5.408071	0.491480	4.5190	5.04500	5.22300	5.877000	6.5500	0.832000	0.090879

The IQR and COV has been explicitly added in the Data Frame. The COV tells us that most of the variables are uniformly distributed. Also, the mean and median are more or less near to each other, indicating somewhat normal distribution. The variable min_payment_amt has COV a bit on the higher side i.e., 40.

Checking the shape of data and duplicates if any: -

```
print('Shape of data is:', df.shape)
print('Duplicates:', df.duplicated().sum())
```

```
Shape of data is: (210, 7)
Duplicates: 0
```

As can be seen there are 0 duplicate records, with 210 rows and 7 columns.

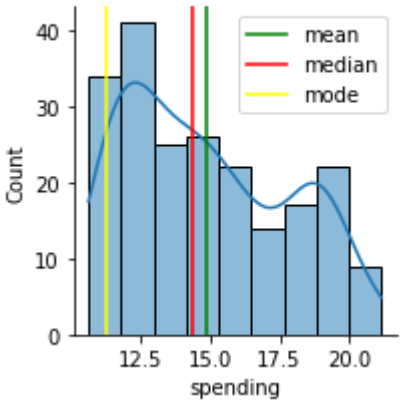
Univariate analysis: -

A function is written in python to visualize the description, distribution and boxplot for every variable. The output is as follows: -

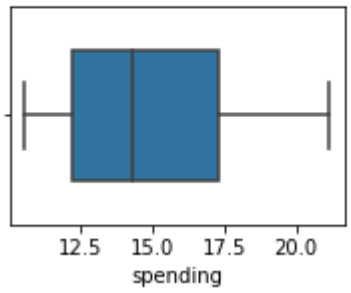
1. Spending: -

```
Description of spending  is: -
count    210.000000
mean      14.847524
std        2.909699
min       10.590000
25%       12.270000
50%       14.355000
75%       17.305000
max       21.180000
Name: spending, dtype: float64

--------------------------------
Mean is:  14.847523809523818
Median is:  14.355
Mode is:  11.23
```

Distribution of spending is: -



Boxplot of spending is: -

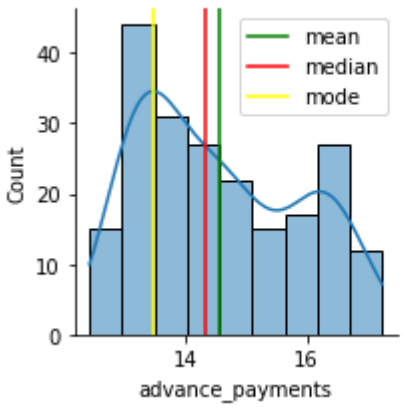


The mean and median are close apart with mean>median>mode. The distribution seems a bit right skewed. The std deviation is very less. No outliers in the data.

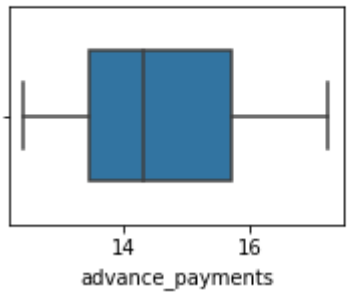
2. Advance Payments

```
Description of advance_payments  is: -
count    210.000000
mean      14.559286
std        1.305959
min       12.410000
25%       13.450000
50%       14.320000
75%       15.715000
max       17.250000
Name: advance_payments, dtype: float64

----------------------------------------
Mean is:  14.559285714285727
Median is:  14.32
Mode is:  13.47
```

Distribution of advance_payments is: -



Boxplot of advance_payments is: -

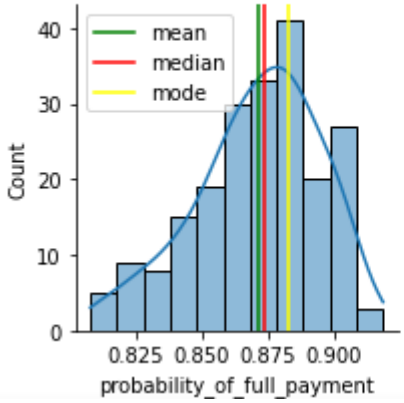


Mean and Median are close apart, with mean>median>mode. The std deviation is 1.3. Distribution appears a bit right skewed. No outliers in the data.

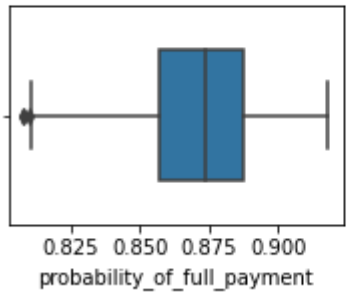
3. Probability of full payment

```
Description of probability_of_full_payment  is: -
count    210.000000
mean       0.870999
std        0.023629
min        0.808100
25%        0.856900
50%        0.873450
75%        0.887775
max        0.918300
Name: probability_of_full_payment, dtype: float64

----------------------------------------------------
Mean is:  0.8709985714285714
Median is:  0.8734500000000001
Mode is:  0.8823
```

```
Distribution of  probability_of_full_payment  is: -
```




```
Boxplot of probability_of_full_payment  is: -
```

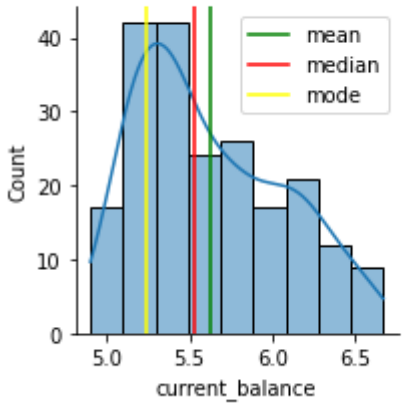


Mean and median are close apart with mode>median>mean. Distribution appears to be left skewed. Negligible standard deviation. No outliers.

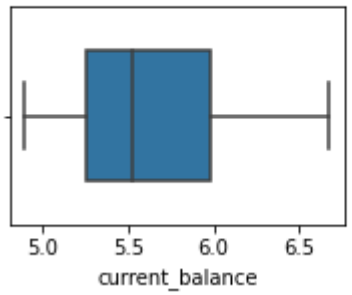
4. Current Balance

```
Description of current_balance  is: -
count    210.000000
mean       5.628533
std        0.443063
min        4.899000
25%        5.262250
50%        5.523500
75%        5.979750
max        6.675000
Name: current_balance, dtype: float64

----------------------------------------
Mean is:  5.628533333333334
Median is:  5.5235
Mode is:  5.236000000000001
```

```
Distribution of  current_balance  is: -
```



Boxplot of current_balance is: -

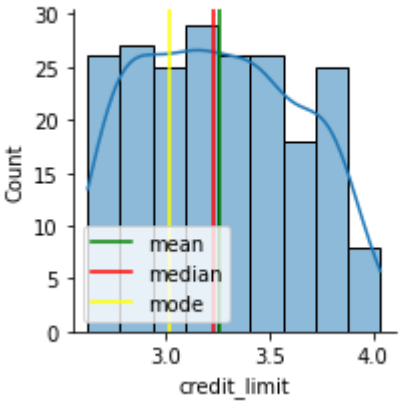


Mean>Median>Mode, with a slight right skewed distribution. Std Deviation of .44, and no outliers.

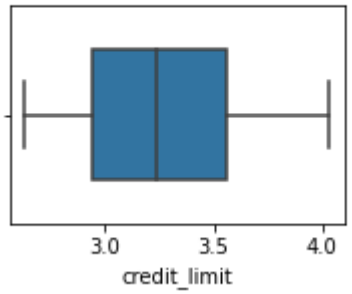
5. Credit limit

```
Description of credit_limit  is: -
count    210.000000
mean       3.258605
std        0.377714
min        2.630000
25%        2.944000
50%        3.237000
75%        3.561750
max        4.033000
Name: credit_limit, dtype: float64

-----------------------------------
Mean is:  3.258604761904763
Median is:  3.237
Mode is:  3.0260000000000002
```

Distribution of credit_limit is:



Boxplot of credit_limit is:

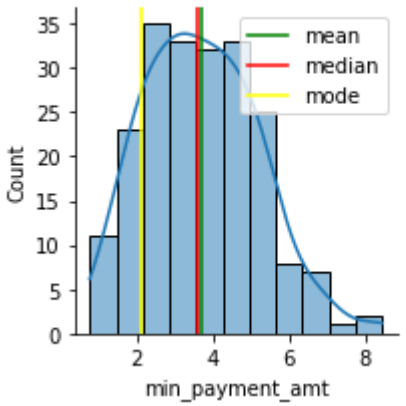


Mean and Median are almost equal, with mode slightly on the lesser side. Very less std deviation and no outliers. The distribution appears close to normal.

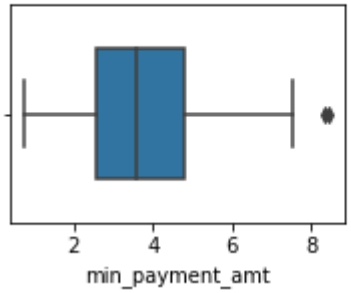
6. Min payment amount

```
Description of min_payment_amt  is: -
count    210.000000
mean       3.700201
std        1.503557
min        0.765100
25%        2.561500
50%        3.599000
75%        4.768750
max        8.456000
Name: min_payment_amt, dtype: float64

---------------------------------------
Mean is:  3.7002009523809507
Median is:  3.599
Mode is:  2.129
```

Distribution of min_payment_amt is:



Boxplot of min_payment_amt is: -

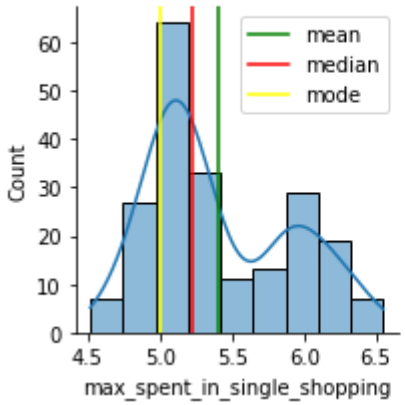


Mean ~ Median, mode is slightly on the lower side. Significant std deviation. A few outliers can be seen in the boxplot. Distribution tends to normal.

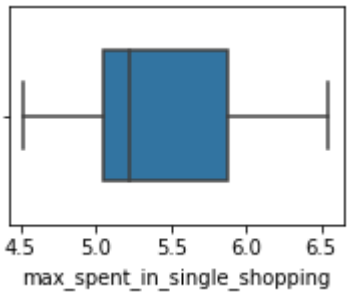
7. Max spent in single shopping

```
Description of max_spent_in_single_shopping  is:
count    210.000000
mean       5.408071
std        0.491480
min        4.519000
25%        5.045000
50%        5.223000
75%        5.877000
max        6.550000
Name: max_spent_in_single_shopping, dtype: float

-----------------------------------------------
Mean is:  5.408071428571429
Median is:  5.223000000000001
Mode is:  5.001
```

Distribution of max_spent_in_single_shopping is:



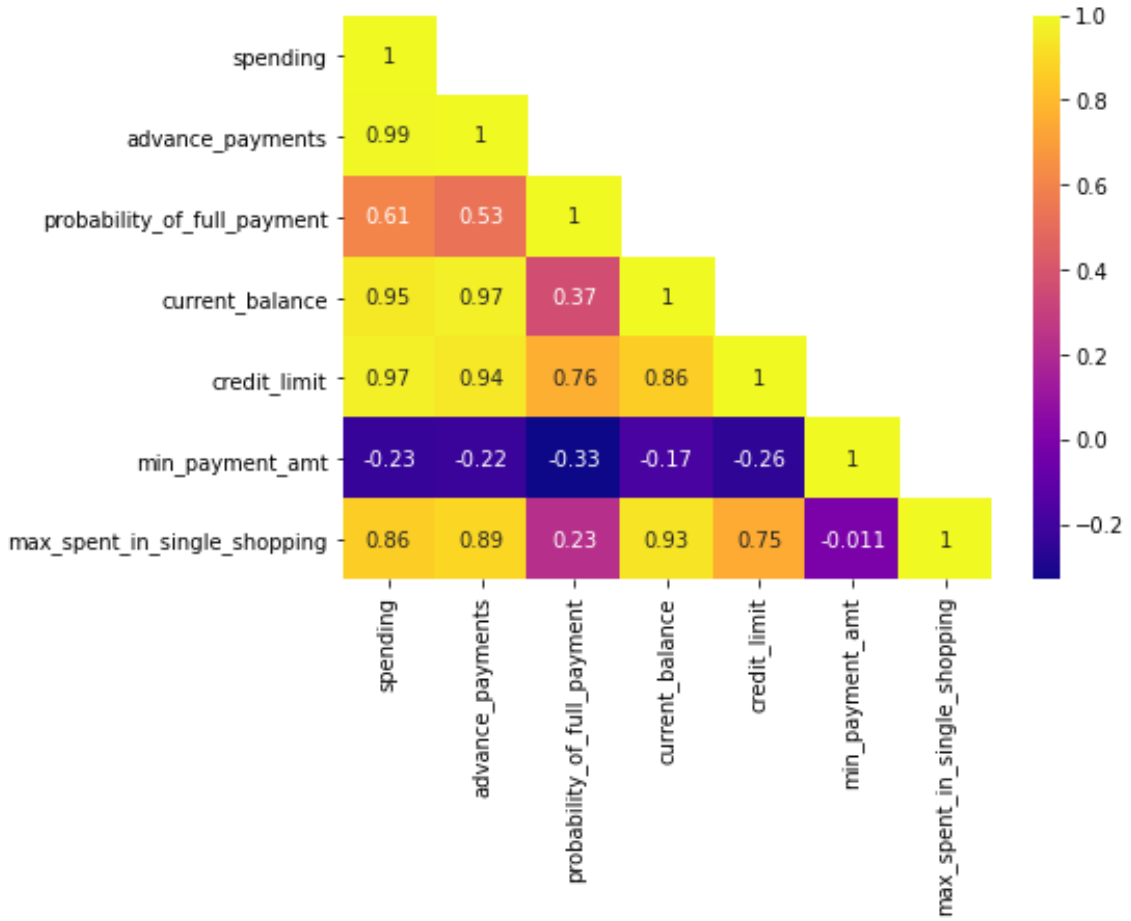
Boxplot: -



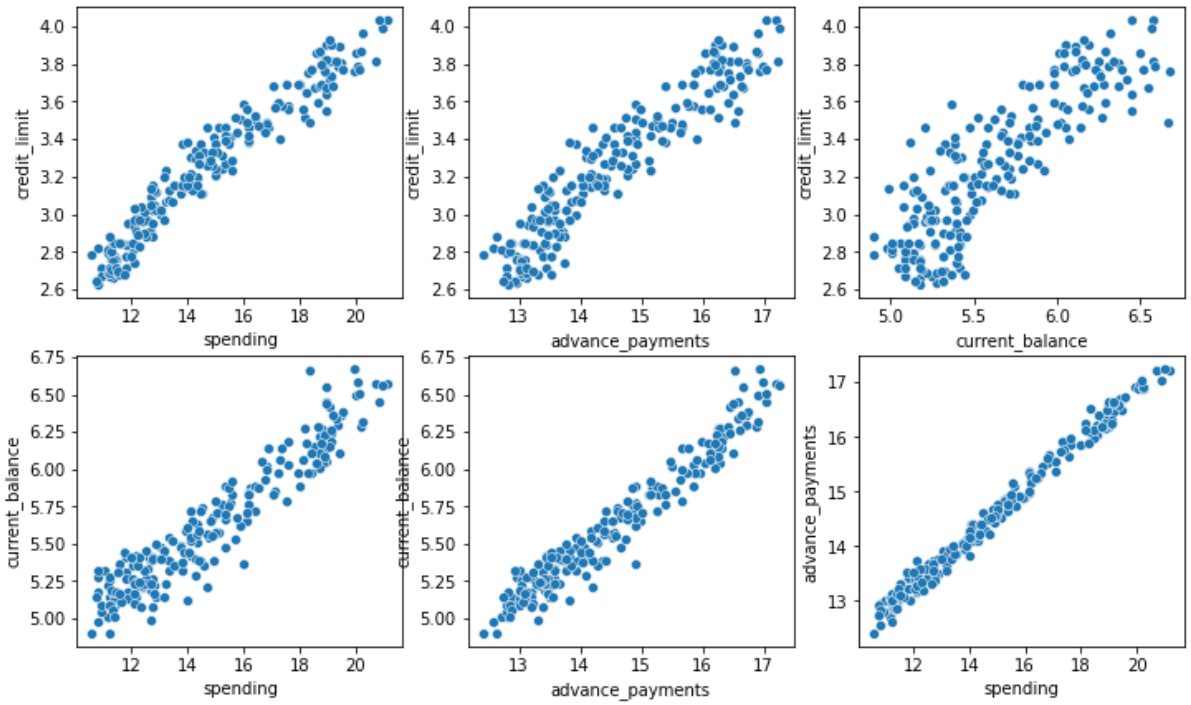
The mean median and mode are well apart, with mean>median>mode. The distribution is not uniform. Appears to be slightly left skewed. No outliers present.

Bivariate analysis

Checking corelations: -



A few variables are highly corelated. We can observe the same with scatter plots



The relations displayed above are self-explanatory. More the credit limit, more will be the spending and current balance. Similarly, more the current balance, more is the spending and advance payments

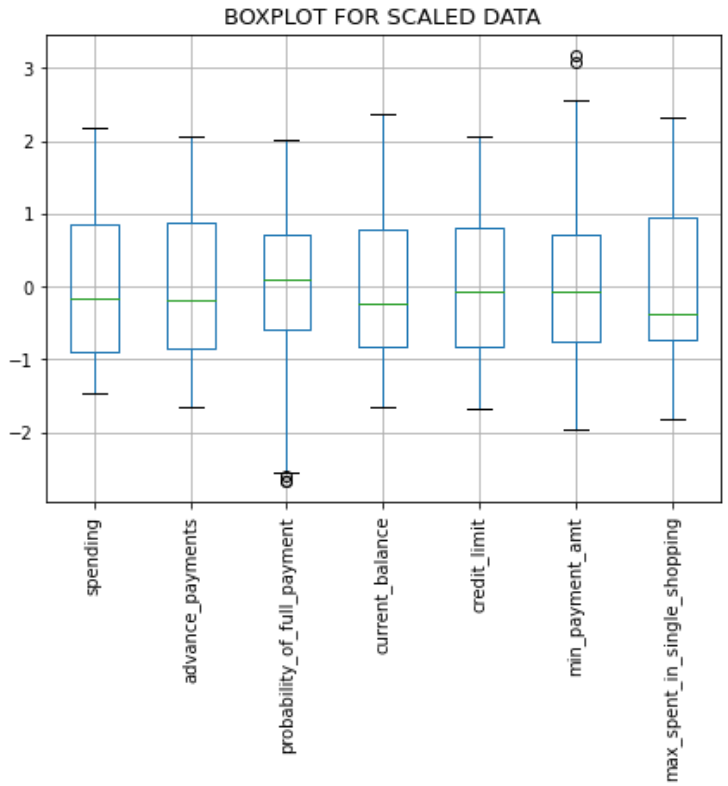
1.2 Do you think scaling is necessary for clustering in this case? Justify

	mean	std	min	25%	50%	75%	max
spending	14.847524	2.909699	10.5900	12.27000	14.35500	17.305000	21.1800
advance_payments	14.559286	1.305959	12.4100	13.45000	14.32000	15.715000	17.2500
probability_of_full_payment	0.870999	0.023629	0.8081	0.85690	0.87345	0.887775	0.9183
current_balance	5.628533	0.443063	4.8990	5.26225	5.52350	5.979750	6.6750
credit_limit	3.258605	0.377714	2.6300	2.94400	3.23700	3.561750	4.0330
min_payment_amt	3.700201	1.503557	0.7651	2.56150	3.59900	4.768750	8.4560
max_spent_in_single_shopping	5.408071	0.491480	4.5190	5.04500	5.22300	5.877000	6.5500

We can clearly see that the variables are not on the same scale. They range from .87 to 14.84. This is due to the probability column, which obviously will be less than 1. However, at this point of time we can't take a call to drop it off and continue, hence it is wise to scale the data.

In clustering algorithms where distance plays a major role, scaling is mandatory.

The scaling of data is done using zscore scaling. The boxplot for the same is as follows: -



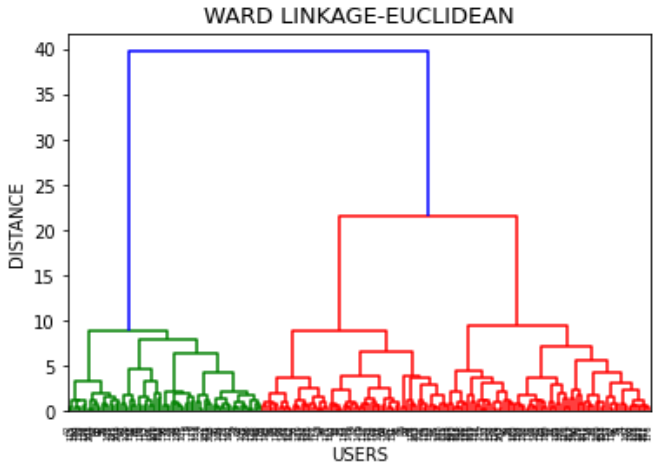
We can clearly observe the data is on the same scale now.

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

Hierarchical clustering can be implemented in 2 ways: using fclusters from SciPy library or using agglomerative clustering from the scikit library.

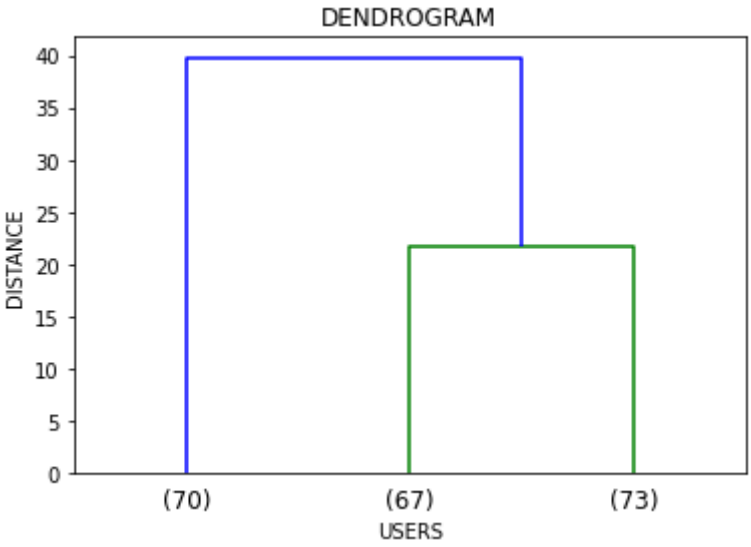
Using Fclusters and dendrograms: -

Using the ward with Euclidean distances, an initial dendrogram is obtained: -



The above dendrogram looks quite messed up. To decide the no of clusters is a very subjective matter, would depend upon person to person, and also the nature of business problem. But looking at dendrogram it is quite obvious that we need to cut the dendrogram at a distance of 15-20, which would give us 3 clusters precisely.

Modifying the dendrogram with no of clusters = 3 gives the following dendrogram: -



The dendrogram appears clean with 3 clusters, with sufficient distances between the clusters. The dendrogram also tells us the frequency of each cluster: 70,67 and 73.

Now we need to assign the clusters which is done using fclusters. The clusters assigned are stored in a separate column “Clusters Hierarchical”.

Checking cluster profiles: -

Clusters_hierarchichal	1	2	3
spending	18.371429	11.872388	14.199041
advance_payments	16.145429	13.257015	14.233562
probability_of_full_payment	0.884400	0.848072	0.879190
current_balance	6.158171	5.238940	5.478233
credit_limit	3.684629	2.848537	3.226452
min_payment_amt	3.639157	4.949433	2.612181
max_spent_in_single_shopping	6.017371	5.122209	5.086178
Frequency	70.000000	67.000000	73.000000

The same can also be verified for agglomerative clustering using the Agglomerative Clustering module from scikit learn.

Cluster profiles for Agglomerative clustering: -

Agglo Clusters	0	1	2
spending	14.199041	18.371429	11.872388
advance_payments	14.233562	16.145429	13.257015
probability_of_full_payment	0.879190	0.884400	0.848072
current_balance	5.478233	6.158171	5.238940
credit_limit	3.226452	3.684629	2.848537
min_payment_amt	2.612181	3.639157	4.949433
max_spent_in_single_shopping	5.086178	6.017371	5.122209
Frequency	73.000000	70.000000	67.000000

The results are same.

Now let's describe the clusters we have achieved: -

Cluster 1: -

- Highest spenders, with high current balance, credit limit
- Makes the most of the advance payments and the probability of full payment is also high.
- Belong to the rich and elite group

Cluster 2: -

- Lowest spenders, with least current balance, and lowest credit limit
- Makes the least advance payments, probability of full payment is quite less comparatively
- Belong to the class with average job pay.

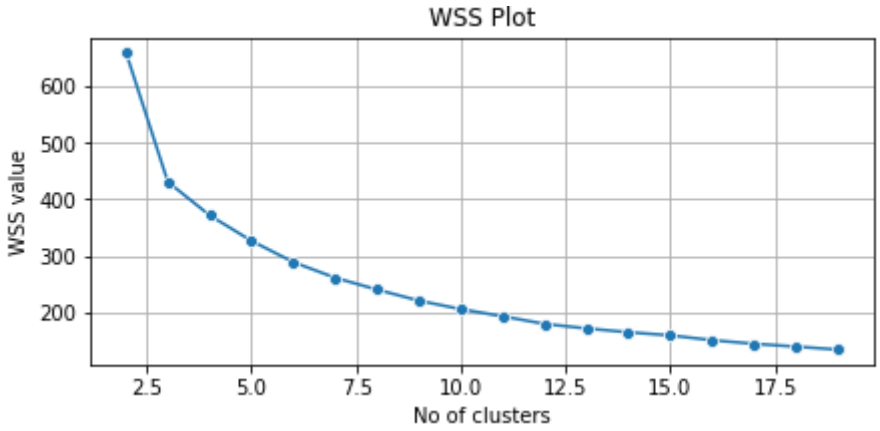
Cluster 3: -

- Mediocre spending, current balance and credit limit
- Makes a decent amount of advance payment, also has a decent probability of full payment
- Must belong to the upper middle class.

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

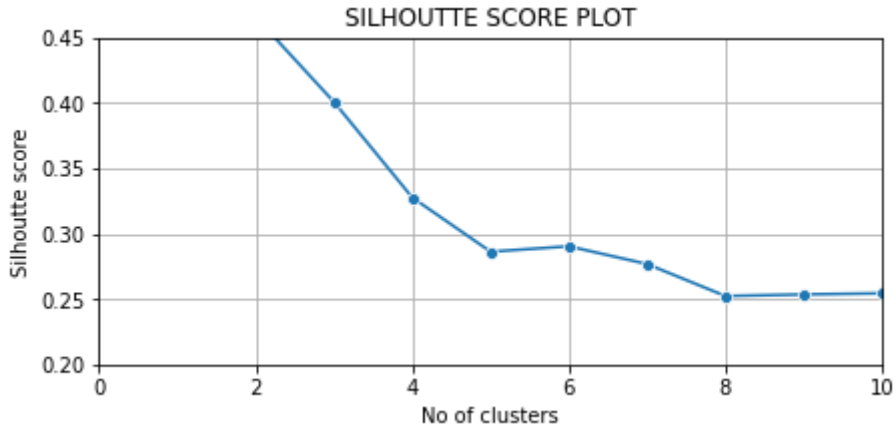
As determined in the hierarchical clustering, we shall proceed with 3 clusters only. Further we would take a look at the WSS plot and Silhouette scores to conclude if the clusters decided were optimum.

The K-Means model is fitted with clusters = 3. The WSS plot is plotted for clusters ranging from 2 to 20: -



As expected, the Within sum of squares go on decreasing with the increase in no of clusters, but at the same time we also need to keep in mind the silhouette scores which indicate the centroid distances: the larger the better.

Observing the silhouette scores: -

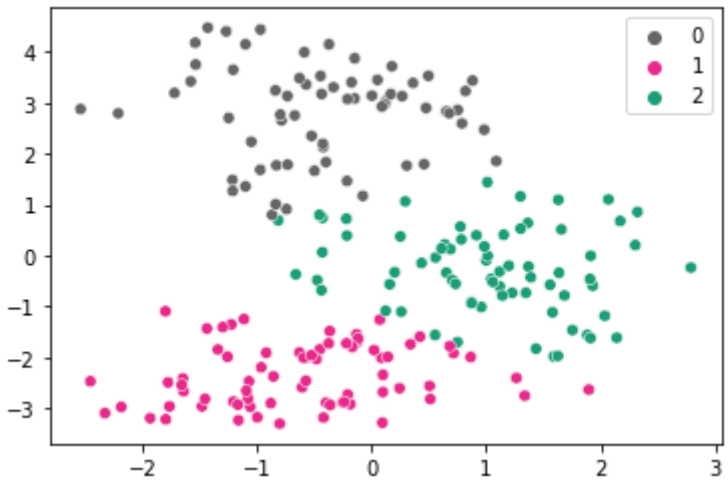


We can clearly see with increase in no of clusters the silhouette score decreases, which is not a good thing. We have to make a trade-off between silhouette score and within sum of squares to arrive on optimum no of clusters.

From the above two plots we can conclude that our initial choice of clusters = 3 was absolutely correct.

To plot the clusters, we can converge the columns using PCA (with components = 2 just for plotting purposes X and Y axis)

The output looks like: -



Accept a few points, the clusters look well defined, which again confirms our choice of optimum clusters = 3.

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

The cluster profiles obtained for K-Means.

Clusters KMeans	0	1	2
spending	18.495373	14.437887	11.856944
advance_payments	16.203433	14.337746	13.247778
probability_of_full_payment	0.884210	0.881597	0.848253
current_balance	6.175687	5.514577	5.231750
credit_limit	3.697537	3.259225	2.849542
min_payment_amt	3.632373	2.707341	4.742389
max_spent_in_single_shopping	6.041701	5.120803	5.101722
Frequency	67.000000	71.000000	72.000000

Cluster profiles obtained for Agglomerative/Hierarchical: -

Clusters_hierarchichal	1	2	3
spending	18.371429	11.872388	14.199041
advance_payments	16.145429	13.257015	14.233562
probability_of_full_payment	0.884400	0.848072	0.879190
current_balance	6.158171	5.238940	5.478233
credit_limit	3.684629	2.848537	3.226452
min_payment_amt	3.639157	4.949433	2.612181
max_spent_in_single_shopping	6.017371	5.122209	5.086178
Frequency	70.000000	67.000000	73.000000

The values obtained in both the profiles are nearly same, the only difference being the frequency. Ignoring the cluster labels in both the cases, we have confirmed the presence of 3 different clusters in the data set. The three different groups can be classified as follows: -

Cluster 1: -

Spending mean:18.3K, advance payments:1.6K, current balance: 6.15K, credit limit: 36.8K, min payment amount: 363, max amt in shopping: 6K.

Adding the spending amount and current balance we get: 18+6 = 24K, which is well below the credit limit.

These are the rich and lavish spenders, in every aspect as can be seen from the data. The min amount spent is the lowest, relatively, which suggests they focus on buying expensive things only.

They are the ones who will be the last ones to default as their probability of full payment is relatively high.

Clsuter2: -

Spending: 11.8K, Advance payments: 1.32K, current balance: 5.2K, credit limit: 28.4K, min payment amount: 494, max spent in single shopping: 5.1K

Adding the spending amount and current balance: 11.8 + 5.2 = 17K, well below the credit limit.

This class do not spend much, the low credit limit might be an indicator of relatively low salaries. If this is true, still they manage their balance pretty well, without exceeding the credit limit. Or we can even speculate this class to be the freshly joined bachelors, as they have the highest relative min payment amount, and also pretty high spent in shopping.

The relatively low probability of full payment makes them prone to defaults the most.

Cluster 3: -

Spending: 14.1K, advance payments: 1.42K, current balance: 5.4K, credit limit: 32.2K, min payment amount: 262, max spent in single shopping: 5.08K.

Adding the spending amount and current balance: 14.1+4.4 = 18.5K, well below the credit limit.

If observed carefully, in spite of having a greater credit limit, they do not tend to spend much. They believe in spending on relatively cheaper things.

This cluster might suggest people belonging to a proper middle-class family, where people insist more on savings. Might be the case, just a speculation.

The probability of full payment is also quite high, which makes them resistant to default.

PROMOTIONAL STRATEGIES: -

Cluster 1: -

Cluster 1 are the lavish spenders. They are spending money to their fullest capacity. They are the ones with highest advance payments and high credit limit. These customers should be targeted on for exceeding credit limit, so that they can spend even more.

Secondly, they can be offered some great surprise gifts every time they spend highly on shopping.

Cluster 2: -

Cluster 2 as speculated are the ones who wish to spend more but can't due to low current balance and credit limit. These customers can be offered no cost EMI loans and extended credit limit for more advance payments.

They can be offered some free credit points for certain items they purchase on credit card, which they can redeem on later purchases.

Cluster 3: -

In-spite of having the potential this group kind of refrains from spending more. They are dedicated for advance payments and also have a higher probability of full payment. These customers should be given heavy discounts to encourage them to shop more.

Also, if needed they must be contacted for loans or extended credit limit to understand why they spend less, and if this could help them to spend a little more.

PROBLEM 2

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.

The data is ingested and a basic EDA performed to know what's happening in the data set. Head and tail of data: -

Head:

	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

Tail:

	Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
2995	28	CWT	Travel Agency	Yes	166.53	Online	364	256.20	Gold Plan	Americas
2996	35	C2B	Airlines	No	13.50	Online	5	54.00	Gold Plan	ASIA
2997	36	EPX	Travel Agency	No	0.00	Online	54	28.00	Customised Plan	ASIA
2998	34	C2B	Airlines	Yes	7.64	Online	39	30.55	Bronze Plan	ASIA
2999	47	JZI	Airlines	No	11.55	Online	15	33.00	Bronze Plan	ASIA

Basic info: -

```
#   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Age            3000 non-null   int64
 1   Agency_Code    3000 non-null   object
 2   Type           3000 non-null   object
 3   Claimed        3000 non-null   object
 4   Commision      3000 non-null   float64
 5   Channel        3000 non-null   object
 6   Duration       3000 non-null   int64
 7   Sales          3000 non-null   float64
 8   Product Name   3000 non-null   object
 9   Destination    3000 non-null   object
```

The data has 10 columns namely age, agency code, type, claimed, commission, cha nnel, duration, sales, product name, destination. Age, commission, sales and duratio n are numeric while the rest are object/categorical.
Following is the information about the columns: -

1. **Age**: age of insured
2. **Agency code**: the unique code of agency
3. **Type**: Type of travel agency
4. **Claimed**: insurance claimed: yes or no (This is the target variable)
5. **Commission**: Commision taken
6. **Channel**: Online/Offline
7. **Duration**: Duration of the tour
8. **Sales**: number of sales of tour insurance
9. **Product Name**: Name of insurance products
10. **Destination**: Destination of the tour

Checking summary of data: -

	count	mean	std	min	25%	50%	75%	max	IQR	COV
Age	3000.0	38.091000	10.463518	8.0	32.0	36.00	42.000	84.00	10.000	0.274698
Commision	3000.0	14.529203	25.481455	0.0	0.0	4.63	17.235	210.21	17.235	1.753809
Duration	3000.0	70.001333	134.053313	-1.0	11.0	26.50	63.000	4580.00	52.000	1.915011
Sales	3000.0	60.249913	70.733954	0.0	20.0	33.00	69.000	539.00	49.000	1.174009

Null value check: -

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

Checking duplicates: -

```
print('The data constains', dups.sum(),'duplicates')
```

```
The data constains 139 duplicates
```

Yes, the data consists of way too many duplicates, which needs to be treated.

```
print('Shape of data with duplicates:', df.shape)
df.drop_duplicates(inplace = True)
print('Shape of data after removing duplicates:', df.shape)
```

```
Shape of data with duplicates: (3000, 10)
Shape of data after removing duplicates: (2861, 10)
```

The no of rows is cut down to 2861 from 3000 after removing duplicates.

For a better description we can perform univariate and bivariate analysis

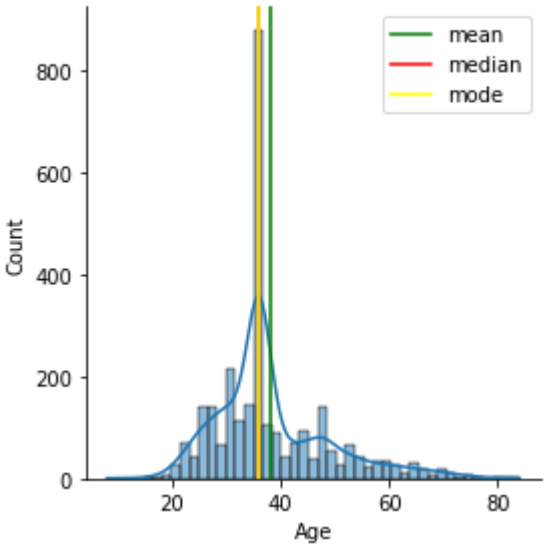
Univariate analysis:

Univariate analysis performed only for int and float variables: -

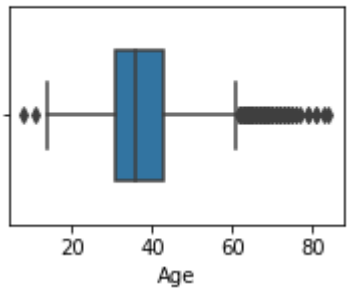
1. Age

```
Description of Age  is: -
count    2861.000000
mean       38.204124
std        10.678106
min         8.000000
25%        31.000000
50%        36.000000
75%        43.000000
max        84.000000
Name: Age, dtype: float64

------------------------------
Mean is:  38.204124432016776
Median is:  36.0
Mode is:  36
```

Distribution of Age is: -



Boxplot of Age is: -

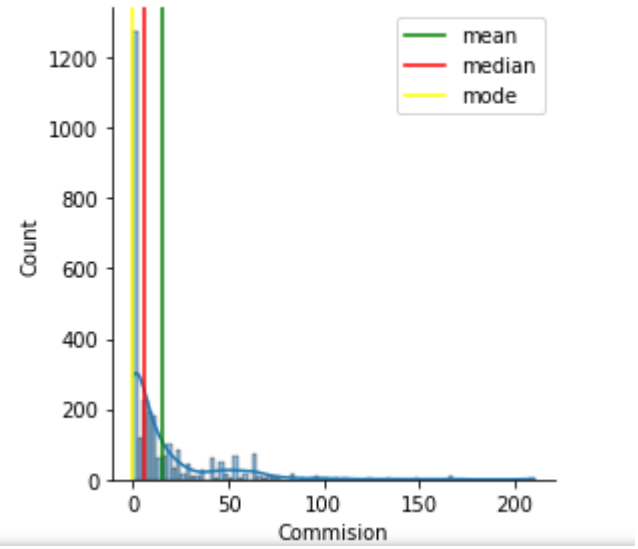


The age column appears to be distributed normally. The median and mode exactly overlap, while the mean is slightly on the higher side. The std deviation is on the higher side. The age column contains many outliers.

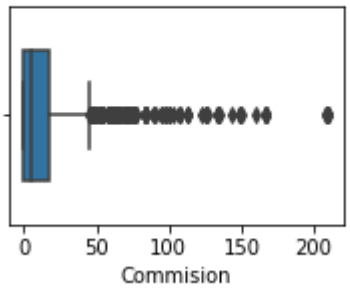
2.Commission

```
Description of Commision  is: -
count    2861.000000
mean       15.080996
std        25.826834
min         0.000000
25%         0.000000
50%         5.630000
75%        17.820000
max       210.210000
Name: Commision, dtype: float64

--------------------------------
Mean is:  15.080996155190423
Median is:  5.63
Mode is:  0.0
--------------------------------
```

Distribution of Commision is: -



Boxplot of Commision is: -

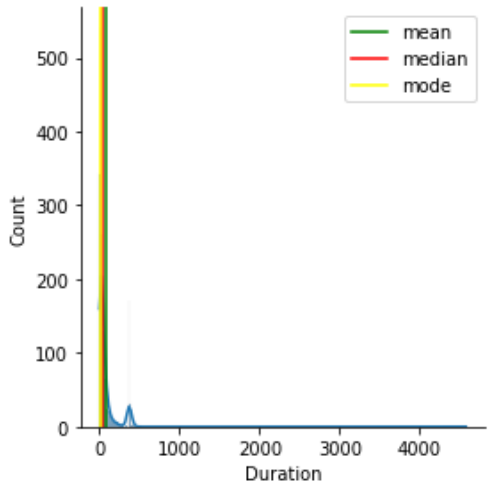


The distribution seems to be completely right skewed, with mean>median>mode. Mode is 0 on this case, which means for max products there is no commission. The commission column also contains outliers.

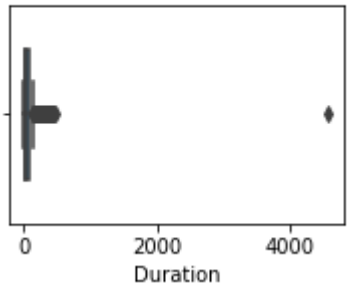
3. Duration

```
Description of Duration  is: -
count    2861.000000
mean       72.120238
std       135.977200
min        -1.000000
25%        12.000000
50%        28.000000
75%        66.000000
max      4580.000000
Name: Duration, dtype: float64

------------------------------
Mean is:  72.12023767913317
Median is:  28.0
Mode is:  10
```

Distribution of Duration is: -



Boxplot of Duration is: -

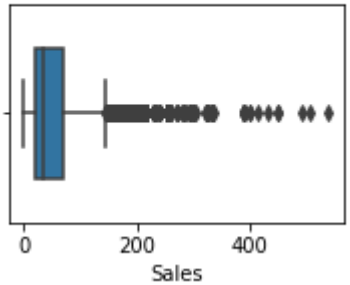


Absolutely right skewed distribution. Mean way greater than median and mode. The boxplot confirms presence of outliers.

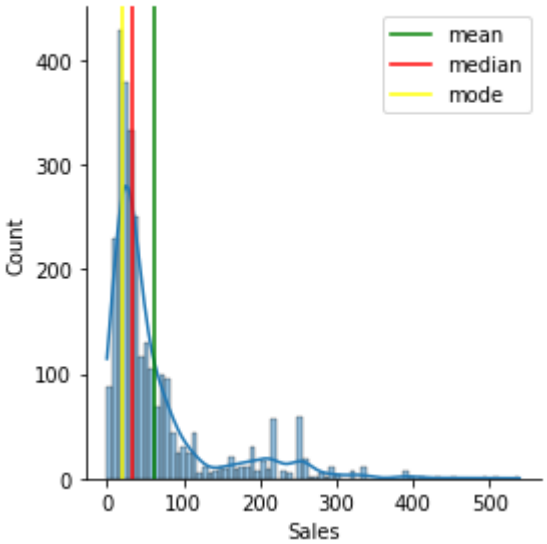
4.Sales

```
Description of Sales  is: -
count    2861.000000
mean       61.757878
std        71.399740
min         0.000000
25%        20.000000
50%        33.500000
75%        69.300000
max       539.000000
Name: Sales, dtype: float64

------------------------------
Mean is:  61.757878364208416
Median is:  33.5
Mode is:  20.0
```

Boxplot of Sales is: -



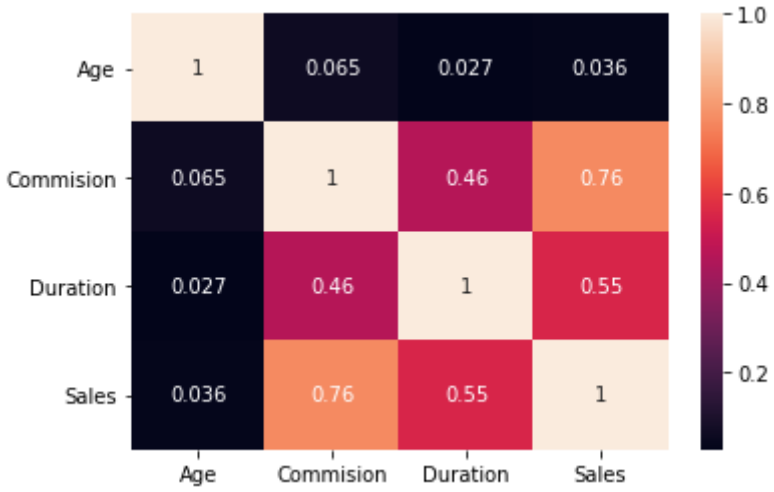
Distribution of Sales is: -



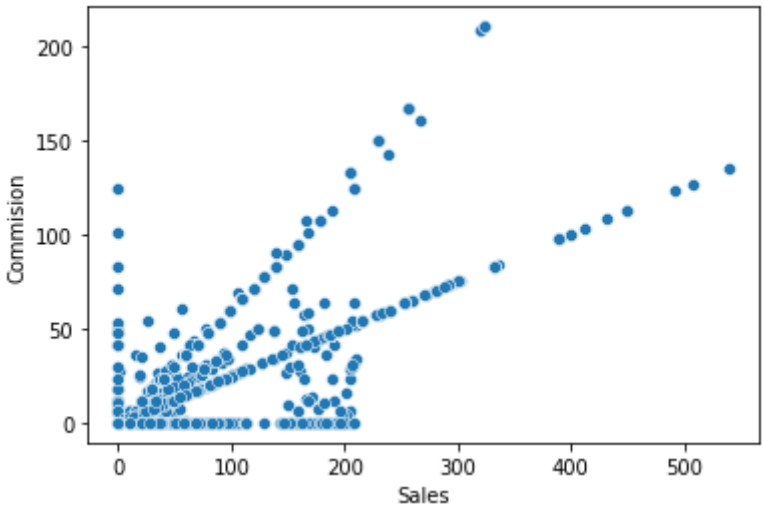
Mean>Median>Mode indicating a right skewed distribution. Boxplot confirms presence of outliers.

Bivariate analysis

Plotting the heatmap we can observe the following: -



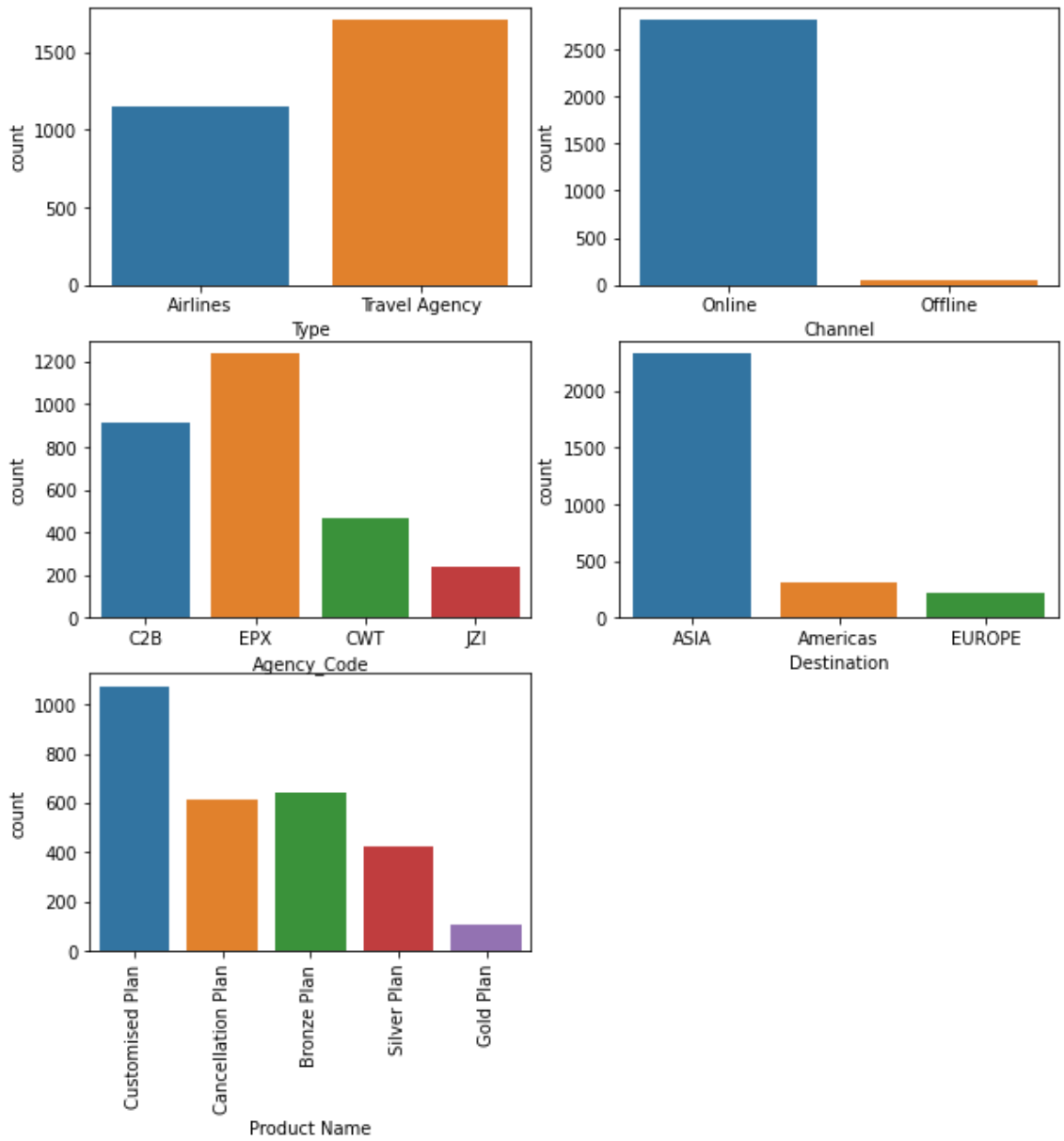
Variables sales and Commission are only the ones with some significant corelation. The relation can also be visualised with a scatterplot: -



Although the trend is not very definite but still, we can settle to agree that the corelation is significant.

Let’s now look at the influence of categorical variables.

Count-plot of categorical variables: -



The specific value counts are displayed below: -

```
df.Type.value_counts()
```

```
Travel Agency    1709
Airlines         1152
```

```
df.Channel.value_counts()
```

```
Online     2815
Offline      46
```

```
df['Product Name'].value_counts()
```

```
Customised Plan      1071
Bronze Plan           645
Cancellation Plan     615
Silver Plan           421
Gold Plan             109
```

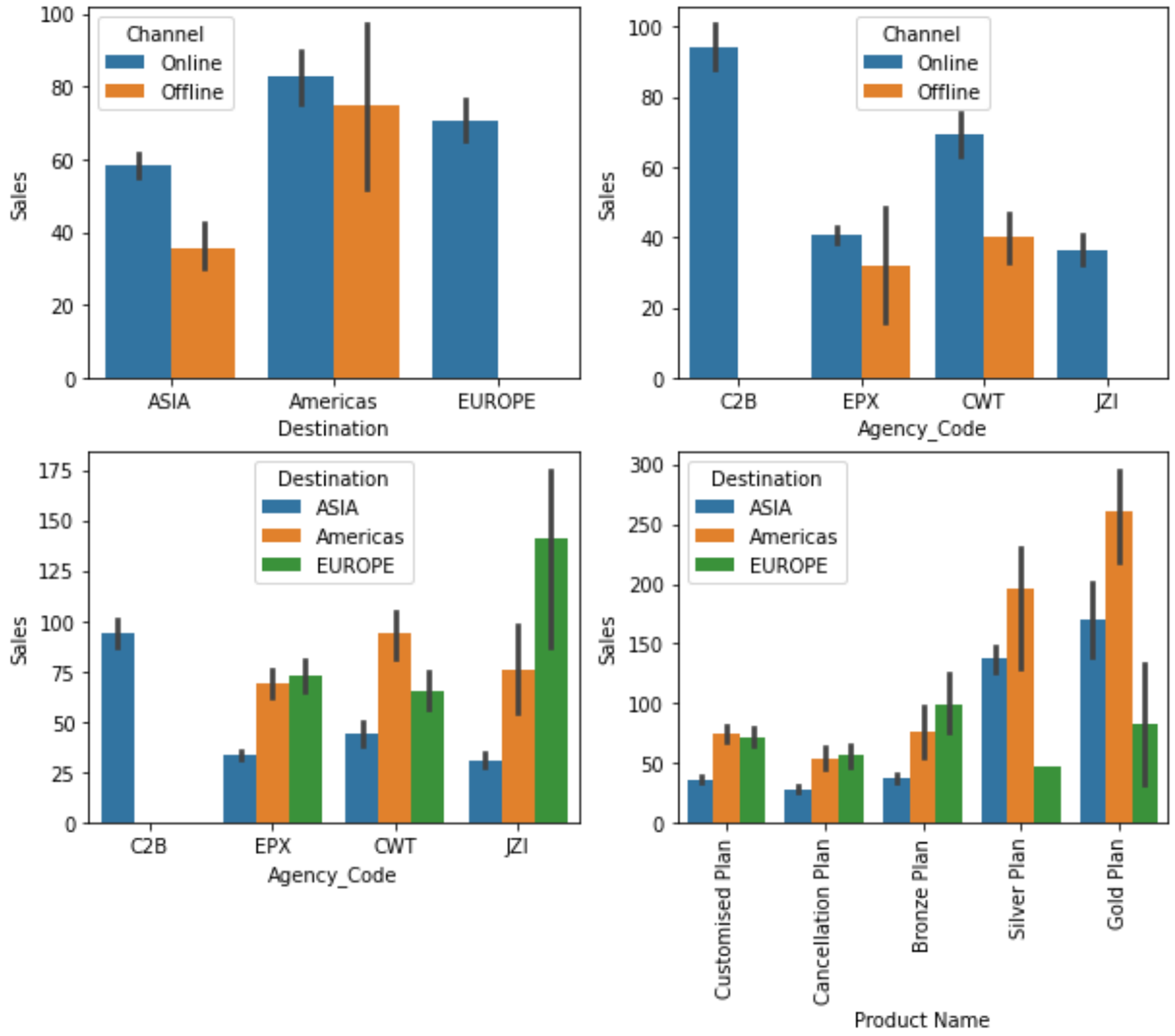
```
df['Destination'].value_counts()
```

```
ASIA        2327
Americas     319
EUROPE       215
```

```
df['Agency_Code'].value_counts()
```

```
EPX     1238
C2B      913
CWT      471
JZI      239
```

Factors affecting Sales: -



Looking at the graph above we can infer the following: -

1. Sales for America is the most, that too for online channel
2. The agency C2B makes the most sales, through online channel, it doesn't even have an offline one.
3. In general, the online channel contributes to sales more than the offline one.
4. The gold plan seems costly, and contributes the most to the sales

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

Looking at the data we can see that some columns are of object types, they need to be coded accordingly for the algorithms to ingest.

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

```
Age              int64
Agency_Code     object
Type            object
Claimed         object
Commision      float64
Channel         object
Duration         int64
Sales          float64
Product Name    object
Destination     object
```

These are the above data types before label encoding. Using label encoding, they are converted to int type: -

Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
48	0	0	0	0.70	1	7	2.51	2	0
36	2	1	0	0.00	1	34	20.00	2	0
39	1	1	0	5.94	1	3	9.90	2	1
36	2	1	0	0.00	1	4	26.00	1	0
33	3	0	0	6.30	1	53	18.00	0	0

```
Age              int64
Agency_Code       int8
Type              int8
Claimed           int8
Commision      float64
Channel           int8
Duration         int64
Sales          float64
Product Name      int8
Destination       int8
```

We can move ahead now, as we have successfully converted the variables to suitable data types.

The data is split into two, namely X and Y. Y contains the target variable: “Claimed”, while X contains the rest of the data.

X contains the following: -

Age	Agency_Code	Type	Commision	Channel	Duration	Sales	Product Name	Destination
48	0	0	0.70	1	7	2.51	2	0
36	2	1	0.00	1	34	20.00	2	0
39	1	1	5.94	1	3	9.90	2	1
36	2	1	0.00	1	4	26.00	1	0
33	3	0	6.30	1	53	18.00	0	0

Y contains only the target variable: -

```
0       0
1       0
2       0
3       0
4       0
       ..
2995    1
2996    0
2997    0
2998    1
2999    0
```

Looking at the distribution of 1’s and 0’s: -

```
df.Claimed.value_counts()
```

```
0    1947
1     914
```

```
df.Claimed.value_counts(normalize = True)
```

```
0    0.680531
1    0.319469
```

The data is not completely balanced, but yes, the distribution seems quite practical. In reality we won’t get a 50-50 idealistic balance.

The once claiming the insurance comprise of almost 32% of data, while, those not claiming comprise of 68%.

Further using the train-test split, the data is split up into training and testing, with training data = 70% and testing = 30%.

The distribution looks like following: -

Training data: -

	Age	Agency_Code	Type	Commision	Channel	Duration	Sales	Product Name	Destination
1300	69	0	0	6.00	1	7	15.0	0	0
2332	36	2	1	0.00	1	29	35.0	2	0
900	60	1	1	41.58	1	8	69.3	2	1
207	36	0	0	9.75	1	70	39.0	4	0
2310	36	2	1	0.00	1	39	51.0	1	2

Train labels: -

```
1300    1
2332    1
900     0
207     1
2310    0
```

Testing data: -

	Age	Agency_Code	Type	Commision	Channel	Duration	Sales	Product Name	Destination
642	31	1	1	0.00	0	402	97.0	2	0
219	68	2	1	0.00	1	60	29.0	1	0
1685	42	0	0	21.00	1	11	84.0	4	0
962	44	1	1	23.76	1	51	39.6	2	0
800	50	1	1	35.64	1	111	59.4	2	0

Test labels: -

```
642     0
219     0
1685    0
962     0
800     1
```

BUILDING THE CART MODEL: -

The cart model is build using Gini index and hyper parameters tuned using the grid search function.

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(random_state=1),
             param_grid={'max_depth': [7, 8, 9, 10],
                         'min_samples_leaf': [30, 45, 60, 90],
                         'min_samples_split': [90, 135, 180, 270]})
```

The min sample leaf is decided based on 1-3% of data; min split is kept 3 times of the min samples leaf. Max depth is speculated from 7-10, with cross validation = 10.

The best parameters are then used to feed the algorithm, which are as follows: -

```
grid_search_dt.best_params_
```

```
{'max_depth': 7, 'min_samples_leaf': 30, 'min_samples_split': 270}
```

The most important variables as per the CART models are: -

	IMP
Agency_Code	0.581006
Sales	0.297166
Product Name	0.045823
Commision	0.043860
Duration	0.018858
Age	0.013286
Type	0.000000
Channel	0.000000
Destination	0.000000

The agency code and Sales are the most significant variables.

Further predictions on the training and testing data are made (in terms of probability)

Training data: -

	0	1
0	0.847222	0.152778
1	0.672414	0.327586
2	0.837500	0.162500
3	0.424603	0.575397
4	0.823529	0.176471

Testing data: -

	0	1
0	0.573171	0.426829
1	0.971223	0.028777
2	0.274112	0.725888
3	0.837500	0.162500
4	0.837500	0.162500

BUILDING THE RANDOM FOREST

Following hyperparameters are used for the grid search cv to find the best parameters. Due to the computation limitations, it was difficult to further fine tune the model.

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=1),
             param_grid={'max_depth': [5, 7, 9], 'max_features': [2, 3, 4],
                         'min_samples_leaf': [30, 45, 60],
                         'min_samples_split': [90, 135, 180],
                         'n_estimators': [101, 201]})
```

The best parameters chosen by the algorithm were: -

```
{'max_depth': 9,
 'max_features': 3,
 'min_samples_leaf': 30,
 'min_samples_split': 90,
 'n_estimators': 201}
```

The most important variables according to random forest: -

	IMP
Agency_Code	0.293556
Product Name	0.226727
Sales	0.184545
Commision	0.121972
Duration	0.066967
Type	0.062397
Age	0.031534
Destination	0.012303
Channel	0.000000

Agency code, product name, sales, commission are the most significant variables.

Further predictions on train and test data are made (based on probability)

Training data: -

	0	1
0	0.698424	0.301576
1	0.765351	0.234649
2	0.744826	0.255174
3	0.433769	0.566231
4	0.867297	0.132703

Testing data: -

	0	1
0	0.622367	0.377633
1	0.920324	0.079676
2	0.320209	0.679791
3	0.769770	0.230230
4	0.709676	0.290324

BUILDING THE ANN

Just to ensure that the entire data is on the same scale, the data is scaled before building the ANN, as we know, ANN depends on weights, and unscaled data could give pathetic results.

Train data: -

	Age	Agency_Code	Type	Commision	Channel	Duration	Sales	Product Name	Destination
0	2.887642	-1.262611	-1.198133	-0.356790	0.119098	-0.448215	-0.653755	-1.313381	-0.447753
1	-0.216661	0.716831	0.834632	-0.588042	0.119098	-0.298327	-0.370328	0.243391	-0.447753
2	2.041014	-0.272890	0.834632	1.014535	0.119098	-0.441402	0.115749	0.243391	1.246769
3	-0.216661	-1.262611	-1.198133	-0.212257	0.119098	-0.018990	-0.313643	1.800164	-0.447753
4	-0.216661	0.716831	0.834632	-0.588042	0.119098	-0.230196	-0.143587	-0.534995	2.941292

Test data: -

	Age	Agency_Code	Type	Commision	Channel	Duration	Sales	Product Name	Destination
0	-0.687010	-0.272890	0.834632	-0.588042	-8.396428	2.242961	0.508295	0.243391	-0.447753
1	2.793573	0.716831	0.834632	-0.588042	0.119098	-0.087121	-0.455356	-0.534995	-0.447753
2	0.347758	-1.262611	-1.198133	0.221340	0.119098	-0.420963	0.324067	1.800164	-0.447753
3	0.535897	-0.272890	0.834632	0.327716	0.119098	-0.148439	-0.305140	0.243391	-0.447753
4	1.100316	-0.272890	0.834632	0.785596	0.119098	0.260348	-0.024548	0.243391	-0.447753

While scaling the data for ANN, we transform and fit the training data while we just fit the testing data.

Following were the parameters given to the grid search algorithm: -

```
GridSearchCV(cv=10, estimator=MLPClassifier(random_state=1),
             param_grid={'activation': ['logistic', 'relu'],
                         'hidden_layer_sizes': [(100, 100, 100),
                                                (500, 400, 300)],
                         'max_iter': [10000], 'solver': ['sgd', 'adam'],
                         'tol': [0.1, 0.01]})
```

After some hit and trial, the size of the hidden layer was narrowed down to two. The best parameter chosen was: -

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

Further predicting on train and test data: -

Train data: -

	0	1
0	0.581906	0.418094
1	0.830568	0.169432
2	0.801626	0.198374
3	0.439509	0.560491
4	0.877401	0.122599

Test data: -

	0	1
0	0.353300	0.646700
1	0.932529	0.067471
2	0.362354	0.637646
3	0.786004	0.213996
4	0.722460	0.277540

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, Write inferences on each model.

The performance metrics include the following: Classification report, confusion matrix, AUC, and ROC AUC score

CART PERFORMANCE METRICS: -

1. Classification report:

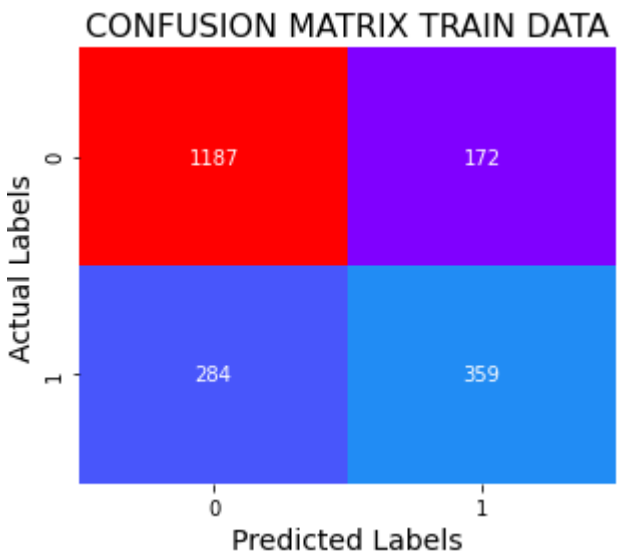
Classification report for train data: -

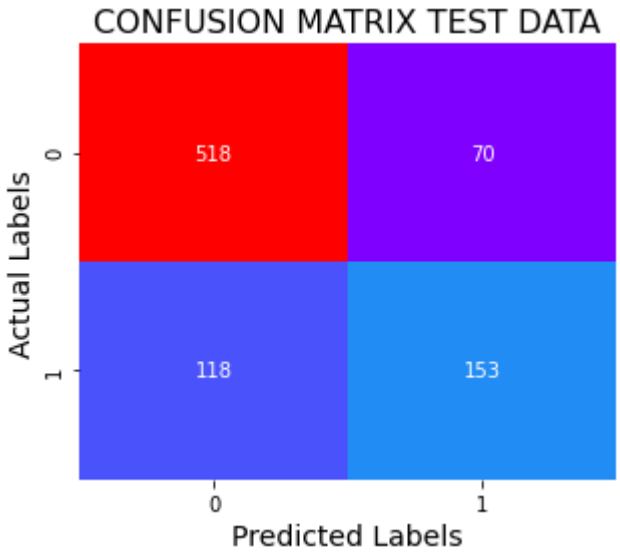
	precision	recall	f1-score	support
0	0.81	0.87	0.84	1359
1	0.68	0.56	0.61	643
accuracy			0.77	2002
macro avg	0.74	0.72	0.73	2002
weighted avg	0.76	0.77	0.77	2002

Classification report for test data: -

	precision	recall	f1-score	support
0	0.81	0.88	0.85	588
1	0.69	0.56	0.62	271
accuracy			0.78	859
macro avg	0.75	0.72	0.73	859
weighted avg	0.77	0.78	0.77	859

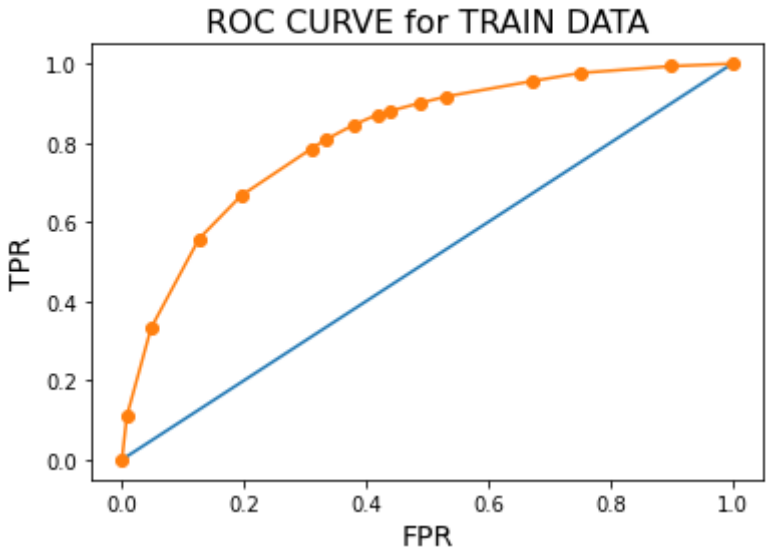
2. Confusion Matrix:



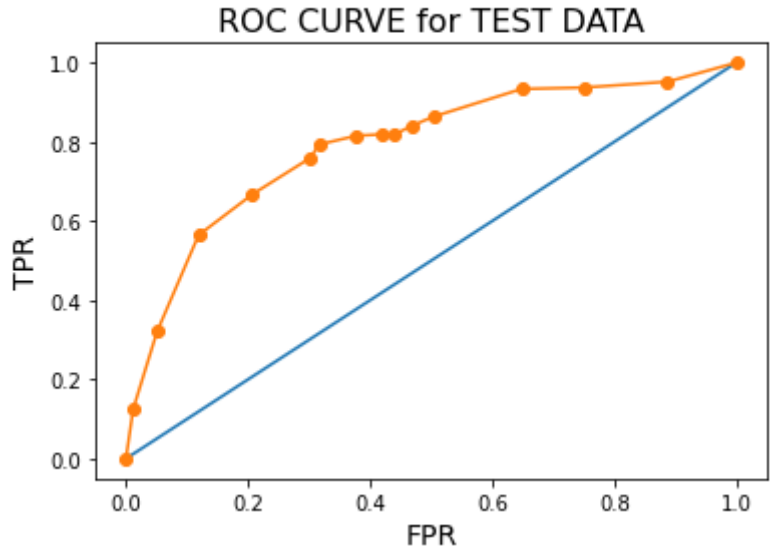


3. ROC curve and AUC score

AUC Score train Data: 0.8153986384188356



```
AUC Score test Data:  0.7914093681753143
```



Inferences: -

	CART TRAIN	CART TEST
ACCURACY	0.772228	0.781141
PRECISION	0.676083	0.686099
RECALL	0.558320	0.564576
F1 SCORE	0.611584	0.619433
AUC	0.815399	0.791409

The accuracy is good enough stating the model identified 77-78% of claims correctly, but increase in accuracy on the test data might be the case of overfitting. The recall is quite low indicating the low power of test. Precision is decent, able to detect the true positives. The AUC curve has reduced on the testing data, but sufficient on the train data. The nearness of weighted average and macro average negates the case of overfitting.

RANDOM FOREST PERFORMANCE METRICS: -

1. Classification report

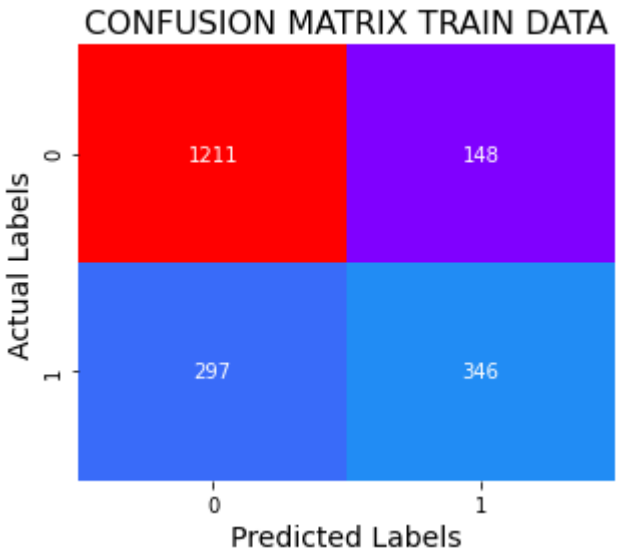
Classification report for Train data: -

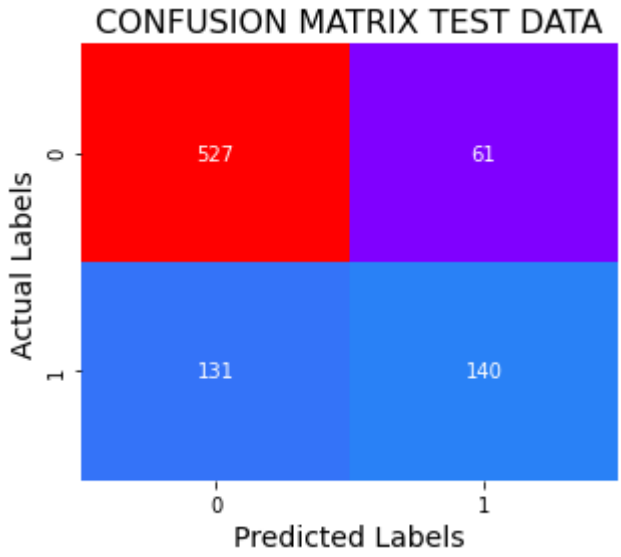
	precision	recall	f1-score	support
0	0.80	0.89	0.84	1359
1	0.70	0.54	0.61	643
accuracy			0.78	2002
macro avg	0.75	0.71	0.73	2002
weighted avg	0.77	0.78	0.77	2002

Classification report for Test data: -

	precision	recall	f1-score	support
0	0.80	0.90	0.85	588
1	0.70	0.52	0.59	271
accuracy			0.78	859
macro avg	0.75	0.71	0.72	859
weighted avg	0.77	0.78	0.77	859

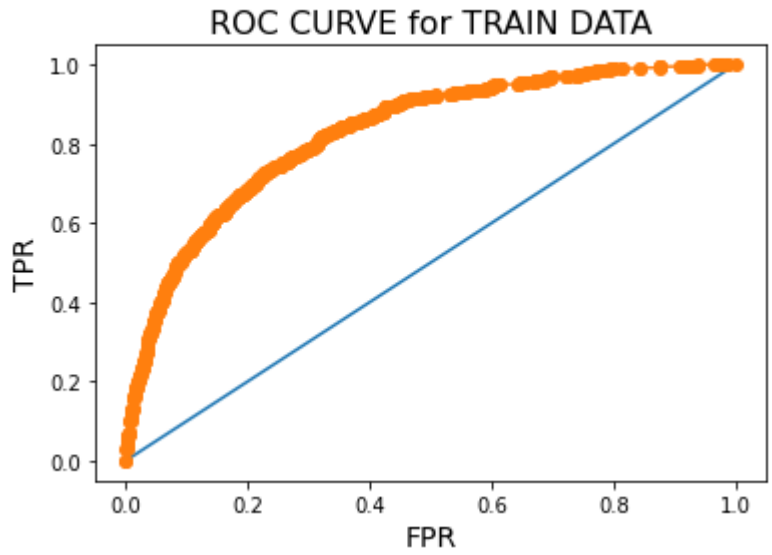
2. Confusion Matrix



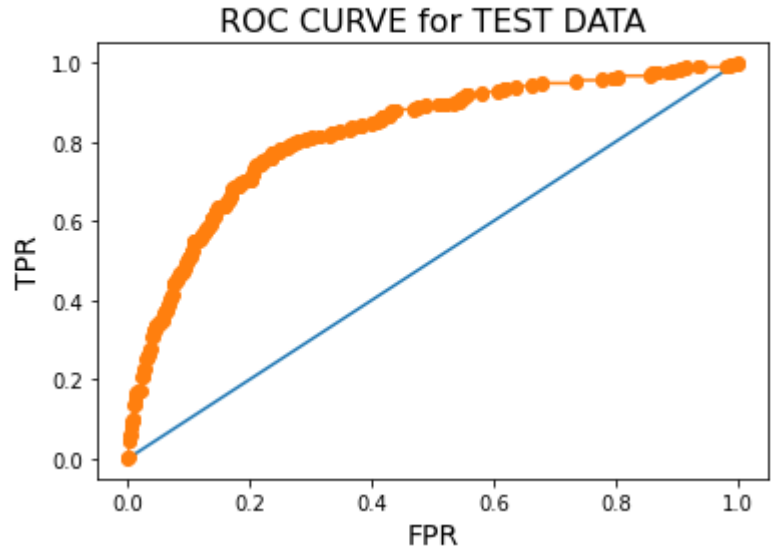


3. ROC Curve and AUC score

```
AUC Score train Data:  0.8281676102064801
```




```
AUC Score train Data:  0.8185292567211386
```



Inferences: -

	Random Forest TRAIN	Random Forest TEST
ACCURACY	0.777722	0.776484
PRECISION	0.700405	0.696517
RECALL	0.538103	0.516605
F1 SCORE	0.608619	0.593220
AUC	0.828168	0.818529

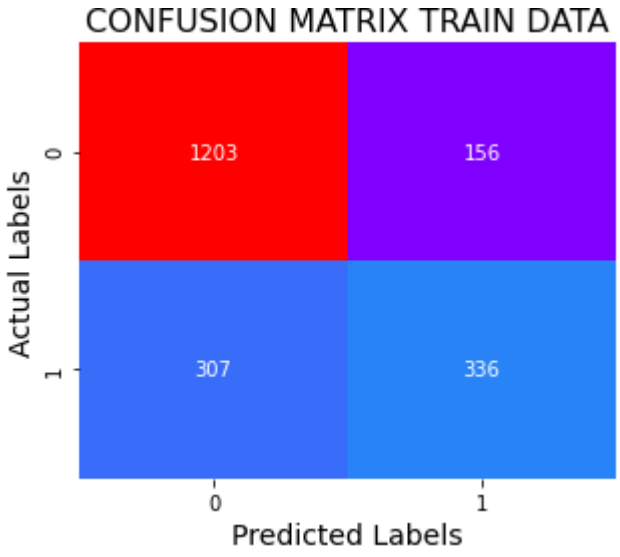
The model seems to be accurate, as the accuracy for train and test is same. The precision and AUC are also good. The only drawback being the recall, which is a trade-off to make. The macro and weighted averages suggest, there is no overfitting.

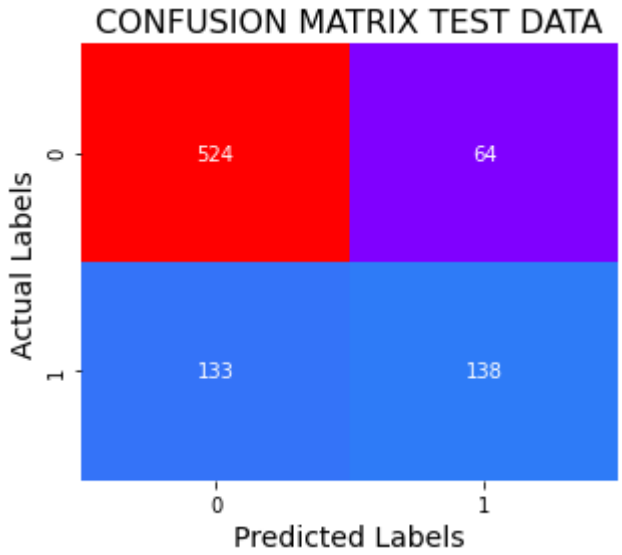
ANN PERFORMANCE METRICS

1. Classification report

```
Classification report for train data: -
              precision    recall  f1-score   support

           0       0.80      0.89      0.84      1359
           1       0.68      0.52      0.59       643

    accuracy                           0.77      2002
   macro avg       0.74      0.70      0.72      2002
weighted avg       0.76      0.77      0.76      2002


Classification report for train data: -
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       588
           1       0.68      0.51      0.58       271

    accuracy                           0.77       859
   macro avg       0.74      0.70      0.71       859
weighted avg       0.76      0.77      0.76       859
```

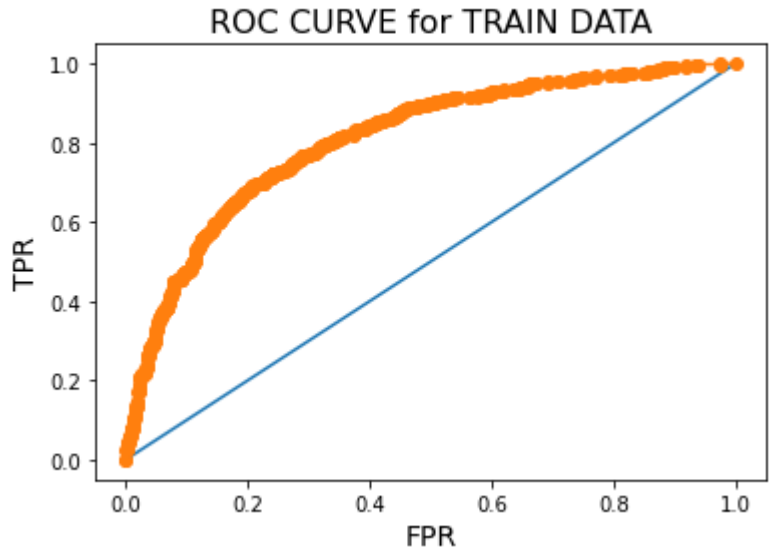
2. Confusion Matrix

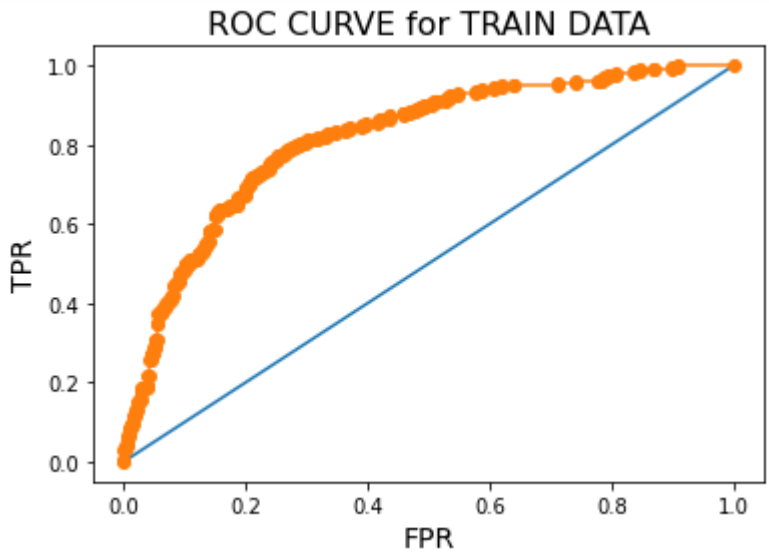




3. ROC Curve and AUC Score

```
AUC Score train Data:  0.809453593748033
```





Inferences: -

	ANN TRAIN	ANN TEST
ACCURACY	0.768731	0.770664
PRECISION	0.682927	0.683168
RECALL	0.522551	0.509225
F1 SCORE	0.592070	0.583510
AUC	0.809454	0.815762

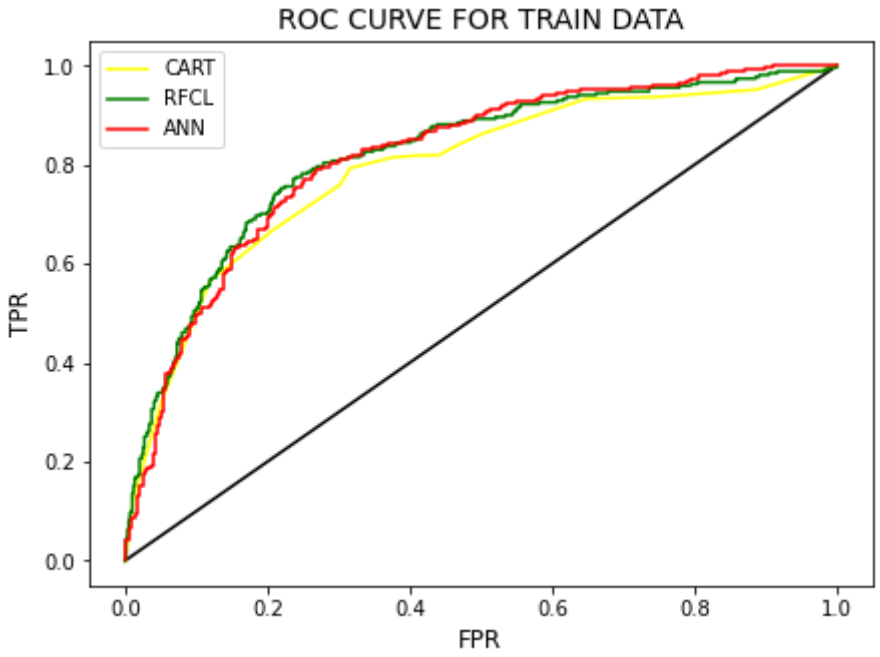
The model has comparatively low accuracy performance recall and AUC. Looking at the macro average and weighted average, model doesn't seem to overfit.

2.4 Final Model - Compare all models on the basis of the performance metrics in a structured tabular manner (3 pts). Describe on which model is best/optimized (2 pts).

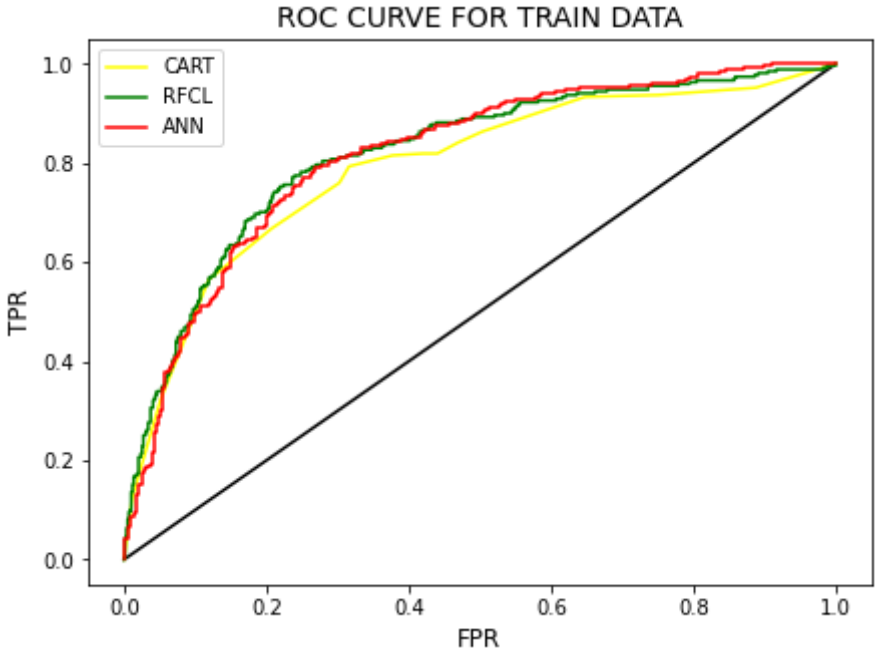
The individual characteristics of every model is stored in variables and a data frame is created to compare all the models at once: -

	CART TRAIN	CART TEST	Random Forest TRAIN	Random Forest TEST	ANN TRAIN	ANN TEST
ACCURACY	0.77	0.78	0.78	0.78	0.77	0.77
PRECISION	0.68	0.69	0.70	0.70	0.68	0.68
RECALL	0.56	0.56	0.54	0.52	0.52	0.51
F1 SCORE	0.61	0.62	0.61	0.59	0.59	0.58
AUC	0.82	0.79	0.83	0.82	0.81	0.82

ROC Curve for Training data: -



ROC Curve for Test data:



Looking at the above table and roc curves, all the three models are ready to put to use, but the best performing model is the Random forest classifier. It has the max accuracy, precision, and AUC score. Though there is some trade-off to make between the precision and recall as they are inversely proportional, but still the recall is not compromised to a greater extent.

Considering the context of business problem, we want our model to be accurate with less type1 and type2 error. The model should ensure that it should not classify the claimed insurance as not claimed, i.e., the false negative must be less. Less false negatives meaning high sensitivity/recall. On the other hand, we can do with a type1 error i.e., model classifying a not claimed insurance to be claimed. Type 2 is more critical here.

2.5 Based on your analysis and working on the business problem, detail out appropriate insights and recommendations to help the management solve the business objective.

Let's look at the several factors pertaining to the insurance claim.

Claimed	Age	Commision	Duration	Sales
No	38.461222	10.075978	53.421674	45.393035
Yes	37.656455	25.742670	111.951860	96.618217

Checking at the mean values for claimed and not claimed, products with more sales, duration and commission are more likely to make a claim.

Similarly observing more crosstabs: -

Agency_Code	C2B	CWT	EPX	JZI
Claimed				
No	361	331	1046	209
Yes	552	140	192	30

Type	Airlines	Travel Agency
Claimed		
No	570	1377
Yes	582	332

Channel	Offline	Online
Claimed		
No	29	1918
Yes	17	897

Destination	ASIA	Americas	EUROPE	All
Claimed				
No	1563	231	153	1947
Yes	764	88	62	914
All	2327	319	215	2861

Product Name	Bronze Plan	Cancellation Plan	Customised Plan	Gold Plan	Silver Plan	All
Claimed						
No	396	573	818	39	121	1947
Yes	249	42	253	70	300	914
All	645	615	1071	109	421	2861

Insights and recommendations: -

1. The agency C2B has more claims, which should be explored further. The specific reasons should be found out
2. Air travellers are making more claims. Air travel is considered the safest travel in the world with 0.07 passenger deaths per billion miles(cityam.com). Still there are so many claims.
3. The online channel is more prone to claims
4. People travelling to Asia have made more claims so far
5. The silver plan and Bronze plan witness more claims. However, the cancellation plan has received very few claims.

Observing the feature importance provided by the CART and random forests: -

	IMP
Agency_Code	0.581006
Sales	0.297166
Product Name	0.045823
Commision	0.043860
Duration	0.018858
Age	0.013286
Type	0.000000
Channel	0.000000
Destination	0.000000

	IMP
Agency_Code	0.293556
Product Name	0.226727
Sales	0.184545
Commision	0.121972
Duration	0.066967
Type	0.062397
Age	0.031534
Destination	0.012303
Channel	0.000000

From the importance features, we can notice that agency code, sales product name and commission are the most important variables for prediction.

The agency C2B should seriously deploy a team to research the reasons for massive claims.

Claims are more prone to more sales. Might suggest the customers are felt cheated on some grounds.

The silver and bronze products are more prone to claims, they might not be to the standard expected by the customers. These must incorporate a cancellation policy in the plans, as the cancellation policy plan has yielded the least claims.