DATA MINING PROJECT REPORT

Contents

Problem 1: Clustering

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

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

Checking the first 5 rows of the data:

	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

```
<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
```

- There are 7 variables and 210 records.
- There are no missing values based on intial analysis.
- All the variables are numeric.

checking for missing value:

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

- There are no missing values
- There are no duplicate rows

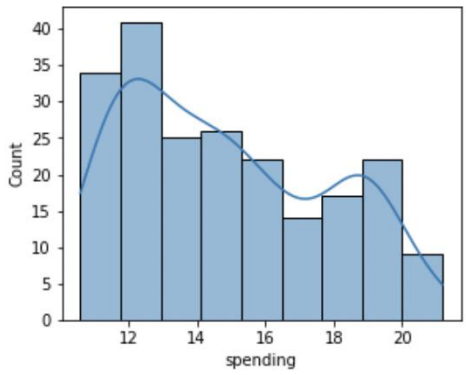
Exploratory Data Analysis:

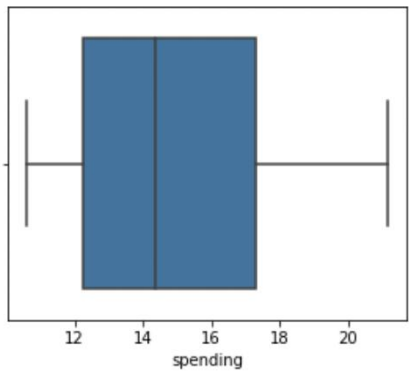
univariate analysis:

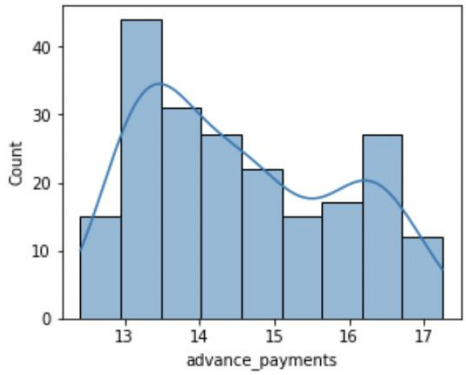
	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000
mean	14.847524	14.559286	0.870999	5.628533	3.258605	3.700201	5.408071
std	2.909699	1.305959	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

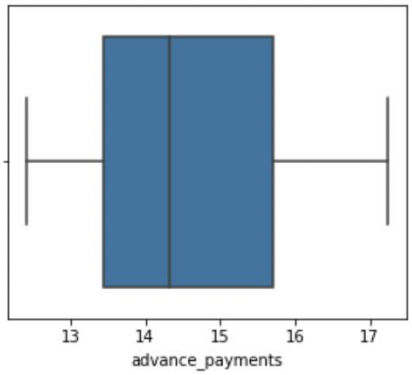
- We see that for most of the variables, mean and median(50th percentile) are nearly equal
- Std Deviation is high for spending variable

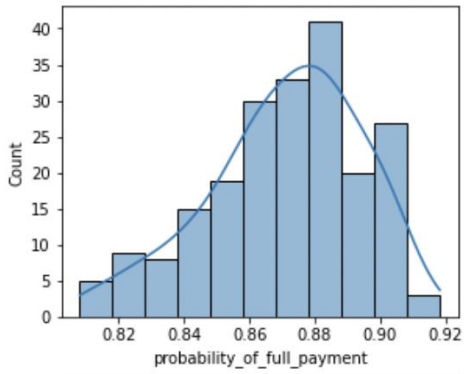
Plotting Box plots and hist plots:

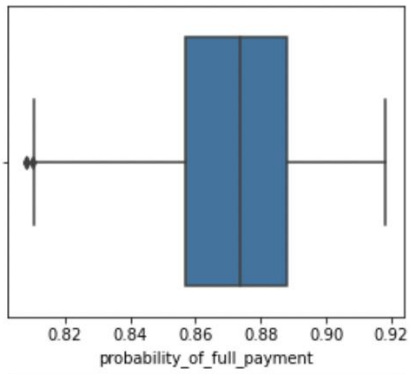


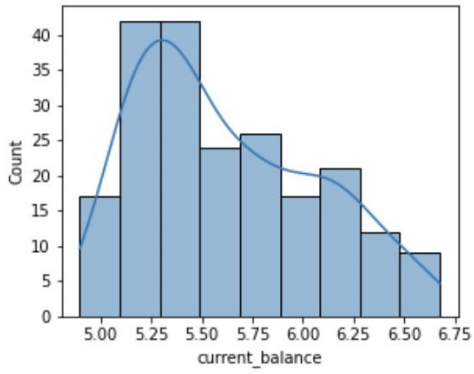


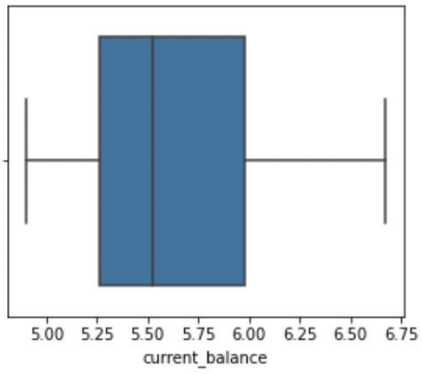


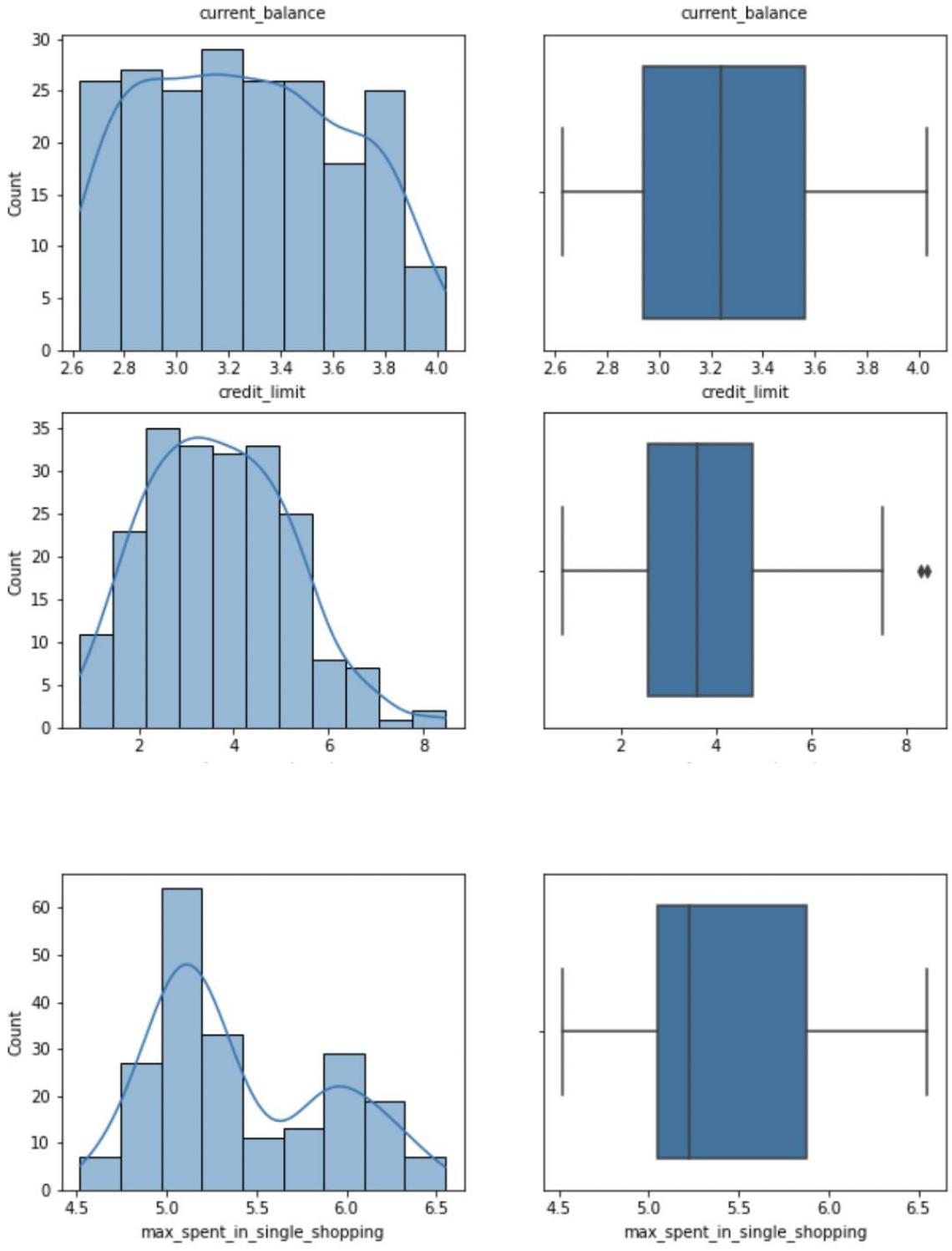












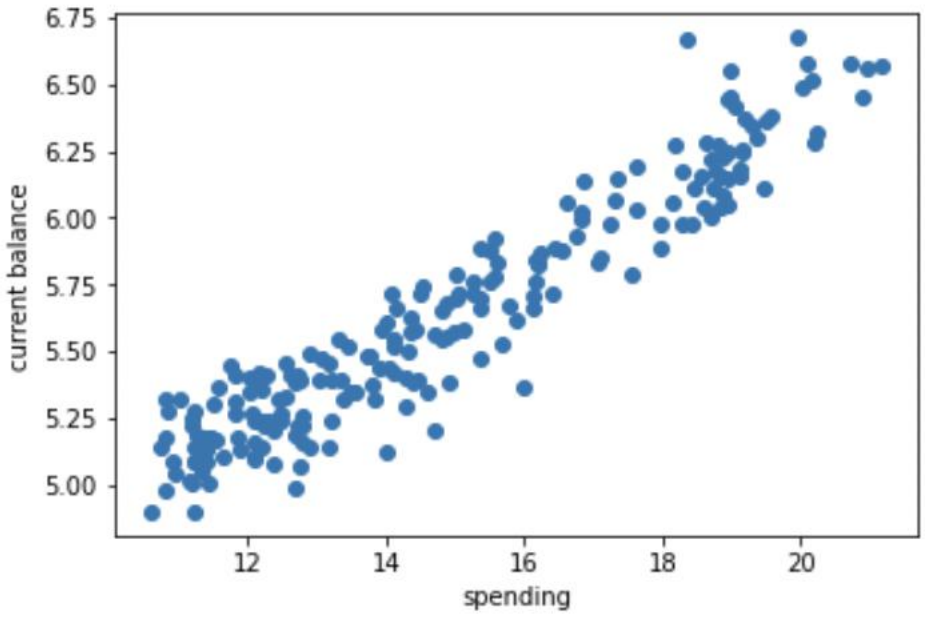
- Both referral_exp_in_years and referral_current_salary have ouliers in upper values.
- Distrubtion is skewed to right tail for all the variable except probability_of_full_payment variable, which has left tail

Checking the skewness values quantitatively:

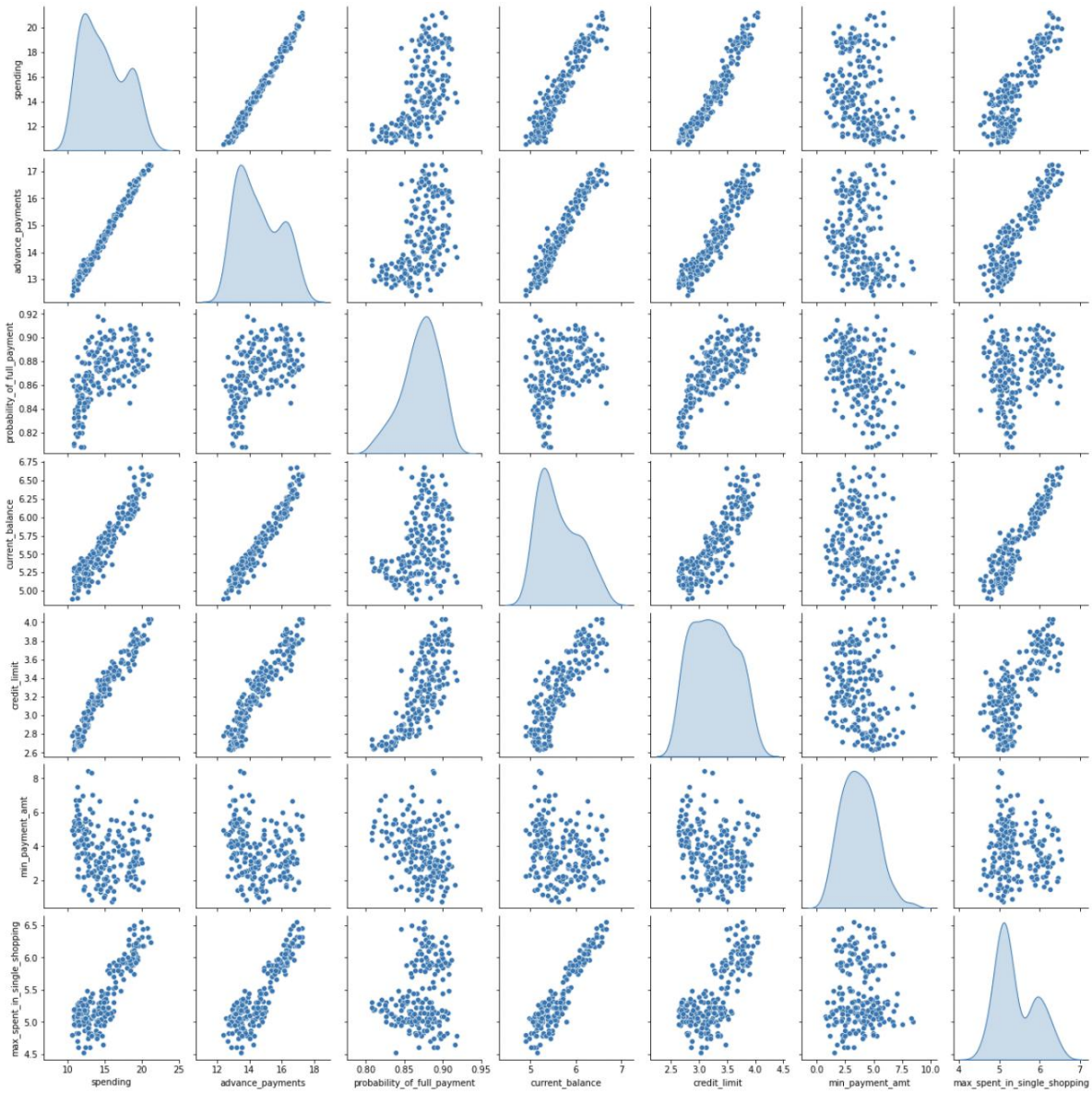
```
max_spent_in_single_shopping     0.561897
current_balance                  0.525482
min_payment_amt                  0.401667
spending                         0.399889
advance_payments                 0.386573
credit_limit                     0.134378
probability_of_full_payment     -0.537954
dtype: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

Multivariate analysis:

checking for multicollinearity:



- From the above plot we see that as the referral experience increases the salary is also increasing showing a
- positive relationship



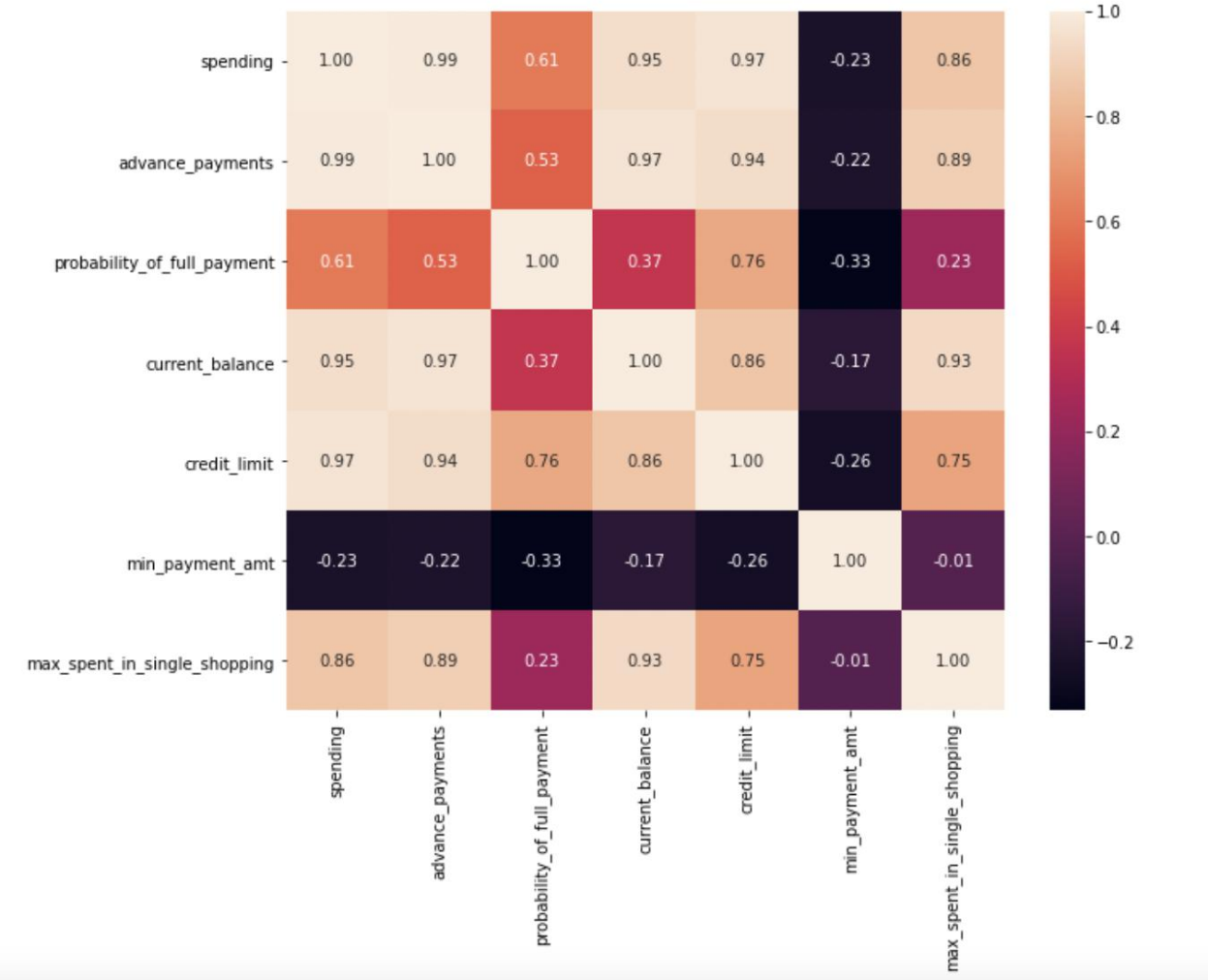
There is a strong positive correlation between the following:

- spending & advance_payments
- advance_payments & current_balance
- credit_limit & spending
- spending & current_balance
- credit_limit & advance_payments
- max_spent_in_single_shopping current_balance

Correlation matrix:

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

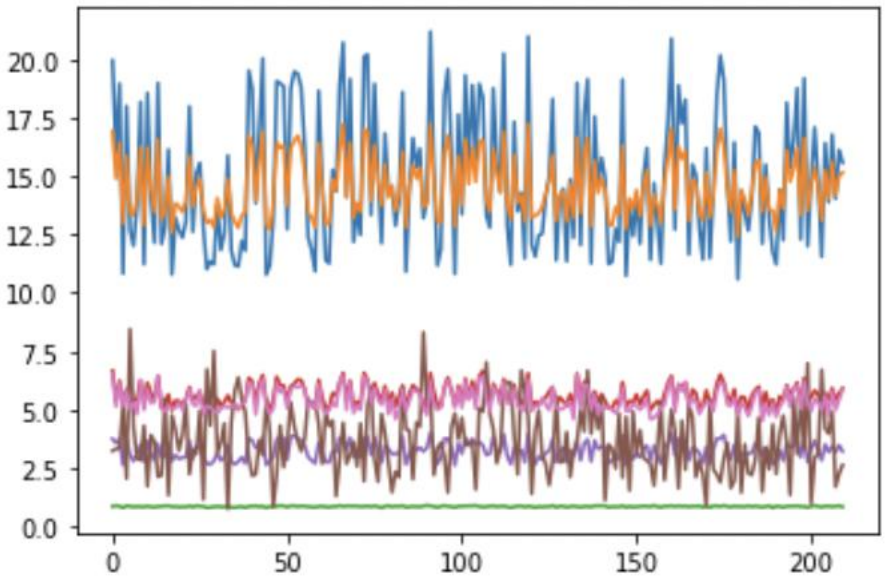
Heatmap:



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

- We can see that the values of the variables are different. Hence, scaling is required.
- Spending, advance_payments are in different values and this may get more weightage.
- After scaling, all the variables will have the same range of values.

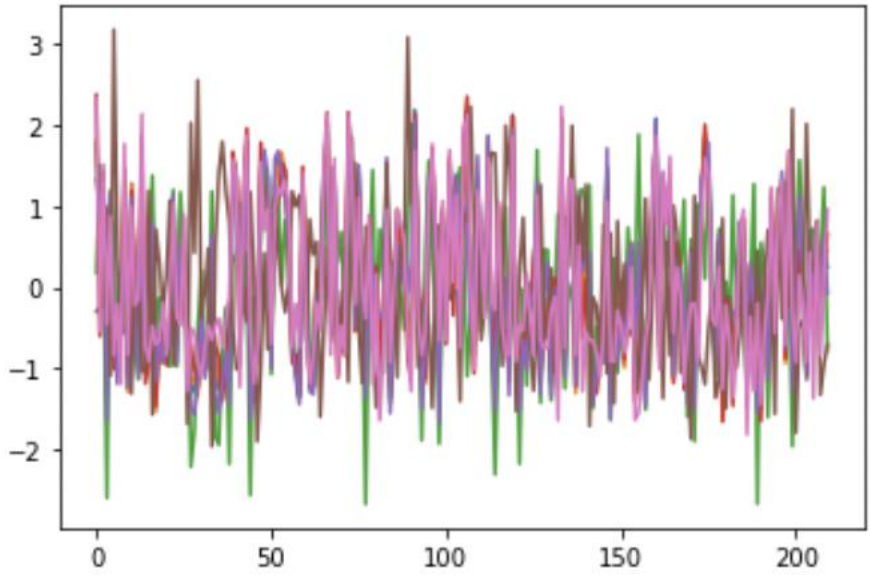
Before Scaling:



Scaled data:

	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

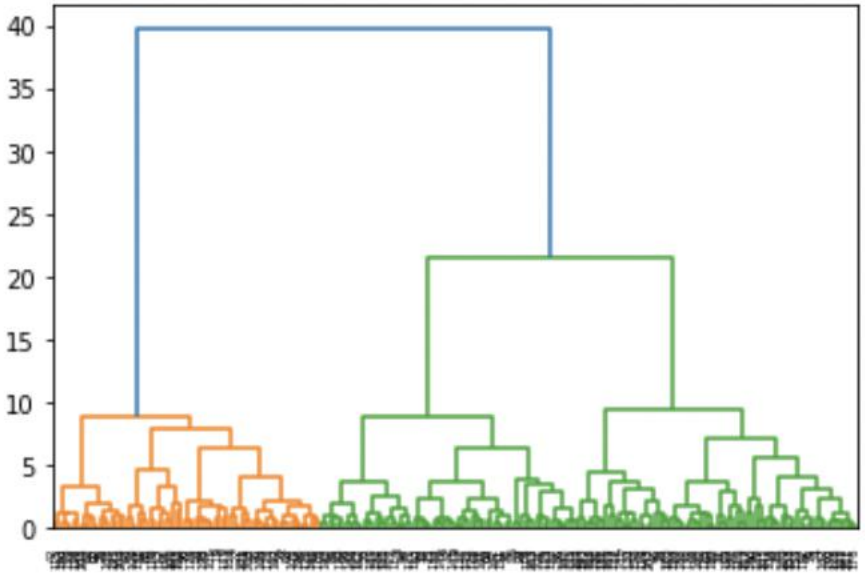
After Scaling:



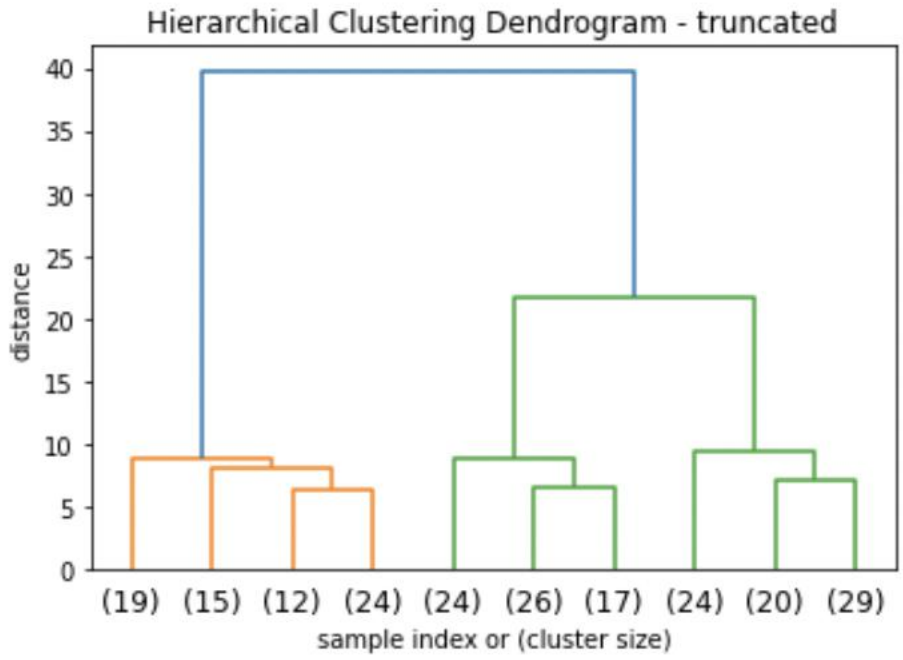
1.3 Apply hierarchical clustering to scaled data. Identify the number of optimum clusters using Dendrogram and briefly describe them

Using ward method:

Dendogram:



Cutting the dendogram with suitable clusters:



To create 3 clusters:

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	H_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

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

H_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

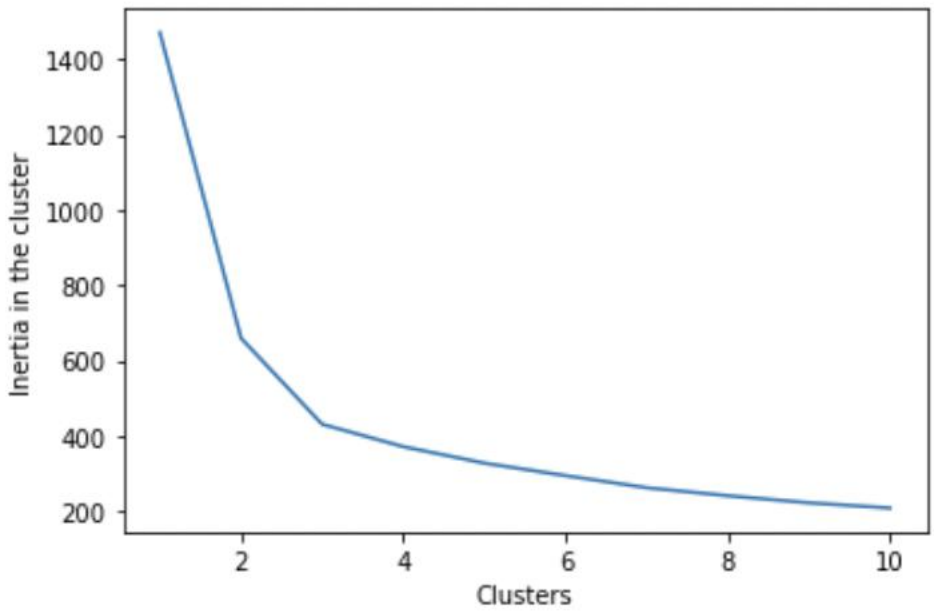
- For cluster grouping based on the dendrogram, 3 groups is good.
- Taking 3 group clusters based on the hierarchical clustering

- Three group cluster solution gives a pattern based on high/medium/low spending with max_spent_in_single_shopping
- (high value item) and probability_of_full_payment(payment made).

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

```
[1469.9999999999995,
 659.1717544870411,
 430.65897315130064,
 371.18461253510196,
 327.4353937624867,
 294.33026069261604,
 262.19460683819483,
 240.9375142500153,
 222.07883941731245,
 208.26405549252087]
```

Elbow curve:



	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Cluster_kmeans
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	0
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	3
4	17.99	15.86	0.8992	5.890	3.694	2.068	5.837	1

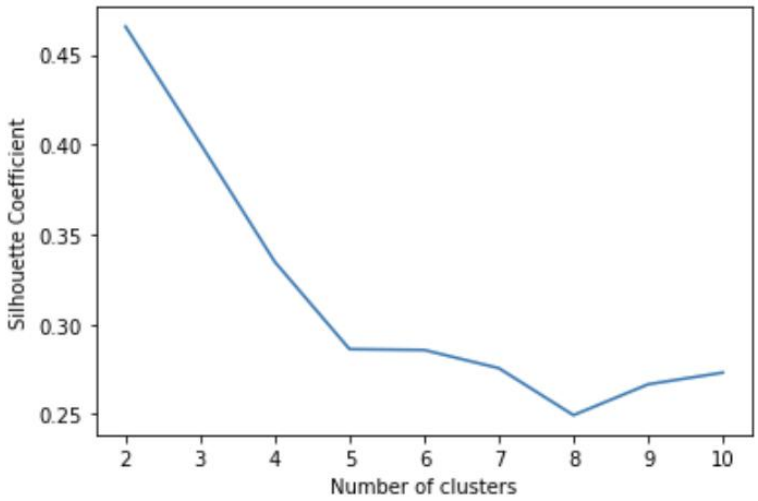
silhouette_score:

0.32757426605518075

Scores:

```
[0.46577247686580914,
 0.40072705527512986,
 0.3347542296283262,
 0.28621461554288646,
 0.285726896652541,
 0.2756098749293962,
 0.2494355873628217,
 0.2666366921192433,
 0.2731288488219916]
```

Plotting the sc scores:



spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Cluster_kmeans	sil_width
19.94	16.92	0.8752	6.675	3.763	3.252	6.550	1	0.445327
15.99	14.89	0.9064	5.363	3.582	3.336	5.144	0	0.049939
18.95	16.42	0.8829	6.248	3.755	3.368	6.148	1	0.443575
10.83	12.96	0.8099	5.278	2.641	5.182	5.185	3	0.532008
17.99	15.86	0.8992	5.890	3.694	2.068	5.837	1	0.081568

Minimum value:

-0.05384082699360069

3 - Cluster:

```
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], dtype=int32)
```

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

K-means clustering and cluster information:

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

Using 3 clusters via kmeans. Based on the dataset given, 3 cluster solution is good to use based on the spending pattern (High, Medium, Low)

Transposing the cluster:

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

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

3 cluster via hierarchical clustering:

H_clusters	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

Group 1 : High Spending

Group 3 : Medium Spending

Group 2 : Low Spending

Group 1:

- Giving any reward points might increase their purchases.
- Maximum max_spent_in_single_shopping is high for this group, so can be offered discount/offer on next transactions upon full payment
- Increase there credit limit and Increase spending habits
- Give loan against the credit card, as they are customers with good repayment record.
- Tie up with luxary brands, which will drive more one_time_maximun spending

Group 3:

- They are potential target customers who are paying bills and doing purchases and maintaining comparatively good credit score. So we can increase credit limit or can lower down interest rate.
- Promote premium cards/loyality cars to increase transcations.
- Increase spending habits by trying with premium ecommerce sites, travel portal, travel airlines/hotel, as this will encourge them to spend more

Group 2:

- customers should be given remainders for payments. Offers can be provided on early payments to improve their payment rate.
- Increase there spending habits by tieing up with grocery stores, utlities (electircity, phone, gas, others)

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 and compare the models' performances in train and test sets.

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

Checking first five rows of the dataset:

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

```
<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
```

Below are the observations:

10 variables Age, Commision, Duration, Sales are numeric variable, rest are categorial variables 3000 records 9 independant variable and one target variable - Clamied

Checking for missing values:

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

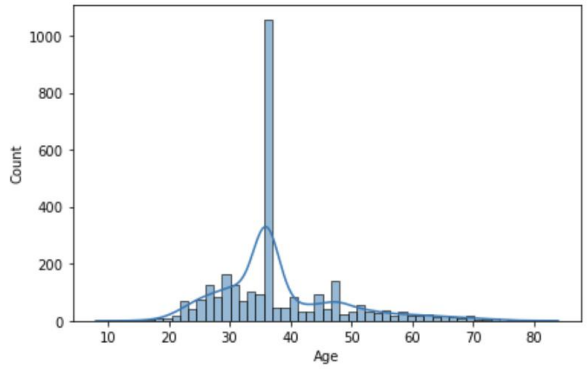
- There are no missing values
- There are 139 duplicate records, but it can be of different customers, There is no customer ID or any unique identifier, so no need to drop duplicates

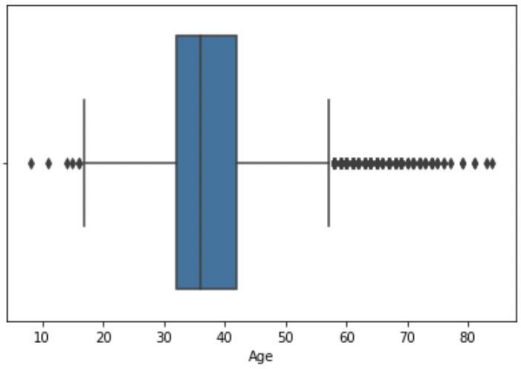
Checking Descriptive Statistics:

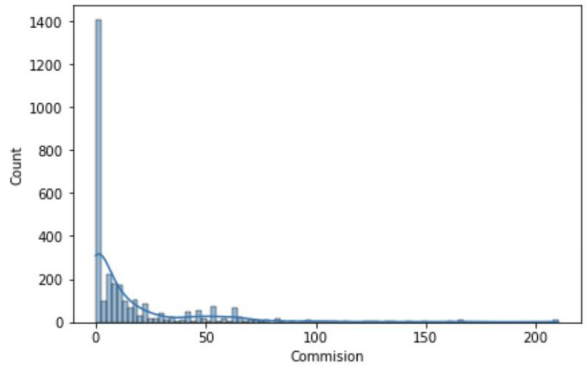
	Age	Commision	Duration	Sales
count	3000.000000	3000.000000	3000.000000	3000.000000
mean	38.091000	14.529203	70.001333	60.249913
std	10.463518	25.481455	134.053313	70.733954
min	8.000000	0.000000	-1.000000	0.000000
25%	32.000000	0.000000	11.000000	20.000000
50%	36.000000	4.630000	26.500000	33.000000
75%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

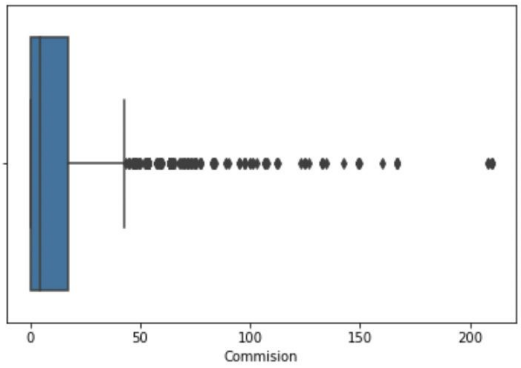
- Duration column has negative value.
- Technically its not possible mean and median varies significantly for commission

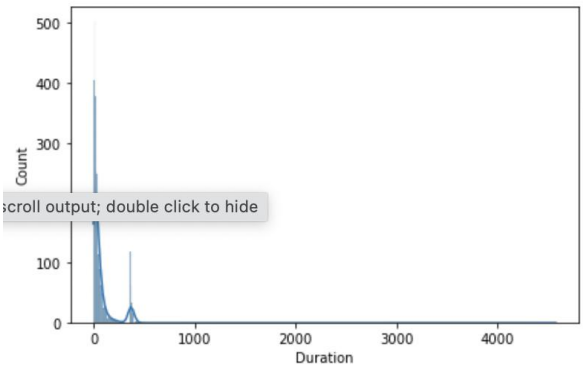
Plotting boxplots and histplots:

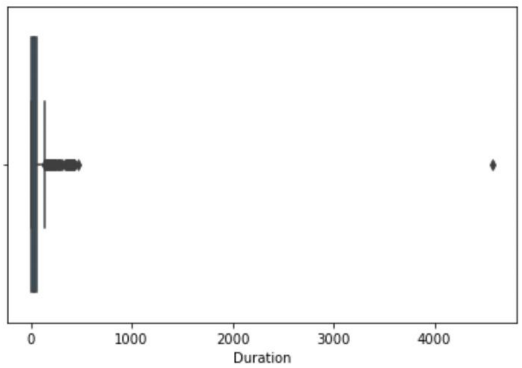


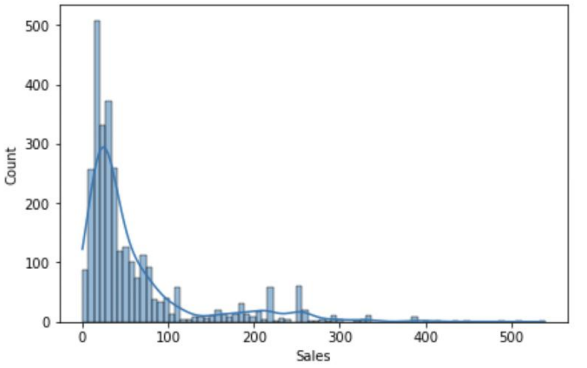


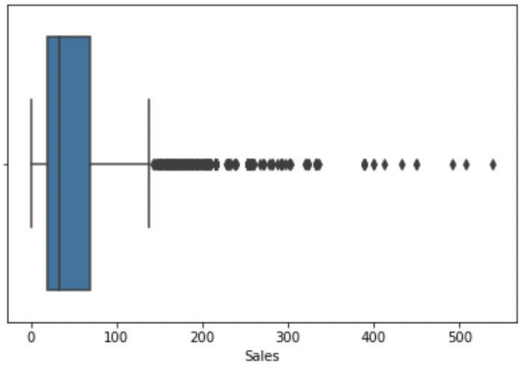




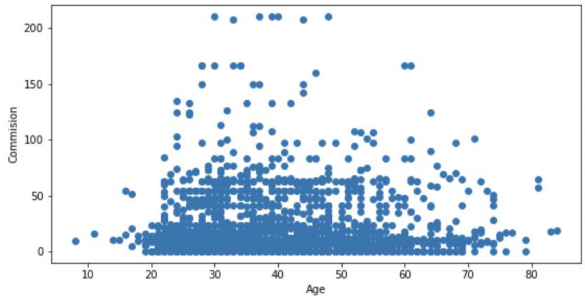


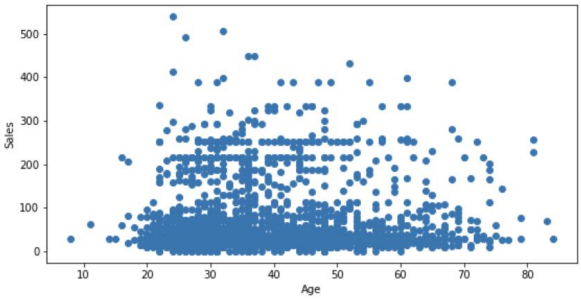


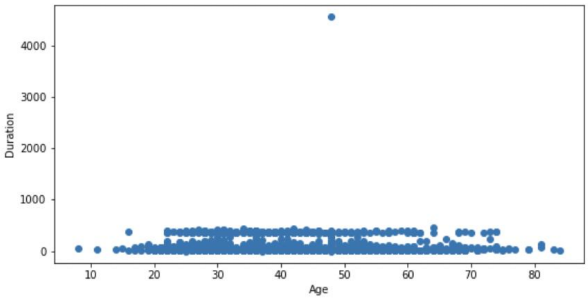




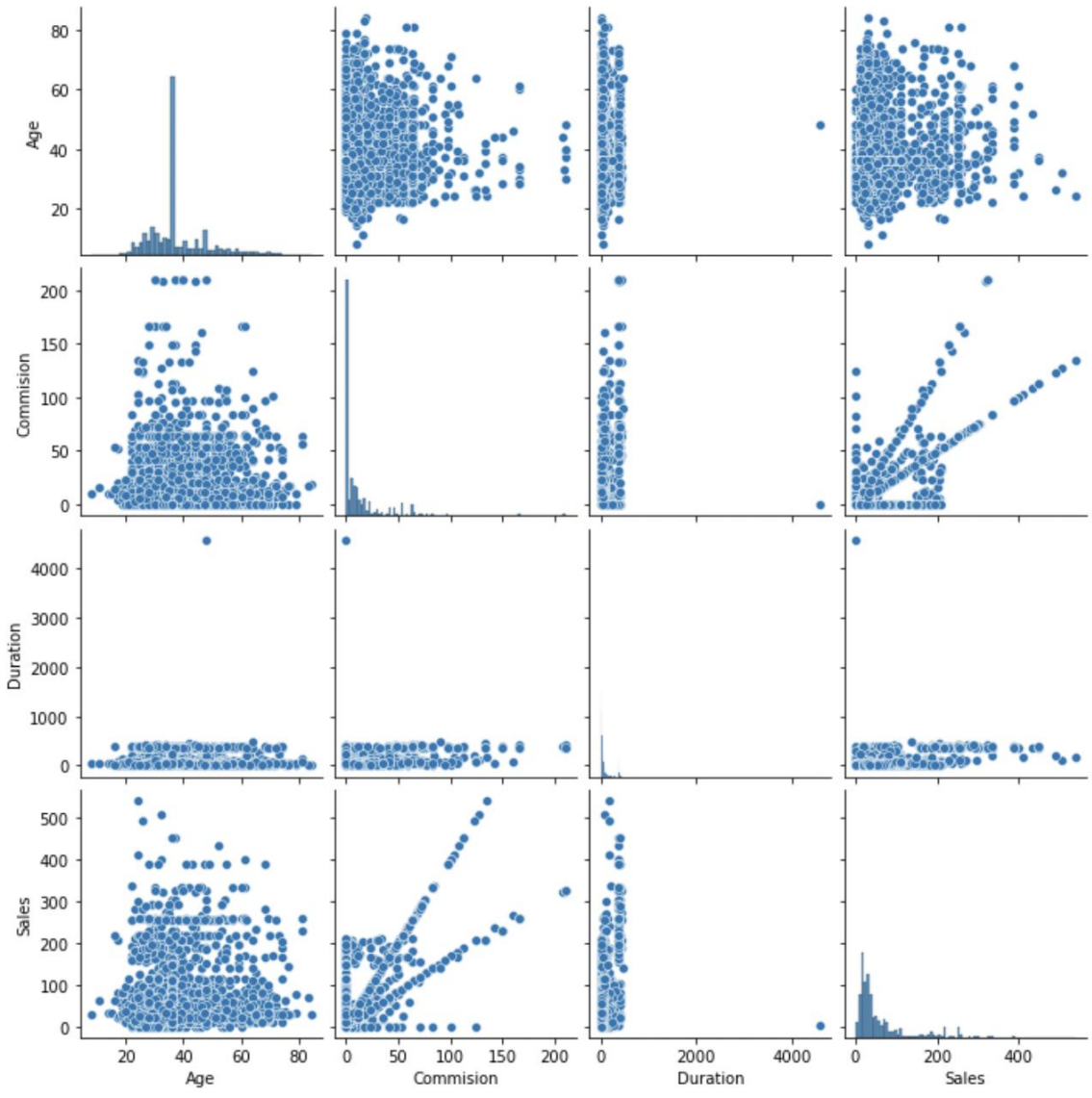
Plotting scatter plot:



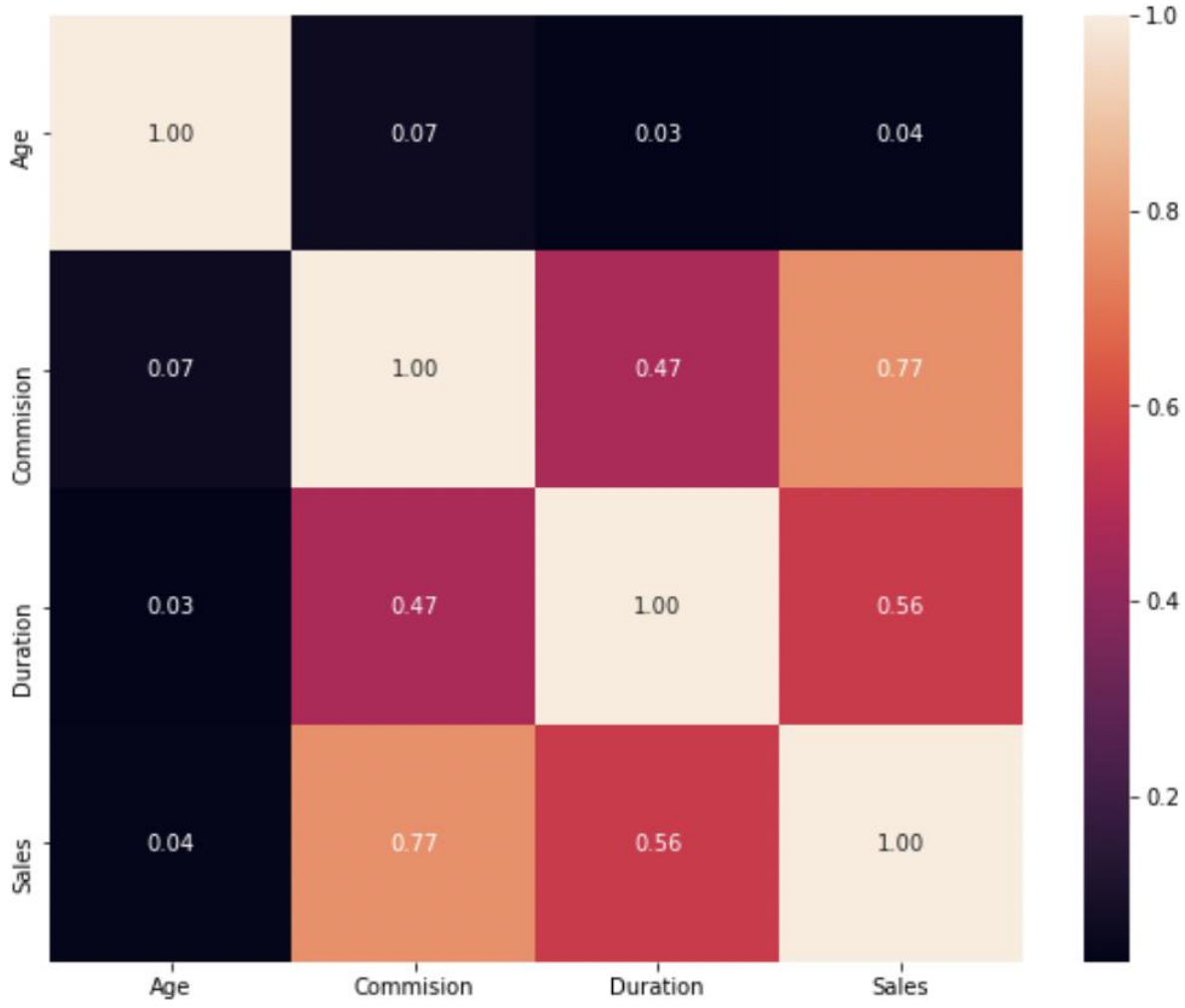




Checking pairwise distribution of the continuous variables:



Checking for correlations:



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

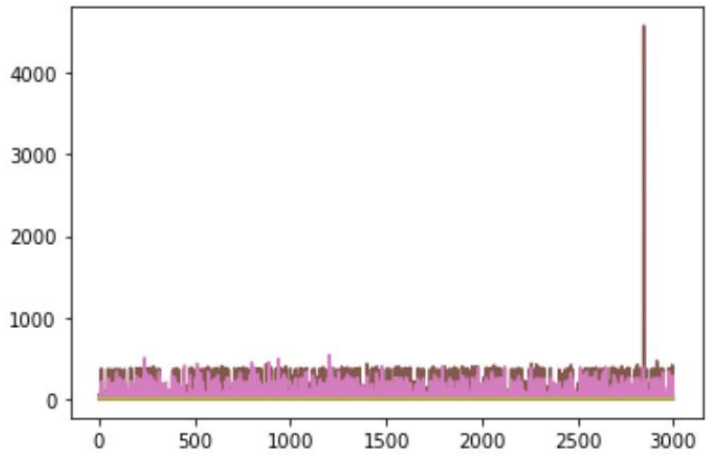
Converting all objects to categorical codes:

```
<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
```

Dropping Claimed column:

	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

Before Scaling:

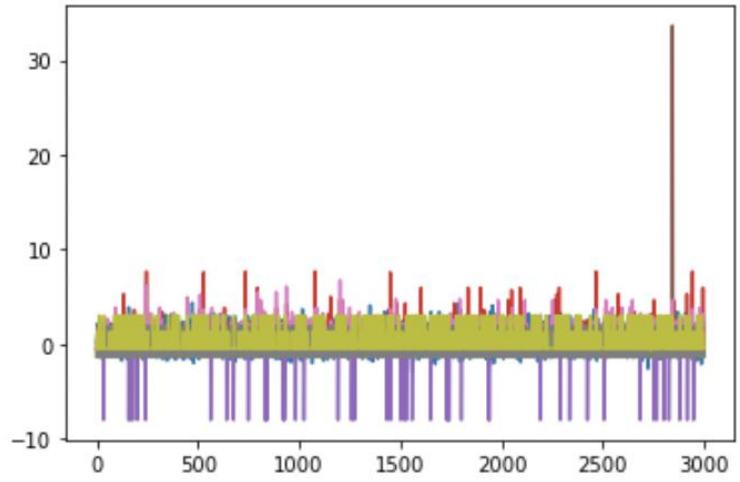


Performing Scaling using zscore:

Scaled data:

	Age	Agency_Code	Type	Commision	Channel	Duration	Sales	Product Name	Destination
0	0.947162	-1.314358	-1.256796	-0.542807	0.124788	-0.470051	-0.816433	0.268835	-0.434646
1	-0.199870	0.697928	0.795674	-0.570282	0.124788	-0.268605	-0.569127	0.268835	-0.434646
2	0.086888	-0.308215	0.795674	-0.337133	0.124788	-0.499894	-0.711940	0.268835	1.303937
3	-0.199870	0.697928	0.795674	-0.570282	0.124788	-0.492433	-0.484288	-0.525751	-0.434646
4	-0.486629	1.704071	-1.256796	-0.323003	0.124788	-0.126846	-0.597407	-1.320338	-0.434646

After Scaling:



Splitting data into training and test set:

Checking the dimensions of training and test data:

X_train (2100, 9)

X_test (900, 9)

train_labels (2100,)

test_labels (900,)

Build classification model CART - Decision Tree Classifier:

	Imp
Age	0.177894
Agency_Code	0.194770
Type	0.000383
Commision	0.095127
Channel	0.007262
Duration	0.262122
Sales	0.199864
Product Name	0.043258
Destination	0.019321

Regularize the Decision Tree and Check the train and test score after regularization:

```
Fitting 5 folds for each of 6498 candidates, totalling 32490 fits

GridSearchCV(estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [5, 6, 7, 9, 10, 11, 13, 15, 16],
                         'min_samples_leaf': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                                              12, 13, 14, 15, 16, 17, 18, 19],
                         'min_samples_split': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                                               11, 12, 13, 14, 15, 16, 17, 18,
                                               19]},
             verbose=1)
```

```
{'criterion': 'entropy',
 'max_depth': 5,
 'min_samples_leaf': 6,
 'min_samples_split': 18}
```

Dt model feature importance:

	Imp
Age	0.177894
Agency_Code	0.194770
Type	0.000383
Commision	0.095127
Channel	0.007262
Duration	0.262122
Sales	0.199864
Product Name	0.043258
Destination	0.019321

Reg dt model feature importance:

	Imp
Age	0.052149
Agency_Code	0.541522
Type	0.000000
Commision	0.027741
Channel	0.000000
Duration	0.045172
Sales	0.265092
Product Name	0.068324
Destination	0.000000

```
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 150}

DecisionTreeClassifier(max_depth=10, min_samples_leaf=50, min_samples_split=150,
                       random_state=1)
```

```
{'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf': 20, 'min_samples_split': 150}

DecisionTreeClassifier(max_depth=7, min_samples_leaf=20, min_samples_split=150,
                       random_state=1)
```

```
{'criterion': 'gini', 'max_depth': 4.0, 'min_samples_leaf': 46, 'min_samples_split': 280}

DecisionTreeClassifier(max_depth=4.0, min_samples_leaf=46,
                       min_samples_split=280, random_state=1)
```

```
{'criterion': 'gini', 'max_depth': 5.0, 'min_samples_leaf': 42, 'min_samples_split': 200}

DecisionTreeClassifier(max_depth=5.0, min_samples_leaf=42,
                       min_samples_split=200, random_state=1)
```

Feature importance:

	Imp
Age	0.026631
Agency_Code	0.604561
Type	0.000000
Commision	0.022472
Channel	0.000000
Duration	0.036387
Sales	0.254851
Product Name	0.055097
Destination	0.000000

Probability:

	0	1
0	0.935714	0.064286
1	0.394089	0.605911
2	0.394089	0.605911
3	0.311111	0.688889
4	0.927586	0.072414

Build Random Forest with n_estimators = 501, random_state=1 and do GridSearchCV with the following parameters.

'criterion' : ["gini", "entropy"]

'max_depth': [5, 7],

'max_features': [4, 6],

'min_samples_leaf': [50, 100],

'min_samples_split': [20, 50],

'n_estimators': [301, 501]

cv = 3

```
{'max_depth': 5, 'max_features': 4, 'min_samples_leaf': 50, 'min_samples_split': 20, 'n_estimators': 501}
```

```
RandomForestClassifier(max_depth=5, max_features=4, min_samples_leaf=50,
                       min_samples_split=20, n_estimators=501, random_state=0)
```

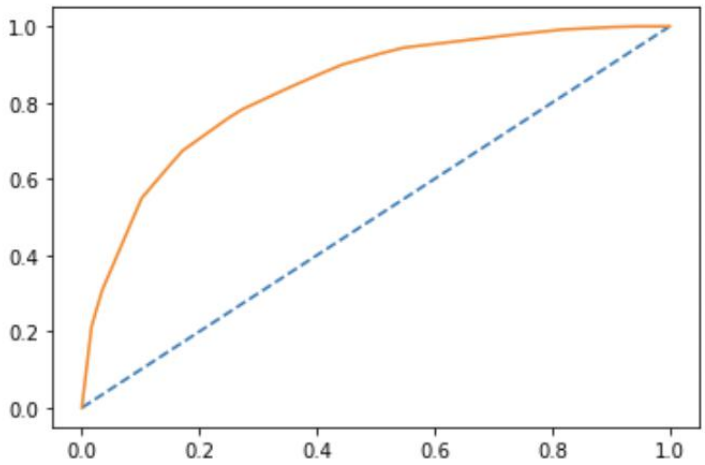
	0	1
0	0.714285	0.285715
1	0.554773	0.445227
2	0.594591	0.405409
3	0.302450	0.697550
4	0.922878	0.077122

	Imp
Age	0.025308
Agency_Code	0.356866
Type	0.059450
Commision	0.126849
Channel	0.000000
Duration	0.040475
Sales	0.150499
Product Name	0.235066
Destination	0.005486

2.3 Performance Metrics: Comment and Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, classification reports for each model.

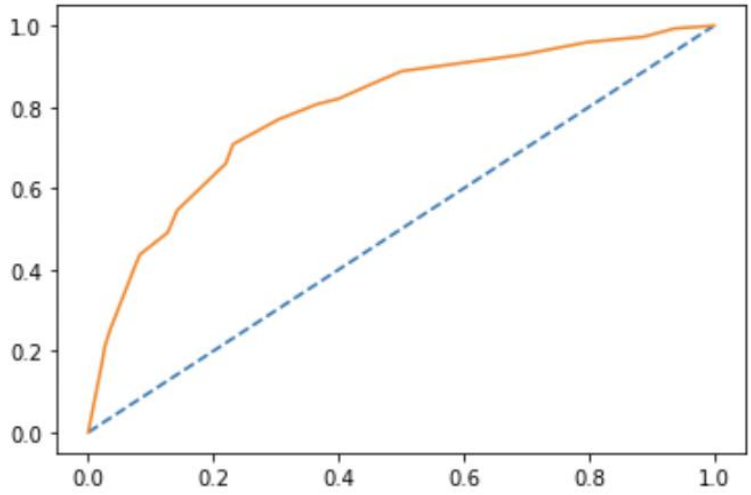
CART - AUC and ROC for the training data:

```
AUC: 0.835
```



CART - AUC and ROC for the test data:

AUC: 0.792



Train accuracy (CART model)

0.7938095238095239

Test Accuracy (CART model)

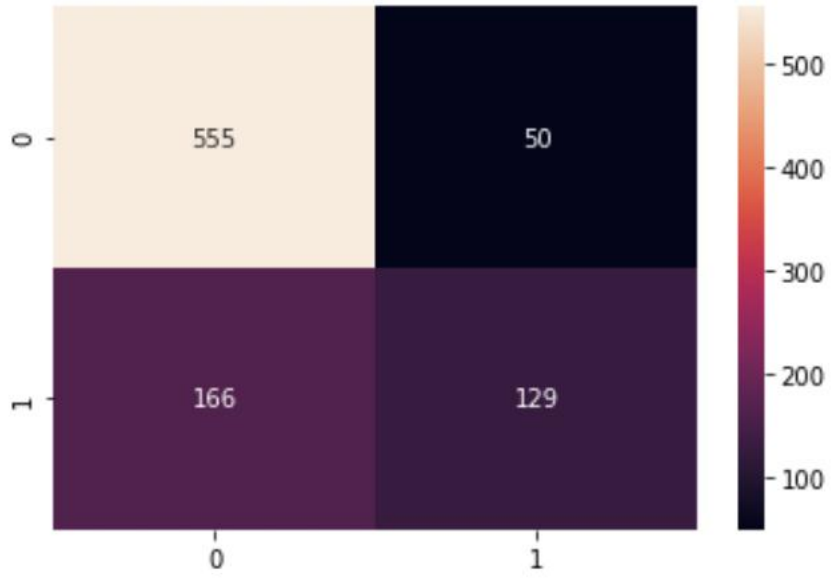
0.76

Confusion matrix for training data (CART model)

array([[1321, 150],
[283, 346]])

confusion matrix for test data (CART model)

array([[555, 50],
       [166, 129]])



Classification Report for train data (CART model)

	precision	recall	f1-score	support
0	0.82	0.90	0.86	1471
1	0.70	0.55	0.62	629
accuracy			0.79	2100
macro avg	0.76	0.72	0.74	2100
weighted avg	0.79	0.79	0.79	2100

Classification Report for test data (CART model)

	precision	recall	f1-score	support
0	0.77	0.92	0.84	605
1	0.72	0.44	0.54	295
accuracy			0.76	900
macro avg	0.75	0.68	0.69	900
weighted avg	0.75	0.76	0.74	900

```
cart_train_precision  0.7
cart_train_recall  0.55
cart_train_f1  0.62
```

```
cart_test_precision  0.72
cart_test_recall  0.44
cart_test_f1  0.54
```

CART Train data:

AUC: 83%

Accuracy: 79%

Precision: 70%

f1-Score: 62%

CART Test data:

AUC: 79%

Accuracy: 76%

Precision: 72%

f1-Score: 54%

Training and Test set results are almost similar with high values

For Random Forest model:

Train data:

Confusion matrix:

```
array([[1326,  145],
       [ 281,  348]])
```

Accuracy: 0.7971428571428572

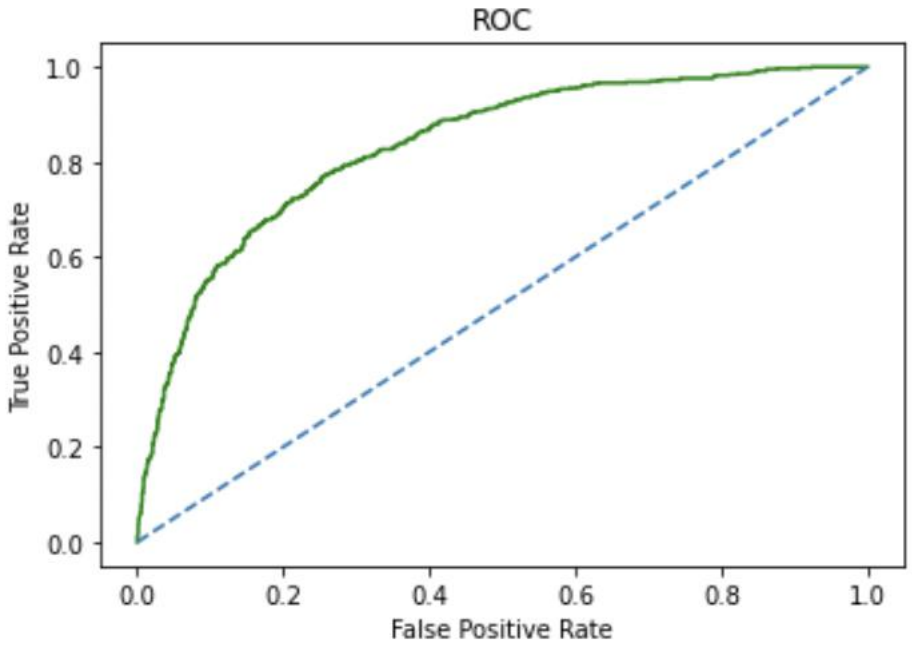
Classification report:

	precision	recall	f1-score	support
0	0.83	0.90	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

```
rf_train_precision  0.71
rf_train_recall  0.55
rf_train_f1  0.62
```

ROC Curve:

```
Area under Curve is 0.8344998535545183
```



Test data:

Confusion matrix:

```
array([[552,  53],
       [162, 133]])
```

Accuracy: 0.7611111111111111

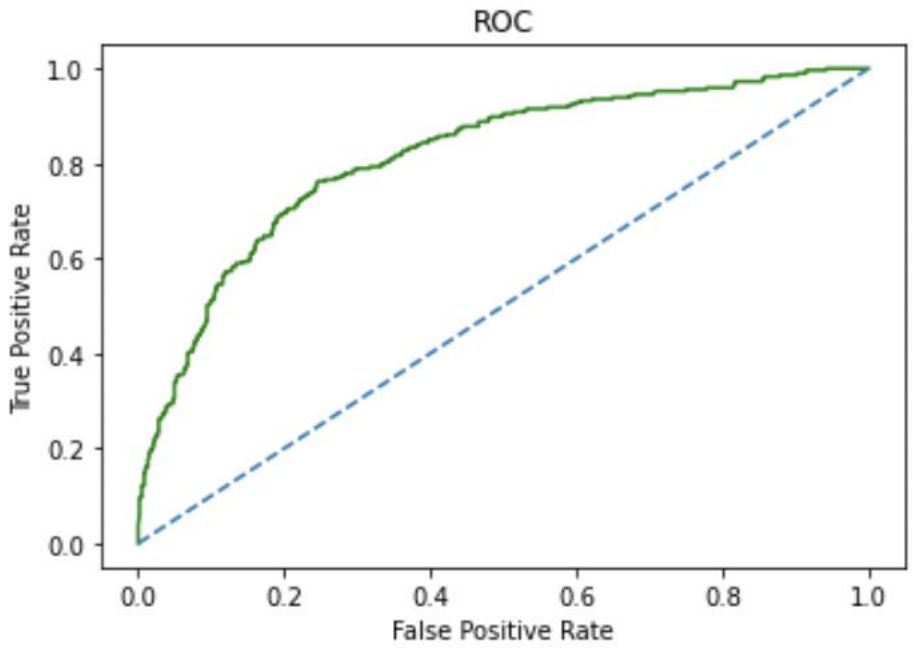
Classification report:

	precision	recall	f1-score	support
0	0.77	0.91	0.84	605
1	0.72	0.45	0.55	295
accuracy			0.76	900
macro avg	0.74	0.68	0.70	900
weighted avg	0.75	0.76	0.74	900

```
rf_test_precision  0.72
rf_test_recall  0.45
rf_test_f1  0.55
```

ROC Curve:

```
Area under Curve is 0.814245692674044
```



RF train data:

AUC: 83%

Accuracy: 80%

Precision: 71%

f1-Score: 62

RF test data:

AUC: 81%

Accuracy: 76%

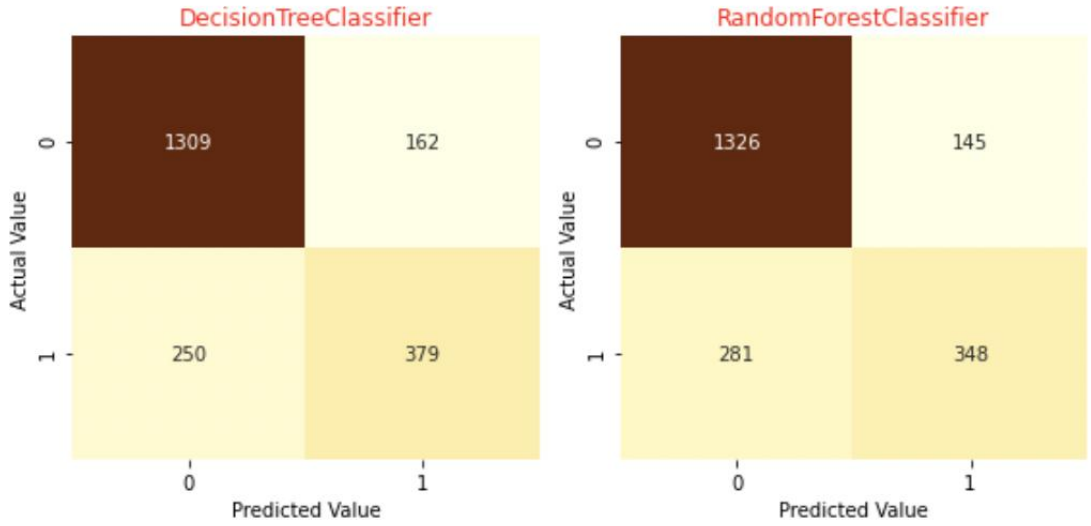
Precision: 72%

f1-Score: 55

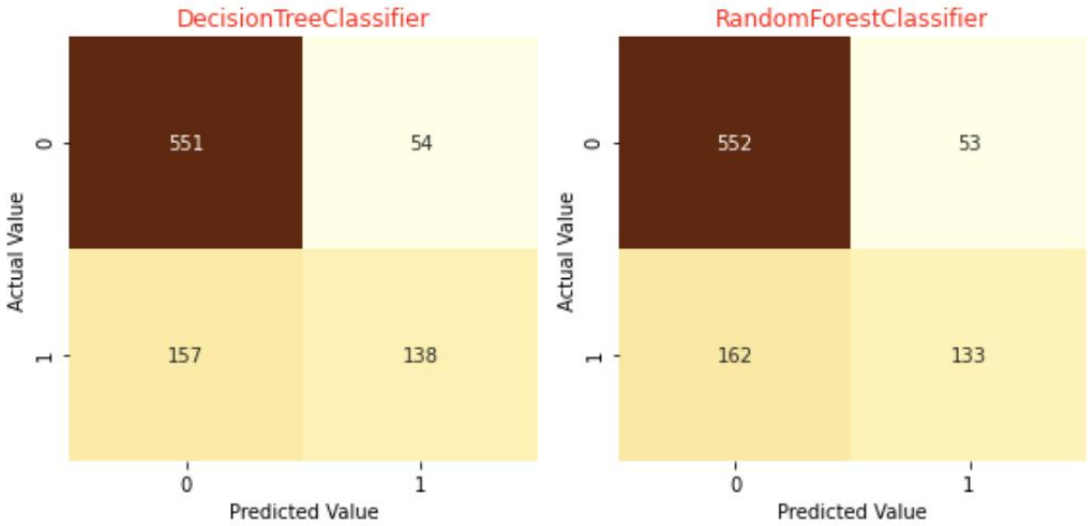
Train and test data also have similar values for random forest model

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

Comparing Confusion Matrices from All the models for the Train Set. [Plot/Print the confusion matrices for all the training sets].



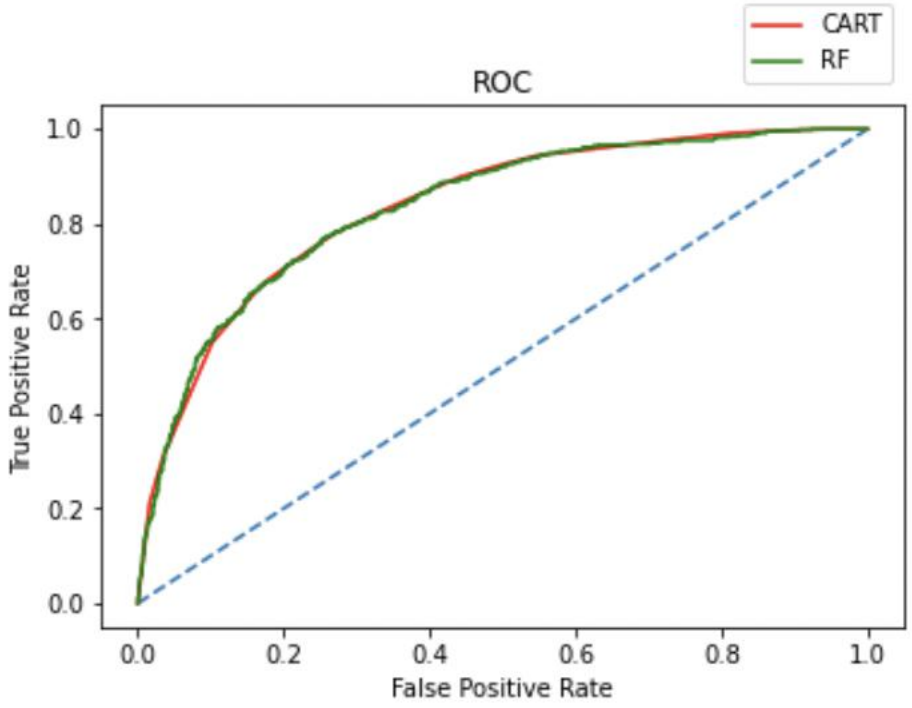
Comparing Confusion Matrices from All the models for the Test Set. [Plot/Print the confusion matrices for all the test sets].



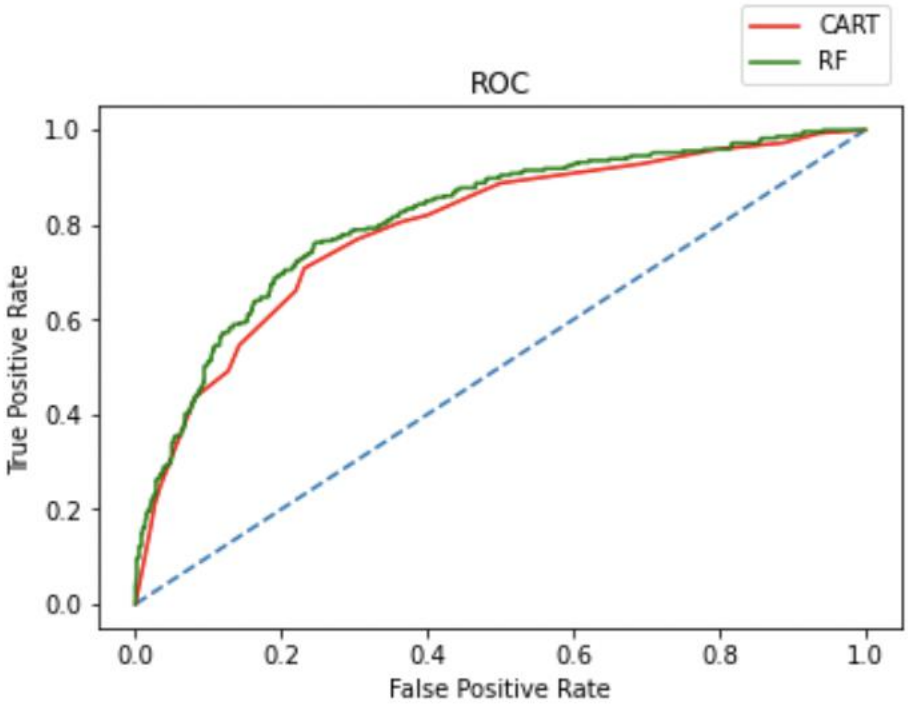
Comparison of the performance metrics from the 3 models

	CART Train	CART Test	Random Forest Train	Random Forest Test
Accuracy	0.79	0.76	0.80	0.76
AUC	0.84	0.79	0.83	0.81
Recall	0.55	0.44	0.55	0.45
Precision	0.70	0.72	0.71	0.72
F1 Score	0.62	0.54	0.62	0.55

ROC Curve for the 2 models on the Training data



ROC Curve for the 2 models on the Test data



RF model is best optimized, as it has little better accuracy, precision, recall, f1 score better than CART model.

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

If there is a possibility of more real time data or already processed data, then we can have a better understanding

This is understood by looking at the insurance data by drawing relations between different variables such as day of the incident, time, age group, and associating it with other external information such as location, behavior patterns, weather information, airline/vehicle types, etc.

• Streamlining online experiences benefitted customers, leading to an increase in conversions, which subsequently raised profits.

• As per the data, 90% of insurance is done by online channel.

• Other interesting fact, is almost all the offline business has a claimed associated, need to find why?

• Also based on the model we are getting 80%accuracy, so we need customer books airline tickets or plans, cross sell the insurance based on the claim data pattern.

Key performance indicators (KPI) The KPI's of insurance claims are:

• Reduce claims cycle time

• Increase customer satisfaction

• Combat fraud

• Optimize claims recovery

• Reduce claim handling costs Insights gained from data and AI-powered analytics could expand the boundaries of insurability, extend existing products, and give rise to new risk transfer solutions in areas like a non-damage business interruption and reputational damage.