Data Mining

Pradeep Kumar Mishra

PGP-DSBA Online

Jun_B_21

Date: 24:Oct:2021

Content View

Problem Statement - 1 3

1.1 Read the data, do the necessary initial steps, and exploratory data analysis (Univariate, Bi-variate, and multivariate analysis). 3

Exploratory Data Analysis 3

Sample of the dataset 3

Let us check the types of variables and missing values in the dataset 3

1.2 Do you think scaling is necessary for clustering in this case? Justify. 9

1.3 Apply hierarchical clustering to scaled data. Identify the number of optimum clusters using Dendrogram and briefly describe them 11

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. 15

1.5 Describe cluster profiles for the clusters defined. Recommend different promotional strategies for different clusters. 17

Problem 2: CART-RF-ANN 18

2.1 Read the data, do the necessary initial steps, and exploratory data analysis (Univariate, Bi-variate, and multivariate analysis). 18

Exploratory Data Analysis 18

Sample of the datasets 18

Let us check the types of variables and missing values in the dataset 18

Check for duplicate data 19

Univariate Analysis : 19

2.2 Data Split: Split the data into test and train, build classification model CART, Random Forest, Artificial Neural Network. 30

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. 36

2.4 Final Model: Compare all the models and write an inference which model is best/optimized. 52

2.5 Inference: Based on the whole Analysis, what are the business insights and recommendations. 53

List of Figures :

Fig.1 Distplot......6
Fig.2 Boxplot......7
Fig.3 Heatmap......8
Fig.4 pairplot......9
Fig.5 plot......10
Fig.6 plot......11
Fig.7 Dendrogram 12
Fig.8 Dendrogram 12
Fig.9 elbow curve......14
Fig10. Distplot......19
Fig. 11 Boxplot......20
Fig.12 countplot......21
Fig.13 Boxplot......24
Fig.14 Heatmap 27
Fig.15 Pairplot 28
Fig. 16 roc curve 37
Fig. 17 roc curve 39
Fig. 18 roc curve 42
Fig. 19 roc curve 44
Fig. 20 roc curve 47
Fig.21 roc curve 49
Fig.22 roc curve 50
Fig.23 roc curve 51

List of Tables :

Table-1 Dataset Sample...... 4
Table-2 Describe the data......5
Table-3 Describe Data 10
Table-4 Describe Data 11
Table-5 Dataset 13
Table-6 cluster 13
Table-7 cluster profiles......13
Table-8 silhouette score 15
Table-9 cluster profiling 15

Table-10 cluster profiling……………………………………………………………………………16
Table -11 Dataset……………………………………………………………………………………..17
Tabel 12.Describe data……………………………………………………………………………...18
Tabel 13 Describe data……………………………………………………………………………...18
Table 14 Matrix……………………………………………………………………………………...50

Problem Statement - 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.

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

Exploratory Data Analysis

Sample of the dataset

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

Table-1 Dataset Sample

Let us check the types of variables and missing values in the dataset

- From the below results we can see that there is no missing value present in the dataset.
- There are a total of 210 rows and 7 columns in the dataset.
- All variables are float64 Data types.

```
<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
```

Univariate analysis:

Helps us understand the distribution of data in the datasets. With univariate analysis we can find patterns and we can summarize the data.

Checking the Summary Statistic :

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000
mean	14.847524	14.559286	0.870999	5.628533	3.258605	3.700201	5.408071
std	2.909699	1.305959	0.023629	0.443063	0.377714	1.503557	0.491480
min	10.590000	12.410000	0.808100	4.899000	2.630000	0.765100	4.519000
25%	12.270000	13.450000	0.856900	5.262250	2.944000	2.561500	5.045000
50%	14.355000	14.320000	0.873450	5.523500	3.237000	3.599000	5.223000
75%	17.305000	15.715000	0.887775	5.979750	3.561750	4.768750	5.877000
max	21.180000	17.250000	0.918300	6.675000	4.033000	8.456000	6.550000

Table-2 Describe the data

- We see that for most of the variables, mean/median are nearly equal.
- Std Deviation is high for the spending variable.

Distplot :

By observing the below figure there is skewness in both sides (left side and right side) and data is not normally distributed.

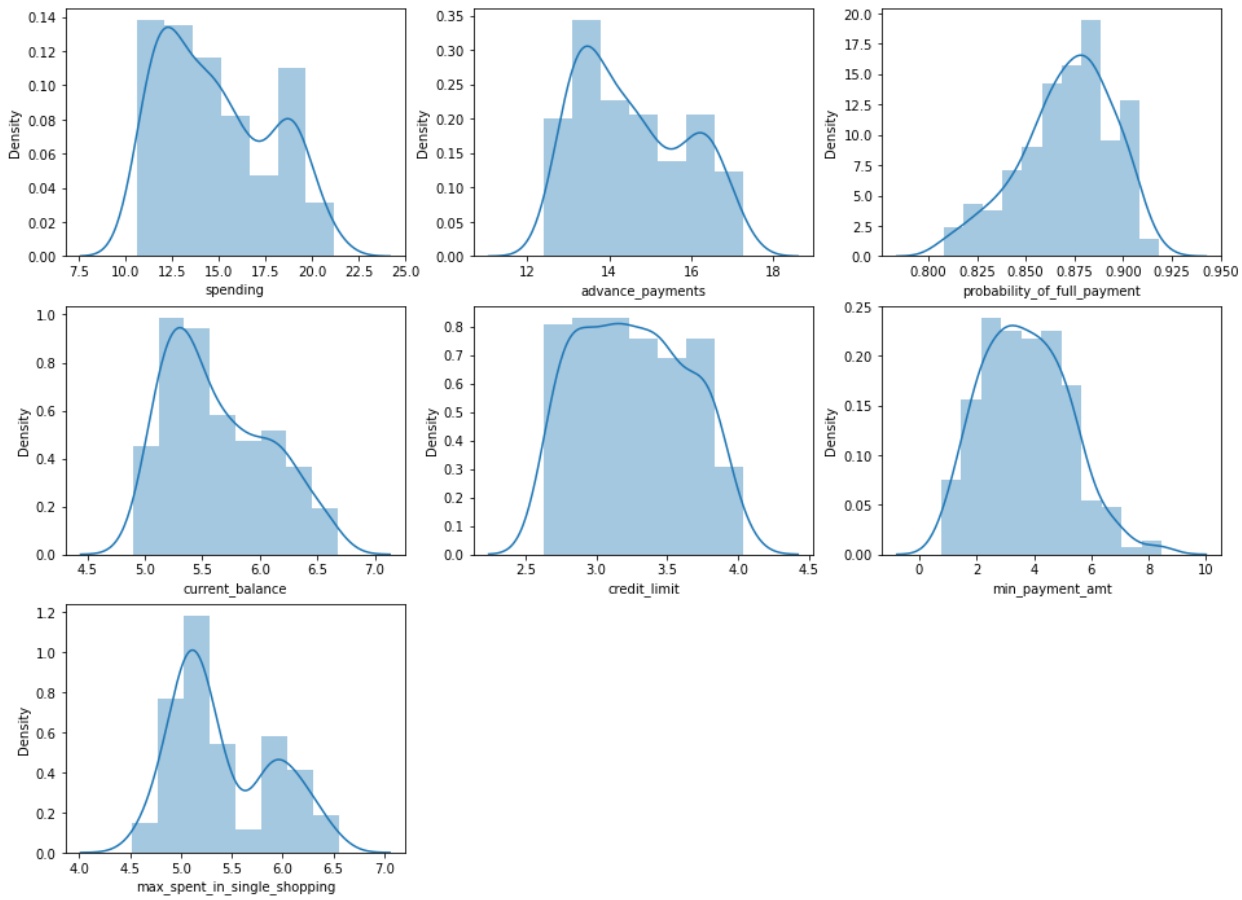


Figure-1 Distplot

Boxplot :

By seeing the below boxplot there are some feature have outliers eg. probability_of_full_payment and min_payment_amt but in the clustering algorithm there is no impact of outliers, so we don't impute the outliers.

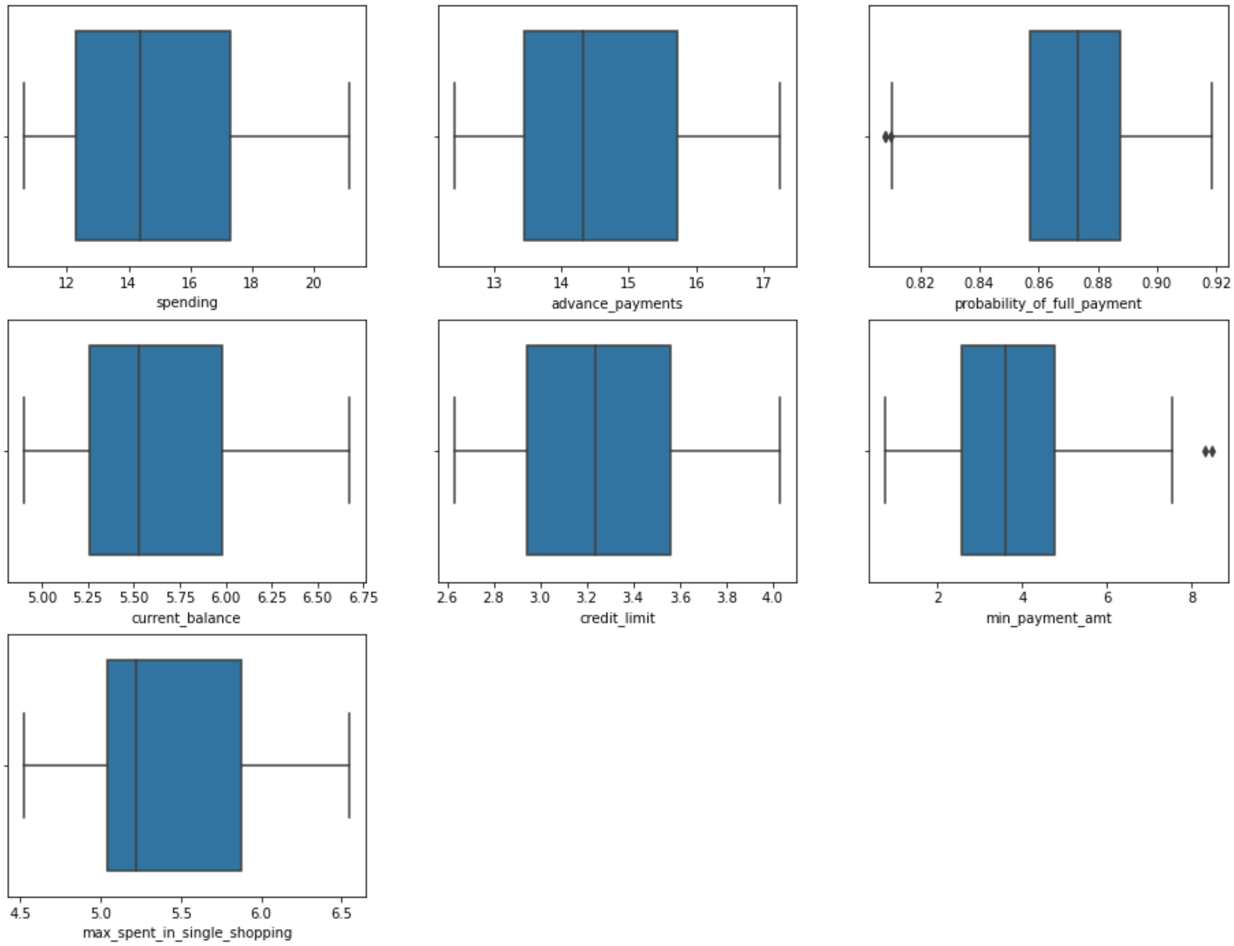


Figure-2 Boxplot

Spending feature :

- The boxplot of the spending variable shows no outliers.
- Spending is positively skewed : 0.3999

Advance_payments feature :

- The boxplot of the spending variable shows no outliers.
- Advance_payments is positively skewed : 0.3866

probability_of_full_payment Skewed feature :

- The boxplot of the probability_of_full_payment shows some outliers.
- probability _of_full_payment is negatively skewed : -0.538

current_balance feature:

- The boxplot of the current_balance shows no outliers.
- Current_balance is positively skewed : 0.5255

Credit_limit feature :

- The boxplot of the credit_limit shows no outliers.
- Credit_limit is positively skewed: 0.1344

Min_payment_amt :

- The boxplot of min_payment_amt shows some outliers.
- Min_payment_amt is positively skewed : 0.4017

Max_spent_in_single_shopping :

- The boxplot of max_spent_in_single_shoping shows no outliers.
- Max_spent_in_single_shoping is positively skewed : 0.5619

Heatmap :

As we see in the below heatmap there is strong correlation between the features but there is no impact of multicollinearity in the clustering algorithm.

Between advance_payments and spending features have high correlation.

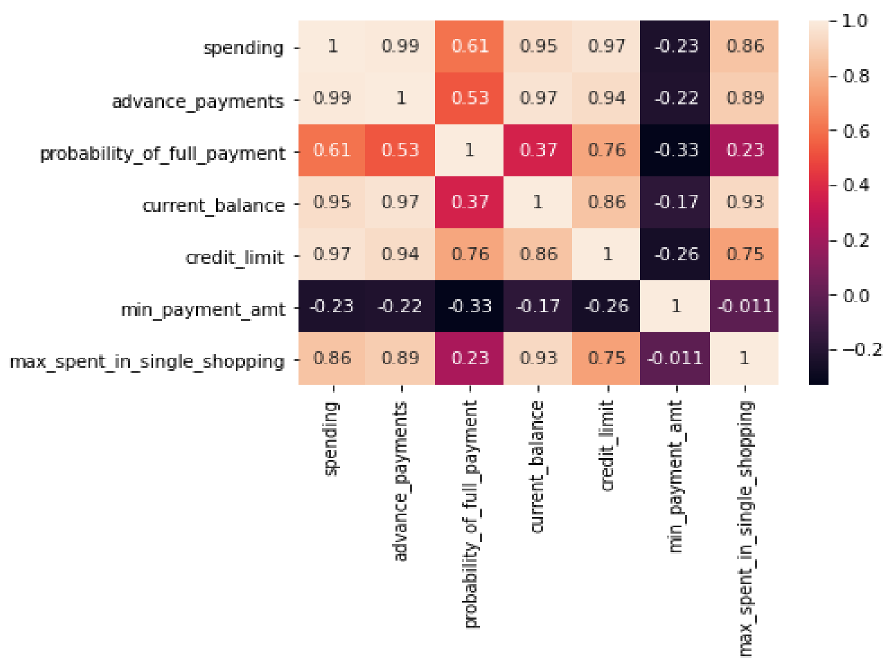


Figure-3 Heatmap

Pairplot :

After seeing the below pairplot figure we can say that there is a strong relationship among each feature.

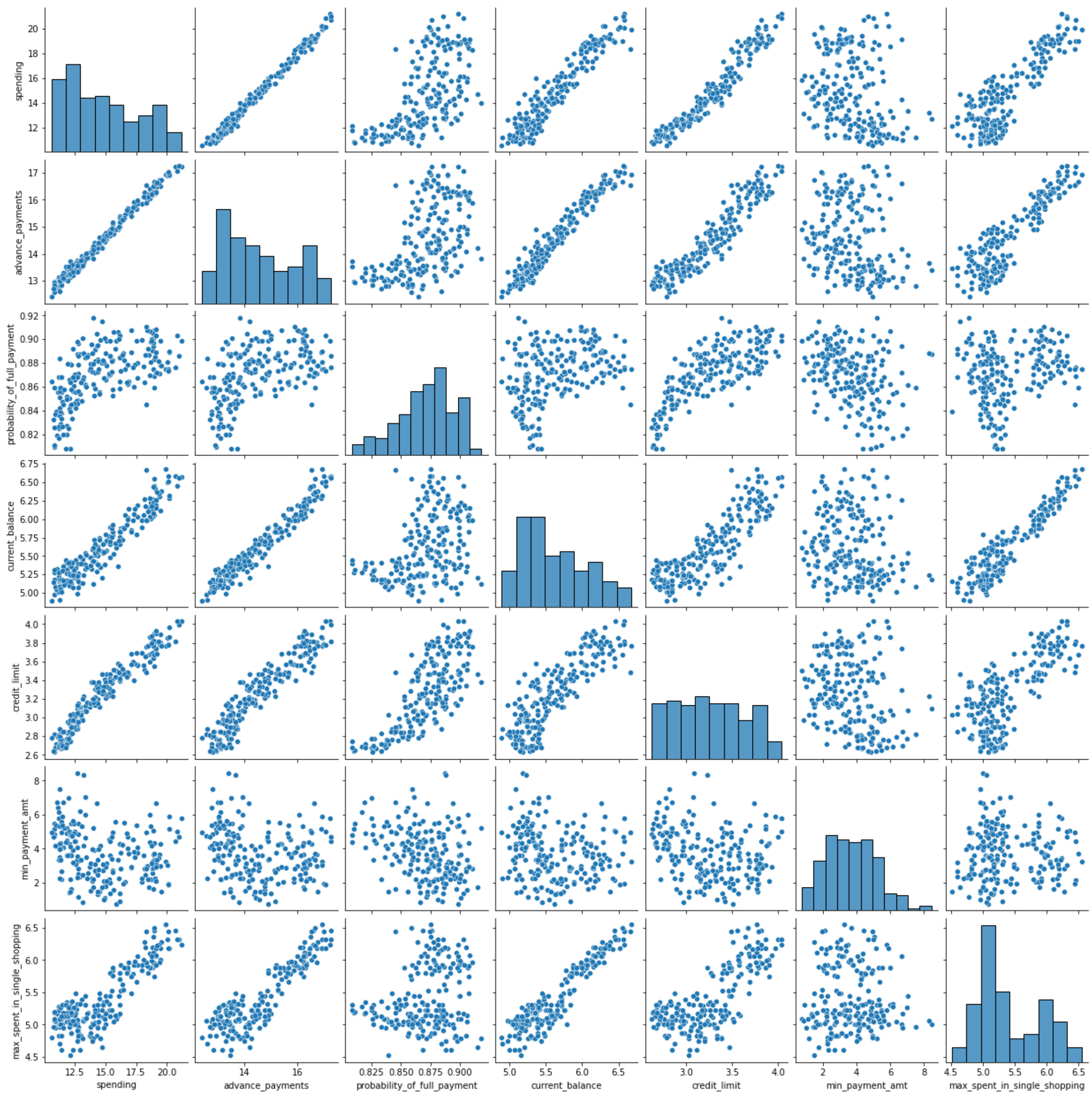


figure -4 pairplot

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

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000
mean	14.847524	14.559286	0.870999	5.628533	3.258605	3.700201	5.408071
std	2.909699	1.305959	0.023629	0.443063	0.377714	1.503557	0.491480
min	10.590000	12.410000	0.808100	4.899000	2.630000	0.765100	4.519000
25%	12.270000	13.450000	0.856900	5.262250	2.944000	2.561500	5.045000
50%	14.355000	14.320000	0.873450	5.523500	3.237000	3.599000	5.223000
75%	17.305000	15.715000	0.887775	5.979750	3.561750	4.768750	5.877000
max	21.180000	17.250000	0.918300	6.675000	4.033000	8.456000	6.550000

Table-3 Describe Data

- Scaling is necessary for clustering in this case because I can see in the above described data and say that data is with different weights. It is recommended to transform the features so that all features are in the same scale.
- For any distance based algorithm we need to scale the data.
- Using Z-score for scaling and can see the below table that seems the same scale.
- Also have shown below the plot of the data prior and after scaling.
- I have used z score to standardise the data to the relative same scale -3 to +3.

prior to scaling:

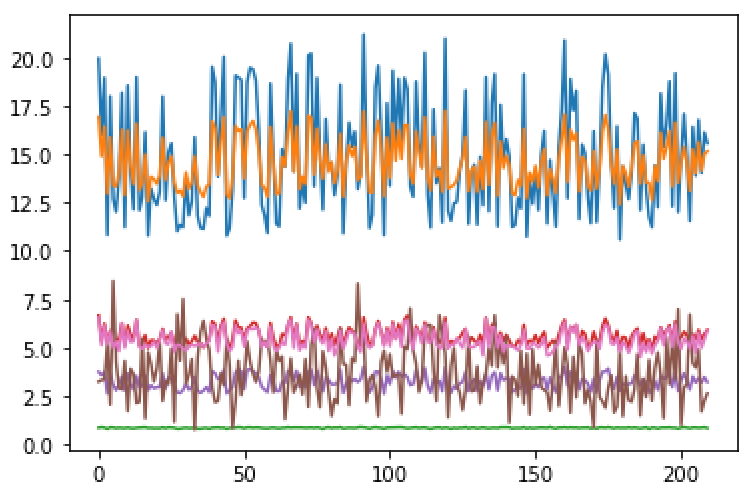


Figure-5 plot

After to scaling:

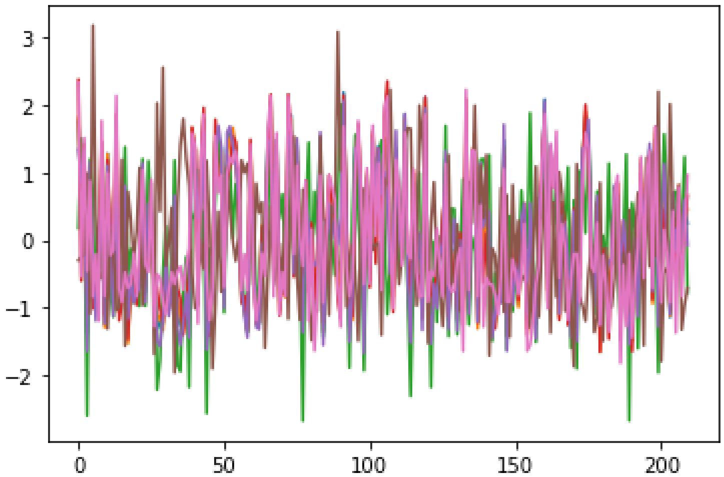


Figure-6 plot

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
count	2.100000e+02	2.100000e+02	2.100000e+02	2.100000e+02	2.100000e+02	2.100000e+02	2.100000e+02
mean	9.148766e-16	1.097006e-16	1.243978e-15	-1.089076e-16	-2.994298e-16	5.302637e-16	-1.935489e-15
std	1.002389e+00	1.002389e+00	1.002389e+00	1.002389e+00	1.002389e+00	1.002389e+00	1.002389e+00
min	-1.466714e+00	-1.649686e+00	-2.668236e+00	-1.650501e+00	-1.668209e+00	-1.956769e+00	-1.813288e+00
25%	-8.879552e-01	-8.514330e-01	-5.980791e-01	-8.286816e-01	-8.349072e-01	-7.591477e-01	-7.404953e-01
50%	-1.696741e-01	-1.836639e-01	1.039927e-01	-2.376280e-01	-5.733534e-02	-6.746852e-02	-3.774588e-01
75%	8.465989e-01	8.870693e-01	7.116771e-01	7.945947e-01	8.044956e-01	7.123789e-01	9.563941e-01
max	2.181534e+00	2.065260e+00	2.006586e+00	2.367533e+00	2.055112e+00	3.170590e+00	2.328998e+00

Table-4 Describe Data

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

For creating the dendrogram I use ward's method linkage type and euclidean distance.

Formula use for dendrogram :

wardlink = linkage(data_scaled, method = 'ward',metric='euclidean')

dend = dendrogram(wardlink)

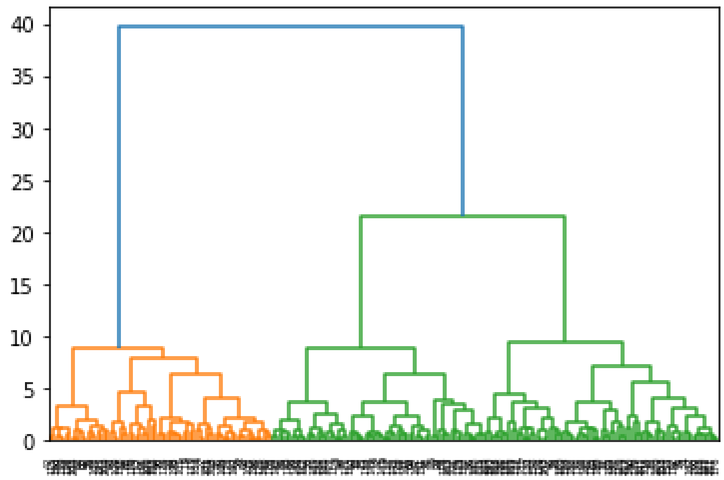


Figure-7 Dendrogram

For better understanding I use below formula

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

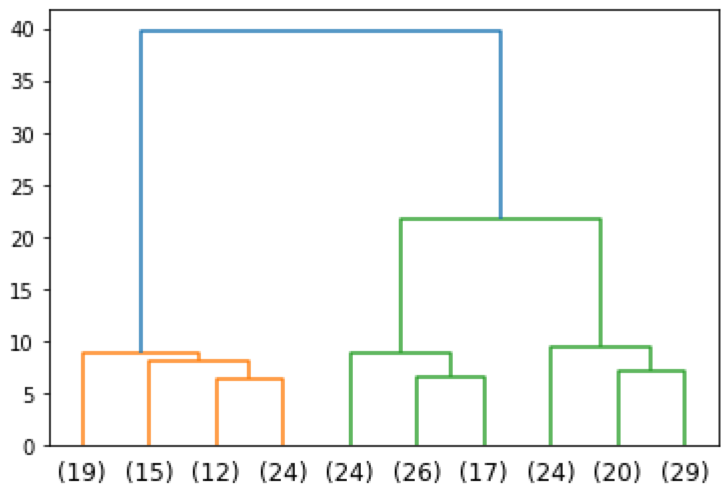


Figure-8 Dendrogram

As seen in the above dendrogram we observe that either we can use 3 clusters or 4 clusters.

But after the cluster profiling we observe that we should go with 3 clusters.

Formula use creating the cluster:

Criterion we can give 'maxclust'

clusters = fcluster(wardlink, 3, criterion='maxclust')

clusters

```
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)
```

After the creating the cluster see the below datasets

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	clusters
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

Table-5 Dataset

Cluster	Total values counts
1	70
2	67
3	73

Table-6 cluster

clusters	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Freq
1	18.371429	16.145429	0.884400	6.158171	3.684629	3.639157	6.017371	70
2	11.872388	13.257015	0.848072	5.238940	2.848537	4.949433	5.122209	67
3	14.199041	14.233562	0.879190	5.478233	3.226452	2.612181	5.086178	73

Table-7 cluster profiles

- By seeing the above table we observe that which users have high credit limit and current balance high they are spending high, their advance payments are high and their probability of full payment is high.
- Which users have less credit limit and current balance they are spending less, their probability of full payment is less and their minimum payment amount is high.

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.

As we use dendrogram in hierarchical clustering to determine the cluster but in the k-means clustering we use elbow curve and silhouette score to determine the cluster for the elbow curve we first use wss (within cluster sum of square).

Wss :

[1469.9999999999995, 659.1717544870411, 430.65897315130064, 371.301721277542, 327.9608240079031, 290.5900305968219, 264.83153087478144, 240.6837259501598, 220.85285825594738, 206.3829103601579]

elbow curve :

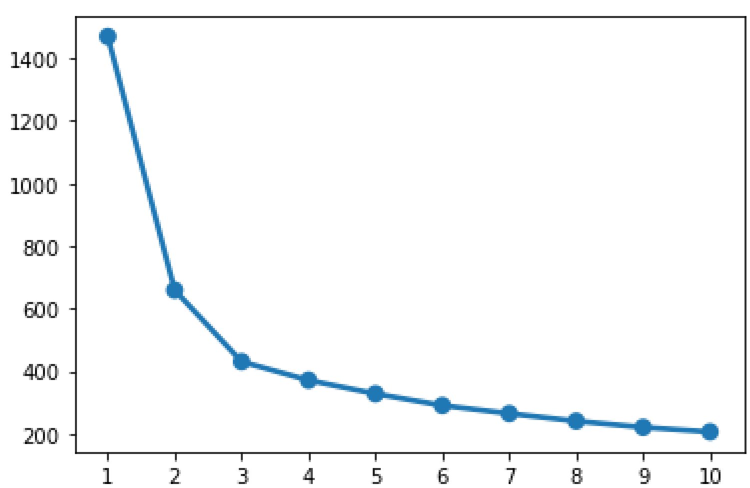


Figure-9 elbow curve

From 1 to 3, there is a significant drop hence 3 is a valuable addition in the k-means algorithm.

Lets evaluate this with differente technique that is called silhouette score :

- If the silhouette score is close to +1 then we can say the clusters are well separated from each other on an average.
- If the silhouette score is close to 0, then we can say the clusters are not separated from each other.
- If the silhouette score is close to -1 then we can say the model has done a blunder in terms of clustering the data.

Formula for silhouette score is :

silhouette_score(data_scaled,labels)

```
For cluster= 2 : 0.46577247686580914
For cluster= 3 : 0.40072705527512986
For cluster= 4 : 0.32757426605518075
For cluster= 5 : 0.2783651415532039 7
For cluster= 6 : 0.28389221057730224
For cluster= 7 : 0.26934344290163237
For cluster= 8 : 0.2578633719728751
For cluster= 9 : 0.2598117260971521 4
```

Table-8 silhouette score

As we see the silhouette score for cluster-3 is high except cluster-2 because we don't use cluster-2.

So we go with cluster-3 as per elbow curve and silhouette score.

Clus_kmeans	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	freq
0	11.856944	13.247778	0.848253	5.231750	2.849542	4.742389	5.101722	72
1	18.495373	16.203433	0.884210	6.175687	3.697537	3.632373	6.041701	67
2	14.437887	14.337746	0.881597	5.514577	3.259225	2.707341	5.120803	71

Table-9 cluster profiling

- By seeing the above table we observe that which users have high credit limit and current balance high they are spending high, their advance payments are high and their probability of full payment is high.

- Which users have less credit limit and current balance they are spending less, their probability of full payment is less and their minimum payment amount is high.

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

Clus_kmeans	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	freq
0	11.856944	13.247778	0.848253	5.231750	2.849542	4.742389	5.101722	72
1	18.495373	16.203433	0.884210	6.175687	3.697537	3.632373	6.041701	67
2	14.437887	14.337746	0.881597	5.514577	3.259225	2.707341	5.120803	71

Table-10 cluster profiling

Some Recommendations:

Cluster-1:

- customers have a high credit limit and current balance so they are spending high as we see in the cluster profiling their high spending. Banks can give them cash back and it may increase spending.
- Increase their credit limit.
- max _spent_in_single_shoping is high so there is more chance they are spending on luxury brands. Give them discounts on luxury brands.
- Cluster-1 groups are more valuable so banks can behave like premium customers and take their feedback.
- Increase spending habits with premium sites, travel portal, travel airlines/hotel, as this will encourage them to spend more.

Cluster-0 :

- Customers have a low credit limit and current balance so they are spending low as we see in the cluster profiling. Bank can give them an instant discount. It will improve their spending amount.
- Offer can be provided on early payments to improve their payments rate.
- Promote premium cards/loyalty cards to increase transactions.

Cluster-2 :

- Customers have a medium spending group. Banks can improve their credit limit and decrease interest rate.

- Promote premium cards/loyalty cards to increase transactions.
- Increase spending habits with premium sites, travel portal, travel airlines/hotel, as this will encourage them to spend more.

Problem 2: CART-RF-ANN

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

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

Exploratory Data Analysis

Sample of the datasets

	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

Table-11 Dataset

Let us check the types of variables and missing values in the dataset

- From the below results we can see that there is no missing value present in the dataset.
- There are a total of 3000 rows and 10 columns in the dataset.
- Out of 10 there are 2 float64, 2 int64 and 6 objects.

```
<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
```

Check for duplicate data

- 4.63 % of data is duplicate.
- I am not going to remove any duplicate values because there might be a different person taking the same feature as taking the other and I am not getting any unique identifiers So keep this data for the algorithm.

Univariate Analysis :

	Age	Commision	Duration	Sales
count	3000.000000	3000.000000	3000.000000	3000.000000
mean	38.091000	14.529203	70.001333	60.249913
std	10.463518	25.481455	134.053313	70.733954
min	8.000000	0.000000	-1.000000	0.000000
25%	32.000000	0.000000	11.000000	20.000000
50%	36.000000	4.630000	26.500000	33.000000
75%	42.000000	17.235000	63.000000	69.000000
max	84.000000	210.210000	4580.000000	539.000000

Table-12 Describe data

	Agency_Code	Type	Claimed	Channel	Product Name	Destination
count	3000	3000	3000	3000	3000	3000
unique	4	2	2	2	5	3
top	EPX	Travel Agency	No	Online	Customised Plan	ASIA
freq	1365	1837	2076	2954	1136	2465

Table-13 Describe data

- There are 4 numerical and 6 categorical values.
- Most frequent destination is ASIA 2465.
- Most of the people are taking online channel 2954.
- EPX agency code frequency is 1365.

Distplot:

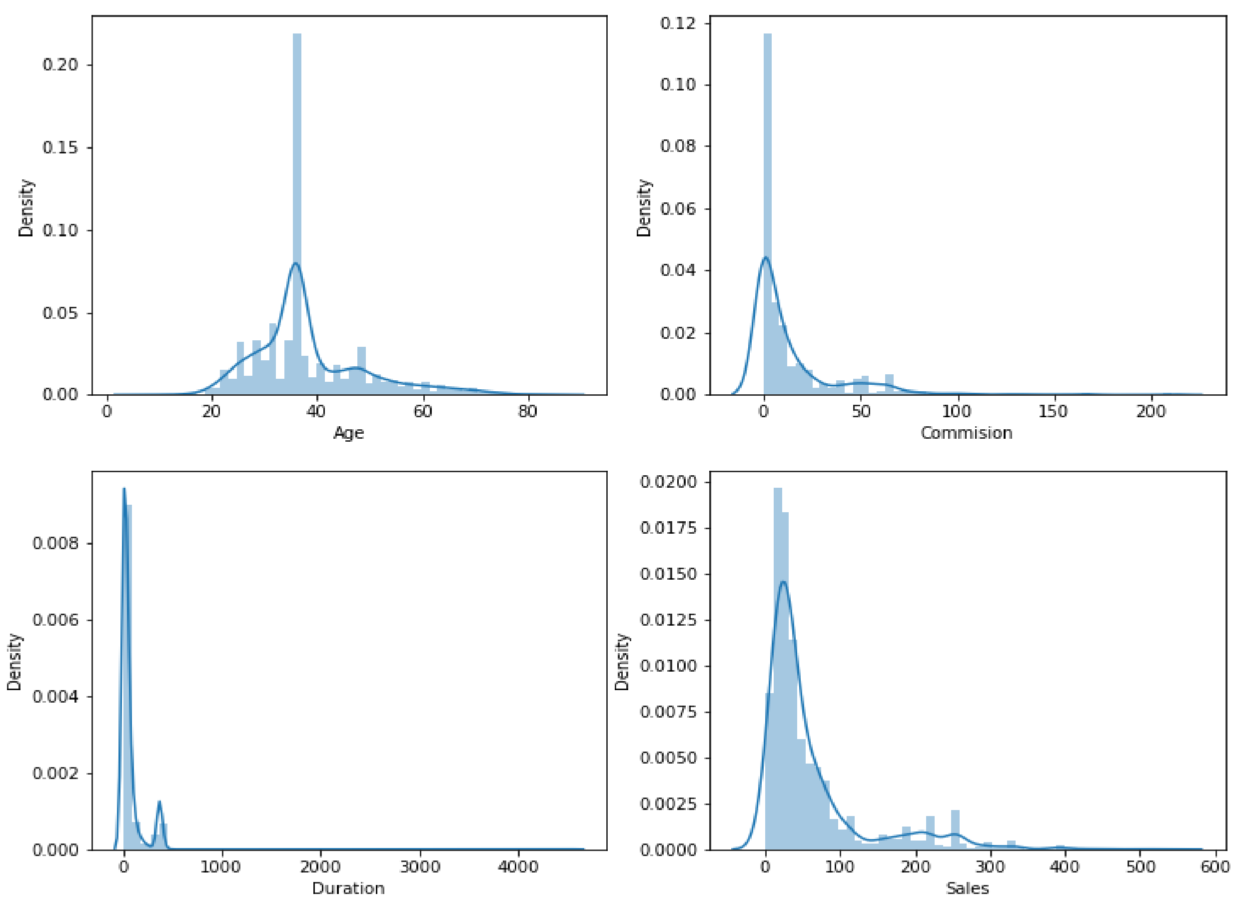


Figure-10 Distplot

Boxplot:

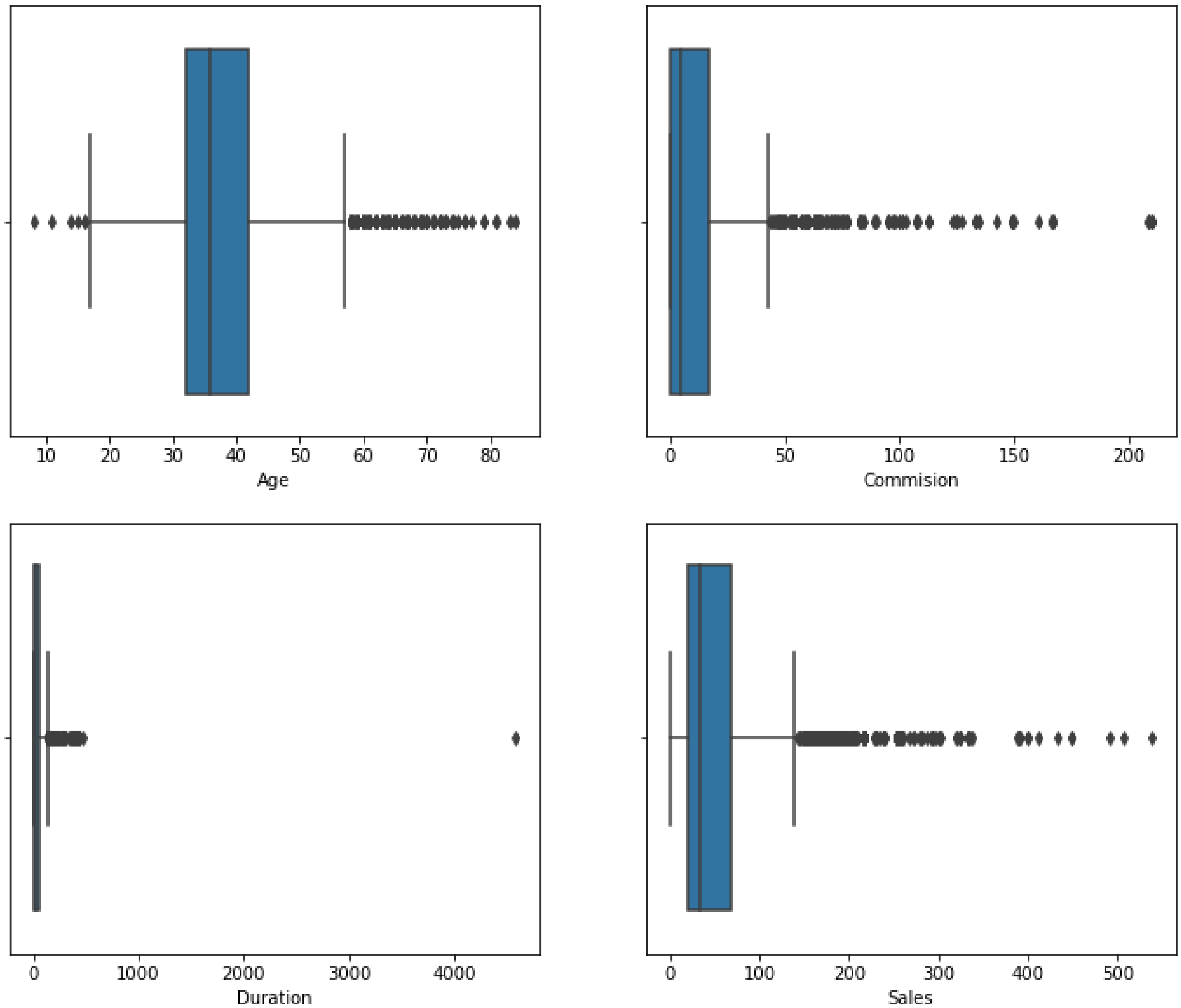


Figure-11 Boxplot

Age feature:

- The boxplot of Age feature shows outliers.
- Age is positively skewed : 1.1497.
- In the boxplot 20 to 30 shows the majority of distribution.

Commision feature :

- The boxplot of the commission feature shows outliers.
- Commission feature positively skewed : 3.1489.

Duration feature :

- The boxplot of the duration feature shows outliers.
- Duration feature positively skewed : 13.7847.

Sales feature :

- The boxplot of the sales feature shows outliers.
- Sales feature positively skewed : 2.3811.

Countplot for categorical variables :

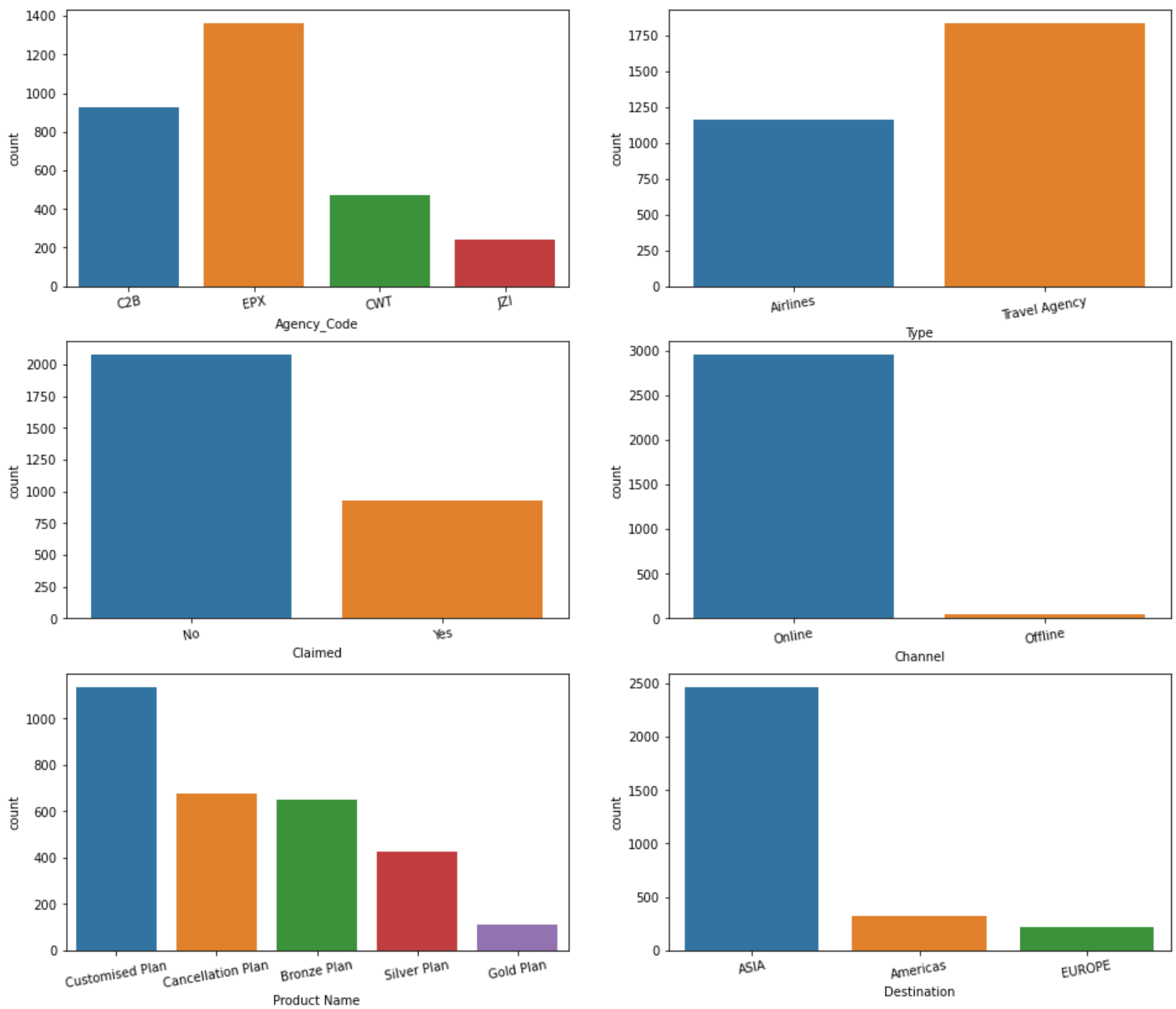


Figure-12 countplot

```
AGENCY_CODE :  4
EPX    1365
C2B     924
CWT     472
JZI     239
Name: Agency_Code, dtype: int64

****************************

TYPE :  2
Travel Agency    1837
Airlines         1163
Name: Type, dtype: int64

****************************

CLAIMED :  2
No     2076
Yes     924
Name: Claimed, dtype: int64

****************************

CHANNEL :  2
Online     2954
Offline      46
Name: Channel, dtype: int64

****************************
```

```
CHANNEL :  2
Online      2954
Offline       46
Name: Channel, dtype: int64

****************************


PRODUCT NAME :  5
Customised Plan      1136
Cancellation Plan     678
Bronze Plan           650
Silver Plan           427
Gold Plan             109
Name: Product Name, dtype: int64

****************************


DESTINATION :  3
ASIA         2465
Americas      320
EUROPE        215
Name: Destination, dtype: int64

****************************
```

Bi-variate Analysis:

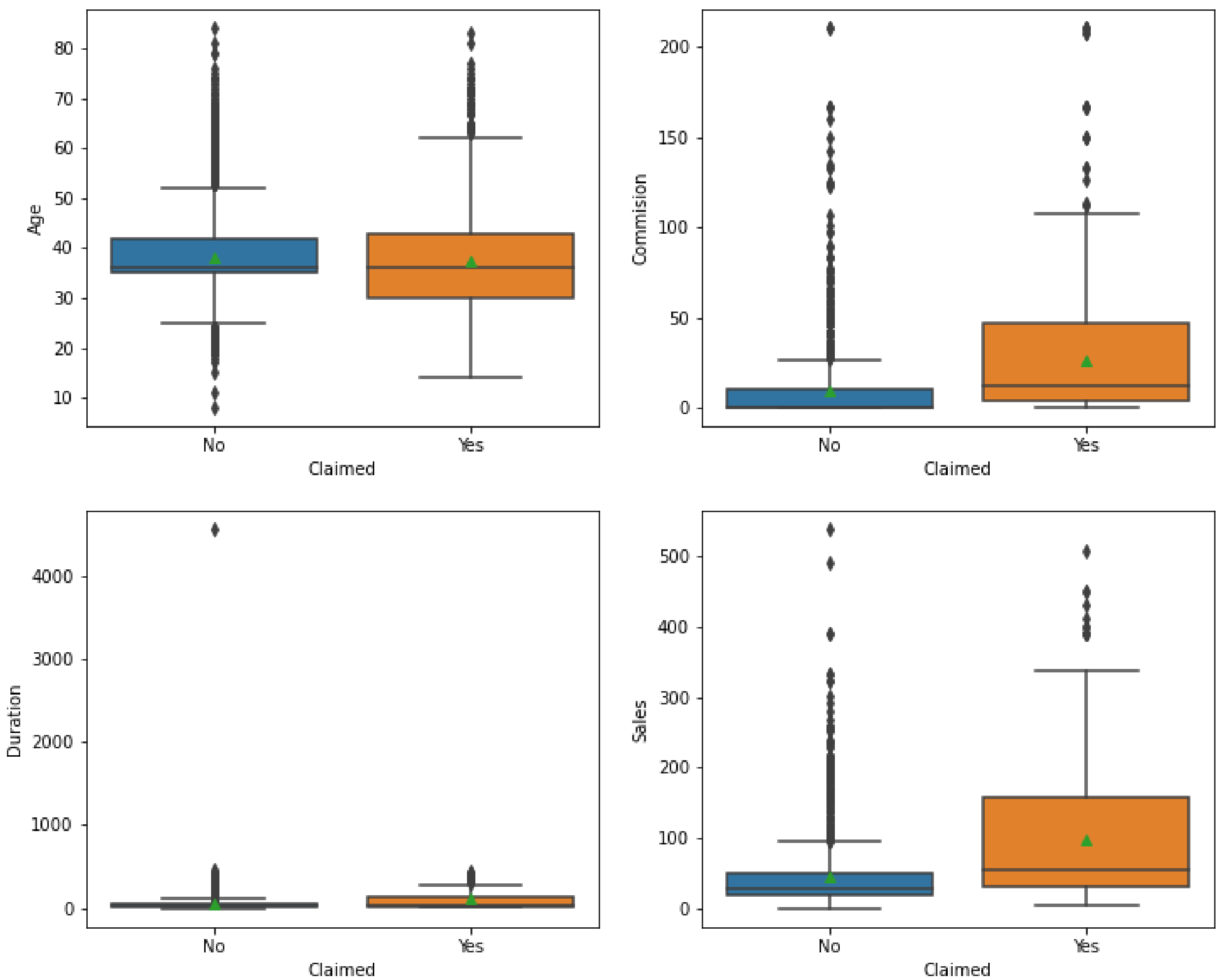
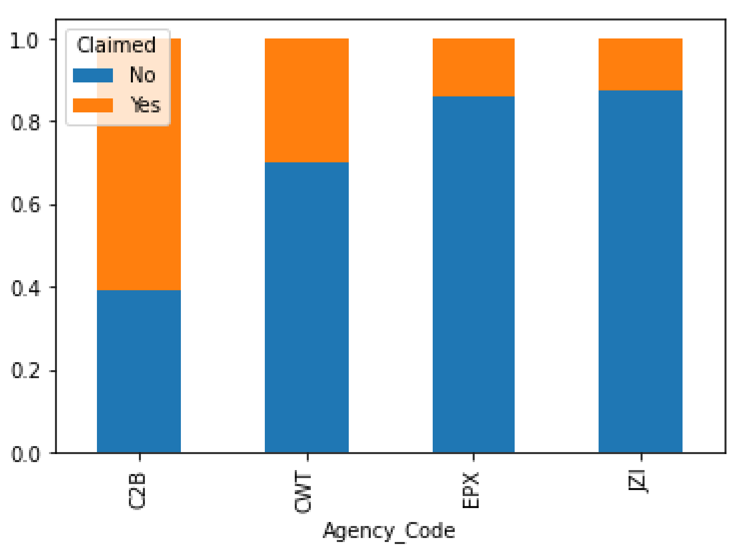
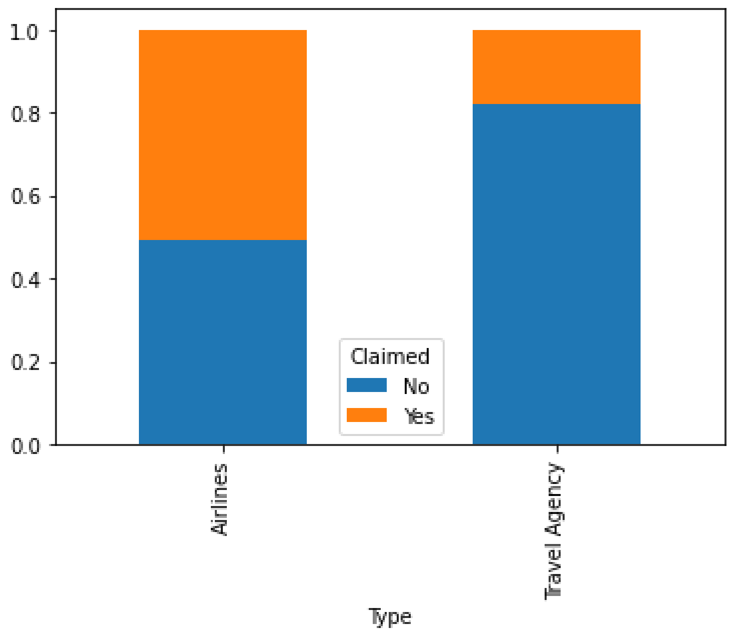


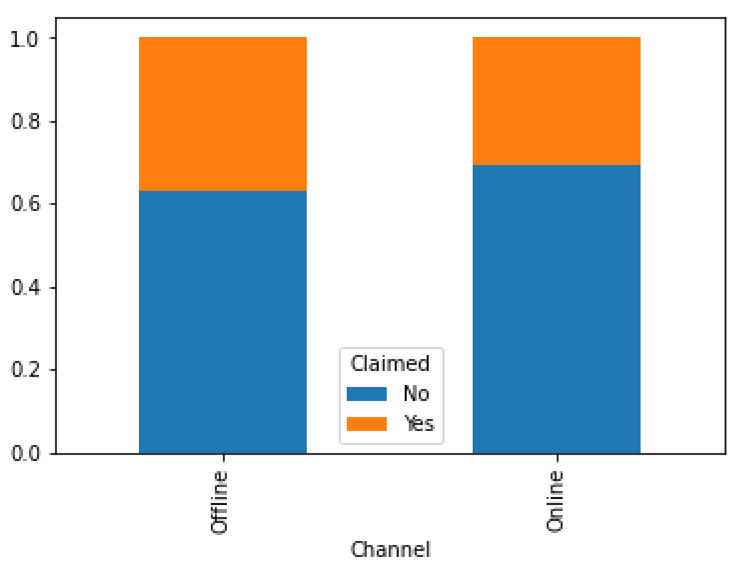
Figure-13 Boxplot



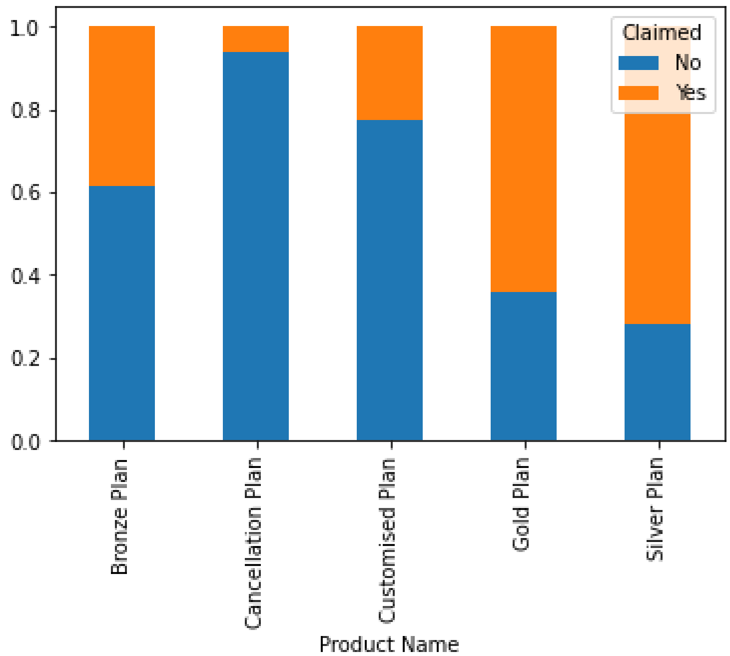
- We can see the above bar graph and observe that C2B agency code has a chance to claim the tour insurance.
- EPX and JZI agency code have less chance to claim the tour insurance.



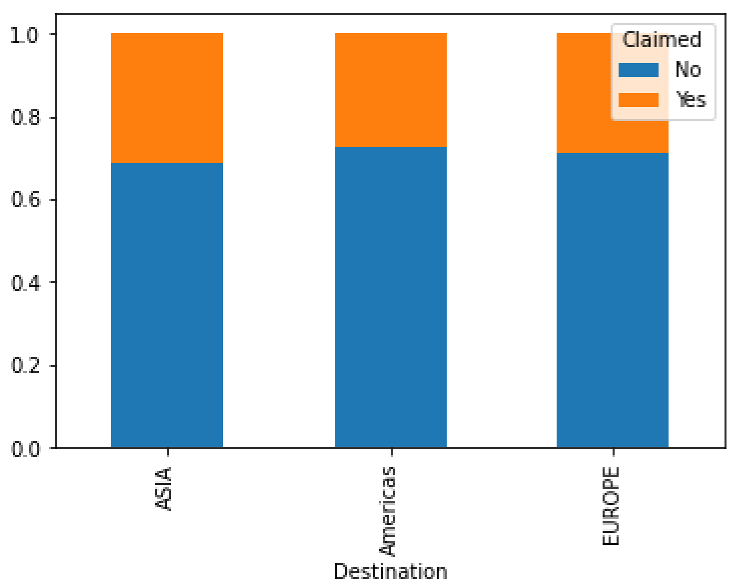
- Airlines have a high chance to claim the tour insurance.
- Travel agencies have less chance to claim the tour insurance.



- Offline channels have a high chance to claim the tour insurance.
- Online channels have less chance to claim the tour insurance.



- Silver plan and Gold plan have a chance to claim the tour insurance.
- Cancellation plans have less chance to claim the tour insurance.



- Asia, Americas and Europe all have almost equal chances to claim.

Multivariate Analysis :

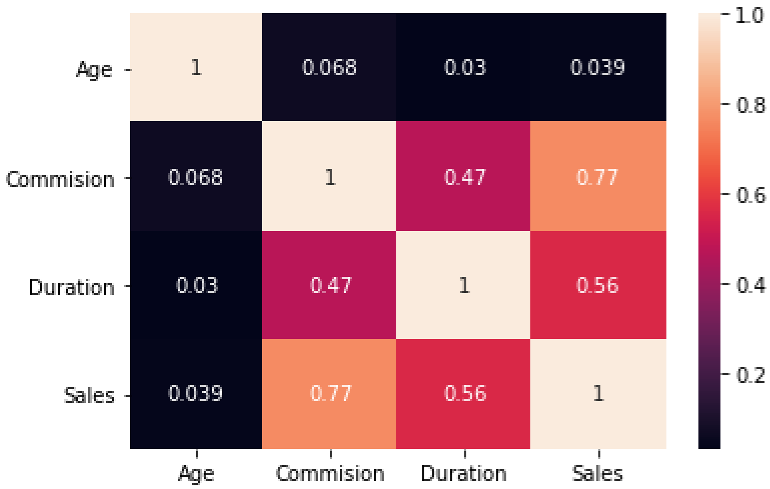


Figure- 14 Heatmap

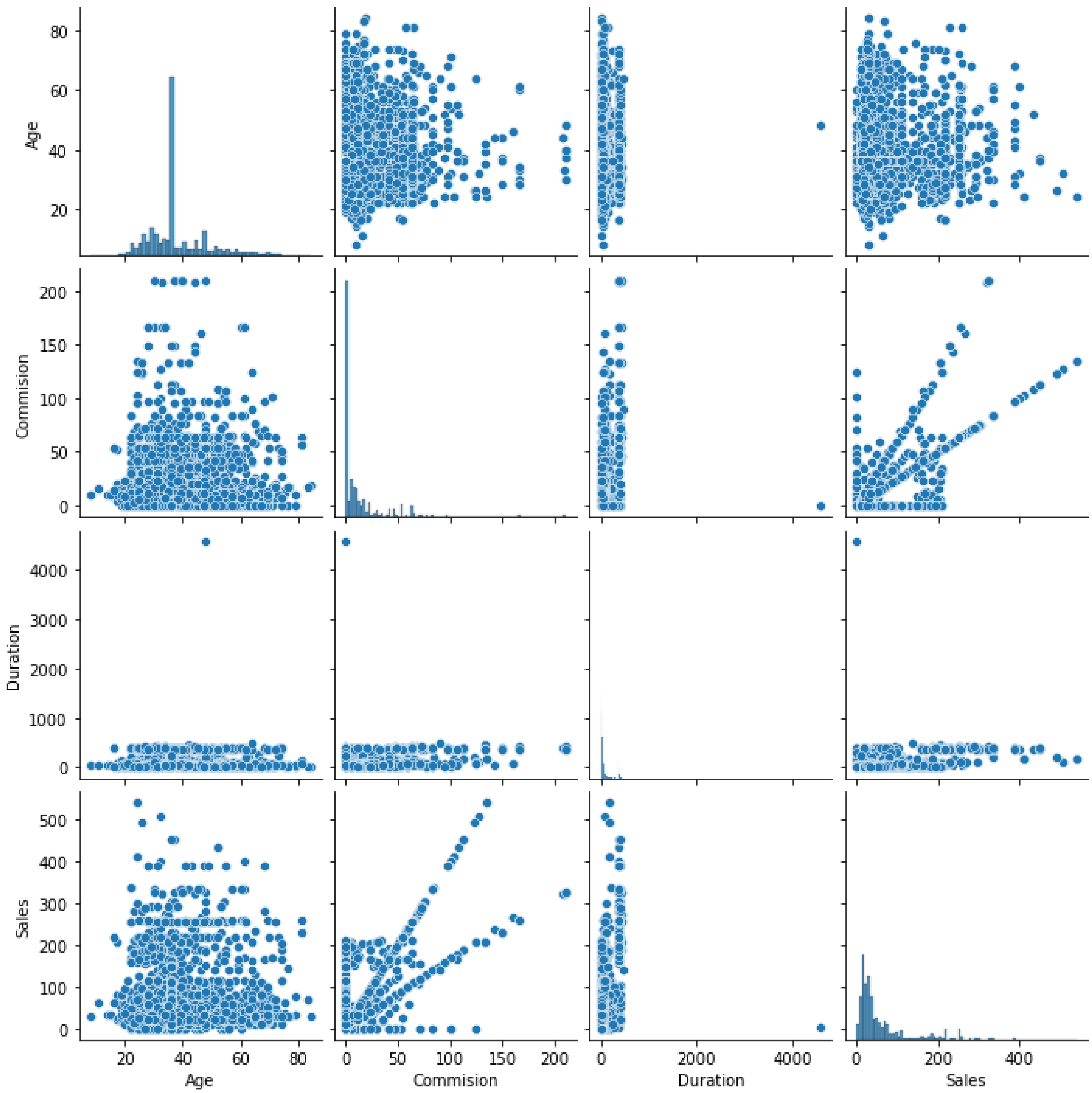


Figure-14 Pairplot

- After observing heatmap and pairplot figures there is not much multicollinearity.
- There is no negative correlation.

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

To build the models we have to change the object data types to numeric values.

```
feature: Agency_Code
['C2B', 'EPX', 'CWT', 'JZI']
Categories (4, object): ['C2B', 'CWT', 'EPX', 'JZI']
[0 2 1 3]


feature: Type
['Airlines', 'Travel Agency']
Categories (2, object): ['Airlines', 'Travel Agency']
[0 1]


feature: Claimed
['No', 'Yes']
Categories (2, object): ['No', 'Yes']
[0 1]


feature: Channel
['Online', 'Offline']
Categories (2, object): ['Offline', 'Online']
[1 0]


feature: Product Name
['Customised Plan', 'Cancellation Plan', 'Bronze Plan', 'Silver Plan', 'Gold Plan']
Categories (5, object): ['Bronze Plan', 'Cancellation Plan', 'Customised Plan', 'Gold Plan', 'Silver Plan']
[2 1 0 4 3]


feature: Destination
['ASIA', 'Americas', 'EUROPE']
Categories (3, object): ['ASIA', 'Americas', 'EUROPE']
[0 1 2]
```

Now we can see the below data types are all numerical values.

```
<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   int8   
 2   Type          3000 non-null   int8   
 3   Claimed       3000 non-null   int8   
 4   Commision     3000 non-null   float64
 5   Channel       3000 non-null   int8   
 6   Duration      3000 non-null   int64  
 7   Sales         3000 non-null   float64
 8   Product Name  3000 non-null   int8   
 9   Destination   3000 non-null   int8   
dtypes: float64(2), int64(2), int8(6)
memory usage: 111.5 KB
```

- Drop the target columns (Claimed) for the training and testing set.
- For training and testing purposes we are splitting the dataset into train and test data in the ratio 70:30 .
- After splitting the dimensions of the training and test data.

```
X_train (2100, 9)
X_test (900, 9)
train_labels (2100,)
test_labels (900,)
```

Model-1 Building a Decision Tree Classifier:

We use criterion “gini”

```
dt_model= DecisionTreeClassifier(criterion='gini')
```

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

executed in 46ms, finished 22:39:34 2021-10-21

```
DecisionTreeClassifier()
```

Grid search for finding optimal values for decision tree.

```
param_grid = {
    'criterion': ['gini'],
    'max_depth': [10,12,14,16],
    'min_samples_leaf': [20,25,50],
    'min_samples_split': [100,150,200,250],
}

dtcl = DecisionTreeClassifier(random_state=1)

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

Getting the optimal values for the training dataset.

```
grid_search.fit(X_train, train_labels)

best_grid_dt = grid_search.best_estimator_
best_grid_dt
```

executed in 7.48s, finished 22:44:38 2021-10-21

```
DecisionTreeClassifier(max_depth=10, min_samples_leaf=25, min_samples_split=150,
                       random_state=1)
```

We get max_depth=10

min_samples_leaf=25

min_samples_split=150

We can check what is the feature importance of the given above optimal values of training dataset.

	Imp
Agency_Code	0.563514
Sales	0.246609
Product Name	0.071979
Age	0.046292
Duration	0.043429
Commision	0.021206
Type	0.006972
Channel	0.000000
Destination	0.000000

Predicting on Training and Test dataset:

```
ytrain_predict_dt = best_grid_dt.predict(X_train)
ytest_predict_dt = best_grid_dt.predict(X_test)
```

Model-2 Building a Random Forest Classifier :

```
rfcl=RandomForestClassifier(n_estimators=100,max_features=6,random_state=1)
rfcl=rfcl.fit(X_train,train_labels)
```

grid search for finding out optimal values for the hyper parameters to build a Random Forest Classifier

```
param_grid = {
    'max_depth': [10,12,15],
    'max_features': [3,4,5],
    'min_samples_split': [40,45,60],
    'n_estimators': [101]
}

rfcl = RandomForestClassifier(random_state=0)

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

Getting the optimal values for the training dataset.

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

```
grid_search.best_params_
```

executed in 4ms, finished 22:57:50 2021-10-21

```
{'max_depth': 12,
 'max_features': 3,
 'min_samples_split': 40,
 'n_estimators': 101}
```

We can check what is the feature importance of the given above optimal values of training dataset.

```
                  Imp
Agency_Code    0.227565
Sales          0.174086
Product Name   0.166902
Commision      0.145033
Duration       0.128431
Age            0.087442
Type           0.045076
Destination    0.015456
Channel        0.010010
```

Predicting on Training and Test dataset:

```
ytrain_predict_rfcl = best_grid_rfcl.predict(X_train)
ytest_predict_rfcl = best_grid_rfcl.predict(X_test)
```

Model-3 Building a Neural Network Classifier:

- For the neural network classifier data should be in the same scale.
- Before moving to building the neural network I scaled the data by using StandardScaler.

```
from sklearn.preprocessing import StandardScaler
std_scale = StandardScaler()
X_train_ncc = std_scale.fit_transform(X_train)
X_test_ncc = std_scale.transform(X_test)
```

```
clf=MLPClassifier(hidden_layer_sizes=100,max_iter=500,solver='sgd',random_state=1,tol=0.01)
```

```
clf.fit(X_train_ncc,train_labels)
```

executed in 343ms, finished 00:00:06 2021-10-22

```
MLPClassifier(hidden_layer_sizes=100, max_iter=500, random_state=1,
              solver='sgd', tol=0.01)
```

grid search for finding out optimal values for the hyper parameters to build the neural network classifier

```
param_grid = {
    'hidden_layer_sizes': [8],
    'max_iter': [2500,3000],
    'solver': ['adam', 'sgd'],
    'tol': [0.001, 0.01],
}

nncl = MLPClassifier(random_state=1)

grid_search = GridSearchCV(estimator = nncl, param_grid = param_grid, cv = 5)
```

Getting the optimal values for the training dataset.

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

executed in 12.5s, finished 23:12:32 2021-10-21

```
{'hidden_layer_sizes': 8, 'max_iter': 2500, 'solver': 'adam', 'tol': 0.001}
```

As we know that we can not get the features' importance in neural networks because we can't give the hidden information that why neural networks are also called the black box.

Predicting on Training and Test dataset:

```
ytrain_predict_nnc = best_grid_nnc.predict(X_train_ncc)
ytest_predict_nnc = best_grid_nnc.predict(X_test_ncc)
```

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.

CART Confusion Matrix for the training set

```
confusion_matrix(train_labels, ytrain_predict_dt)
```

executed in 29ms, finished 22:29:14 2021-10-22

```
array([[1332,  139],
       [ 282,  347]], dtype=int64)
```

In the confusion matrix we get 2X2 table

True Negative (TN)

[0][0]= 1332

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

[0][1]=139

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

False Negative (FN) – Type 2 error

[1][0]=282

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

True Positive (TP)

[1][1]=347

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

Accuracy of the training data

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

executed in 26ms, finished 22:30:20 2021-10-22

```
0.7995238095238095
```

Accuracy of the Model for the training data is around 80%.

CART Classification Report for the training data

```
print(classification_report(train_labels, ytrain_predict_dt))
```

executed in 19ms, finished 22:30:40 2021-10-22

	precision	recall	f1-score	support
0	0.83	0.91	0.86	1471
1	0.71	0.55	0.62	629
accuracy			0.80	2100
macro avg	0.77	0.73	0.74	2100
weighted avg	0.79	0.80	0.79	2100

- recall(1):- it means how many of the actual true data points are identified as true data points by the model.
- As we can in the above table 55% of the recall-1.
- precision(1):- it means among the points by the model, how many are really positive.
- As we can in the above table 71% of precision-1.
- F1 score - F1 Score is the weighted average of Precision and Recall.
- F1 Score = 2*(Recall * Precision) / (Recall + Precision)

AUC and ROC for the training data of the CART

```
AUC: 0.845

Text(0, 0.5, 'True Positive Rate')
```

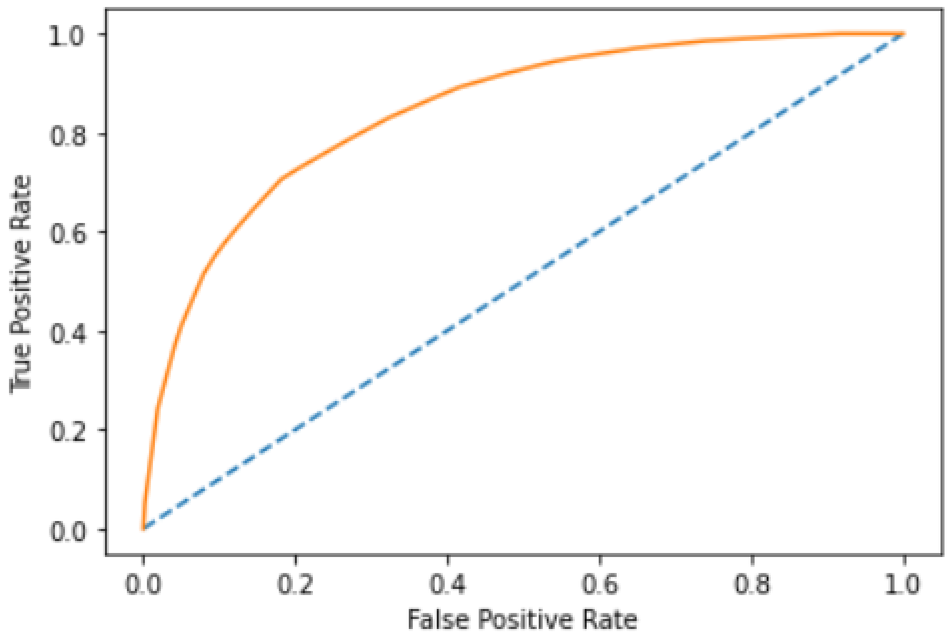


Fig. 16 roc curve

- AUC stands for area under the curve for the training data is 0.845.
- ROC graph is a trade off between True positive rate and false positive rate.

CART Confusion Matrix for the test set

```
: confusion_matrix(test_labels, ytest_predict_dt)
executed in 25ms, finished 22:34:58 2021-10-22

: array([[557,  48],
         [171, 124]], dtype=int64)
```

In the confusion matrix we get 2X2 table

True Negative (TN)

[0][0]= 557

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

[0][1]=48

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

False Negative (FN) – Type 2 error

[1][0]=171

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

True Positive (TP)

[1][1]=124

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

Accuracy of the test data

```
cart_train_acc=best_grid_dt.score(X_test,test_labels)
cart_train_acc
```

executed in 25ms, finished 22:35:18 2021-10-22

```
0.7566666666666667
```

Accuracy of the Model for the testing data is around 75.7%.

CART Classification Report for the test data

```
print(classification_report(test_labels, ytest_predict_dt))
```

executed in 19ms, finished 22:35:30 2021-10-22

	precision	recall	f1-score	support
0	0.77	0.92	0.84	605
1	0.72	0.42	0.53	295
accuracy			0.76	900
macro avg	0.74	0.67	0.68	900
weighted avg	0.75	0.76	0.74	900

- recall(1):- it means how many of the actual true data points are identified as true data points by the model.
- As we can in the above table 42% of the recall-1.
- precision(1):- it means among the points by the model, how many are really positive.
- As we can in the above table 72% of precision-1.
- F1 score - F1 Score is the weighted average of Precision and Recall.
- F1 Score = 2*(Recall * Precision) / (Recall + Precision)

AUC and ROC for the test data of the CART

```
AUC: 0.799

[<matplotlib.lines.Line2D at 0x1f999d6fdc0>]
```

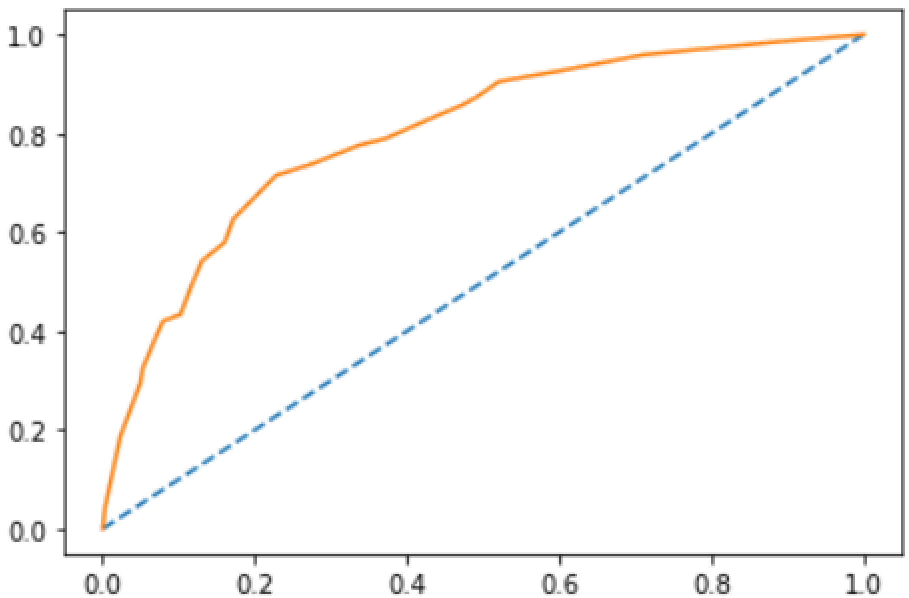


Fig.17 roc curve

- AUC stands for area under the curve for the training data is 0.799.
- ROC graph is a trade off between True positive rate and false positive rate.

Validation of models CART

Accuracy of training data is 80%

Accuracy of testing data is 75.6%

As we can see all observations of training data and testing data we can say that model is similar for training and testing data.

There is no overfitting or underfitting chances if the difference between accuracy of training data and testing data is greater than 10% then there is a chance of overfitting or underfitting.

RF Confusion Matrix for the training set

```
confusion_matrix(train_labels, ytrain_predict_rfcl)
```

executed in 17ms, finished 21:29:30 2021-10-23

```
array([[1350,  121],
       [ 235,  394]], dtype=int64)
```

In the confusion matrix we get 2X2 table

True Negative (TN)

[0][0]= 1350

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

[0][1]=121

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

False Negative (FN) – Type 2 error

[1][0]=235

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

True Positive (TP)

[1][1]=394

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

Accuracy of the training data

```
rf_train_acc=best_grid_rfcl.score(X_train,train_labels)
rf_train_acc
```

executed in 46ms, finished 21:29:33 2021-10-23

```
0.8304761904761905
```

Accuracy of the Model for the training data is around 83%.

RF Classification Report for the training data

```
print(classification_report(train_labels, ytrain_predict_rfcl))
```

executed in 27ms, finished 21:29:35 2021-10-23

	precision	recall	f1-score	support
0	0.85	0.92	0.88	1471
1	0.77	0.63	0.69	629
accuracy			0.83	2100
macro avg	0.81	0.77	0.79	2100
weighted avg	0.83	0.83	0.83	2100

- recall(1):- it means how many of the actual true data points are identified as true data points by the model.
- As we can in the above table 63% of the recall-1.
- precision(1):- it means among the points by the model, how many are really positive.
- As we can in the above table 77% of precision-1.
- F1 score - F1 Score is the weighted average of Precision and Recall.
- F1 Score = 2*(Recall * Precision) / (Recall + Precision)

AUC and ROC for the training data of the RF

```
AUC: 0.894
```

```
Text(0, 0.5, 'True Positive Rate')
```

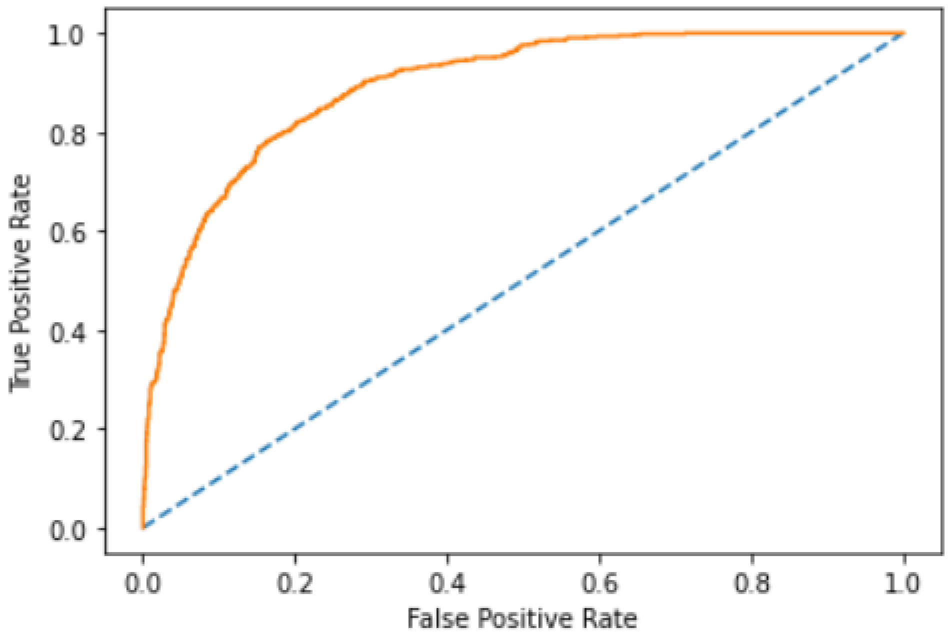


Fig.18 roc curve

- AUC stands for area under the curve for the training data is 0.894.
- ROC graph is a trade off between True positive rate and false positive rate.

RF Confusion Matrix for the testing set

```
confusion_matrix(test_labels, ytest_predict_rfcl)
```

executed in 26ms, finished 21:29:44 2021-10-23

```
array([[553,  52],
       [159, 136]], dtype=int64)
```

In the confusion matrix we get 2X2 table

True Negative (TN)

[0][0]= 553

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

[0][1]=52

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

False Negative (FN) – Type 2 error

[1][0]=159

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

True Positive (TP)

[1][1]=136

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

Accuracy of the testing data

```
rf_test_acc=best_grid_rfcl.score(X_test,test_labels)
rf_test_acc
```

executed in 43ms, finished 21:29:47 2021-10-23

```
0.7655555555555555
```

Accuracy of the Model for the testing data is around 76.5%.

RF Classification Report for the testing data

```
print(classification_report(test_labels, ytest_predict_rfcl))
```

executed in 15ms, finished 21:29:50 2021-10-23

```
              precision    recall  f1-score   support

           0       0.78      0.91      0.84       605
           1       0.72      0.46      0.56       295

    accuracy                           0.77       900
   macro avg       0.75      0.69      0.70       900
weighted avg       0.76      0.77      0.75       900
```

- recall(1):- it means how many of the actual true data points are identified as true data points by the model.
- As we can in the above table 46% of the recall-1.
- precision(1):- it means among the points by the model, how many are really positive.
- As we can in the above table 72% of precision-1.
- F1 score - F1 Score is the weighted average of Precision and Recall.
- F1 Score = 2*(Recall * Precision) / (Recall + Precision)

AUC and ROC for the testing data of the RF

```
AUC: 0.824
```

```
[<matplotlib.lines.Line2D at 0x1e0cddeae50>]
```

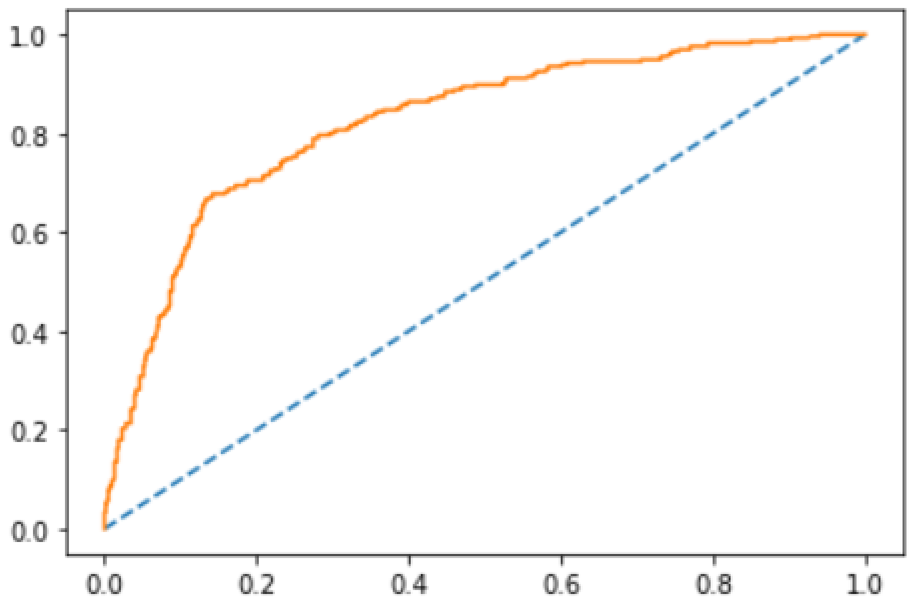


Fig.19 roc curve

- AUC stands for area under the curve for the training data is 0.824.
- ROC graph is a trade off between True positive rate and false positive rate.

Validation of models RF

Accuracy of training data is 83%

Accuracy of testing data is 76.5%

As we can see all observations of training data and testing data we can say that model is similar for training and testing data.

There is no overfitting or underfitting chances if the difference between accuracy of training data and testing data is greater than 10% then there is a chance of overfitting or underfitting.

ANN Confusion Matrix for the training set

```
confusion_matrix(train_labels, ytrain_predict_nnc)
executed in 29ms, finished 21:30:00 2021-10-23

array([[1308,  163],
       [ 328,  301]], dtype=int64)
```

In the confusion matrix we get 2X2 table

True Negative (TN)

[0][0]= 1308

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

[0][1]=163

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

False Negative (FN) – Type 2 error

[1][0]=328

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

True Positive (TP)

[1][1]=301

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

Accuracy of the training data

```
ann_train_acc=best_grid_nnc.score(X_train_ncc,train_labels)
ann_train_acc
```

executed in 7ms, finished 21:30:02 2021-10-23

```
0.7661904761904762
```

Accuracy of the Model for the training data is around 76.6%.

ANN Classification Report for the training data

```
print(classification_report(train_labels, ytrain_predict_nnc))
```

executed in 29ms, finished 21:30:05 2021-10-23

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84      1471
           1       0.65      0.48      0.55       629

    accuracy                           0.77      2100
   macro avg       0.72      0.68      0.70      2100
weighted avg       0.75      0.77      0.75      2100
```

- recall(1):- it means how many of the actual true data points are identified as true data points by the model.
- As we can in the above table 48% of the recall-1.
- precision(1):- it means among the points by the model, how many are really positive.
- As we can in the above table 65% of precision-1.
- F1 score - F1 Score is the weighted average of Precision and Recall.
- F1 Score = 2*(Recall * Precision) / (Recall + Precision)

AUC and ROC for the training data of the ANN

```
AUC: 0.798
```

```
Text(0, 0.5, 'True Positive Rate')
```

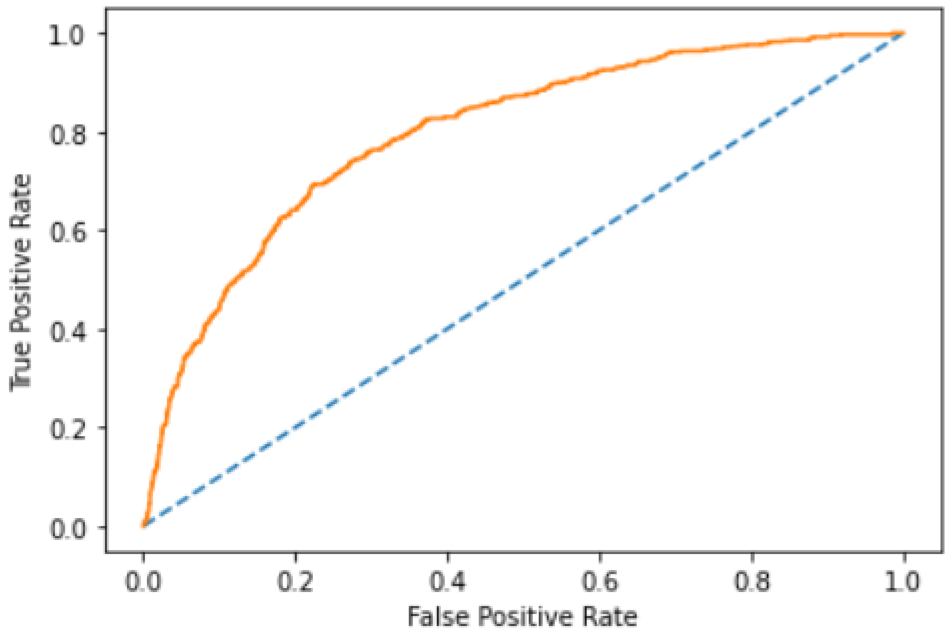


Fig. 20 roc curve

ANN Confusion Matrix for the testing set

```
confusion_matrix(test_labels, ytest_predict_nnc)
```

executed in 21ms, finished 21:30:15 2021-10-23

```
array([[559,  46],
       [181, 114]], dtype=int64)
```

In the confusion matrix we get 2X2 table

True Negative (TN)

[0][0]= 559

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

[0][1]=46

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

False Negative (FN) – Type 2 error

[1][0]=181

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

True Positive (TP)

[1][1]=114

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

Accuracy of the testing data

```
ann_test_acc=best_grid_nnc.score(X_test_ncc,test_labels)
ann_test_acc
```

executed in 17ms, finished 21:30:18 2021-10-23

```
0.7477777777777778
```

ANN Classification Report for the testing data

```
print(classification_report(test_labels, ytest_predict_nnc))
```

executed in 23ms, finished 21:30:21 2021-10-23

```
              precision    recall  f1-score   support

           0       0.76      0.92      0.83       605
           1       0.71      0.39      0.50       295

    accuracy                           0.75       900
   macro avg       0.73      0.66      0.67       900
weighted avg       0.74      0.75      0.72       900
```

- recall(1):- it means how many of the actual true data points are identified as true data points by the model.
- As we can in the above table 39% of the recall-1.

- precision(1):- it means among the points by the model, how many are really positive.
- As we can in the above table 71% of precision-1.
- F1 score - F1 Score is the weighted average of Precision and Recall.
- F1 Score = 2*(Recall * Precision) / (Recall + Precision)

AUC and ROC for the testing data of the ANN

```
AUC: 0.790

Text(0, 0.5, 'True Positive Rate')
```

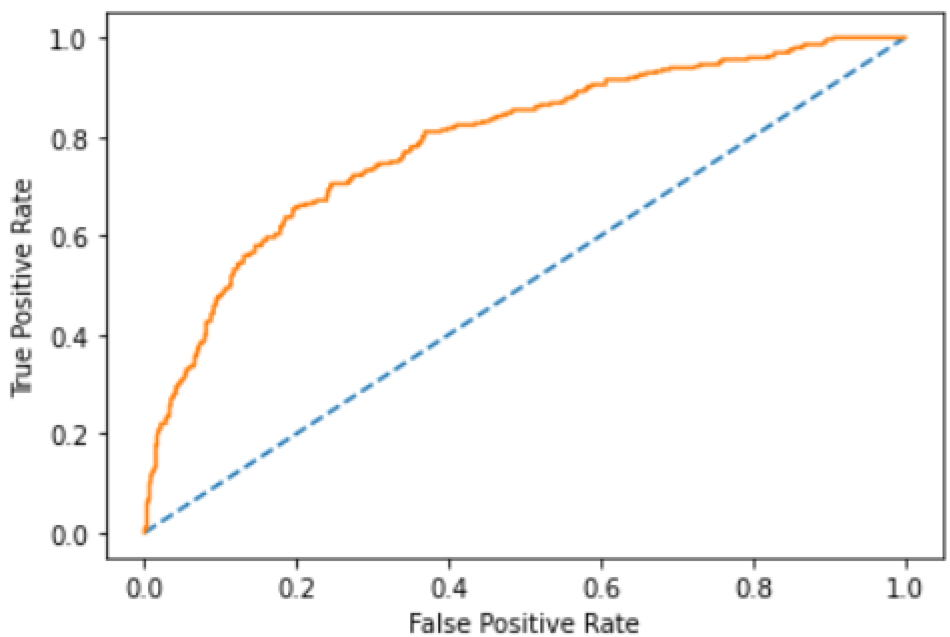


Fig. 21 roc curve

- AUC stands for area under the curve for the training data is 0.790.
- ROC graph is a trade off between True positive rate and false positive rate.

Validation of models ANN

Accuracy of training data is 76.6%

Accuracy of testing data is 74.7%

As we can see all observations of training data and testing data we can say that model is similar for training and testing data.

There is no overfitting or underfitting chances if the difference between accuracy of training data and testing data is greater than 10% then there is a chance of overfitting or underfitting.

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

Comparison of the performance metrics of all 3 models

	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.80	0.76	0.83	0.77	0.77	0.75
AUC	0.84	0.80	0.89	0.82	0.80	0.79
Recall	0.55	0.42	0.63	0.46	0.48	0.39
Precision	0.71	0.72	0.77	0.72	0.65	0.71
F1 Score	0.62	0.53	0.69	0.56	0.55	0.50

Table -14 Matrix

ROC Curve for the 3 models on the Training data

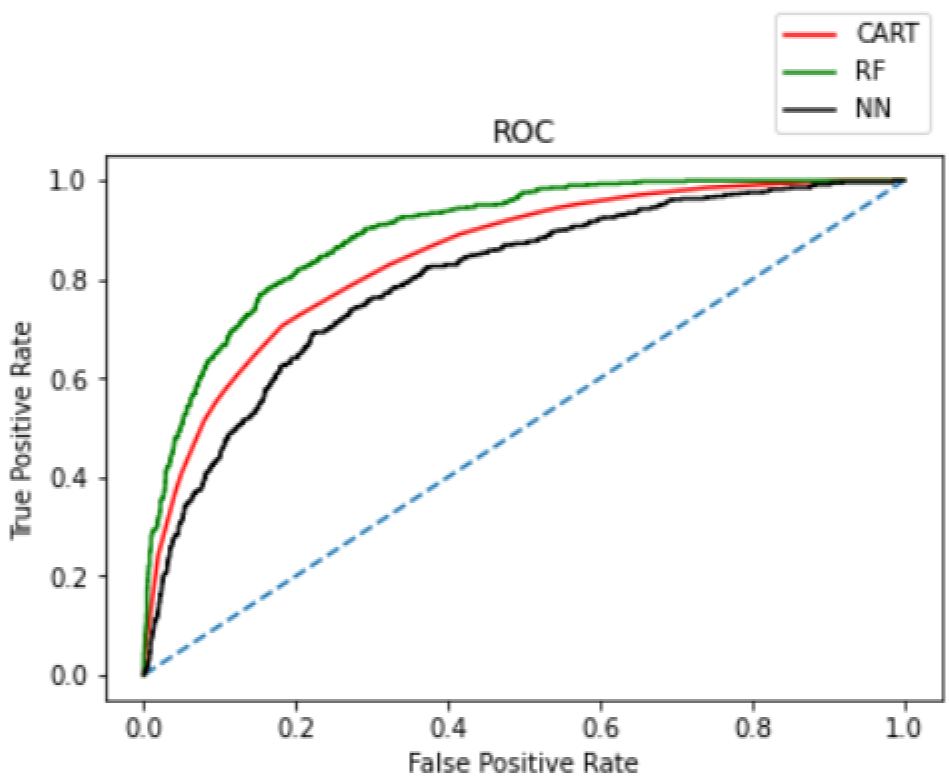


Fig. 22 roc curve

ROC Curve for the 3 models on the Test data

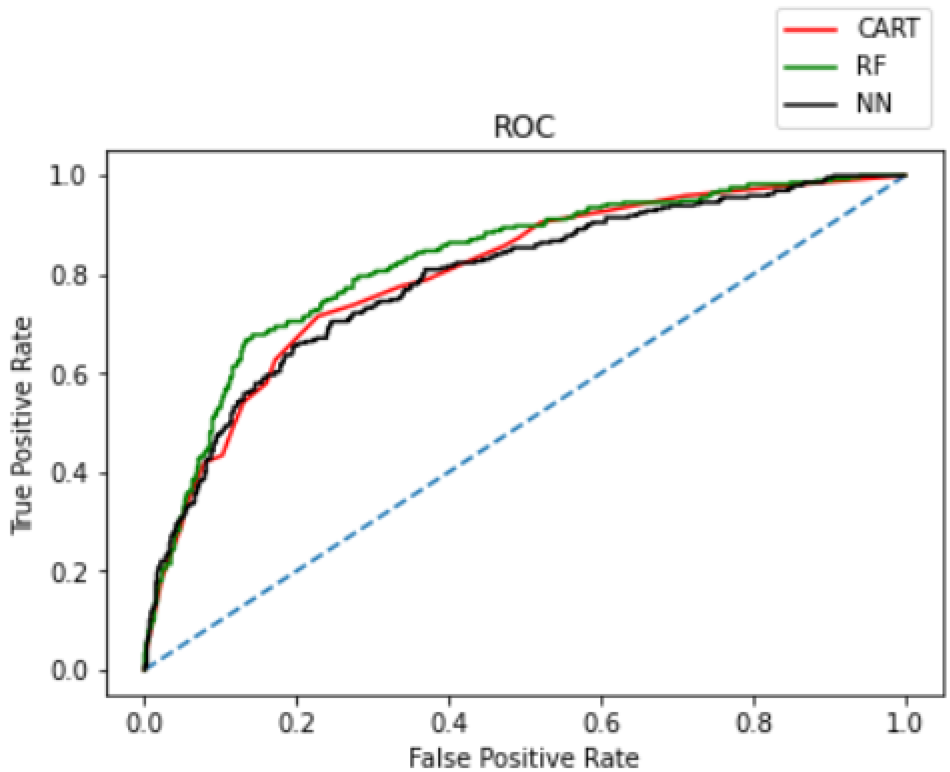


Fig. 23 roc curve

CONCLUSION:

We can see the above matrix table and roc for training and testing datasets. We would like to like the RF-model because their performance of training and testing data is better than the other two models.

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

- In the whole dataset there are around 4.63 % of the data is duplicate so discuss with the data collector and if that is not duplicate then we don't need to remove data and if that is duplicate data then before modeling the data we must remove data it will impact model performance.
- need to run promotional marketing campaigns or evaluate if we need to tie up with an alternate agency. It will increase sales.
- Gold plan sales are less as per other we should campaigns for this and can give them reward it will increase the gold plan.

- In the destination point Asia graph is large as compared with Americas and Europe, so the agency should target the Americas and Europe destinations to increase the revenue.
- C2B agency code is claimed more so we should evaluate this.
- We have also built the model for predicting the claim status so after customers book tour insurance based on their data try to predict the claim status as per pattern.
- Key performance indicators (KPI).
    1. Increase the customer's satisfaction.
    2. Optimized claims recovery methods.
    3. Reduce the claim handling costs.