2021

DATA MINING

BUSINESS REPORT

The following business report provides brief explanation of approach to each and every problem given in the project/assignment and relative information and recommendations with regards to solving the problem.

PRABHJOT KAUR CHAHAL
PGP-DSBA -Online Mar_C 2021/GREAT LEARNING'S
7/1/2021

Contents

1. PROBLEM 1- CLUSTERING 4

1.1 Read the data, do the necessary initial steps, and exploratory data analysis (Univariate, Bi- variate, and multivariate analysis). 4

1.2 Do you think scaling is necessary for clustering in this case? Justify 15

1.3 Apply hierarchical clustering to scaled data. Identify the number of optimum clusters using Dendrogram and briefly describe them 17

1.4 Apply K-Means clustering on scaled data and determine optimum clusters. Apply elbow curve and silhouette score. Explain the results properly. Interpret and write inferences on the finalized clusters. 20

1.5 Describe cluster profiles for the clusters defined. Recommend different promotional strategies for different clusters. 23

2. PROBLEM 2 - CART-RF-ANN 27

2.1 Read the data, do the necessary initial steps, and exploratory data analysis (Univariate, Bi-variate, and multivariate analysis). 27

2.2 Data Split: Split the data into test and train, build classification model CART, Random Forest, Artificial Neural Network 36

2.3 Performance Metrics: Comment and Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score, Classification reports for each model. 47

2.4 Final Model: Compare all the three models and write an inference which model is best /optimized.. 54

2.5 Inference: Based on the whole Analysis, what are the business insights and recommendations? 56

TABLE OF FIGURES

Figure 1 7
Figure 2 7
Figure 3 7
Figure 4 8
Figure 5 8
Figure 6 8
Figure 7 9
Figure 8 11
Figure 9 13
Figure 10 15
Figure 11 16
Figure 12 17
Figure 13 18
Figure 14 21
Figure 15 22
Figure 16 30
Figure 17 30
Figure 18 30
Figure 19 31
Figure 20 31
Figure 21 32
Figure 22 32
Figure 23 33
Figure 24 34
Figure 25 35
Figure 26 36
Figure 27 37
Figure 28 38
Figure 29 39
Figure 30 42
Figure 31 48
Figure 32 50
Figure 33 52
Figure 34 54
Figure 35 55

1.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.

Data Dictionary for Market Segmentation:

- 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) Data

1.1Read the data, do the necessary initial steps, and exploratory data analysis (Univariate, Bi- variate, and multivariate analysis).

Load the required libraries and read the csv file using the pd.read command.

	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
...	...	...	...	...	...	...	...
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

210 rows × 7 columns

DATA DESCRIPTION:

- **INFORMATION OF THE DATASET:**

1) We use info function to know the index and column's data type.
   - All the variables are continuous and of float data type.

```
<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
```

```
spending                      float64
advance_payments              float64
probability_of_full_payment   float64
current_balance               float64
credit_limit                  float64
min_payment_amt               float64
max_spent_in_single_shopping  float64
dtype: object
```

- **CHECKING THE SHAPE, MISSING VALUES & DUPLICATES IN THE DATA SET:**

  - We use shape function to get the number of rows and columns in the dataset
  - The Shape of the data is (210, 7).

  - **Isnull** function helps you to identify at variable level, the missing value in the given dataset. The missing values or "NA" needs to be checked and dropped from the dataset for the ease of evaluation and null values can give errors or discrepancies in results.
  - There were No Null Values in the Data.

  - **.duplicated** function helps to identify, if there is any duplicated series value.
  - We have not observed any duplication of records in given dataset.

```
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
```

- **SUMMARY OF THE DATASET:**

1) The summary of the dataset can be determined using **.describe** () function.
- It gives you the statistical details like count, mean, Standard deviation, Q1, Q2, Q3, max and min of different variables in the data set.

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

- From the summary, it can be seen that the Std-Deviation of spending is high when compared with the other variables. Also, there are no null values present in any of the variables.
- Out of all the variables, standard deviation is the lowest for "probability_of_full_payment" and highest for "spending"
- The data looks much aligned and good.

- We see that, mean for (spending and advance payments) are almost equal and the same goes for (max spent in single shopping and current balance) along with(credit limit and min payment amt)

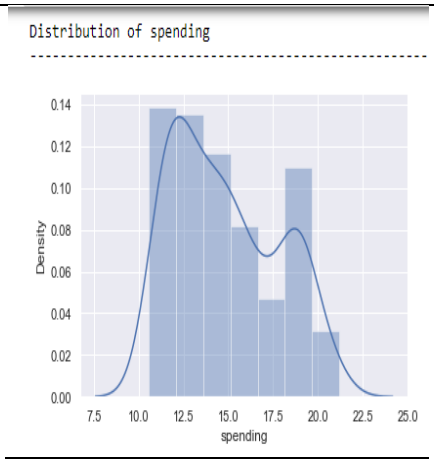
- Observed outliers in two variables called min_payment_amount and probability_of_full_payment.

- As per the statistical summary, mean and median are almost the same for all the variables.

UNIVARIATE ANALYSIS :

✓ For spending:

```
Description of spending
--------------------------------
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
```



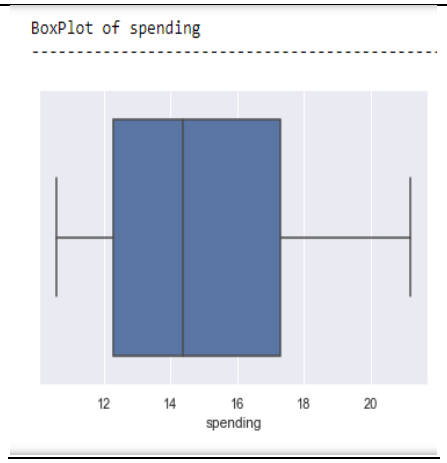
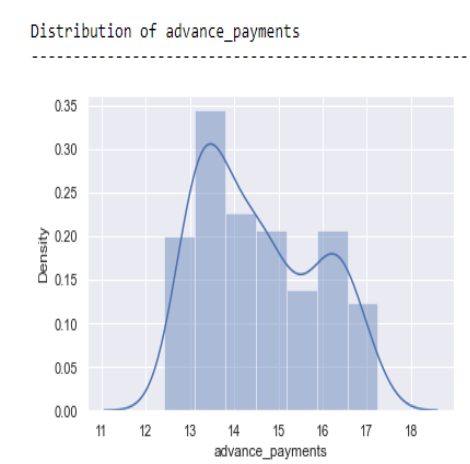


Figure 1

✓ For advance_payments:

```
Description of advance_payments
---------------------------------------
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
```



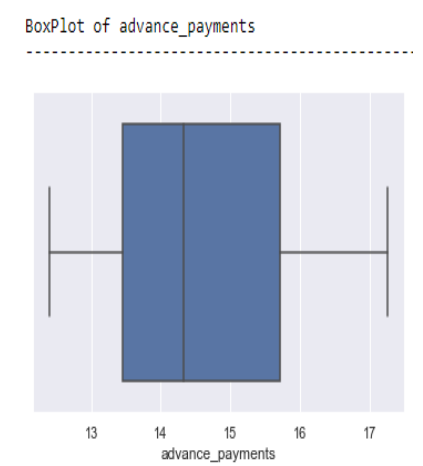
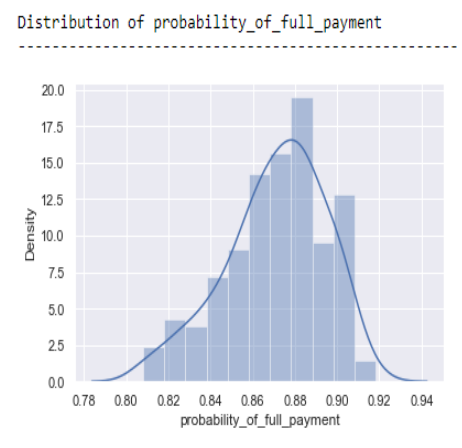


Figure 2

✓ For probability_of_full_payment:

```
Description of probability_of_full_payment
-------------------------------------------
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
```



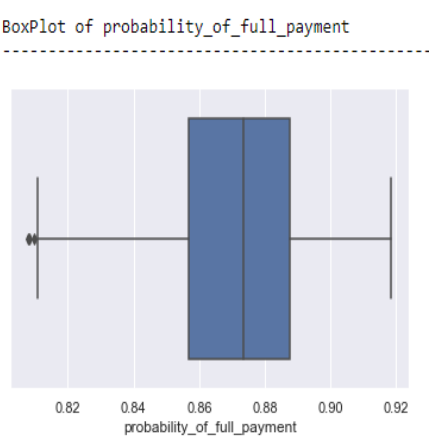
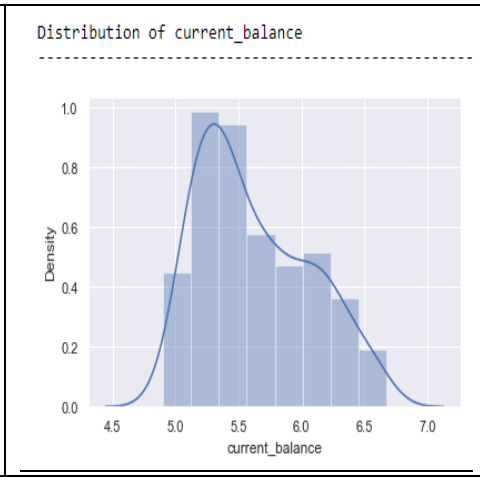


Figure 3

✓ For current_balance:

```
Description of current_balance
-----------------------------------
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
```



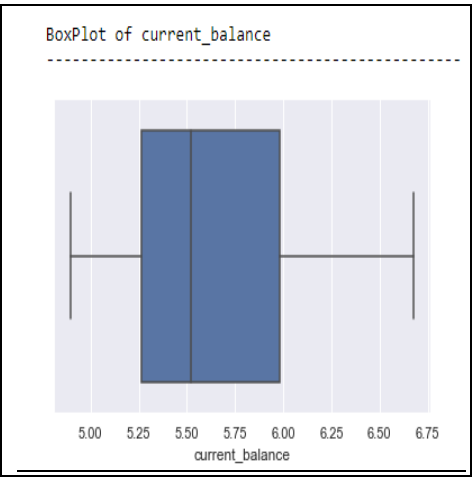
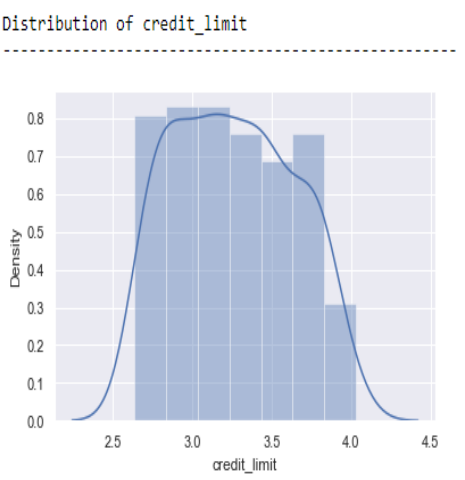


Figure 4

✓ For credit_limit:

```
Description of credit_limit
-----------------------------------
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
```



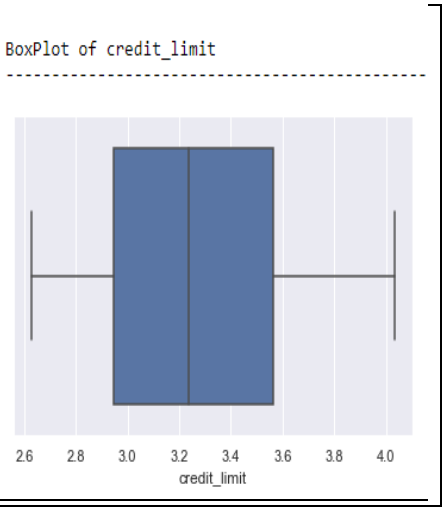
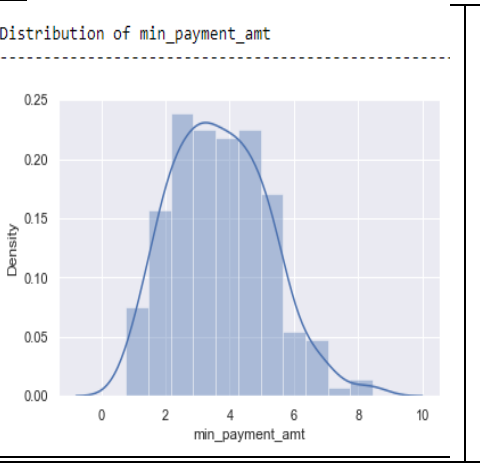


Figure 5

✓ For min_payment_amount:

```
Description of min_payment_amt
-----------------------------------
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
```



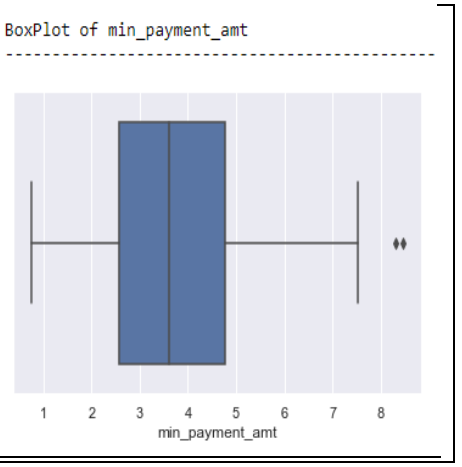


Figure 6

✓ For max_spent_in_single_shopping:

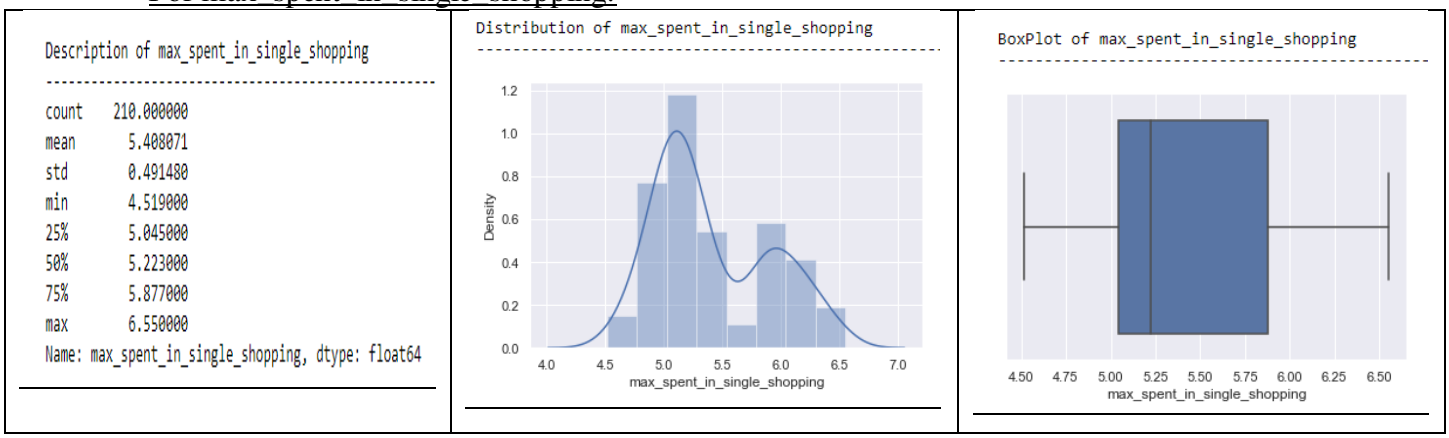


Figure 7

INFERENCE:

- ✓ Boxplot is an indication of how the values in the data are spread out and also indicates if any outlier is present.
- ✓ Displot gives the univariant set of observations.
- ✓ From above analysis, we observed outliers in two variables called as min_payment_amount and probability of full payment.

BY CHECKING OUTLIERS AND DETECTING OUTLIERS, WE FOUND OUT THAT:

- ✓ Outlier present in min_payment_amt clearly indicates 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 very low which indicates there are few customers whose probability to pay to the bank in full is on the lower side of average.
- ✓ Probability_of_full_payment variables also have very small Outliers; hence the requirement for Outlier Treatment is not necessary.

WITH THE HELP OF BOXPLOT:

- ✓ After plotting Boxplots for all the variables,we can conclude that a few outliers are present in the variable namely, **min_payment_amt** which means that there are only a few customers whose minimum payment amount falls on the higher side on an average. Since only one of the seven variable have a very small outlier value, hence there is no need to treat the outliers. This small value will not create any difference in our analysis.
- ✓ We can conclude from the above graphs that the most of the customers in our data have a higher spending capacity, high current balance in their accounts and these customers spent a higher amount during a single shopping event. Majority of the customers have a higher probability to make full payment to the bank.

WITH THE HELP OF DISTPLOT:

- ✓ The variables are rightly skewed except for the variable probability of full payment.
- ✓ Probability of _full_payment is -0.537954 (negatively skewed). From, the displot Visualization the data distribution takes on from 0.80 to 0.92 which is a good factor. Only this Variable is negatively skewed apart from other variables which are positively skewed.

✓ SKEWNESS as follow:

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

BI-VARIATE ANALYSIS : PAIR PLOT

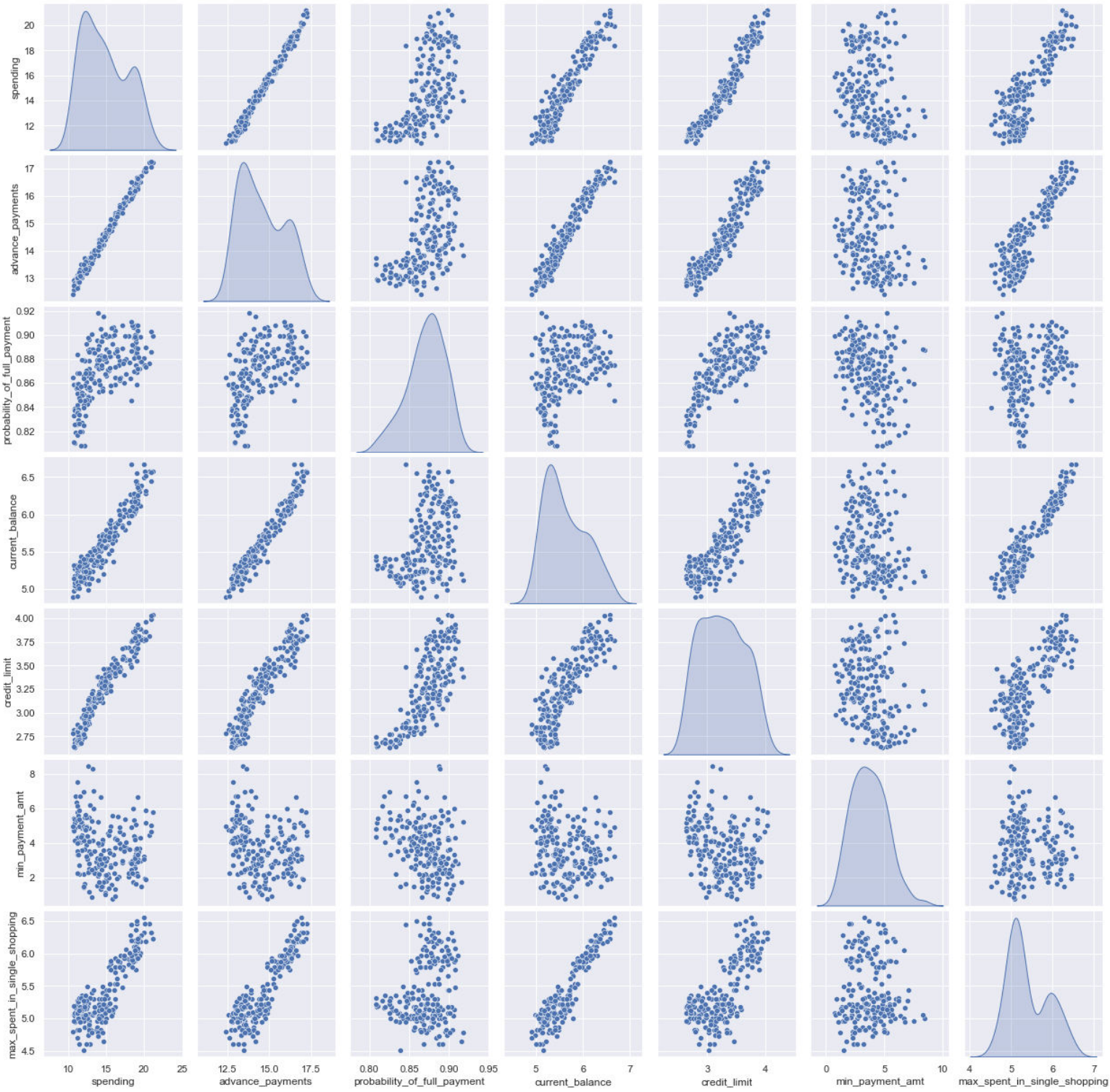


Figure 8

- ✓ Helps to check for Multi Collinearity
- ✓ A Scatter plot gives us an idea of the association between two variables

- ✓ With the help of the above pair plot we can understand the Univariate and Bivariate trends for all the variables in the dataset.
- ✓ In the above plot, scatter diagrams are plotted for all the numerical columns in the dataset. A scatter plot is a visual representation of the degree of correlation between any two columns. The pair plot function in seaborn makes it very easy to generate joint scatter plots for all the columns in the data.
- ✓ There exist a linear relationship between advance_payments and credit_limit i.e. are highly correlated.
- ✓ Also max_spent_in_single_shopping and current_balance are highly correlated.

\

MULTI-VARIATE ANALYSIS :

✓ CORRELATION:

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_
spending	1.000000	0.994341	0.608288	0.949985	0.970771	-0.229572	
advance_payments	0.994341	1.000000	0.529244	0.972422	0.944829	-0.217340	
probability_of_full_payment	0.608288	0.529244	1.000000	0.367915	0.761635	-0.331471	
current_balance	0.949985	0.972422	0.367915	1.000000	0.860415	-0.171562	
credit_limit	0.970771	0.944829	0.761635	0.860415	1.000000	-0.258037	
min_payment_amt	-0.229572	-0.217340	-0.331471	-0.171562	-0.258037	1.000000	
max_spent_in_single_shopping	0.863693	0.890784	0.226825	0.932806	0.749131	-0.011079	

✓ HEATMAP:

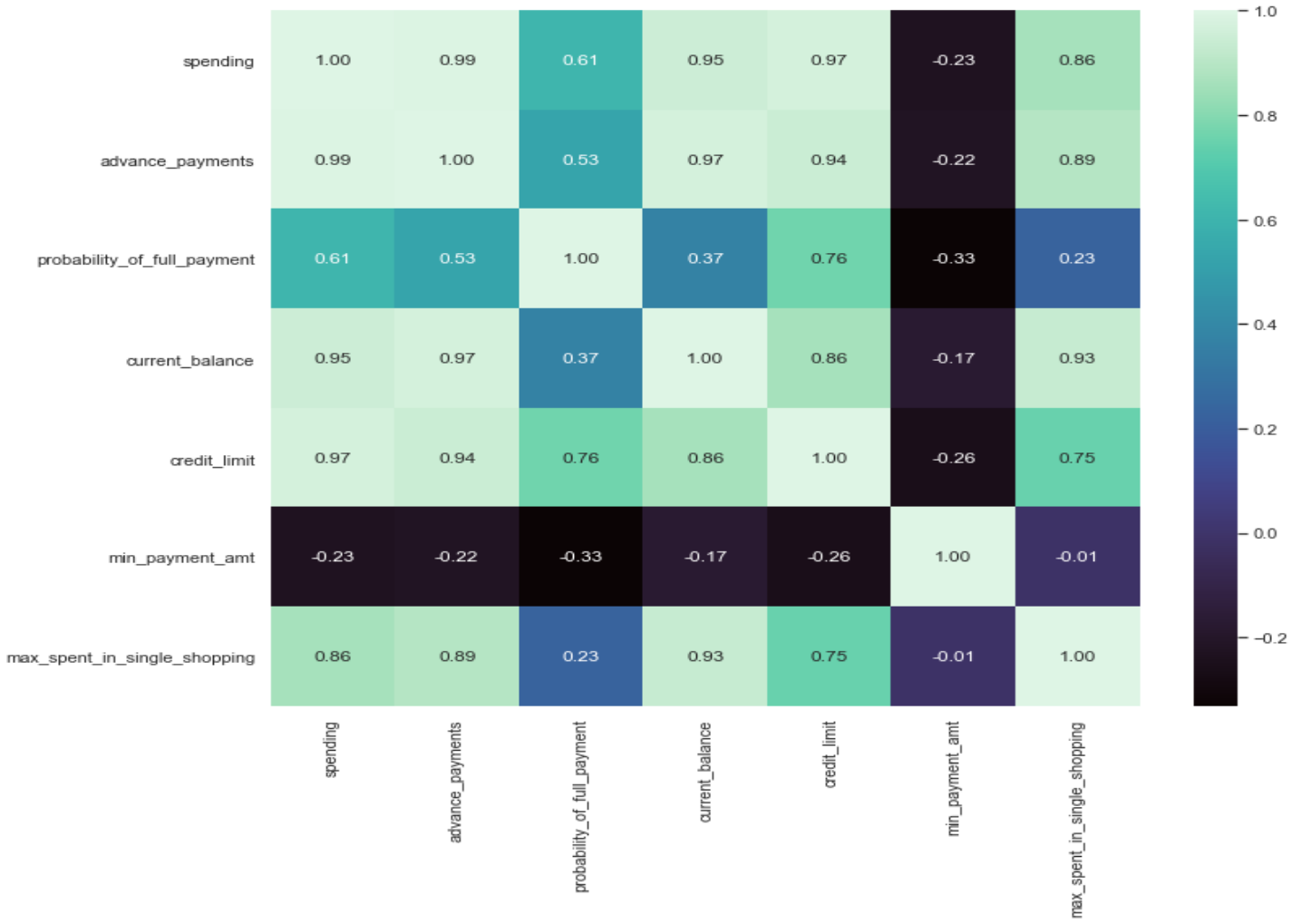


Figure 9

INFERENCE:

As per the Heat Map, we can conclude that the following variables are highly correlated:

- Spending and advance_payments, spending and current_balance, spending and credit_limit
- Advance_payment and current_balance, advance_payment and credit limit
- Current balance and max spent in single shopping

By this we can conclude that the customers who are spending very high have a higher current balance and high credit limit. Advance payments and maximum expenditure done in single shopping are done by majority of those customers who have high current balance in their bank accounts.

Probability of full payments are higher for those customers who have a higher credit limit.

Minimum payment amount is not correlated to any of the variables; hence, it is not affected by any changes in the current balance or credit limit of the customers.

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

Yes, the Data Scaling is necessary in this case, because feature scaling is an important pre-processing step for many machine learning algorithms.

Scaling involves rescaling the features such that they have the properties of a standard normal distribution with a mean of zero and a standard deviation of one. This makes our data **unit less.**

Machine learning algorithm just sees number — if there is a vast difference in the range says few ranging in Ten thousand and few ranging in the hundreds, and it makes the underlying assumption that higher ranging numbers have superiority of some sort. So, more significant number starts playing a more decisive role while training the model.

If we consider 5th Observation, it has Spending of 17.99 which is in Thousands and Advance payments for same observation have 15.86 which are in Hundreds. These two represent completely two different units which can be easily understood by humans, but for a model as a feature, it treats both as same.

So, this more significant number starts playing a more decisive role while training the model. Thus, feature scaling is needed to bring every feature in the same footing without any upfront importance.

We can use zscore in order to standardise the data as follow:

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
0	1.754355	1.811968	0.178230	2.367533	1.338579	-0.298806	2.328998
1	0.393582	0.253840	1.501773	-0.600744	0.858236	-0.242805	-0.538582
2	1.413300	1.428192	0.504874	1.401485	1.317348	-0.221471	1.509107
3	-1.384034	-1.227533	-2.591878	-0.793049	-1.639017	0.987884	-0.454961
4	1.082581	0.998364	1.196340	0.591544	1.155464	-1.088154	0.874813

Graph before scaling the data:

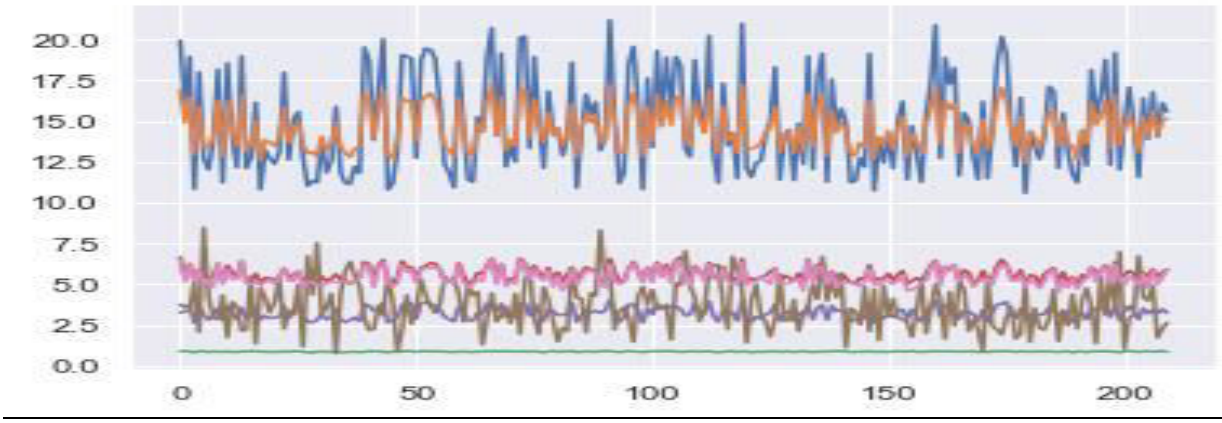


Figure 10

Graph after scaling the data:

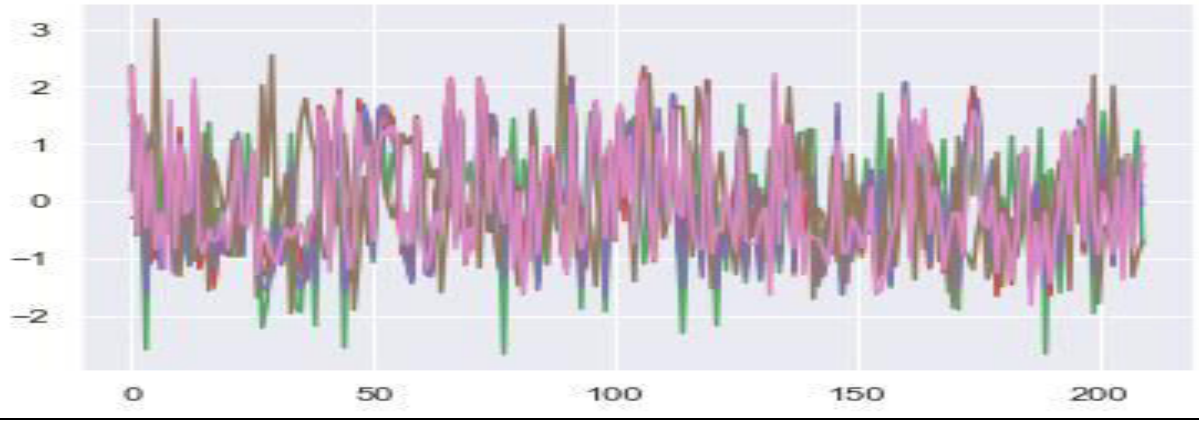


Figure 11

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

IN HIERARCHICAL CLUSTERING:

Records are sequentially grouped to create clusters, based on distances between records and distances between clusters. Hierarchical clustering also produces a useful graphical display of the clustering process and results, called a **dendrogram**.

Strengths of Hierarchical Clustering:

- There is no assumptions on the number of clusters
- Any desired number of clusters can be obtained by 'cutting' the dendrogram at the optimum level.
- Hierarchical clustering may correspond to meaningful taxonomies.

Hierarchical clustering relies on clustering techniques, to find a hierarchy of clusters, where this hierarchy resembles to a dendrogram as shown:

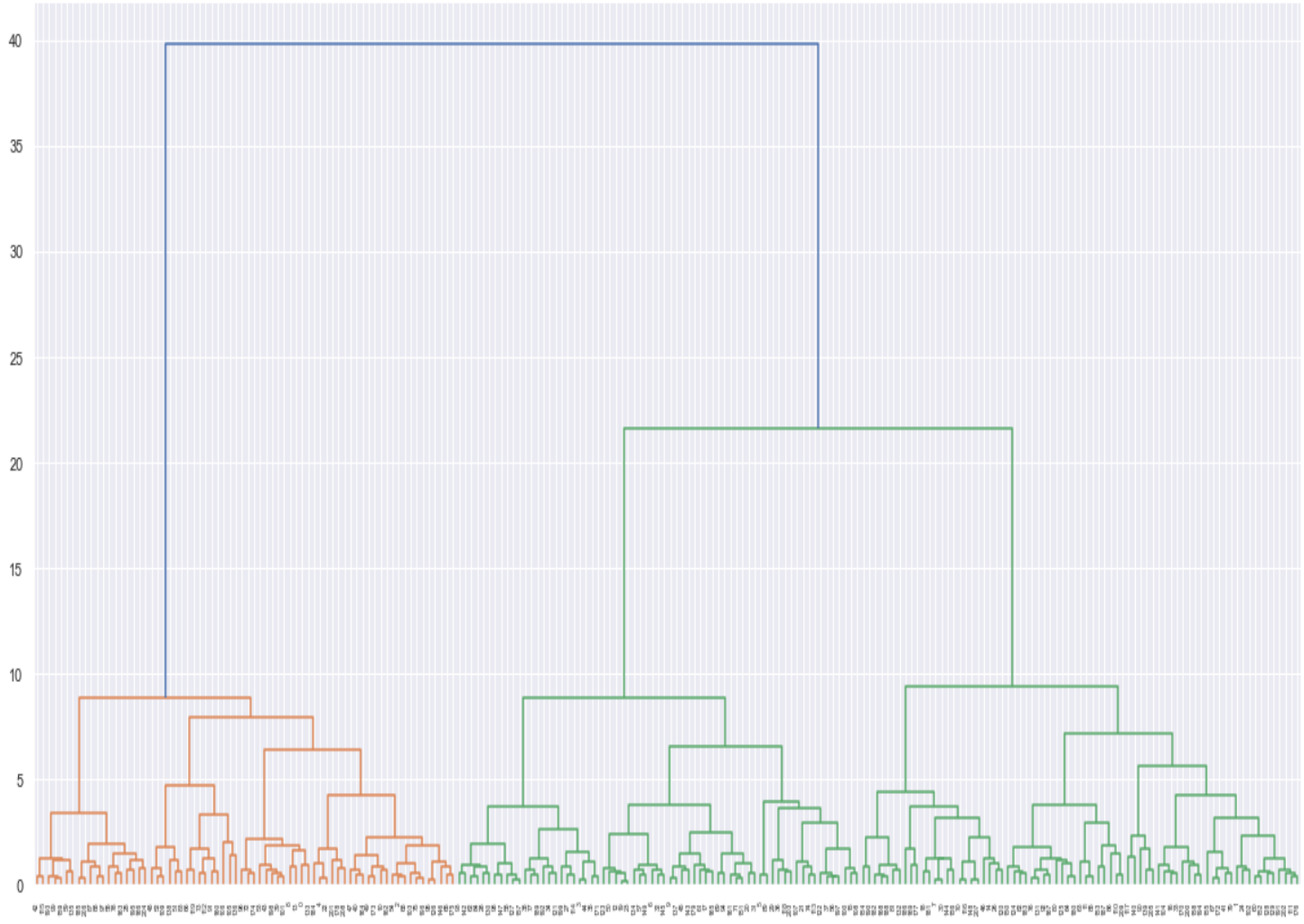


Figure 12

The above mentioned dendrogram signify that all the data points have been clustered into different clusters by Ward's Method.

To find the optimum number of cluster to solve the problem, further we can use truncatemode = lastp.

Cutting the Dendrogram with suitable clusters using 'Ward' as the linkage method:

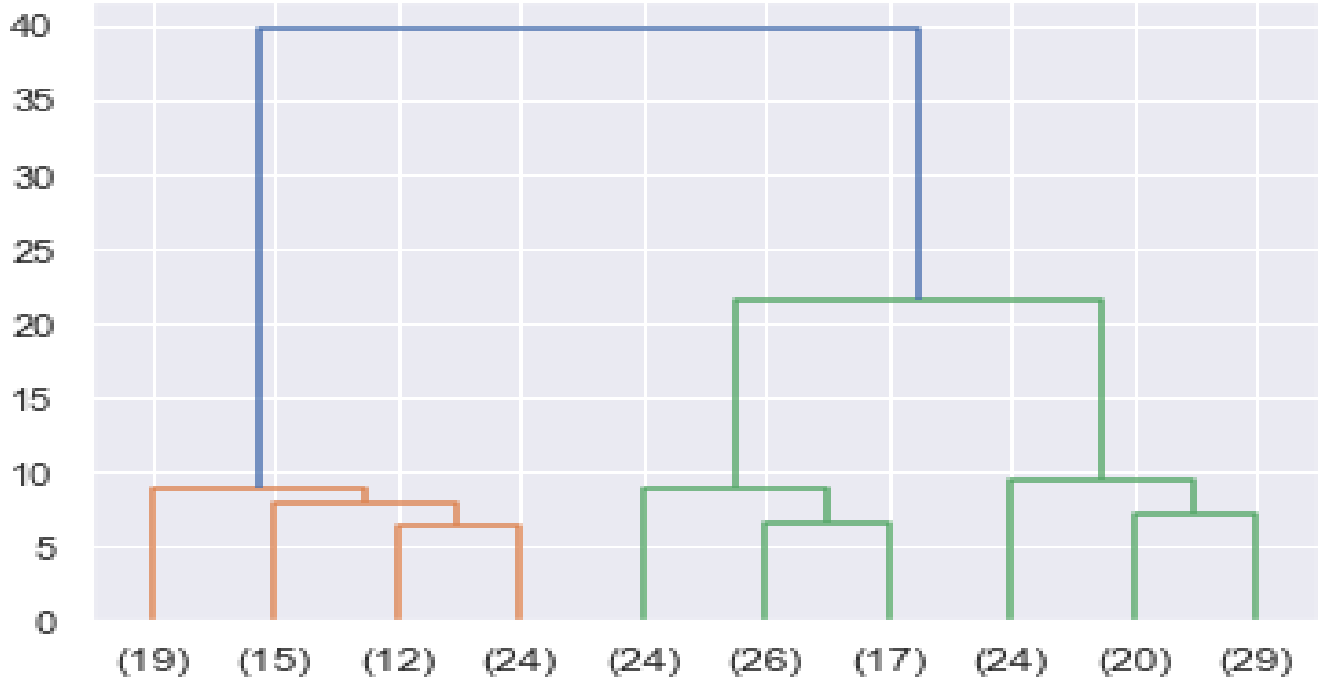


Figure 13

Now, with the help of above method, we can observe that the data has been clustered into 3 different Clusters. We can also visualize that the maximum customers fall into the green Cluster.

Next step is to append these clusters into the dataset. We can use: criterion = 'maxclust' where a cut is defined based on the number of clusters as shown:

```
wardlinkage_clusters_1 = fcluster(wardlinkage, 3, criterion='maxclust')
wardlinkage_clusters_1
```

```
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)
```

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

```
wardlinkage_clusters_2 = fcluster(wardlinkage, 10, criterion='distance')
wardlinkage_clusters_2
```

```
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)
```

From here we can see that, both the criterion have resulted in the same output. The distance criterion gives a glance to find the optimal number of clusters. With the distance criterion we can see that the optimum number of clusters would be 3.

Appending these observations into the original data.

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

OBSERVATIONS:

- After plotting the dendrogram and further analysis based on hierarchical clustering, I had proceeded with 3 clusters.
- These 3 clusters reveal a pattern based on high/medium/low spending with max_spent_in_single_shopping (high value item) and probability_of_full_payment (payment made).

  Cluster 1: High Value Customers, with highest avg. spending, current_balance, credit limit, max_spent_in_single_shopping and max probability_of_full_payment.

  Cluster 2: Low Value customer, with least avg. spending and current_balance and least probability of full payment but with max avg. min_payment_amt volumes.

  Cluster 3: Average Value customers, with medium avg. spending and current_balance and medium probability of full payment but the least avg min_payment amt volumes.

1.4 Apply K-Means clustering on scaled data and determine optimum clusters. Apply elbow curve and silhouette score. Explain the results properly. Interpret and write inferences on the finalized clusters.

- A non-hierarchical approach to forming good clusters is to pre-specify a desired number of clusters, k
- The 'means' in the K-means refers to averaging of the data; that is, finding the centroid.
- K-means clustering is widely used in large dataset applications.
- K-means clustering is one of the unsupervised machine learning algorithms. K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.
- For the dataset, we will be using K-means clustering on scaled data and identify the clusters formed and use them further to devise tools to target each group separately.

Here we, apply K-Means Technique to the scaled dataset & identify the clusters formed. Then we will calculate the value of inertia and store it in WSS

CLUSTER OUTPUT FOR ALL OBSERVATIONS: We have 3 Clusters- 0, 1, 2.

```
#fitting the Kmeans
kmeans_3 = KMeans(n_clusters = 3,random_state=123)
kmeans_3.fit(scaled_data)
labels_3 = kmeans_3.labels_
labels_3
```

```
array([1, 0, 1, 2, 1, 2, 2, 0, 1, 2, 1, 0, 2, 1, 0, 2, 0, 2, 2, 2, 2, 2,
       1, 2, 0, 1, 0, 2, 2, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 1, 1, 0, 1, 1,
       2, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 0, 2, 2, 0, 0, 1,
       1, 0, 1, 2, 0, 2, 1, 1, 2, 1, 0, 2, 1, 0, 0, 0, 0, 1, 2, 0, 1, 0,
       1, 2, 0, 1, 0, 2, 2, 1, 1, 1, 2, 1, 0, 1, 0, 1, 0, 1, 1, 2, 2, 1,
       0, 0, 1, 2, 2, 1, 0, 0, 2, 1, 0, 2, 2, 2, 0, 0, 1, 2, 0, 0, 2, 0,
       0, 1, 2, 1, 1, 2, 1, 0, 0, 0, 2, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 0,
       2, 0, 0, 2, 0, 1, 1, 2, 1, 1, 1, 2, 0, 0, 0, 2, 0, 2, 0, 1, 1, 1,
       0, 2, 0, 2, 0, 0, 0, 0, 1, 1, 2, 0, 0, 2, 2, 0, 2, 1, 0, 1, 1, 2,
       1, 2, 0, 1, 0, 2, 1, 0, 1, 0, 0, 0])
```

The two methods to determine the optimal number of clusters are within sum of squares (wss) method and average silhouette scores method.

WSS METHOD:

```
wss =[]
```

```
for i in range(1,11):
    KM = KMeans(n_clusters=i,random_state=0)
    KM.fit(scaled_data)
    wss.append(KM.inertia_)
```

```
wss
```

```
[1469.9999999999995,
 659.1717544870411,
 430.65897315130064,
 371.5811909715524,
 327.5038437438554,
 289.46717056412893,
 267.5489624840095,
 240.84436687230493,
 221.55712182925916,
 206.4900073930761]
```

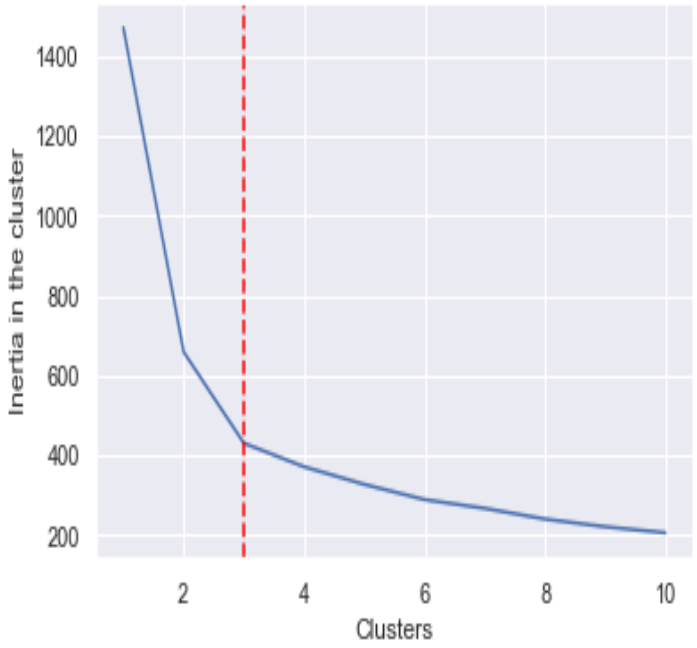


Figure 14

- The above gives a clear idea on the inertia values of Clusters from 1 to 11.
- From the above visualization, we can tell 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 :

```
silhouette_score(scaled_data,labels_3)
```

```
0.4007270552751298
```

```
silhouette_samples(scaled_data,labels_3).min()
```

```
0.002713089347678376
```

- The silhouette scores and silhouette widths are calculated using silhouette_samples and silhouette_score package from sklearn.metrics. The average silhouettes score is coming to be 0.400 and minimum silhouette score is 0.002. The silhouette score ranges from -1 to +1 and higher the silhouette score better the clustering.

```
scores

[0.46577247686580914,
 0.4007270552751298,
 0.3369008229710853,
 0.283139370986886,
 0.2903443931779972,
 0.26540705308849605,
 0.25193918824703637,
 0.2555836756561569,
 0.25951922181277004]
```

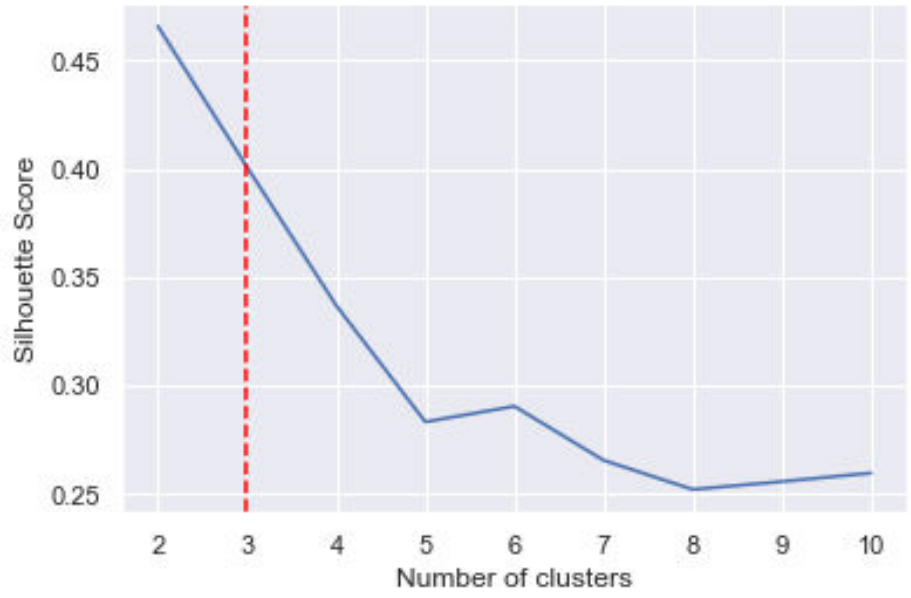


Figure 15

- As per the plot of Average silhouette scores it can be seen that the highest Average score is corresponding to k=3.
- The Silhouette Score for 3 Clusters is better than nearest possible Optimum Clusters which is n_clusters = 4 , n_clusters=5.Also the elbow curve seen above shows, there is no huge drop in values so, we can finalize on 3 clusters. Hence,
- Hence, as per both the methods i.e. within sum of squares and silhouette method we can conclude that the optimal number of k or clusters that needs to be taken for k-means clustering is 3.
- The 3 Group Cluster Solution gives a pattern based on each categories of spending.

ding	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Kmeans_Cluster	sil_width
19.94	16.92	0.8752	6.675	3.763	3.252	6.550	0	0.527315
15.99	14.89	0.9064	5.363	3.582	3.336	5.144	3	0.160653
18.95	16.42	0.8829	6.248	3.755	3.368	6.148	0	0.575506
10.83	12.96	0.8099	5.278	2.641	5.182	5.185	2	0.507451
17.99	15.86	0.8992	5.890	3.694	2.068	5.837	0	0.181433

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

To describe cluster profiles for the clusters defined and recommend different promotional strategies for different clusters:

3 GROUP CLUSTER VIA KMEANS

```
pd.Series(kmeans_3.labels_).value_counts().sort_index()
```

```
0    71
1    67
2    72
dtype: int64
```

cluster	1	2	3
spending	14.4	11.9	18.5
advance_payments	14.3	13.2	16.2
probability_of_full_payment	0.9	0.8	0.9
current_balance	5.5	5.2	6.2
credit_limit	3.3	2.8	3.7
min_payment_amt	2.7	4.7	3.6
max_spent_in_single_shopping	5.1	5.1	6.0

3 GROUP CLUSTER VIA HIERARCHICAL CLUSTERING

```
wardlinkage_clustered_1_data['3Clusters'].value_counts().sort_index()
```

```
1    70
2    67
3    73
Name: 3Clusters, dtype: int64
```

3Clusters	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
Freq	70.000000	67.000000	73.000000

OBSERVATIONS FOR HIERARCHICAL CLUSTERING:

Based on the clustering done, the groups can be defined basis 3 cluster profiles considering the spending in each group.

Cluster Profiles:

- Cluster 1: High Spending
- Cluster 3: Medium Spending
- Cluster 2: Low Spending

FOR CLUSTER 1:

- Spending is on higher side, averaging 18371.
- Probability of Full Payment is very high, averaging around 0.8844.
- It has the highest advance payments around 1614.
- Current Balance is around 6158 which is highest among three clusters.
- Credit Limit is also high for this cluster ranging around 36846.

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

FOR CLUSTER 2

- The average Spending cluster is on lower side, averaging 11872.
- 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.
- And also, minimum payment amount is found max in this cluster, it is around 494.
- Max_spent_in_single_shopping is around 5122 .

FOR CLUSTER 3

- Spending is 14199.
- Probability of Full Payment is little on the higher side, averaging around 0.879.
- Current Balance is around 5478 which is average among three clusters.
- Credit Limit is ranging in between for this cluster ranging around 32264.
- And also, minimum payment amount is found least in this cluster, it is around 261.
- Max_spent_in_single_shopping is the least around 5086.

OBSERVATIONS FOR K-MEANS CLUSTERING:

Based on the clustering done, the groups can be defined basis 3 cluster profiles considering the spending in each group.

Cluster Profiles:

- Cluster 1: Medium Spending
- Cluster 2: Low Spending
- Cluster 3: High Spending

FOR CLUSTER 1

- Spending is average spending cluster, averaging 14437.
- Probability of Full Payment is least among other Clusters, averaging around 0.8815.
- It has the Highest advance payments around 1433.
- Current Balance is around 5514. It is average when compared to other two clusters.
- Credit Limit is for this cluster ranging around 32592.
- Minimum payment is averaging 270. This is the least amongst others.
- And also, max_spent_in_single_shopping is around 5120.

FOR CLUSTER 2

- This is least Spending cluster, averaging 11856.
- 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.
- And also, minimum payment amount is found max in this cluster, it is around 474.
- Max_spent_in_single_shopping is around 5101.

FOR CLUSTER 3

- Spending is highest amongst other clusters, averaging 18495.
- Probability of Full Payment is little 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 ranging around 36975. It is highest amongst three clusters.
- And also, minimum payment amount is found least in this cluster, it is around 363.
- Max_spent_in_single_shopping is the highest around 6041.

PROMOTIONAL STRATEGY:

HIERARCHIAL CLUSTERS:

- **Cluster 1** are premium 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. Increase their credit limit & giving any reward points might increase their purchases.

- **Cluster 2** are low spending customers ;Poor spending customers has the least Credit limit and so maybe they spend least and also they have least current balance. Bank can also think of providing them offers for shopping at various websites & promotions that may increase their max_spent_in_single_shopping by tying up with basic amenities groups such as groceries, utilities etc. Customers should be given remainders for payments often. This Cluster need huge promotions schemes & targets must be set in order to move customers from this cluster to Medium spending Cluster.

- **Cluster 3** medium spending customers. 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. Increase spending habits by tying with ecommerce sites, premium hotels, luxury etc. Hence Bank should provide more promotional offers for customers & encourage them to spend more

FOR K-MEANS CLUSTERS:

- **Cluster 0 (converted as 1)** medium spending customers. Bank should give customers in this Cluster more promotional offers because there are more chances that these customers may move high spending category of customers. Increase spending habits by tying with ecommerce sites, premium hotels, luxury etc. Hence Bank should provide more promotional offers for customers & encourage them to spend more

- **Cluster 1 (converted as 2)** are low spending customers ;Poor spending customers has the least Credit limit and so maybe they spend least and also they have least current balance. Bank can also think of providing them offers for shopping at various websites & promotions that may increase their max_spent_in_single_shopping by tying up with basic amenities groups such as groceries, utilities etc. Customers should be given remainders for payments often. This Cluster need huge promotions schemes & targets must be set in order to move customers from this cluster to Medium spending Cluster.

- **Cluster 2 (converted as 2)** are premium customers .Bank should focus on Cluster 0 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 purchases.

2.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.

The dataset provided to us is stored as "insurance_part2_data.csv" which contains data of 3000 customers and 10 variables namely:

- Age: Age of insured
- Agency_Code: Code of tour firm
- Type: Type of tour insurance firms
- Claimed Target: Claim Status
- Commission: The commission received for tour insurance firm
- Channel: Distribution channel of tour insurance agencies
- Duration: Duration of the tour
- Sales: Amount of sales of tour insurance policies
- Product Name: Name of the tour insurance products
- Destination: Destination of the tour

2.1 Read the data, do the necessary initial steps, and exploratory data analysis (Univariate, Bi-variate, and multivariate analysis).

Importing the Dataset:

The dataset in question is imported in jupyter notebook using **pd.read_csv ()** function and will store the dataset in "**ins**". The top 5 rows of the dataset are viewed using **ins.head ()** function.

	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

DATA DESCRIPTION:

- INFORMATION OF THE DATASET:

1) We use info function to know the index and column data types.
2) The Info of the dataset shows that the dataset contains integer,float & object values.Henceforth,we need to convert the object datatypes into a numeric value

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 10 columns):
 #   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 
dtypes: float64(2), int64(2), object(6)
memory usage: 234.5+ KB
```

- CHECKING THE SHAPE & MISSING VALUES $ DUPLICATES IN THE DATA SET:
  - We use shape function to get the number of rows and columns in the dataset
  - The Shape of the Data is (3000, 10).
  - And out of these 10 columns, 4 columns have numerical data and 6 columns has categorical data.

  - Isnull function helps you identify the missing value in the given data set at variable level. The missing values or "NA" needs to be checked and dropped from the dataset for the ease ofevaluation and null values can give errors or disparities in results.
  - There were No Null Values in the Data.

  - .duplicated function helps identify, if there is any duplicated series value.
  - 139 are duplicate rows.
  - But after removing duplicates the shape for data set reduced to (2861,10)

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

- SUMMARY OF THE DATASET:

1) The summary of the dataset can be computed using **.describe** () function.
- It gives you the statistical details like count, mean, Standard deviation, Q1, Q2, Q3, max and min of different variables in the data set.

	count	mean	std	min	25%	50%	75%	max
Age	2861.0	38.204124	10.678106	8.0	31.0	36.00	43.00	84.00
Agency_Code	2861.0	1.280671	1.003773	0.0	0.0	2.00	2.00	3.00
Type	2861.0	0.597344	0.490518	0.0	0.0	1.00	1.00	1.00
Commision	2861.0	15.080996	25.826834	0.0	0.0	5.63	17.82	210.21
Channel	2861.0	0.983922	0.125799	0.0	1.0	1.00	1.00	1.00
Duration	2861.0	72.120238	135.977200	-1.0	12.0	28.00	66.00	4580.00
Sales	2861.0	61.757878	71.399740	0.0	20.0	33.50	69.30	539.00
Product Name	2861.0	1.666550	1.277822	0.0	1.0	2.00	2.00	4.00
Destination	2861.0	0.261797	0.586239	0.0	0.0	0.00	0.00	2.00

- ✓ There are in total 4 numeric variables & 6 Categorical Variables. The Most Preferred type is Travel Agency; Channel is online; Customized Plans are most preferred by the customers. Destination ASIA is the preferred destination by the customers.
- ✓ There are only 4 continuous variables Age, Commission, Duration and Sales, the result is shown for them only.
- ✓ Duration has negative value. It is not possible, seems like a wrong entry.
- ✓ Mean and Median of Commission & Sales varies significantly
- ✓ There are 9 independent variables and one target variable, i.e. 'Claimed'

- Look for unique values in all the nominal variables

```
AGENCY_CODE :  4
JZI     239
CWT     471
C2B     913
EPX    1238
Name: Agency_Code, dtype: int64


TYPE :  2
Airlines         1152
Travel Agency    1709
Name: Type, dtype: int64


CLAIMED :  2
Yes     914
No     1947
Name: Claimed, dtype: int64


CHANNEL :  2
Offline      46
Online     2815
Name: Channel, dtype: int64
```

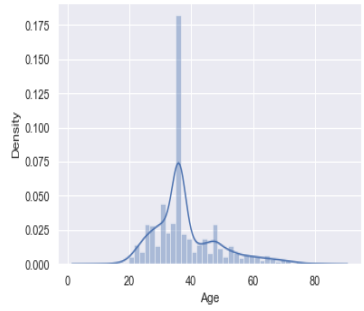
```
PRODUCT NAME :  5
Gold Plan              109
Silver Plan            421
Cancellation Plan      615
Bronze Plan            645
Customised Plan       1071
Name: Product Name, dtype: int64


DESTINATION :  3
EUROPE        215
Americas      319
ASIA         2327
Name: Destination, dtype: int64
```

- ✓ Agency code EPX has a frequency of 1365
- ✓ Customized plan is the most sought plan by customers
- ✓ Destination ASIA seems to be most sought destination place by customers.

UNIVARIATE ANALYSIS :

For Age:

```
Description of Age
----------------------------------------
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

Distribution of Age
----------------------------------------
```



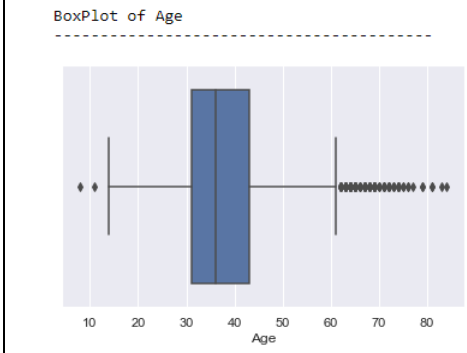
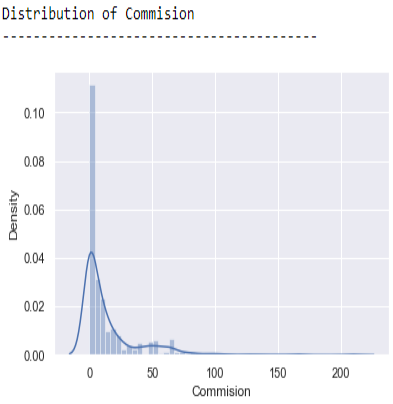


Figure 16

For Commission:

```
Description of Commision
----------------------------------------
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

Distribution of Commision
----------------------------------------
```



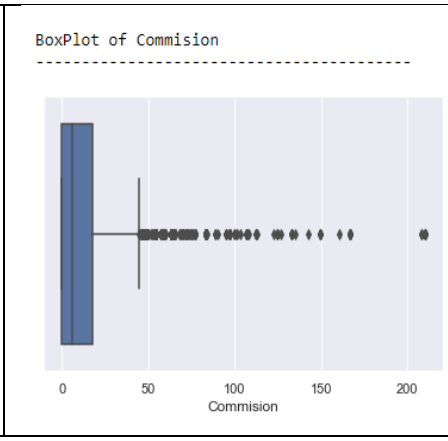
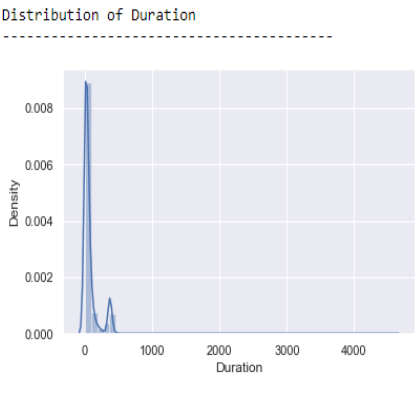


Figure 17

For Duration:

```
Description of Duration
----------------------------------------
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

Distribution of Duration
----------------------------------------
```



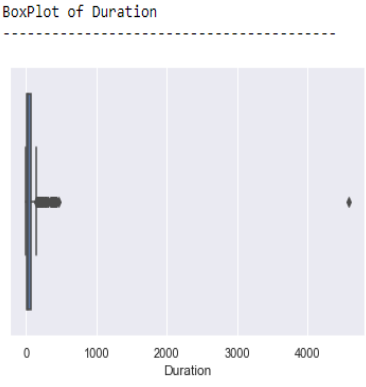
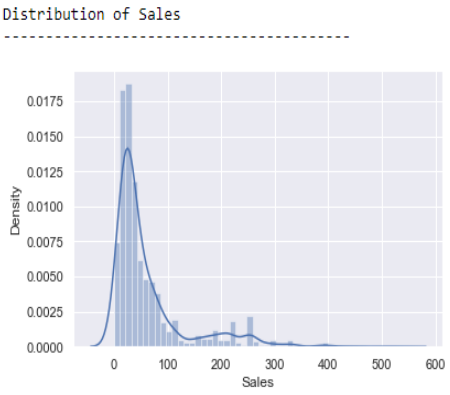


Figure 18

For Sales:

```
Description of Sales
-----------------------------------------
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

Distribution of Sales
-----------------------------------------
```



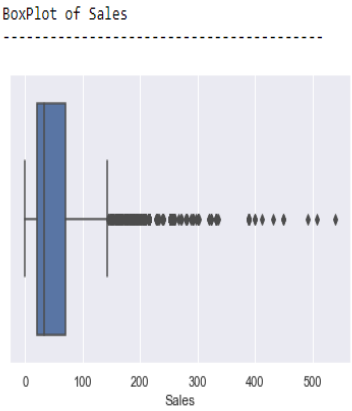


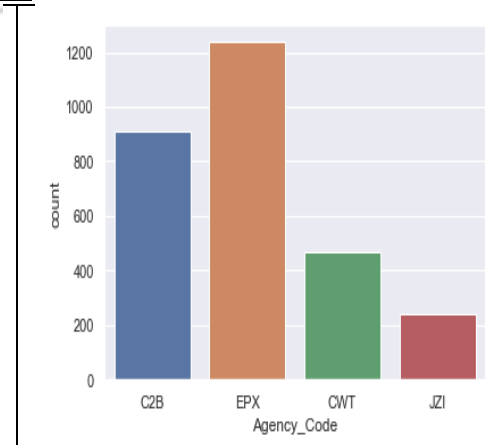
Figure 19

- ✓ From the above visualization, we can see that all the variables has outliers.
- ✓ All the variables are positively skewed.
- ✓ Through, distplot we can see the pattern of data distribution.
- ✓ By checking the Boxplots for all the continuous variables we can conclude that a very high number of outliers are present in all the continuous variables .Hence, it is better to remove the present outliers which lie in our dataset to further build the models for the problem.

CATEGORICAL VARIABLES:

For Agency_code variable:

```
EPX    1238
C2B     913
CWT     471
JZI     239
Name: Agency_Code, dtype: int64
```



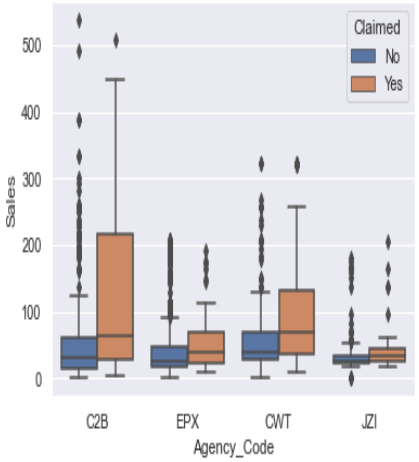


Figure 20

- From the above,we can see that Agency EPX has higher frequency
- The box plot shows the split of sales with different agency code and also hue having claimed column.
- It seems that C2B have claimed more claims than other agency.

For Channel:

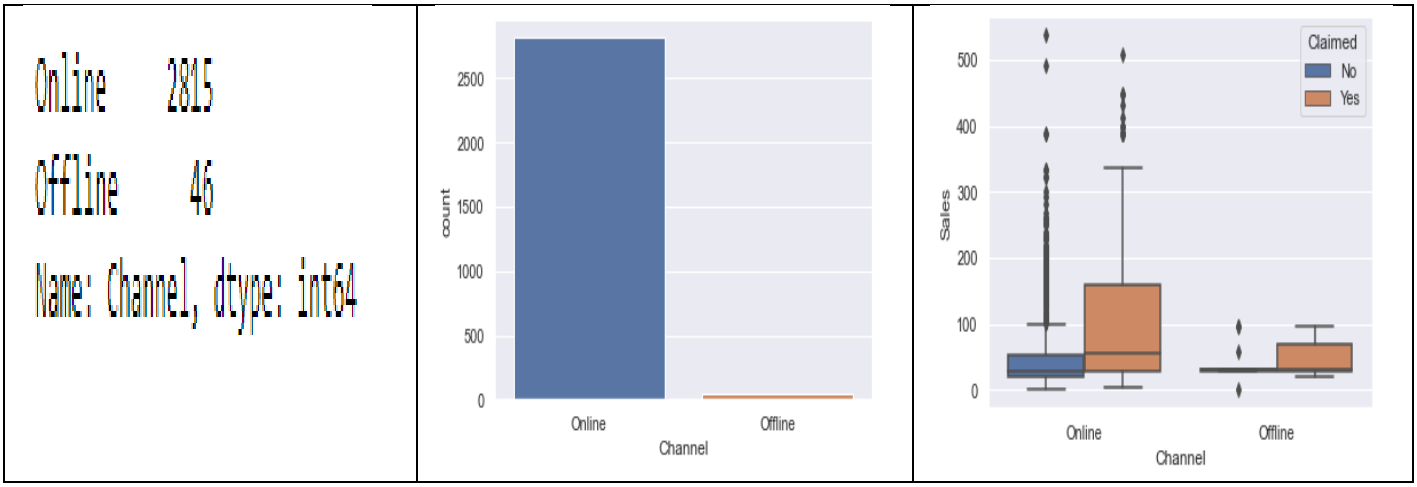


Figure 21

- Channel Online is very much used than offline
- The box plot shows the split of sales with different type and also hue having claimed column. We could understand airlines type has more claims.

For Product Names:

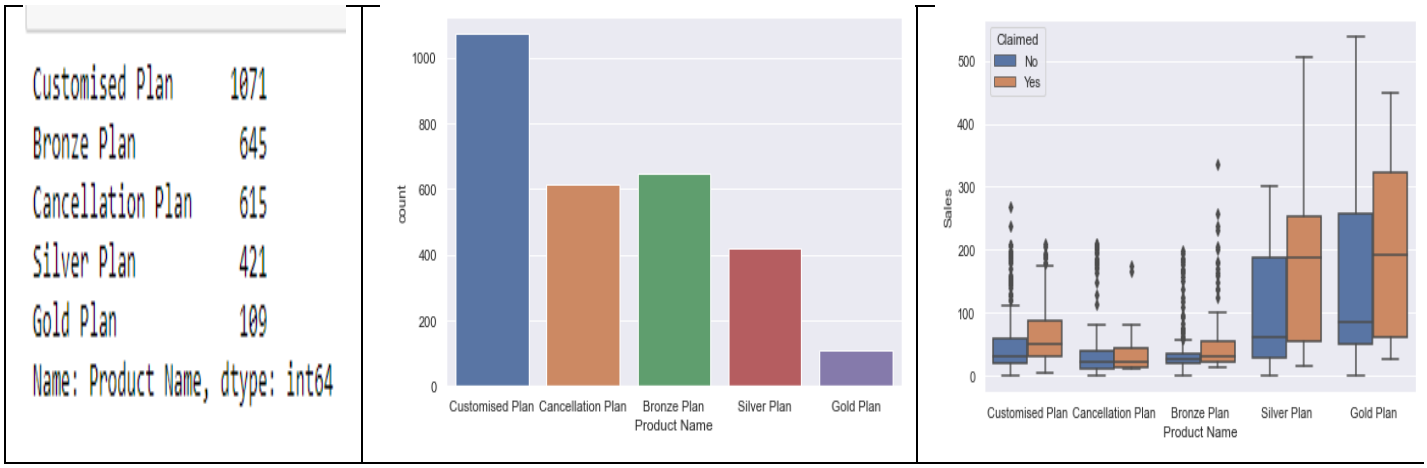


Figure 22

- Customized Plans are more sought by Customers than the rest plans.
- The box plot shows the split of sales with different destination and also hue having claimed column.

For Destination:

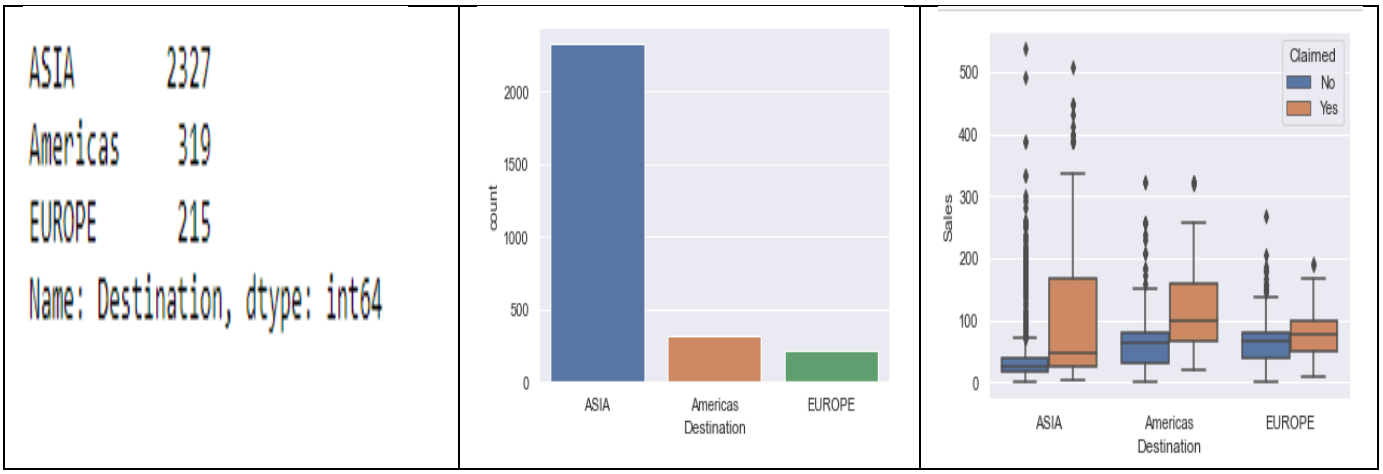


Figure 23

- Most Sale & Most Claimed Destination is ASIA.
- The box plot shows the split of sales with different destination and also hue having claimed column.

FURTHER TO OUR INFERENCE:

We can see that maximum 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 30-40 contribute to the highest number of claims, the type of Tour Agency was Travel Agency, Product name was Customized Plan, Channel was Online and Destination was Asia.

BI-VARIATE & MULTI-VARIATE ANALYSIS:

PAIR-PLOT: Populate the pair plot with all the numeric variables to see the pairwise distribution.

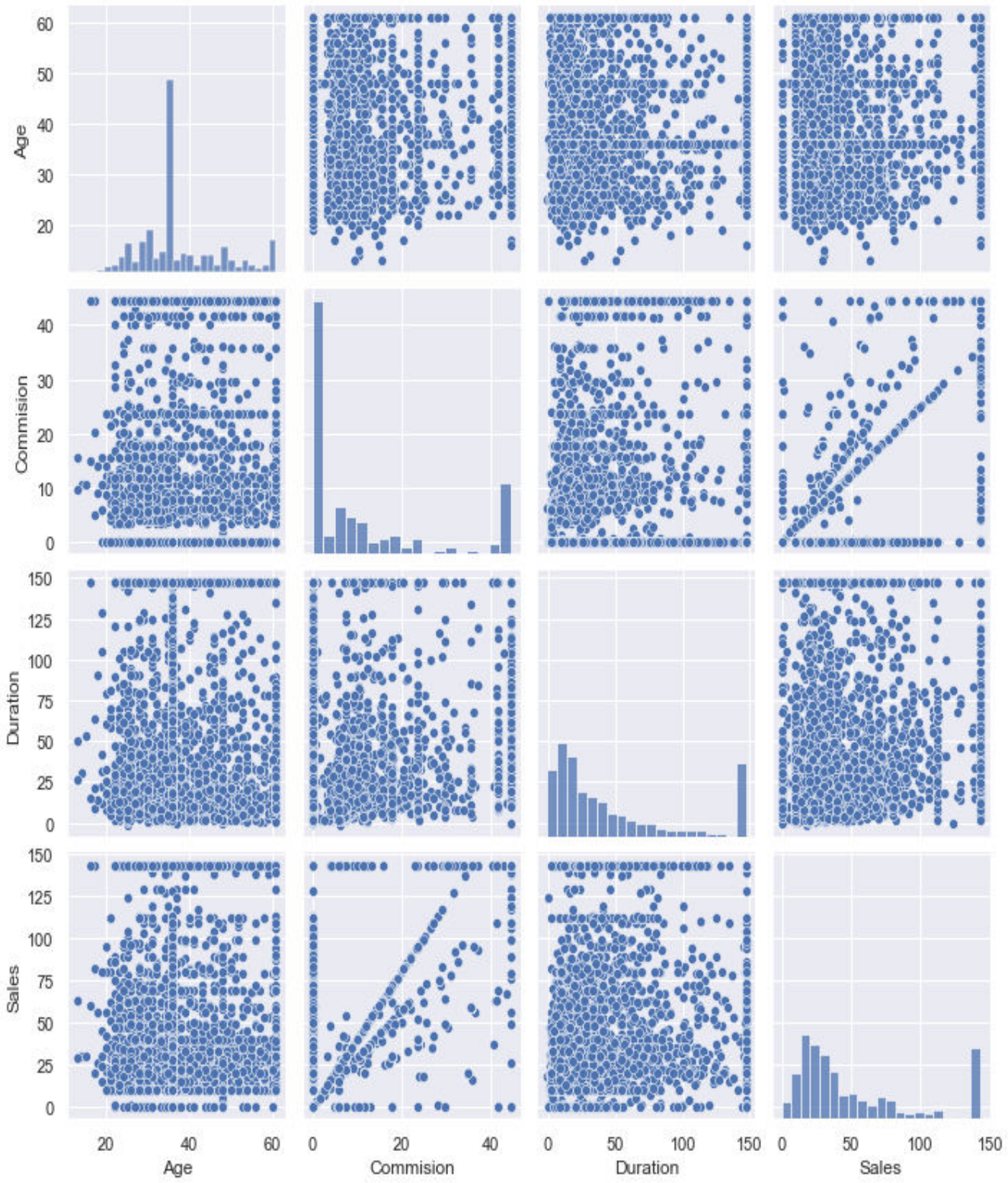


Figure 24

HEAT MAP (RELATIONSHIP ANALYSIS)

We will now plot a Heat Map or Correlation Matrix to evaluate the relationship between different variables in our dataset. This graph can help us to check for any correlations between different variables.

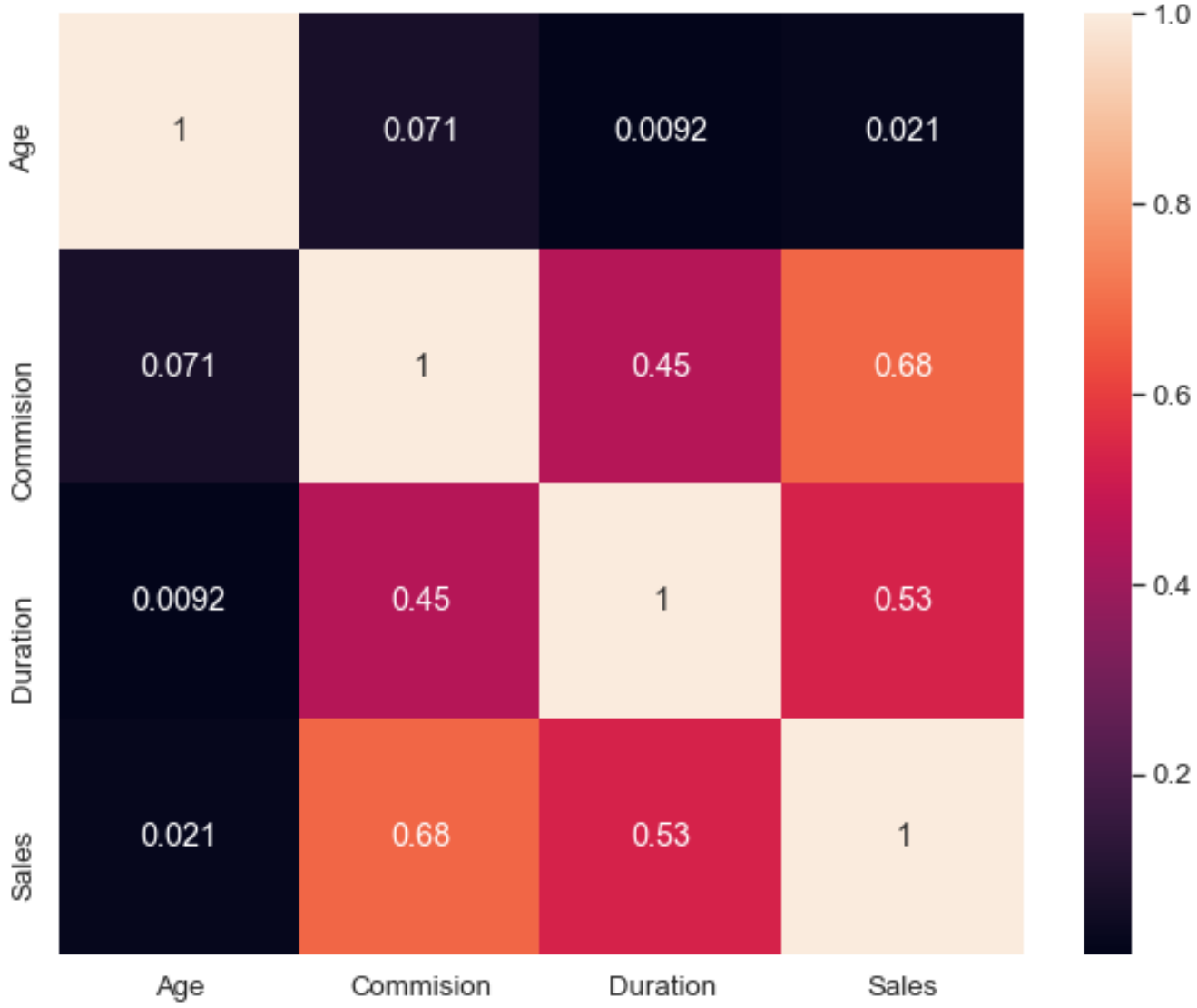


Figure 25

INFERENCE:

- We can see there is no negative correlation
- Only positive correlation is observed.
- Not much multi-collinearity was observed.

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

By checking the Boxplots for all the continuous variables we can conclude that a very high number of outliers are present in all the continuous variables .

Hence, it is better to remove the present outliers which lies in our dataset to further build the models for the problem.

Checking for outliers:

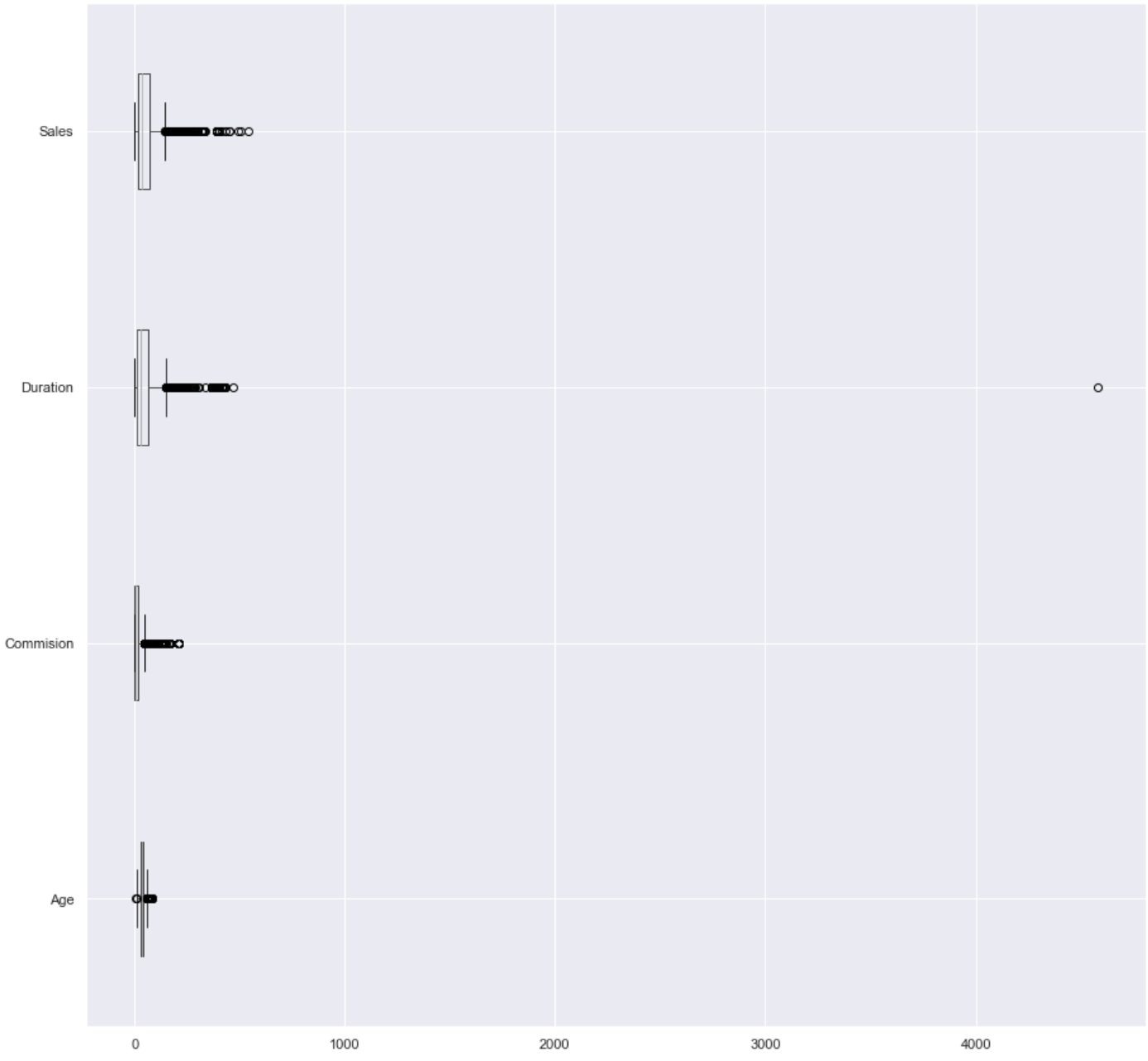


Figure 26

Treating outliers:

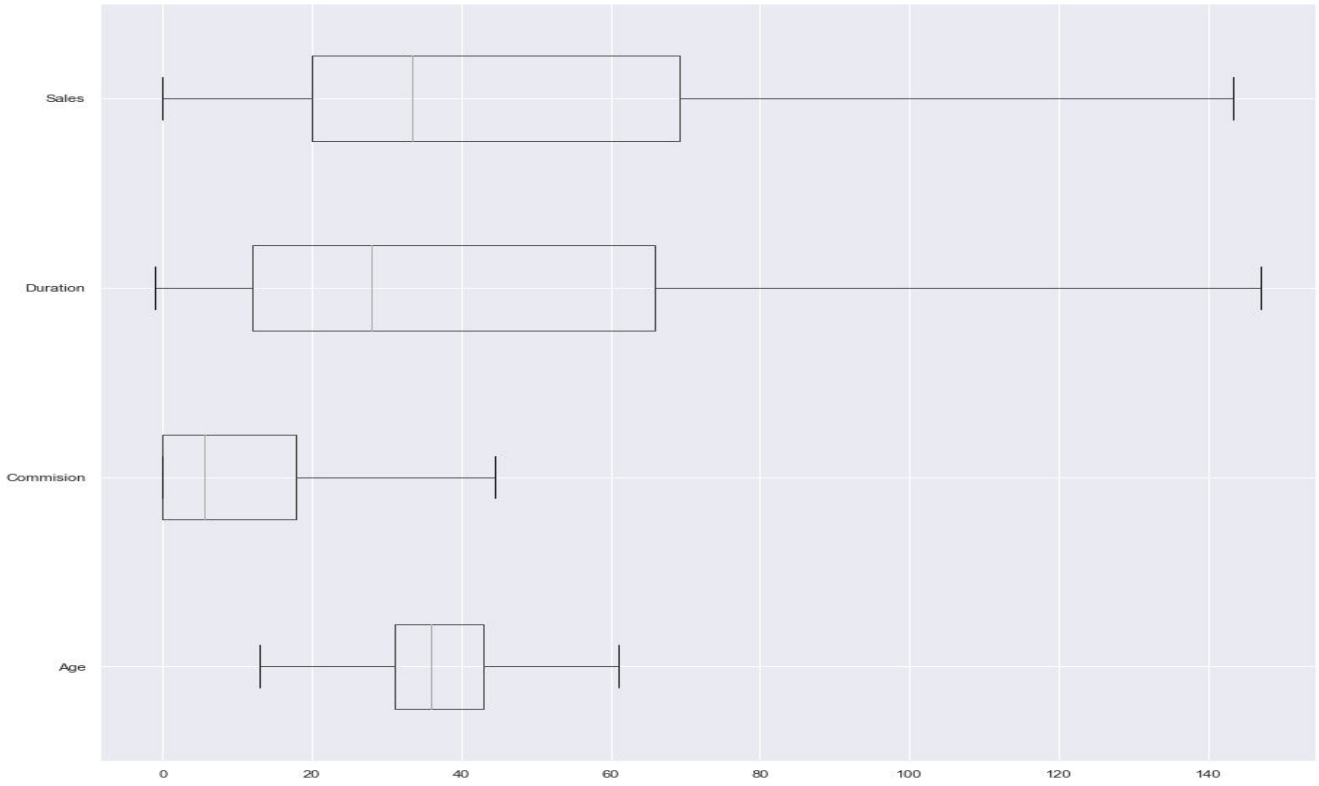


Figure 27

Converting 'object' datatype to 'int'

For our analysis and building Decision tree and Random Forest, we have to convert the variables which have 'object' datatype and convert them into integer.

```
for feature in ins.columns:
    if ins[feature].dtype == 'object':
        print('\n')
        print('feature:',feature)
        print(pd.Categorical(ins[feature].unique()))
        print(pd.Categorical(ins[feature].unique()).codes)
        ins[feature] = pd.Categorical(ins[feature]).codes
```

```
ins.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2861 entries, 0 to 2999
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Age            2861 non-null   int64
 1   Agency_Code    2861 non-null   int8
 2   Type           2861 non-null   int8
 3   Claimed        2861 non-null   int8
 4   Commision      2861 non-null   float64
 5   Channel        2861 non-null   int8
 6   Duration       2861 non-null   int64
 7   Sales          2861 non-null   float64
 8   Product Name   2861 non-null   int8
 9   Destination    2861 non-null   int8
dtypes: float64(2), int64(2), int8(6)
memory usage: 193.1 KB
```

- Now, all the objects variables have been converted into Categorical ones.

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

```
0    0.680531
1    0.319469
Name: Claimed, dtype: float64
```

Extracting the target column into separate vectors for training set and test set and performing standardization to scale the data.- Claimed is our Target Variable.

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

```
X = ins.drop("Claimed", axis=1)
y = ins.pop("Claimed")
X.head()
```

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

Prior to scaling

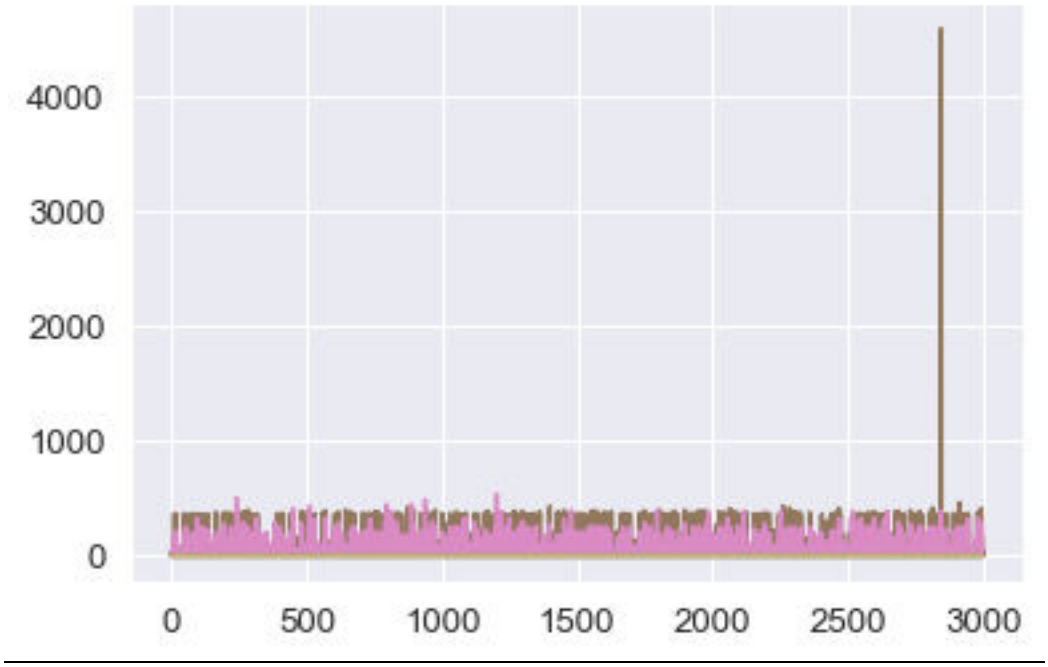


Figure 28

Used zscore in order to standardise the data as follow:

```
# Scaling the attributes.
from scipy.stats import zscore
X_scaled=X.apply(zscore)
X_scaled.head()
```

	Age	Agency_Code	Type	Commision	Channel	Duration	Sales	Product Name	Destination
0	0.917540	-1.276081	-1.217993	-0.556921	0.127832	-0.478989	-0.829950	0.260997	-0.446648
1	-0.206451	0.716750	0.821023	-0.584029	0.127832	-0.280392	-0.584949	0.260997	-0.446648
2	0.074546	-0.279665	0.821023	-0.353996	0.127832	-0.508411	-0.726430	0.260997	1.259439
3	-0.206451	0.716750	0.821023	-0.584029	0.127832	-0.501056	-0.500900	-0.521721	-0.446648
4	-0.487449	1.713166	-1.217993	-0.340055	0.127832	-0.140638	-0.612965	-1.304440	-0.446648

After scaling the data

Figure 29

Splitting Dataset in Train and Test Data (70:30)

For building the models we will now have to split the dataset into Training and Testing Data with the ratio of 70:30 as we want 30% of the data is to be tested 70% to be trained.

Random state we have given as 0,as it will give similar results with similar random state respectively.

These two datasets are stored in X_train and X_test with their corresponding dimensions as follows: the samples are almost equally distributed between the train and test datasets:

```
from sklearn.model_selection import train_test_split
X_train, X_test, train_labels, test_labels = train_test_split(X_scaled, y, test_size=.30,random_state=0)
```

```
#Checking the dimensions of the training and test data
```

```
print('X_train',X_train.shape)
print('X_test',X_test.shape)
print('train_labels',train_labels.shape)
print('test_labels',test_labels.shape)
```

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

Model 1 - Building Decision Tree- CART Model:

Classification and Regression Trees(CART) are a type of Decision trees used in Data mining. It is a type of Supervised Learning Technique where the predicted outcome is either a discrete or class (classification) of the dataset or the outcome is of continuous or numerical in nature(regression).

Using the Train Dataset (X_train) we will be creating a CART model and then further testing the model on Test Dataset (X_test).

For creating the CART Model two packages were imported namely, “DecisionTreeClassifier” and “tree” from sklearn.

Classification and Regression Trees (CART) are a type of Decision trees used in Data mining. It is a type of Supervised Learning Technique where the predicted outcome is either a discrete or class (classification) of the dataset or the outcome is of continuous or numerical in nature (regression).
Using the Train Dataset (X_train) we will be creating a CART model and then further testing the model on Test Dataset (X_test).

For creating the CART Model two packages were imported namely, “DecisionTreeClassifier” and “tree” from sklearn.

Model 1 - Building Decision Tree- CART Model

```
param_grid_dtcl = {
    'criterion': ['gini'],
    'max_depth': [10,20,30,50],
    'min_samples_leaf': [50,100,150],
    'min_samples_split': [150,300,450],
}
dtcl = DecisionTreeClassifier()
grid_search = GridSearchCV(estimator = dtcl, param_grid = param_grid_dtcl, cv = 10)
```

```
grid_search.fit(X_train, train_labels)
print(grid_search.best_params_)
best_grid = grid_search.best_estimator_
best_grid
```

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

DecisionTreeClassifier(max_depth=10, min_samples_leaf=50, min_samples_split=300)
```

```
#Generating decision Tree
```

```
train_char_label = ['no', 'yes']
tree_regularized = open('tree_regularized.dot','w')
dot_data = tree.export_graphviz(best_grid, out_file= tree_regularized , feature_names =list(X_train), class_names = list(train_ch
tree_regularized.close()
dot_data
```

With the help of Decision Tree Classifier we will create a decision tree model and using the “gini” criteria we will fit the train data into this model.

Grid search is essentially an optimization algorithm which enables us to classify best parameters for the optimization of the problem from a list of parameter. Fitting our train dataset to the grid search. After fitting the values, we will get the best parameters.

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

DecisionTreeClassifier(max_depth=10, min_samples_leaf=50, min_samples_split=300)
```

After this using the tree package we will create a dot file namely, tree_regularised.dot to help visualize the tree.

By adding the best grid parameters & saving it on to an output dot file & by pasting the code on http://webgraphviz.com/ to view the tree chart as follows: which will aid in carrying out the test & train data analysis.

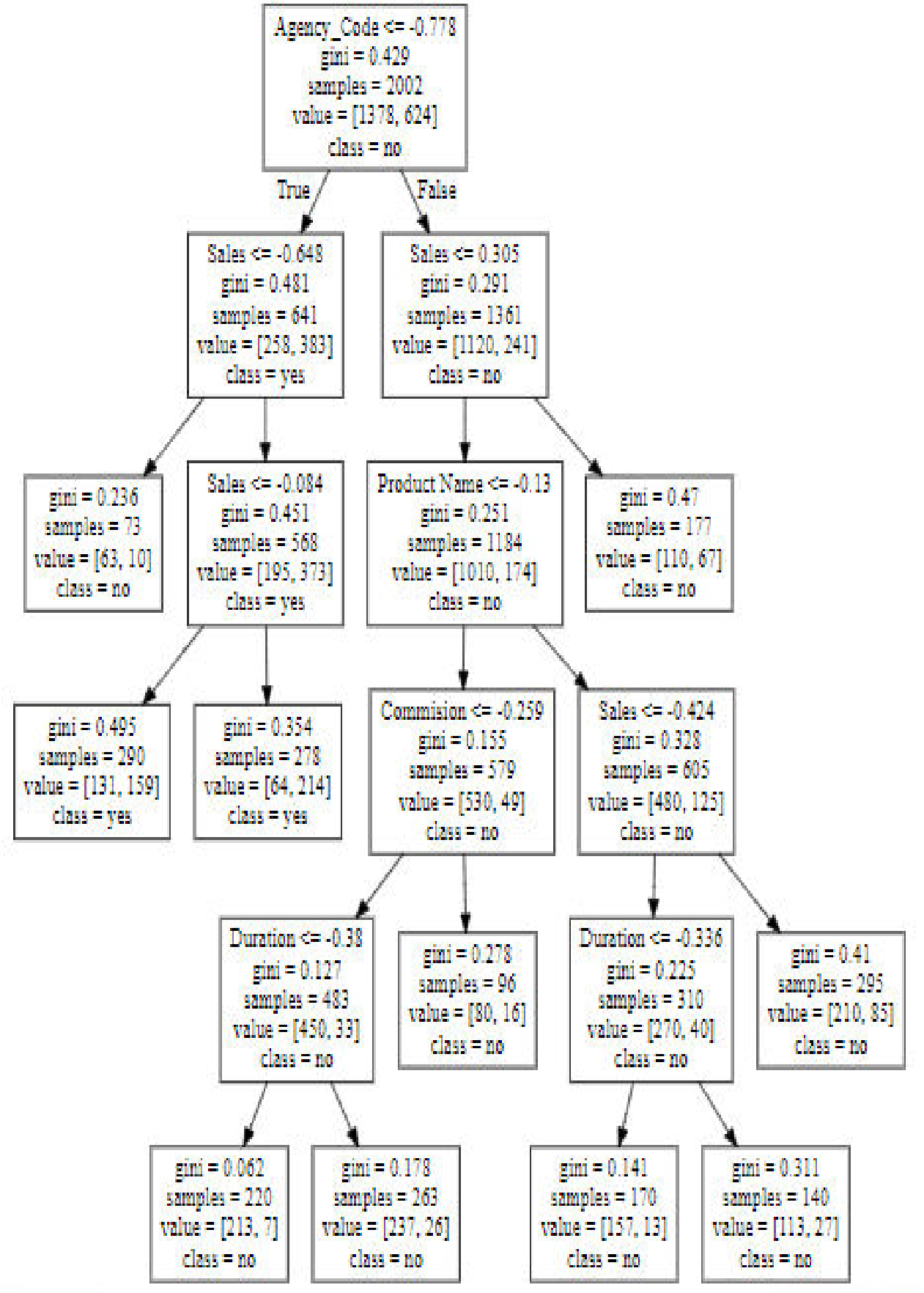

Agency_Code <= -0.778
gini = 0.429
samples = 2002
value = [1378, 624]
class = no
True
False
Sales <= -0.648
gini = 0.481
samples = 641
value = [258, 383]
class = yes
Sales <= 0.305
gini = 0.291
samples = 1361
value = [1120, 241]
class = no
gini = 0.236
samples = 73
value = [63, 10]
class = no
Sales <= -0.084
gini = 0.451
samples = 568
value = [195, 373]
class = yes
Product Name <= -0.13
gini = 0.251
samples = 1184
value = [1010, 174]
class = no
gini = 0.47
samples = 177
value = [110, 67]
class = no
gini = 0.495
samples = 290
value = [131, 159]
class = yes
gini = 0.354
samples = 278
value = [64, 214]
class = yes
Commision <= -0.259
gini = 0.155
samples = 579
value = [530, 49]
class = no
Sales <= -0.424
gini = 0.328
samples = 605
value = [480, 125]
class = no
Duration <= -0.38
gini = 0.127
samples = 483
value = [450, 33]
class = no
gini = 0.278
samples = 96
value = [80, 16]
class = no
Duration <= -0.336
gini = 0.225
samples = 310
value = [270, 40]
class = no
gini = 0.41
samples = 295
value = [210, 85]
class = no
gini = 0.062
samples = 220
value = [213, 7]
class = no
gini = 0.178
samples = 263
value = [237, 26]
class = no
gini = 0.141
samples = 170
value = [157, 13]
class = no
gini = 0.311
samples = 140
value = [113, 27]
class = no

Figure 30

Below are the variable importance values or the feature importance to build the tree used to built a tree:

	Imp
Agency_Code	0.640280
Sales	0.303567
Product Name	0.036594
Duration	0.013120
Commision	0.006438
Age	0.000000
Type	0.000000
Channel	0.000000
Destination	0.000000

Model 2 – Random Forest:

Random Forest is another Supervised Learning Technique used in Machine Learning which consists of many decision trees that helps in predictions using individual trees and selects the best output from them.

The random forest classifier can use for both classification and the regression task. Random forest classifier will handle the missing values.

When we have more trees in the forest, random forest classifier won't over fit the model. We can model the random forest classifier for categorical values also.

Using the Train Dataset (X_train) we will be creating a Random Forest model and then further testing the model on Test Dataset(X_test)

For creating the Random Forest, the package "RandomForestClassifier" is imported from sklearn.metrics.

Using the GridSearchCV package from sklearn.model_selection we will identify the best parameters to build a Random Forest namely, rfcl. Hence, doing a few iterations with the values we got the best parameters to build the RF Model,which is further used for model performance evaluation as follows:

Model 2 - Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
rfcl=RandomForestClassifier(n_estimators=300,
                            oob_score=True,
                            max_depth=10,
                            max_features=6,
                            min_samples_leaf=20,
                            min_samples_split=60)
```

```
rfcl.fit(X_train,train_labels)
```

```
RandomForestClassifier(max_depth=10, max_features=6, min_samples_leaf=20,
                       min_samples_split=60, n_estimators=300, oob_score=True)
```

```
rfcl.oob_score_
```

```
0.7837162837162838
```

```
param_grid = {
    'max_depth': [10,20],
    'max_features': [6,8],
    'min_samples_leaf': [10,12],
    'min_samples_split': [50,60],
    'n_estimators': [300,400]
}
rfcl = RandomForestClassifier()
grid_search = GridSearchCV(estimator = rfcl, param_grid = param_grid, cv = 5)
```

```
grid_search.fit(X_train, train_labels)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [10, 20], 'max_features': [6, 8],
                         'min_samples_leaf': [10, 12],
                         'min_samples_split': [50, 60],
                         'n_estimators': [300, 400]})
```

```
grid_search.best_params_
```

```
{'max_depth': 10,
 'max_features': 6,
 'min_samples_leaf': 10,
 'min_samples_split': 60,
 'n_estimators': 300}
```

```
best_grid = grid_search.best_estimator_
```

```
best_grid
```

```
RandomForestClassifier(max_depth=10, max_features=6, min_samples_leaf=10,
                       min_samples_split=60, n_estimators=300)
```

Variable Importance & train and test data with best parameters prediction results for Random Forest Classifier are displayed below:

	Imp
Agency_Code	0.369105
Sales	0.202530
Product Name	0.166400
Duration	0.090648
Commision	0.089476
Age	0.051607
Type	0.018686
Destination	0.008715
Channel	0.002832

Model 3 – Artificial Neural Network (ANN):

Artificial Neural Network (ANN) is a computational model that consists of several processing elements that receive inputs and deliver outputs based on their predefined activation functions.

Using the train dataset (X_train) and test dataset(X_test) we will be creating a Neural Network using MLPClassifier from sklearn.metrics.

Firstly, we will have to Scale the two datasets using Standard Scaler package.

Model 3 - Neural Network Classifier

```
from sklearn.neural_network import MLPClassifier
```

```
#Scaling the data using standardScaler
```

```
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
```

```
X_train=sc.fit_transform(X_train)
```

```
X_train
```

```
array([[-0.20493937,  0.7144367 ,  0.82092207, ..., -0.73904567,
        -0.52656514, -0.44760203],
       [-0.20493937,  0.7144367 ,  0.82092207, ..., -0.6528162 ,
        -0.52656514, -0.44760203],
       [-0.58360906,  0.7144367 ,  0.82092207, ..., -0.59532988,
         0.25896646, -0.44760203],
       ...,
       [-1.05694618, -1.27762775, -1.21814243, ...,  0.18648401,
         1.83002967, -0.44760203],
       [-0.20493937,  0.7144367 ,  0.82092207, ..., -0.72467409,
        -0.52656514, -0.44760203],
       [-0.20493937,  0.7144367 ,  0.82092207, ...,  0.36756591,
```

```
X_test=sc.transform(X_test)
```

```
X_test
```

```
array([[ 0.55240002, -1.27762775, -1.21814243, ...,  1.96281117,
        -1.31209675, -0.44760203],
       [ 0.74173487, -0.28159552,  0.82092207, ..., -0.74048283,
         0.25896646,  1.25600874],
       [ 0.17373033, -1.27762775, -1.21814243, ...,  0.28852223,
         1.83002967, -0.44760203],
       ...,
       [-0.20493937,  0.7144367 ,  0.82092207, ..., -0.29352672,
         0.25896646, -0.44760203],
       [-0.20493937,  0.7144367 ,  0.82092207, ..., -0.5809583 ,
         0.25896646, -0.44760203],
       [-0.96227875, -1.27762775, -1.21814243, ..., -0.61688725,
         1.83002967, -0.44760203]])
```

Using the GridSearchCV package from sklearn.model_selection we will identify the best parameters to build an Artificial Neural Network Model namely, nncl. Hence, doing a few

iterations with the values we got the best parameters to build the ANN Model which are as follows:

```
param_grid={
    'hidden_layer_sizes': [520,100,500],
    'max_iter':[2500,3000],
    'solver': ['adam'],
    'tol': [0.01],
}
nncl=MLPClassifier(random_state=1)
grid_search=GridSearchCV(estimator=nncl,param_grid=param_grid,cv=10)
```

```
grid_search.fit(X_train,train_labels)
grid_search.best_params_
```

```
{'hidden_layer_sizes': 100, 'max_iter': 2500, 'solver': 'adam', 'tol': 0.01}
```

```
best_grid=grid_search.best_estimator_
best_grid
```

```
MLPClassifier(hidden_layer_sizes=100, max_iter=2500, random_state=1, tol=0.01)
```

Using these best parameters evaluated using GridSeachCV an Artificial Neural Network Model is created which is further used for model performance evaluation.

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

To check the Model Performances of the three models created above certain model evaluators are used i.e., Classification Report, Confusion Matrix, ROC_AUC Score and ROC Plot. They are calculated first for train data and then for test data.

CART Model Performance Metrics:

1) Classification Report:

```
print(classification_report(train_labels, ytrain_predict))
```

	precision	recall	f1-score	support
0	0.82	0.86	0.84	1378
1	0.66	0.60	0.63	624
accuracy			0.78	2002
macro avg	0.74	0.73	0.73	2002
weighted avg	0.77	0.78	0.77	2002

```
print(classification_report(test_labels, ytest_predict))
```

	precision	recall	f1-score	support
0	0.79	0.85	0.82	569
1	0.66	0.56	0.60	290
accuracy			0.75	859
macro avg	0.72	0.70	0.71	859
weighted avg	0.75	0.75	0.75	859

2) Confusion Matrix:

```
#Confusion Matrix for the training data
```

```
confusion_matrix(train_labels, ytrain_predict)
```

```
array([[1183,  195],
       [ 251,  373]], dtype=int64)
```

```
#Confusion Matrix for test data
```

```
confusion_matrix(test_labels, ytest_predict)
```

```
array([[484,  85],
       [128, 162]], dtype=int64)
```

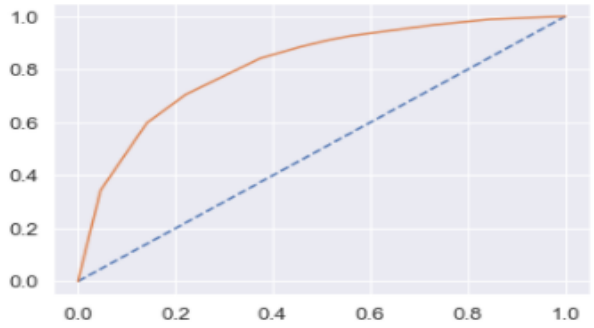
3) ROC_AUC Score and ROC Curve:

```
#AUC and ROC for the training data
```

```
# predict probabilities
probs = best_grid.predict_proba(X_train)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
cart_train_auc = roc_auc_score(train_labels, probs)
print('AUC: %.3f' % cart_train_auc)
# calculate roc curve
cart_train_fpr, cart_train_tpr, cart_train_thresholds = roc_curve(train_labels, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(cart_train_fpr, cart_train_tpr)
```

```
AUC: 0.815
```

```
[<matplotlib.lines.Line2D at 0x1cda6896fa0>]
```




```
#AUC and ROC for the test data
```

```
# predict probabilities
probs = best_grid.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
cart_test_auc = roc_auc_score(test_labels, probs)
print('AUC: %.3f' % cart_test_auc)
# calculate roc curve
cart_test_fpr, cart_test_tpr, cart_testthresholds = roc_curve(test_labels, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(cart_test_fpr, cart_test_tpr)
```

```
AUC: 0.778
```

```
[<matplotlib.lines.Line2D at 0x1cda49ad9d0>]
```

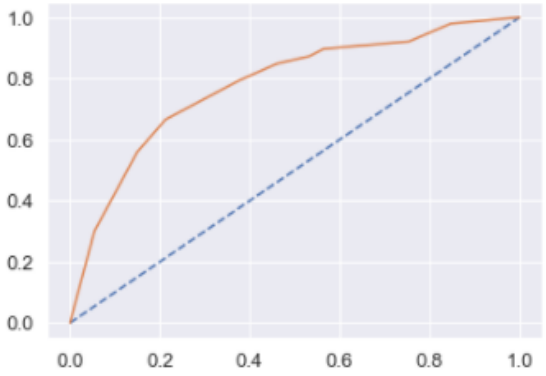


Figure 31

4) Accuracy:

```
#Train Data Accuracy

cart_train_acc=best_grid.score(X_train,train_labels)
cart_train_acc
```

```
0.7772227772227772
```

```
#Test Data Accuracy
cart_test_acc=best_grid.score(X_test,test_labels)
cart_test_acc
```

```
0.7520372526193247
```

Train Data Accuracy: 78%; Test Data Accuracy: 75%
Train Data Precision: 66 %; Test Data Precision: 66%
Train Data f1-score: 63%; Test Data f1-score: 60%
Train Data AUC: 82%; Test Data AUC: 78%

From observing the characteristics of the CART training & testing dataset, we can observe that the results are close & almost similar. Overall, the model is a good model.

RANDOM FOREST Model Performance Metrics:

1) Classification Report:

```
print(classification_report(train_labels,ytrain_predict))
```

	precision	recall	f1-score	support
0	0.84	0.90	0.87	1378
1	0.73	0.61	0.67	624
accuracy			0.81	2002
macro avg	0.78	0.76	0.77	2002
weighted avg	0.80	0.81	0.80	2002

```
print(classification_report(test_labels, ytest_predict))
```

	precision	recall	f1-score	support
0	0.78	0.88	0.82	569
1	0.67	0.51	0.58	290
accuracy			0.75	859
macro avg	0.73	0.69	0.70	859
weighted avg	0.74	0.75	0.74	859

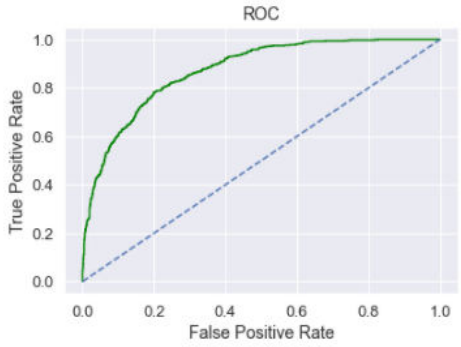
2) Confusion Matrix:

```
#Confusion Matrix for the training data

confusion_matrix(train_labels,ytrain_predict)

array([[1238,  140],
       [ 241,  383]], dtype=int64)
```

```
#Confusion Matrix for test data

confusion_matrix(test_labels,ytest_predict)

array([[487,  82],
       [125, 165]], dtype=int64)
```

3) ROC_AUC Score and ROC Curve:

```
#AUC and ROC for the training data

rf_train_fpr, rf_train_tpr,_=roc_curve(train_labels,best_grid.predict_proba(X_train)[:,1])
plt.plot(rf_train_fpr,rf_train_tpr,color='green')
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
rf_train_auc=roc_auc_score(train_labels,best_grid.predict(X_train))
print('Area under Curve is', rf_train_auc)

Area under Curve is 0.7560927672955975
```




```
#AUC and ROC for the test data

rf_test_fpr,rf_test_tpr,_=roc_curve(test_labels,best_grid.predict_proba(X_test)[:,1])
plt.plot(rf_test_fpr,rf_test_tpr,color='green')
plt.plot([0,1], [0,1],linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
rf_test_auc=roc_auc_score(test_labels,best_grid.predict_proba(X_test)[:,1])
print('Area under Curve is',rf_test_auc)

Area under Curve is 0.7908157081389007
```

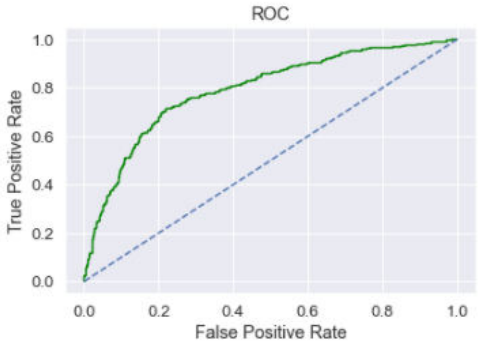


Figure 32

4) Accuracy:

```
#Train Data Accuracy
rf_train_acc=best_grid.score(X_train,train_labels)
rf_train_acc
```

```
0.8096903096903096
```

```
#Test Data Accuracy
rf_test_acc=best_grid.score(X_test,test_labels)
rf_test_acc
```

```
0.7590221187427241
```

Train Data Accuracy: 81%; Test Data Accuracy: 75%
Train Data Precision: 73 %; Test Data Precision: 67%
Train Data f1-score: 67%; Test Data f1-score: 58%
Train Data AUC: 76%; Test Data AUC: 79%

From observing the characteristics of the RF training & testing data set, RF model has better accuracy, precision, recall & f1 score than the CART model.

ARTIFICIAL NEURAL NETWORK Model Performance Metrics:

1) Classification Report:

```
print(classification_report(train_labels,ytrain_predict))
```

	precision	recall	f1-score	support
0	0.80	0.88	0.84	1378
1	0.66	0.51	0.58	624
accuracy			0.77	2002
macro avg	0.73	0.70	0.71	2002
weighted avg	0.76	0.77	0.76	2002

```
print(classification_report(test_labels,ytest_predict))
```

	precision	recall	f1-score	support
0	0.78	0.88	0.82	569
1	0.67	0.51	0.58	290
accuracy			0.75	859
macro avg	0.73	0.69	0.70	859
weighted avg	0.74	0.75	0.74	859

2) Confusion Matrix:

```
#Confusion Matrix for the training data
confusion_matrix(train_labels,ytrain_predict)
```

```
array([[1213,  165],
       [ 304,  320]], dtype=int64)
```

```
#Confusion Matrix for test data
```

```
confusion_matrix(test_labels,ytest_predict)
```

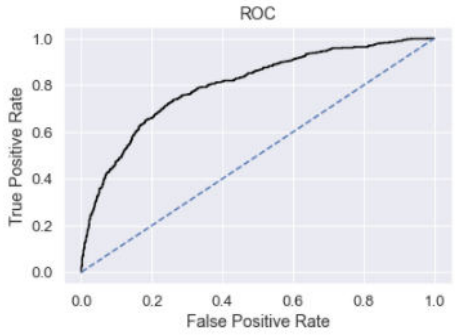
```
array([[498,  71],
       [143, 147]], dtype=int64)
```

3) ROC_AUC Score and ROC Curve:

```
#AUC and ROC for the training data
```

```
nn_train_fpr,nn_train_tpr,_=roc_curve(train_labels,best_grid.predict_proba(X_train)[:,1])
plt.plot(nn_train_fpr,nn_train_tpr,color='black')
plt.plot([0,1], [0,1],linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
nn_train_auc=roc_auc_score(train_labels,best_grid.predict_proba(X_train)[:,1])
print('Area under Curve is',nn_train_auc)
```

```
Area under Curve is 0.7992137201816083
```




```
#AUC and ROC for the test data
```

```
nn_test_fpr,nn_test_tpr,_=roc_curve(test_labels,best_grid.predict_proba(X_test)[:,1])
plt.plot(nn_test_fpr,nn_test_tpr,color='black')
plt.plot([0,1], [0,1],linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
nn_test_auc=roc_auc_score(test_labels,best_grid.predict_proba(X_test)[:,1])
print('Area under Curve is',nn_test_auc)
```

```
Area under Curve is 0.7845221501727168
```

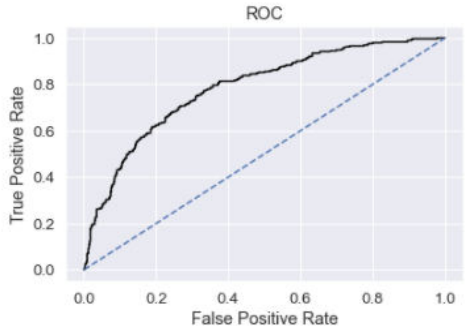


Figure 33

4) Accuracy:

```
#Train Data Accuracy
nn_train_acc=best_grid.score(X_train,train_labels)
nn_train_acc
```

```
0.7657342657342657
```

```
#Confusion Matrix for test data
```

```
confusion_matrix(test_labels,ytest_predict)
```

```
array([[498,  71],
       [143, 147]], dtype=int64)
```

Train Data Accuracy: 77%; Test Data Accuracy: 75%
Train Data Precision: 66 %; Test Data Precision: 58%
Train Data f1-score: 57%; Test Data f1-score: 58%
Train Data AUC: 80%; Test Data AUC: 78%

From observing the characteristics of the ANN training & testing data set, RF model has better accuracy, precision, recall & f1 score than the model. So we can finalize on RF Model.

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

Comparison of all the performance evaluators for the three models is given in the following table.
We are using Precision, F1 Score and AUC Score for our evaluation.

	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.78	0.75	0.81	0.76	0.77	0.75
AUC	0.82	0.78	0.76	0.79	0.80	0.78
Recall	0.60	0.56	0.67	0.61	0.51	0.51
Precision	0.66	0.66	0.61	0.57	0.66	0.67
F1 Score	0.63	0.60	0.73	0.67	0.58	0.58

ROC Curve for all the 3 models (Training data):

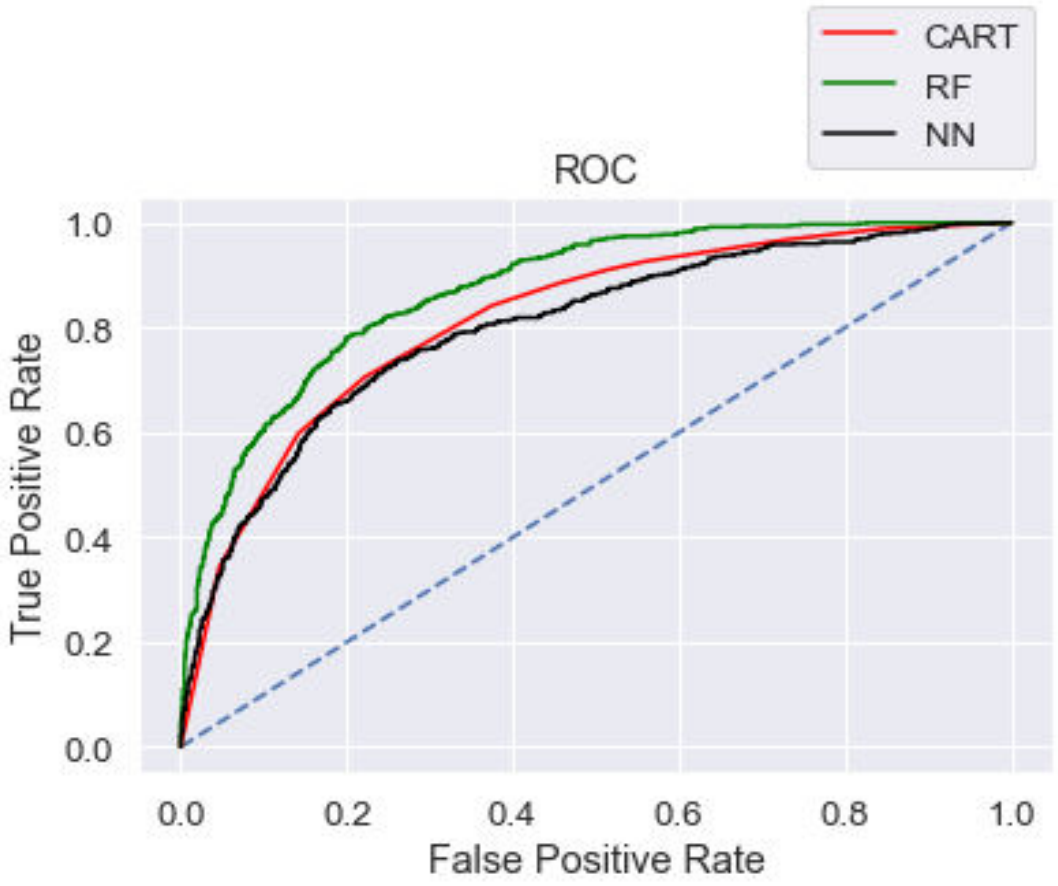


Figure 34

ROC Curve for all the 3 models (Testing data):

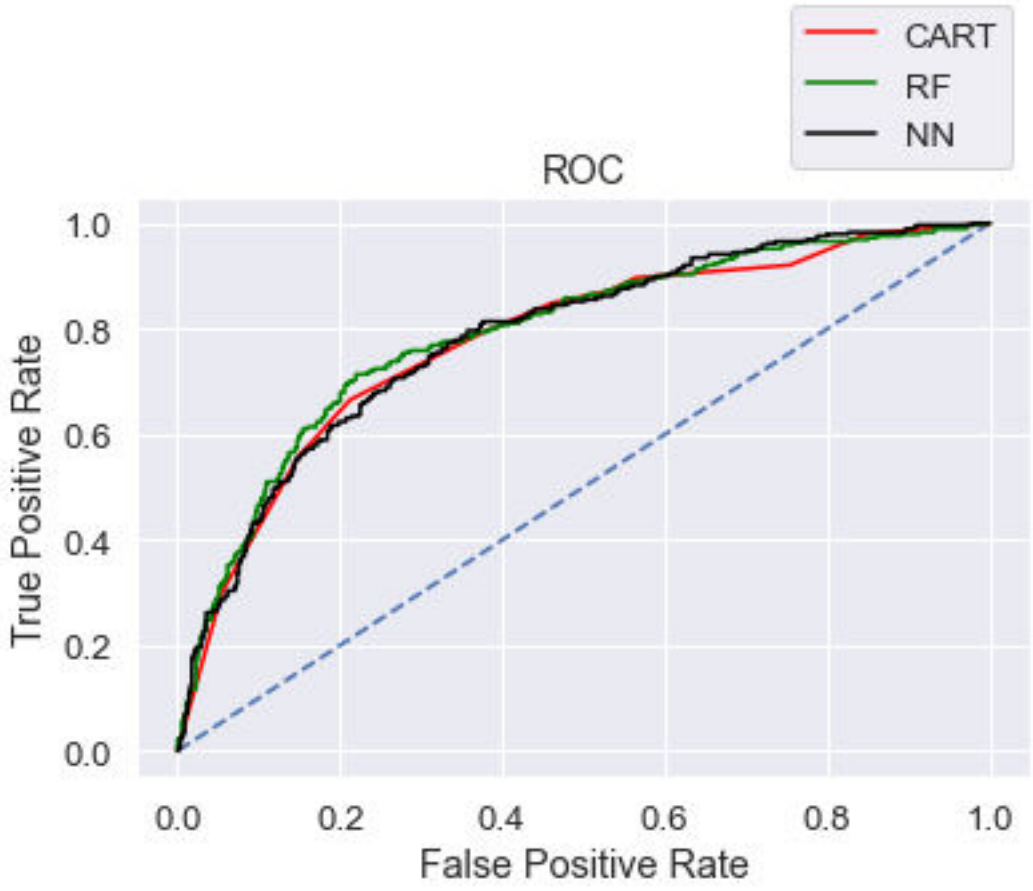


Figure 35

INFERENCE:

From observing the characteristics of the CART, RF, ANN training & testing data set, RF model has better accuracy, precision, recall & f1 score than the other two models.

The Random Forest method has the best performance that means best accuracy compared to all the three models. The percentage deviation between Training and Testing Dataset also is reasonably under control, classifying a good model. The Percentage deviation between Training & Testing Dataset is very minimal among the three models.

Choosing Random Forest Model is the best option in this case as it will exhibit very less variance as compared to a single decision tree or a multi – layered Neural Network.

2.5 Inference: Based on the whole Analysis, what are the business insights and recommendations?

INFERENCES:

- For the business problem of an Insurance firm providing Tour Insurance, we have attempted to make a few Data Models for predictions of probabilities. The models that are attempted are namely, CART or Classification and Regression Trees, Random Forest and Artificial Neural Network(MLP). The three models are then evaluated on training and testing datasets and their model performance scores are calculated.

- The Accuracy, Precision, F1 Score are computed using Classification Report. The confusion matrix, AUC_ROC Scores and ROC plot are computed for each model separately and compared. All the three models have performed well but to increase our accuracy in determining the claims made by the customers we can choose the Random Forest Model. Instead of creating a single Decision Tree it can create a multiple decision trees and hence can provide the best claim status from the data.

- As seen from the above model performance measures, for all the models i.e. CART, Random Forest and ANN have performed exceptionally well. Hence, we can choose either of the models but choosing Random Forest Model is a great option as even though they exhibit the same accuracy but choosing Random Forest over Cart model is way better as they have much less variance than a single decision tree.

BUSINESS INSIGHTS AND RECOMMENDATION:

1) The Basic objective for building the predictive model was to see & also classify if an insurance firm providing insurance is facing low/medium/high claim frequency.

2) The data had Outliers. After, performing & running the 3 models we inferred that the data is well balanced for conducting the models but, more data will assist & help understand to predict the models better. Past Data could also help in understanding & structuring the data to dive into deeper business problems. Overall all the models are stable enough for making any future predictions because there is no over fitting.

3) Claims are higher for ASIA destination; the management could take actions & follow stipulated protocols before structuring a policy in ASIA. This, could be done by increasing rates via Premium on the policy for recovering the Cost which has been claimed

4) The Management should make more Customers opt for an insurance policy more affordable & increase complexion of the policy in such a way that more customers would choose the insurance policy for the competitive rate. This would attract more customers & also certainly reduce fraud claims by the set of complexions

5) As we could observe from the data maximum of the insurance claim has been done by online channel than offline channel, reason being more convenient & good experience which is a good factor generating profits. But, subsequently, management should also promote offline channels

additional promotions, offers, low cost than online to pull customers which could generate more customer base.

6) We could observe that most of the Sales were generated from Type Sales than Airlines & also we could observe that Insurance Claim is more at Airlines than Agencies. This could be because of more customer satisfaction & customer trust at Airlines than at Agencies. This could be put into better use by suggestive selling insurance plans at Airlines. Management could deploy better customer service at Agencies to match the likeliness of Airlines which could give better Sales Performance. Low performing Agencies such as JZI, CWT could be given more promotional drives for better sales conversion rates.

7) Customized Plans have been chosen more compared to other plans. Targets & incentives could be given to promote Silver Plans & Gold plans which would generate more sales via Promo offers, Ad Campaigns, Marketing drives etc.

8) Management should focus more on Customer Satisfaction, Customer Claim Turn around Times, methods to actively reduce false transactions via strong Artificial Intelligence Methods. Focusing on making the process efficient, affordable & reducing overall operational costs

THE END