greatlearning

Data Mining Project

Data Science and Business Analytics (PGP-DSBA)
Online September_A 2021-22

Anmol Tripathi 1/23/22

Table of Contents:

Problem 1:

Problem 2:

List of Tables:

List of Graph:

List of Formulas:

Problem 1:

Executive Summary:

The given data set has number columns and rows in them. The given data provides us the information with regards to various people's usage on the credit card and their expenditure. Clustering or cluster analysis is a machine learning technique, which groups the unlabeled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group.

Background of the problem:

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 Description:

- The shape of the data set seems to be with 210 rows and 7 columns.

```
orginal_dataset.shape
(210, 7)
```

Table 1.1 Shape and size

- All the columns are float type.
- There are no duplicate values present in the data set.
- Also, the entire data set does not have any null or missing values.

```
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
```

Table 1.2 No null Values

```
<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
```

Table 1.3 Data type

Sample Data Set:

	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 1.4 Data Set

- The above table represents the actual representation of the data set.

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

General Data Set:

	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 1.5 Data Set

Null Information:

```
<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
```

Table 1.6 Checking the null values

7 variables and 210 records. No missing record based on initial analysis. All the variables numeric type.

Data Type:

```
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
```

Table 1.7 Count of the Null Values

- No missing value.

Univariate Analysis:

	count	mean	std	min	25%	50%	75%	90%	max
spending	210.0	14.847524	2.909699	10.5900	12.27000	14.35500	17.305000	18.9880	21.1800
advance_payments	210.0	14.559286	1.305959	12.4100	13.45000	14.32000	15.715000	16.4540	17.2500
probability_of_full_payment	210.0	0.870999	0.023629	0.8081	0.85690	0.87345	0.887775	0.8993	0.9183
current_balance	210.0	5.628533	0.443063	4.8990	5.26225	5.52350	5.979750	6.2733	6.6750
credit_limit	210.0	3.258605	0.377714	2.6300	2.94400	3.23700	3.561750	3.7865	4.0330
min_payment_amt	210.0	3.700201	1.503557	0.7651	2.56150	3.59900	4.768750	5.5376	8.4560
max_spent_in_single_shopping	210.0	5.408071	0.491480	4.5190	5.04500	5.22300	5.877000	6.1850	6.5500

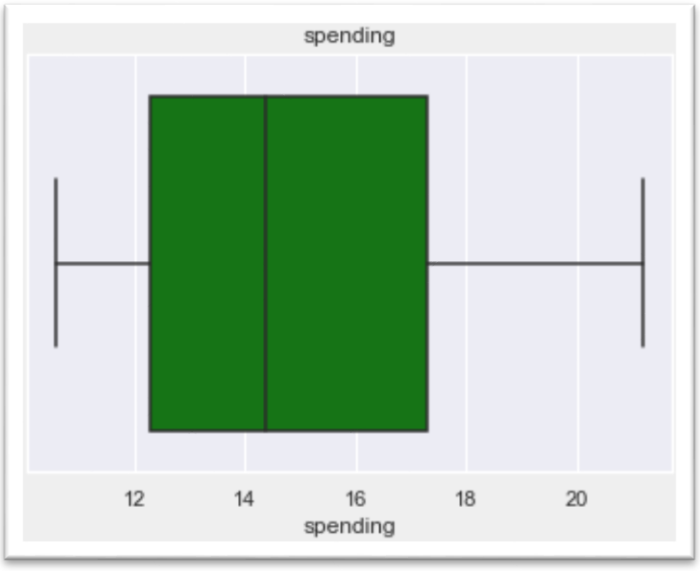
Table 1.8 Initial descriptive analysis of the data

- Based on summary descriptive, the data looks good.
- We see for most of the variable, mean/medium are nearly equal
- Include a 90% to see variations and it looks distributed evenly
- Std Deviation is high for spending variable

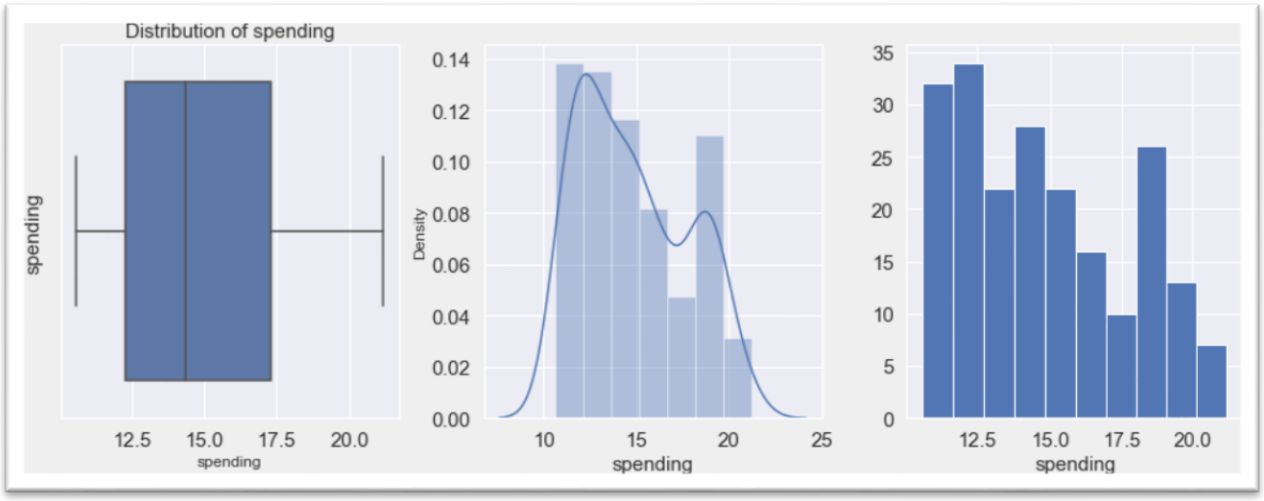
i) Spending Variable:

Important Details:

- Range of values: 10.59
- Minimum spending: 10.59
- Maximum spending: 21.18
- Mean value: 14.847523809523818
- Median value: 14.355
- Standard deviation: 2.909699430687361
- Null values: False
- spending - 1st Quartile (Q1) is: 12.27
- spending - 3st Quartile (Q3) is: 17.305
- Interquartile range (IQR) of spending is: 5.035
- Lower outliers in spending: 4.717499999999999
- Upper outliers in spending: 24.8575
- Number of outliers in spending upper: 0
- Number of outliers in spending lower: 0
- % of Outlier in spending upper: 0 %
- % of Outlier in spending lower: 0 %



Graph 1.1 Box Plot



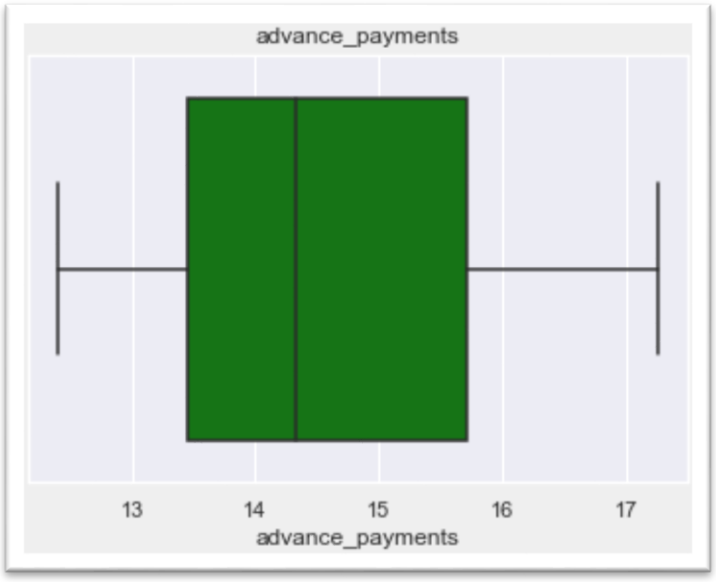
Graph 1.2 Box plot || Dist-Plot || Histogram

ii) advance_payments variable:

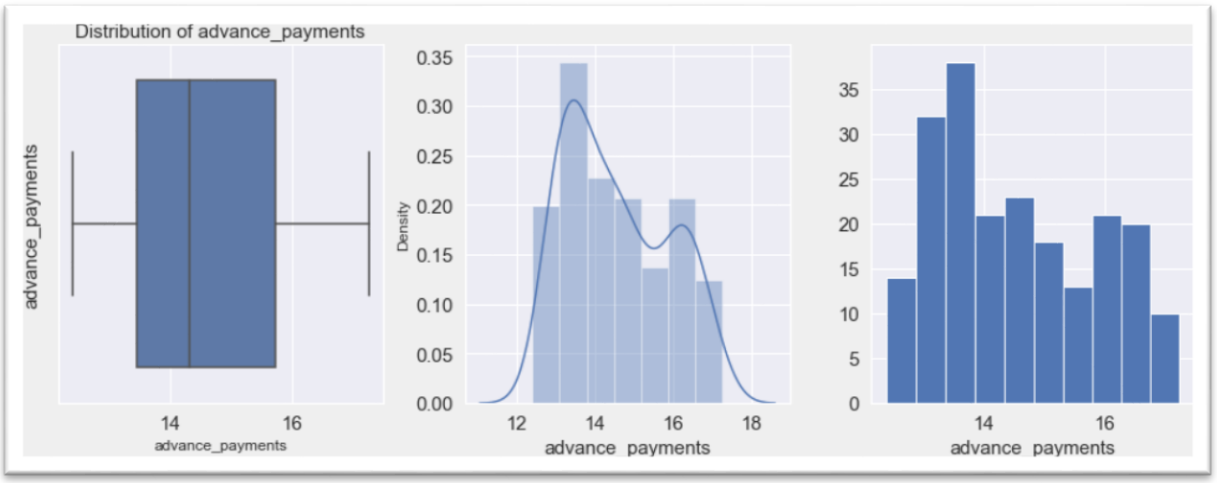
Important Details:

- Range of values: 4.84
- Minimum advance_payments: 12.41
- Maximum advance_payments: 17.25
- Mean value: 14.559285714285727
- Median value: 14.32
- Standard deviation: 1.305958726564022
- Null values: False
- advance_payments - 1st Quartile (Q1) is: 13.45

- advance_payments - 3st Quartile (Q3) is: 15.715
- Interquartile range (IQR) of advance_payments is: 2.2650000000000006
- Lower outliers in advance_payments: 10.052499999999998
- Upper outliers in advance_payments: 19.1125
- Number of outliers in advance_payments upper : 0
- Number of outliers in advance_payments lower : 0
- % of Outlier in advance_payments upper: 0 %
- % of Outlier in advance_payments lower: 0 %



Graph 1.3 Box Plot

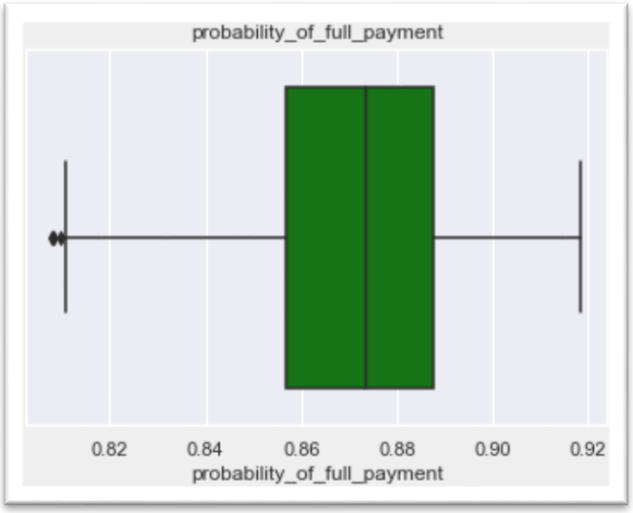


Graph 1.4 Box plot || Dist-Plot || Histogram

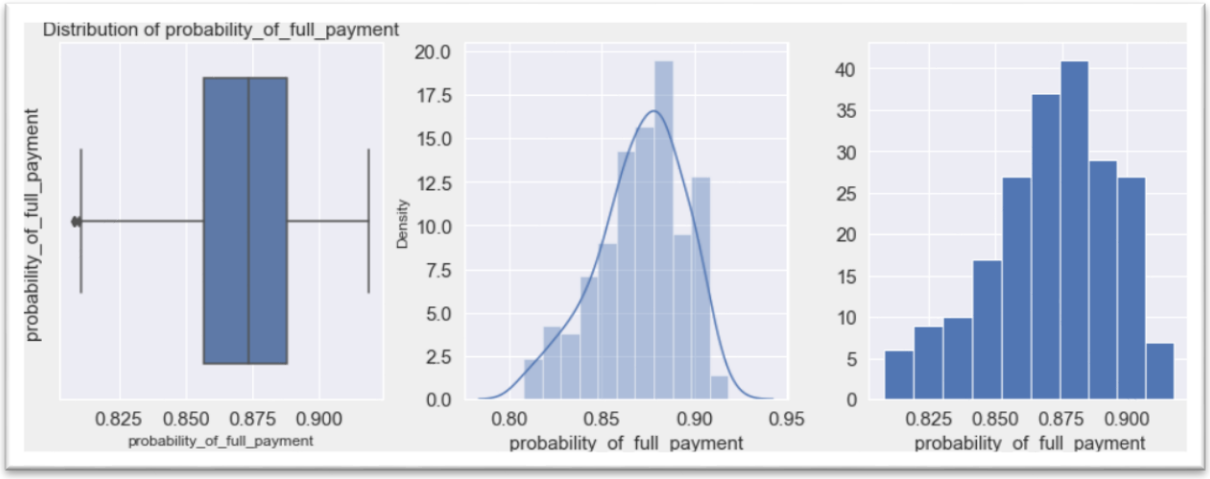
iii) probability_of_full_payment variable:

Important Details:

- Range of values: 0.11019999999999996
- Minimum probability_of_full_payment: 0.8081
- Maximum probability_of_full_payment: 0.9183
- Mean value: 0.8709985714285714
- Median value: 0.8734500000000001
- Standard deviation: 0.0236294165838465
- Null values: False
- probability_of_full_payment - 1st Quartile (Q1) is: 0.8569
- probability_of_full_payment - 3st Quartile (Q3) is: 0.887775
- Interquartile range (IQR) of probability_of_full_payment is: 0.030874999999999986



Graph 1.5 Box Plot



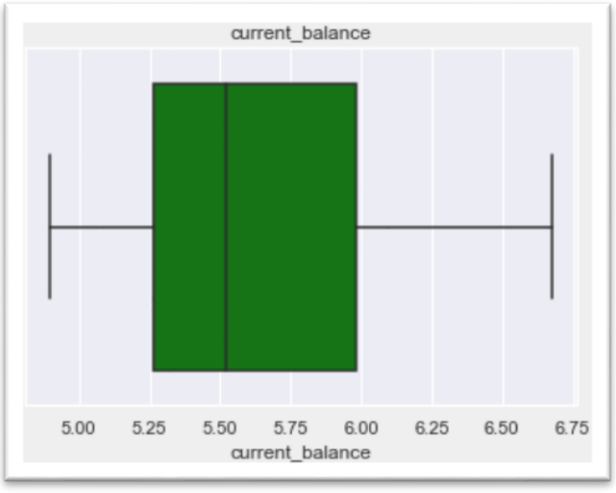
Graph 1.6 Box plot || Dist-Plot || Histogram

- Lower outliers in probability_of_full_payment: 0.8105875
- Upper outliers in probability_of_full_payment: 0.9340875
- Number of outliers in probability_of_full_payment upper: 0
- Number of outliers in probability_of_full_payment lower: 3
- % of Outlier in probability_of_full_payment upper: 0 %
- % of Outlier in probability_of_full_payment lower: 1 %

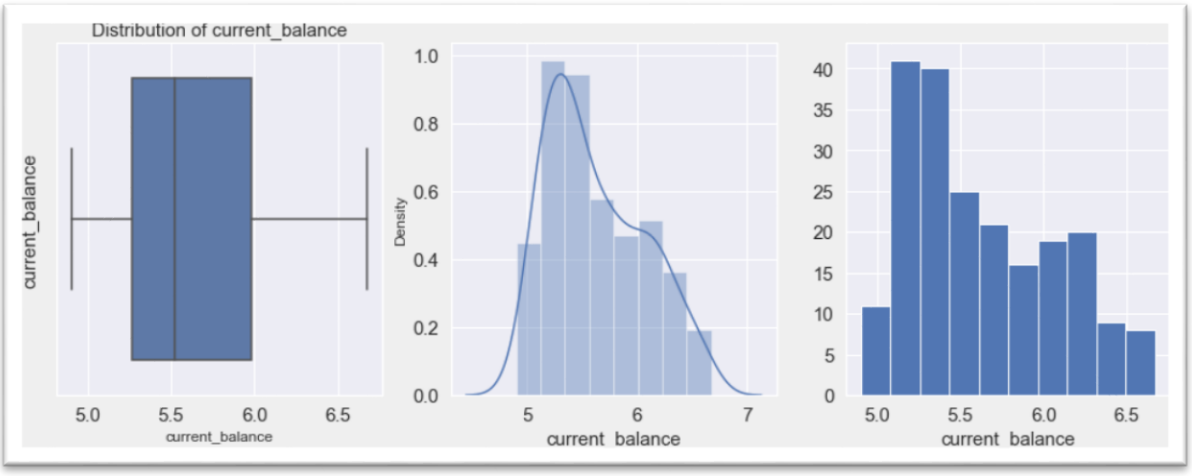
iv) current_balance variable:

Important Details:

- Range of values: 1.7759999999999998
- Minimum current_balance: 4.899
- Maximum current_balance: 6.675
- Mean value: 5.628533333333335
- Median value: 5.5235
- Standard deviation: 0.44306347772644944
- Null values: False
- current_balance - 1st Quartile (Q1) is: 5.26225
- current_balance - 3st Quartile (Q3) is: 5.97975
- Interquartile range (IQR) of current_balance is: 0.7175000000000002
- Lower outliers in current_balance: 4.186
- Upper outliers in current_balance: 7.056000000000001
- Number of outliers in current_balance upper: 0
- Number of outliers in current_balance lower: 0
- % of Outlier in current_balance upper: 0 %
- % of Outlier in current_balance lower: 0 %



Graph 1.7 Box Plot

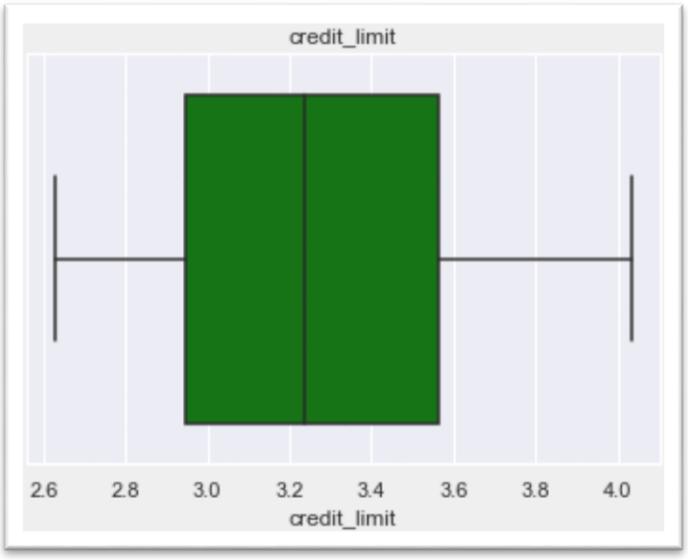


Graph 1.8 Box plot || Dist-Plot || Histogram

v) credit_limit variable:

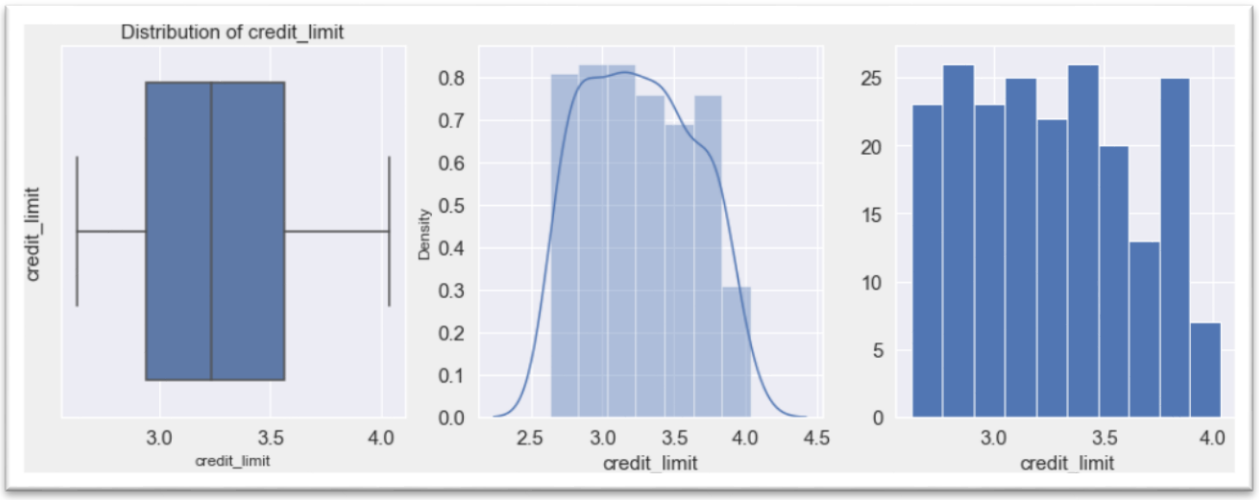
Important Details:

- Range of values: 1.4030000000000005
- Minimum credit_limit: 2.63
- Maximum credit_limit: 4.033
- Mean value: 3.258604761904763
- Median value: 3.237
- Standard deviation: 0.37771444490658734
- Null values: False
- credit_limit - 1st Quartile (Q1) is: 2.944
- credit_limit - 3st Quartile (Q3) is: 3.56175



Graph 1.9 Box Plot

- Interquartile range (IQR) of credit_limit is: 0.61775
- Lower outliers in credit_limit: 2.017375
- Upper outliers in credit_limit: 4.488375
- Number of outliers in credit_limit upper: 0
- Number of outliers in credit_limit lower: 0
- % of Outlier in credit_limit upper: 0 %
- % of Outlier in credit_limit lower: 0 %



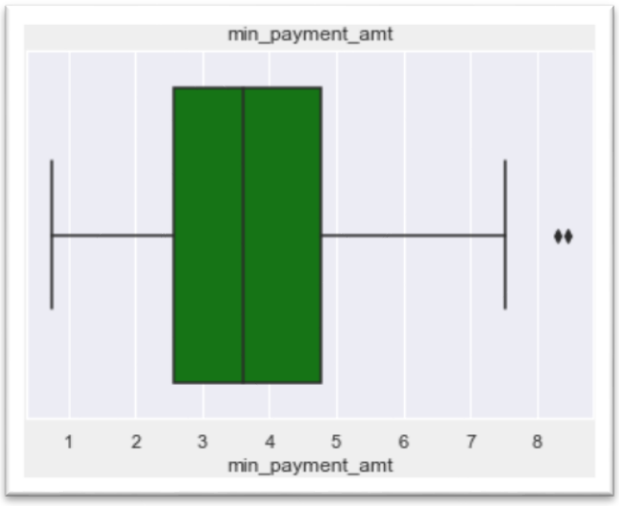
Graph 1.10 Box plot || Dist-Plot || Histogram

vi) min_payment_amt variable:

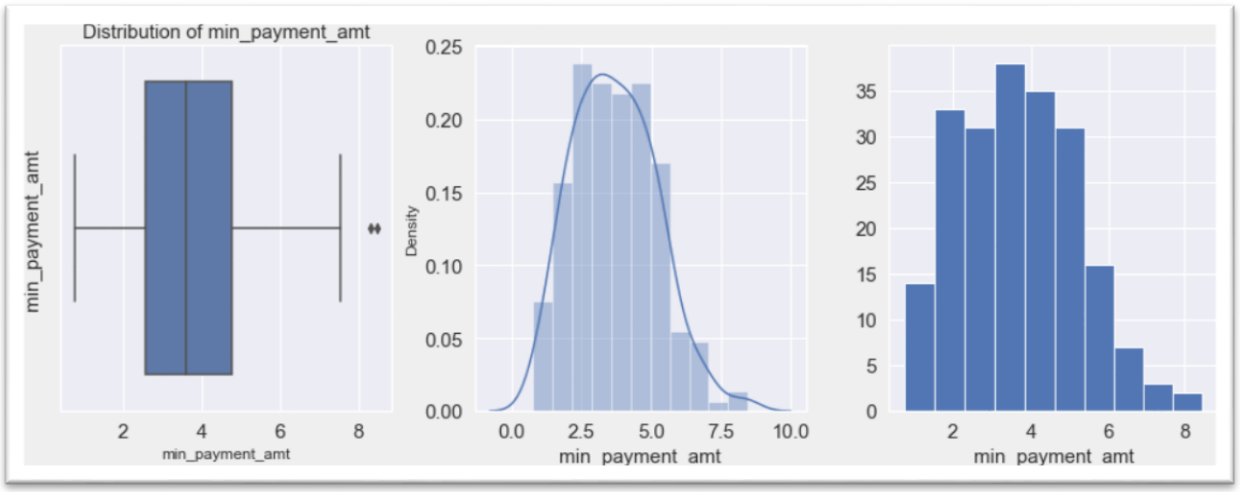
Important Details:

- Range of values: 7.690899999999999
- Minimum min_payment_amt: 0.7651
- Maximum min_payment_amt: 8.456
- Mean value: 3.7002009523809503
- Median value: 3.599
- Standard deviation: 1.5035571308217792
- Null values: False
- min_payment_amt - 1st Quartile (Q1) is: 2.5615
- min_payment_amt - 3st Quartile (Q3) is: 4.76875
- Interquartile range (IQR) of min_payment_amt is: 2.2072499999999997
- Lower outliers in min_payment_amt: -0.7493749999999992
- Upper outliers in min_payment_amt: 8.079625
- Number of outliers in min_payment_amt upper: 2

- Number of outliers in min_payment_amt lower: 0
- % of Outlier in min_payment_amt upper: 1 %
- % of Outlier in min_payment_amt lower: 0 %



Graph 1.11 Box Plot



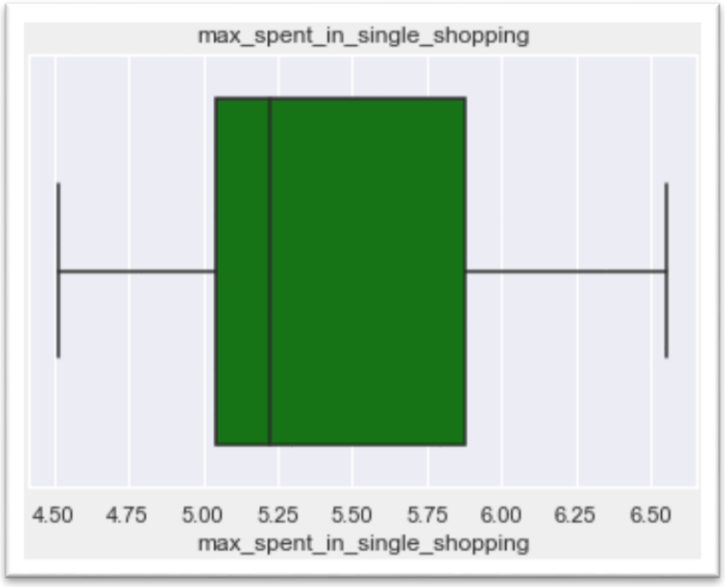
Graph 1.12 Box plot || Dist-Plot || Histogram

vii) max_spent_in_single_shopping variable:

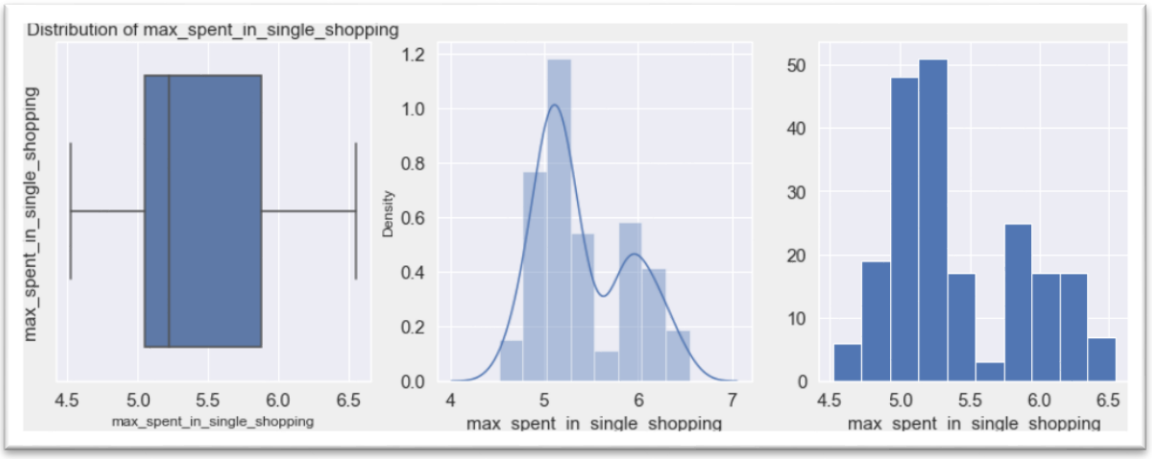
Important Details:

- Range of values: 2.0309999999999997
- Minimum max_spent_in_single_shopping: 4.519
- Maximum max_spent_in_single_shoppings: 6.55
- Mean value: 5.408071428571429
- Median value: 5.223000000000001

- Standard deviation: 0.49148049910240543
- Null values: False
- max_spent_in_single_shopping - 1st Quartile (Q1) is: 5.045
- max_spent_in_single_shopping - 3st Quartile (Q3) is: 5.877
- Interquartile range (IQR) of max_spent_in_single_shopping is: 0.8319999999999999
- Lower outliers in max_spent_in_single_shopping: 3.797
- Upper outliers in max_spent_in_single_shopping: 7.125
- Number of outliers in max_spent_in_single_shopping upper : 0
- Number of outliers in max_spent_in_single_shopping lower : 0
- % of Outlier in max_spent_in_single_shopping upper: 0 %
- % of Outlier in max_spent_in_single_shopping lower: 0 %

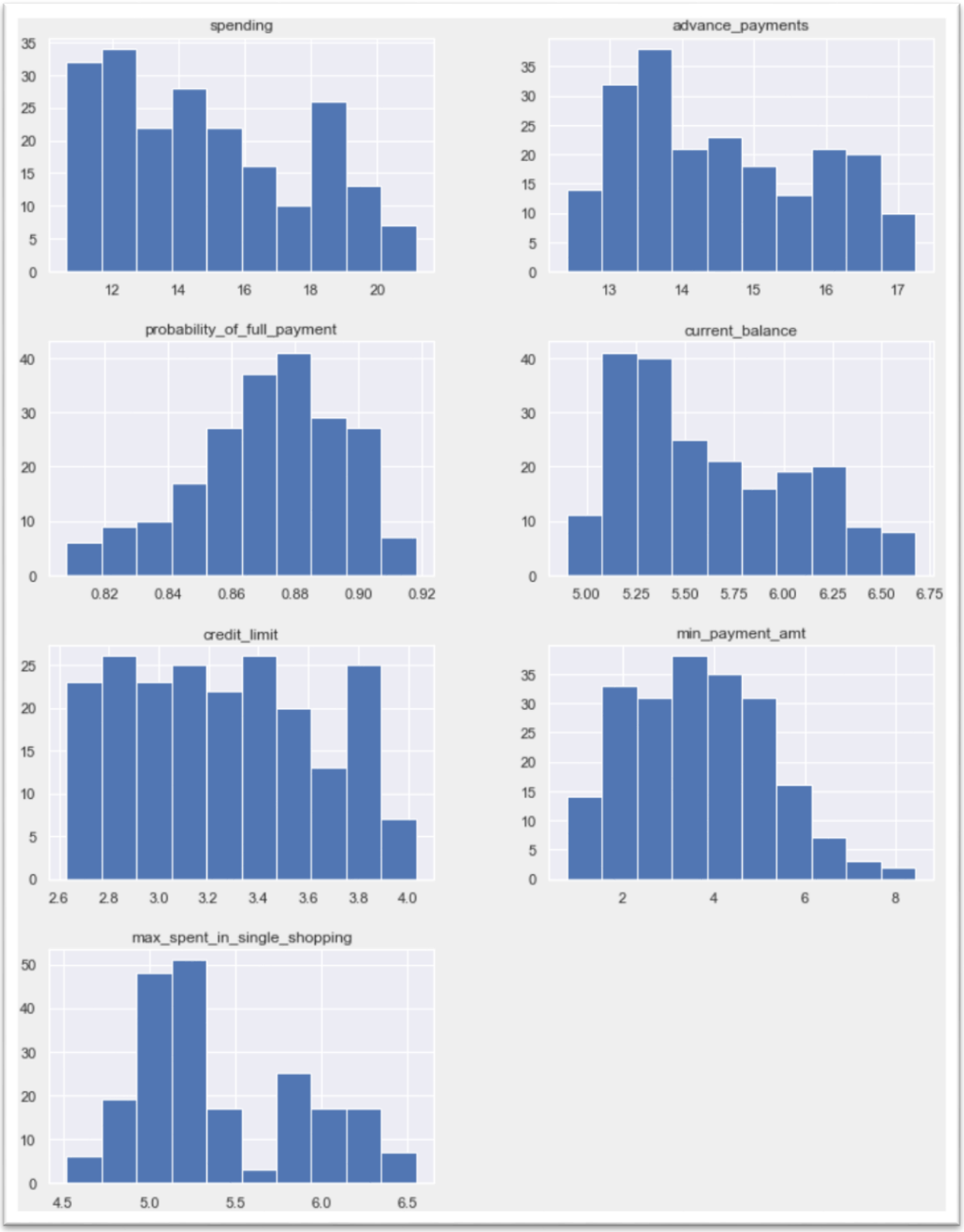


Graph 1.13 Box Plot



Graph 1.14 Box plot || Dist-Plot || Histogram

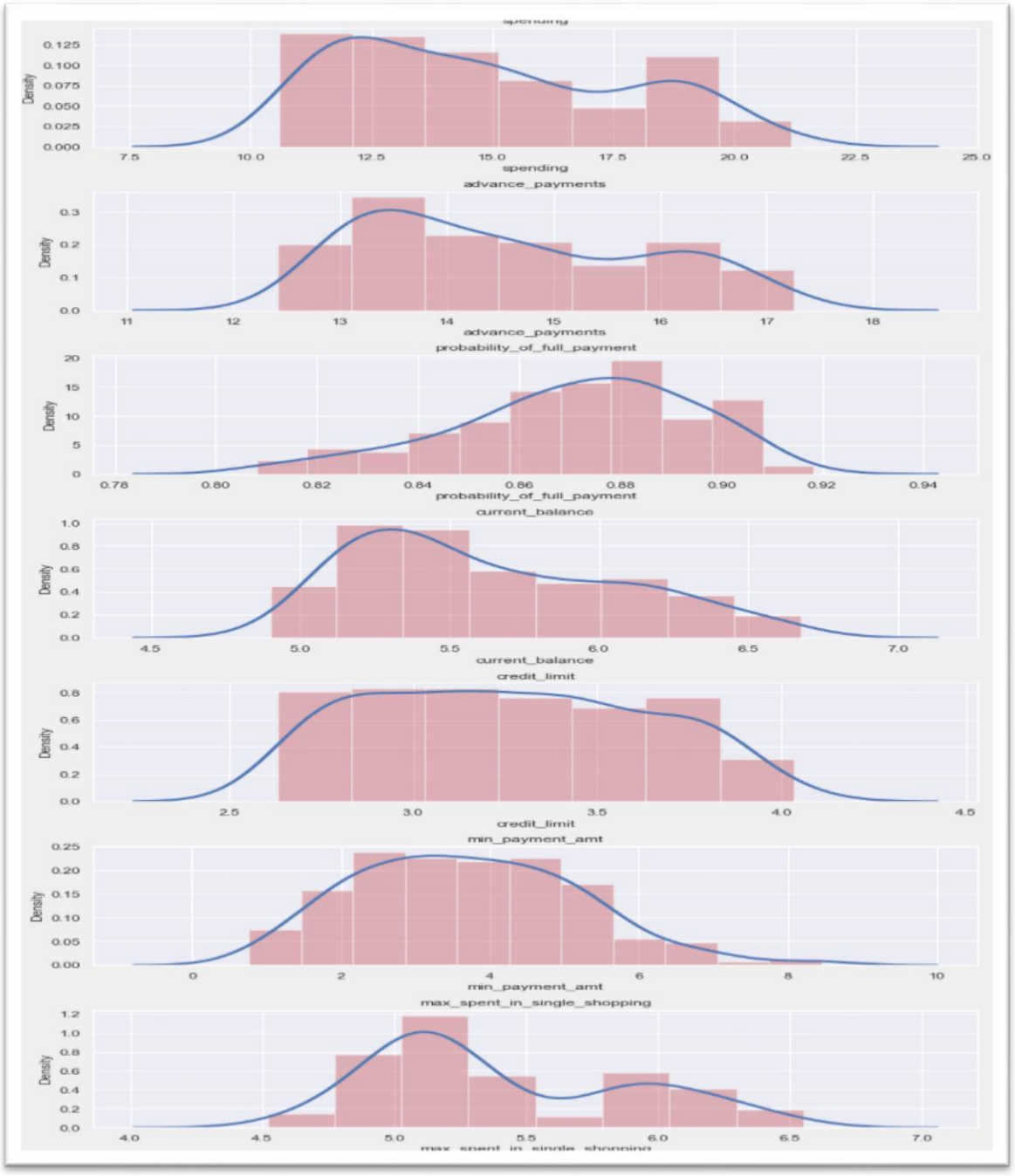
Plotting the distributions of independent attributes:



Graph 1.15 Plotting the distributions of independent attributes

Skewness Values Quantitively:

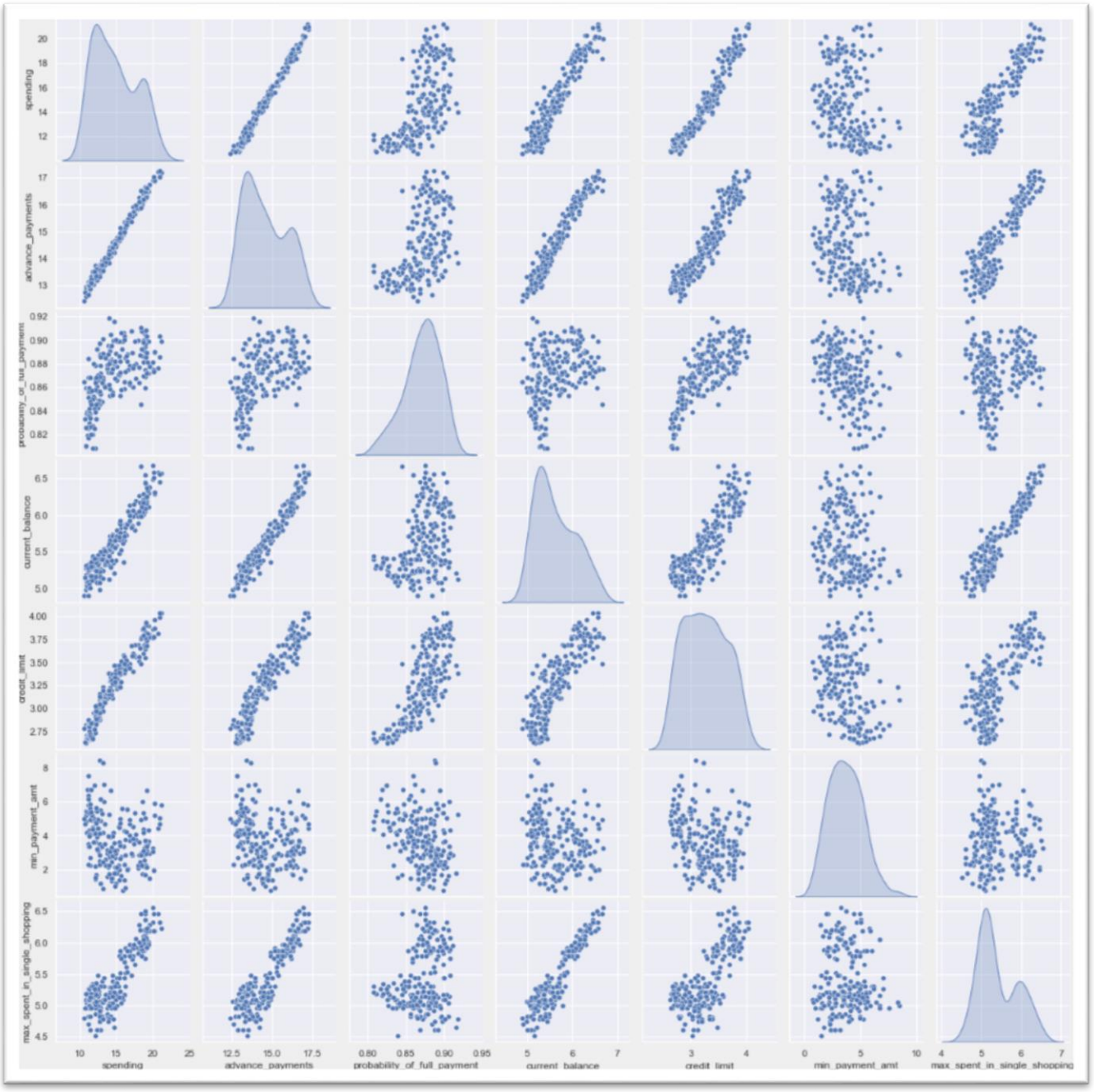
max_spent_in_single_shopping 0.561897
current_balance 0.525482
min_payment_amt 0.401667
spending 0.399889
advance_payments 0.386573
credit_limit 0.134378
probability_of_full_payment -0.537954
dtype: float64



Graph 1.16 Skewness Values Quantitively

Credit limit average is around $3.258(10000s)
Distribution is skewed to right tail for all the variable except probability_of_full_payment variable, which has left tail

Multivariate analysis:



Graph 1.17 Pair Plot

Strong positive correlation between

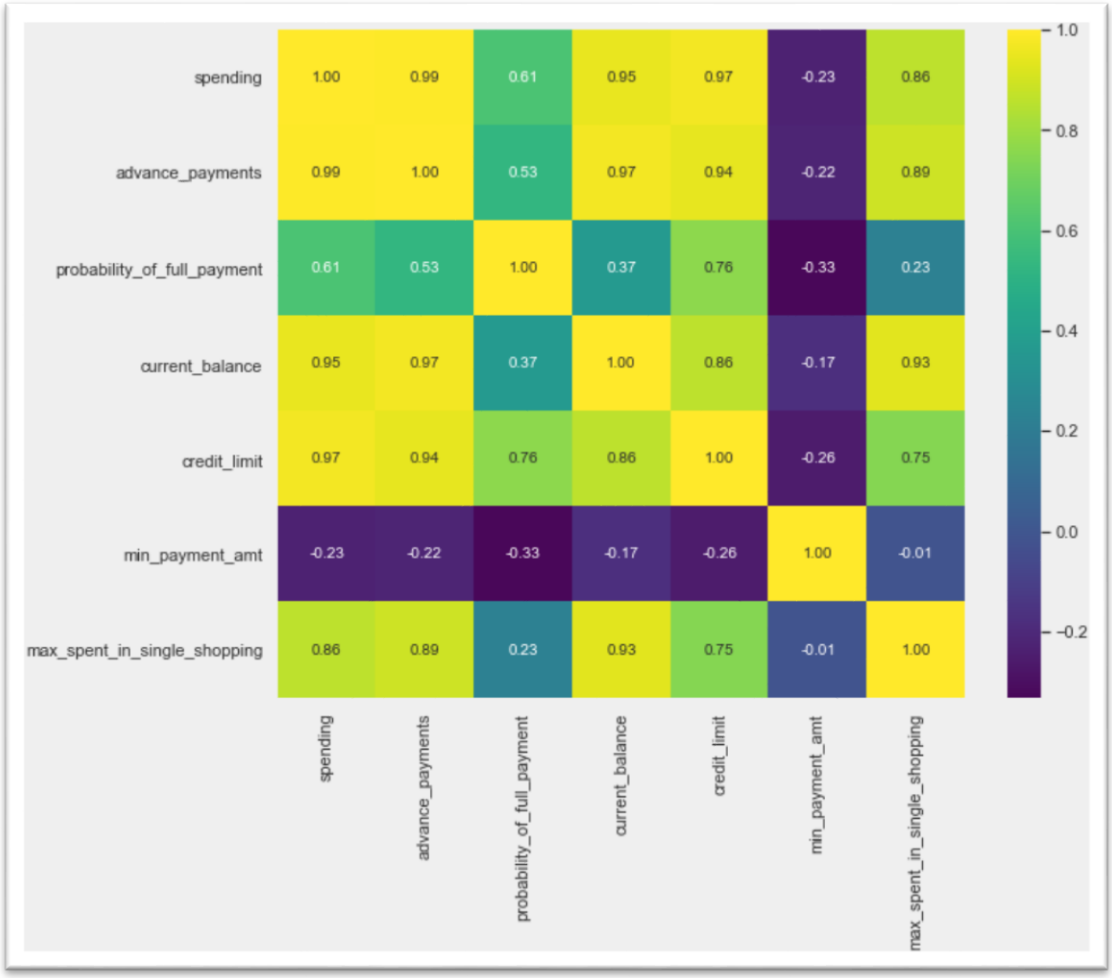
- spending & advance_payments,
- advance_payments & current_balance,
- credit_limit & spending

- 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_
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	

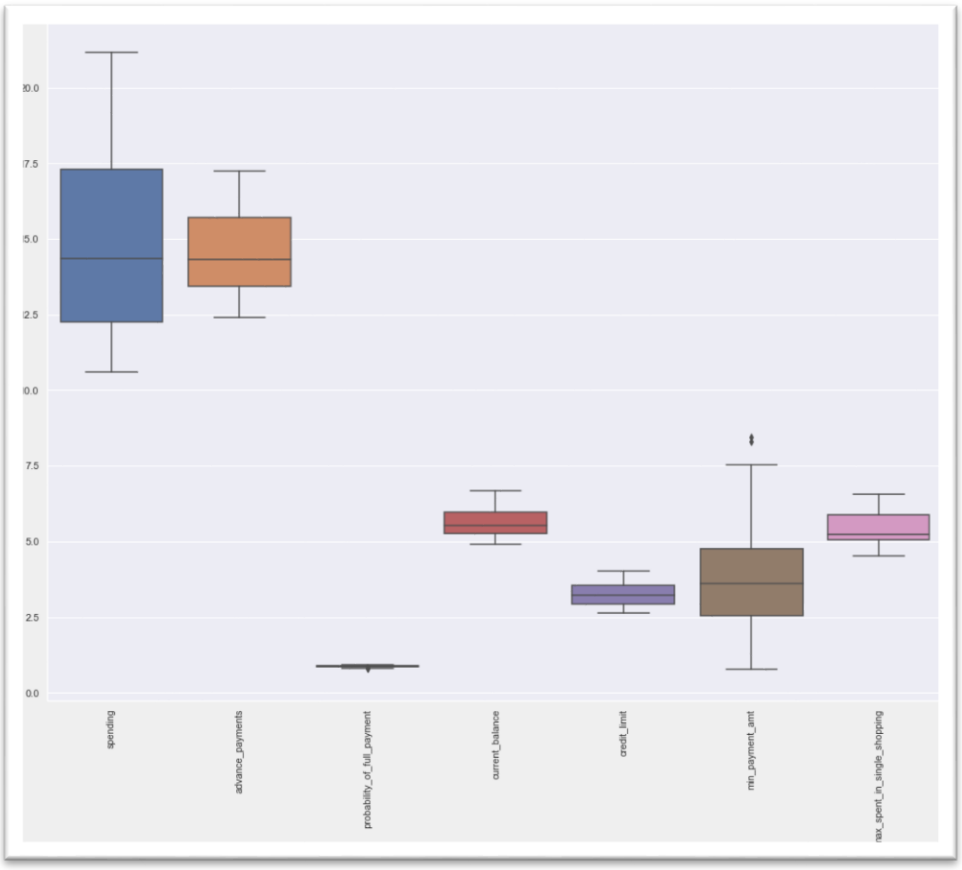
Table 1.9 Correlation Matrix



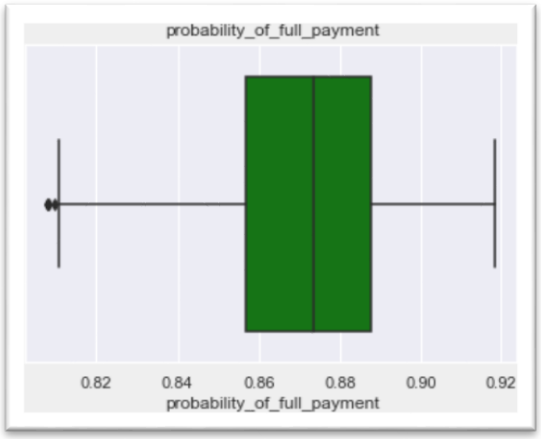
Graph 1.18 Heat Map

Checking for Outliers:

- No. of outliers in probability_of_full_payment: 3
- No. of outliers in min_payment_amt: 2
- No of attributes with outliers are: 2



Graph 1.19 Box Plots



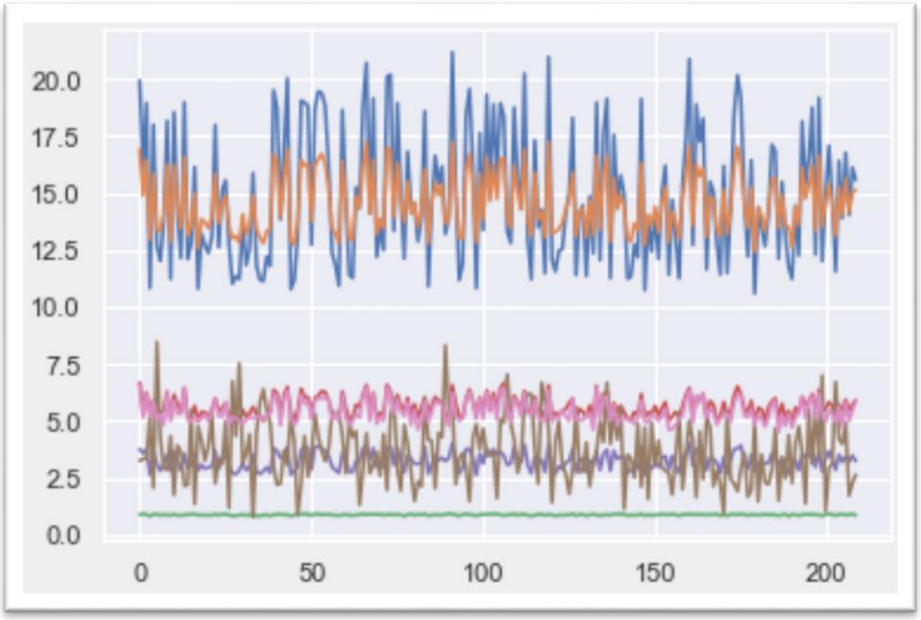
Graph 1.20 Box Plot

Though we did treat the outlier, we still see one as per the boxplot, it is okay, as it is no extreme and on lower band.

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

Normalization is used to eliminate redundant data and ensures that good quality clusters are generated which can improve the efficiency of clustering algorithms. So, it becomes an essential step before clustering as Euclidean distance is very sensitive to the changes in the differences. all dimensions are equally important.

- Scaling needs to be done as the values of the variables are different.
- spending, advance_payments are in different values and this may get more weightage.
- Also have shown below the plot of the data prior and after scaling.
- Scaling will have all the values in the relative same range.
- I have used z-score to standardise the data to relative same scale -3 to +3.

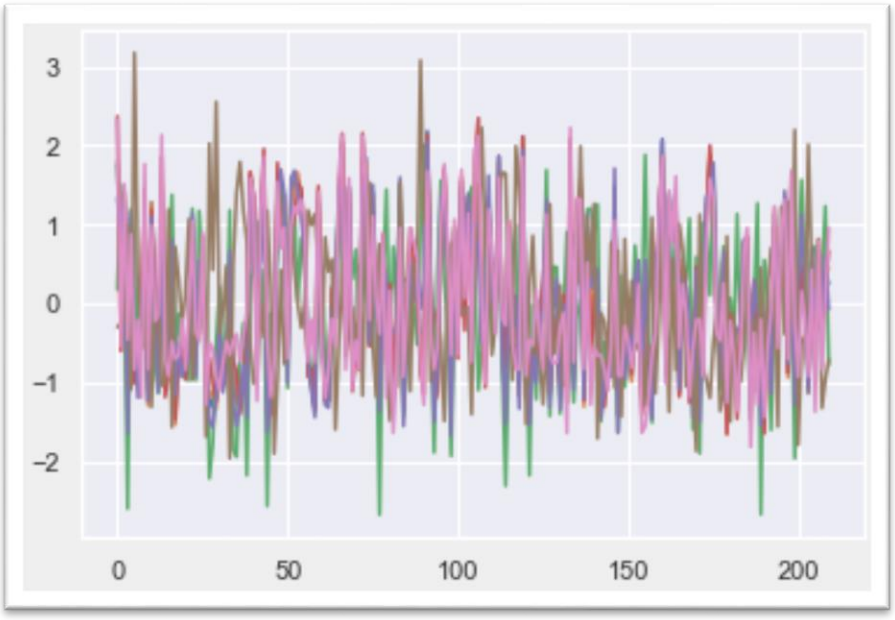


Graph 1.21 Prior to Scaling

Data after implementing the scaling:

	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

Table 1.10 Scaled table

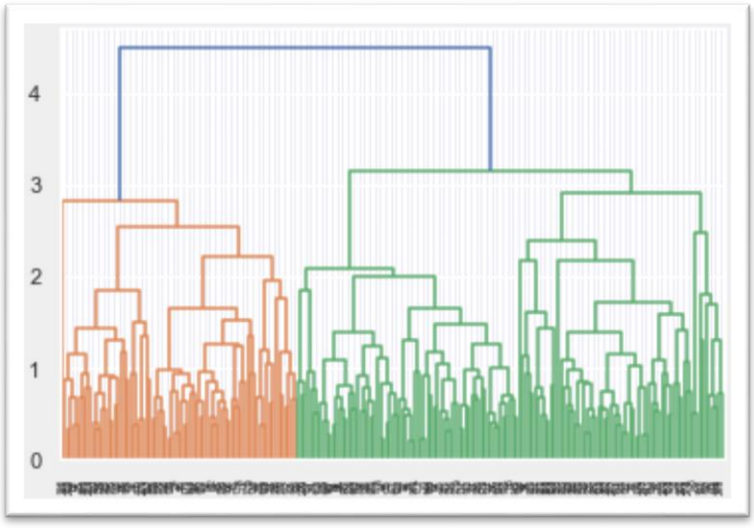


Graph 1.21 After Scaling

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

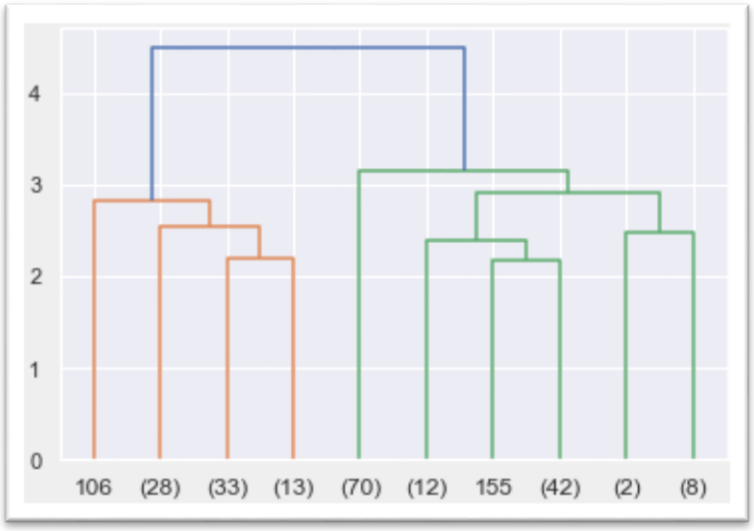
- Creating the Dendrogram
- Importing dendrogram and linkage module

Here I have made use of average linkage method:



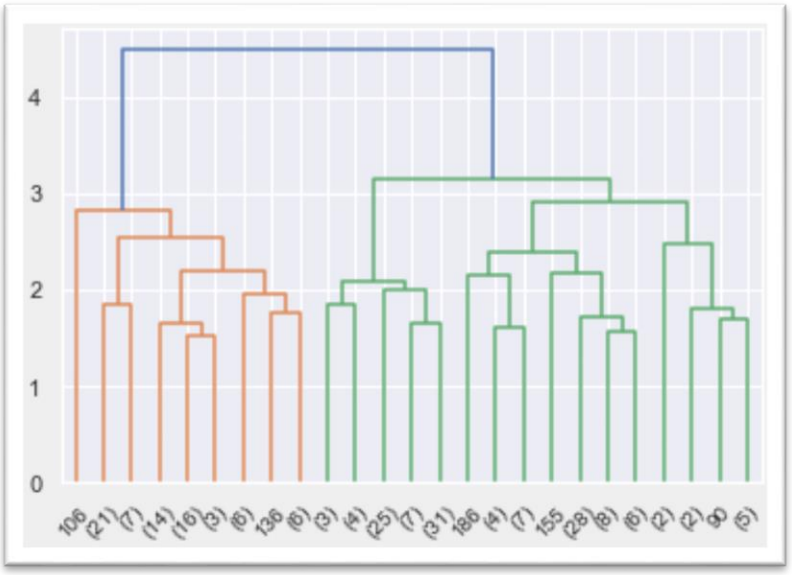
Graph 1.22 Dendrogram

After cutting the dendrogram with suitable clusters.
p=10



Graph 1.23 Dendrogram with p=10

p=25



Graph 1.24 Dendrogram with p=25

We have Set criterion as maxclust, then have created 3 clusters, and store the result in another object 'clusters'.

```
array([1, 3, 1, 2, 1, 3, 2, 2, 1, 2, 1, 1, 2, 1, 3, 3, 3, 2, 2, 2, 2, 2,
       1, 2, 3, 1, 3, 2, 2, 2, 2, 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, 1, 3, 1,
       1, 3, 1, 2, 3, 2, 1, 1, 2, 1, 3, 2, 1, 3, 3, 3, 3, 1, 2, 1, 1, 1,
       1, 3, 3, 1, 3, 2, 2, 1, 1, 1, 2, 1, 3, 1, 3, 1, 3, 1, 1, 2, 3, 1,
       1, 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, 2, 2, 1, 2, 3, 2, 3, 2, 3, 1,
       3, 3, 2, 2, 3, 1, 1, 2, 1, 1, 1, 2, 1, 3, 3, 2, 3, 2, 3, 1, 1, 1,
       3, 2, 3, 2, 3, 2, 3, 3, 1, 1, 3, 1, 3, 2, 3, 3, 2, 1, 3, 1, 1, 2,
       1, 2, 3, 3, 3, 2, 1, 3, 1, 3, 3, 1], dtype=int32)
```

Table 1.11 Array Table

Data set including clusters:

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	clusters-3
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 1.12 Data set including clusters

Cluster frequency:

```
1    75
2    70
3    65
Name: clusters-3, dtype: int64
```

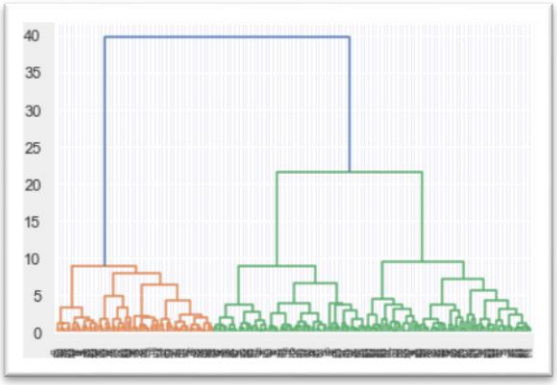
Table 1.13 Cluster Frequency

Cluster Profiles:

clusters-3	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Freq
1	18.129200	16.058000	0.881595	6.135747	3.648120	3.650200	5.987040	75
2	11.916857	13.291000	0.846766	5.258300	2.846000	4.619000	5.115071	70
3	14.217077	14.195846	0.884869	5.442000	3.253508	2.768418	5.055569	65

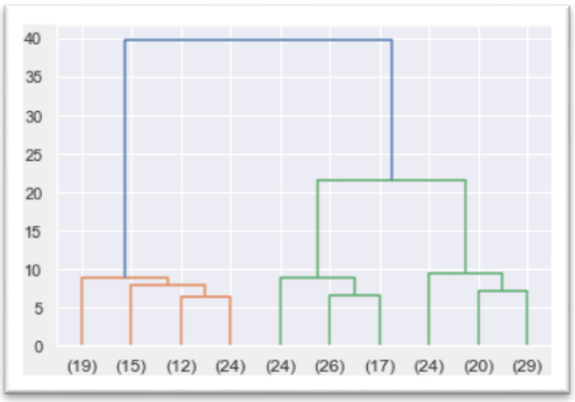
Table 1.14 Cluster Profiles

Linkage through Ward Method:



Graph 1.25 Dendrogram with Ward Method

P=10



Graph 1.26 Dendrogram with p=10

Maxcust:

```
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)
```

Table 1.15 Maxcust Array

Data Set including clusters:

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	clusters-3
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 1.16 Dataset including clusters

Cluster frequency:

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

Table 1.17 Cluster Frequency

Cluster Profiles:

clusters-3	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 1.18 Cluster Profiles

Both the method are almost similar means, minor variation, which we know it occurs. We for cluster grouping based on the dendrogram, 3 or 4 looks good. Did the further analysis, and based on the dataset had gone for 3 group cluster solution based on the hierarchical clustering Also in real time, there could have been more variables value captured - tenure, BALANCE_FREQUENCY, balance, purchase, installment of purchase, others. And 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 and silhouette score. Explain the results properly. Interpret and write inferences on the finalized clusters.

K-Means value for n_cluster 1:
1469.9999999999995

K-Means value for n_cluster 2:
659.1717544870411

K-Means value for n_cluster 3
430.65897315130064

K-Means value for n_cluster 4:
371.301721277542

WSS array:

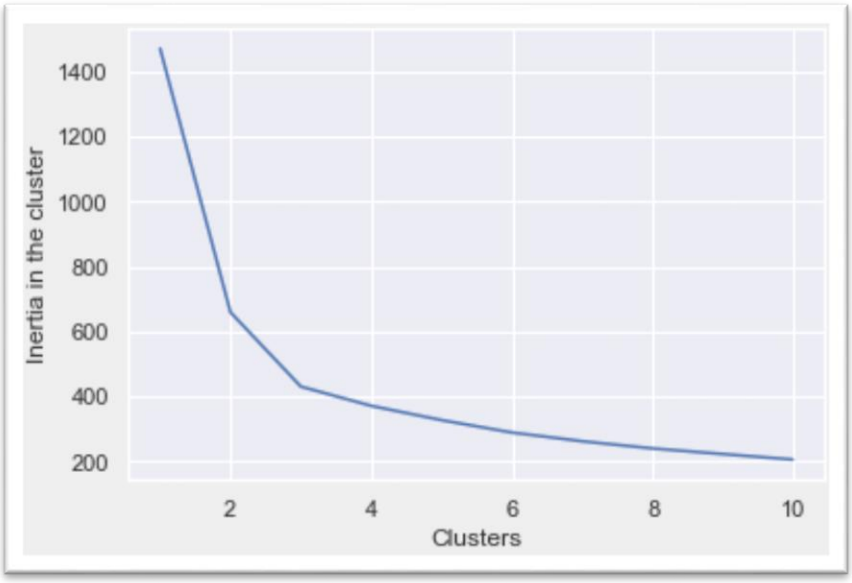
```
for i in range(1,11):
    KM = KMeans(n_clusters=i)
    KM.fit(clean_dataset_Scaled)
    wss.append(KM.inertia_)
```

```
wss
```

```
[1469.9999999999995,
 659.1717544870411,
 430.65897315130064,
 371.3943680507405,
 327.4353937624867,
 289.33181835099384,
 262.54486145291906,
 240.32824189373784,
 223.12875205523733,
 206.46042605624365]
```

Table 1.19 WSS Array

Graph through WSS:



Graph 1.27 Graph through WSS

K-Means data set head for n_cluster=4

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Clus_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	2
4	17.99	15.86	0.8992	5.890	3.694	2.068	5.837	1

Table 1.20 K-Means data set head for n_cluster = 4

Silhouette Score:

```
silhouette_score(clean_dataset_Scaled,labels_4)
```

```
0.3291966792017613
```

Table 1.21 Silhouette Score

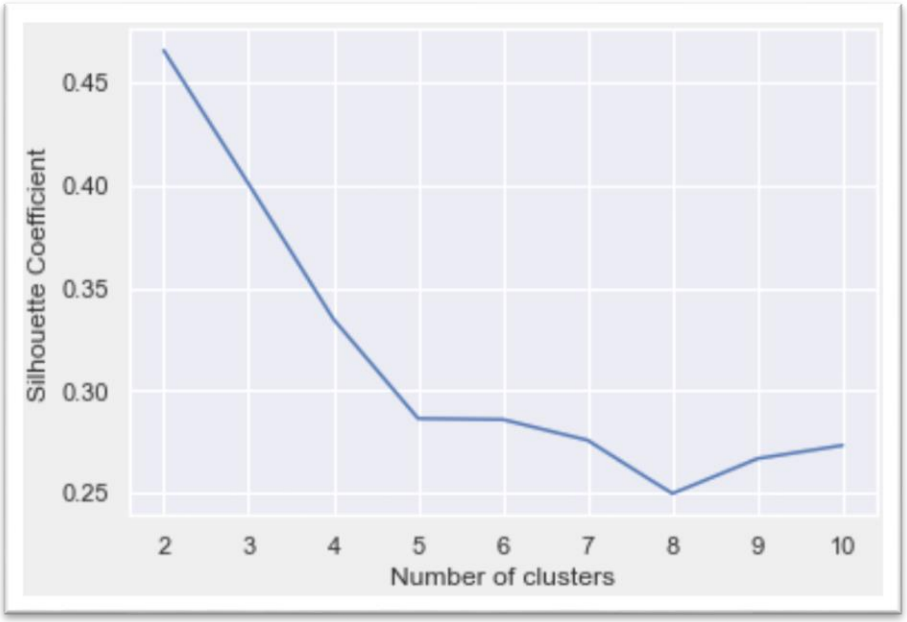
Array for Silhouette Score:

```
scores = []
k_range = range(2, 11)

for k in k_range:
    km = KMeans(n_clusters=k, random_state=2)
    km.fit(clean_dataset_Scaled)
    scores.append(metrics.silhouette_score(clean_dataset_Scaled, km.labels_))

scores
```

```
[0.46577247686580914,
 0.40072705527512986,
 0.3347542296283262,
 0.28621461554288646,
 0.285726896652541,
 0.2756098749293962,
 0.24943558736282168,
 0.2666366921192433,
 0.2731288488219916]
```

Table 1.22 Array for Silhouette Score

Silhouette Coefficient graph:



Graph 1.28 Silhouette Coefficient graph

From SC Score, the number of optimal clusters could be 3 or 4.

Data set including Silhouette width:

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	sil_width
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.550	0.432658
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144	0.099543
2	18.95	16.42	0.8829	6.248	3.755	3.368	6.148	0.425893
3	10.83	12.96	0.8099	5.278	2.641	5.182	5.185	0.529852
4	17.99	15.86	0.8992	5.890	3.694	2.068	5.837	0.082791

Table 1.23 Data set including Silhouette width

```
silhouette_samples(clean_dataset_Scaled,labels_4).min()
```

```
-0.051158059328679784
```

Table 1.24 Silhouette Min value

3 Cluster Solution:

Fitting the K Means

```
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])
```

Table 1.25 Fitting the K Means Array

Proportion of labels classified:

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

Table 1.26 Proportion of labels classified

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

Table 1.27 K-Means Clustering and Cluster Information

Cluster Percentage:

	Cluster_Size	Cluster_Percentage
1	71	33.81
2	72	34.29
3	67	31.90

Table 1.28 Cluster Percentage

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

Table 1.29 Transposing the cluster

Note:

I am going with 3 clusters via k-means, but am showing the analysis of 4 and 5 k-means cluster, I see we based on current dataset given, 3 cluster solution makes sense based on the spending pattern (High, Medium, Low)

4 Cluster Solution:

Fitting the K-Means:

```
array([0, 3, 0, 1, 0, 1, 1, 3, 0, 1, 0, 2, 1, 0, 3, 1, 3, 1, 3, 1, 1, 1,
       0, 1, 3, 2, 3, 1, 1, 1, 3, 1, 1, 3, 1, 1, 1, 1, 1, 0, 0, 3, 2, 0,
       1, 1, 3, 0, 0, 0, 1, 0, 0, 0, 0, 2, 1, 1, 1, 0, 3, 1, 1, 2, 3, 0,
       0, 3, 0, 3, 3, 1, 0, 0, 1, 0, 3, 1, 2, 3, 3, 3, 3, 0, 1, 2, 2, 2,
       2, 1, 3, 0, 3, 1, 3, 0, 0, 2, 1, 0, 3, 0, 2, 0, 3, 0, 0, 1, 3, 0,
       2, 3, 0, 1, 1, 2, 3, 2, 1, 0, 3, 1, 1, 1, 3, 3, 0, 1, 3, 3, 1, 3,
       3, 0, 1, 0, 0, 1, 2, 3, 2, 3, 1, 1, 3, 1, 0, 1, 3, 1, 3, 1, 3, 2,
       3, 3, 3, 1, 3, 0, 0, 1, 0, 2, 0, 1, 2, 3, 3, 1, 3, 1, 3, 0, 0, 0,
       3, 3, 2, 1, 3, 3, 3, 3, 2, 2, 3, 2, 3, 1, 3, 3, 1, 0, 3, 2, 0, 1,
       0, 1, 3, 2, 3, 1, 2, 3, 2, 3, 2, 2])
```

Table 1.30 Fitting the K-Means Array

Proportion of labels classified:

```
3    65
1    64
0    51
2    30
dtype: int64
```

Table 1.31 Proportion of labels classified

K-Means Clustering & Cluster Information:

cluster	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
1	19.1	16.4	0.9	6.3	3.8	3.5	6.1
2	11.9	13.3	0.8	5.2	2.8	4.9	5.1
3	16.1	15.2	0.9	5.8	3.4	4.0	5.6
4	14.1	14.1	0.9	5.4	3.2	2.4	5.0

Table 1.32 K-Means Clustering & Cluster Information

Cluster Percentage:

	Cluster_Size	Cluster_Percentage
1	51	24.29
2	68	32.38
3	30	14.29
4	61	29.05

Table 1.33 Cluster Percentage

Transposing the cluster:

cluster	1	2	3	4
spending	19.1	11.9	16.1	14.1
advance_payments	16.4	13.3	15.2	14.1
probability_of_full_payment	0.9	0.8	0.9	0.9
current_balance	6.3	5.2	5.8	5.4
credit_limit	3.8	2.8	3.4	3.2
min_payment_amt	3.5	4.9	4.0	2.4
max_spent_in_single_shopping	6.1	5.1	5.6	5.0

Table 1.34 Transposing the cluster

5 Cluster Solution:

K-Means Clustering & Cluster Information:

cluster	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
1	11.6	13.2	0.8	5.3	2.8	4.7	5.2
2	19.1	16.4	0.9	6.3	3.8	3.5	6.1
3	14.2	14.2	0.9	5.5	3.2	2.3	5.0
4	12.3	13.3	0.9	5.2	3.0	5.0	5.0
5	16.2	15.2	0.9	5.9	3.4	3.9	5.7

Table 1.35 K-Means Clustering & Cluster Information

Cluster Percentage:

	Cluster_Size	Cluster_Percentage
1	39	18.57
2	50	23.81
3	55	26.19
4	36	17.14
5	30	14.29

Table 1.36 Cluster Percentage

Transposing the cluster:

cluster	1	2	3	4	5
spending	11.6	19.1	14.2	12.3	16.2
advance_payments	13.2	16.4	14.2	13.3	15.2
probability_of_full_payment	0.8	0.9	0.9	0.9	0.9
current_balance	5.3	6.3	5.5	5.2	5.9
credit_limit	2.8	3.8	3.2	3.0	3.4
min_payment_amt	4.7	3.5	2.3	5.0	3.9
max_spent_in_single_shopping	5.2	6.1	5.0	5.0	5.7

Table 1.37 Transposing the cluster

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

3 group cluster via K-Means:

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

Table 1.38 3-Group cluster via K-Means

3 group cluster via hierarchical clustering:

clusters-3	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

Table 1.39 3 group cluster via hierarchical clustering

Cluster Group Profiles:

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

Group 1: High Spending Group

- 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 luxury brands, which will drive more one_time_maximun spending

Group 3: Medium Spending Group

- 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 transactions.
- Increase spending habits by trying with premium ecommerce sites, travel portal, travel airlines/hotel, as this will encourage them to spend more.

Group 2: Low Spending Group

- 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, utilities (electricity, phone, gas, others)

Problem 2:

Executive Summary:

The given data set has number columns and rows in them. The given data provides us the information with regards to the insurance firm. A Classification and Regression Tree (CART), is a predictive model, which explains how an outcome variable's values can be predicted based on other values. A CART output is a decision tree where each fork is a split in a predictor variable and each end node contains a prediction for the outcome variable. Random Forest (RF) is one of the many machine learning algorithms used for supervised learning, this means for learning from labelled data and making predictions based on the learned patterns. RF can be used for both classification and regression tasks. Artificial Neural networks (ANN) or neural networks are computational algorithms. It intended to simulate the behavior of biological systems composed of “neurons”. ANNs are computational models inspired by an animal's central nervous systems. It is capable of machine learning as well as pattern recognition.

Background of the problem:

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.

Data Description:

```
<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
```

Table 2.1 Data Description

- 10 variables
- Age, Commision, Duration, Sales are numeric variable
- rest are categorial variables
- 3000 records, no missing one
- 9 independant variable and one target variable – Clamied

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

Table 2.2 Null Values

Sample Data Set:

	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 2.3 Sample Data set

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

Data Set:

	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 2.4 Sample Data set

Data Type:

```
<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
```

Table 2.5 Data Type

- 10 variables
- Age, Commision, Duration, Sales are numeric variable
- rest are categorial variables
- 3000 records, no missing one
- 9 independent 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
```

Table 2.6 Checking for missing values

- No missing values

Descriptive Statistics Summary:

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

Table 2.7 Descriptive Statistics Summary

	count	mean	std	min	25%	50%	75%	90%	max
Age	3000.0	38.091000	10.463518	8.0	32.0	36.00	42.000	53.000	84.00
Commision	3000.0	14.529203	25.481455	0.0	0.0	4.63	17.235	48.300	210.21
Duration	3000.0	70.001333	134.053313	-1.0	11.0	26.50	63.000	224.200	4580.00
Sales	3000.0	60.249913	70.733954	0.0	20.0	33.00	69.000	172.025	539.00

Table 2.8 Descriptive Statistics Summary

- Duration has negative value; it is not possible. Wrong entry.
- Commision & Sales- mean and median varies significantly.

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Age	3000.0	NaN	NaN	NaN	38.091	10.463518	8.0	32.0	36.0	42.0	84.0
Agency_Code	3000	4	EPX	1365	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Type	3000	2	Travel Agency	1837	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Claimed	3000	2	No	2076	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Commision	3000.0	NaN	NaN	NaN	14.529203	25.481455	0.0	0.0	4.63	17.235	210.21
Channel	3000	2	Online	2954	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Duration	3000.0	NaN	NaN	NaN	70.001333	134.053313	-1.0	11.0	26.5	63.0	4580.0
Sales	3000.0	NaN	NaN	NaN	60.249913	70.733954	0.0	20.0	33.0	69.0	539.0
Product Name	3000	5	Customised Plan	1136	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Destination	3000	3	ASIA	2465	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 2.9 Descriptive Statistics Summary

- Categorial code variable maximum unique count is 5.

Data dimension:

```
df.shape

(3000, 10)
```

Table 2.10 Data Dimension

Getting unique counts of all Nominal Variables:

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


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


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


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


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

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

Table 2.11 Getting unique counts of all Nominal Variables

Checking for duplicate data:

Number of duplicate rows = 139

	Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
63	30	C2B	Airlines	Yes	15.0	Online	27	60.0	Bronze Plan	ASIA
329	36	EPX	Travel Agency	No	0.0	Online	5	20.0	Customised Plan	ASIA
407	36	EPX	Travel Agency	No	0.0	Online	11	19.0	Cancellation Plan	ASIA
411	35	EPX	Travel Agency	No	0.0	Online	2	20.0	Customised Plan	ASIA
422	36	EPX	Travel Agency	No	0.0	Online	5	20.0	Customised Plan	ASIA
...	...	...	...	...	...	...	...	...	...	...
2940	36	EPX	Travel Agency	No	0.0	Online	8	10.0	Cancellation Plan	ASIA
2947	36	EPX	Travel Agency	No	0.0	Online	10	28.0	Customised Plan	ASIA
2952	36	EPX	Travel Agency	No	0.0	Online	2	10.0	Cancellation Plan	ASIA
2962	36	EPX	Travel Agency	No	0.0	Online	4	20.0	Customised Plan	ASIA
2984	36	EPX	Travel Agency	No	0.0	Online	1	20.0	Customised Plan	ASIA

139 rows × 10 columns

Table 2.12 Checking for duplicate data

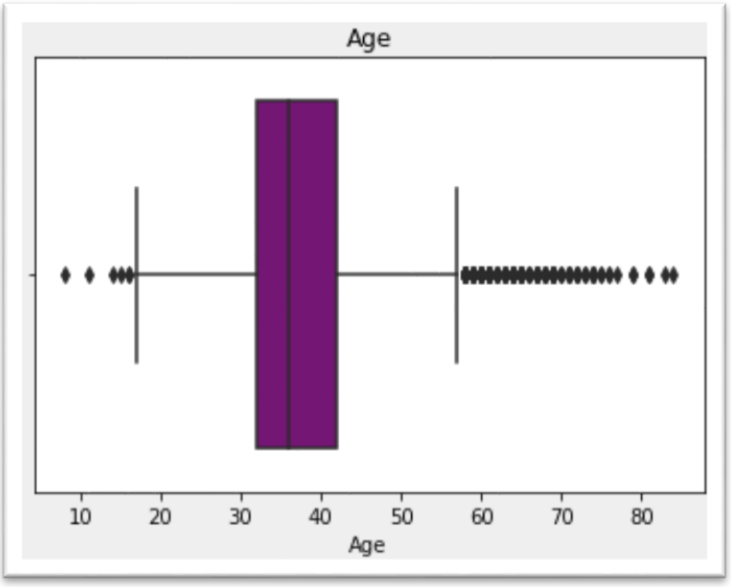
- Though it shows there are 139 records, but it can be of different customers, there is no customer ID or any unique identifier, so I am not dropping them off.

Univariate Analysis:

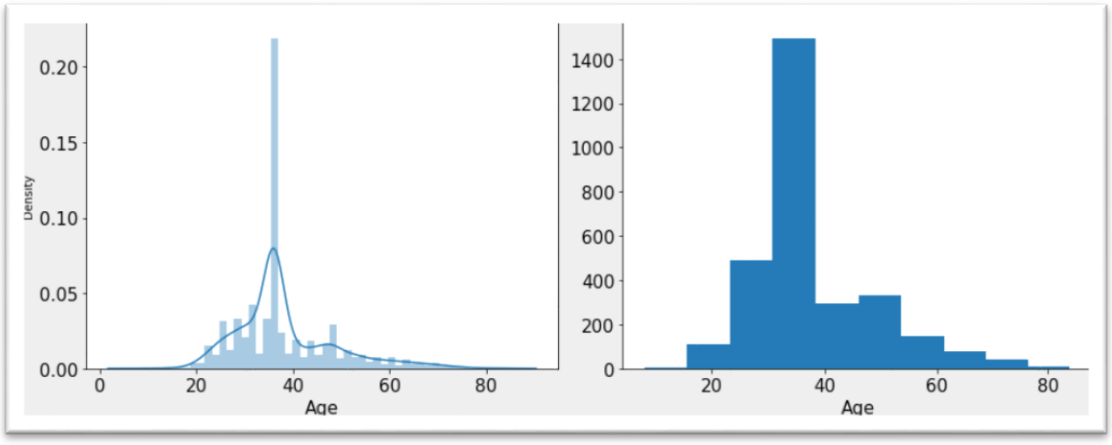
i) Age Variable:

Important Details:

- Range of values: 76
- Minimum Age: 8
- Maximum Age: 84
- Mean value: 38.091
- Median value: 36.0
- Standard deviation: 10.463518245377944
- Null values: False
- spending - 1st Quartile (Q1) is: 32.0
- spending - 3st Quartile (Q3) is: 42.0
- Interquartile range (IQR) of Age is: 10.0
- Lower outliers in Age: 17.0
- Upper outliers in Age: 57.0
- Number of outliers in Age upper: 198
- Number of outliers in Age lower: 6
- % of Outlier in Age upper: 7 %
- % of Outlier in Age lower: 0 %



Graph 2.1 Box Plot

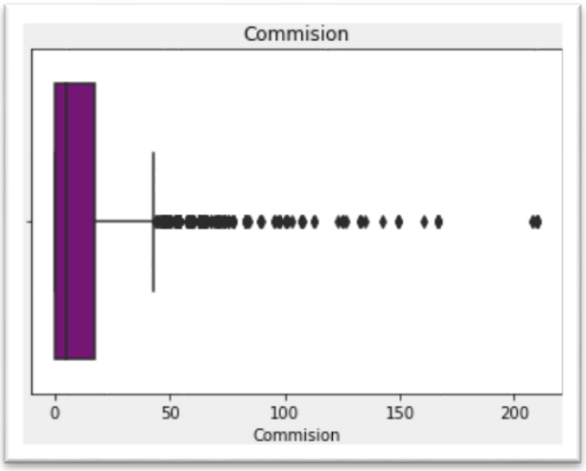


Graph 2.2 Dist-Plot || Histogram

ii) Commision Variable:

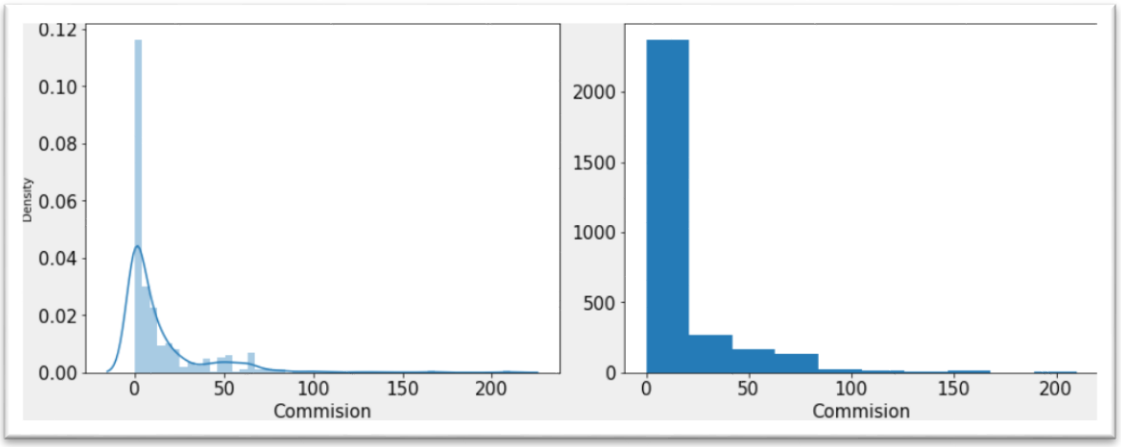
Important Details:

- Range of values: 210.21
- Minimum Commision: 0.0
- Maximum Commision: 210.21
- Mean value: 14.529203333333266
- Median value: 4.63
- Standard deviation: 25.48145450662553
- Null values: False
- Commision - 1st Quartile (Q1) is: 0.0
- Commision - 3st Quartile (Q3) is: 17.235
- Interquartile range (IQR) of Commision is: 17.235



Graph 2.3 Box Plot

- Lower outliers in Commision: -25.8525
- Upper outliers in Commision: 43.0875
- Number of outliers in Commision upper: 362
- Number of outliers in Commision lower: 0
- % of Outlier in Commision upper: 12 %
- % of Outlier in Commision lower: 0 %

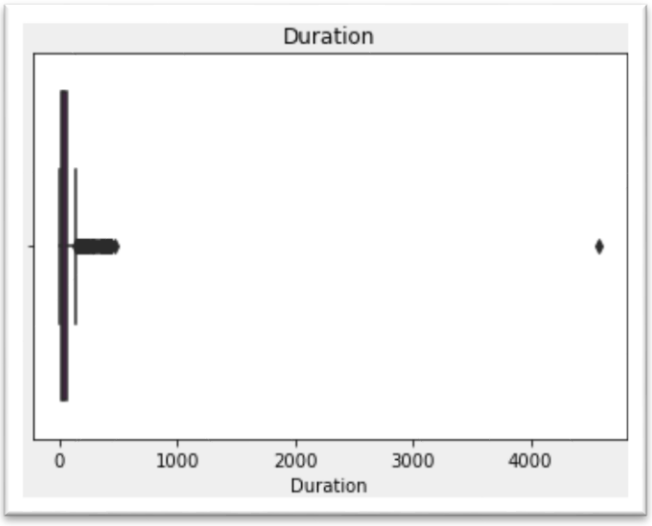


Graph 2.4 Dist-Plot || Histogram

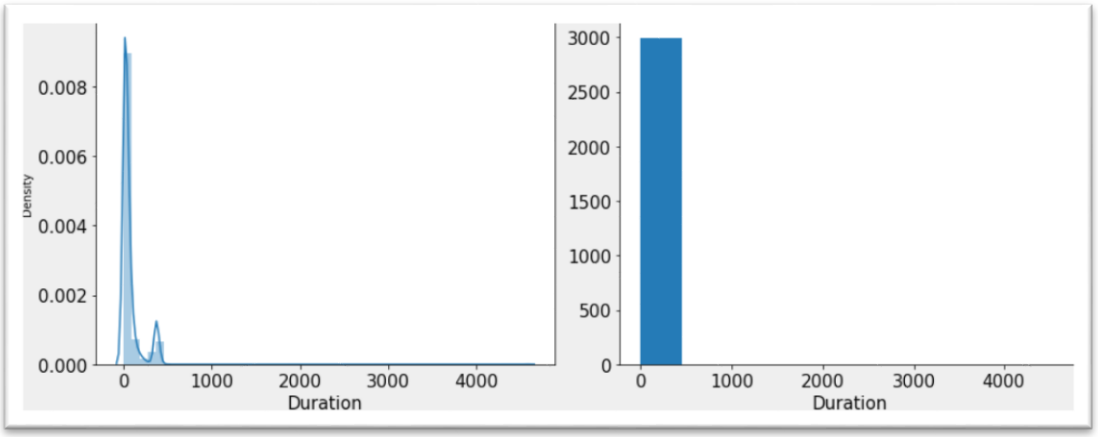
iii) Duration variable:

Important Details:

- Range of values: 4581
- Minimum Duration: -1
- Maximum Duration: 4580
- Mean value: 70.00133333333333
- Median value: 26.5
- Standard deviation: 134.05331313253495
- Null values: False
- Duration - 1st Quartile (Q1) is: 11.0
- Duration - 3st Quartile (Q3) is: 63.0
- Interquartile range (IQR) of Duration is 52.0
- Lower outliers in Duration: -67.0
- Upper outliers in Duration: 141.0
- Number of outliers in Duration upper : 382
- Number of outliers in Duration lower : 0
- % of Outlier in Duration upper: 13 %
- % of Outlier in Duration lower: 0 %



Graph 2.5 Box Plot



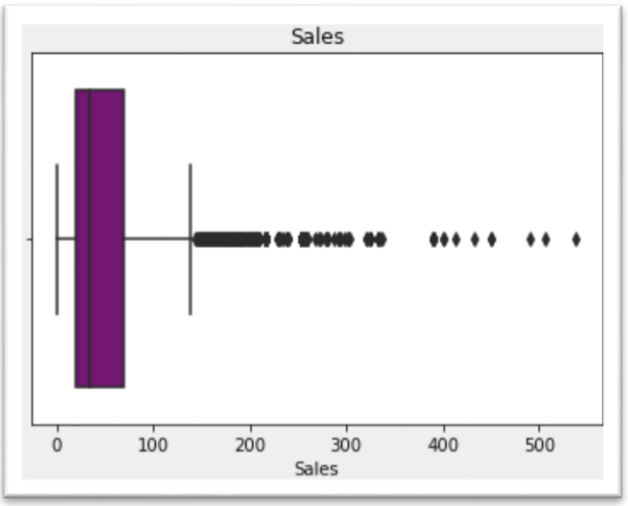
Graph 2.6 Dist-Plot || Histogram

iv) Sales Variable:

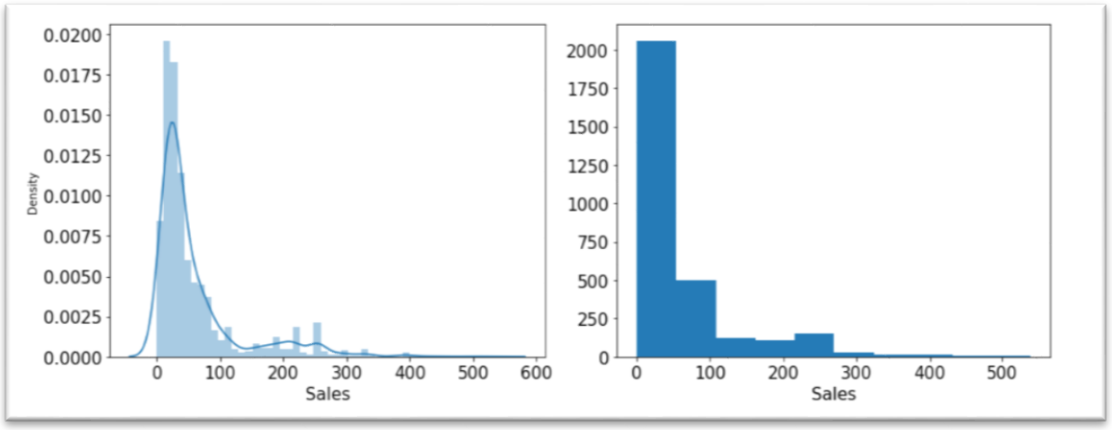
Important Details:

- Range of values: 539.0
- Minimum Sales: 0.0
- Maximum Sales: 539.0
- Mean value: 60.24991333333344
- Median value: 33.0
- Standard deviation: 70.73395353143047
- Null values: False
- Sales - 1st Quartile (Q1) is: 20.0
- Sales - 3st Quartile (Q3) is: 69.0

- Interquartile range (IQR) of Sales is: 49.0
- Lower outliers in Sales: -53.5
- Upper outliers in Sales: 142.5
- Number of outliers in Sales upper : 353
- Number of outliers in Sales lower : 0
- % of Outlier in Sales upper: 12 %
- % of Outlier in Sales lower: 0 %



Graph 2.7 Box Plot



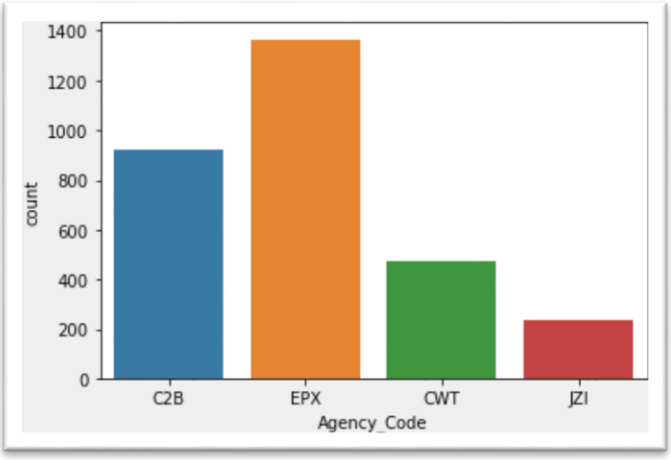
Graph 2.8 Dist-Plot || Histogram

There are outliers in all the variables, but the sales and commision can be a genuine business value. Random Forest and CART can handle the outliers. Hence, Outliers are not treated for now, we will keep the data as it is.

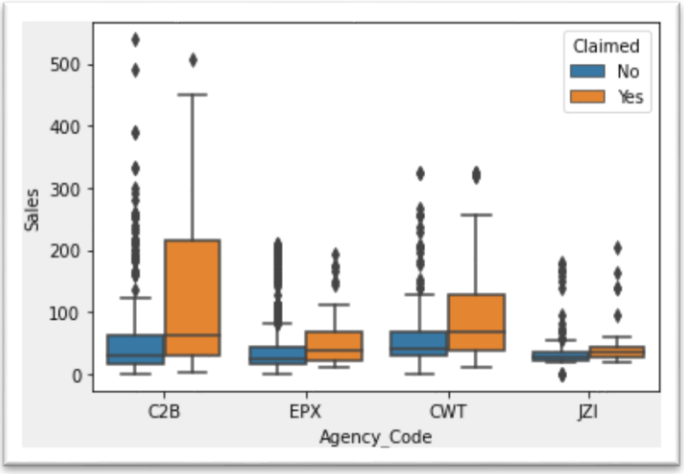
I will treat the outliers for the ANN model to compare the same after the all the steps just for comparison.

Categorical Variable:

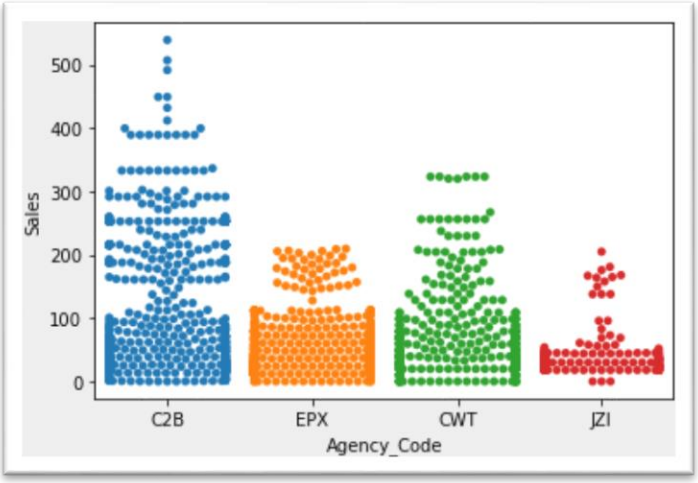
i) Agency_Code:



Graph 2.9 Count Plot

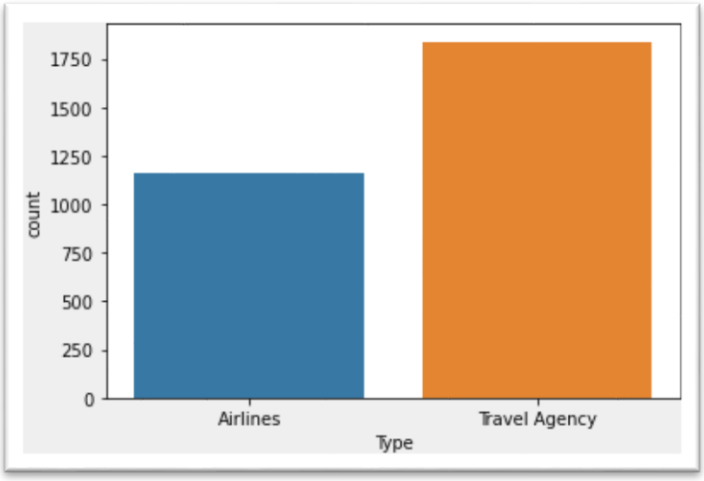


Graph 2.10 Box Plot

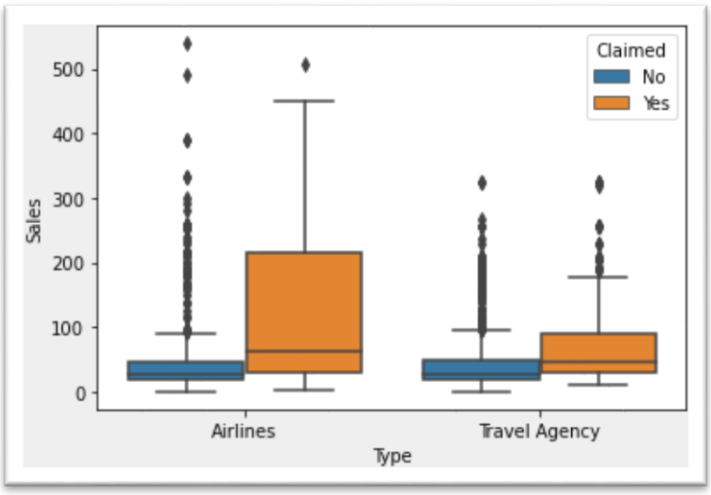


Graph 2.11 Swarm Plot

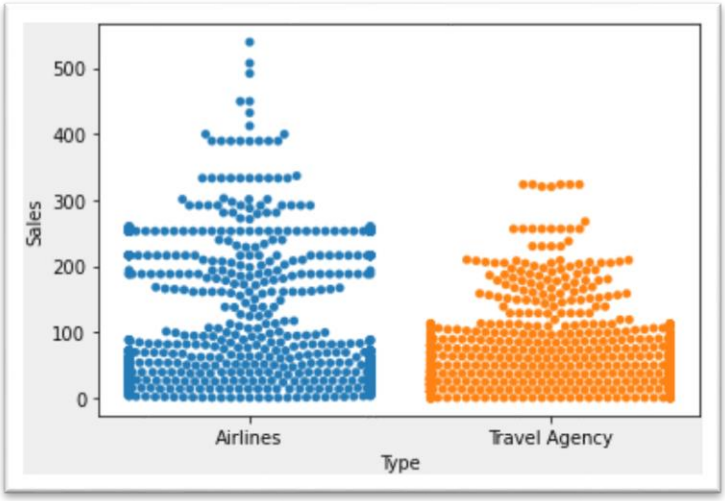
ii) Type:



Graph 2.12 Count Plot

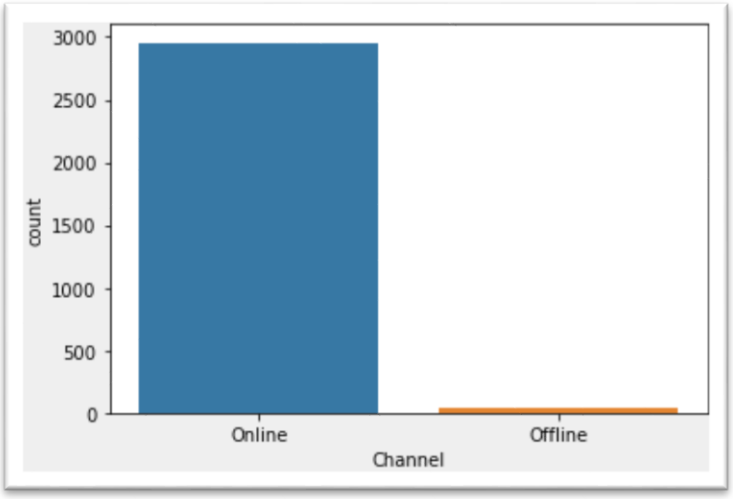


Graph 2.13 Box Plot

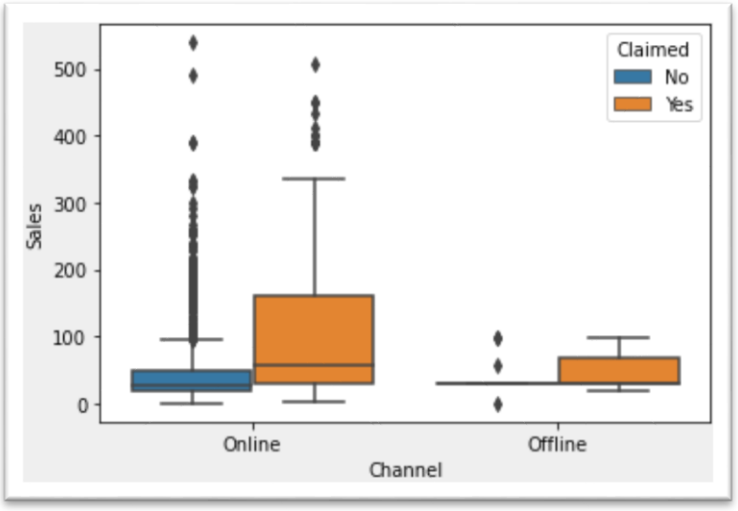


Graph 2.14 Swarm Plot

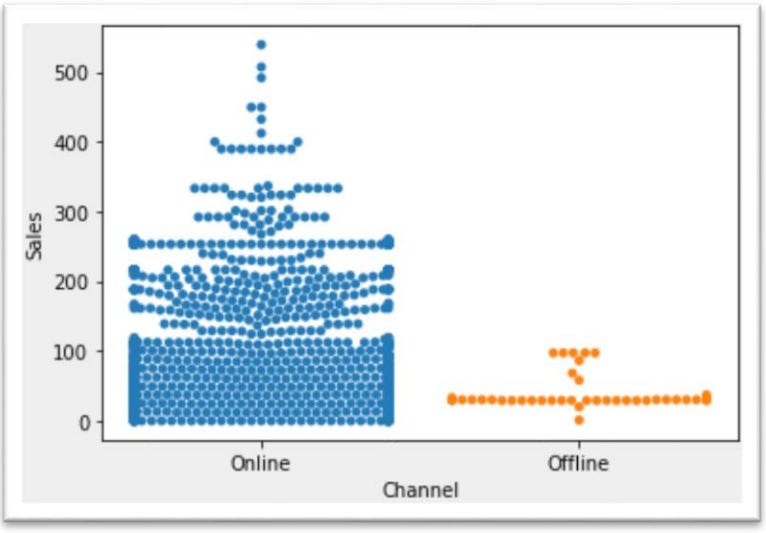
iii) Channel:



Graph 2.15 Count Plot

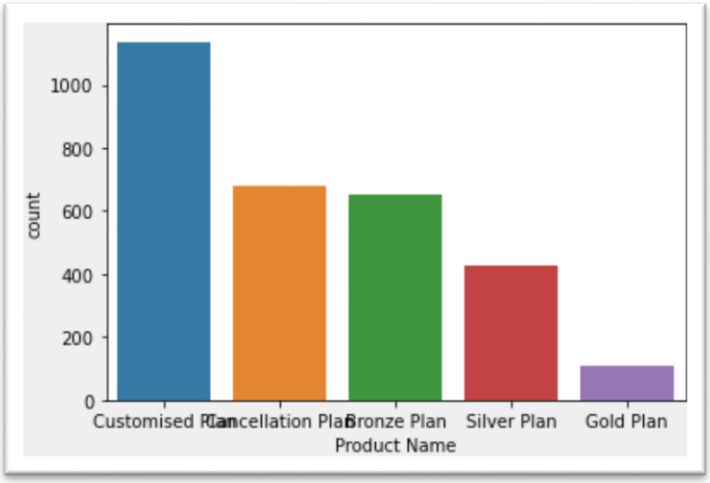


Graph 2.16 Box Plot

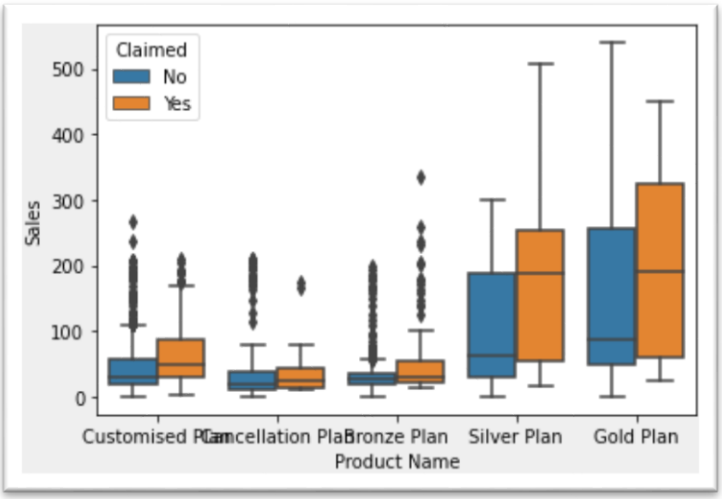


Graph 2.17 Swarm Plot

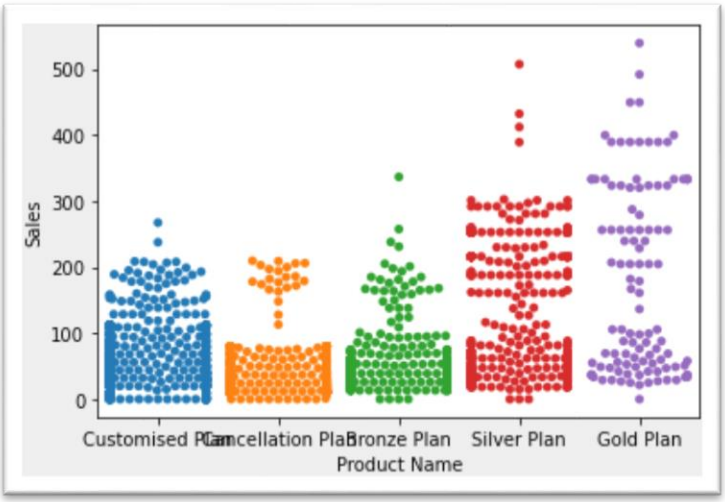
iv) Product Name:



Graph 2.18 Count Plot

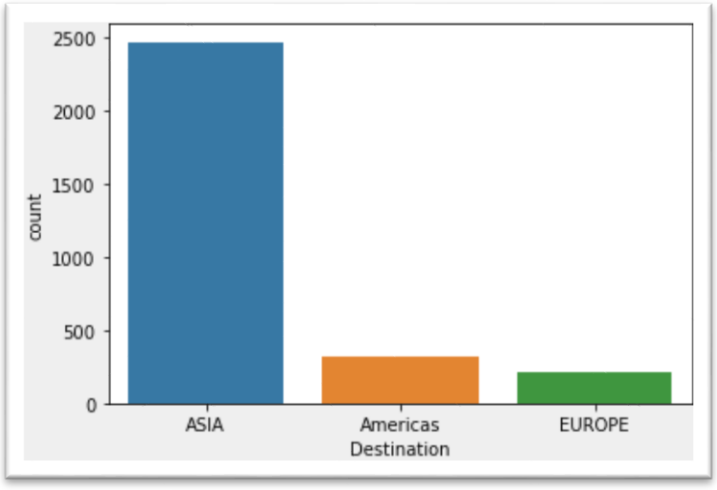


Graph 2.19 Box Plot

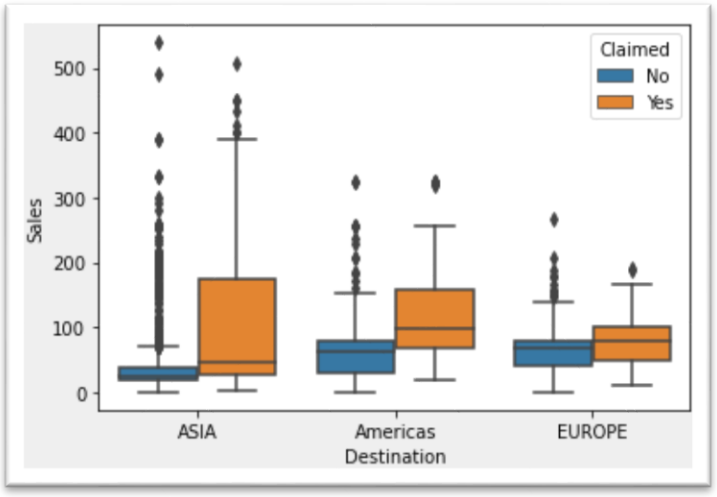


Graph 2.20 Swarm Plot

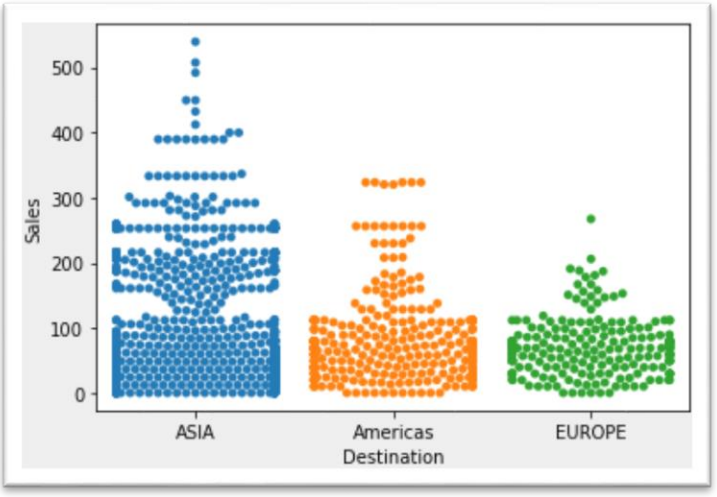
v) Destination:



Graph 2.21 Count Plot

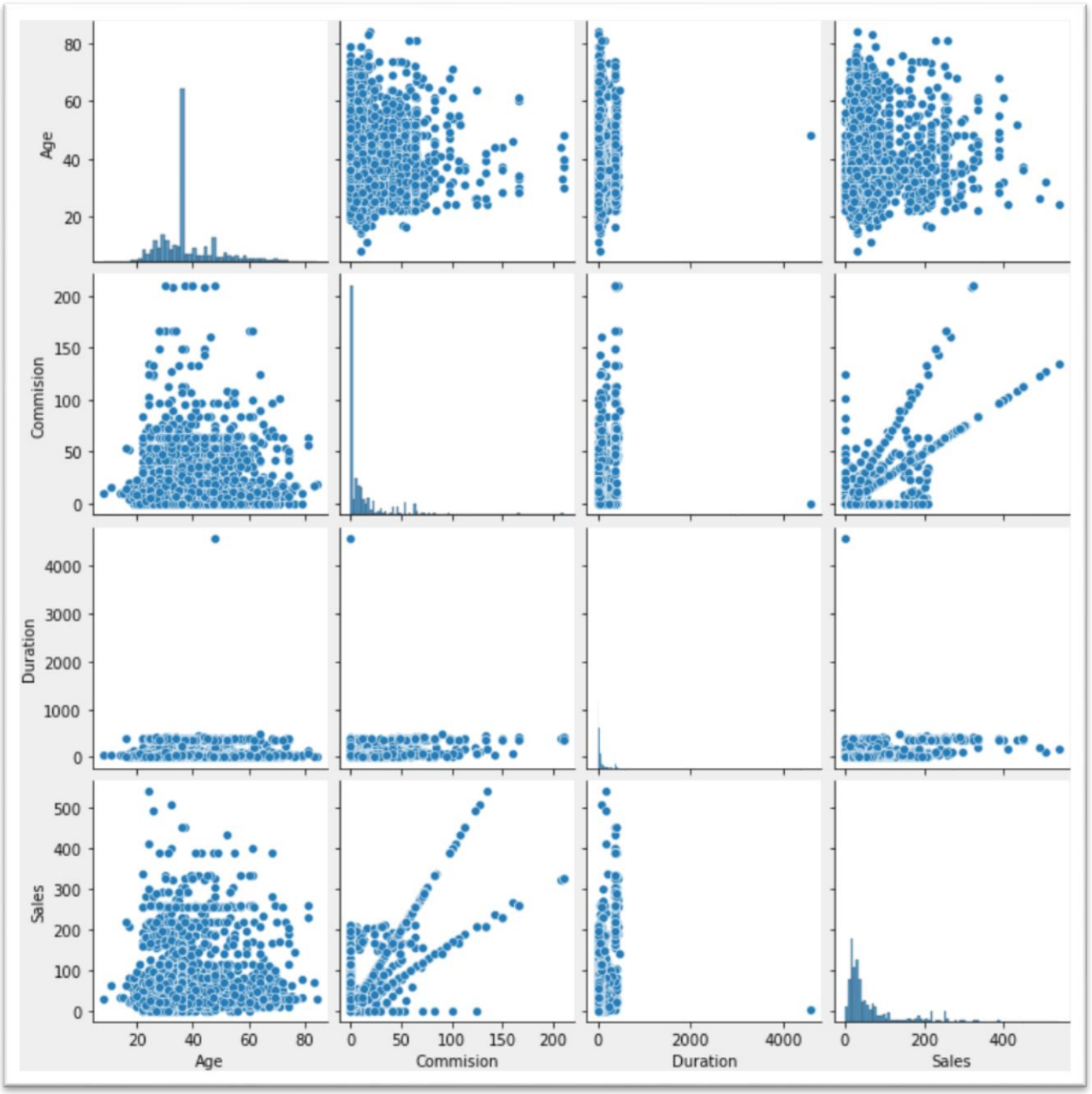


Graph 2.22 Box Plot



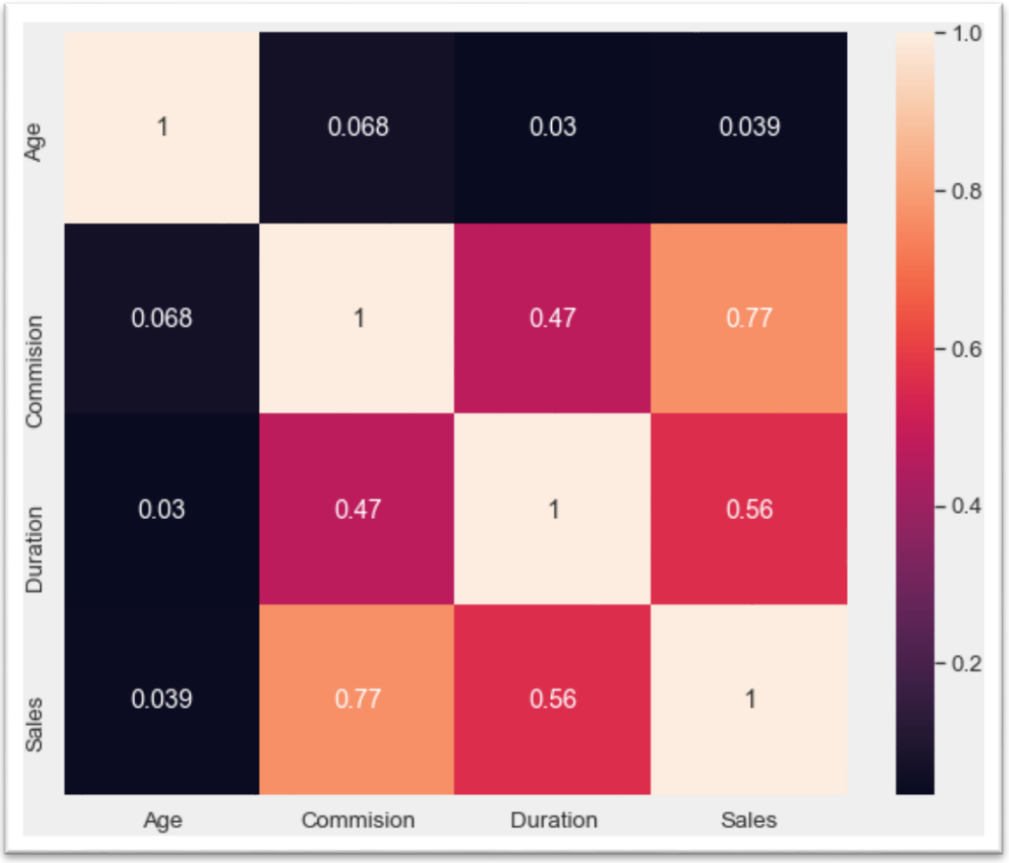
Graph 2.23 Swarm Plot

Checking pairwise distribution of the continuous variables:



Graph 2.24 Checking pairwise distribution of the continuous variables

Checking for Correlations:



Graph 2.25 Heat Map

Converting all objects to categorical codes:

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]

Data information after converting object to categorical variable:

```
<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
```

Table 2.13 Data information after converting object to categorical variable

Modified Data Set:

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

Table 2.14 Modified Data Set

Proportions with 0's and 1's:

```
0    0.692
1    0.308
Name: Claimed, dtype: float64
```

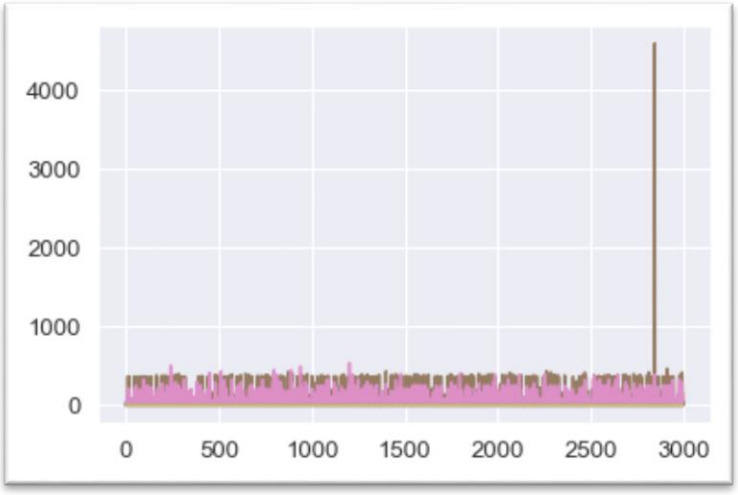
Table 2.15 Proportions with 0's and 1's

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

	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

Table 2.16 Dropping the claimed segment

Prior to scaling:

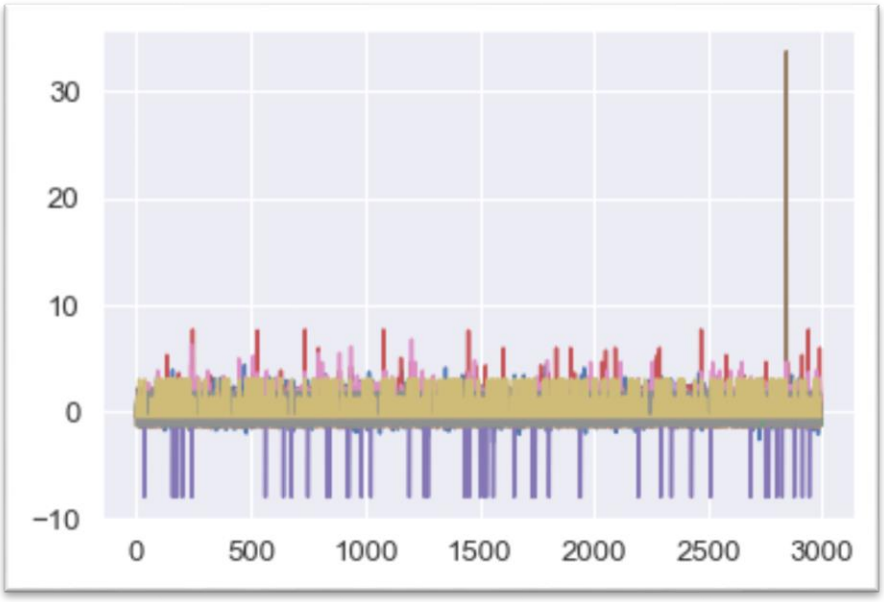


Graph 2.26 Prior to scaling

Scaled Table set:

	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

Table 2.17 Scaled Table set



Graph 2.27 Scaled graph

Splitting data into training and test set:

```
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=5)
```

Formula 2.1 Predicting on Training and Test dataset

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

Table 2.18 Checking the dimensions of the training and test data

Building a Decision Tree Classifier:

```
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(random_state=1)

grid_search_dtcl = GridSearchCV(estimator = dtcl, param_grid = param_grid_dtcl, cv = 10)
```

```
grid_search_dtcl.fit(X_train, train_labels)
print(grid_search_dtcl.best_params_)
best_grid_dtcl = grid_search_dtcl.best_estimator_
best_grid_dtcl
#{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 450}
```

```
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 450}

DecisionTreeClassifier(max_depth=10, min_samples_leaf=50, min_samples_split=450,
                       random_state=1)
```

Table 2.19 Building a Decision Tree Classifier

```
param_grid_dtcl = {
    'criterion': ['gini'],
    'max_depth': [3, 5, 7, 10,12],
    'min_samples_leaf': [20,30,40,50,60],
    'min_samples_split': [150,300,450],
}

dtcl = DecisionTreeClassifier(random_state=1)

grid_search_dtcl = GridSearchCV(estimator = dtcl, param_grid = param_grid_dtcl, cv = 10)
```

```
grid_search_dtcl.fit(X_train, train_labels)
print(grid_search_dtcl.best_params_)
best_grid_dtcl = grid_search_dtcl.best_estimator_
best_grid_dtcl
#{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 450}
```

```
{'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 20, 'min_samples_split': 150}

DecisionTreeClassifier(max_depth=5, min_samples_leaf=20, min_samples_split=150,
                       random_state=1)
```

Table 2.20 Building a Decision Tree Classifier

```
param_grid_dtcl = {
    'criterion': ['gini'],
    'max_depth': [3.5,4.0,4.5, 5.0,5.5],
    'min_samples_leaf': [40, 42, 44,46,48,50,52,54],
    'min_samples_split': [250, 270, 280, 290, 300,310],
}

dtcl = DecisionTreeClassifier(random_state=1)

grid_search_dtcl = GridSearchCV(estimator = dtcl, param_grid = param_grid_dtcl, cv = 10)
```

```
grid_search_dtcl.fit(X_train, train_labels)
print(grid_search_dtcl.best_params_)
best_grid_dtcl = grid_search_dtcl.best_estimator_
best_grid_dtcl
#{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 450}
```

```
{'criterion': 'gini', 'max_depth': 3.5, 'min_samples_leaf': 44, 'min_samples_split': 250}

DecisionTreeClassifier(max_depth=3.5, min_samples_leaf=44,
                       min_samples_split=250, random_state=1)
```

Table 2.21 Building a Decision Tree Classifier

```
param_grid_dtcl = {
    'criterion': ['gini'],
    'max_depth': [4.85, 4.90,4.95, 5.0,5.05,5.10,5.15],
    'min_samples_leaf': [40, 41, 42, 43, 44],
    'min_samples_split': [150, 175, 200, 210, 220, 230, 240, 250, 260, 270],
}

dtcl = DecisionTreeClassifier(random_state=1)

grid_search_dtcl = GridSearchCV(estimator = dtcl, param_grid = param_grid_dtcl, cv = 10)
```

```
grid_search_dtcl.fit(X_train, train_labels)
print(grid_search_dtcl.best_params_)
best_grid_dtcl = grid_search_dtcl.best_estimator_
best_grid_dtcl
#{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 450}
```

```
{'criterion': 'gini', 'max_depth': 4.85, 'min_samples_leaf': 44, 'min_samples_split': 260}

DecisionTreeClassifier(max_depth=4.85, min_samples_leaf=44,
                       min_samples_split=260, random_state=1)
```

Table 2.22 Building a Decision Tree Classifier

Generating Tree:

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

tree_regularized.close()
dot_data
```

```
## http://webgraphviz.com/   (check for the graph over here with this website)
```

Table 2.23 Generating Tree

Variable Importance - DTCL:

	Imp
Agency_Code	0.634112
Sales	0.220899
Product Name	0.086632
Commision	0.021881
Age	0.019940
Duration	0.016536
Type	0.000000
Channel	0.000000
Destination	0.000000

Table 2.24 Variable Importance - DTCL

Predicting on Training and Test dataset:

```
ytrain_predict_dtcl = best_grid_dtcl.predict(X_train)
ytest_predict_dtcl = best_grid_dtcl.predict(X_test)
```

Formula 2.2 Predicting on Training and Test dataset

Getting the Predicted Classes and Probs:

	0	1
0	0.697947	0.302053
1	0.979452	0.020548
2	0.921171	0.078829
3	0.510417	0.489583
4	0.921171	0.078829

Table 2.25 Getting the Predicted Classes and Probs

Building a Random Forest Classifier:

param_grid_rfcl = {'max_depth': [5,10,15], #20,30,40 'max_features': [4,5,6,7], ## 7,8,9 'min_samples_leaf': [10,50,70], ## 50,100 'min_samples_split': [30,50,70], ## 60,70 'n_estimators': [200, 250,300] ## 100,200 }
rfcl = RandomForestClassifier(random_state=1)
grid_search_rfcl = GridSearchCV(estimator = rfcl, param_grid = param_grid_rfcl, cv = 5)
grid_search_rfcl.fit(X_train, train_labels) print(grid_search_rfcl.bestparams) best_grid_rfcl =grid_search_rfcl.bestestimator best_grid_rfcl

```
param_grid_rfcl = {
    'max_depth': [4,5,6],#20,30,40
    'max_features': [2,3,4,5],## 7,8,9
    'min_samples_leaf': [8,9,11,15],## 50,100
    'min_samples_split': [46,50,55], ## 60,70
    'n_estimators': [290,350,400] ## 100,200
}

rfcl = RandomForestClassifier(random_state=1)

grid_search_rfcl = GridSearchCV(estimator = rfcl, param_grid = param_grid_rfcl, cv = 5)
```

```
grid_search_rfcl.fit(X_train, train_labels)
print(grid_search_rfcl.best_params_)
best_grid_rfcl = grid_search_rfcl.best_estimator_
best_grid_rfcl
#{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 50, 'min_samples_split': 450}
```

```
{'max_depth': 6, 'max_features': 3, 'min_samples_leaf': 8, 'min_samples_split': 46, 'n_estimators': 350}

RandomForestClassifier(max_depth=6, max_features=3, min_samples_leaf=8,
                       min_samples_split=46, n_estimators=350, random_state=1)
```

Table 2.26 Getting the Predicted Classes and Probs

Predicting the Training and Testing data:

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

Formula 2.3 Predicting on Training and Test dataset

Getting the Predicted Classes and Probs:

	0	1
0	0.778010	0.221990
1	0.971910	0.028090
2	0.904401	0.095599
3	0.651398	0.348602
4	0.868406	0.131594

Table 2.27 Getting the Predicted Classes and Probs

Variable Importance via RF:

	Imp
Agency_Code	0.276015
Product Name	0.235583
Sales	0.152733
Commision	0.135997
Duration	0.077475
Type	0.071019
Age	0.039503
Destination	0.008971
Channel	0.002705

Table 2.28 Variable Importance via RF

Building a Neural Network Classifier:

```
param_grid_nncl = {
    'hidden_layer_sizes': [50,100,200], # 50, 200
    'max_iter': [2500,3000,4000], #5000,2500
    'solver': ['adam'], #sgd
    'tol': [0.01],
}

nncl = MLPClassifier(random_state=1)

grid_search_nncl = GridSearchCV(estimator = nncl, param_grid = param_grid_nncl, cv = 10)
```

```
grid_search_nncl.fit(X_train, train_labels)
grid_search_nncl.best_params_
best_grid_nncl = grid_search_nncl.best_estimator_
best_grid_nncl
```

```
MLPClassifier(hidden_layer_sizes=200, max_iter=2500, random_state=1, tol=0.01)
```

Table 2.29 Building a Neural Network Classifier

Predicting the Training and Testing data:

```
ytrain_predict_nncl = best_grid_nncl.predict(X_train)
ytest_predict_nncl = best_grid_nncl.predict(X_test)
```

Formula 2.4 Predicting on Training and Test dataset

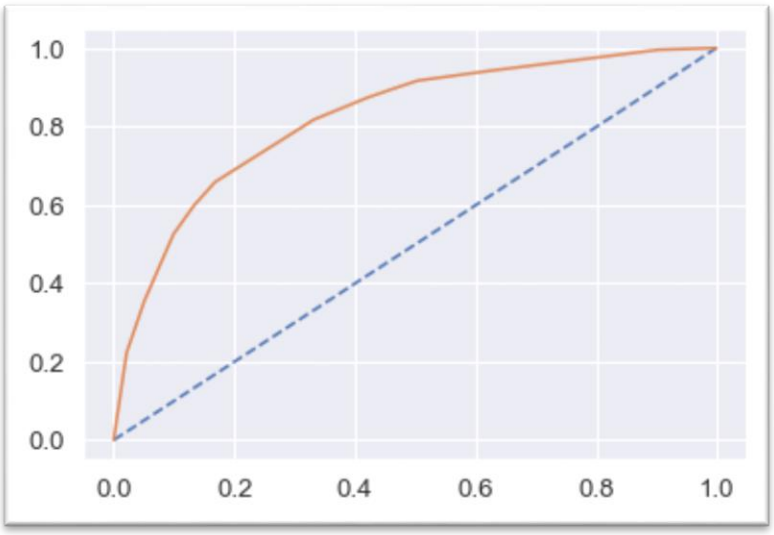
Getting the Predicted Classes and Probs:

	0	1
0	0.822676	0.177324
1	0.933407	0.066593
2	0.918772	0.081228
3	0.688933	0.311067
4	0.913425	0.086575

Table 2.30 Getting the Predicted Classes and Probs

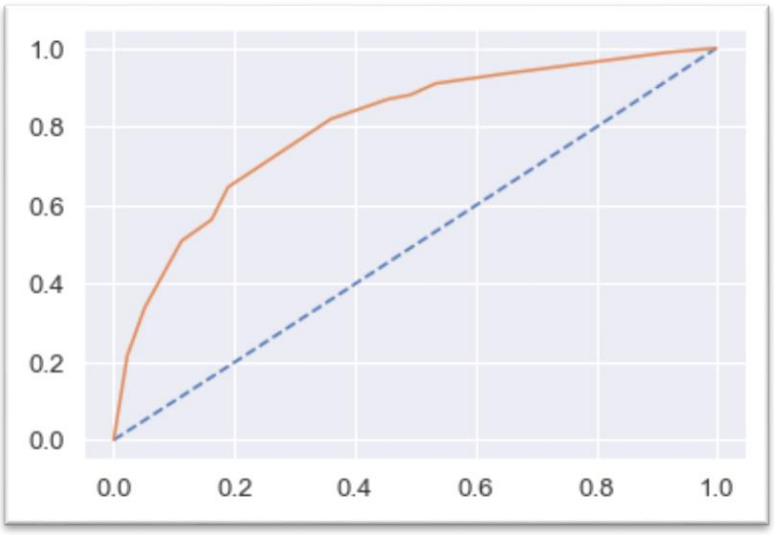
2.3 Performance Metrics: Comment and Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score, classification reports for each model.

CART - AUC and ROC for the training data:



Graph 2.28 CART - AUC and ROC for the training data

CART -AUC and ROC for the test data:



Graph 2.29 CART -AUC and ROC for the test data

CART Confusion Matrix and Classification Report for the training data:

```
array([[1309,  144],
       [ 307,  340]], dtype=int64)
```

Table 2.31 CART Confusion Matrix and Classification Report for the training data

Train Data Accuracy:

0.7852380952380953

	precision	recall	f1-score	support
0	0.81	0.90	0.85	1453
1	0.70	0.53	0.60	647
accuracy			0.79	2100
macro avg	0.76	0.71	0.73	2100
weighted avg	0.78	0.79	0.78	2100

Table 2.32 Train Data Accuracy

```
cart_train_precision  0.7
cart_train_recall  0.53
cart train f1  0.6
```

Table 2.33 Train Data Accuracy

CART Confusion Matrix and Classification Report for the testing data:

```
array([[553,  70],
       [136, 141]], dtype=int64)
```

Table 2.34 CART Confusion Matrix and Classification Report for the testing data

Test Data Accuracy:

0.7711111111111111

	precision	recall	f1-score	support
0	0.80	0.89	0.84	623
1	0.67	0.51	0.58	277
accuracy			0.77	900
macro avg	0.74	0.70	0.71	900
weighted avg	0.76	0.77	0.76	900

Table 2.35 Test Data Accuracy

```
cart_test_precision  0.67
cart_test_recall  0.51
cart_test_f1  0.58
```

Table 2.36 Test Data Accuracy

Cart Conclusion:

Train Data:

- AUC: 82%
- Accuracy: 79%
- Precision: 70%
- f1-Score: 60%

Test Data:

- AUC: 80%
- Accuracy: 77%
- Precision: 80%
- f1-Score: 84%

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

Change is the most important variable for predicting diabetes

Table 2.37 Test Data Accuracy

RF Model Performance Evaluation on Training data:

```
array([[1297,  156],
       [ 255,  392]], dtype=int64)
```

Table 2.38 RF Model Performance Evaluation on Training data

Accuracy:

0.8042857142857143

	precision	recall	f1-score	support
0	0.84	0.89	0.86	1453
1	0.72	0.61	0.66	647
accuracy			0.80	2100
macro avg	0.78	0.75	0.76	2100
weighted avg	0.80	0.80	0.80	2100

Table 2.39 Accuracy

```
rf_train_precision  0.72
rf_train_recall  0.61
rf_train_f1  0.66
```

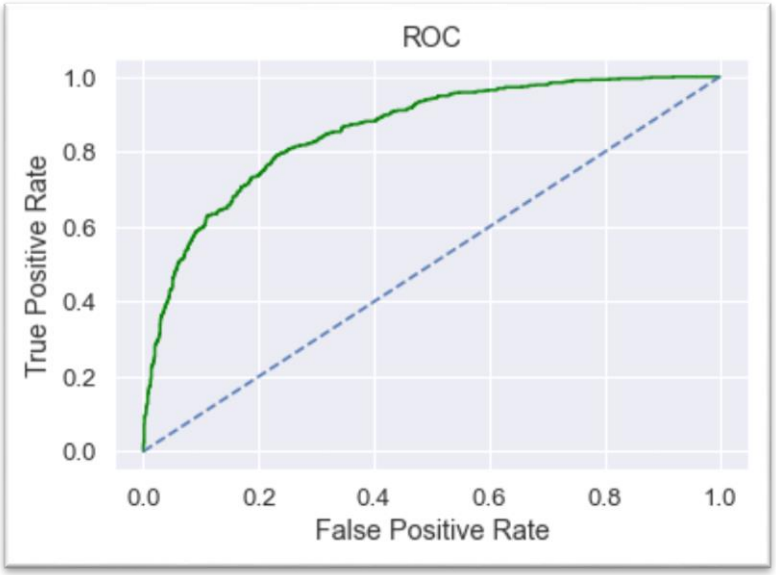
Table 2.40 Accuracy

Area under Curve:

```
rf_train_fpr, rf_train_tpr,_=roc_curve(train_labels,best_grid_rfcl.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_rfcl.predict_proba(X_train)[:,1])
print('Area under Curve is', rf_train_auc)

Area under Curve is 0.8563713512840778
```

Table 2.41 Area under Curve

ROC Curve:



Graph 2.30 ROC Curve

RF Model Performance Evaluation on Test data:

```
array([[550,  73],
       [121, 156]], dtype=int64)
```

Table 2.42 RF Model Performance Evaluation on Test data

Accuracy:

0.7844444444444445

	precision	recall	f1-score	support
0	0.82	0.88	0.85	623
1	0.68	0.56	0.62	277
accuracy			0.78	900
macro avg	0.75	0.72	0.73	900
weighted avg	0.78	0.78	0.78	900

Table 2.43 Accuracy

```
rf_test_precision  0.68
rf_test_recall  0.56
rf test f1  0.62
```

Table 2.44 Accuracy

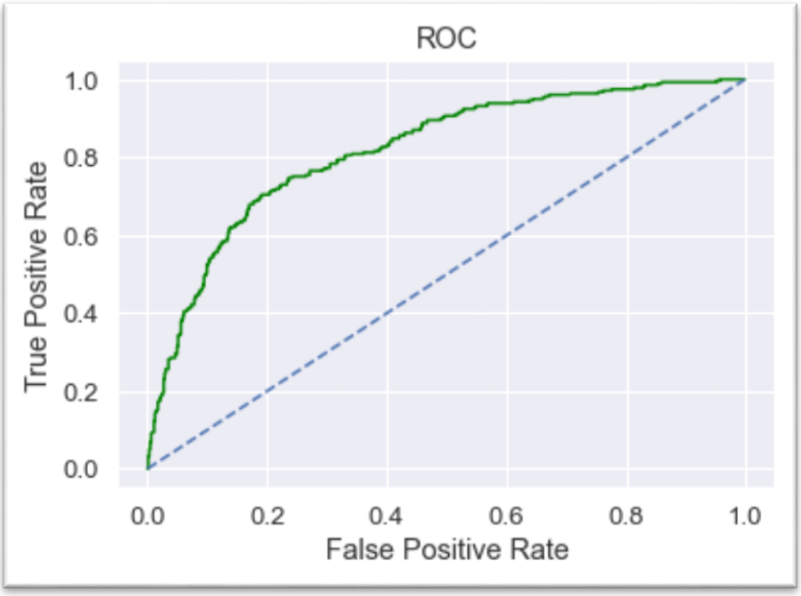
Area under Curve:

```
rf_test_fpr, rf_test_tpr,_=roc_curve(test_labels,best_grid_rfcl.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_rfcl.predict_proba(X_test)[:,1])
print('Area under Curve is', rf_test_auc)
```

```
Area under Curve is 0.8181994657271499
```

Table 2.45 Area under Curve

ROC Curve:



Graph 2.31 ROC Curve

Random Forest Conclusion:

Train Data:

- AUC: 86%
- Accuracy: 80%
- Precision: 72%
- f1-Score: 66%

Test Data:

- AUC: 82%
- Accuracy: 78%
- Precision: 68%
- f1-Score: 62

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

Change is again the most important variable for predicting diabetes

Table 2.46 Random Forest Conclusion

NN Model Performance Evaluation on Training data:

```
array([[1298,  155],
       [ 315,  332]], dtype=int64)
```

Table 2.47 NN Model Performance Evaluation on Training data

Accuracy:

```
nn_train_acc=best_grid_nncl.score(X_train,train_labels)
nn_train_acc
```

```
0.7761904761904762
```

Table 2.48 Accuracy

	precision	recall	f1-score	support
0	0.80	0.89	0.85	1453
1	0.68	0.51	0.59	647
accuracy			0.78	2100
macro avg	0.74	0.70	0.72	2100
weighted avg	0.77	0.78	0.77	2100

Table 2.49 Accuracy

```
nn_train_precision  0.68
nn_train_recall  0.51
nn_train_f1  0.59
```

Table 2.50 Accuracy

Area under Curve:

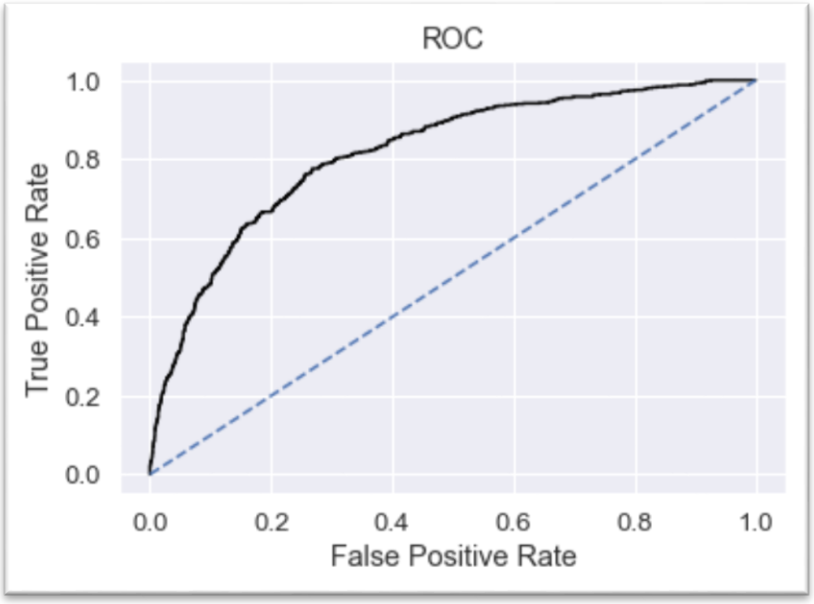
‘’

```
nn_train_fpr, nn_train_tpr,_=roc_curve(train_labels,best_grid_nncl.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_nncl.predict_proba(X_train)[:,1])
print('Area under Curve is', nn_train_auc)
```

```
Area under Curve is 0.8166831721609928
```

Table 2.51 Area under Curve

ROC Curve:



Graph 2.32 ROC Curve

NN Model Performance Evaluation on Test data:

```
array([[553,  70],
       [138, 139]], dtype=int64)
```

Table 2.52 NN Model Performance Evaluation on Test data

Accuracy:

```
nn_test_acc=best_grid_nncl.score(X_test,test_labels)
nn_test_acc
```

```
0.7688888888888888
```

Table 2.53 Accuracy

	precision	recall	f1-score	support
0	0.80	0.89	0.84	623
1	0.67	0.50	0.57	277
accuracy			0.77	900
macro avg	0.73	0.69	0.71	900
weighted avg	0.76	0.77	0.76	900

Table 2.54 Accuracy

```
nn_test_precision  0.67
nn_test_recall  0.5
nn_test_f1  0.57
```

Table 2.55 Accuracy

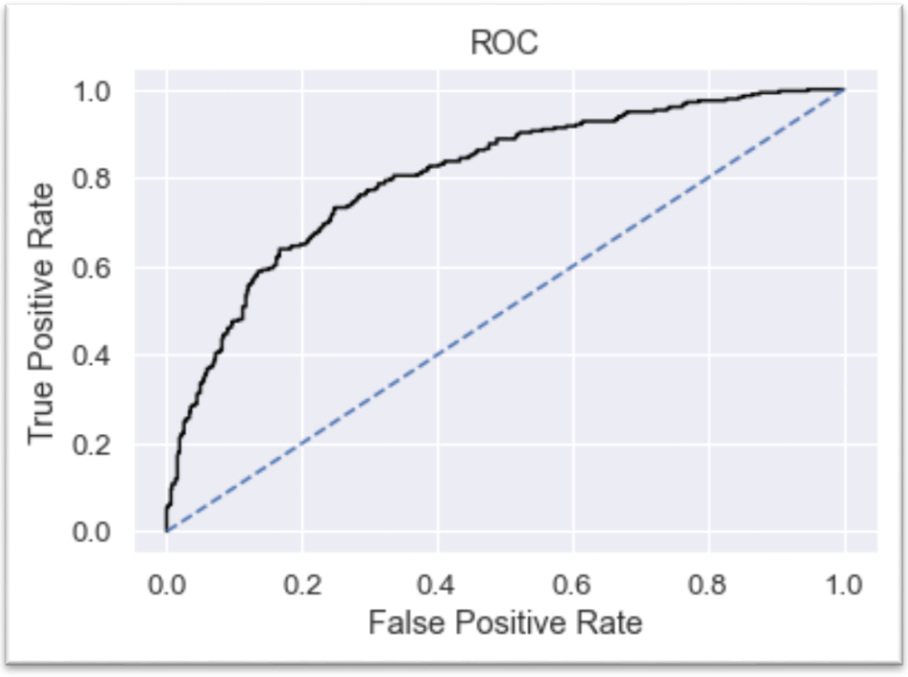
Area under Curve:

```
nn_test_fpr, nn_test_tpr,_=roc_curve(test_labels,best_grid_nncl.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_nncl.predict_proba(X_test)[:,1])
print('Area under Curve is', nn_test_auc)
```

```
Area under Curve is 0.8044225275393896
```

Table 2.56 Area under Curve

ROC Curve:



Graph 2.33 ROC Curve

Neural Network Conclusion:

Train Data:

- AUC: 82%
- Accuracy: 78%
- Precision: 68%
- f1-Score: 59

Test Data:

- AUC: 80%
- Accuracy: 77%
- Precision: 67%
- f1-Score: 57%

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

Table 2.57 Neural Network Conclusion

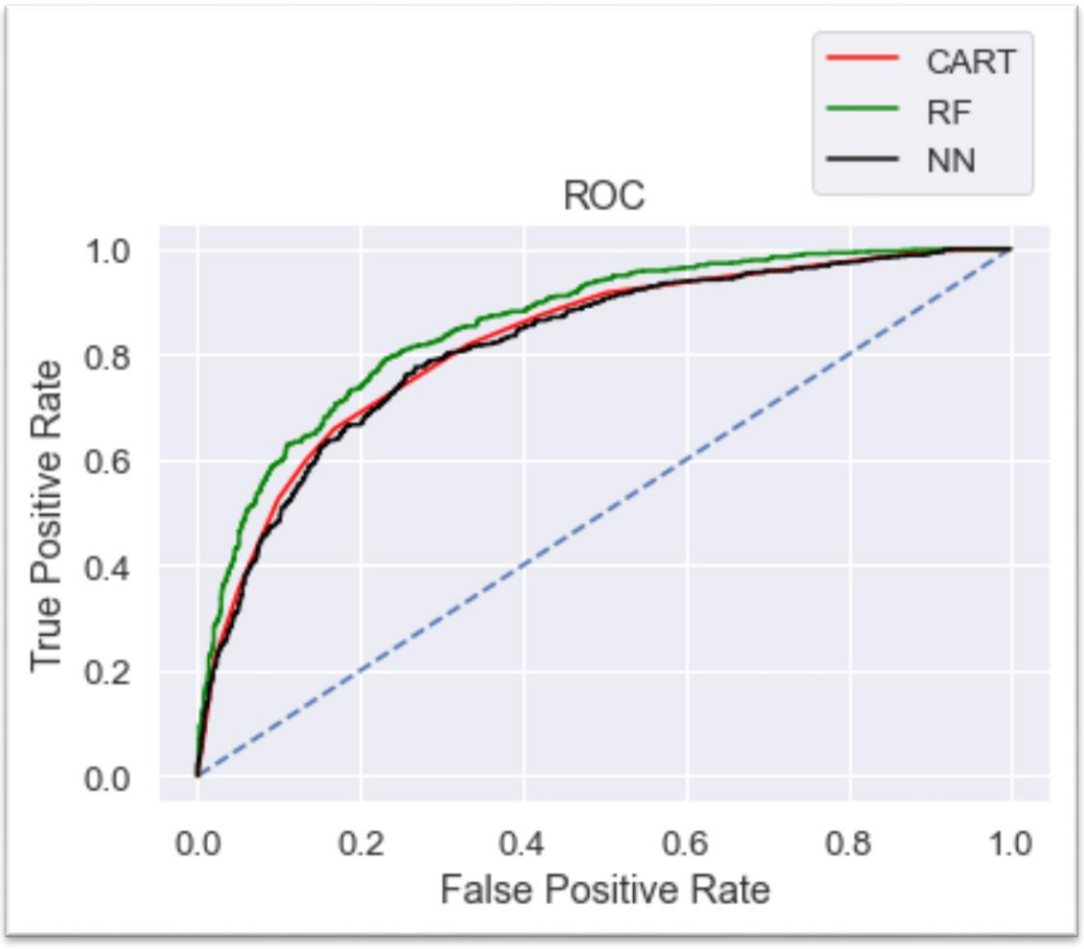
2.4 Final Model: Compare all the models and write an inference which model is best/optimized.

Combined Tabular Data Interpretation:

	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.79	0.77	0.80	0.78	0.78	0.77
AUC	0.82	0.80	0.86	0.82	0.82	0.80
Recall	0.53	0.51	0.61	0.56	0.51	0.50
Precision	0.70	0.67	0.72	0.68	0.68	0.67
F1 Score	0.60	0.58	0.66	0.62	0.59	0.57

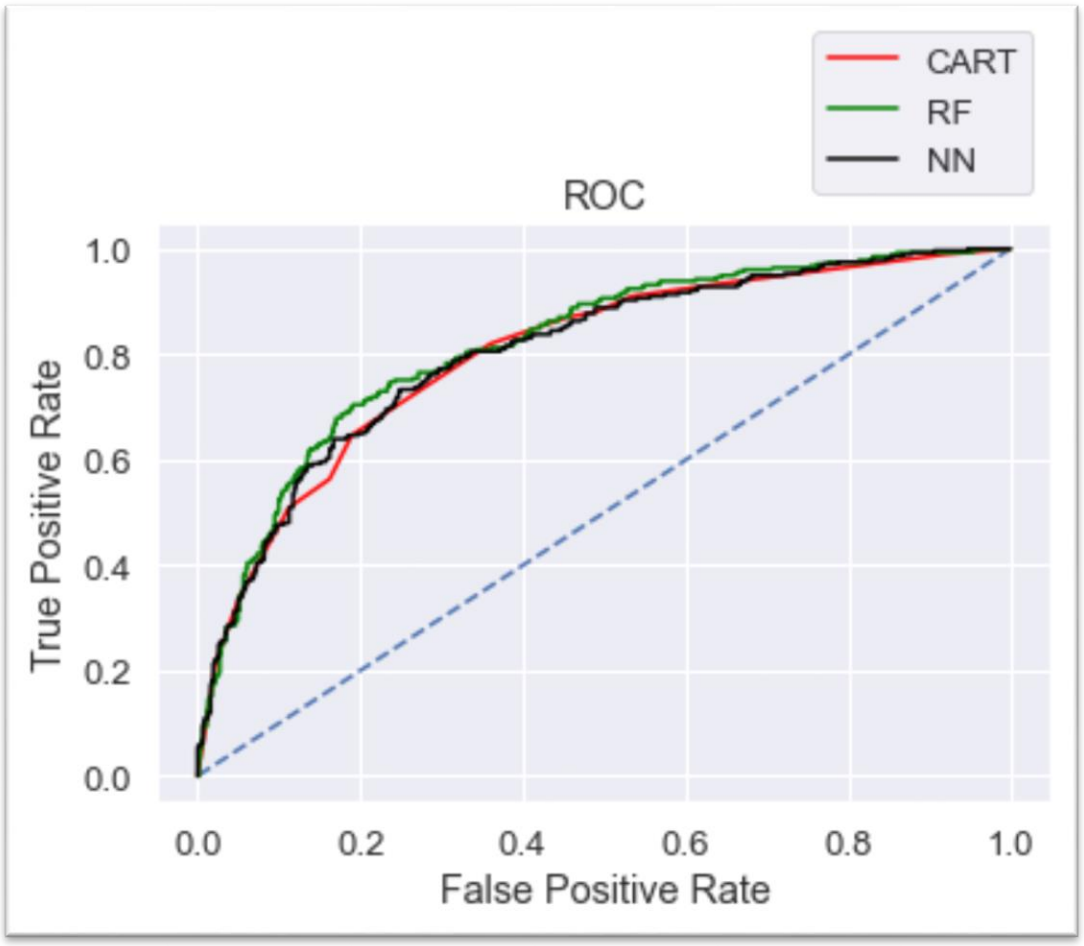
Table 2.58 Combined Tabular Data Interpretation

ROC Curve for the 3 models on the Training data:



Graph 2.34 ROC Curve for the 3 models on the Training data

ROC Curve for the 3 models on the Test data:



Graph 2.35 ROC Curve for the 3 models on the Test data

CONCLUSION:

I am selecting the RF model, as it has better accuracy, precision, recall, f1 score better than other two CART & NN.

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

I strongly recommended we collect more real time unstructured data and past data if possible.
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, behaviour 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?
• Need to train the JZI agency resources to pick up sales as they are in bottom, need to run promotional marketing campaign or evaluate if we need to tie up with alternate agency
• 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.
• Other interesting fact is more sales happen via Agency than Airlines and the trend shows the claim are processed more at Airline. So, we may need to deep dive into the process to understand the workflow and why?
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.