Business Report
Data Mining

Arnab Ghosal

23 May 2021

Problem 1 : Clustering

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

Questions:

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

While reading the data and understanding to do exploratory data analysis we followed below steps likewise :

Importing necessary libraries :

```
# Importing Libraries

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import zscore
from scipy.cluster.hierarchy import fcluster
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
```

Read the dataset :

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

Information of the data (data-types & row-column) :

```
<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
```

Descriptive summary of the data :

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

Shape of the data (no. of row & columns available) :

We got to know there are 210 rows & 7 columns available

Checking for Null values existence :

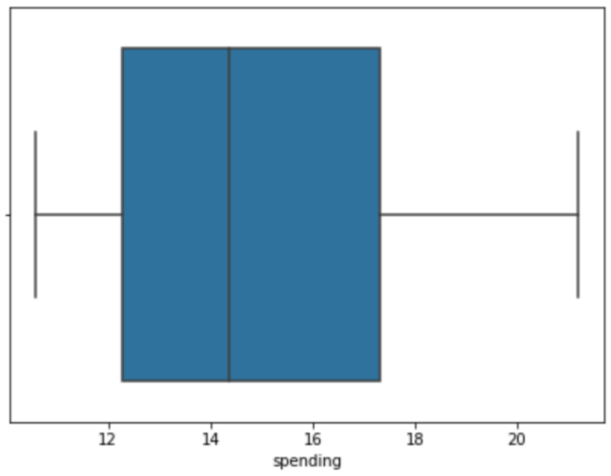
We checked the null values by a method called **isnull()** and it returned **false** , that indicates this dataset does not have any null values .

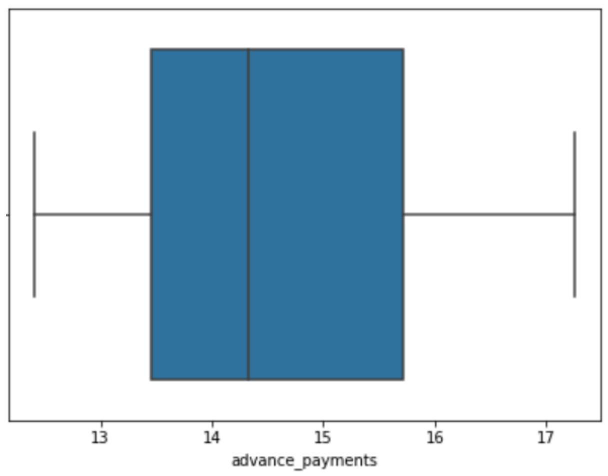
Checking for duplicate values existence :

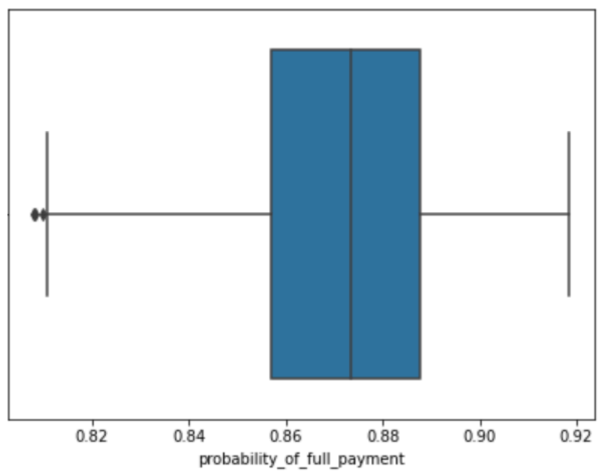
We checked the duplicate values by a method called **duplicated()** and it returned **False** , that indicates this dataset does not have any duplicate values .

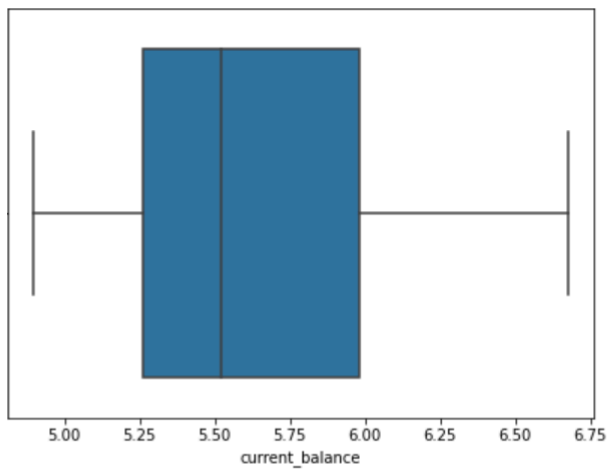
Checking for outliers & treating them :

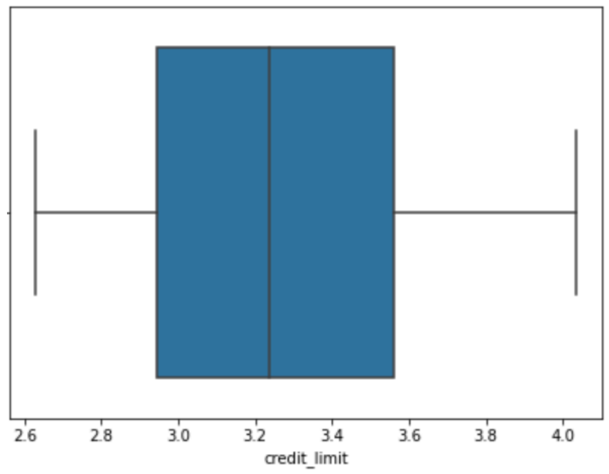
We wanted to check the outliers by putting all the columns into box plots and they looked like :

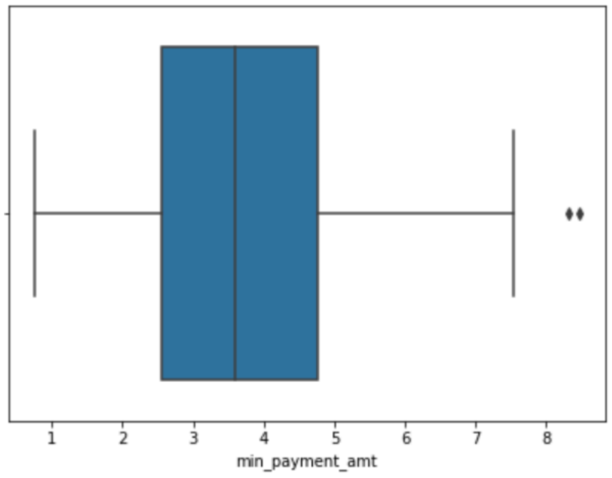


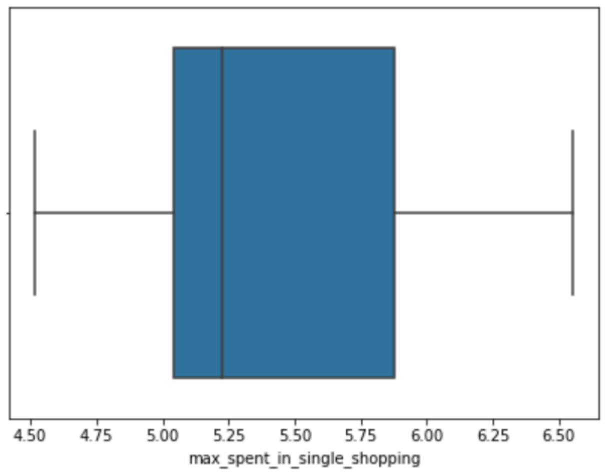

13
14
15
16
17
advance_payments


0.82
0.84
0.86
0.88
0.90
0.92
probability_of_full_payment


5.00
5.25
5.50
5.75
6.00
6.25
6.50
6.75
current_balance


2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
credit_limit


1
2
3
4
5
6
7
8
min_payment_amt


4.50
4.75
5.00
5.25
5.50
5.75
6.00
6.25
6.50
max_spent_in_single_shopping

From the above box-plots we could figure out that there are two columns which have minimal outliers , those are :

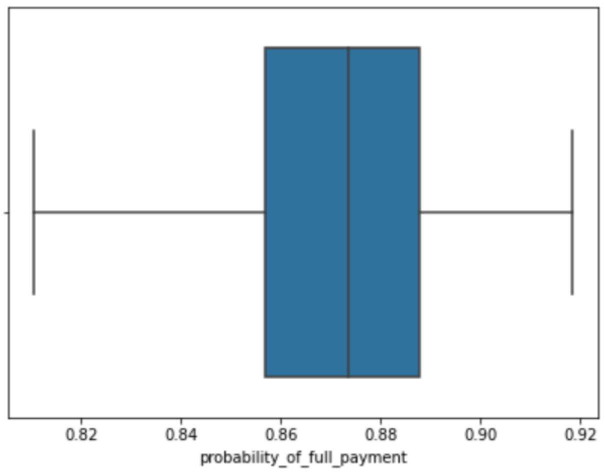
probability_of_full_payment

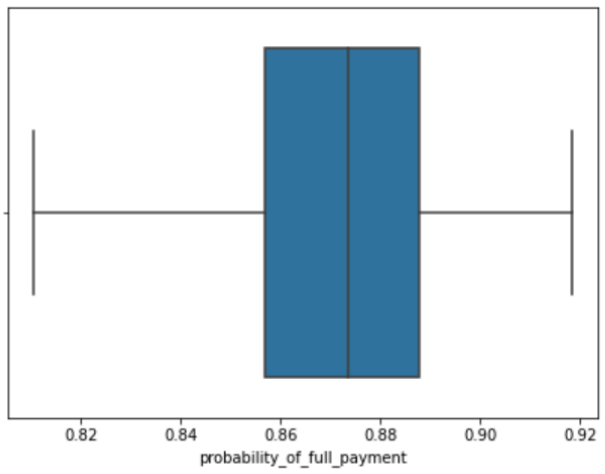
min_payment_amt

We know that while doing clustering implementation outliers plays a major role , hence we decided to treat them by

lower_range = Q1 -(1.5 * IQR) & upper_range = Q3 + (1.5 * IQR)

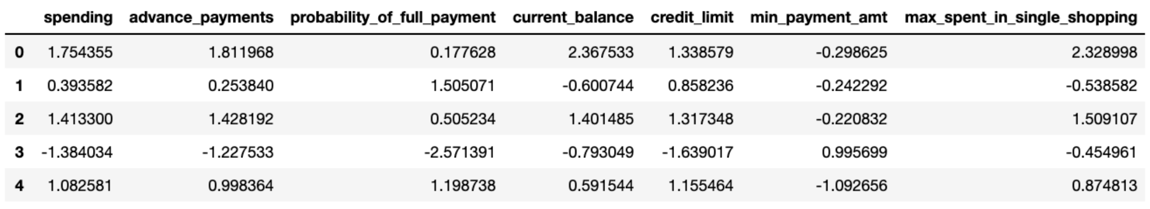
After we replaced the values with lower_range and upper_range , the above mentioned columns (**probability_of_full_payment & min_payment_amt**) were outliers free and the same also displayed in the box-plot as well .





Before doing Univariate , Bi-variate & Multivariate analysis we decided to scale the data as the magnitude of the column's values are not same . Hence we applied **z-score** mechanism for the same .

After scaling them the data looks like :



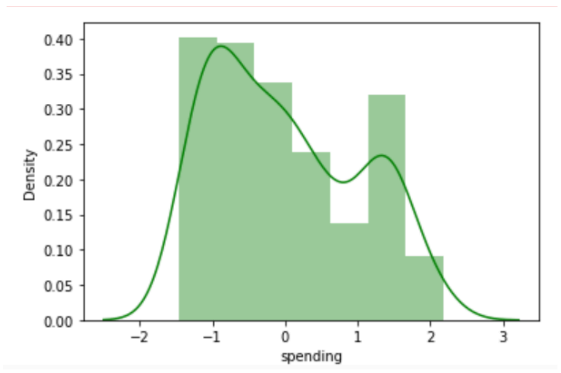
	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
0	1.754355	1.811968	0.177628	2.367533	1.338579	-0.298625	2.328998
1	0.393582	0.253840	1.505071	-0.600744	0.858236	-0.242292	-0.538582
2	1.413300	1.428192	0.505234	1.401485	1.317348	-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

<u>Univariate Analysis :</u>

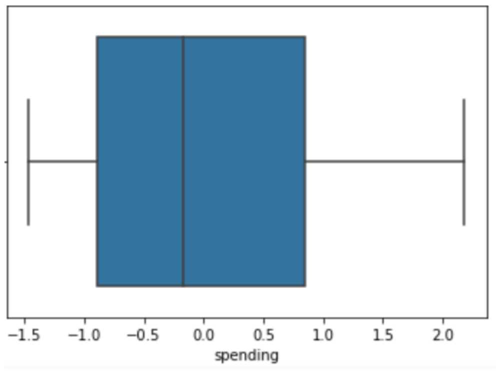
Univariate analysis is the simplest form of analysing data. "Uni" means "one", so in other words our data takes only one variable at a time. It doesn't deal with causes or relationships (unlike regression) and it's major purpose is to describe; It takes data (individual column) separately , summarises that data and finds patterns in the data.

(Please refer notebook for better clarity)

```
Description of spending
--------------------------------------------------------------
--------------
count    2.100000e+02
mean     9.148766e-16
std      1.002389e+00
min     -1.466714e+00
25%     -8.879552e-01
50%     -1.696741e-01
75%      8.465989e-01
max      2.181534e+00
Name: spending, dtype: float64
--------------------------------------------------------------
--------------
Distribution of spending
--------------------------------------------------------------
--------------
```



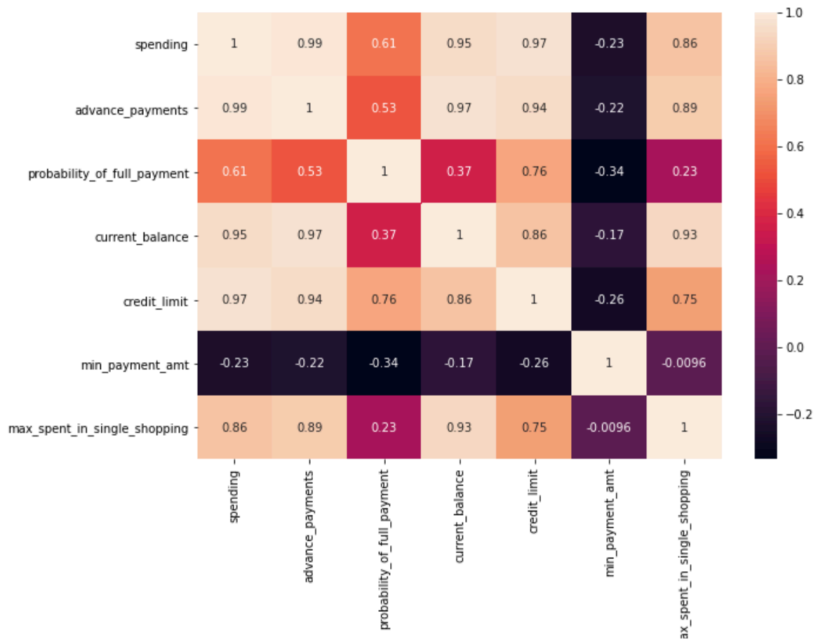
Box-plot of spending



The output displays, total 7*3 = 21 distinct charts/columns & descriptions. Hence I have put the screenshot of only one variable which is **Spending** . **(Please refer the python notebook for better clarity)**

Bi-variate Analysis

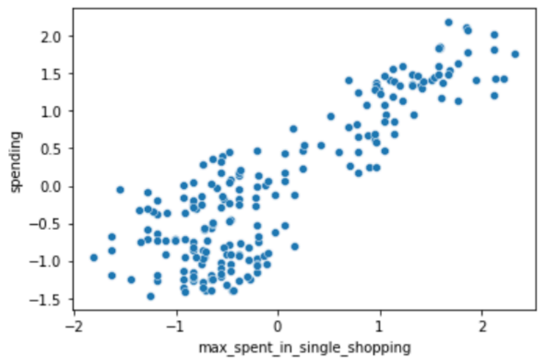
As the name suggest it gives analysis of two specific variable at a time and we can derive correlation b/w them. We can use Heat-map for the same and further to highlight each correlated pair we can use scatter-plot .

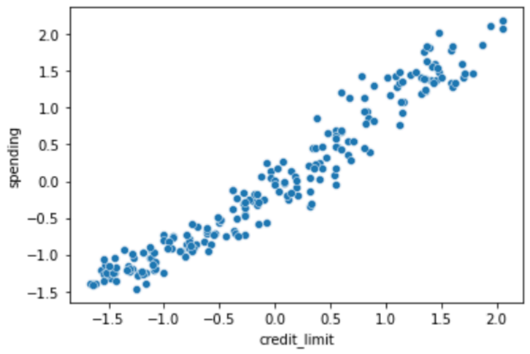


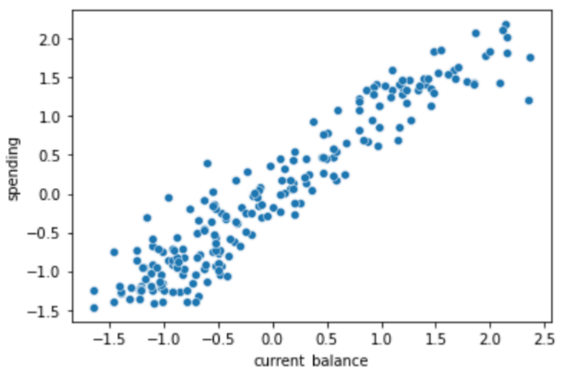
By looking at the heat-map we can see that there are few pair of variable which are highly correlated .

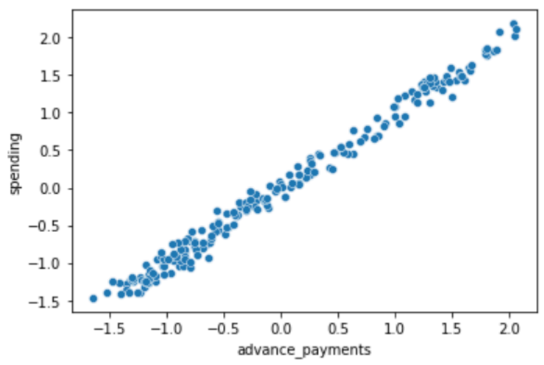
1. max_spent_in_single_shopping & spending
2. credit_limit & spending
3. current_balance & spending
4. advance_payments & spending
5. credit_limit & advance_payments
6. current_balance & advance_payments
7. credit_limit & probability_of_full_payment
8. credit_limit & current_balance

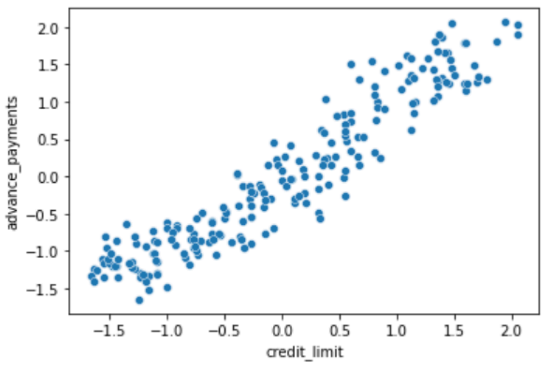
Scatter-plots to highlight each correlated pairs are :

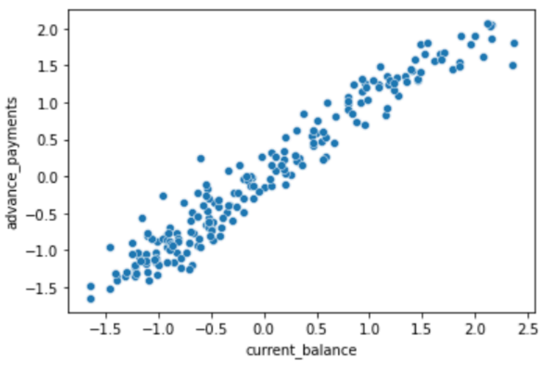


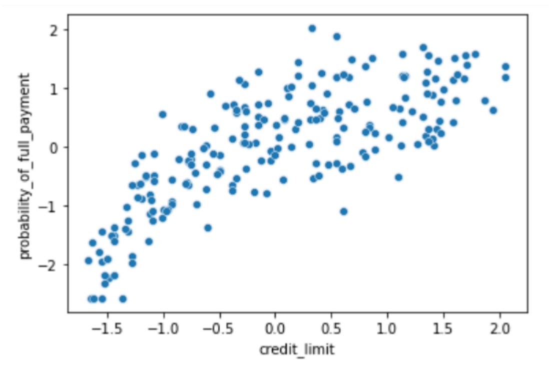


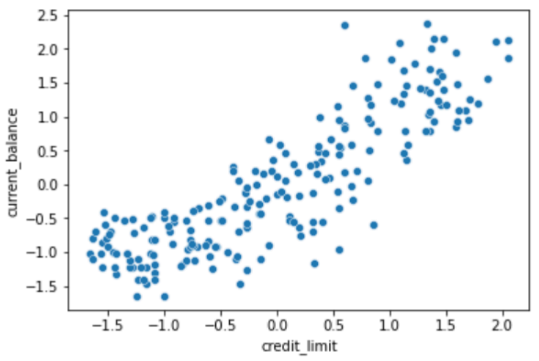








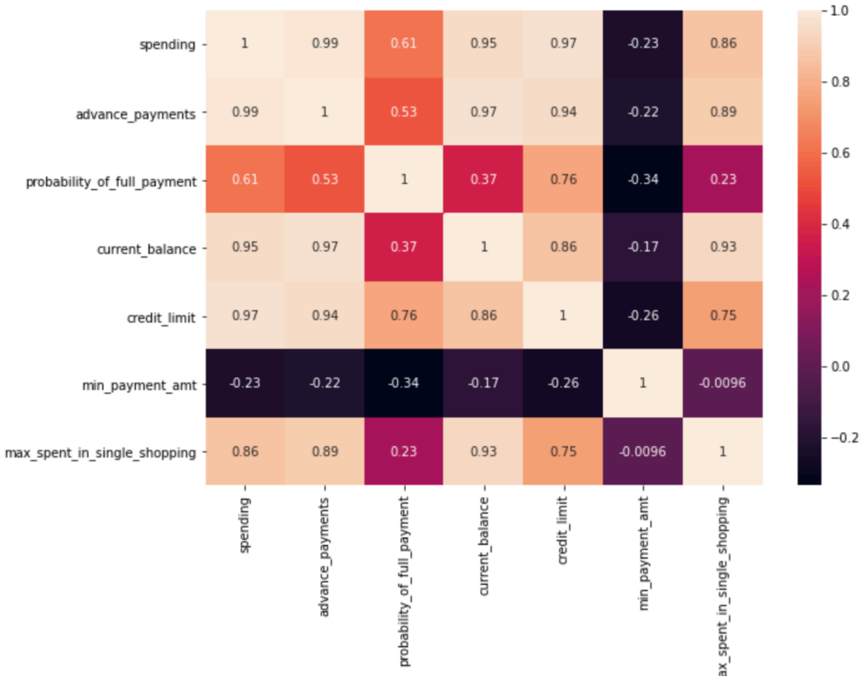


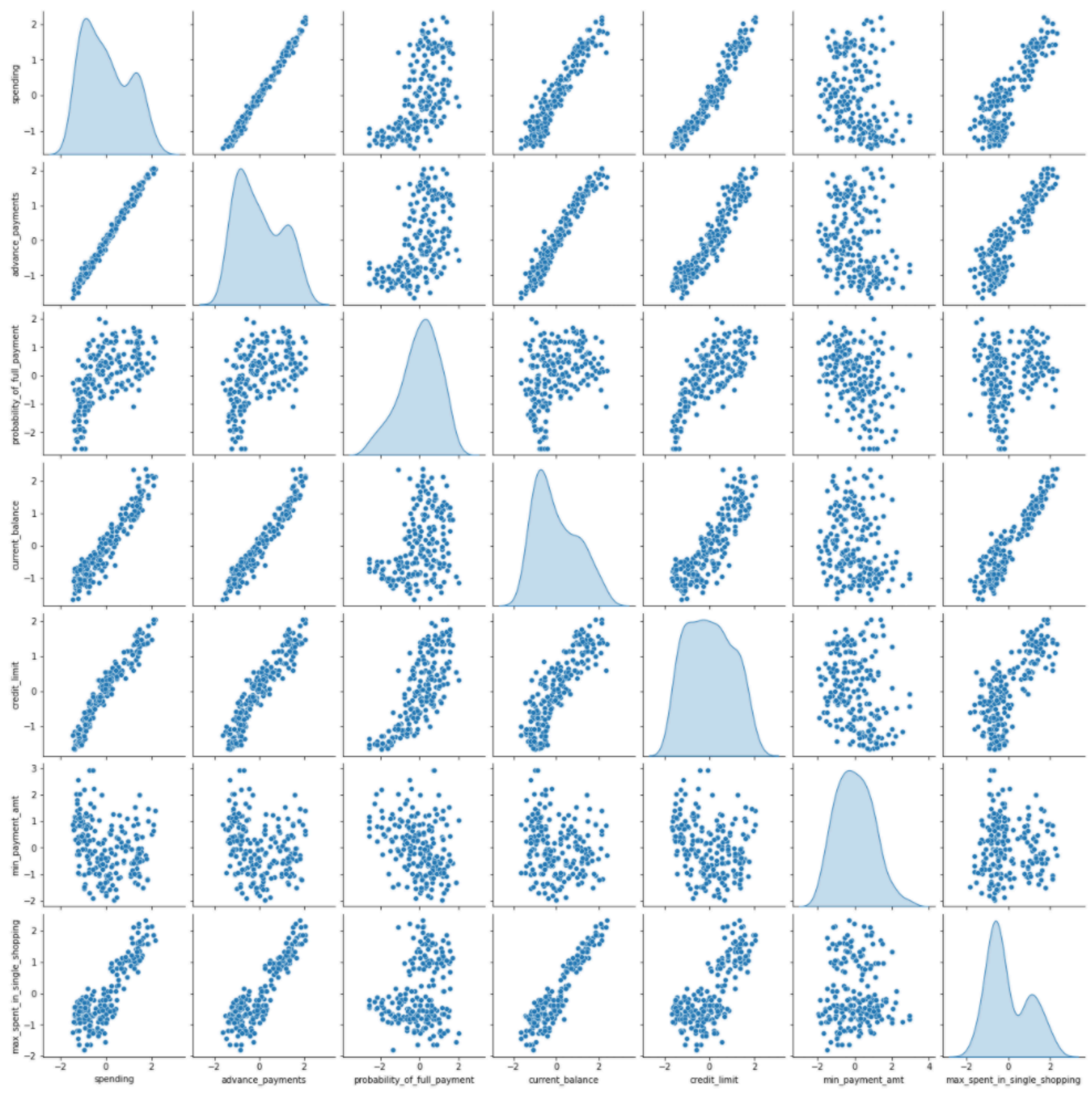


Multivariate Analysis

Multivariate means involving multiple dependent variables resulting in one outcome. For example, we cannot predict the weather of any year based on the season. There are multiple factors like pollution, humidity, precipitation, etc.

To derive **Multivariate Analysis** we have implemented **pair plot & Heat map** .


spending
advance_payments
probability_of_full_payment
current_balance
credit_limit
min_payment_amt
max_spent_in_single_shopping
1 0.99 0.61 0.95 0.97 -0.23 0.86
0.99 1 0.53 0.97 0.94 -0.22 0.89
0.61 0.53 1 0.37 0.76 -0.34 0.23
0.95 0.97 0.37 1 0.86 -0.17 0.93
0.97 0.94 0.76 0.86 1 -0.26 0.75
-0.23 -0.22 -0.34 -0.17 -0.26 1 -0.0096
0.86 0.89 0.23 0.93 0.75 -0.0096 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


spending
advance_payments
probability_of_full_payment
current_balance
credit_limit
min_payment_amt
max_spent_in_single_shopping

1.2) Do 1you think scaling is necessary for clustering in this case? Justify

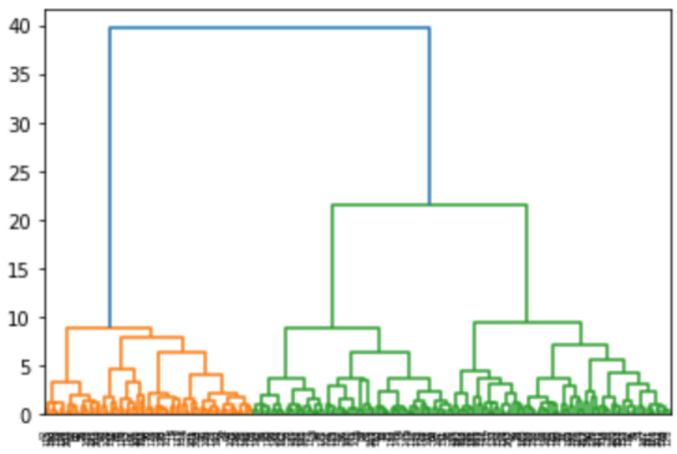
Basically if all the columns in a table or dataset are not in a same magnitude then its really difficult to compare them and come to a conclusion , so by scaling them we bring their magnitude into a certain range where we can compare them and come to conclusion , derive business implications.
Here we can see the same , hence we need to do scaling before we derive any business implications .
The most common techniques of feature scaling are Normalization and Standardization.
Normalization is used when we want to bound our values between two numbers, typically, between [0,1] or [-1,1]. While Standardization transforms the data to have zero mean and a variance of 1 .

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

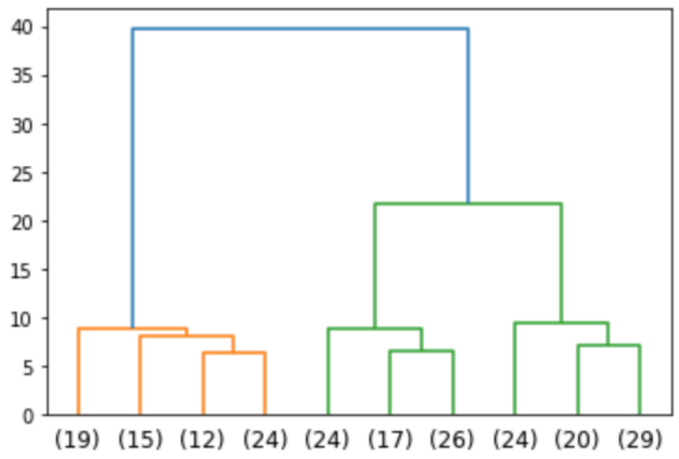
Clustering comes under unsupervised learning. In Hierarchical clustering datas are merged to create clusters, based on the distances between records and clusters. Here we have imported dendrogram followed by linkage to identify and depict the optimal number of clusters.

Once we performed hierarchical clustering on the scaled data , Ward linkage method applied to get the result. **(Please refer notebook for better clarity)**

There are two different clusters , which are in colour "Orange and Green" are achieved through this dendrogram. We have found that maximum number of records are falling under the "Green" cluster.



As we can see the above dendrogram is bit clumsy and difficult to figure out the grouping , hence we have truncated the dendrograms by using **truncate_mode** function with the value P =10 to get a better output of the same.



From the above dendrogram , we can see that if I draw a horizontal line above 15 and below 20 then there are 3 vertical lines falling within that range . By using **maxclust** we can verify that those 3 clusters are good enough or not .

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

From the above analysis we can see that 3 clusters are good enough here.

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

As in K-Means clustering we need pre-determine the number of clusters , here I take a random value of K =3 .Here we need to get the optimal WSS plot to derive optimal number of clusters.

we applied K-Mean algorithm on the scaled data by using **fit()** function and derive the labels .

```
array([1, 2, 1, 0, 1, 0, 0, 2, 1, 0, 1, 2, 0, 1, 2, 0, 2, 0, 0, 0, 0, 0,
       1, 0, 2, 1, 2, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1,
       0, 0, 2, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 2, 0, 0, 2, 2, 1,
       1, 2, 1, 0, 2, 0, 1, 1, 0, 1, 2, 0, 1, 2, 2, 2, 2, 1, 0, 2, 1, 2,
       1, 0, 2, 1, 2, 0, 0, 1, 1, 1, 0, 1, 2, 1, 2, 1, 2, 1, 1, 0, 0, 1,
       2, 2, 1, 0, 0, 1, 2, 2, 0, 1, 2, 0, 0, 0, 2, 2, 1, 0, 2, 2, 0, 2,
       2, 1, 0, 1, 1, 0, 1, 2, 2, 2, 0, 0, 2, 0, 1, 0, 2, 0, 2, 0, 2, 2,
       0, 2, 2, 0, 2, 1, 1, 0, 1, 1, 1, 0, 2, 2, 2, 0, 2, 0, 2, 1, 1, 1,
       2, 0, 2, 0, 2, 2, 2, 2, 1, 1, 0, 2, 2, 0, 0, 2, 0, 1, 2, 1, 1, 0,
       1, 0, 2, 1, 2, 0, 1, 2, 1, 2, 2, 2], dtype=int32)
```

After deriving the labels we wanted to check **sum of squars** or **inertia** when K=3 . Which has come **430.298481751223.**

(Please refer notebook for better clarity)

First we need to declare a blank array : **wss []**

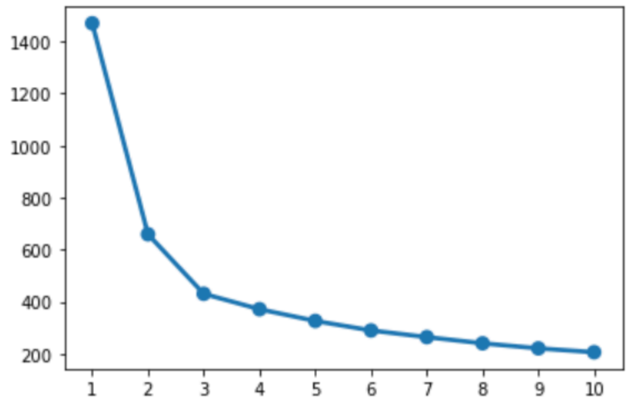
Once **inertia** values are derived by using different k-values will append the same in that array.

Here we will try with different clusters **(K value)** and find the inertia. The larger the drop better the **WSS**. If the drop is not significant enough the additional cluster is not useful or not be considered .

We tried k-values from 1-10 and below values we obtained :

```
 1469.999999999999,
 659.14740095485,
 430.298481751223,
 371.0356644664012,
 325.9741284729876,
 289.45524862464833,
 263.859944426353,
 239.94446635017925,
 220.59353946108112,
 205.7633419678701
```

From the above values we can see only better drop observed in **first 3 values** , to confirm the same we can plot **elbow-curve** and visualise the same. **(Please refer notebook for better clarity)**



From the above curve as well we can see that only till x axis value 3 , there are significant drop in inertia, rest are not useful here.

Here we are implementing **silhouette score** by which we can find out the mapping of each variable into the specific cluster is appropriate/correct or not. **(Please refer notebook for better clarity)**

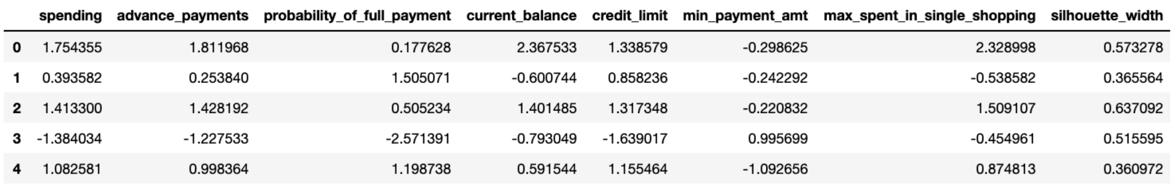
We tried with k value = 3 & K value = 4 , idea behind this was the better **silhouette score** defines the optimal number of clusters.

By using 3 as K-value we got **silhouette score = 0.4008059221522216**

By using 4 as K-value we got **silhouette score = 0.3373662527862716**

From the above analysis we can conclude **3 is the optimum cluster** , as silhouette score is giving a better answer for no. of cluster = 3 instead no. of cluster = 4.

Along with **silhouette score** through **silhouette width** also we can cross verify that none of the variables are wrongly mapped into any of clusters , and ideally they should not be negative .



	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	silhouette_width
0	1.754355	1.811968	0.177628	2.367533	1.338579	-0.298625	2.328998	0.573278
1	0.393582	0.253840	1.505071	-0.600744	0.858236	-0.242292	-0.538582	0.365564
2	1.413300	1.428192	0.505234	1.401485	1.317348	-0.220832	1.509107	0.637092
3	-1.384034	-1.227533	-2.571391	-0.793049	-1.639017	0.995699	-0.454961	0.515595
4	1.082581	0.998364	1.198738	0.591544	1.155464	-1.092656	0.874813	0.360972

Here we derived **silhouette width** and attached them with our scaled data. **(Please refer notebook for better clarity)**

Let us check the minimum value of **silhouette width** which should not be negative .

Minimum value we got here **0.002768541128616063 8**

From the above analysis we can see that minimum value of silhouette width is 0.002 , which indicates here that none of the variables are incorrectly mapped into cluster.

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

(Please refer notebook for better clarity)

Here to answer the same better we added the K-means=3 labels in the dataset which records are belong to which cluster.

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Clus_kmeans_3
0	19.94	16.92	0.875200	6.675	3.763	3.252	6.550	1
1	15.99	14.89	0.906400	5.363	3.582	3.336	5.144	2
2	18.95	16.42	0.882900	6.248	3.755	3.368	6.148	1
3	10.83	12.96	0.810588	5.278	2.641	5.182	5.185	0
4	17.99	15.86	0.899200	5.890	3.694	2.068	5.837	1

We wanted to check each cluster has how many records in it in a sorted fashion.

```
0    72
1    67
2    71
Name: Clus_kmeans_3, dtype: int64
```

Now to profile them and based on cluster , we need to display the records in such a way so that cluster grouping used here as a part of index and records are displayed under them.

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	freq
Clus_kmeans_3								
0	11.856944	13.247778	0.848330	5.231750	2.849542	4.733892	5.101722	72
1	18.495373	16.203433	0.884210	6.175687	3.697537	3.632373	6.041701	67
2	14.437887	14.337746	0.881597	5.514577	3.259225	2.707341	5.120803	71

Now from the above pic we can derive few inferences as below :

1. **Cluster 0:** This group is least on spending & advance_payments . Probability of the full payment is also very less . So they are mostly the liable to the bank . Bank needs to concentrate on this group more .

2. **Cluster 1:** Pristine customers who spent most of the money among the other groups. Along with spending , credit limit and few other parameters are also relatively high compared to other members belong to different clusters but the minimum payment amount is less compared to cluster 0 group or least spending group. Bank can look into the same and helps these customers to improve on the minimum payment.

3. **Cluster 2:** These customers a lying in b/w rest other two clusters or groups. They are medium in spending . We can observe that there is a high advance payments made by these people and Probability of full payments is also higher than rest other groups which Is a very good sign for a bank , this are most trustworthy customers . Bank can increase credit limit to these people by looking at the payment tendency of these people , which will be very much beneficial for the bank eventually.

Problem 2 : CART-RF-ANN

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

Questions:

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

While reading the data and understanding to do exploratory data analysis we followed below steps likewise :

Importing necessary libraries :

```
# Importing Libraries

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score,roc_curve,classification_report,confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
```

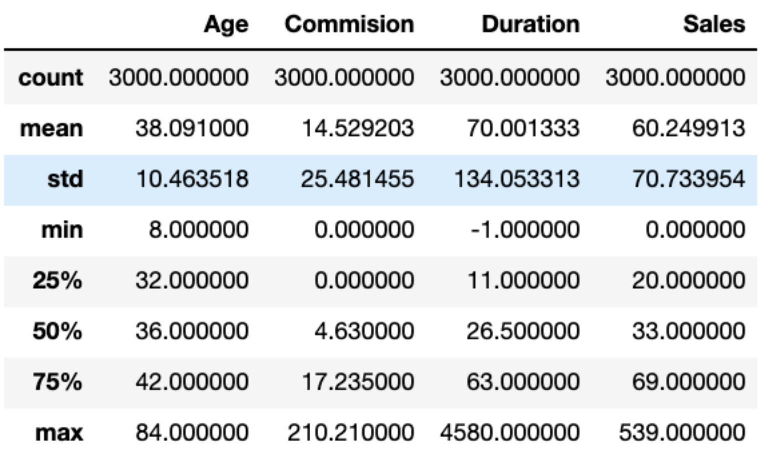
Read the data :

	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

Information of the data (data-types & row-column) :

```
<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
```

Descriptive statistics to summarize the data :



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

Shape of the data (No. of Row's & Column's)

We got to know there are 3000 rows & 10 columns available

Checking for null values (if it is there)

We checked the null values by a method called **isnull()** and it returned **false** , that indicates this dataset does not have any null values .

Checking for duplicate values existence :

We checked the duplicate values by a method called **duplicated()** and it returned **True** , that indicates this dataset has duplicate values .

Now we wanted to check how many duplicate row's are available in this dataset and we got to know there are **139** rows which are duplicate and we cleaned them.
(Please refer notebook for better clarity)

Columns in this dataset :

['Age', 'Agency_Code', 'Type', 'Claimed', 'Commision', 'Channel','Duration', 'Sales', 'Product Name', 'Destination']

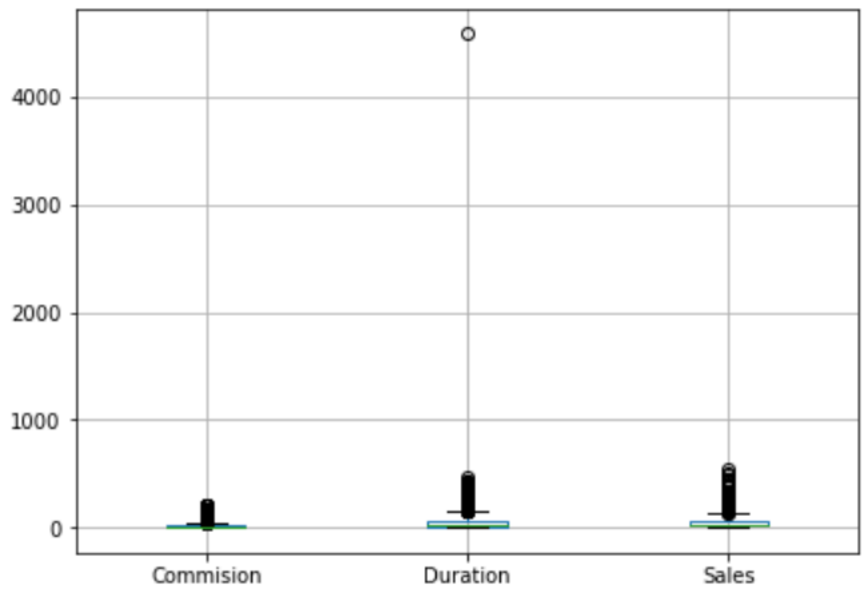
Here in this dataset from the **.info()** we got to know few columns are numeric and few are categoric , so before start working with this data we need to do label encoding .

In this dataset the problem statement is to find out the **“claimed”** status , hence that is our target or dependent variable .
Here from the problem statement we can derive that , “**Age”** column will not going to help us to derive the claim status of the customer. Hence dropping it . After dropping the same the data looks like :

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

Checking for outliers only for the numeric columns :

We can evaluate the outliers by plotting them into box-plots and the result is as follows :



Here we can see that most of the values are in these above columns are lying below and above Lower range & upper range

Where

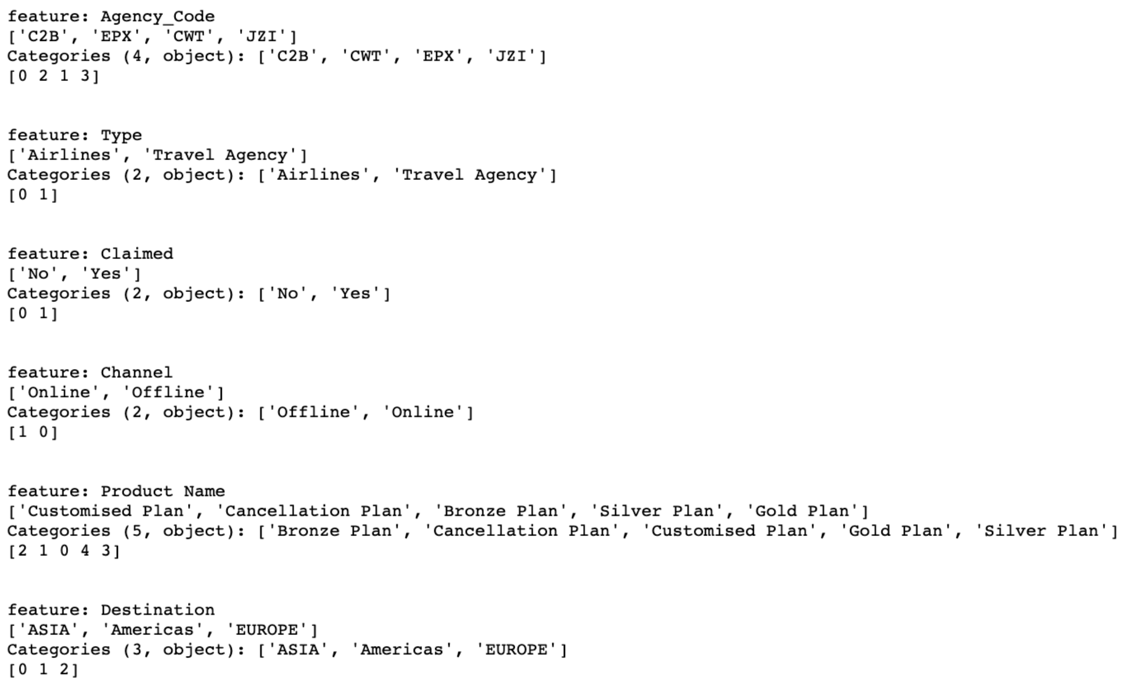
lower_range = Q1 -(1.5 * IQR)

upper_range = Q3 + (1.5 * IQR)

Hence we are not treating outliers here

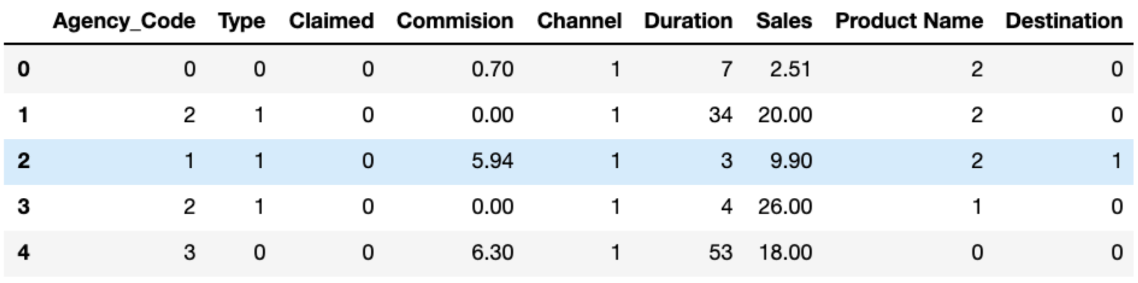
Changing all categorical variable to continuous type :

(Please refer notebook for better clarity)



```
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]
```

Checking the dataset once after the label encoding :



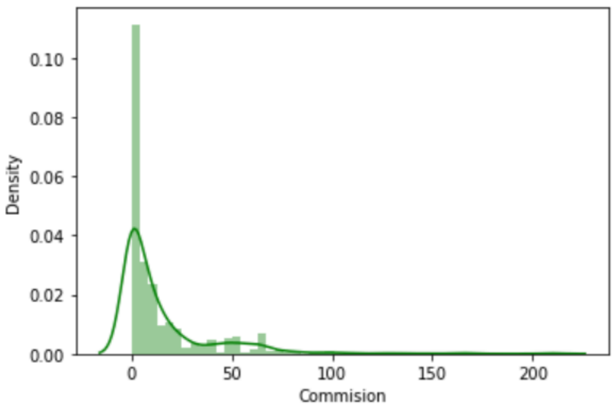
	Agency_Code	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
0	0	0	0	0.70	1	7	2.51	2	0
1	2	1	0	0.00	1	34	20.00	2	0
2	1	1	0	5.94	1	3	9.90	2	1
3	2	1	0	0.00	1	4	26.00	1	0
4	3	0	0	6.30	1	53	18.00	0	0

Univariate Analysis :

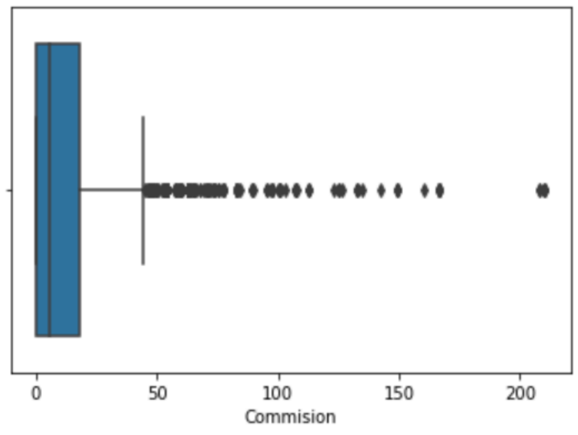
Univariate analysis is the simplest form of analysing data. “Uni” means “one”, so in other words our data takes only one variable at a time. It doesn't deal with causes or relationships (unlike regression) and it's major purpose is to describe; It takes data (individual column) separately , summarises that data and finds patterns in the data.

As we know that Univariate analysis is only applied only for the numeric columns , so here we are only dealing with numeric columns.
(Please refer notebook for better clarity)

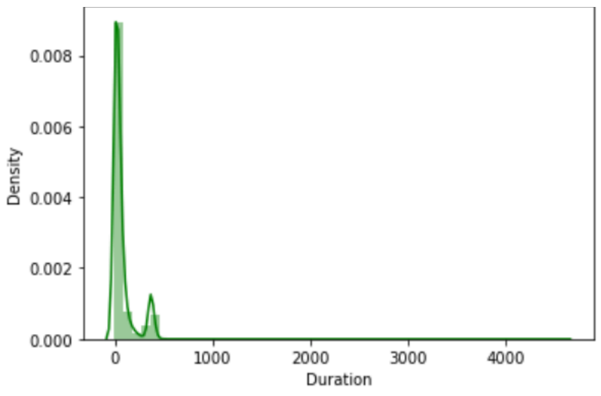
```
Description of Commision
----------------------------------------------------------------------------
count    2861.000000
mean       15.080996
std        25.826834
min         0.000000
25%         0.000000
50%         5.630000
75%        17.820000
max       210.210000
Name: Commision, dtype: float64
----------------------------------------------------------------------------
Distribution of Commision
----------------------------------------------------------------------------
```



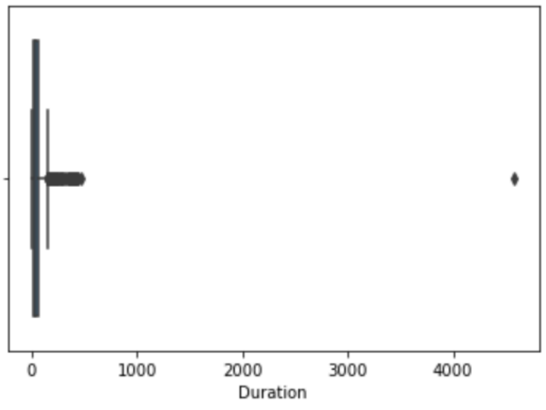
```
BoxPlot of Commision
---------------------------------------------------------------------
```



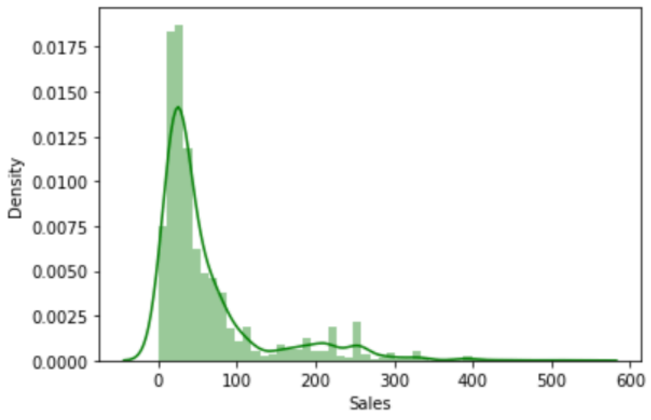
```
Description of Duration
----------------------------------------------------------------------------
count     2861.000000
mean        72.120238
std        135.977200
min         -1.000000
25%         12.000000
50%         28.000000
75%         66.000000
max       4580.000000
Name: Duration, dtype: float64
----------------------------------------------------------------------------
Distribution of Duration
----------------------------------------------------------------------------
```



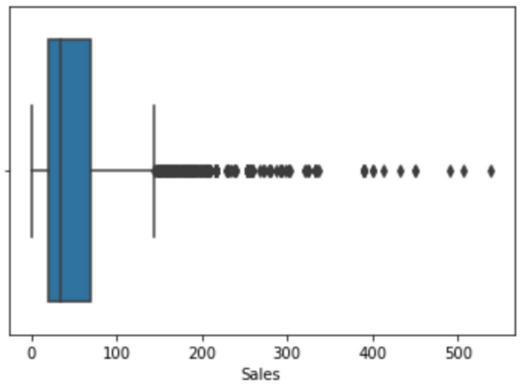
```
BoxPlot of Duration
-----------------------------------------------------------------
```



```
Description of Sales
----------------------------------------------------------------------------
count     2861.000000
mean        61.757878
std         71.399740
min          0.000000
25%         20.000000
50%         33.500000
75%         69.300000
max        539.000000
Name: Sales, dtype: float64
----------------------------------------------------------------------------
Distribution of Sales
----------------------------------------------------------------------------
```



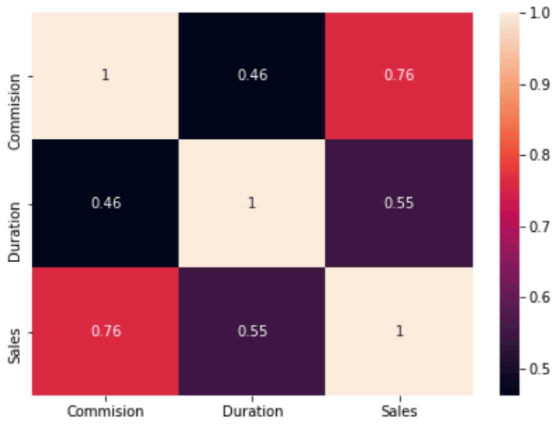
```
BoxPlot of Sales
----------------------------------------------------------------------------
```



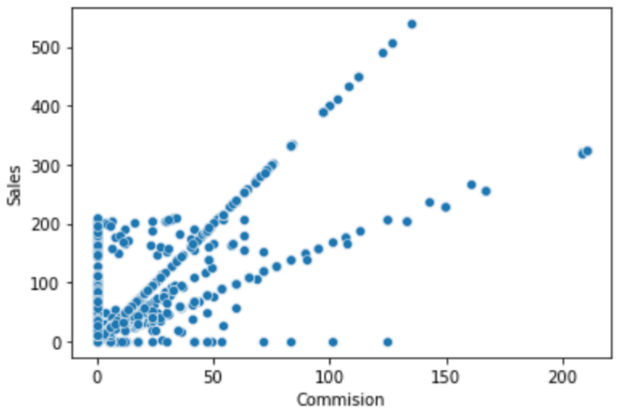
Bi-variate Analysis

As the name suggest it gives analysis of two specific variable at a time and we can derive correlation b/w them. We can use Heat-map for the same and further to highlight each correlated pair we can use scatter-plot .

Like Univariate analysis here also we can only consider only numeric fields.



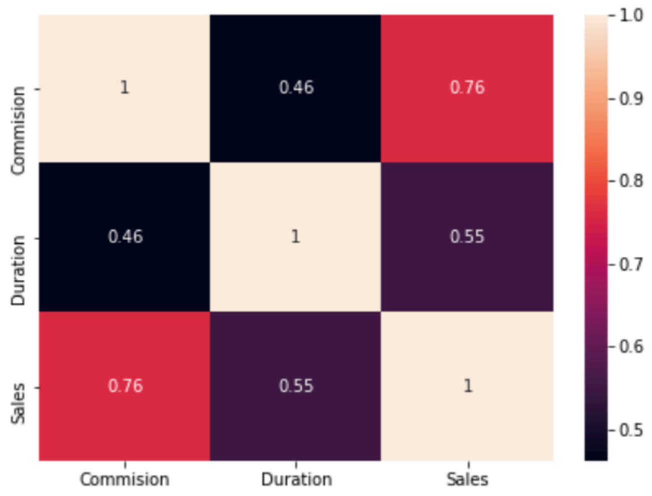
By looking at the heat-map we can see that there are 2 variables which are highly correlated . Those are **“Sales”** & **“Commision”**

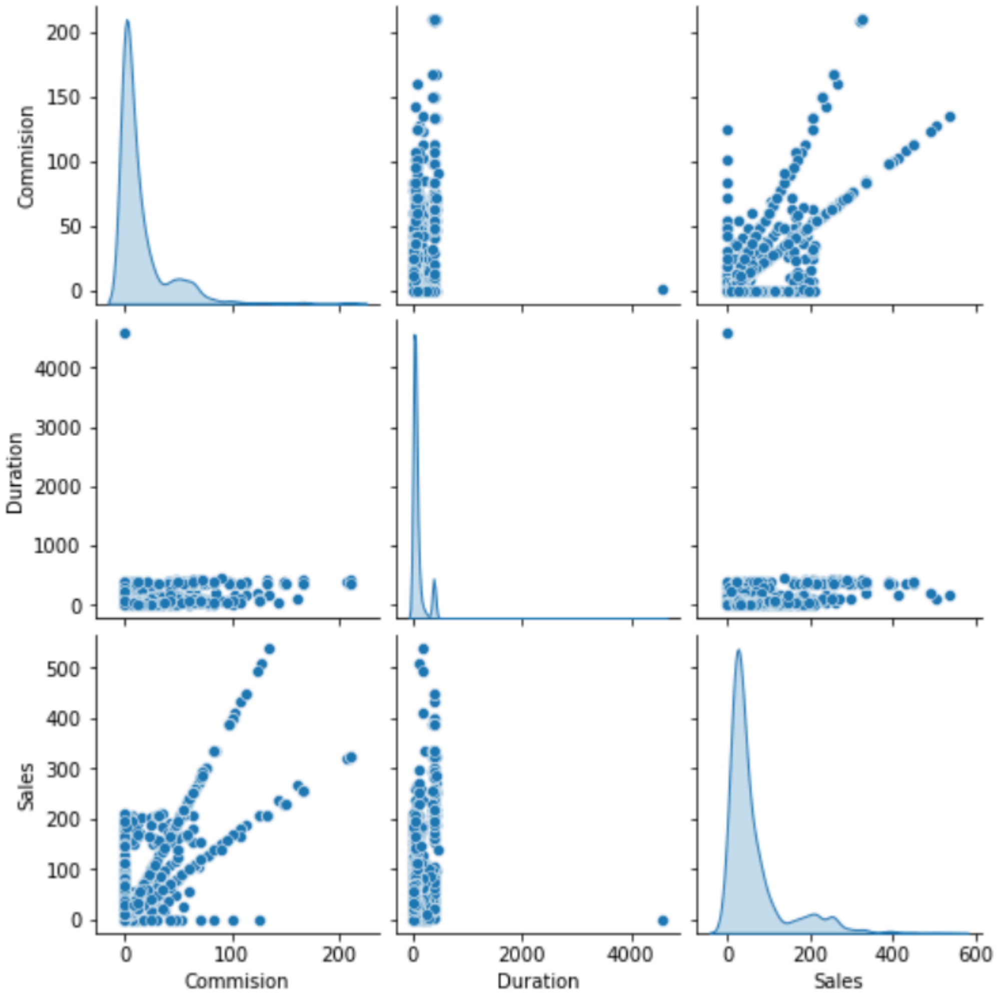


Multivariate Analysis

Multivariate means involving multiple dependent variables resulting in one outcome. For example, we cannot predict the weather of any year based on the season. There are multiple factors like pollution, humidity, precipitation, etc.

To derive **Multivariate Analysis** we have implemented **pair plot & Heat map** .





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

(Please refer notebook for better clarity)

Before doing **Train & Test split** , wanted to check the proportion of Claimed & Not Claimed

```
No     0.680531
Yes    0.319469
```

From this we can get to know almost **32%** of people has claimed the insurance and rest **68%** have not.

Here we capturing the target column **("Claimed")** into separate vectors for training set and test set by **drop()** & **pop()** mechanism.

(Please refer notebook for better clarity)

We tried keeping here 30% data in the test set and splitting data into training and test set .

Building classification model CART :

We tried building Decision Tree Classifier parameters grid and Gini-index measures the divergences between the probability distributions of the target attribute's values and splits a node in such a way which gives the least amount of impurity.

(Please refer notebook for better clarity)

A tree is generated **(hr_tree.dot)** and the same can be visualised using http://webgraphviz.com/

We saw the tree is over grown hence we wanted to prune the tree and re-generate a regularised tree.

Hence after making the model we wanted to fit the same with **X_train & train_labels**

```
GridSearchCV(cv=3, estimator=DecisionTreeClassifier(random_state=0),
             param_grid={'max_depth': [6, 7, 8, 9],
                         'min_samples_leaf': [5, 10, 15, 20],
                         'min_samples_split': [45, 60, 75]})
```

We wanted to check the **best parameters** among what we used

```
{'max_depth': 7, 'min_samples_leaf': 5, 'min_samples_split': 75}
```

After using the best parameter , regularised tree **(hr_tree_pruned.dot)** also got generated .

Post this we generated the classification report of train & test data :

Classification report of Train data :

```
              precision    recall  f1-score   support

           0       0.83      0.87      0.85      1359
           1       0.70      0.62      0.65       643

    accuracy                           0.79      2002
   macro avg       0.76      0.74      0.75      2002
weighted avg       0.79      0.79      0.79      2002
```

Classification report of Test data :

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.83       588
           1       0.64      0.58      0.61       271

    accuracy                           0.76       859
   macro avg       0.73      0.72      0.72       859
weighted avg       0.76      0.76      0.76       859
```

We wanted to check the **most important features** which helped this model to predict the analysis .

```
                   Imp
Agency_Code   0.495988
Type          0.000000
Commision     0.046303
Channel       0.009335
Duration      0.102527
Sales         0.273973
Product Name  0.045202
Destination   0.026672
```

Building classification model Random Forest :

Similar to decision tree we tried building Random forest model also based on parameter of the grid and random forest classifier , where we used

1. max_depth
2. max_features
3. min_samples_leaf
4. min_samples_split
5. n_estimators

(Please refer notebook for better clarity)

After making the model we wanted to fit the same with **X_train & train_labels**

```
GridSearchCV(cv=3, estimator=RandomForestClassifier(random_state=0),
             param_grid={'max_depth': [7, 8], 'max_features': [4, 6],
                         'min_samples_leaf': [5, 10],
                         'min_samples_split': [50, 100],
                         'n_estimators': [101, 201, 301]})
```

We wanted to check the **best parameters** among what we used

```
{'max_depth': 7,
 'max_features': 4,
 'min_samples_leaf': 5,
 'min_samples_split': 50,
 'n_estimators': 201}
```

Post this we generated the classification report of train & test data :

Classification report of Train data :

```
              precision    recall  f1-score   support

           0       0.82      0.89      0.85      1359
           1       0.72      0.59      0.65       643

    accuracy                           0.79      2002
   macro avg       0.77      0.74      0.75      2002
weighted avg       0.79      0.79      0.79      2002
```

Classification report of Test data :

```
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       588
           1       0.70      0.58      0.64       271

    accuracy                           0.79       859
   macro avg       0.76      0.73      0.74       859
weighted avg       0.78      0.79      0.78       859
```

We wanted to check the **most important features** which helped this model to predict the analysis .

```
                   Imp
Agency_Code   0.289713
Type          0.033457
Commision     0.117295
Channel       0.005229
Duration      0.097328
Sales         0.227060
Product Name  0.215517
Destination   0.014400
```

Building classification model Artificial Neural Network :

We tried building Neural Network model by using MLP Classifier along with standard scaler by fitting the train data into it .

(Please refer notebook for better clarity)

After making the model we wanted to fit the same with **X_train & train_labels**

```
GridSearchCV(cv=3, estimator=MLPClassifier(random_state=0),
             param_grid={'activation': ['logistic', 'relu'],
                         'hidden_layer_sizes': [(100, 100, 100)],
                         'max_iter': [10000], 'solver': ['sgd', 'adam'],
                         'tol': [0.1, 0.01]})
```

We wanted to check the **best parameters** among what we used

```
{'activation': 'relu',
 'hidden_layer_sizes': (100, 100, 100),
 'max_iter': 10000,
 'solver': 'adam',
 'tol': 0.01}
```

Post this we generated the classification report of train & test data :

Classification report of Train data :

```
              precision    recall  f1-score   support

           0       0.80      0.88      0.84      1359
           1       0.68      0.54      0.60       643

    accuracy                           0.77      2002
   macro avg       0.74      0.71      0.72      2002
weighted avg       0.76      0.77      0.76      2002
```

Classification report of Test data :

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       588
           1       0.69      0.51      0.59       271

    accuracy                           0.77       859
   macro avg       0.74      0.70      0.72       859
weighted avg       0.76      0.77      0.76       859
```

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.

Decision Tree / CART

(Please refer notebook for better clarity)

Classification report of Train data :

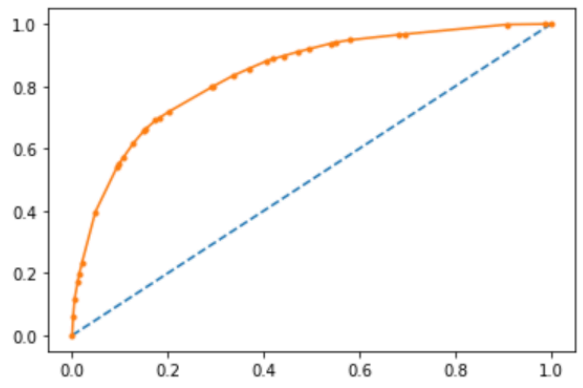
```
              precision    recall  f1-score   support

           0       0.82      0.89      0.85      1359
           1       0.72      0.59      0.65       643

    accuracy                           0.79      2002
   macro avg       0.77      0.74      0.75      2002
weighted avg       0.79      0.79      0.79      2002
```

Classification report of Train data :

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.83       588
           1       0.64      0.58      0.61       271

    accuracy                           0.76       859
   macro avg       0.73      0.72      0.72       859
weighted avg       0.76      0.76      0.76       859
```

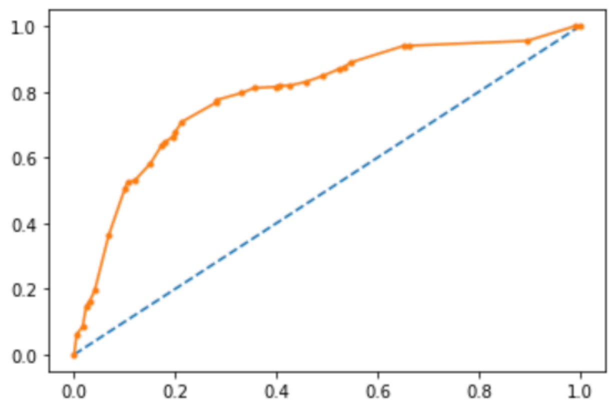
ROC Curve of Train Data :

```
AUC: 0.838
```



ROC Curve of Test Data :

```
AUC: 0.793
```



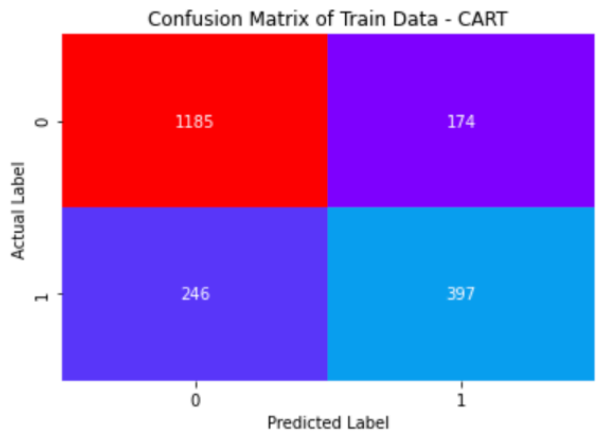
From this above report we can figure out that Accuracy , precision , recall & f1-score of the train data & test data are as below

cart_train_acc = 0.79
cart_train_precision = 0.70
cart_train_recall = 0.62
cart_train_f1_score = 0.65
cart_train_auc = 0.838

cart_test_acc = 0.76
cart_test_precision = 0.64
cart_test_recall = 0.58
cart_test_f1_score = 0.61
cart_test_auc = 0.793

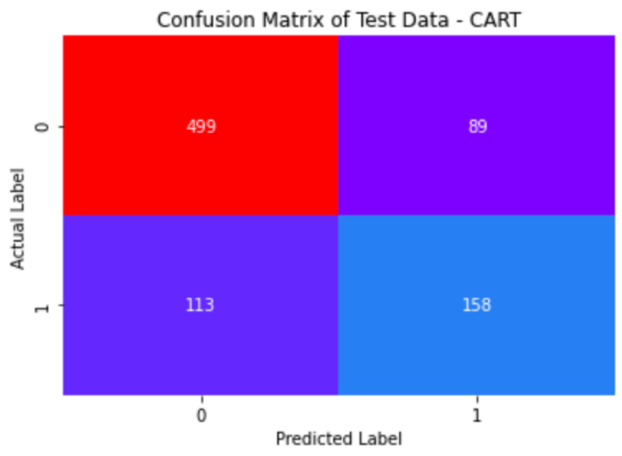
Hence we can conclude saying its a good-fit model as there is not much difference b/w train & test data. Recall is little better in test data where as accuracy , precision & f1-score are little below but close to train data.

Confusion matrix of train data :



True positive is 397
True negative is 1185
False positive is 174
False negative is 246

Confusion matrix of test data :



True positive is 158
True negative is 499
False positive is 89
False negative is 113

Random Forest

(Please refer notebook for better clarity)

Classification report of Train data :

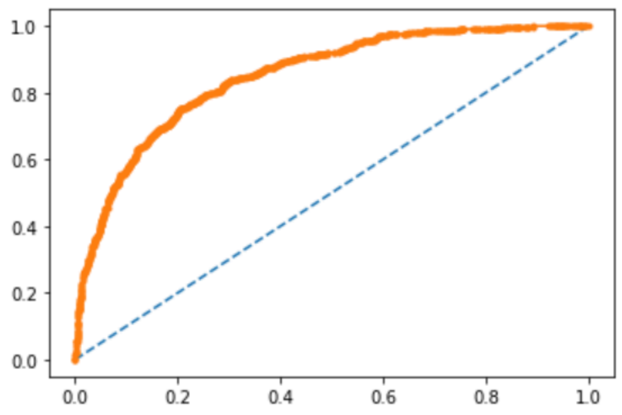
```
              precision    recall  f1-score   support

           0       0.82      0.89      0.85      1359
           1       0.72      0.59      0.65       643

    accuracy                           0.79      2002
   macro avg       0.77      0.74      0.75      2002
weighted avg       0.79      0.79      0.79      2002
```

Classification report of Train data :

```
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       588
           1       0.70      0.58      0.64       271

    accuracy                           0.79       859
   macro avg       0.76      0.73      0.74       859
weighted avg       0.78      0.79      0.78       859
```

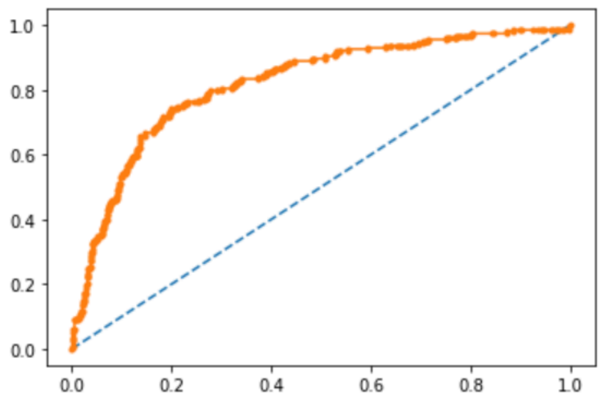
ROC Curve of Train Data :

```
AUC: 0.850
```



ROC Curve of Test Data :

```
AUC: 0.821
```



From this above report we can figure out that Accuracy , precision , recall & f1-score of the train data & test data are as below

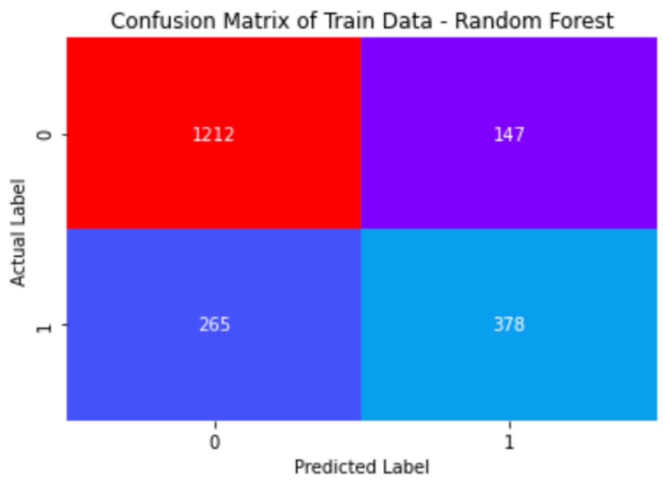
rf_train_acc = 0.79
rf_train_precision = 0.72
rf_train_recall = 0.59
rf_train_f1_score = 0.65
rf_train_auc = 0.850

rf_test_acc = 0.79
rf_test_precision = 0.70
rf_test_recall = 0.58

rf_test_f1_score = 0.64
rf_test_auc = 0.821

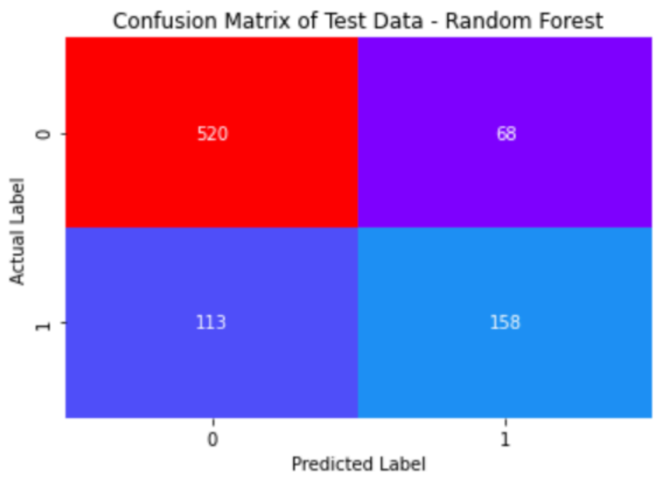
From this above value we can derive that compared to CART this model is even better as Accuracy , Precision , f1-score are better in both train & test data . In comparison with train data with test there is no much difference so we can consider this also as a good-fit model .

Confusion matrix of train data :



True positive is 378
True negative is 1212
False positive is 147
False negative is 265

Confusion matrix of train data :



True positive is 158
True negative is 520
False positive is 68
False negative is 113

We can see that here false positive rate is reduced that means (people who are not claimed but marked as claimed) its a better model compared to cart.

Artificial Neural Network

(Please refer notebook for better clarity)

Classification report of Train data :

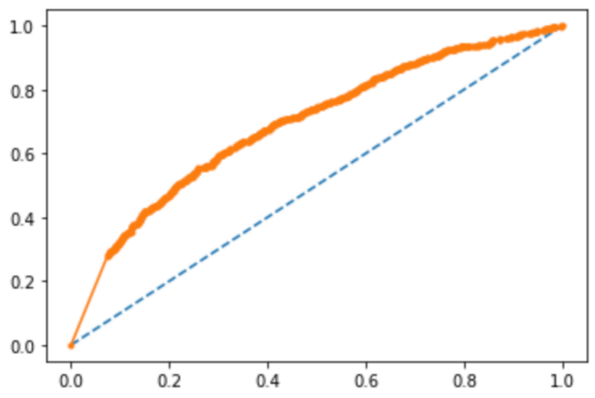
```
              precision    recall  f1-score   support

           0       0.80      0.88      0.84      1359
           1       0.68      0.54      0.60       643

    accuracy                           0.77      2002
   macro avg       0.74      0.71      0.72      2002
weighted avg       0.76      0.77      0.76      2002
```

Classification report of Test data :

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       588
           1       0.69      0.51      0.59       271

    accuracy                           0.77       859
   macro avg       0.74      0.70      0.72       859
weighted avg       0.76      0.77      0.76       859
```

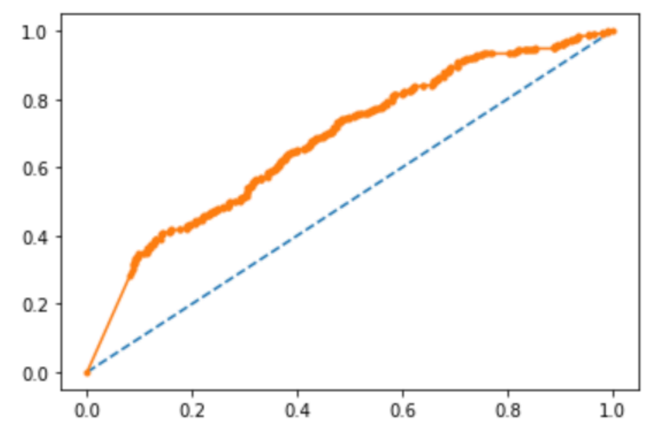
ROC Curve of Train Data :

```
AUC: 0.693
```



ROC Curve of Test Data :

AUC: 0.682



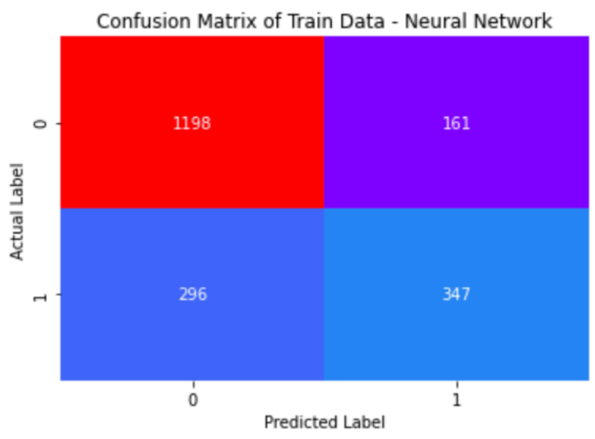
From this above report we can figure out that Accuracy , precision , recall & f1-score of the train data & test data are as below

nl_train_acc = 0.77
nl_train_precision = 0.68
nl_train_recall = 0.54
nl_train_f1_score = 0.60
nl_train_auc = 0.693

nl_test_acc = 0.77
nl_test_precision = 0.69
nl_test_recall = 0.51
nl_test_f1_score = 0.59
nl_test_auc = 0.682

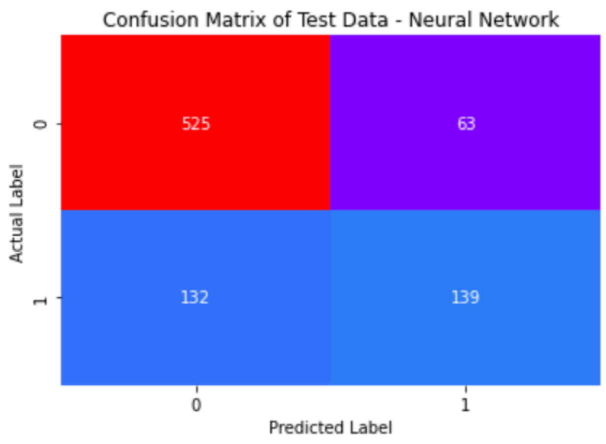
From this above value we can derive that compared to CART & random forest this model is performed worse as Accuracy , Precision , f1-score are worst in both train & test data among all 3 models. In comparison with train data with test there is no much difference so we can consider this also as a good-fit model .

Confusion matrix of train data :



True positive is 347
True negative is 1198
False positive is 161
False negative is 296

Confusion matrix of test data :



True positive is 139
True negative is 525
False positive is 63
False negative is 132

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

(Please refer notebook for better clarity)

From the above evaluation of all 3 models (Through classification report & AUC score) we got below values as performance marix

CART Train classification report :

```
#           precision    recall  f1-score   support

#        0       0.83      0.87      0.85      1359
#        1       0.70      0.62      0.65       643

#    accuracy                           0.79      2002
#   macro avg       0.76      0.74      0.75      2002
# weighted avg       0.79      0.79      0.79      2002

#  AUC: 0.838
```

cart_train_acc = 0.79
cart_train_precision = 0.70
cart_train_recall = 0.62
cart_train_f1_score = 0.65
cart_train_auc = 0.838

##### CART Test classification report :

```
#           precision    recall  f1-score   support

#        0       0.82      0.85      0.83       588
#        1       0.64      0.58      0.61       271

#    accuracy                           0.76       859
#   macro avg       0.73      0.72      0.72       859
# weighted avg       0.76      0.76      0.76       859

#  AUC: 0.793
```

cart_test_acc = 0.76
cart_test_precision = 0.64
cart_test_recall = 0.58
cart_test_f1_score = 0.61
cart_test_auc = 0.793

RF Train classification report :

```
#            precision    recall  f1-score   support

#          0       0.82      0.89      0.85      1359
#          1       0.72      0.59      0.65       643

#     accuracy                          0.79      2002
#    macro avg       0.77      0.74      0.75      2002
# weighted avg       0.79      0.79      0.79      2002

#   AUC: 0.850
```

rf_train_acc = 0.79
rf_train_precision = 0.72
rf_train_recall = 0.59
rf_train_f1_score = 0.65
rf_train_auc = 0.850

RF Test classification report :

```
#            precision    recall  f1-score   support

#          0       0.82      0.88      0.85       588
#          1       0.70      0.58      0.64       271

#     accuracy                          0.79       859
#    macro avg       0.76      0.73      0.74       859
# weighted avg       0.78      0.79      0.78       859

#   AUC: 0.821
```

rf_test_acc = 0.79
rf_test_precision = 0.70
rf_test_recall = 0.58
rf_test_f1_score = 0.64
rf_test_auc = 0.821

ANN Train classification report :

```
#            precision    recall  f1-score   support

#          0       0.80      0.88      0.84      1359
#          1       0.68      0.54      0.60       643

#     accuracy                          0.77            2002
#    macro avg       0.74      0.71      0.72      2002
# weighted avg       0.76      0.77      0.76      2002

# AUC: 0.693
```

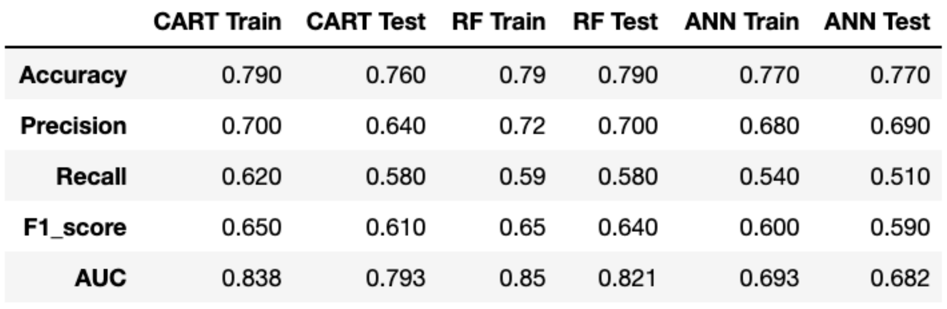
nl_train_acc = 0.77
nl_train_precision = 0.68
nl_train_recall = 0.54
nl_train_f1_score = 0.60
nl_train_auc = 0.693

ANN Test classification report :

```
#            precision    recall  f1-score   support

#          0       0.80      0.89      0.84       588
#          1       0.69      0.51      0.59       271

#     accuracy                          0.77            859
#    macro avg       0.74      0.70      0.72       859
# weighted avg       0.76      0.77      0.76       859

# AUC: 0.682
```

nl_test_acc = 0.77
nl_test_precision = 0.69
nl_test_recall = 0.51
nl_test_f1_score = 0.59
nl_test_auc = 0.682

We are creating a Comparison table based on above 3 classification (CART, RF ,ANN) reports
(Please refer notebook for better clarity)



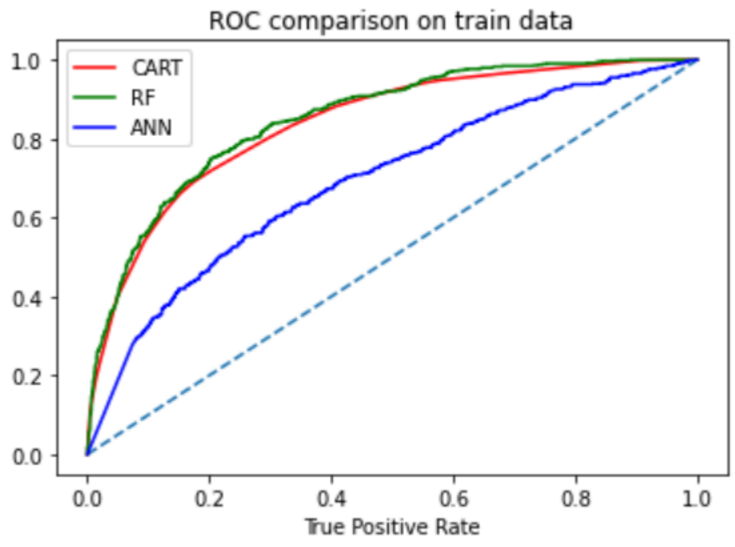
	CART Train	CART Test	RF Train	RF Test	ANN Train	ANN Test
Accuracy	0.790	0.760	0.79	0.790	0.770	0.770
Precision	0.700	0.640	0.72	0.700	0.680	0.690
Recall	0.620	0.580	0.59	0.580	0.540	0.510
F1_score	0.650	0.610	0.65	0.640	0.600	0.590
AUC	0.838	0.793	0.85	0.821	0.693	0.682

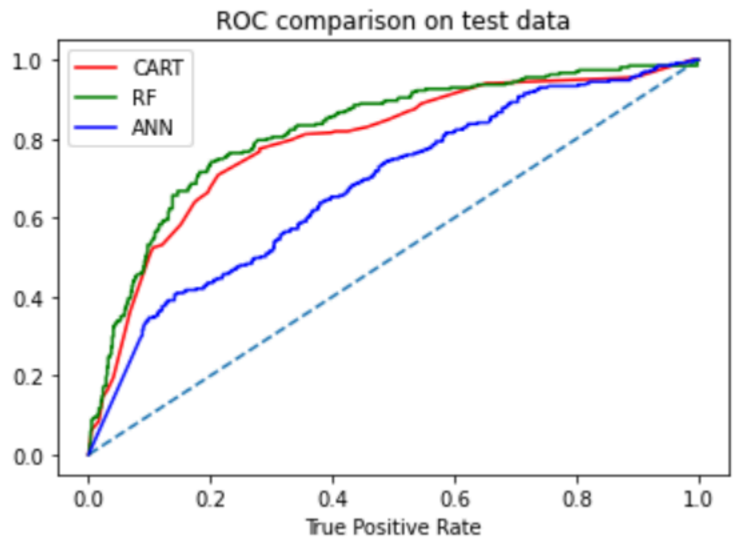
From the above table it is evident that by comparing all the performance matrix among all 3 classification models that ANN is not a considerable model for this problem statement as all the performance matrix values are pretty less than other two model (CART & RF)

Comparing CART & RF , we felt Random Forest (RF) performed better as mostly all performance matrix has a better result than CART .

We can derive the similar conclusion by plotting all ROC curve in same axis (based on train & test data)

(Please refer notebook for better clarity)



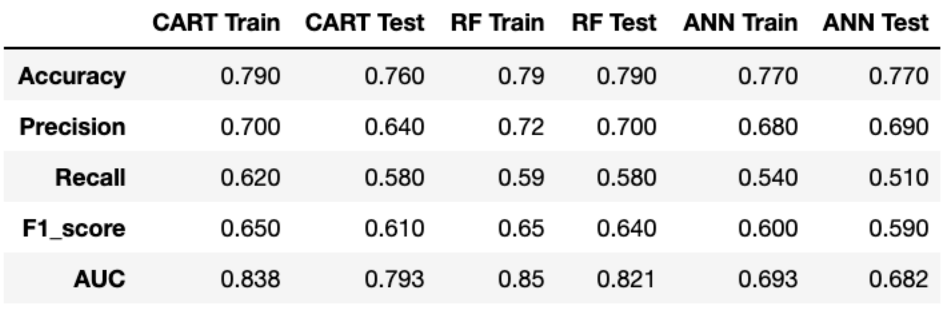


From the above ROC curve on the test data among all 3 classification models Random Forest has the maximum area of coverage. Hence we can conclude that Random Forest (RF) is the best fit model for the same .

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

(Please refer notebook for better clarity)

From the Comparison table based on performance matrix below



	CART Train	CART Test	RF Train	RF Test	ANN Train	ANN Test
Accuracy	0.790	0.760	0.79	0.790	0.770	0.770
Precision	0.700	0.640	0.72	0.700	0.680	0.690
Recall	0.620	0.580	0.59	0.580	0.540	0.510
F1_score	0.650	0.610	0.65	0.640	0.600	0.590
AUC	0.838	0.793	0.85	0.821	0.693	0.682

it is evident that by comparing all the performance matrix among all 3 classification models that ANN is not a considerable model for this problem statement as all the performance matrix values are pretty less than other two model (CART & RF)

Comparing CART & RF , we felt Random Forest (RF) performed better as mostly all performance matrix has a better result than CART .

Here in Random Forest has relatively better accuracy, Precision , f1-score & AUC compared with other 2 models.

Hence Random Forest is the best suited model for this .

As per the feature importance "Channel" & "Destination" are the features which helps majorly to predict future. Insurance company needs to concentrate on this two variable.

THE END