DATA MINING PROJECT REPORT

DSBA

BY,

RAGAVEDHNI K R

Problem 1: Clustering

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

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

- The data is read from '*bank_marketing_part1_Data.csv*' dataset, initial set of rows can be viewed as below.

	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

- The data has 7 columns and 210 rows.

```
Shape of Data: (210, 7)
```

- All the columns are in float64 data type.
- As we can see in the information below, all the columns have 210 records of non 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
```

- The Descriptive Statistics is shown using the 5 point summary.
- The 5 point summary has Count, Mean, Std, Min, 25%, 50%, 75%, Max values calculated for all the columns.
- The Mean and Median (50%) are almost same for all the columns, they are very few outliers present in the data.
- The Mean and Median are fairly close together, hence the all columns are normally distributed (in Bell - Shaped Curve).

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

EXPLORATORY DATA ANALYSIS:

- The data has no null values present in them.

  NULL value check:

```
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
```

- The data has no duplicate records present in them.

  Duplicated Records check:

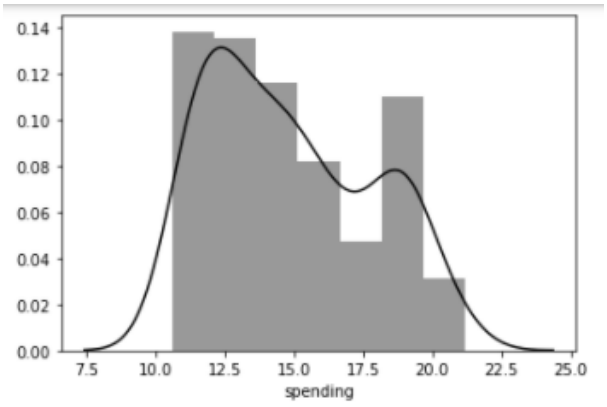
```
The Number of Duplicate records in the data : 0
```

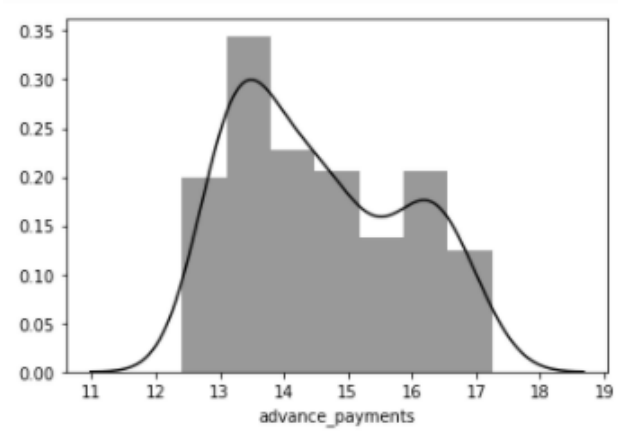
spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping

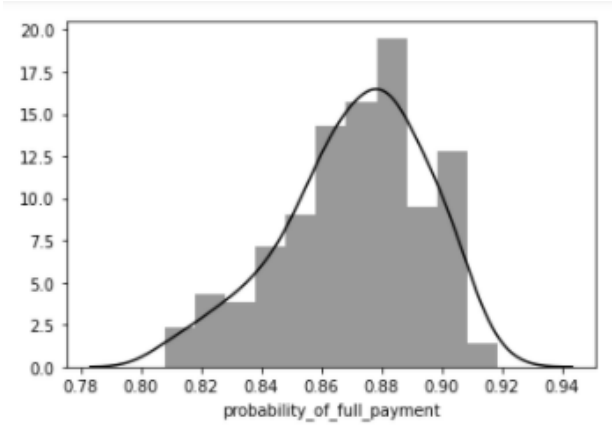
UNIVARIATE ANALYSIS:

- The plots provide information about the distribution of the observations in the single data variable.
- Here we consider the distplot and violinplot (from sns package) to know the distribution of the individual variables from the dataset.
- The right skewed (or positively skewed) distribution has large occurrence in the left side and few in the right side. Here, mean is greater than the median.
- The left skewed (or negatively skewed) distribution has large occurrence in the right side and few in the left side. Here, mean is less than the median
- The symmetric distribution is the bell-shaped or normal distribution.
- Here, the mean is slightly greater than the median. All the variables are Right Skewed (Positively Skewed) distribution.

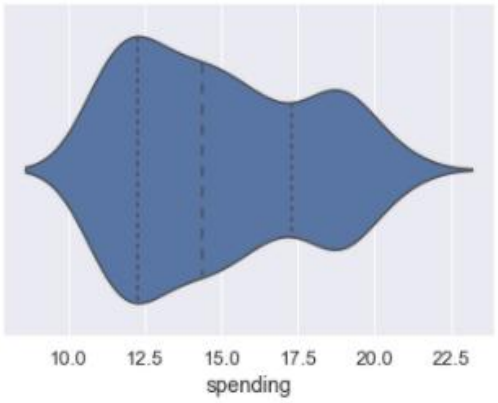
DISTRIBUTION PLOTS:

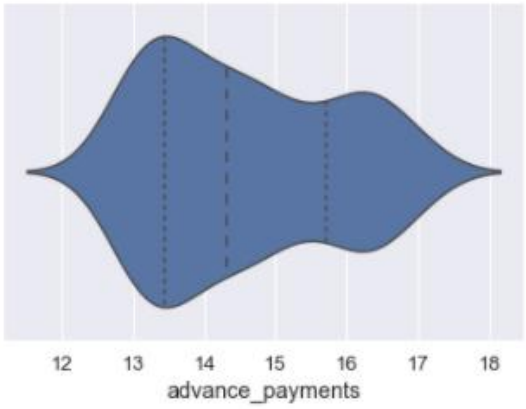


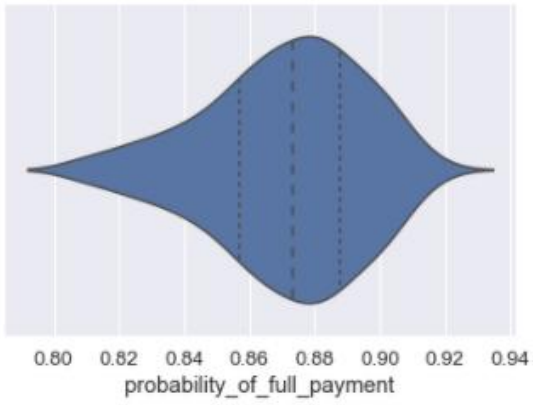


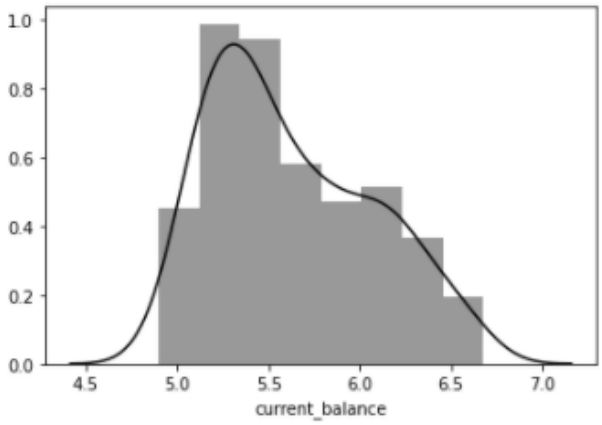


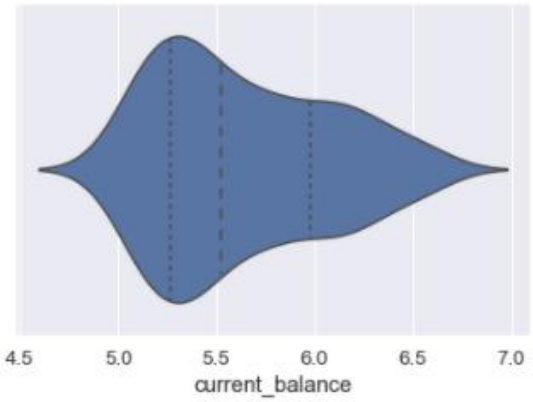
VIOLIN PLOTS:

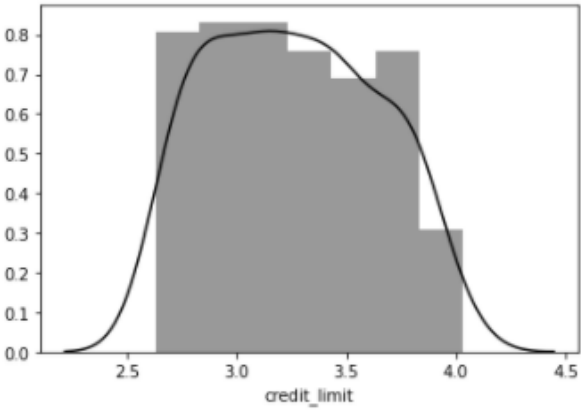


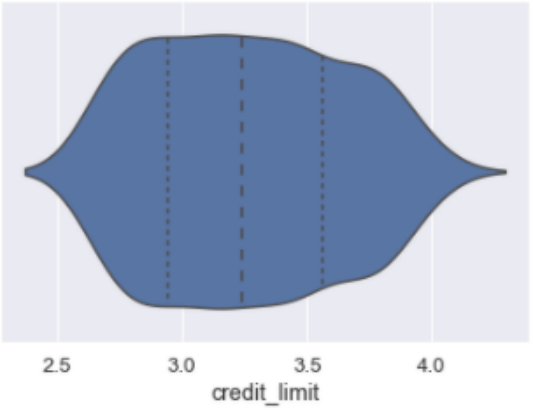


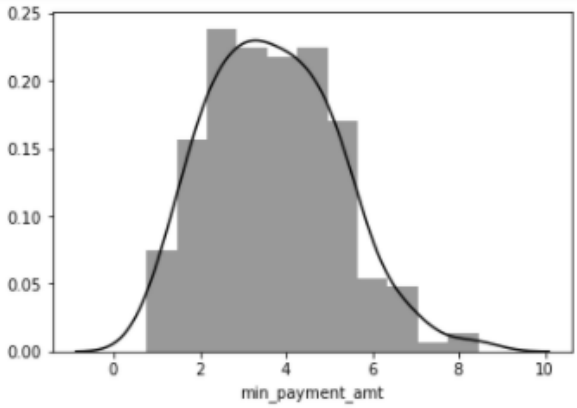


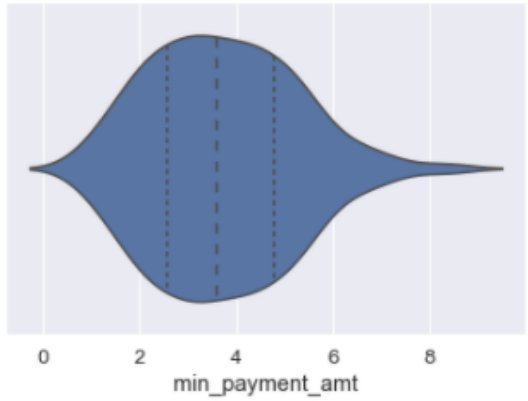


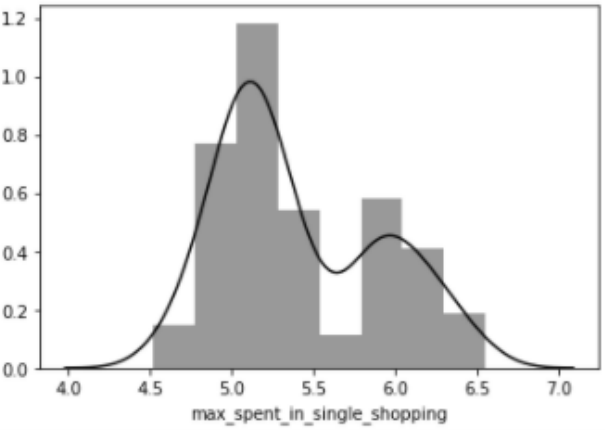


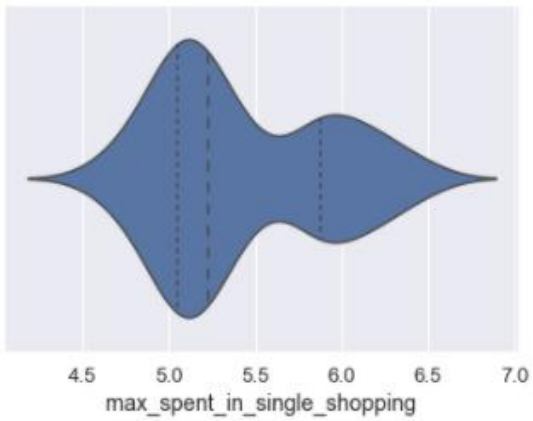










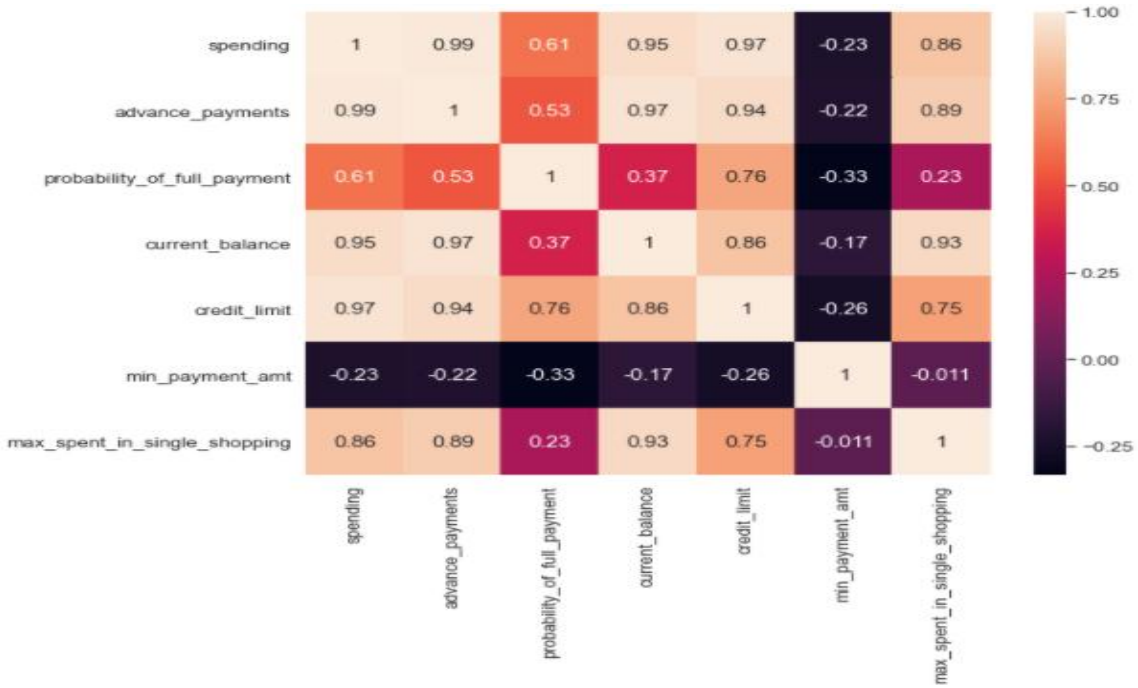


MULTIVARIATE ANALYSIS:

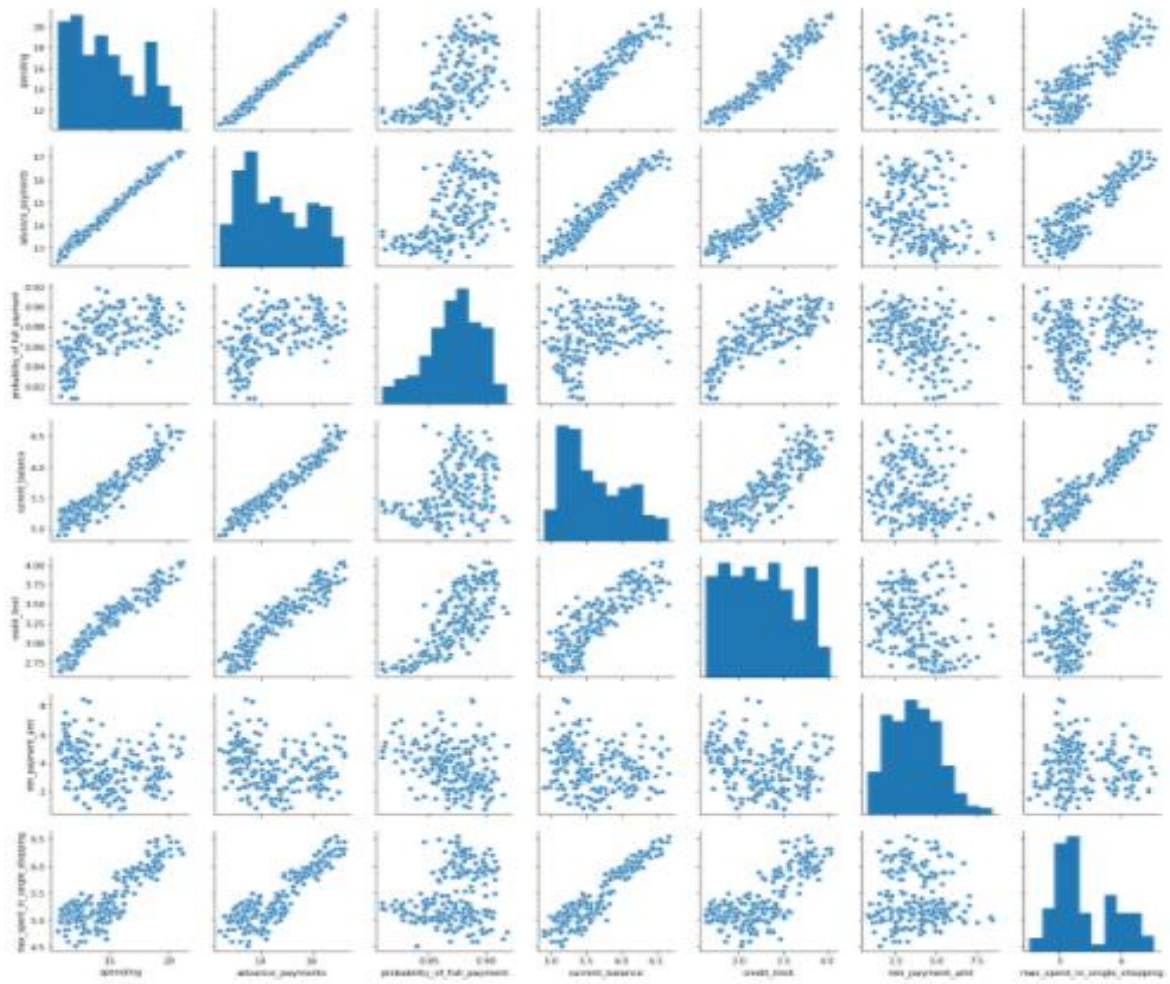
- The correlation across the variables can be found using corr() function in a matrix form.

	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
ability_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
nt_in_single_shopping	0.863693	0.890784	0.226825	0.932806	0.749131	-0.011079	1.000000

- To indicate and visualize the clusters within the data using the heatmap (from sns package).
- There is good correlation (0.99) between spending and advance_payments. The customers paid advance amount nearly equal to their spending amount.
- Next good correlation (0.97) is between advance_payments and current_balance. If the customers had no spending for that month, the advance amount is stored as current balance in his account.
- The next good correlation (0.95) is between spending and current_balance. The customer plans for the spending based on the current balance amount available.
- The min_payment_amt is very less correlated with all other variables. As, only few customers have paid minimum amount while making payments for purchases made monthly.
- We have to look carefully into min_payment_amt variable , as it will be the major deciding factor whether to provide promotional offers to the customers or not.



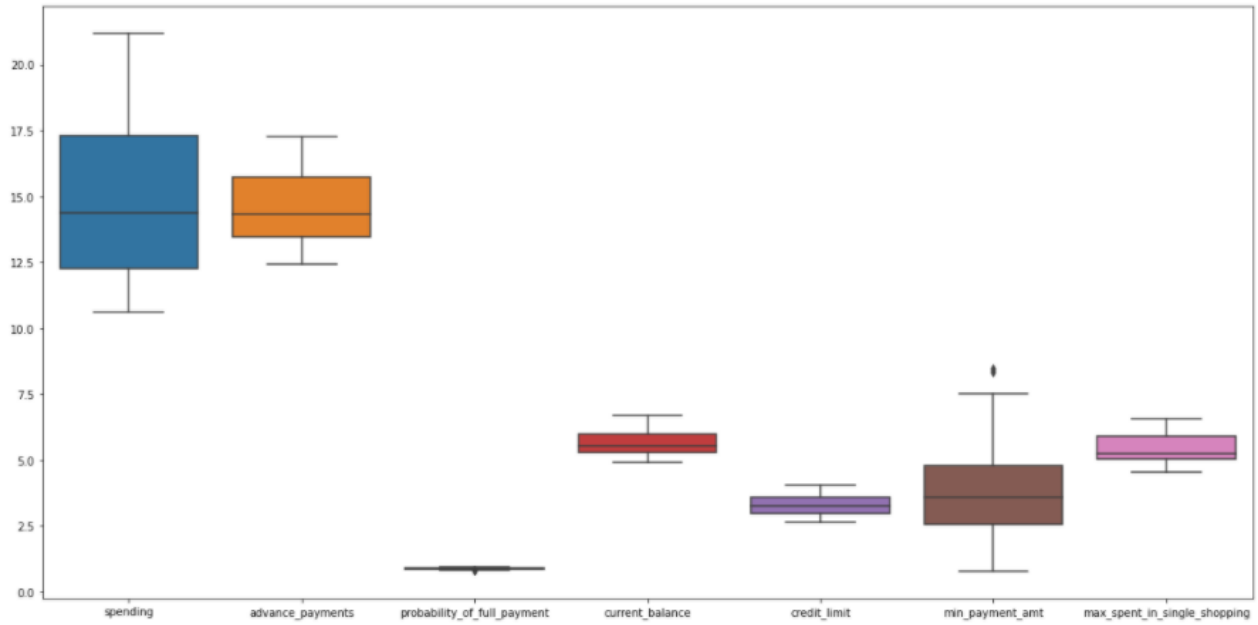
- The pairplot (from sns package) is used for visualizing the pair wise relationship across entire dataframe.
- There is a linear relationship between almost all the variables, mainly between spending and advance_payments.
- There is sparse relation between min_payment_amt and all other variables.



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

- For the columns with different units of measurement, we have to scale them.
- The clustering model is using the distance based model to find similarities between the rows in the table.
- The aim of clustering is to group or cluster the records, which are homogenous within the group and heterogeneous between the groups.
- So scaling the variables will avoid the model giving more importance for the larger magnitude columns.
- We can scale and center the data, while we are unsure of the exact relationship between the attributes for the safer side.
- According to our dataset, the columns are measured in different ranges (units), the column (credit_limit) is measured in 10000s, while other columns are measured in either 100s or 1000s.

- In our case, scaling the data is necessary for the clustering to make the model, give equal importance to all the column variables.



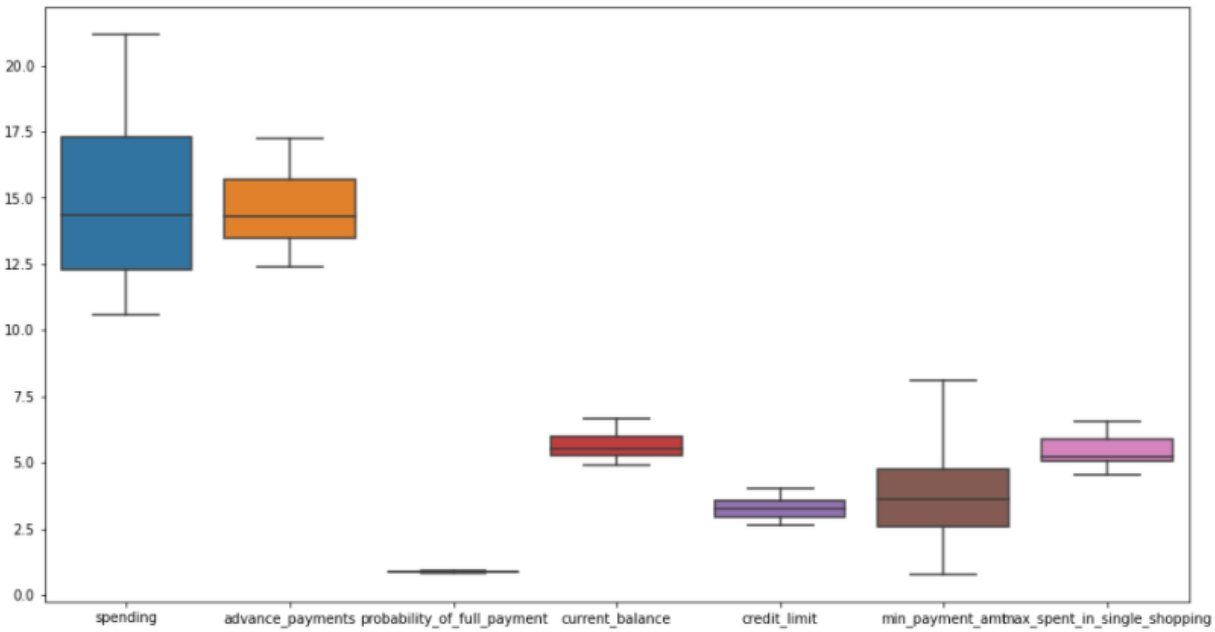
- In Data Preprocessing step, before scaling the variables, the outliers have to be checked, if outliers are present, then they have to be treated or removed.
- We can know the outliers in the data while we club all the variables together as they depend on each other and project the outliers clearly as seen above.
- Our dataset has outliers in **probability_of_full_payment and min_payment_amt** the variables.
- Here, we use IQR (Inter-Quartile Range) imputation method for the outlier treatment.

Before Outlier Treatment

```
(205, 7) (210, 7)
```

After Outlier Treatment (using IQR)

```
(210, 7) (210, 7)
```

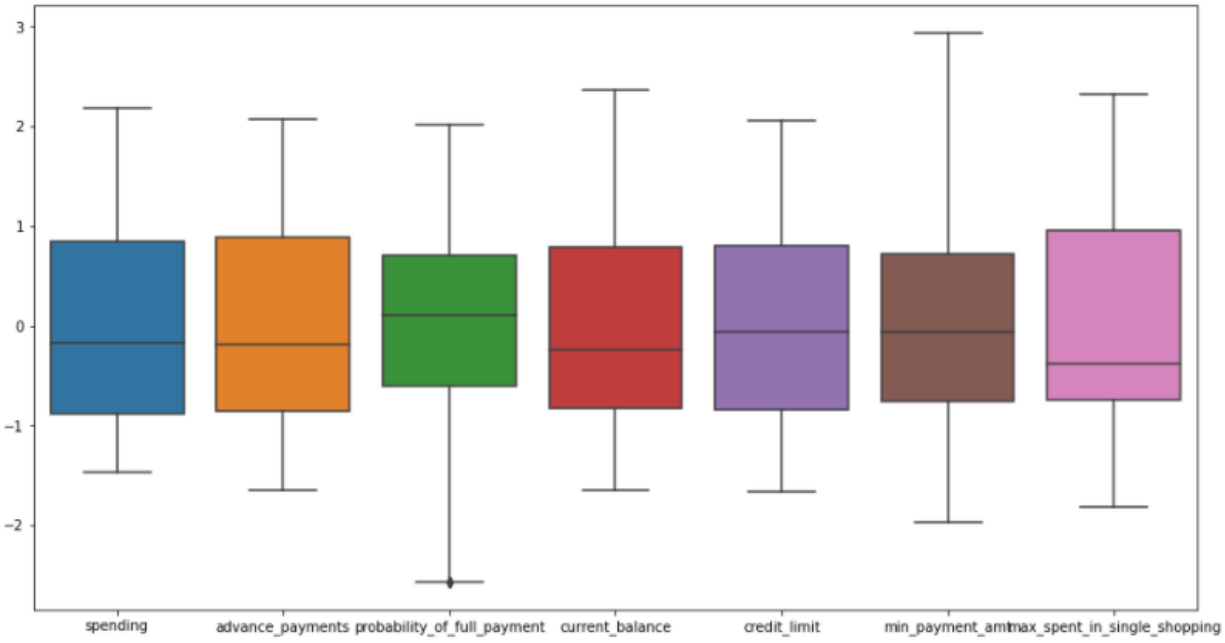


Scaling Data After Outlier Treatment:

- The various scaling techniques are Min-Max scaler, StandardScaler (Z-Score) and Log Transformation.
- The variables are scaled using the Standardization technique (commonly called Z-Score normalization) which has the property of standard normal distribution with mean as 0 and standard deviation as 1.
- The scaled data as a dataframe can be seen below.

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
0	1.754355	1.811968	0.177628	2.367533	1.338579	-0.298625	2.328998
1	0.393582	0.253840	1.505071	-0.600744	0.858236	-0.242292	-0.538582
2	1.413300	1.428192	0.505234	1.401485	1.317348	-0.220832	1.509107
3	-1.384034	-1.227533	-2.571391	-0.793049	-1.639017	0.995699	-0.454961
4	1.082581	0.998364	1.198738	0.591544	1.155464	-1.092656	0.874813

- Visualizing the scaled data in the box plot below.



- The shape does not change after scaling data. Hence, the shape remains same as before.

```
Shape after Scaled data after Outlier Treatment (210, 7) (210, 7)
```

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

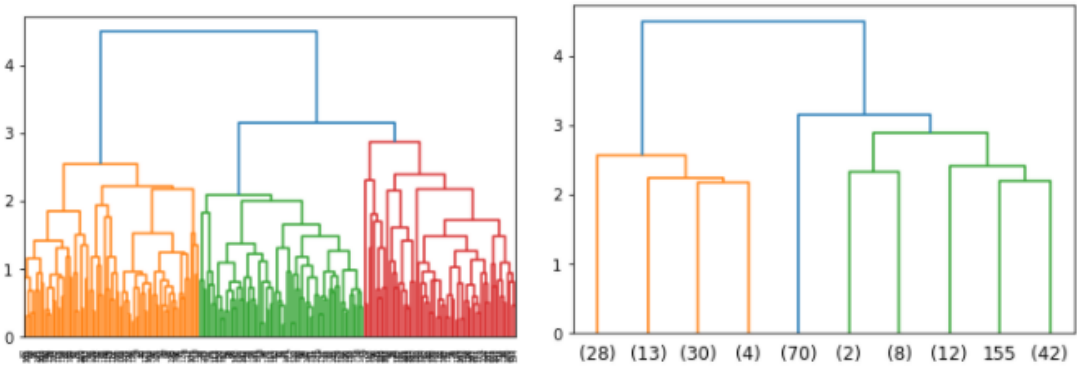
- In Hierarchical clustering, the records are sequentially grouped to create clusters, based on the distance between the records and distance between the clusters.
- The 2 types of Hierarchical clustering are,
  - Agglomerative Clustering
  - Divisive Clustering

- The similar records can be grouped based on the various Distance measures:
  - Euclidean Distance
  - Manhattan Distance
  - Minkowski Distance
  - Chebyshev Distance
  - Hamming Distance
- The similar clusters can be grouped based on the following Linkage types:
  - Single Linkage
  - Complete Linkage
  - Average Linkage
  - Centroid Linkage
  - Wards Linkage
- Using the Dendogram, we can produce the graphical display of clustering process and decide a suitable cut-off.
- Here, we apply the Agglomerative clustering.

AVERAGE LINKAGE METHOD:

USING DENDOGRAM:

- With the help of scipy.cluster.hierarchy package, we use dendogram module.



- We get the above Dendogram, using the *Average Linkage method* on the scaled dataset.
- We can make the cut-off at Y-axis between 2-3 units, to get 3 clusters.

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

- By adding the clusters to our original dataset, we can see that every record falls under a cluster.
- We can group the data by clusters and calculate the frequency of the clusters to understand the pattern of the grouped clusters.

clusters	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

- The Frequencies seems closer and not much significant with each other.

USING AGGLOMERATIVE CLUSTERING:

- With the help of AgglomerativeClustering module in sklearn package, we create the clusters.
- We can set the parameters for the clustering as *n_cluster, linkage type and affinity method.*
- We add the cluster to the original data *as Agglo_Clusters* column.

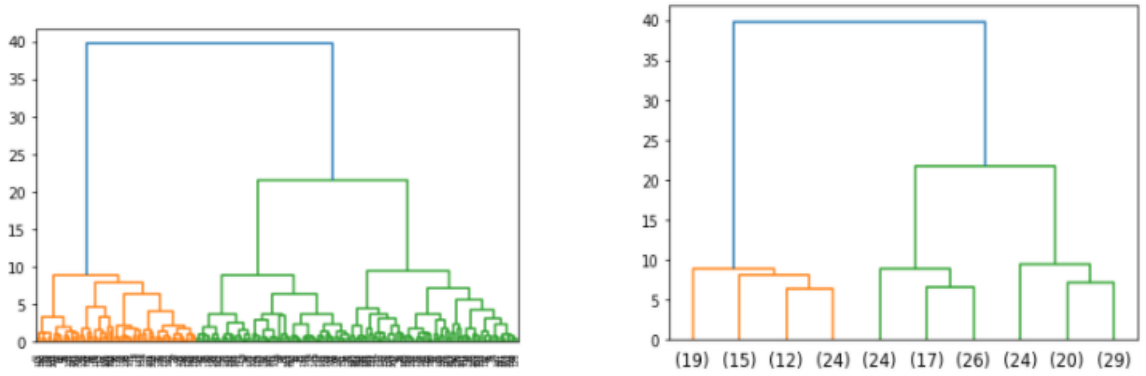
	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Agglo_CLusters
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.550	1
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144	0
2	18.95	16.42	0.8829	6.248	3.755	3.368	6.148	1
3	10.83	12.96	0.8099	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

- We group the data by the mean of the clusters and according to the frequency of the clusters.

Agglo_CLusters	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Freq
0	14.217077	14.195846	0.884869	5.442000	3.253508	2.768418	5.055569	65
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

WARD LINKAGE METHOD:

- We try using the Wards Linkage method on the scaled dataset to get the dendogram.



- The combinations of 2 lines are not joined on the Y-axis from 20 to 40, for about 20 units.
- So, the optimal number of clusters will be 2 for hierarchical clustering.

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	clusters
0	19.94	16.92	0.8752	6.675	3.763	3.252	6.550	1
1	15.99	14.89	0.9064	5.363	3.582	3.336	5.144	2
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

- Adding the clusters as a column to the original data, every record is either in cluster 1 or cluster 2.
- By grouping the clusters according to their frequency, to understand the pattern of grouped records in each cluster.

CLUSTER PROFILES:

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	13.085571	13.766214	0.864298	5.363714	3.045593	3.730723	5.103421	140

- From the above cluster profiling, we can see the customers are grouped based on the patterns of the credit card usage as 2 clusters.
- The Inference we get is:
  - Cluster 1 : Moderate Credit Card Usage
  - Cluster 2 : High Credit Card Usage

USING AGGLOMERATIVE CLUSTERING:

- With the help of AgglomerativeClustering module in sklearn package, we create the clusters.
- The basic parameters to be given are *n_clusters* for number of clusters to be formed, *affinity* for distance metric, *linkage* type for grouping cluster.
- The parameters can be chosen from the dendogram generated above.
  Eg: n_clusters=2, affinity='euclidean', linkage='ward'
- The cluster is added to the original data as *Agglo_Clusters* column.

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

- The records are grouped under 2 clusters. We get the same result as we achieved using the dendogram using the *ward linkage* type and *maxclust* as the criterion for forming flat clusters.
- The records are grouped by the mean of the *Agglo_Clusters* column and from the Freq column we can understand the pattern of grouping of the records.

Agglo_Clusters	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Freq
0	13.085571	13.766214	0.864298	5.363714	3.045593	3.730723	5.103421	140
1	18.371429	16.145429	0.884400	6.158171	3.684629	3.639157	6.017371	70

- From the above clusters, we can see that the customers use their credit cards very frequently or in moderate.
- The Inference we get:
  - Cluster 1 : High Credit Card Usage
  - Cluster 2 : Very Moderate Credit Card Usage

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

- In the Partitioning Clustering (or non-hierarchical clustering) approach, we specify the number of clusters (as ***K***) required as output. Hence called K-Means Clustering.
- The clusters are homogeneous within themselves and heterogeneous between/among them themselves, based on the distance.
- The K-Means by default uses the Euclidean distance method, for finding the distance between every data to its centroid.
- The K-Means Clustering is used for the large dataset applications.

USING K-MEANS CLUSTERING:

- We import the KMeans module from the sklearn package for achieving the KMeans clustering.
- For our dataset, we consider K=2 and fit the scaled dataset to the KMeans object.

```
KMeans(n_clusters=2, random_state=1)
```

- We can see the labels of each point in the data using *labels_* attribute.
- To find the suitable K value for the given dataset, we calculate the Within Sum of Squares (WSS) using the inertia_ attribute.
- When K=2, we get the below WSS.

```
Within Cluster Sum of Squares for 2 clusters:  659.1474009548498
```

- We try calculating the WSS for different K values.
- When K=3, the WSS is

```
Within Cluster Sum of Squares for 3 clusters:  430.298481751223
```

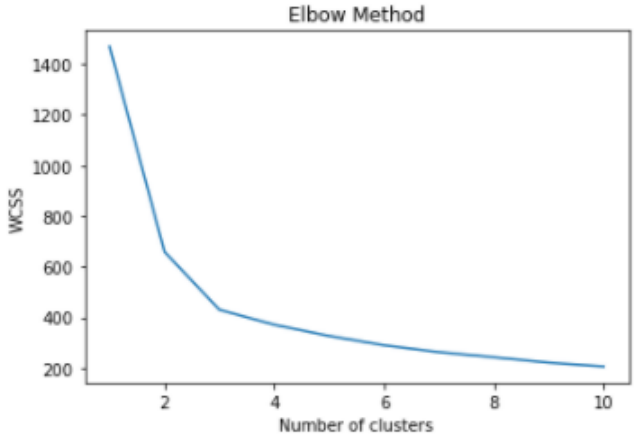
- When K=4, the WSS is

```
Within Cluster Sum of Squares for 4 clusters:  371.221763926848
```

- When K=5, the WSS is

```
Within Cluster Sum of Squares for 5 clusters:  326.8846407681858
```

- The WSS keeps reducing, as K keeps increasing.
- We can plot the WSS values using the Elbow plot or the Distortion plot.



- From the above plot, we need to identify the location of the elbow on the X-axis. Both at X-axis at 2 and X-axis at 3, there is an elbow.
- So, we can consider the K value at K = 2 or K = 3.
- The Silhouette Score can be used to get the optimum K value.
- The *silhouette_score* method uses the scaled data with labels of the data for the required K value to be calculated.
- When K=2, the Silhouette Score is

```
Silhouettte Score for K=2:  0.46560100442748986
```

- When K=3, the Silhouette Score is

```
Silhouettte Score for K=3:  0.4008059221522216
```

- When K=4, the Silhouette Score is

```
Silhouettte Score for K=4:  0.32943733699973826
```

- The optimum K value can be chosen when the Silhouette Score is close to +1. For K=2, the Silhouette score is high comparing with the Silhouette score for other K values.

- Taking K = 2 and adding the labels for the K value as a column (Clus_kmeans) with their respective Silhouette Width (sil_width) to the original dataset.

spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Clus_kmeans2	sil_wid
19.94	16.92	0.8752	6.675	3.763	3.252	6.550	1	0.6033
15.99	14.89	0.9064	5.363	3.582	3.336	5.144	0	0.0090
18.95	16.42	0.8829	6.248	3.755	3.368	6.148	1	0.6773
10.83	12.96	0.8099	5.278	2.641	5.182	5.185	0	0.4969
17.99	15.86	0.8992	5.890	3.694	2.068	5.837	1	0.5470

- The cluster profiling is done by grouping the data by mean of the clusters column and

sorts them by the frequency of every cluster.

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	sil_width	freq
ans2									
0	12.930602	13.693459	0.863577	5.339699	3.025917	3.827444	5.081737	0.439944	133
1	18.158571	16.054805	0.883817	6.127429	3.660519	3.480417	5.971740	0.509917	77

- The entire dataset is split up into 2 clusters as seen above.
- One cluster has customers who use credit cards quite frequently and other cluster of customers does not use credit card much frequently but used them in some cases.

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

❖ Taking the Average Linkage method and doing the cluster profiling as below.

clusters	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

❖ **CLUSTER 1 :** Max payers or Full Payers
❖ **CLUSTER 2 :** Non Payers
❖ **CLUSTER 3 :** Revolvers
❖ The people in Cluster 1 who pay their outstanding credit in full and most responsible kind of people.
❖ Having high spending and maximum spent in single shopping.
❖ The people in the Cluster 2 try to obtain all the credit cards offered and use them all up to the maximum credit limit and not pay up at all.
❖ Maintaining low balance but high spending.
❖ The people in the Cluster 3 do not believe in paying their dues to full and pay minimum amount due and continue their purchasing on cards as usual.
❖ Having equal advance payments and spending, keeping them hand in hand.
❖ The credit companies consider the people from Cluster 3 as the darlings as the companies make money at their expense.
❖ The company must encourage the people from Cluster 1 and Cluster 2 to increase their credit card usage.
❖ They can be provided with the Coupon, Rewards, Refunds.
❖ The company must keep the Cluster 3 peoples by providing them Loyalty Cards and Card Linked Offers.

Problem 2: CART-RF-ANN

2. An Insurance firm providing tour insurance is facing higher claim frequency. The management decides to collect data from the past few years. You are assigned the task to make a model which predicts the claim status and provide recommendations

to management. Use CART, RF & ANN and compare the models' performances in train and test sets.

2.1. Data Ingestion: Read the dataset. Do the descriptive statistics and do null value condition check, write an inference on it.

- The dataset 'insurance_part2_data.csv' is read for the business problem and the initial set if rows can be seen as below.

	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

- The number of Rows and Columns in the dataset can be known using *shape* on the dataset.

```
Shape of Data: (3000, 10)
```

- All the columns have no null value and they are of *int64, object, float64* 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
```

- The Descriptive Statistics on the data can be viewed using describe function with include='all' parameter, where all the columns of the input data will be included in the output.
- It shows the count, unique, top, freq, mean, std, min, 25%, 50% (median), 75%, max values for the respective columns in the dataset.

	Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
count	3000.000000	3000	3000	3000	3000.000000	3000	3000.000000	3000.000000	3000	3000
unique	NaN	4	2	2	NaN	2	NaN	NaN	5	3
top	NaN	EPX	Travel Agency	No	NaN	Online	NaN	NaN	Customised Plan	ASIA
freq	NaN	1365	1837	2076	NaN	2954	NaN	NaN	1136	2465
mean	38.091000	NaN	NaN	NaN	14.529203	NaN	70.001333	60.249913	NaN	NaN
std	10.463518	NaN	NaN	NaN	25.481455	NaN	134.053313	70.733954	NaN	NaN
min	8.000000	NaN	NaN	NaN	0.000000	NaN	-1.000000	0.000000	NaN	NaN
25%	32.000000	NaN	NaN	NaN	0.000000	NaN	11.000000	20.000000	NaN	NaN
50%	36.000000	NaN	NaN	NaN	4.630000	NaN	26.500000	33.000000	NaN	NaN
75%	42.000000	NaN	NaN	NaN	17.235000	NaN	63.000000	69.000000	NaN	NaN
max	84.000000	NaN	NaN	NaN	210.210000	NaN	4580.000000	539.000000	NaN	NaN

- The mean and median for all the columns vary drastically. Hence, we can say that there are outliers present in the data.
- Null check for the columns is done using *isnull() or isna()*.

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

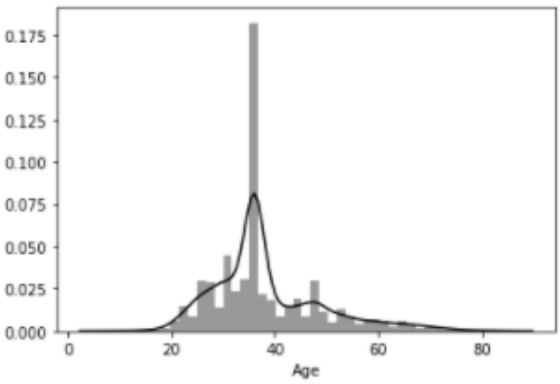
- Duplicates check for the data is done using duplicated() and we see 139 duplicates present in our data.

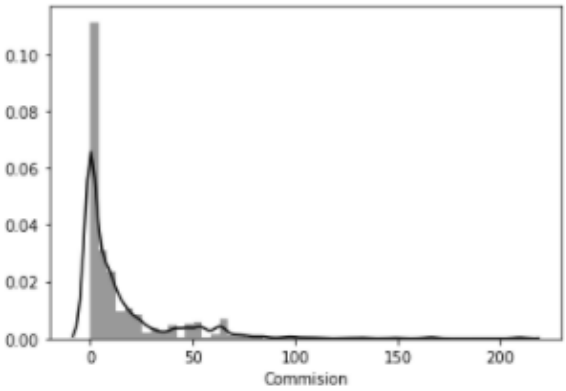
```
Number of duplicates present in the data = 139
Number of rows before discarding duplicates = 3000
Number of rows after discarding duplicates = 2861
```

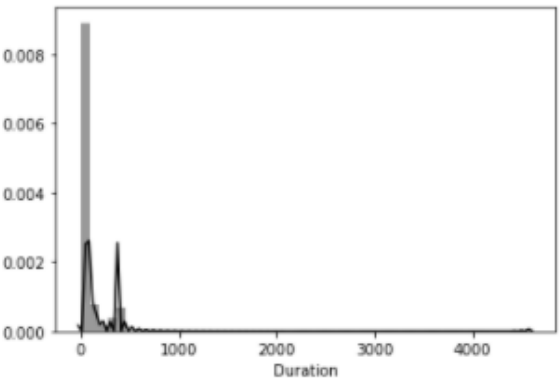
- Checking the data after removing duplicates.

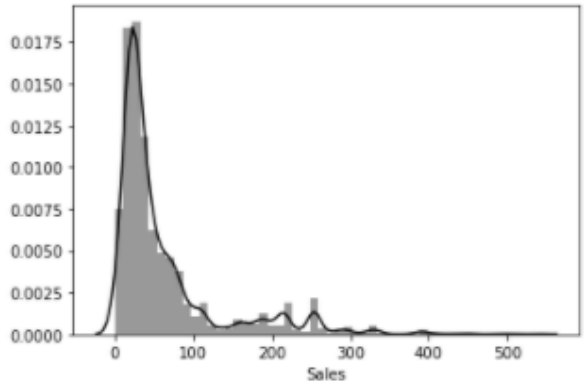
Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination

- For Univariate visualization and analysis, we can use distribution plot for numeric type columns to understand the data distribution pattern.

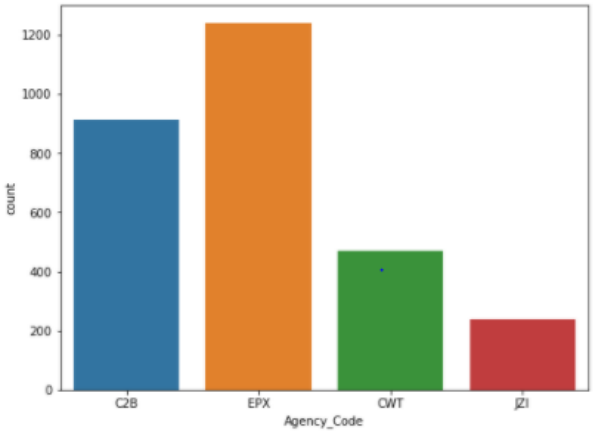


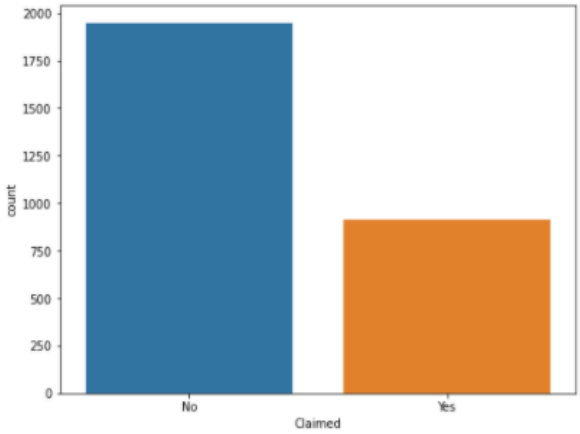


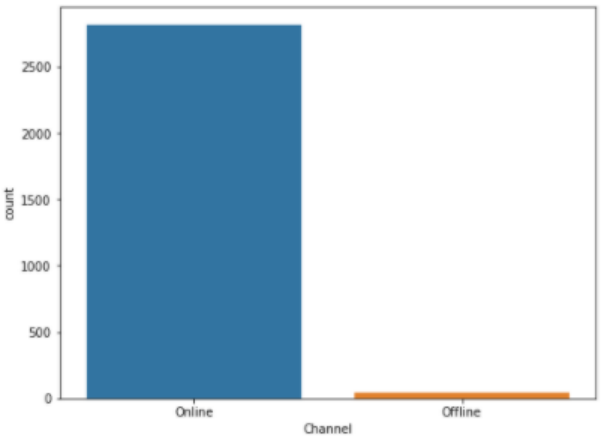


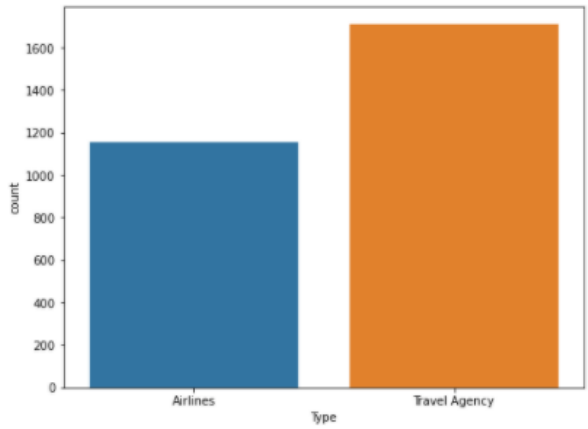


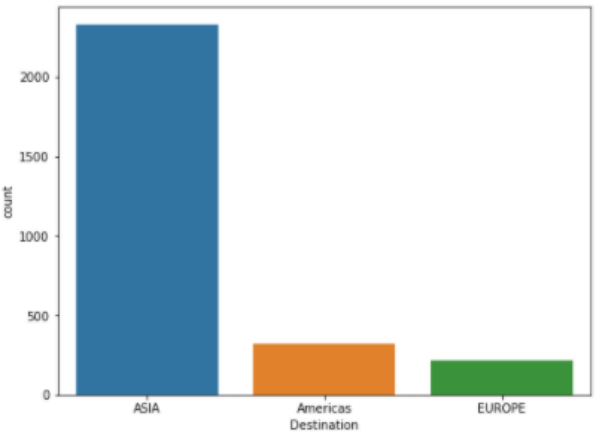
- For the category columns, we can view count plot, box plot or bar plot to know how many categories are there and frequency of each of them in a column.

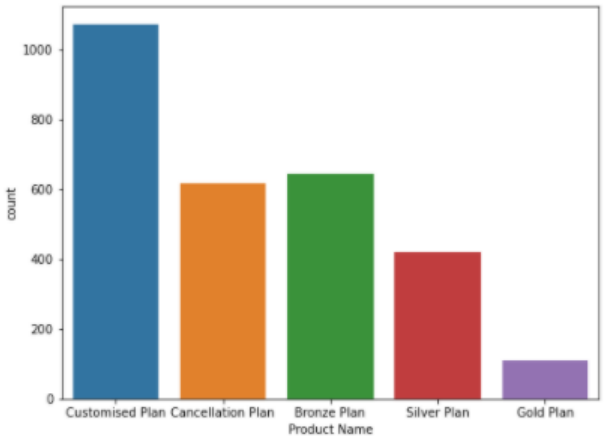






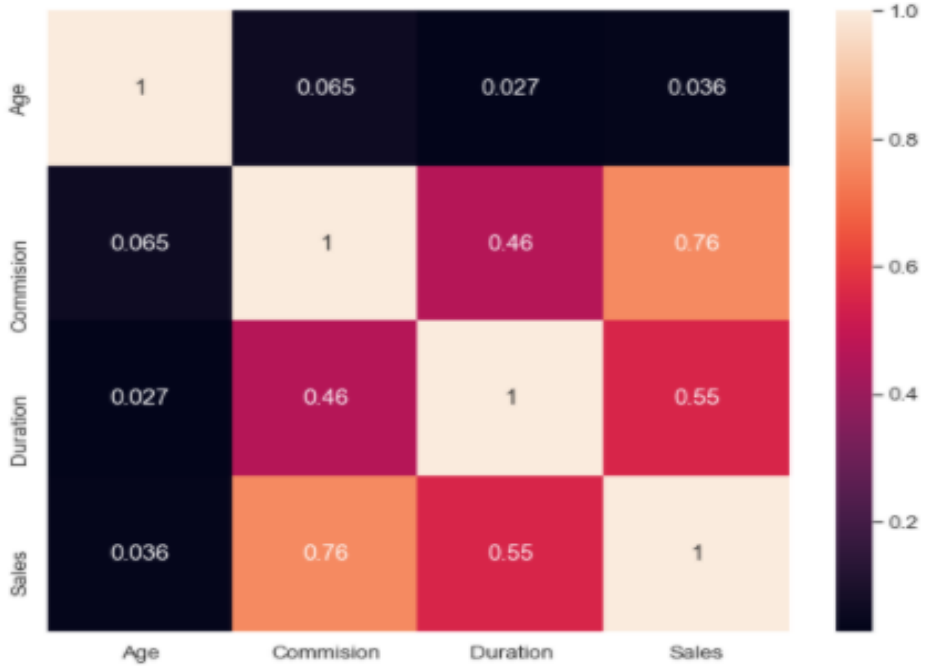




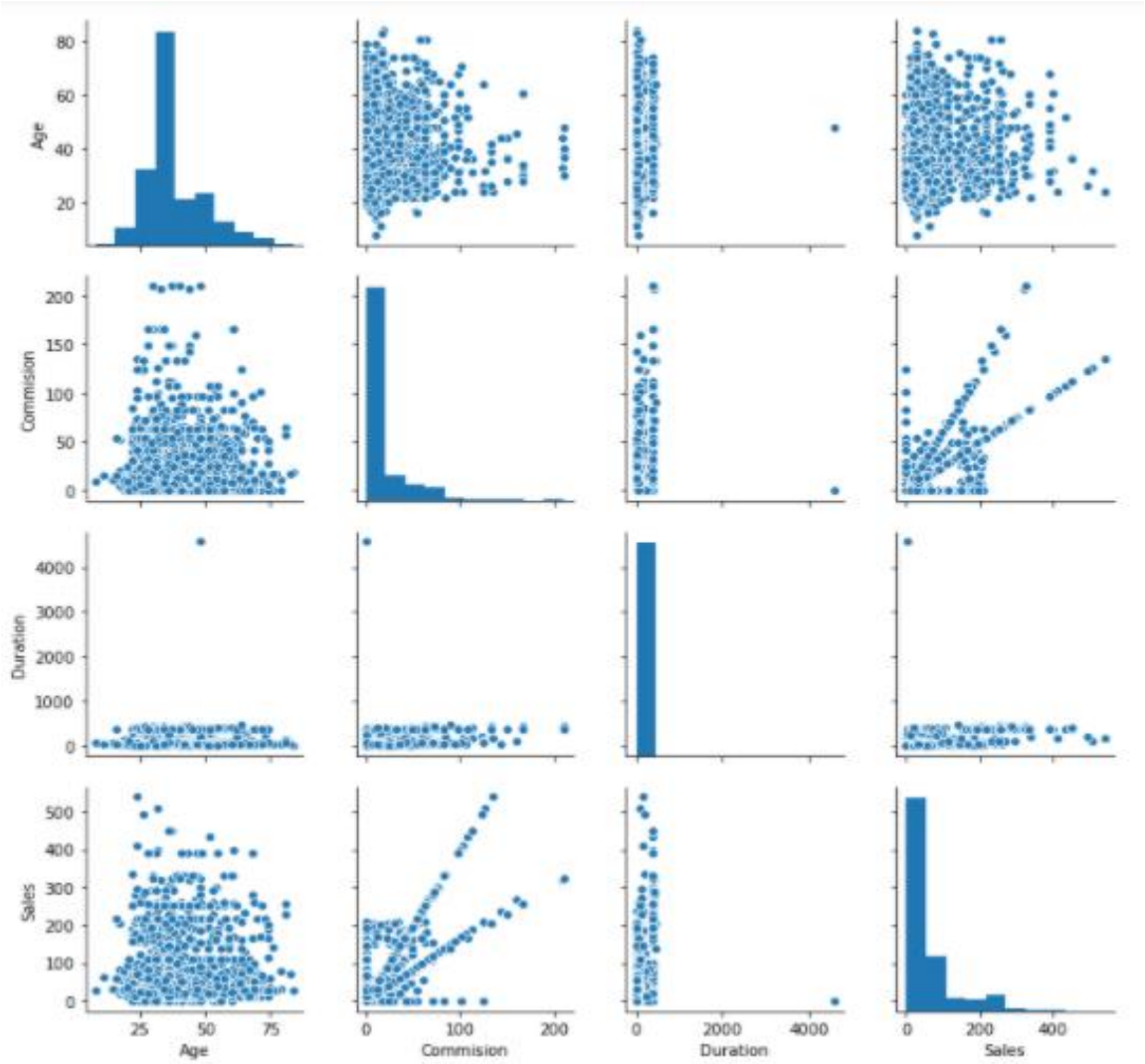


- For Multivariate visualization, we use pair plot which has the columns that are correlated with each other.

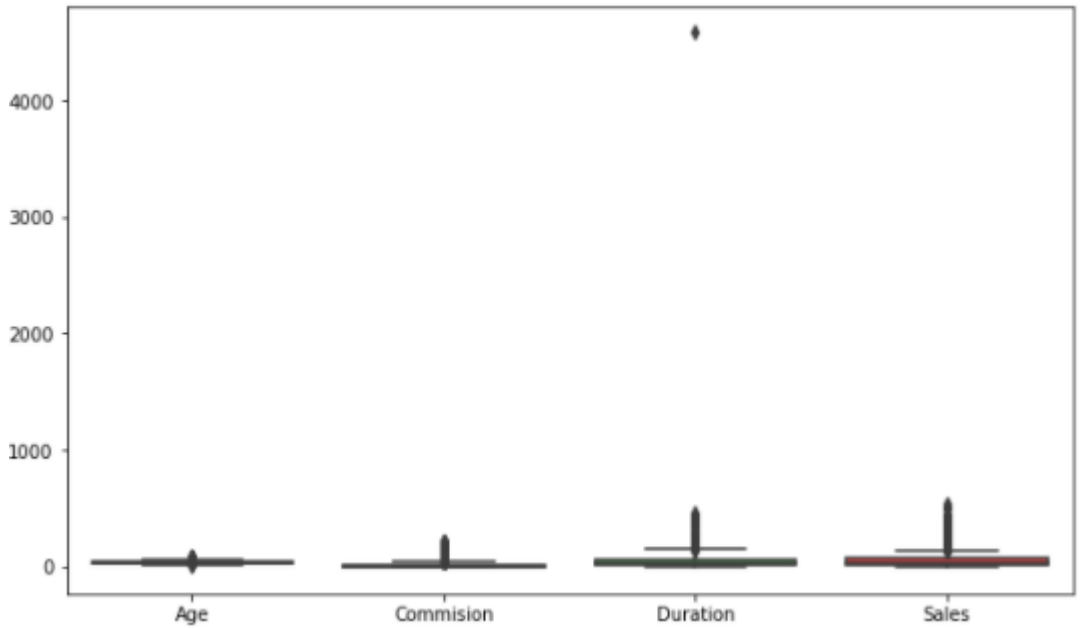
- The correlated columns can be viewed using heat map using the corr() on the dataframe.



- The Age column to other columns is very less correlated with each other.
- We can see better correlation between Commision and Sales column with 0.76.
- Next better correlation is seen between Duration and Sales column.



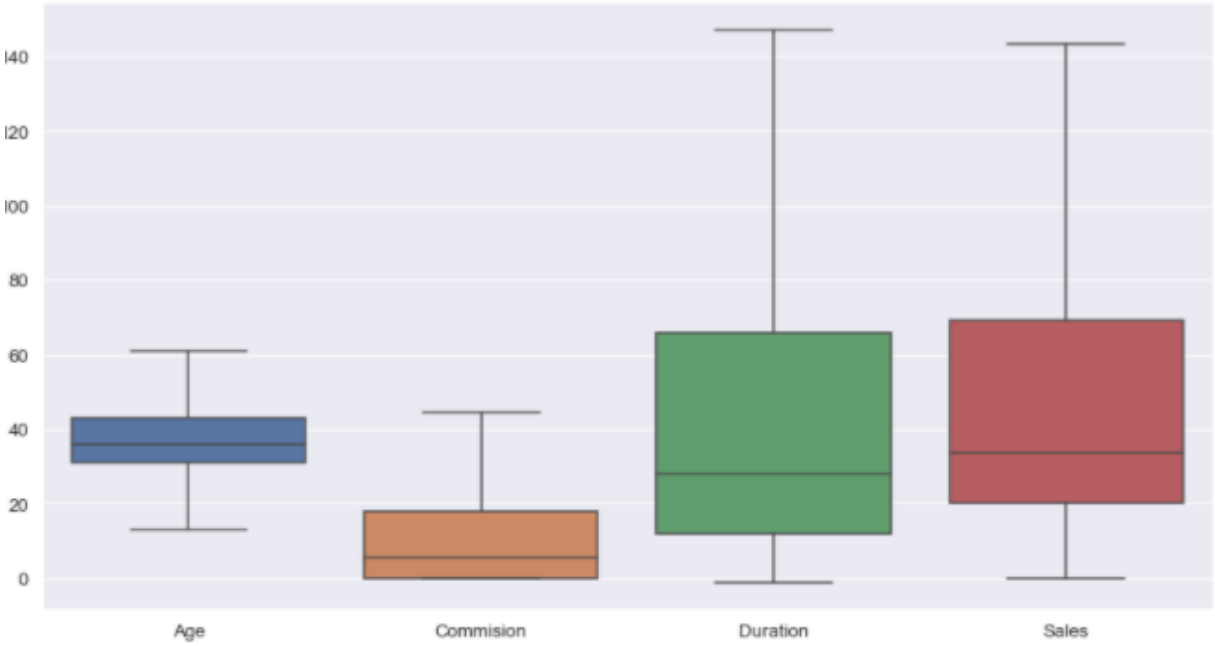
- We can visually view all the columns (of numeric type) together in the box plot.



- From the box plot, we see that the outliers are present in the data.
- The outliers are treated using Inter Quartile Range (IQR) method.

```
Shape of the data without outlier treatment : (2216, 10)
Shape of the original data  : (2861, 10)
```

- After the outlier treatment, we view the box plot.



- The Decision Tree, Random Forest, Artificial Neural Networks model can handle only numerical or categorical columns, so converting the object column type to categorical with each distinct value becoming a category or code.

```
2    1238
0     913
1     471
3     239
Name: Agency_Code, dtype: int64
1    1709
0    1152
Name: Type, dtype: int64
0    1947
1     914
Name: Claimed, dtype: int64
1    2815
0      46
Name: Channel, dtype: int64
2    1071
0     645
1     615
4     421
3     109
Name: Product Name, dtype: int64
0    2327
1     319
2     215
Name: Destination, dtype: int64
```

```
Reading info after coverting object type into categorical

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2861 entries, 0 to 2999
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Age           2861 non-null   float64
 1   Agency_Code   2861 non-null   int8
 2   Type          2861 non-null   int8
 3   Claimed       2861 non-null   int8
 4   Commision     2861 non-null   float64
 5   Channel       2861 non-null   int8
 6   Duration      2861 non-null   float64
 7   Sales         2861 non-null   float64
 8   Product Name  2861 non-null   int8
 9   Destination   2861 non-null   int8
dtypes: float64(4), int8(6)
memory usage: 208.5 KB
```

- The columns Agency_Code, Type, Claimed, Channel, Product Name, Destination are changed into int8 type from object type.
- Finally, we get the below set of records after Data Preprocessing.

	Age	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
0	48.0	0	0	0	0.70	1	7.0	2.51	2	0
1	36.0	2	1	0	0.00	1	34.0	20.00	2	0
2	39.0	1	1	0	5.94	1	3.0	9.90	2	1
3	36.0	2	1	0	0.00	1	4.0	26.00	1	0
4	33.0	3	0	0	6.30	1	53.0	18.00	0	0

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

- To train the model, we split the entire data as a set of records and their labels in either of the *Train Data or Test Data*.
- Before splitting the data, we check the proportion of the target column, i.e., Claimed in our case.

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

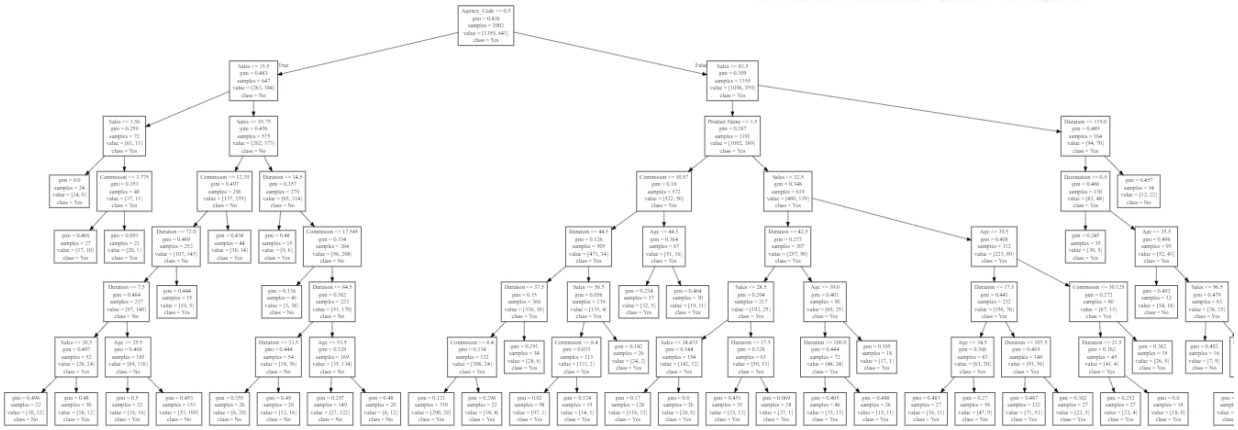
- The target column is captured into separate vector for training set and testing set.
- From sklearn.model_selection package, we import the train_test_split module to divide the data as training set and testing set.
- We can build the CART on DecisionTreeClassifier module, Ranform Forest on RandomForestClassifier, Artificial Neural Networks on MLPClassifier.
- To get the best parameters for the model, we can use the GridSearchCV from sklearn.model_selection package.
- It uses cross-validation for the number of times to loop through the predefined hyperparameters and fit our model(CART,RF,ANN) on the training set.
- This can be used to evaluate the test set and compare its performance with the previous training set.

DECISION TREE CLASSIFIER (CART):

- Using the DecisionTreeClassifier module from sklearn.tree package, we create the decision tree models.
- To avoid the tree to be overgrown and model to be overfitted, we prune them using the parameters in the model.
- The various parameters used in the model are criterion, max_depth, min_samples_leaf,min_samples_split.

```
DecisionTreeClassifier(max_depth=7, min_samples_leaf=15, min_samples_split=45)
```

- Fitting the train and test split data into the decision tree model and execute them.
- To view the tree graphically, we use export_graphviz module from sklearn.tree package.



GRIDSEARCHCV WITH DECISION TREE:

- Giving the hyperparameters with our model as decision tree and fitting the training data.

```
GridSearchCV(cv=3, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [7, 8, 9, 10, 11],
                         'min_samples_leaf': [15, 20, 22, 25],
                         'min_samples_split': [30, 35, 38, 40, 60, 70]})
```

- We get the best parameters from our gridsearchcv using best_params_ attribute.

```
{'max_depth': 8, 'min_samples_leaf': 22, 'min_samples_split': 35}
```

- Using the estimator that gave highest score which searching, to predict on both the train data and test data.

RANDOM FOREST CLASSIFIER:

- Using the RandomForestClassifier module from the sklearn.Ensemble package, we generate the Random Forest Model.
- The parameters used in the model to help prune the forest and avoid overfitting of the model.
- The various parameters are n_estimators, max_features, oob_score.
- The max_features is used to decide on considering the number of features for the best split is used for the tree splitting.
- The oob_score parameter is used to decide whether to use out-of-bag samples to estimate the generalization accuracy, pass it as true for the model to predict Out Of Bag (OOB) score.

```
RandomForestClassifier(max_features=7, n_estimators=201, oob_score=True,
                       random_state=1)
```

- To know the Score of the training dataset obtained using an out-of-bag estimate.

```
The Out of Bag score :  0.7312687312687313
```

- The RF model object can be used on train and test data to predict its performance.

GRIDSEARCHCV WITH RANDOM FOREST:

- To get the best parameters for the model, use hyper parameters in the GridSearchCV module from sklearn. model_selection package.

- The various parameters are max_depth, max_features, min_samples_leaf, min_samples_split, n_estimators that are given in param_grid.
- The random forest model is fitted to the GridSearchCV with the param_grid and cross validation.

```
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [8, 9, 10, 11],
                         'max_features': [4, 5, 6, 7],
                         'min_samples_leaf': [15, 18, 20, 25],
                         'min_samples_split': [45, 50, 60, 75],
                         'n_estimators': [101, 301]})
```

- To know the best selected parameters by the model, using best_params_ attribute.

```
{'max_depth': 9,
 'max_features': 6,
 'min_samples_leaf': 15,
 'min_samples_split': 45,
 'n_estimators': 101}
```

- Assigning the best estimator attribute to the model and using them for predicting the train and test data.

ARTIFICIAL NEURAL NETWORKS:

- The model is created using MLPClassifier module from the sklearn.neural_network package.
- The Artificial Neural Network model needs the data to be scaled before applying data on the model, because of the non-linearity in the activation function and numerical rounding errors.
- Also scaling can accelerate learning and improve performance of the model.
- Here, we use StandardScaler module from the sklearn.preprocessing package, which uses Z-Scaling mechanism.
- The training data is fitted and transformed, while the test data is fitted alone to the model to use the same Mean and Standard Deviation for both the data.
- The various parameters used are Number of Hidden Layers, Number of Neurons in hidden layers, Maximum Iteration, Type of Solver, Type of Activation, and Tolerance Rate.

```
Iteration 1, loss = 0.64719552
Iteration 2, loss = 0.63226539
Iteration 3, loss = 0.61608144
Iteration 4, loss = 0.60186285
Iteration 5, loss = 0.58953378
Iteration 6, loss = 0.57912906
Iteration 7, loss = 0.57203613
Iteration 8, loss = 0.56538675
Iteration 9, loss = 0.55797697
Iteration 10, loss = 0.55129080
Iteration 11, loss = 0.54588809
Iteration 12, loss = 0.54149327
Iteration 13, loss = 0.53775456
Iteration 14, loss = 0.53507044
Iteration 15, loss = 0.53281065
Iteration 16, loss = 0.53130139
Iteration 17, loss = 0.52964472
Training loss did not improve more than tol=0.010000 for 10 consecutive epochs. Stopping.

MLPClassifier(hidden_layer_sizes=100, max_iter=5000, random_state=21,
              solver='sgd', tol=0.01, verbose=True)
```

- Using the obtained the model we can predict on the train data and test data.

GRIDSEARCHCV FOR THE NEURAL NETWORKS:

- Various set of parameters called hyper parameters are given to the GridSearchCV, and the best set of parameters are returned by the model.
- Here, the MLPClassifier model is fitted to the GridSearchCV with the hyper parameters and tuning them.

```
GridSearchCV(cv=5, estimator=MLPClassifier(random_state=1),
             param_grid={'hidden_layer_sizes': [100, 200, 300, 500],
                         'max_iter': [2500, 3000, 5000, 6000],
                         'solver': ['sgd', 'adam'], 'tol': [0.01]})
```

- The best parameters are selected as below by the model.

```
{'hidden_layer_sizes': 500, 'max_iter': 2500, 'solver': 'adam', 'tol': 0.01}
```

- This is used to predict the train data and test data.

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

PREDICTION USING ACCURACY:

- The Accuracy is calculated using the accuracy_score function from sklearn.metrics package.
- Here, we predict and find accuracy usin the best parameters from GridSearchCV for the models.

ACCURACY OF CART ON TRAIN DATA:

```
Accuracy of CART for Train Data: 0.79
```

ACCURACY OF CART ON TEST DATA:

```
Accuracy of CART for Test Data: 0.75
```

ACCURACY OF RF ON TRAIN DATA:

```
Accuracy of RF for Train Data: 0.8
```

ACCURACY OF RF ON TEST DATA:

```
Accuracy of RF for Test Data: 0.79
```

ACCURACY OF ANN ON TRAIN DATA:

```
Accuracy of ANN for Train Data: 0.76
```

ACCURACY OF ANN ON TEST DATA:

```
Accuracy of ANN for Test Data: 0.77
```

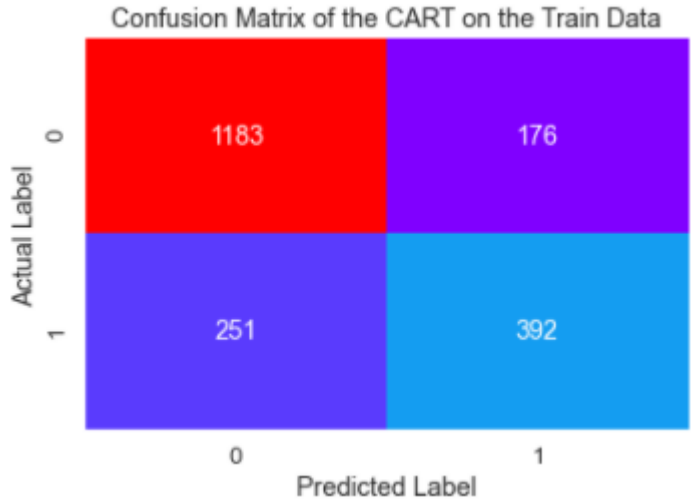
INFERENCE ON ACCURACY ACROSS CART,RF,ANN:

- Across the CART, RF, ANN Models, the Accuracy score is high for the Random Forest Model with 0.80 for the Train data and 0.79 for the Test data.

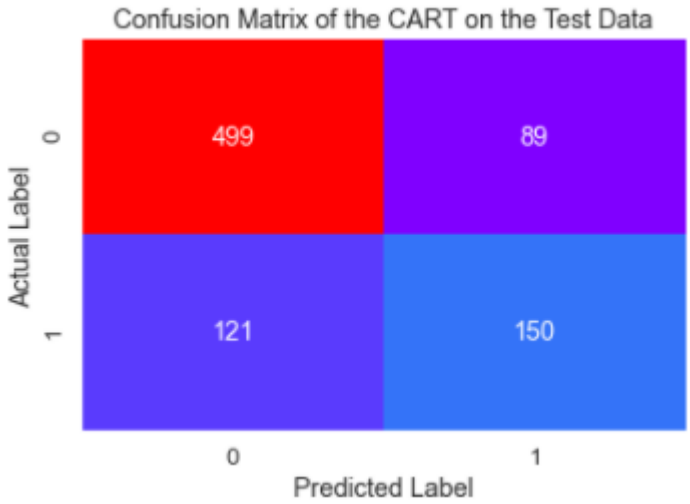
CONFUSION MATRIX:

- The Confusion matrix is created using the confusion_matrix module from sklearn.metrics package.

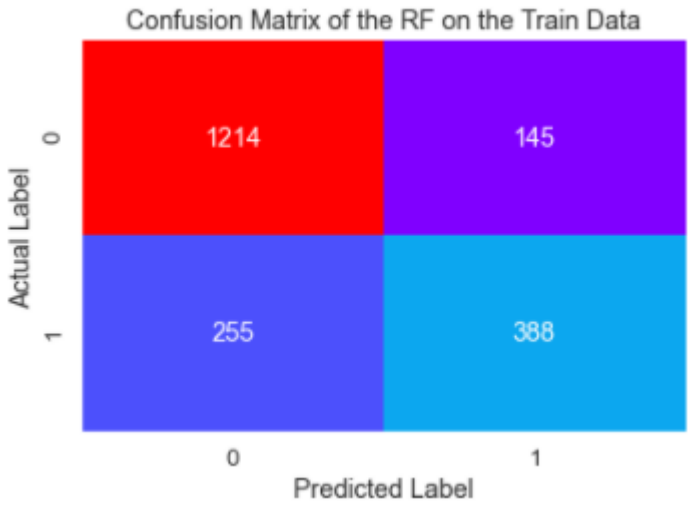
CONFUSION MATRIX OF CART ON TRAIN DATA:



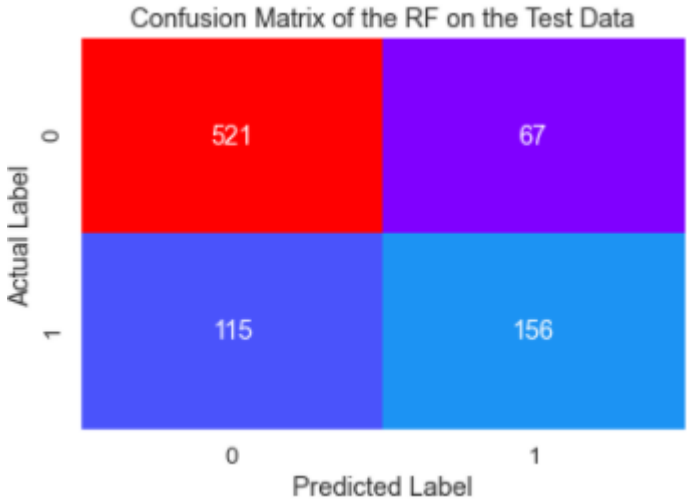
CONFUSION MATRIX OF CART ON TEST DATA:



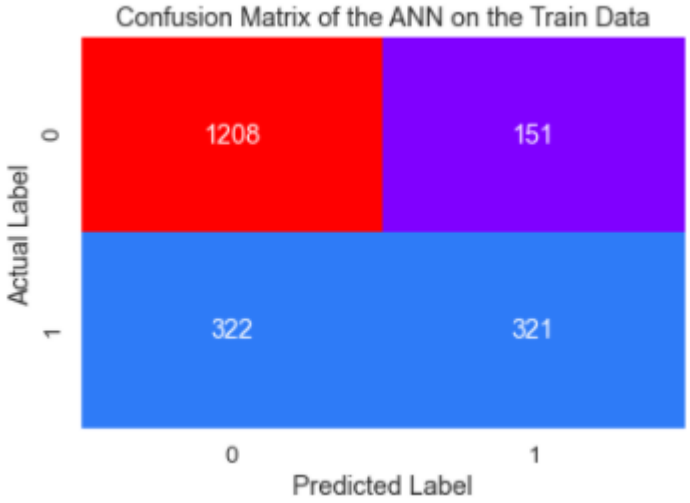
CONFUSION MATRIX OF RF ON TRAIN DATA:



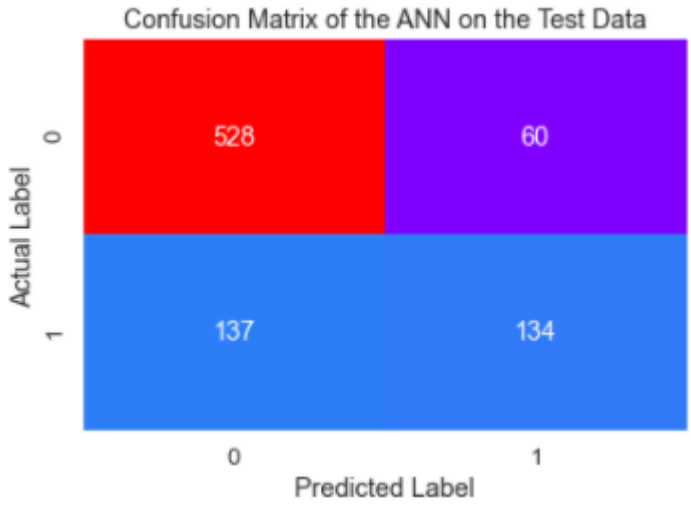
CONFUSION MATRIX OF RF ON TEST DATA:



CONFUSION MATRIX OF ANN ON TRAIN DATA:



CONFUSION MATRIX OF ANN ON TEST DATA:



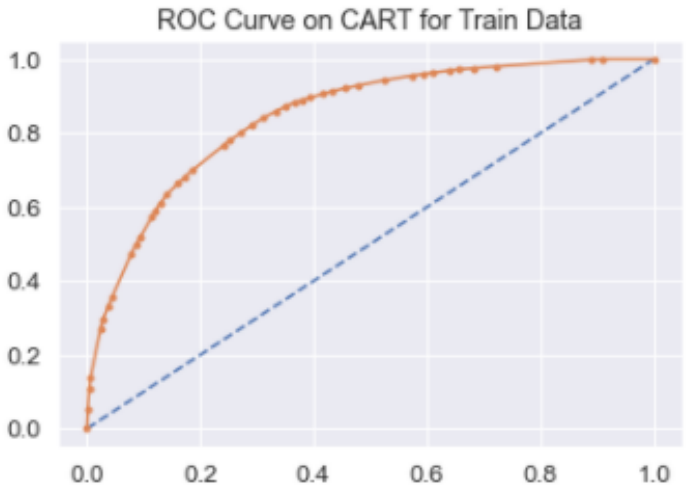
INFERENCE FROM CONFUSION MATRIX ON CART, RF, ANN MODELS:

- Comparing the Confusion matrix on the Train data across CART, RF and ANN models, the Random Forest has lower False Positive and False Nagative.
- For the Confusion matrix on the Test Data, the Random Forest model has lower False Negative and moderate False Positive to ANN Test data.

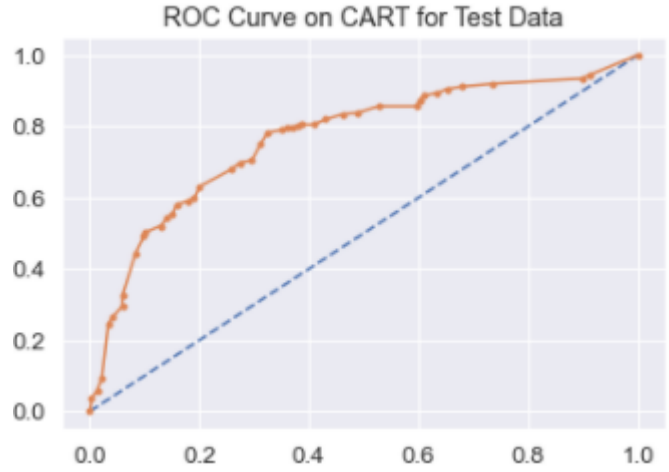
ROC CURVE:

- The ROC Curve is plotted using roc_curve module from sklearn.metrics package.

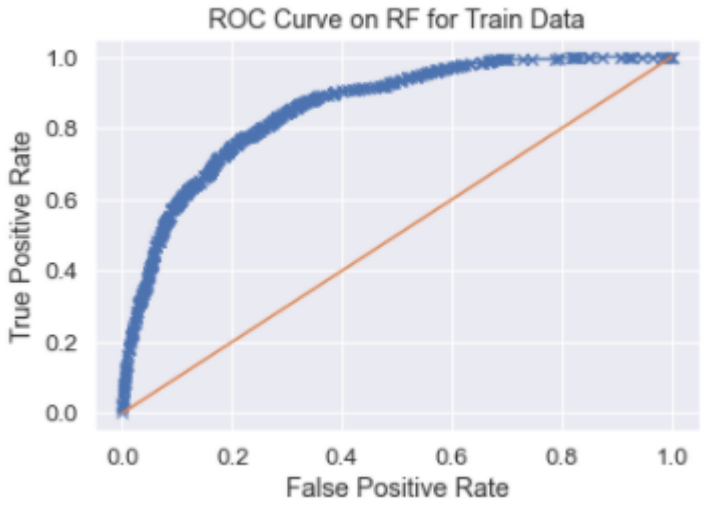
ROC CURVE OF CART ON TRAIN DATA:



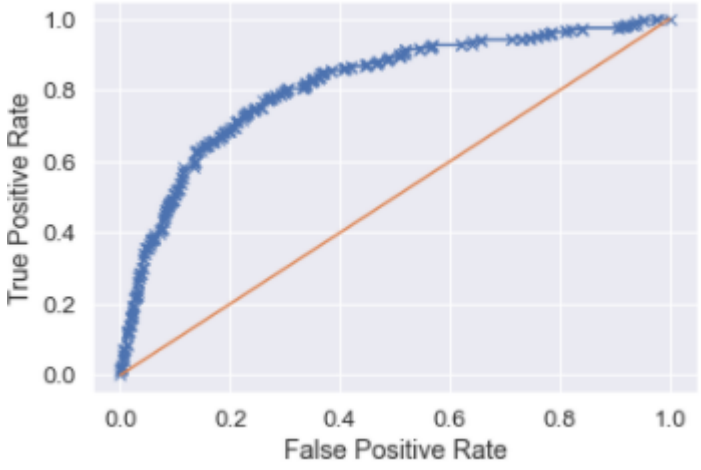
ROC CURVE OF CART ON TEST DATA:



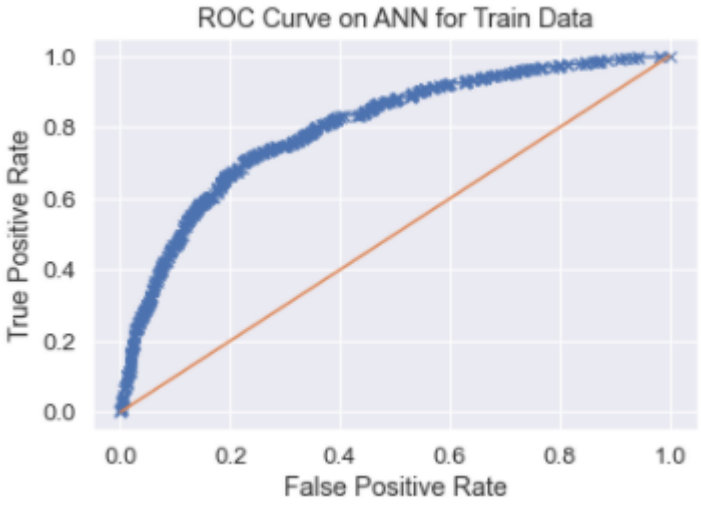
ROC CURVE OF RF ON TEST DATA:



ROC CURVE OF RF ON TEST DATA:



ROC CURVE OF ANN ON TRAIN DATA:



ROC CURVE OF ANN ON TEST DATA:

ROC-AUC SCORE:

- The ROC-AUC score is calculated using roc_auc_score module from sklearn.metrics package.

ROC-AUC SCORE OF CART ON TRAIN DATA:

```
AUC score on CART for Train Data: 0.845
```

ROC-AUC SCORE OF CART ON TEST DATA:

```
AUC score on CART for Test Data: 0.768
```

ROC-AUC SCORE OF RF ON TRAIN DATA:

```
AUC score on RF for Train Data: 0.85767254075989
```

ROC-AUC SCORE OF RF ON TEST DATA:

```
AUC score on RF for Test Data: 0.8157052488892235
```

ROC-AUC SCORE OF ANN ON TRAIN DATA:

```
AUC score on ANN for Train Data: 0.8004587812143454
```

ROC-AUC SCORE OF ANN ON TEST DATA:

```
AUC score on ANN for Test Data: 0.8076850666465847
```

INFERENCE ON ROC CURVE AND ROC-AUC SCORE ACROSS THE MODELS:

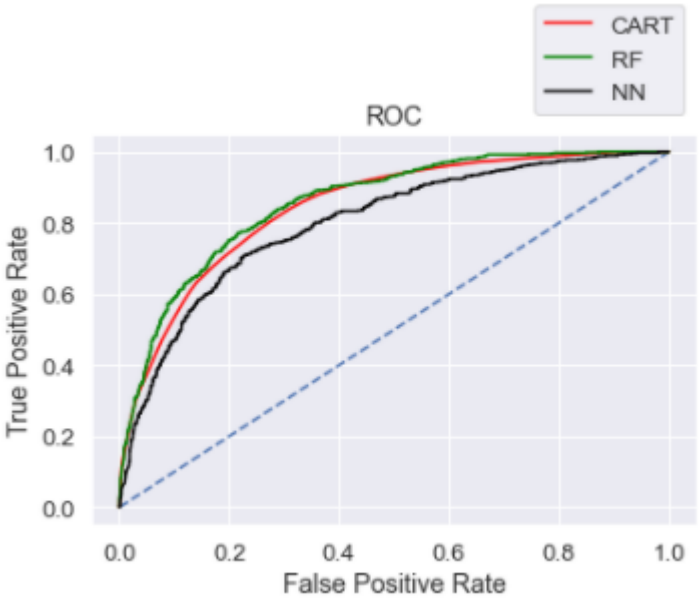
- Comparing the AUC score across CART, RF and ANN models, the Random Forest model has high score for Train data with 0.86 score and Test data with 0.82 score.

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

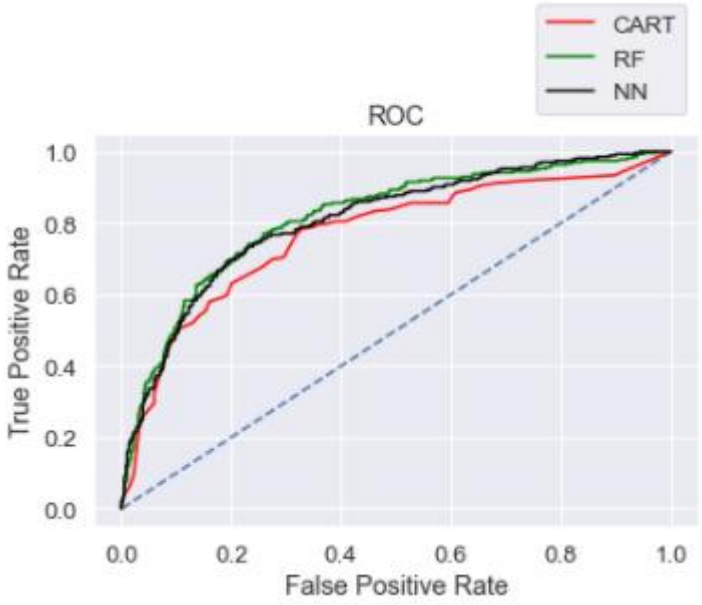
- By combining all the performance metrics from CART, RF and ANN models into a dataframe, we get the below table.

	CART Train	CART Test	RF Train	RF Test	ANN Train	ANN Test
Accuracy	0.79	0.75	0.80	0.79	0.76	0.77
AUC Score	0.85	0.77	0.86	0.82	0.80	0.81
Recall	0.61	0.55	0.60	0.58	0.50	0.49
Precision	0.69	0.62	0.73	0.70	0.68	0.69
F1 Score	0.65	0.59	0.66	0.63	0.58	0.58

- From the above table, we can say that none of the models have been Overfitted and gives good values for their Test data.
- All the models give good results for the given problem, but we select the one that is best run or well optimized for the Business Problem.
- We could see that the Random Forest model has high values of Accuracy on the Train data (0.80) and Test data (0.79) and lower Accuracy on the Artificial Neural Networks for Train data (0.76) and CART for Test data (0.75).
- Much higher AUC score with Random Forest model for both the Train data (0.86) and Test data (0.82) and lower AUC score on ANN Train data (0.80) and CART test data (0.77).
- Recall is considerably good with Random Forest model.
- Precision is much higher with Random Forest model.
- The F1 Score is high with Random Forest model.
- Checking the ROC Curve for all the 3 models for their respective Train data.



- The Random Forest well broader curve comparatively.
- Checking ROC curve for all the 3 models for their respective Test data.



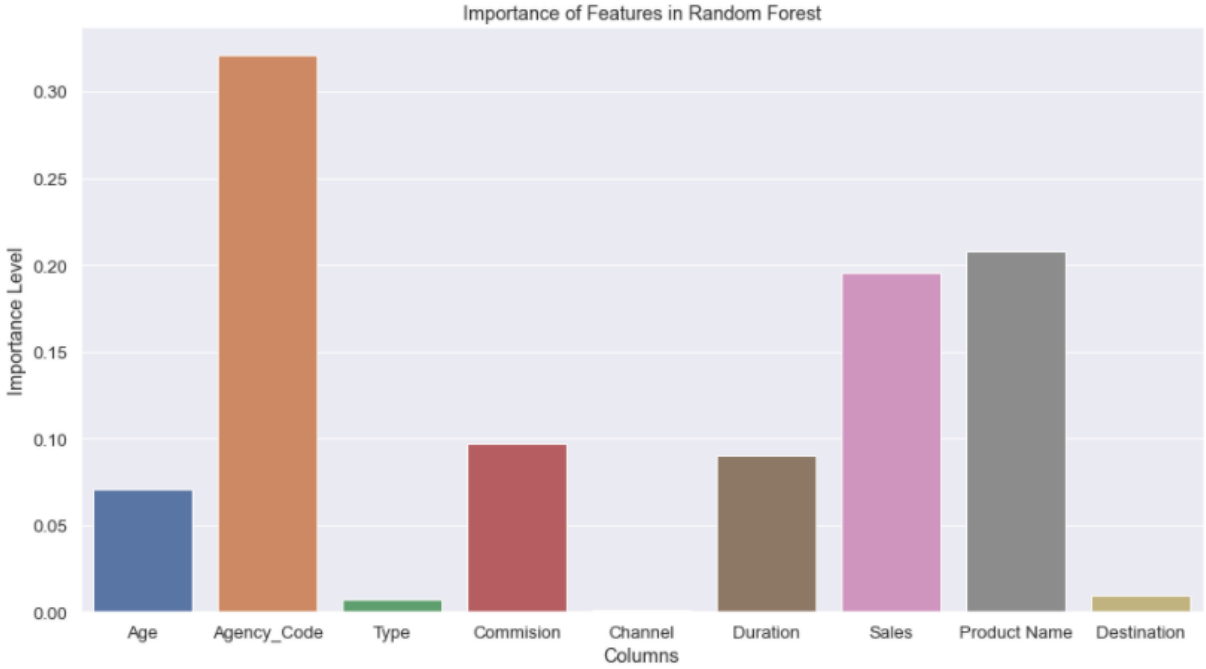
- The Artificial Neural Networks and Random Forest models have nearly same curve.
- We can conclude that the Random Forest model is the best optimized model for the given Business Problem.

2.5. Inference: Basis on these predictions, what are the business insights and recommendations.

- ❖ The Random Forest is the preferred model for this Business Problem.
- ❖ We can understand the pattern of split of the tress in the Random forest to understand the Claim.
- ❖ From the columns chosen for the split helps to understand which feature is to be given importance.

	Imp
Agency_Code	0.320964
Product Name	0.207977
Sales	0.195424
Commision	0.097431
Duration	0.090315
Age	0.070402
Destination	0.009861
Type	0.006861
Channel	0.000766

- Here we can see that the column Agency_Code is given as high priority for the choosing the firm with 32%.
- The Product Name is selected next as it has various tour insurances of about 20%.
- The people select the Sales and the amount for the tour insurance policy with 19%.
- The least preferred attributes are Channel and Type with significant value.



- The claim is made mostly based on Agency code.