# DATA MINING (PROJECT)

Business Report

By: Shubhank Katarey

27 June 2021

## Table of Contents

## Problem 1: Clustering

**Problem Statement:**

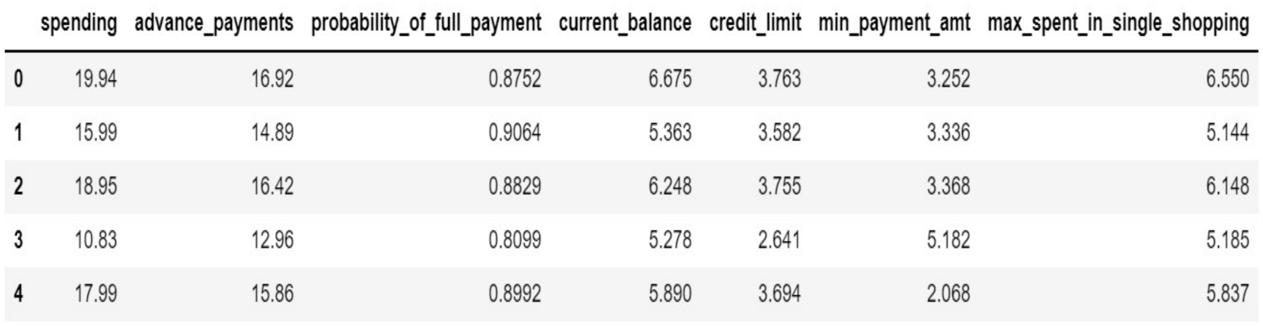
A leading bank wants to develop a customer segmentation to give promotional offers to its customers. They collected a sample that summarizes the activities of users during the past few months. You are given the task to identify the segments based on credit card usage.

**Data Dictionary for Market Segmentation:**

1) spending: Amount spent by the customer per month (in 1000s)
2) advance_payments: Amount paid by the customer in advance by cash (in 100s)
3) probability_of_full_payment: Probability of payment done in full by the customer to the bank
4) current_balance: Balance amount left in the account to make purchases (in 1000s)
5) credit_limit: Limit of the amount in credit card (10000s)
6) min_payment_amt : minimum paid by the customer while making payments for purchases made monthly (in 100s)
7) max_spent_in_single_shopping: Maximum amount spent in one purchase (in 1000s)

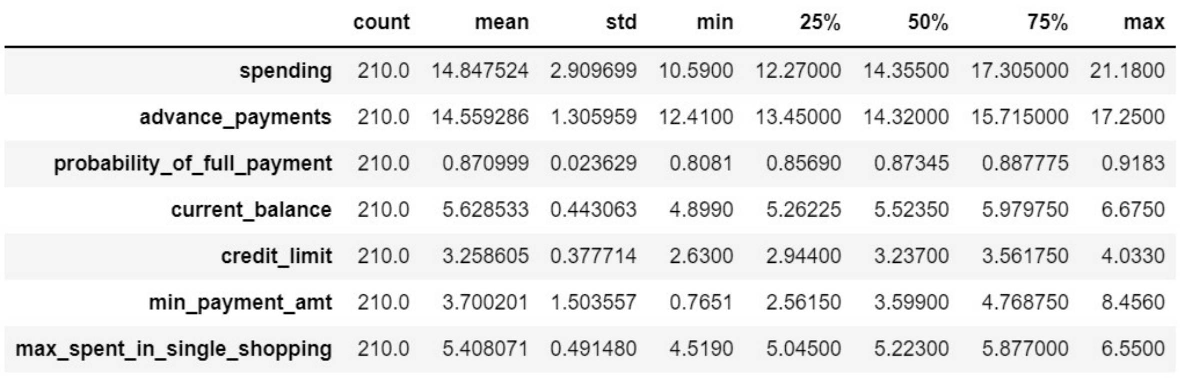
### 1.1 Read the data, do the necessary initial steps, and exploratory data analysis (Univariate, Bi-variate, and multivariate analysis).

Checking head of the data:



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

Describing the data:

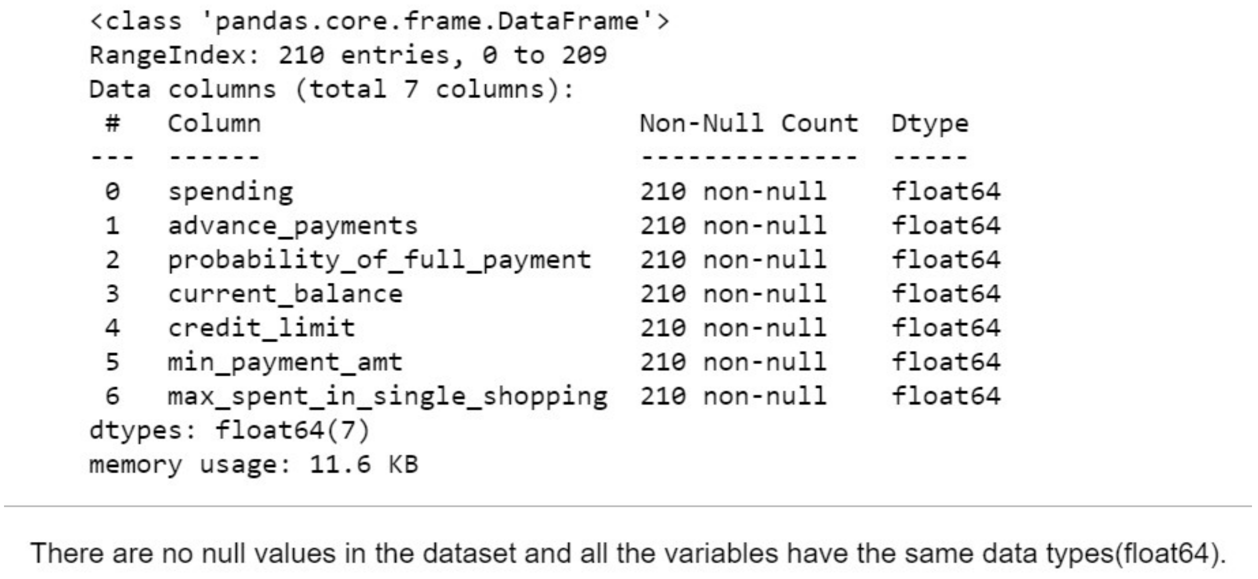


| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| spending | 210.0 | 14.847524 | 2.909699 | 10.5900 | 12.27000 | 14.35500 | 17.305000 | 21.1800 |
| advance_payments | 210.0 | 14.559286 | 1.305959 | 12.4100 | 13.45000 | 14.32000 | 15.715000 | 17.2500 |
| probability_of_full_payment | 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 |

<u>Checking shape of the data:</u>

Number of rows: 210
Number of columns: 7

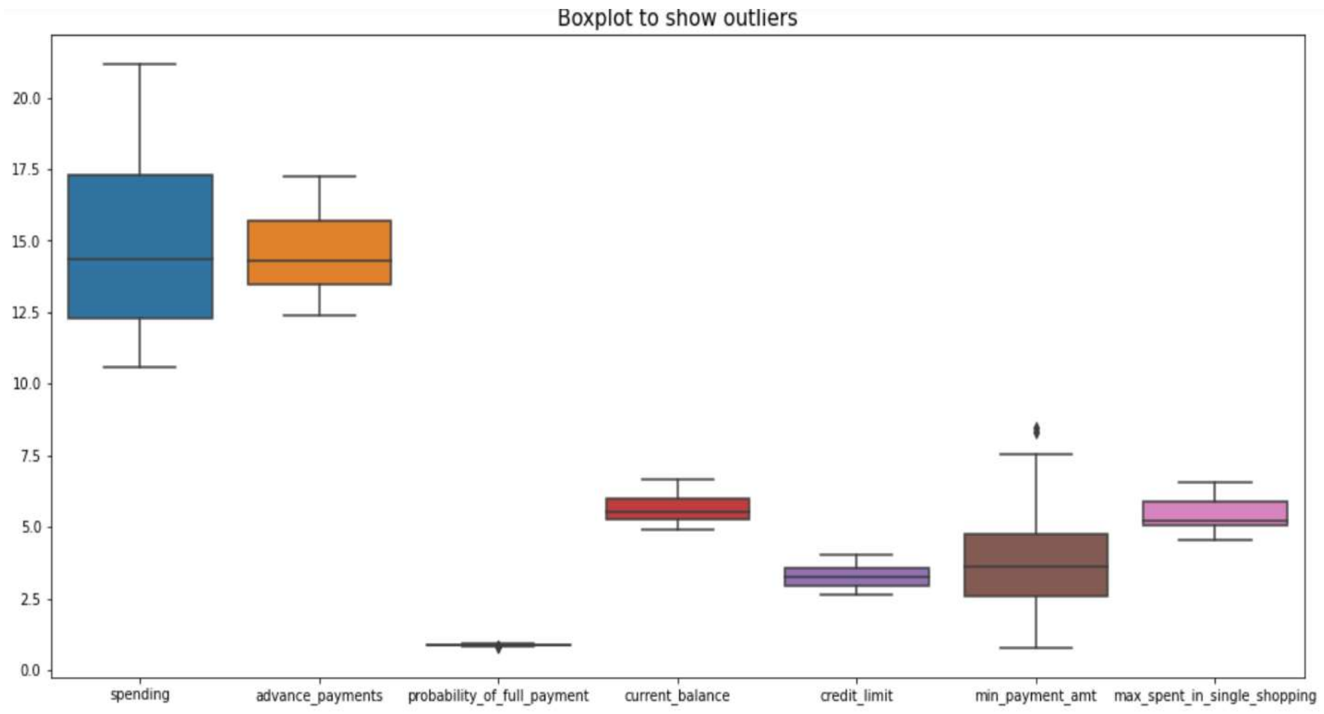
<u>Checking the info of the data:</u> We can see that there are no null/missing values in the dataset and all the variables have the same datatype float.



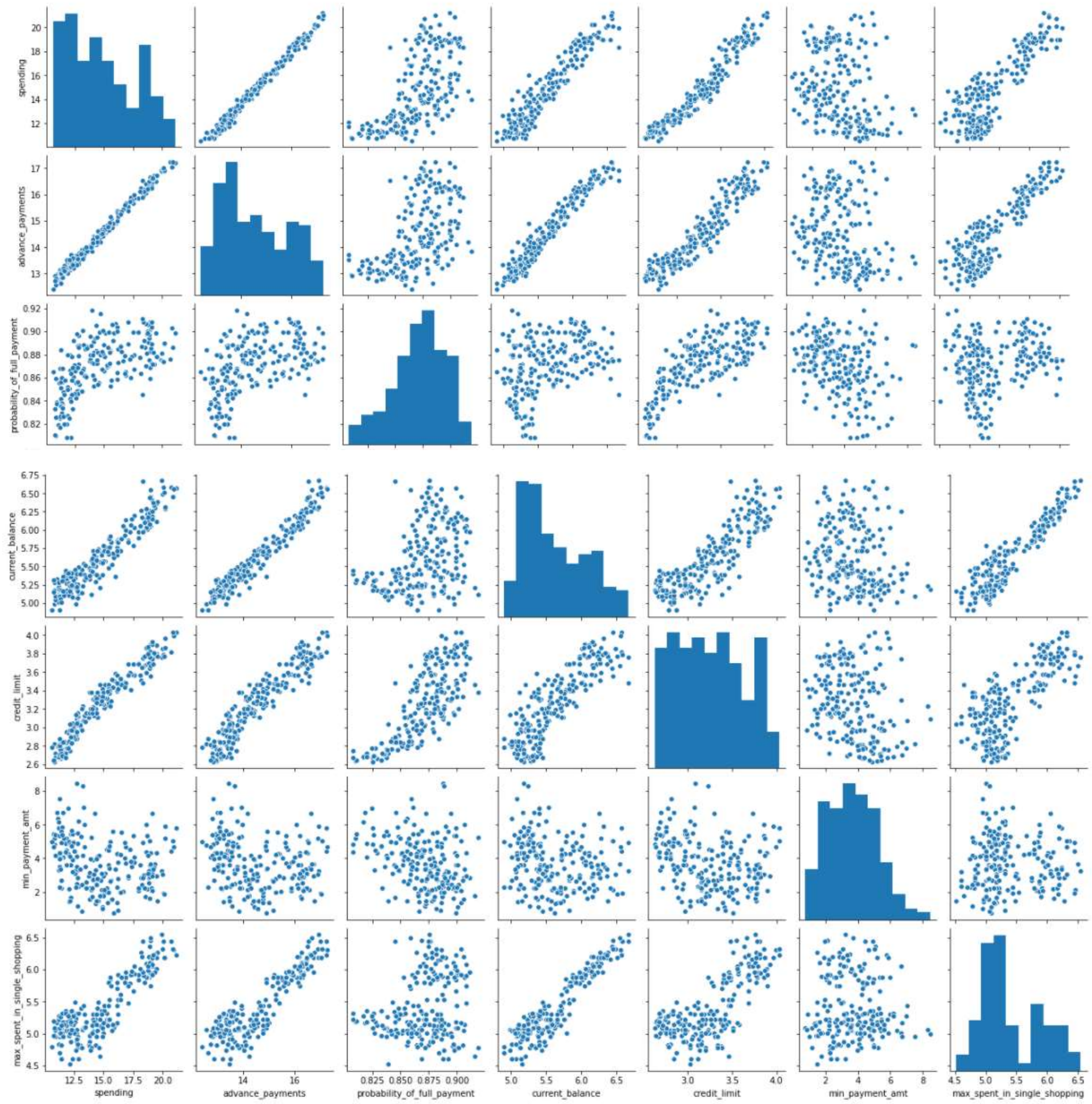
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210 entries, 0 to 209
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   spending                      210 non-null    float64
 1   advance_payments              210 non-null    float64
 2   probability_of_full_payment   210 non-null    float64
 3   current_balance               210 non-null    float64
 4   credit_limit                  210 non-null    float64
 5   min_payment_amt               210 non-null    float64
 6   max_spent_in_single_shopping  210 non-null    float64
dtypes: float64(7)
memory usage: 11.6 KB
```

There are no null values in the dataset and all the variables have the same data types(float64).

<u>Checking the Outliers using Boxplot:</u> Boxplot performed on non-scaled data. That's why it is not unif orm in nature. There is outlier present in 'min_payment_amt'.



**Pair plot:**



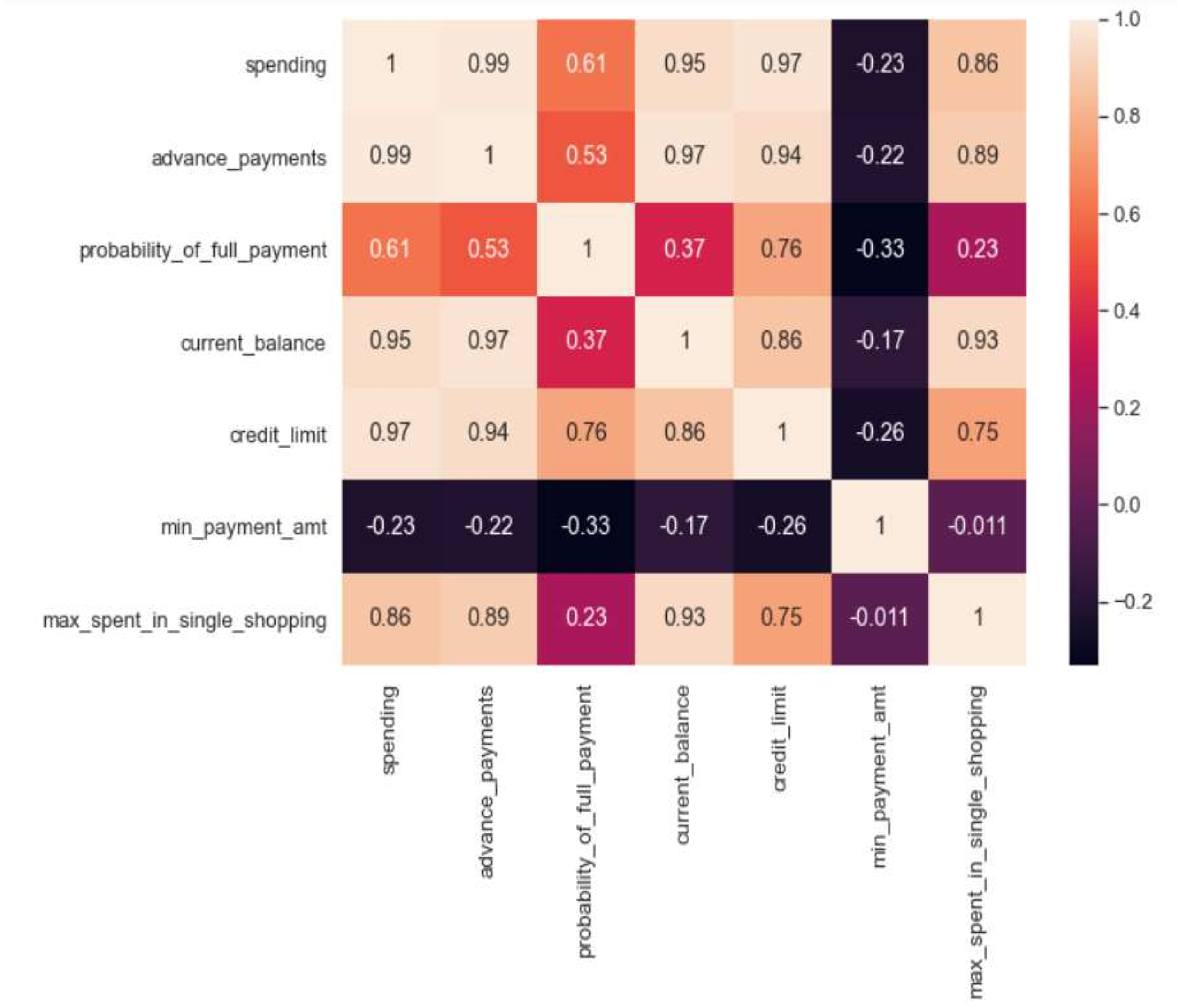
Few Observations from the above Pair Plot:

By observing the above pair plot, we can see that, more the credit limit, more the spending.

Users with more credit limit has the more probability of full payment.

If the spending is less, the advance payment is more. With the increase in spending, customers using credit card instead of advance payment by cash.

Heatmap Showing the Correlation:



| | spending | advance_payments | probability_of_full_payment | current_balance | credit_limit | min_payment_amt | max_spent_in_single_shopping |
|---|---|---|---|---|---|---|---|
| spending | 1 | 0.99 | 0.61 | 0.95 | 0.97 | -0.23 | 0.86 |
| advance_payments | 0.99 | 1 | 0.53 | 0.97 | 0.94 | -0.22 | 0.89 |
| probability_of_full_payment | 0.61 | 0.53 | 1 | 0.37 | 0.76 | -0.33 | 0.23 |
| current_balance | 0.95 | 0.97 | 0.37 | 1 | 0.86 | -0.17 | 0.93 |
| credit_limit | 0.97 | 0.94 | 0.76 | 0.86 | 1 | -0.26 | 0.75 |
| min_payment_amt | -0.23 | -0.22 | -0.33 | -0.17 | -0.26 | 1 | -0.011 |
| max_spent_in_single_shopping | 0.86 | 0.89 | 0.23 | 0.93 | 0.75 | -0.011 | 1 |

Observations from the above Heat Map:

The correlation is very good (0.97) between **credit_limit** and **spending**.

The correlation is very poor (-0.33) between **min_payment_amt and probability_of_full_payment.**

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

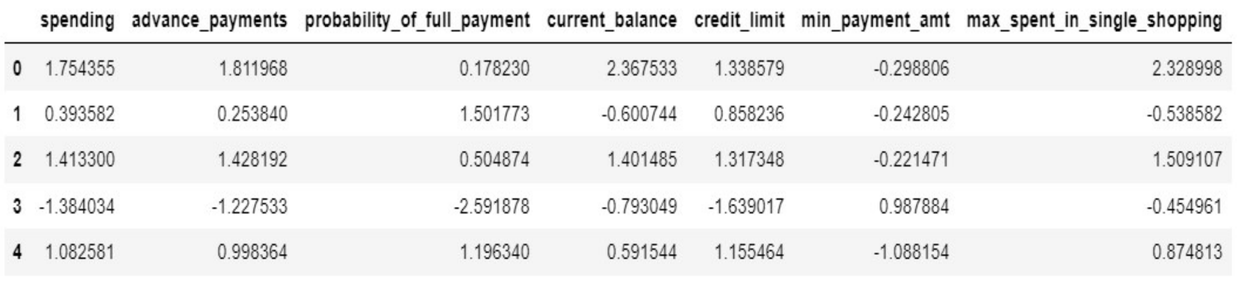
To bring all features in the same standing, we need to do scaling in this case, so that one significant number doesn’t impact the further calculations just because of their large magnitude.

We have variables in our dataset which are measured in different scales, some are in 100's , some in 1000's and some in 10000's, so it is useful to scale the data.

We are using 'zscore' from 'scipy.stats' to scale our dataset.

Z scores indicate how many standard deviations an observation is above or below the mean. These scores are a useful way of putting data from different sources onto the same scale.

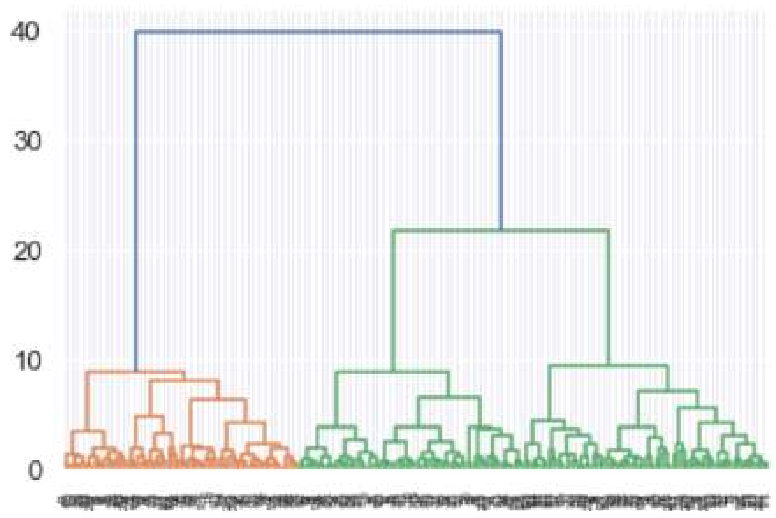
Scaled Dataset:



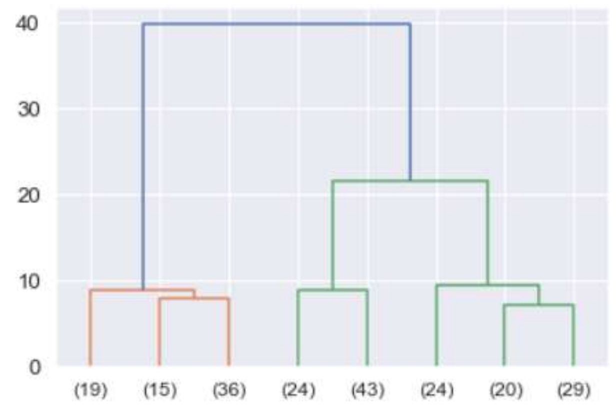
| | spending | advance_payments | probability_of_full_payment | current_balance | credit_limit | min_payment_amt | max_spent_in_single_shopping |
|---|---|---|---|---|---|---|---|
| 0 | 1.754355 | 1.811968 | 0.178230 | 2.367533 | 1.338579 | -0.298806 | 2.328998 |
| 1 | 0.393582 | 0.253840 | 1.501773 | -0.600744 | 0.858236 | -0.242805 | -0.538582 |
| 2 | 1.413300 | 1.428192 | 0.504874 | 1.401485 | 1.317348 | -0.221471 | 1.509107 |
| 3 | -1.384034 | -1.227533 | -2.591878 | -0.793049 | -1.639017 | 0.987884 | -0.454961 |
| 4 | 1.082581 | 0.998364 | 1.196340 | 0.591544 | 1.155464 | -1.088154 | 0.874813 |

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

Below is the Dendrogram after applying Hierarchical Clustering:

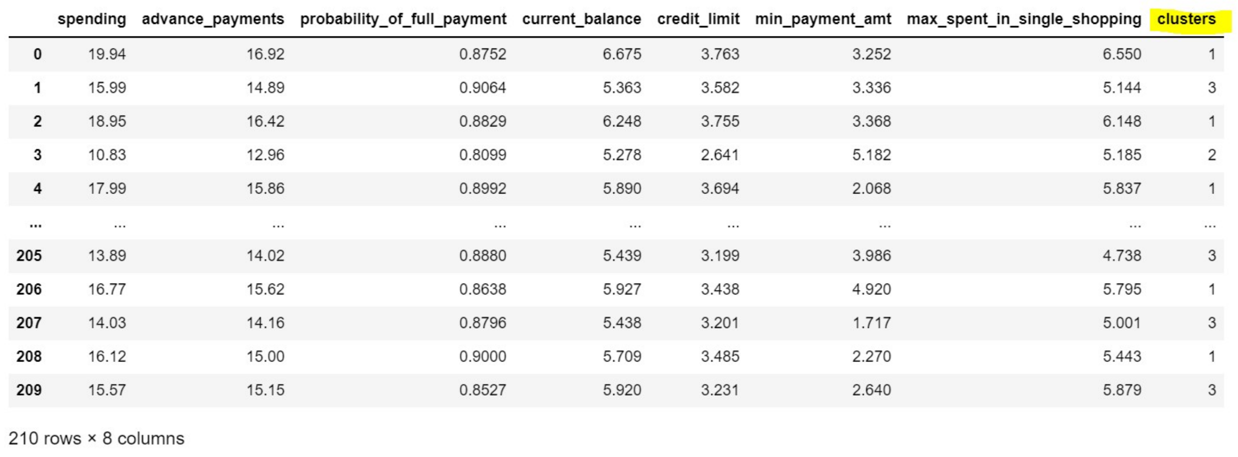


Dendrogram after passing p=8:



Here, truncate_mode='lastp', shows only the last p merged clusters and p=8, shows only the last 8 merged clusters.

We are using 'maxclust' criteria for making clusters and adding a column into the dataset and then the data will divide into the clusters accordingly.

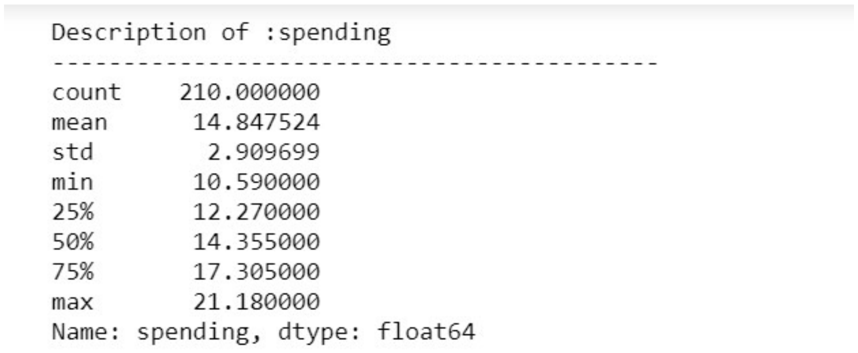


| | spending | advance_payments | probability_of_full_payment | current_balance | credit_limit | min_payment_amt | max_spent_in_single_shopping | clusters |
|---|---|---|---|---|---|---|---|---|
| **0** | 19.94 | 16.92 | 0.8752 | 6.675 | 3.763 | 3.252 | 6.550 | 1 |
| **1** | 15.99 | 14.89 | 0.9064 | 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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **205** | 13.89 | 14.02 | 0.8880 | 5.439 | 3.199 | 3.986 | 4.738 | 3 |
| **206** | 16.77 | 15.62 | 0.8638 | 5.927 | 3.438 | 4.920 | 5.795 | 1 |
| **207** | 14.03 | 14.16 | 0.8796 | 5.438 | 3.201 | 1.717 | 5.001 | 3 |
| **208** | 16.12 | 15.00 | 0.9000 | 5.709 | 3.485 | 2.270 | 5.443 | 1 |
| **209** | 15.57 | 15.15 | 0.8527 | 5.920 | 3.231 | 2.640 | 5.879 | 3 |

210 rows × 8 columns

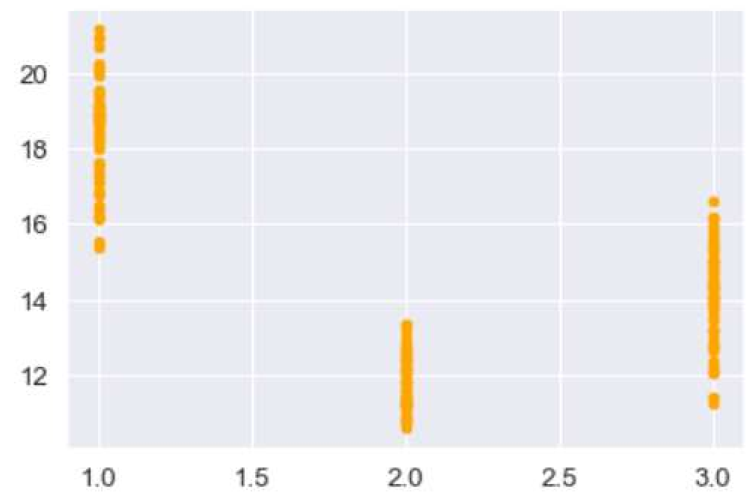
We also have to understand the customer's segmentation with respect to main features like spending, credit_limit, max_spent_in_single_shopping etc.

So first we are getting the information about customer's spending:



```
Description of :spending
--------------------------------------------
count    210.000000
mean      14.847524
std        2.909699
min       10.590000
25%       12.270000
50%       14.355000
75%       17.305000
max       21.180000
Name: spending, dtype: float64
```

```
Customer segmentation with respect to :spending
-------------------------------------------
```



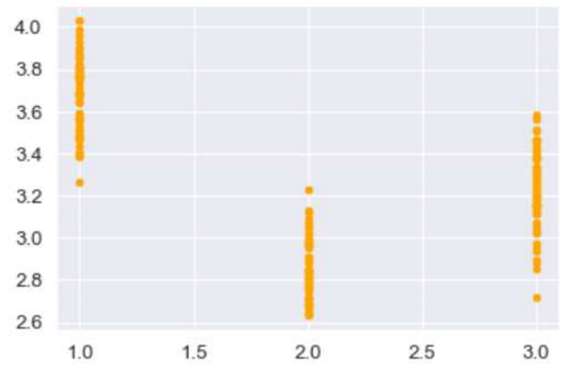
**Observation**:

We can see maximum monthly expenditure is done by the customers of cluster 1. The maximum amount that spent monthly is around 21000 by the user from cluster 1. The minimum amount spent monthly is around 10000 from the user of cluster 2.

Description of credit_limit and customer's segmentation with respect to credit_limit:

```
Description of :credit_limit
-------------------------------------------
count    210.000000
mean       3.258605
std        0.377714
min        2.630000
25%        2.944000
50%        3.237000
75%        3.561750
max        4.033000
Name: credit_limit, dtype: float64
```

```
Customer segmentation with respect to :credit_limit
-------------------------------------------
```



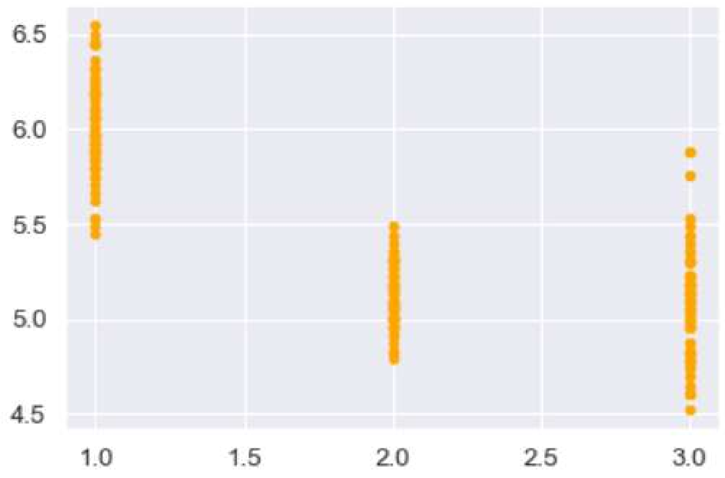
**Observation:**

We can see that the max credit_limit is around 40k and this falls under cluster 1, and minimum credit limit is around 2.6k which comes under the cluster 2.

Due to the high credit limit holding by most of the customers, the mean is coming around 3.2k.

Description of max_spent_in_single_shopping:

```
Description of :max_spent_in_single_shopping
--------------------------------------------
count    210.000000
mean       5.408071
std        0.491480
min        4.519000
25%        5.045000
50%        5.223000
75%        5.877000
max        6.550000
Name: max_spent_in_single_shopping, dtype: float64

Customer segmentation with respect to :max_spent_in_single_shopping
--------------------------------------------
```



**Observation:**

Maximum amount spent in single shopping is also done by the user of cluster 1 that is around 6500. Minimum amount spent in single shopping is done by the user of cluster 3 that is around 4500.

## 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.

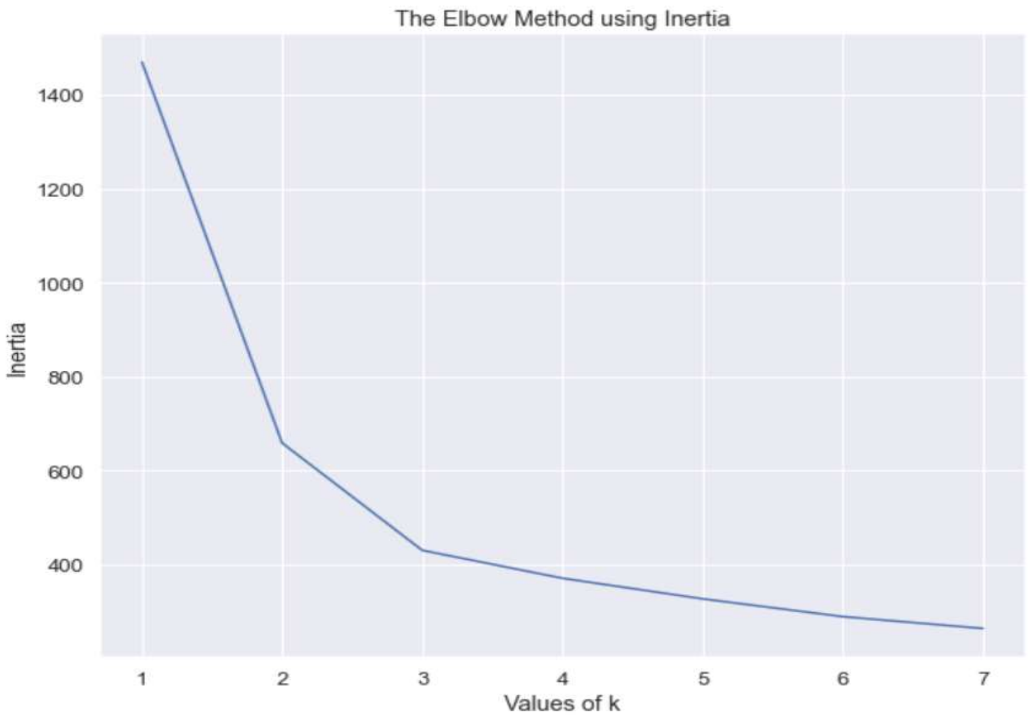
Now by using KMeans clustering we can check for the number of optimum clusters required to decide the customer's segmentation for the given dataset.

So we are assigning values of k from k=1 till k=7 and will get the value of within sum of squares (wss) or inertia to decide the number of optimum clusters:

```
For k=1, the inertia is 1469.9999999999998
For k=2, the inertia is 659.171754487041
For k=3, the inertia is 430.6589731513006
For k=4, the inertia is 371.1846125351018
For k=5, the inertia is 327.3281094192775
For k=6, the inertia is 289.46717056412893
For k=7, the inertia is 264.3052234087485
```

We can see, there is significant drop in the values from 'k=1' till 'k=3', and after that there is not much difference in the values as we go further from 4th to 7th.

To get more clarity about the drop in values and to decide the clusters selection, we will go ahead and draw an elbow graph using the inertia values that we have calculated above for different values of k:



To get the optimal number of clusters, we have to select the value of k at the 'elbow' that we can see in the above plot. Generally, 'elbow' is the point after which the values of the inertia start decreasing in a linear fashion.

Similarly, in our above elbow plot/method we can see that the values of inertia dropping slowly after k=3.

Therefore for our dataset, we conclude that the optimal number of clusters for the data is 3.

Now once we have decided the number of clusters for the dataset, we can assign the labels of kmeans clusters into our dataset and the records will divide into those clusters by using '.fit' method.

Further we can calculate silhouette score and sil-width:

Silhouette Score: 0.4007270552751299

As we can see that the Silhouette score for our data is positive. This means that the clusters are well separated.

Sil-width is also positive, that means the mapping is correct to its centroid.

After assigning the sil-width into our dataset, we have stored and saved the dataset into different csv file.

We come out with few following inferences:

**Inference:**

We can see that there is not much decline in the values as we go further from 4th to 7th clusters, that's why we have decided to keep the optimum number of cluster equal to 3.

By using KMeans we got the number of clusters is 3 but still for more clarity we can see it on the elbow method using inertia. In the above elbow plot/method also, we can see that the values of inertia dropping significantly after k=3.

The data is well separated within these 3 clusters. This we can say by directly looking at the silhouette score.

Therefore for our dataset, we conclude that the optimal number of clusters for the data is 3.

#NOTE: Inertia is the sum of squared distances of samples to their closest cluster center.

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

**Observation:**

Once the dataset is clustered properly, we can look into the data and say that:

In first cluster, there are the people with credit limit between 30,000 and 40,000 are spending around 6000 to 7000 every time they go to shopping and also their monthly spending is high around 19,000.

In second cluster, there are the people with credit limit between 25,000 and 30,000(around) are spending around 4000 to 5500 every time they go to shopping and also their monthly spending is moderate around 11,000.

In third cluster, there are the people with credit limit around 30,000 are spending around 4000 to 5000 every time they go to shopping but their monthly spending is less in compared to the people from first cluster, around 14,000 and their probability of full payment is higher than others.

**Recommendation:**

So based on above observations, my recommendation is that bank should give attractive offers to the customers present in the third cluster, so that they can utilize their credit limit and spend somewhere around or more than the people of first cluster. This we can say because customers in first and third clusters have approximately same amount of credit limit but there is difference in their monthly spending.

Also, bank should give promotional offers to the second cluster members, because their monthly spend is less. The bank should collect information about their area of interest, like where they usually visit, which category they often chose to buy while shopping, so that bank can give some descent offers them as per their requirement.

The bank should provide some easy to pay EMI options to pay their credit card bill to the members of second and third clusters, so that they can utilize their credit card amount to some further extent and can pay the bill without any fear.

As per the records after clustering, the bank should focus more on members of third clusters and should give them the promotional offers more often.

____________________________________END________________________________________

## Problem 2: CART-RF-ANN

**Problem Statement:**

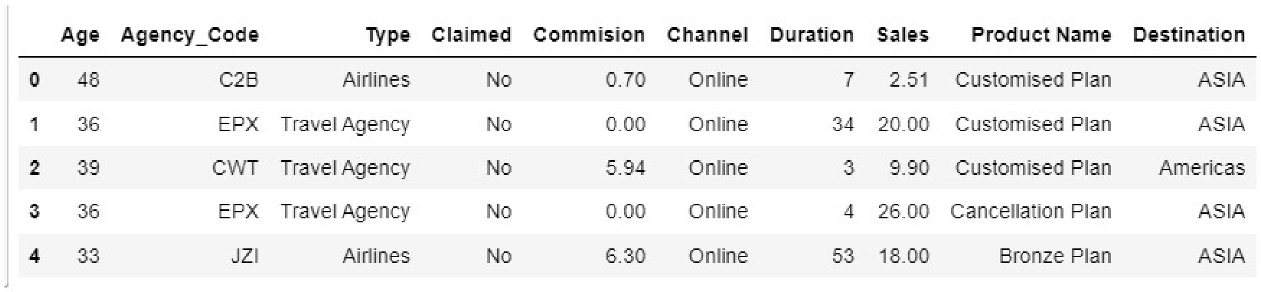
An Insurance firm providing tour insurance is facing higher claim frequency. The management decides to collect data from the past few years. You are assigned the task to make a model which predicts the claim status and provide recommendations to management. Use CART, RF & ANN and compare the models'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

**Attribute Information:**

- Target: Claim Status (Claimed)
- Code of tour firm (Agency_Code)
- Type of tour insurance firms (Type)
- Distribution channel of tour insurance agencies (Channel)
- Name of the tour insurance products (Product)
- Duration of the tour (Duration)
- Destination of the tour (Destination)
- Amount of sales of tour insurance policies (Sales)
- The commission received for tour insurance firm (Commission)
- 10. Age of insured (Age)

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

After importing all the required libraries and loading the dataset, we can check for the head of the data:



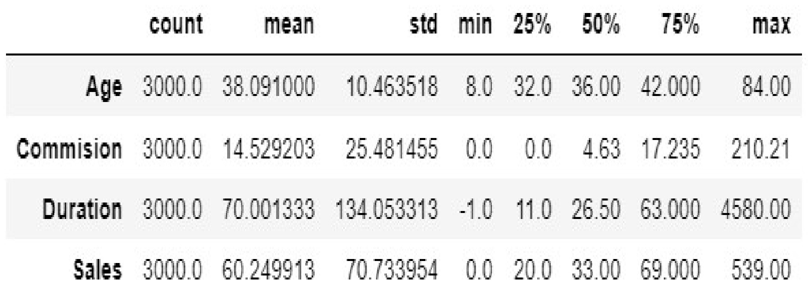
| | Age | Agency_Code | Type | Claimed | Commision | Channel | Duration | Sales | Product Name | Destination |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 48 | C2B | Airlines | No | 0.70 | Online | 7 | 2.51 | Customised Plan | ASIA |
| 1 | 36 | EPX | Travel Agency | No | 0.00 | Online | 34 | 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 |

**Understanding the data:**

Shape of the data:

```
The number of rows: 3000

The number of columns: 10
```

Describing the data:

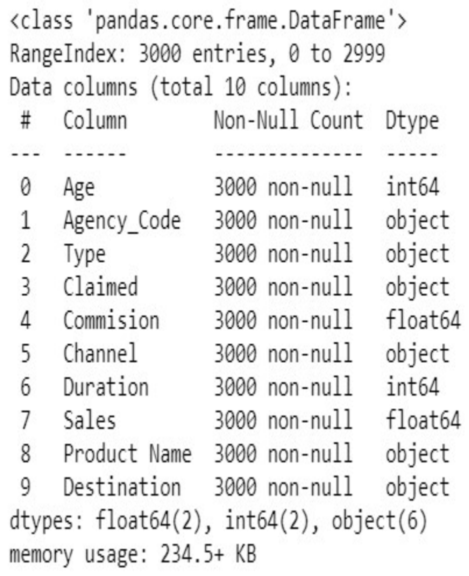


| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Age** | 3000.0 | 38.091000 | 10.463518 | 8.0 | 32.0 | 36.00 | 42.000 | 84.00 |
| **Commision** | 3000.0 | 14.529203 | 25.481455 | 0.0 | 0.0 | 4.63 | 17.235 | 210.21 |
| **Duration** | 3000.0 | 70.001333 |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

From the above summary of the dataset, we can observe few details about the data like,

1. Maximum age of the insured is 84 years and minimum age is 8 years.
2. Mean age is around 38.
3. One odd thing we can observe here is that the minimum duration of the tour is -1., but duration can't be negative, so this means it is an outlier, and also maximum duration is 4580, that can also be an outlier in our dataset, that we will treat later.
4. Maximum commission taken by agency is 210.21 and the minimum is 0.

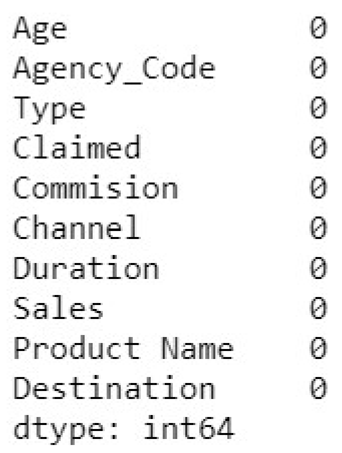
Info about the data:



```
<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
```

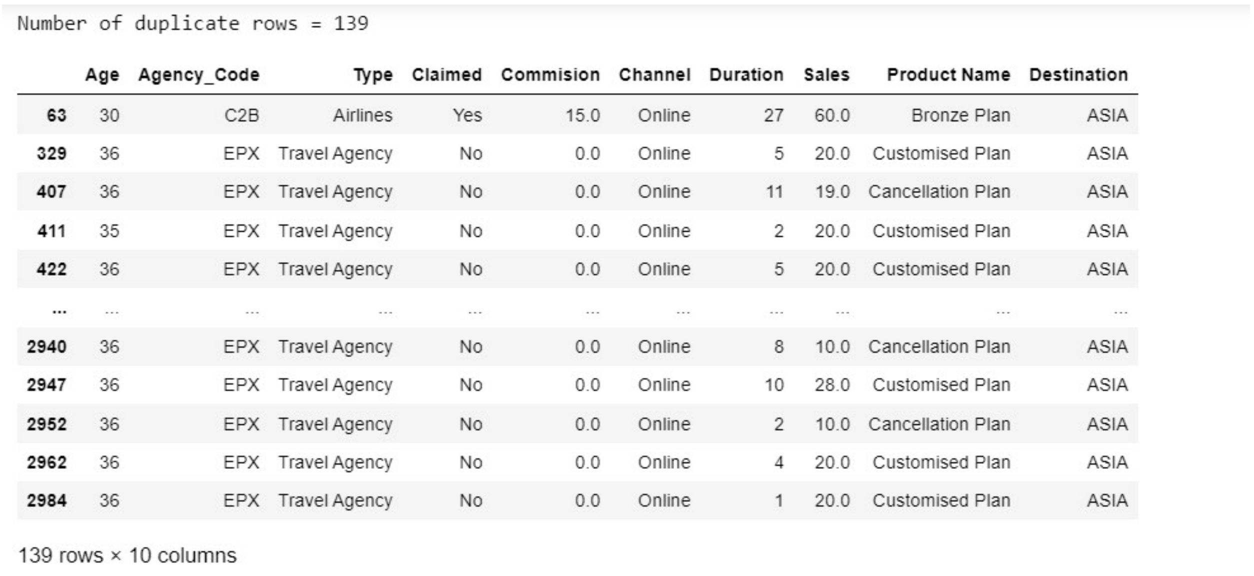
In the dataset, we can see from the info that there are 2 variables with datatype, 2 with integer datatype, 6 with object or string datatype. There are no null values present.

Checking Null/missing values:



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

Number of Duplicate Rows:



```
Number of duplicate rows = 139
```

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

139 rows × 10 columns

Reason for not removing the duplicates:

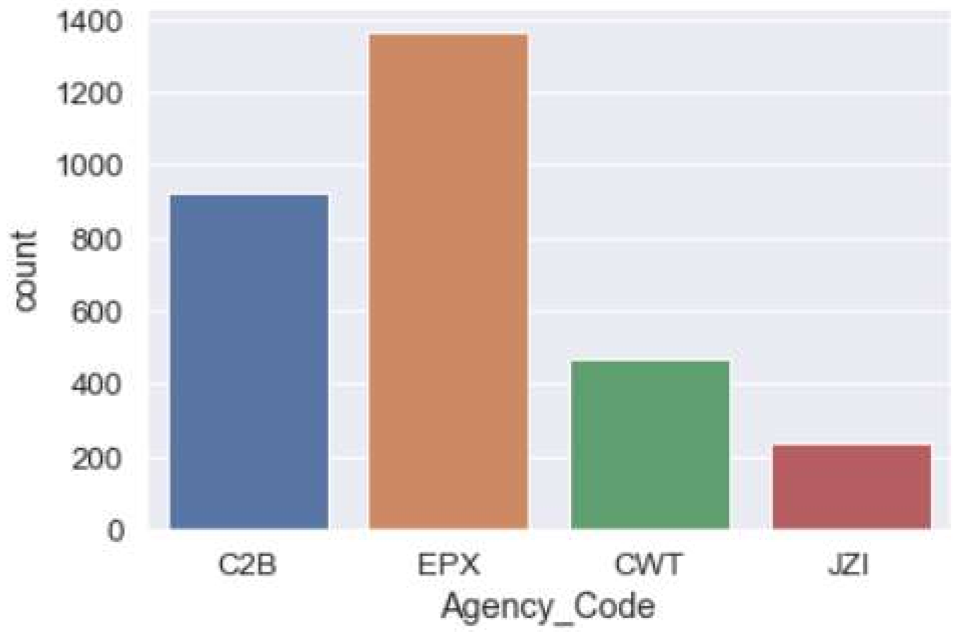
As we can see there are 139 duplicate records are present in the dataset.

But no need to remove those records because as we can see that there is difference in the values of few variables in every record, not everything is same, for example; there is difference in person's age or the duration of the tour, or tour insurance firms are different or the difference in product name that one has chosen.

So we are not removing the duplicates from our dataset.

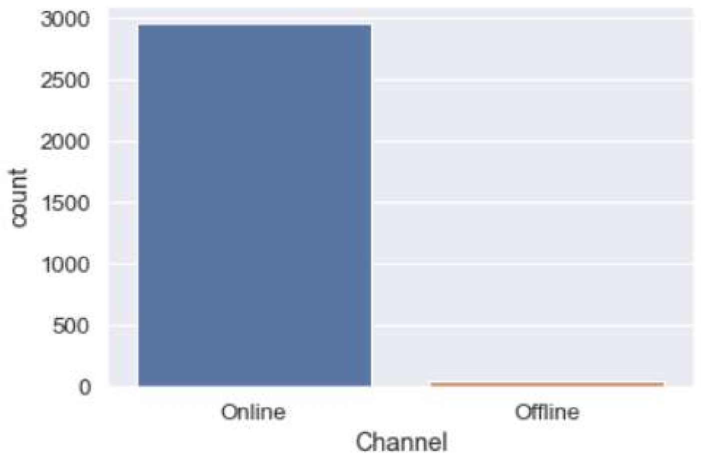
**Few Countplots for checking counts:**

Countplot for ‘Agency Code’:



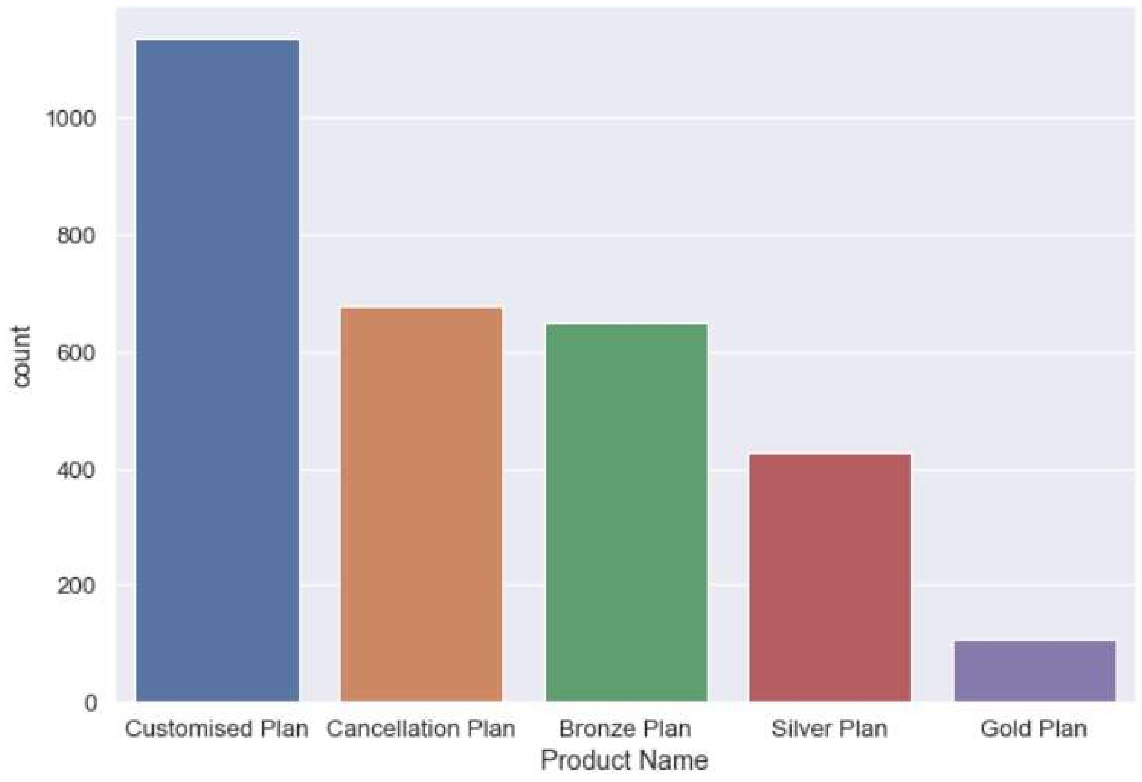
EPX (Agency_Code) has highest number of customers, and JZI has the lowest number of customers.

Countplot for ‘Channel’:



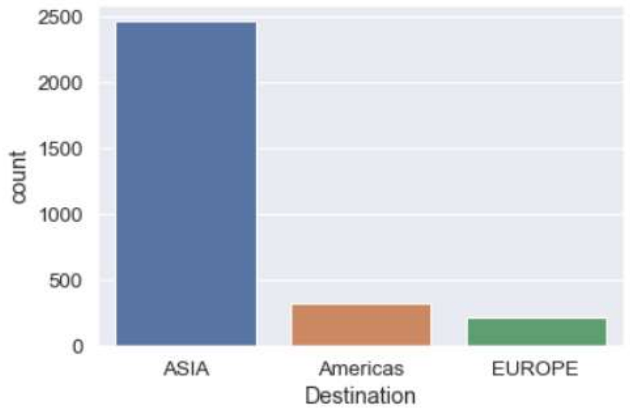
We can see that the majority of the customers used 'Online' Channel for the insurance services.

Countplot for Product Name:



As per the above shown figure, we can observe that most of the customers want to customize their plan as per their requirement and so the customers who opt for the 'Customized Plan' are more than 1000.

Countplot for ‘Destination:



Majority of the customers choose 'ASIA' as their preferred Destination.

**Crosstab Validations:**

```
Count of different type of tour insurance firms:
 Travel Agency    1837
Airlines         1163
Name: Type, dtype: int64
```

| Claimed | No | Yes |
|---|---|---|
| **Type** | | |
| **Airlines** | 573 | 590 |
| **Travel Agency** | 1503 | 334 |

In Airlines tour insurance firm, out of total 1163 people, 590 people has claimed and 573 did not claimed their insurance.

In Travel Agency, out of total 1837 people, only 334 people has taken the claim and 1503 did not go for it.

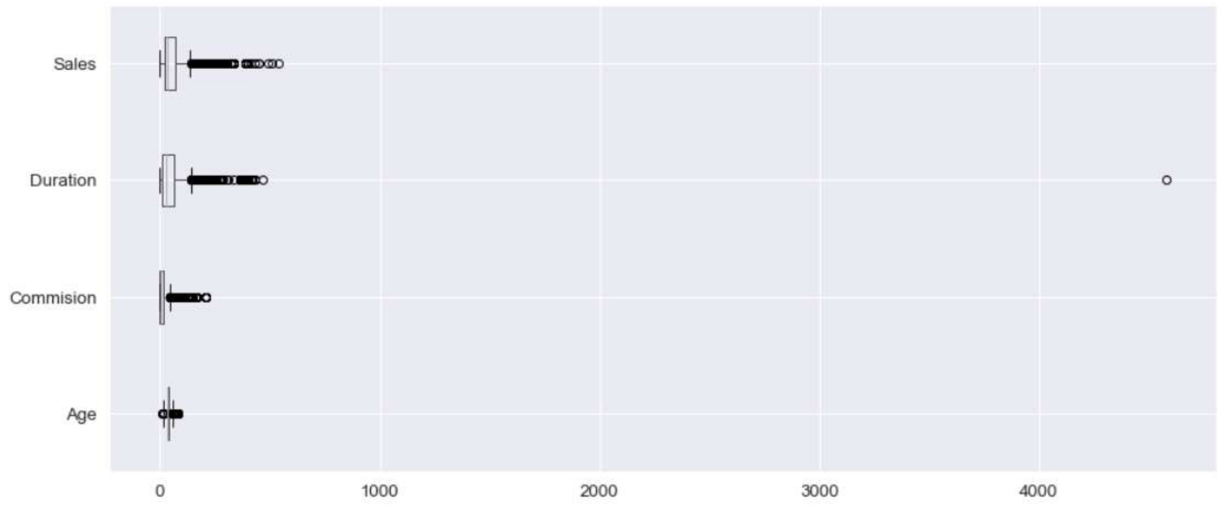
```
Count of different Distribution channel of tour insurance agencies:
 Online     2954
Offline      46
Name: Channel, dtype: int64
```

| Claimed | No | Yes |
|---|---|---|
| **Channel** | | |
| **Offline** | 29 | 17 |
| **Online** | 2047 | 907 |

Through online channel, out of total 2954 members, only 907 claimed their insurance and 2047 did not go for it.

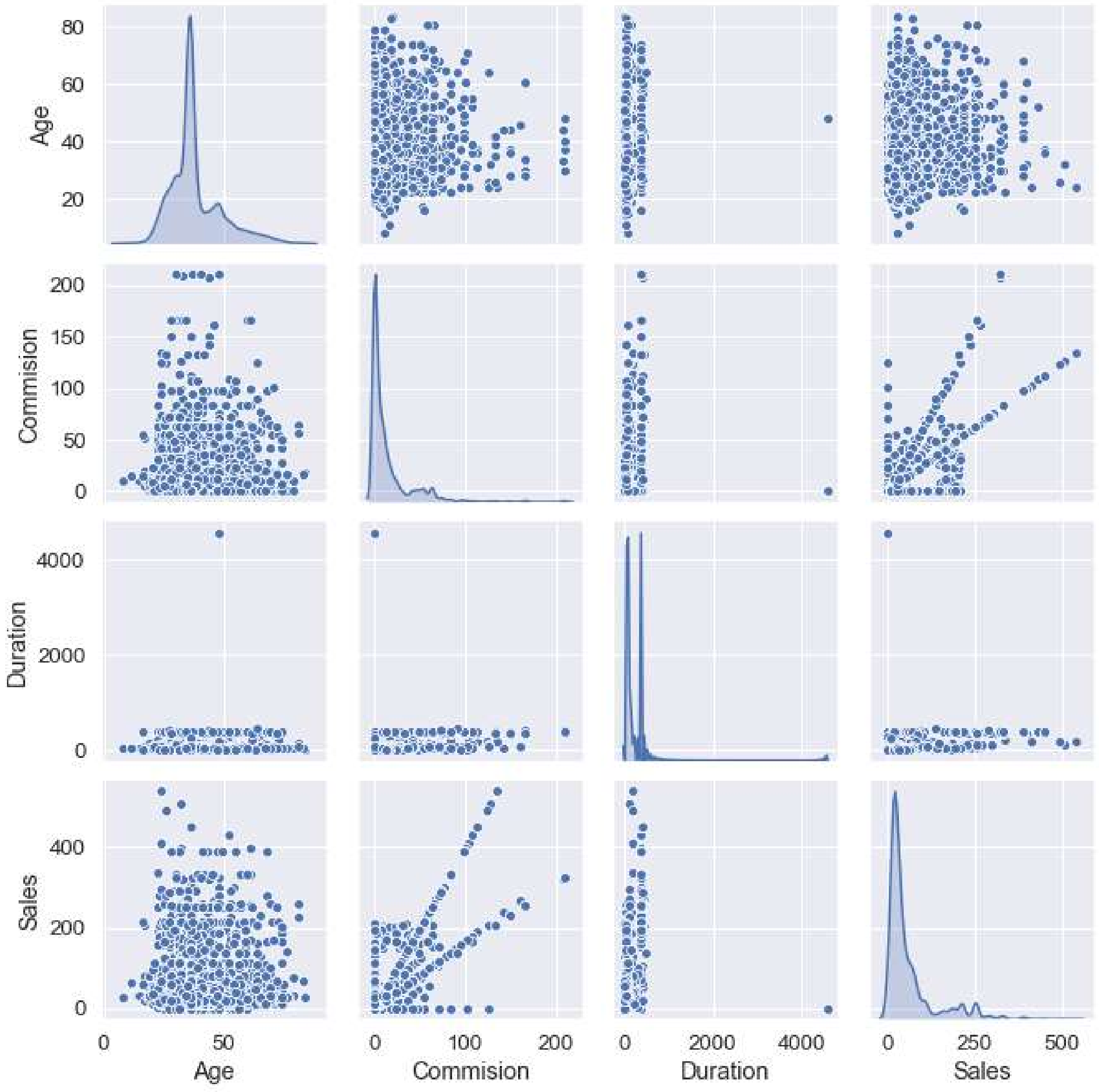
Through offline channel, out of total 46, only 17 claimed and 29 did not claimed.

**Checking for Outliers for Continuous variables using Boxplot:**



As we can see that there are outliers present in all 4 features of our data. The duration of 4000 days affecting the boxplot most, due to this outlier the boxplot has stretched till 4000 point.

**Displaying Pairplot to see pair wise distribution of continuous variables**:



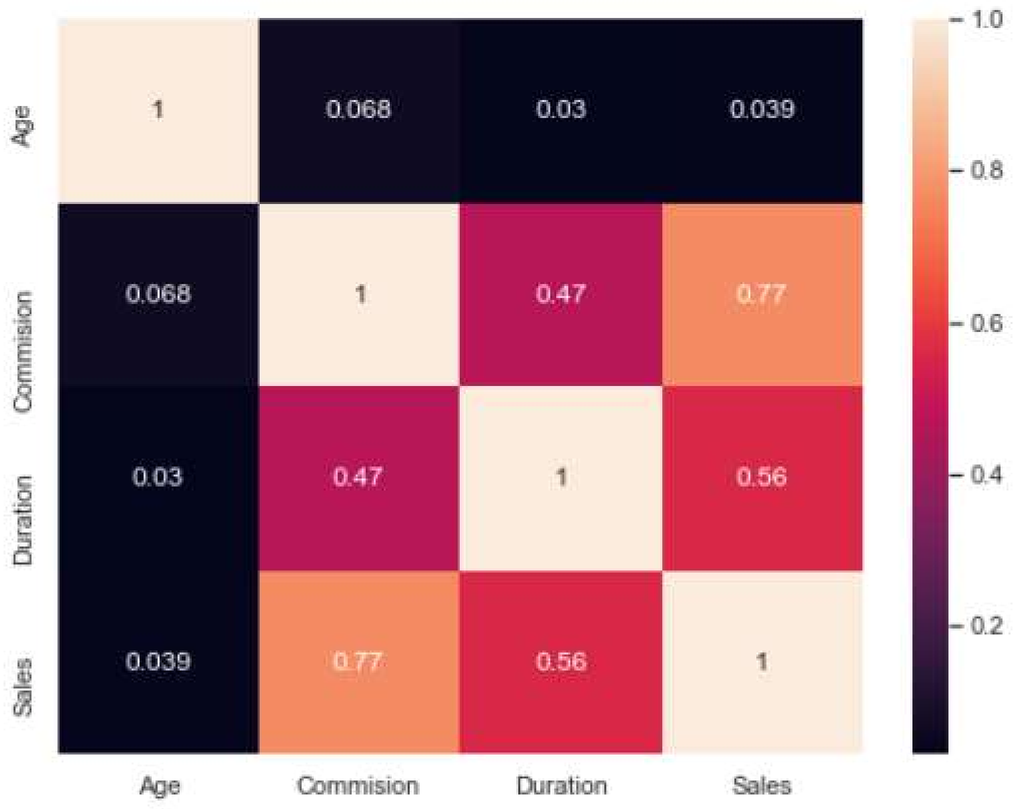
**Observations:**

Most of the sales of insurance happened for the age from 20 to 60.

Most of people opted for insurance for the less number of days of trip duration.

Commission has earned from most of the customers ranges between 0 to 100 rupees.

**Displaying Correlations with Heatmap for only continuous variables:**



**Observation:**

There is strong correlation between Sales and Commission.

There is very weak correlation between Age and Duration.

There is moderate correlation between Duration and Sales.

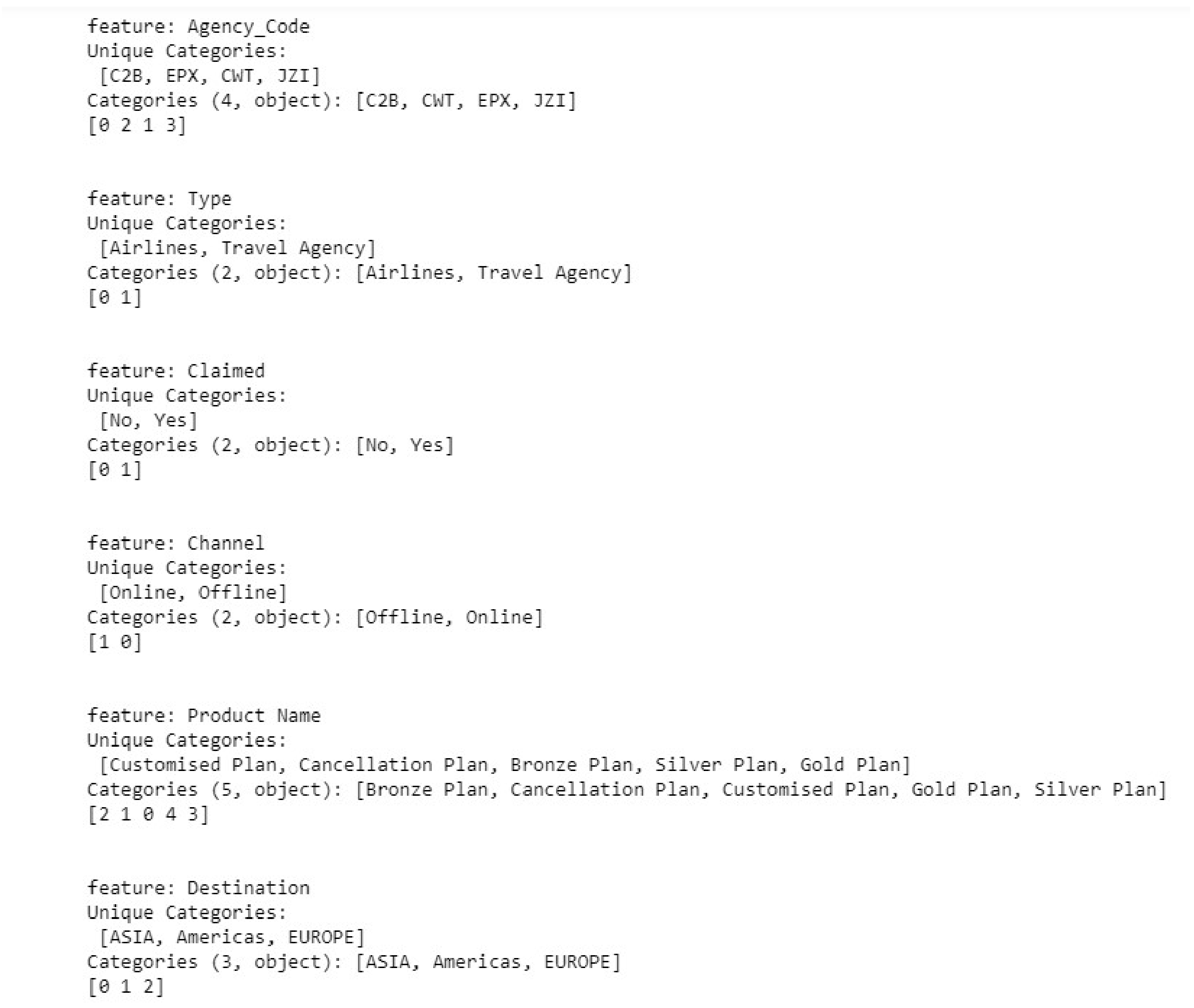
Duration and Commission has the moderate correlation between them.

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

Decision tree Model in Python can take only numerical (categorical) columns.

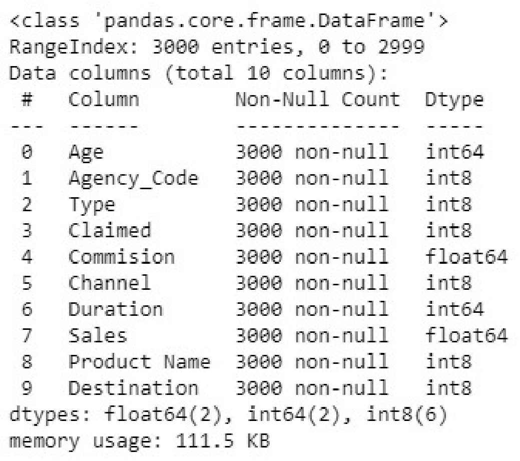
It cannot take string (object) datatypes.

So, we are converting object datatypes into categorical with each distinct value becoming a category or code.



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


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


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


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


feature: Product Name
Unique Categories:
 [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
Unique Categories:
 [ASIA, Americas, EUROPE]
Categories (3, object): [ASIA, Americas, EUROPE]
[0 1 2]
```

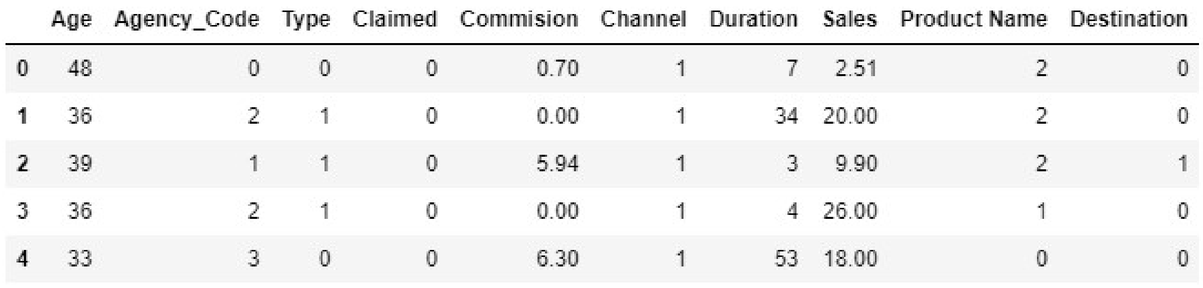
<u>After changing the datatype, checking for info again:</u>



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Age           3000 non-null   int64  
 1   Agency_Code   3000 non-null   int8   
 2   Type          3000 non-null   int8   
 3   Claimed       3000 non-null   int8   
 4   Commision     3000 non-null   float64
 5   Channel       3000 non-null   int8   
 6   Duration      3000 non-null   int64  
 7   Sales         3000 non-null   float64
 8   Product Name  3000 non-null   int8   
 9   Destination   3000 non-null   int8   
dtypes: float64(2), int64(2), int8(6)
memory usage: 111.5 KB
```

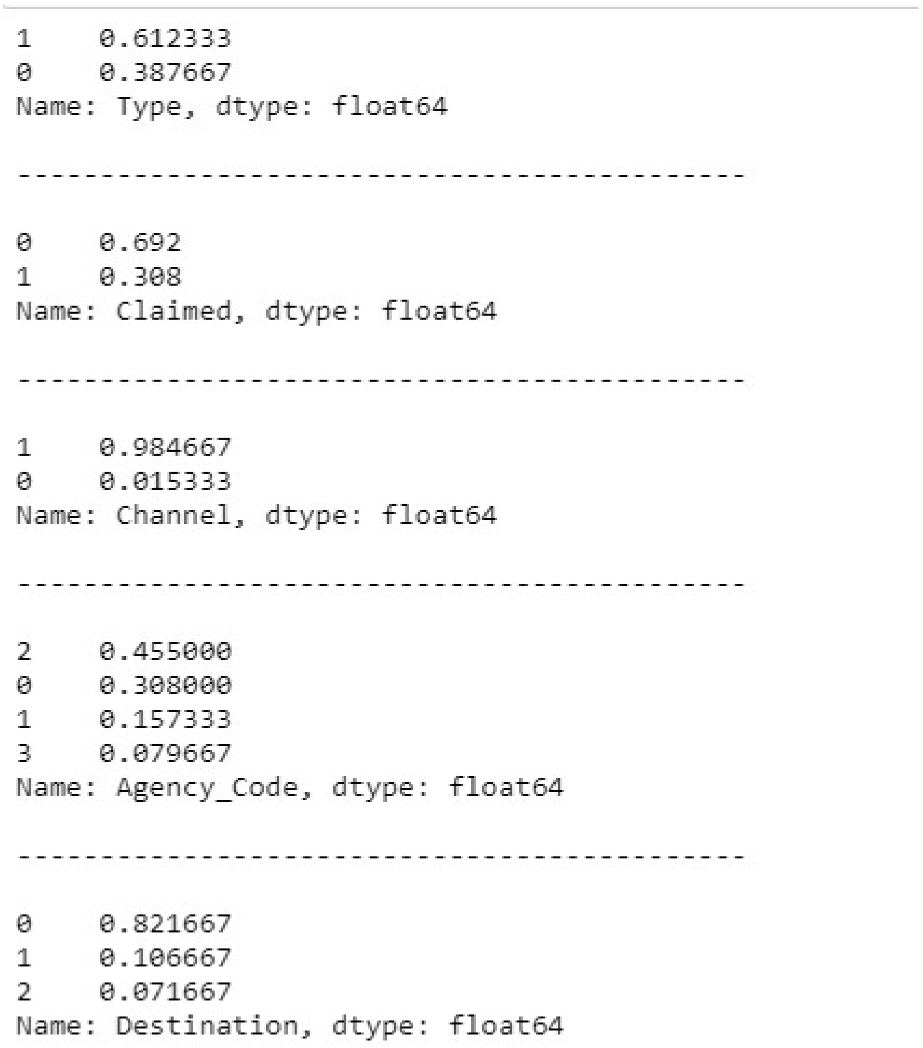
We can see that all the object data types variables are now changed into categorical values.

Checking the Head again:



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

As per the codes, checking the Proportion of 1s and 0s for every feature:



```
1    0.612333
0    0.387667
Name: Type, dtype: float64

--------------------------------------------

0    0.692
1    0.308
Name: Claimed, dtype: float64

--------------------------------------------

1    0.984667
0    0.015333
Name: Channel, dtype: float64

--------------------------------------------

2    0.455000
0    0.308000
1    0.157333
3    0.079667
Name: Agency_Code, dtype: float64

--------------------------------------------

0    0.821667
1    0.106667
2    0.071667
Name: Destination, dtype: float64
```

Checking the value counts of ‘Claimed’ individuals:

```
0    2076
1     924
Name: Claimed, dtype: int64
```

Here, 1 indicates the number of customers who claimed the insurance, and 0 indicates the number Of customer who did not claimed their insurance.

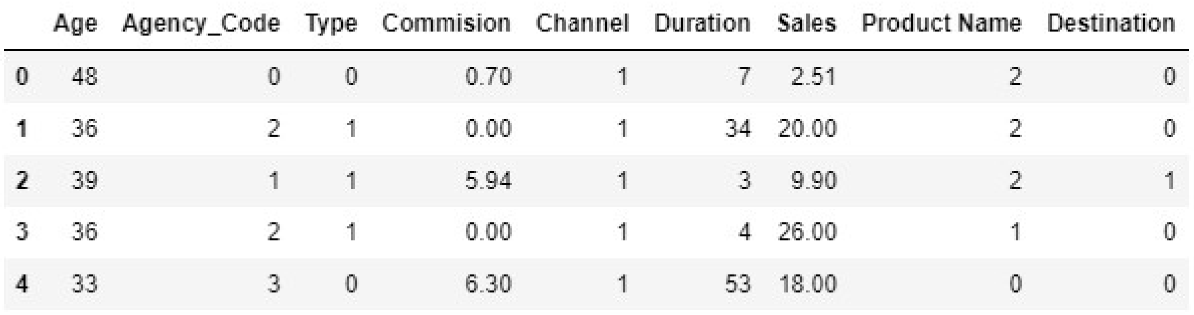
## Splitting the data into training and test set

First step is to separate the dependent variable (target column/variable) and independent variable. Target column is 'Claimed' for this data.

After separated the ‘Claimed’ column from the dataset and assigned the dataset into the new variable named ‘X’ with all the independent columns, we can now check the head again to confirm the separation.

Assigned the dependent column (Claimed) to another variable named ‘y’:

Head of the independent data:



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

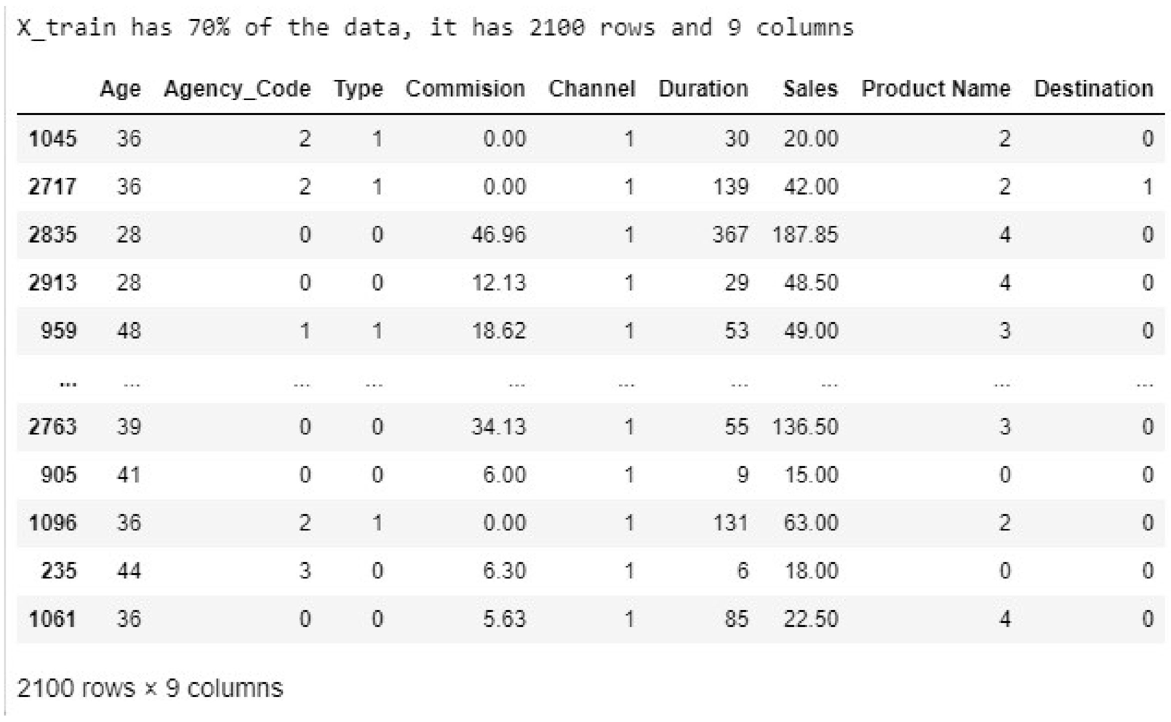
Now, we need to import *train_test_split* package from *sklearn.model_selection* library to split the train and test data for further calculation.

Separating data into training and testing sets is an important part of evaluating data mining models. By using similar data for training and testing, you can minimize the effects of data discrepancies and better understand the characteristics of the model.

The purpose of both the datasets is that the “training” data set is the general term for the samples used to create the model, while the “test” or “validation” data set is used to qualify performance. Perhaps traditionally the dataset used to evaluate the final model performance is called the “test set”.

We are splitting train set and test set to 70% and 30% of the data respectively.

Checking the training dataset X_train:

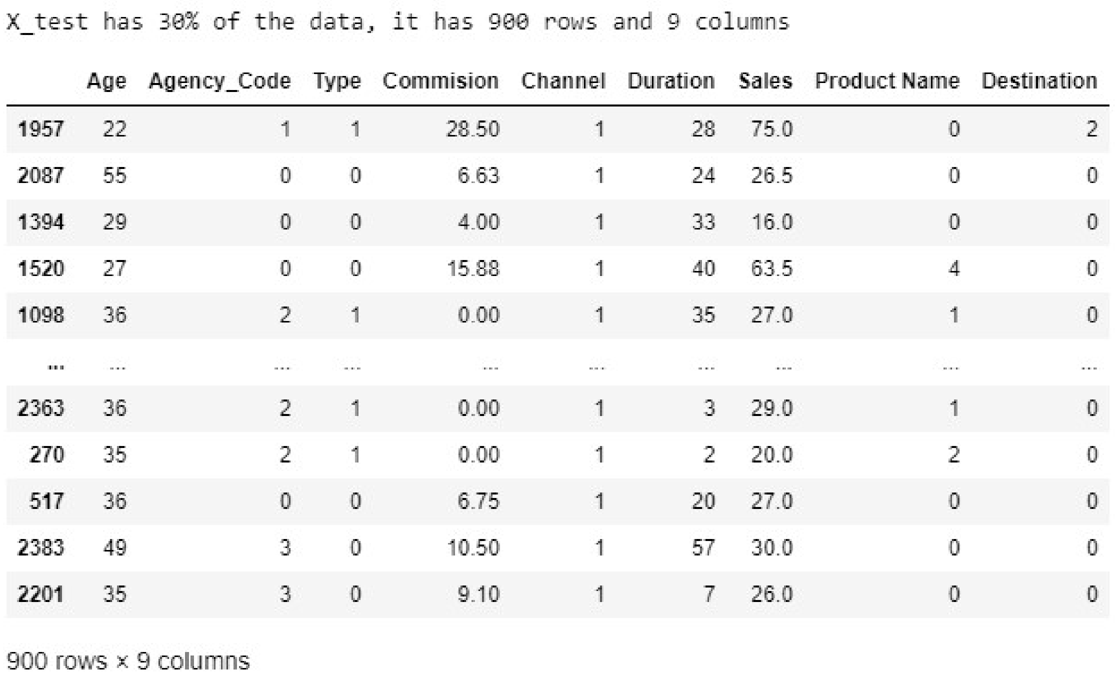


```
X_train has 70% of the data, it has 2100 rows and 9 columns
```

| | Age | Agency_Code | Type | Commision | Channel | Duration | Sales | Product Name | Destination |
|---|---|---|---|---|---|---|---|---|---|
| **1045** | 36 | 2 | 1 | 0.00 | 1 | 30 | 20.00 | 2 | 0 |
| **2717** | 36 | 2 | 1 | 0.00 | 1 | 139 | 42.00 | 2 | 1 |
| **2835** | 28 | 0 | 0 | 46.96 | 1 | 367 | 187.85 | 4 | 0 |
| **2913** | 28 | 0 | 0 | 12.13 | 1 | 29 | 48.50 | 4 | 0 |
| **959** | 48 | 1 | 1 | 18.62 | 1 | 53 | 49.00 | 3 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2763** | 39 | 0 | 0 | 34.13 | 1 | 55 | 136.50 | 3 | 0 |
| **905** | 41 | 0 | 0 | 6.00 | 1 | 9 | 15.00 | 0 | 0 |
| **1096** | 36 | 2 | 1 | 0.00 | 1 | 131 | 63.00 | 2 | 0 |
| **235** | 44 | 3 | 0 | 6.30 | 1 | 6 | 18.00 | 0 | 0 |
| **1061** | 36 | 0 | 0 | 5.63 | 1 | 85 | 22.50 | 4 | 0 |

2100 rows × 9 columns

Checking the testing dataset X_test:



```
X_test has 30% of the data, it has 900 rows and 9 columns
```

| | Age | Agency_Code | Type | Commision | Channel | Duration | Sales | Product Name | Destination |
|---|---|---|---|---|---|---|---|---|---|
| **1957** | 22 | 1 | 1 | 28.50 | 1 | 28 | 75.0 | 0 | 2 |
| **2087** | 55 | 0 | 0 | 6.63 | 1 | 24 | 26.5 | 0 | 0 |
| **1394** | 29 | 0 | 0 | 4.00 | 1 | 33 | 16.0 | 0 | 0 |
| **1520** | 27 | 0 | 0 | 15.88 | 1 | 40 | 63.5 | 4 | 0 |
| **1098** | 36 | 2 | 1 | 0.00 | 1 | 35 | 27.0 | 1 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2363** | 36 | 2 | 1 | 0.00 | 1 | 3 | 29.0 | 1 | 0 |
| **270** | 35 | 2 | 1 | 0.00 | 1 | 2 | 20.0 | 2 | 0 |
| **517** | 36 | 0 | 0 | 6.75 | 1 | 20 | 27.0 | 0 | 0 |
| **2383** | 49 | 3 | 0 | 10.50 | 1 | 57 | 30.0 | 0 | 0 |
| **2201** | 35 | 3 | 0 | 9.10 | 1 | 7 | 26.0 | 0 | 0 |

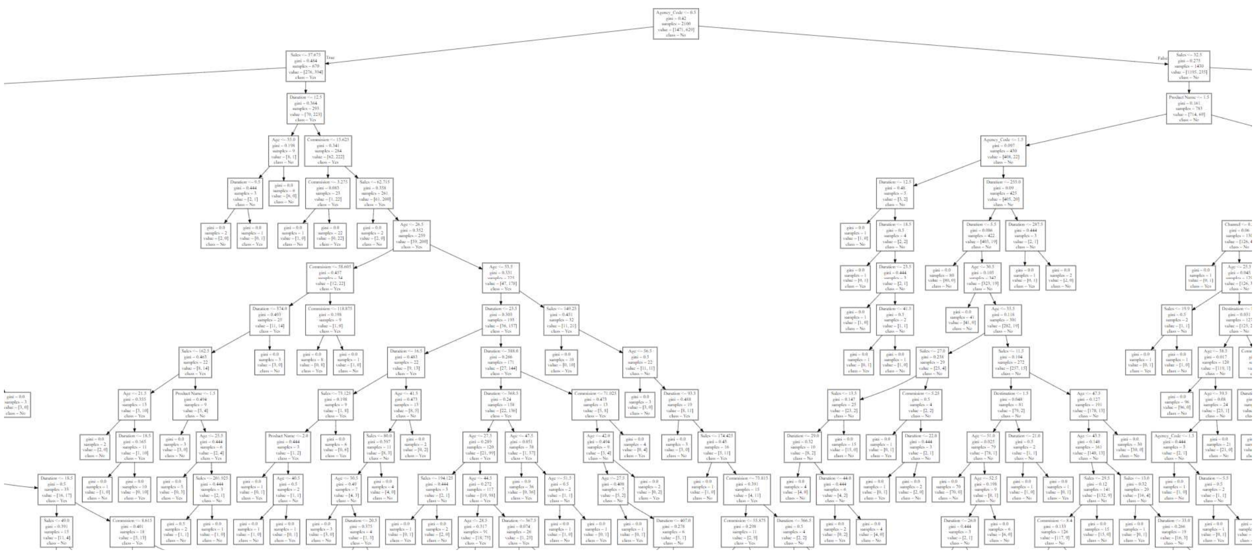
900 rows × 9 columns

After splitting the data, now we will build a Decision Tree Classifier using 'Gini' criterion and also we will apply grid search for each model and make the models on best_params.

By fitting the X_train and the train_labels into the dt_model we can generate Decision Tree. To see or to visualize the tree, first we need to export the tree in a dot file and then need to copy that exported data and paste into webgraphviz. (www.webgraphviz.com).

*Webgraphviz* is actually a Graphviz in a browser. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks.

After checking the tree on the webgraphviz, we can see that the graph is overfitting, so we are defining some parameters using param_grid and using GridSearchCV to loop through predefined hyper parameters, and rebuilding the tree.



The above tree is too big to display.

It is overfitting model.

## <u>**Pruning needs to be done to avoid overgrowing of sub-trees/branches:**</u>

In pruning we will set some parameters to the tree and see how it will work, like min_samples_split which specifies the minimum number of samples required to split an internal node, and min_samples_leaf which specifies the minimum number of samples required to be at a leaf node.

GridSearchCV is a function that helps to loop through predefined hyper parameters, and to fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyper parameters.

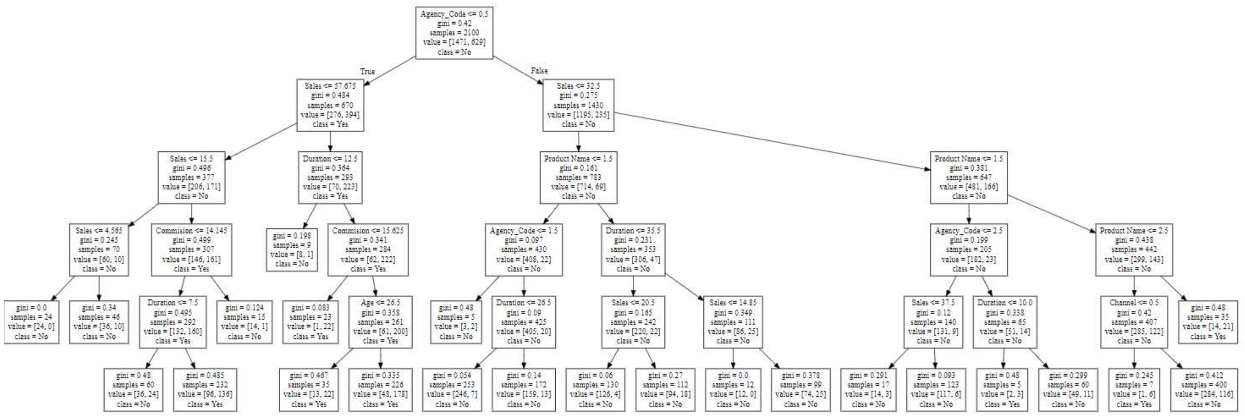
CV: (integer value). It determines the cross-validation splitting strategy.

Fitting the X_train and train _labels into the grid_search and getting the best estimator using grid_search.best_estimator. We will make the tree again using the best grid parameters

DecisionTreeClassifier (max_depth=5, min_samples_leaf=5, min_samples_split=60, random_state=1)

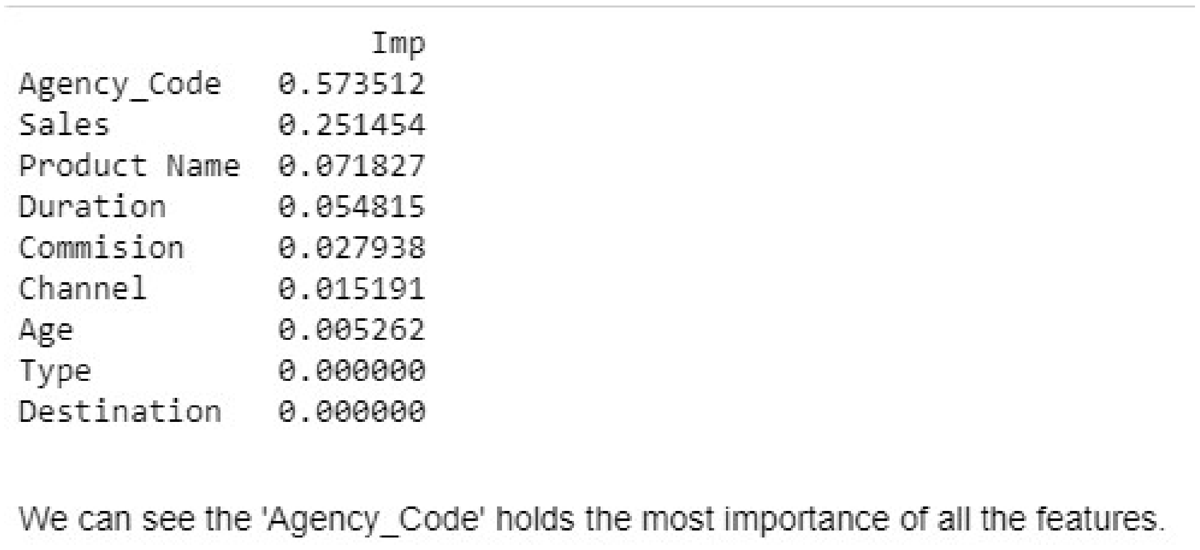
We will again creating a new dot file and exporting the tree into the file and again open the

Webgraphviz:


Agency_Code <= 0.5 gini = 0.42 samples = 2100 value = [1471, 629] class = No
True
False
Sales <= 57.675 gini = 0.484 samples = 670 value = [276, 394] class = Yes
Sales <= 32.5 gini = 0.275 samples = 1430 value = [1195, 235] class = No
Sales <= 15.5 gini = 0.496 samples = 377 value = [206, 171] class = No
Duration <= 12.5 gini = 0.364 samples = 293 value = [70, 223] class = Yes
Product Name <= 1.5 gini = 0.161 samples = 783 value = [714, 69] class = No
Product Name <= 1.5 gini = 0.381 samples = 647 value = [481, 166] class = No
Sales <= 4.565 gini = 0.245 samples = 70 value = [60, 10] class = No
Commision <= 14.145 gini = 0.499 samples = 307 value = [146, 161] class = Yes
gini = 0.198 samples = 9 value = [8, 1] class = No
Commision <= 15.625 gini = 0.341 samples = 284 value = [62, 222] class = Yes
Agency_Code <= 1.5 gini = 0.097 samples = 430 value = [408, 22] class = No
Duration <= 35.5 gini = 0.231 samples = 353 value = [306, 47] class = No
Agency_Code <= 2.5 gini = 0.199 samples = 205 value = [182, 23] class = No
Product Name <= 2.5 gini = 0.438 samples = 442 value = [299, 143] class = No
gini = 0.0 samples = 24 value = [24, 0] class = No
gini = 0.34 samples = 46 value = [36, 10] class = No
Duration <= 7.5 gini = 0.495 samples = 292 value = [132, 160] class = Yes
gini = 0.124 samples = 15 value = [14, 1] class = No
gini = 0.083 samples = 23 value = [1, 22] class = Yes
Age <= 26.5 gini = 0.358 samples = 261 value = [61, 200] class = Yes
gini = 0.48 samples = 5 value = [3, 2] class = No
Duration <= 26.5 gini = 0.09 samples = 425 value = [405, 20] class = No
Sales <= 20.5 gini = 0.165 samples = 242 value = [220, 22] class = No
Sales <= 14.85 gini = 0.349 samples = 111 value = [86, 25] class = No
Sales <= 37.5 gini = 0.12 samples = 140 value = [131, 9] class = No
Duration <= 10.0 gini = 0.338 samples = 65 value = [51, 14] class = No
Channel <= 0.5 gini = 0.42 samples = 407 value = [285, 122] class = No
gini = 0.48 samples = 35 value = [14, 21] class = Yes
gini = 0.48 samples = 60 value = [36, 24] class = No
gini = 0.485 samples = 232 value = [96, 136] class = Yes
gini = 0.467 samples = 35 value = [13, 22] class = Yes
gini = 0.335 samples = 226 value = [48, 178] class = Yes
gini = 0.054 samples = 253 value = [246, 7] class = No
gini = 0.14 samples = 172 value = [159, 13] class = No
gini = 0.06 samples = 130 value = [126, 4] class = No
gini = 0.27 samples = 112 value = [94, 18] class = No
gini = 0.0 samples = 12 value = [12, 0] class = No
gini = 0.378 samples = 99 value = [74, 25] class = No
gini = 0.291 samples = 17 value = [14, 3] class = No
gini = 0.093 samples = 123 value = [117, 6] class = No
gini = 0.48 samples = 5 value = [2, 3] class = Yes
gini = 0.299 samples = 60 value = [49, 11] class = No
gini = 0.245 samples = 7 value = [1, 6] class = Yes
gini = 0.412 samples = 400 value = [284, 116] class = No

This is a new CART model.

## Getting the 'feature_importances'



```
                    Imp
Agency_Code    0.573512
Sales          0.251454
Product Name   0.071827
Duration       0.054815
Commision      0.027938
Channel        0.015191
Age            0.005262
Type           0.000000
Destination    0.000000
```

We can see the 'Agency_Code' holds the most importance of all the features.

## Building Random Forest Classifier

It is a Supervised Classification Algorithm, as the name suggests, this algorithm creates the forest with a number of trees in random order. Random forest classifier can handle the missing values. When we have more trees in the forest, random forest classifier won't over fit the model.

It is an ensemble technique wherein we construct multiple models and take the average output of all the models to take final decision/make predictions.

We will be passing some below parameters into the model for the execution:

1. n_estimators is the number of trees, which we want to build within the RandomForestClassifier.
2. Out of bag (OOB) score is a way of validating the Random forest model. OOB score is computed as the number of correctly predicted rows from the out of bag sample.
3. max_features will be used to select randomly features from the available number of features.

Now we will fit the X_train and train_labels into rfcl (RandomForestClassifier).

When we create a bootstrapped dataset, ~1/3 of the original data does not end up in the bootstrapped dataset. This is called out-of-bag dataset. Every bootstrapped dataset has oob data points.

Getting the oob_score: `0.7847619047619048`

Proportion of OOB samples incorrectly classified are out-of-bag error.

Getting the error rate: 1 – (oob_score)

```
error_rate (in %): 0.21523809523809523
```

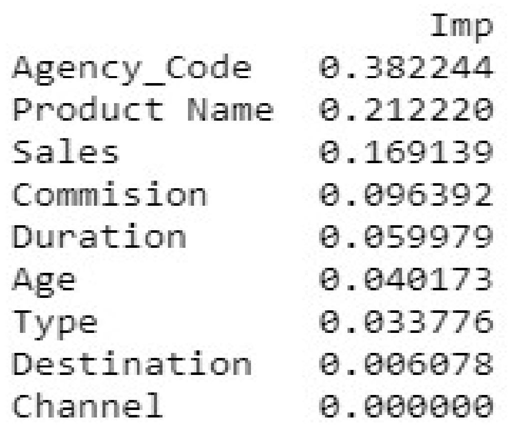
Error rate is 21.5 %

For the above specified grid parameters, we got the rfcl model with oob score 78.4% and error rate 21.5%, which shows the model is good to go for further evaluation but this time again we will set grid parameters with different values and see how the model works and what is the error rate for the new parameters.

**Getting the best parameters for the rfcl:**

```
RandomForestClassifier (max_depth=7, max_features=5, min_samples_leaf=30,
                        min_samples_split=50, n_estimators=401,
                        oob_score=True, random_state=1)
```

**Getting the Feature Importance :**



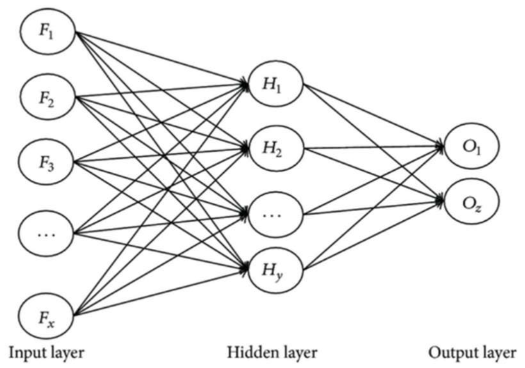
```
                    Imp
Agency_Code    0.382244
Product Name   0.212220
Sales          0.169139
Commision      0.096392
Duration       0.059979
Age            0.040173
Type           0.033776
Destination    0.006078
Channel        0.000000
```

## Building a Neural Network Classifier

Similar to a biological neural network, an artificial neural network has the ability to learn, generalize and adapt.

It is a machine learning algorithm that is roughly modelled around what is currently known about how the human brain functions.

It is made up of 3 layers- input, hidden and output.



There are single layer and multilayer neural networks.

We use neural network algorithm because of 3 primary reasons:

1. Ability to learn.

2. Ability to generalize.

3. Understands and working adaptively.

Passing the best parameters by the param_grid and getting the best grid from grid search CV estimator.

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

### Calculating the performance metrics:

Confusion Matrix for Train and Test Data for Decision Tree Classifier

For Train data:

```
Array ([[1296, 175],
       [241, 388]], dtype=int64)
```

For Test data:

```
Array ([[544, 61],
       [142, 153]], dtype=int64)
```

Confusion Matrix for Train and Test Data for Random Forest Classifier

For Train data:

```
Array ([[1331, 140],
       [270, 359]], dtype=int64)
```

For Test data:

```
Array ([[552, 53],
       [156, 139]], dtype=int64)
```

Confusion Matrix for Train and Test Data for Neural Network Classifier

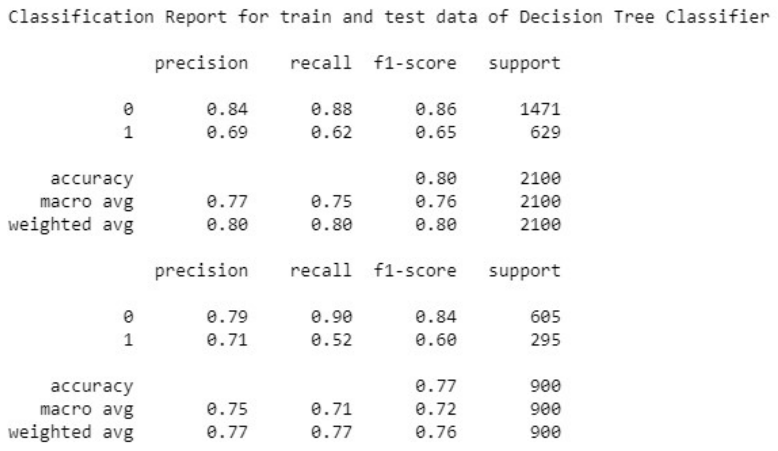
For Train data:

```
Array ([[1350, 121],
       [368, 261]], dtype=int64)
```

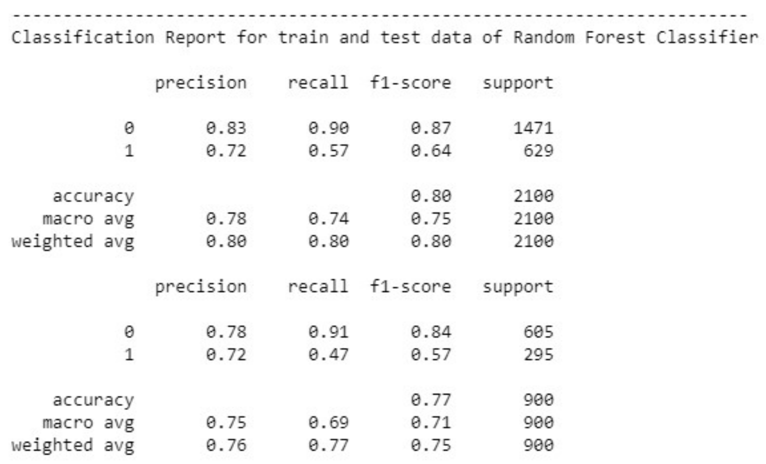
For Test data:

```
Array ([[570, 35],
       [194, 101]], dtype=int64)
```

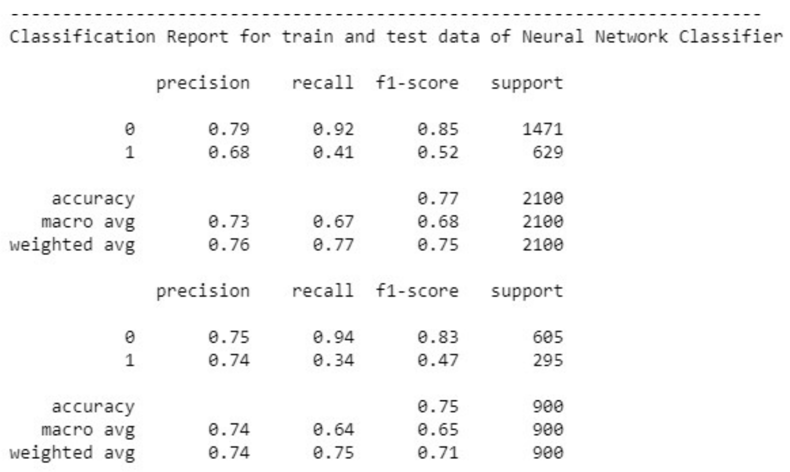
## Classification Reports for all 3 Classifiers:



```
Classification Report for train and test data of Decision Tree Classifier

              precision    recall  f1-score   support

           0       0.84      0.88      0.86      1471
           1       0.69      0.62      0.65       629

    accuracy                           0.80      2100
   macro avg       0.77      0.75      0.76      2100
weighted avg       0.80      0.80      0.80      2100

              precision    recall  f1-score   support

           0       0.79      0.90      0.84       605
           1       0.71      0.52      0.60       295

    accuracy                           0.77       900
   macro avg       0.75      0.71      0.72       900
weighted avg       0.77      0.77      0.76       900
```



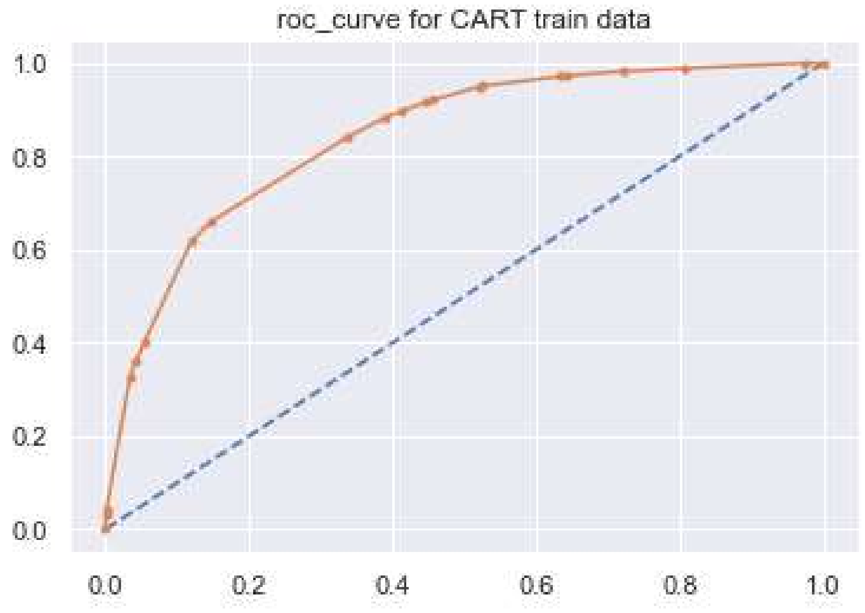
```
 -----------------------------------------------------------------------
 Classification Report for train and test data of Random Forest Classifier

              precision    recall  f1-score   support

           0       0.83      0.90      0.87      1471
           1       0.72      0.57      0.64       629

    accuracy                           0.80      2100
   macro avg       0.78      0.74      0.75      2100
weighted avg       0.80      0.80      0.80      2100

              precision    recall  f1-score   support

           0       0.78      0.91      0.84       605
           1       0.72      0.47      0.57       295

    accuracy                           0.77       900
   macro avg       0.75      0.69      0.71       900
weighted avg       0.76      0.77      0.75       900
```



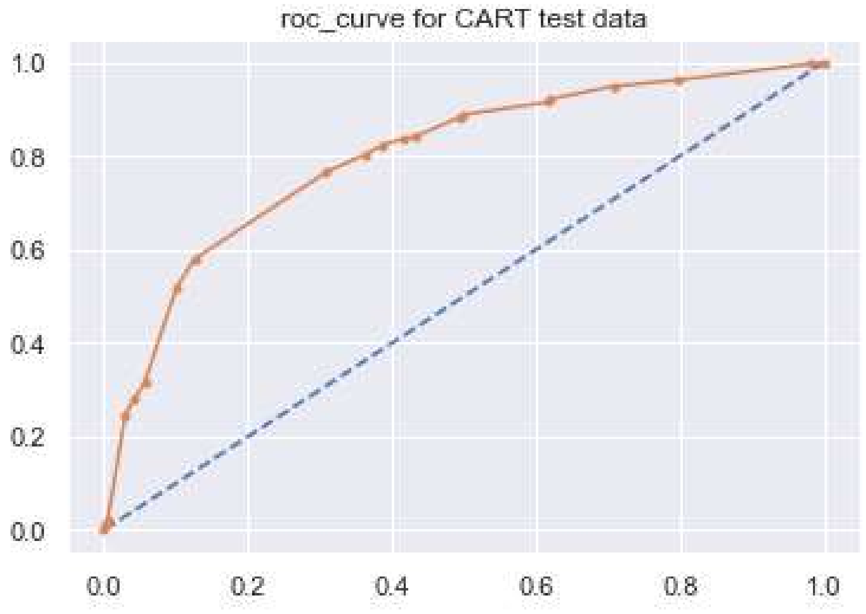
```
 -----------------------------------------------------------------------
 Classification Report for train and test data of Neural Network Classifier

              precision    recall  f1-score   support

           0       0.79      0.92      0.85      1471
           1       0.68      0.41      0.52       629

    accuracy                           0.77      2100
   macro avg       0.73      0.67      0.68      2100
weighted avg       0.76      0.77      0.75      2100

              precision    recall  f1-score   support

           0       0.75      0.94      0.83       605
           1       0.74      0.34      0.47       295

    accuracy                           0.75       900
   macro avg       0.74      0.64      0.65       900
weighted avg       0.74      0.75      0.71       900
```

**ROC_AUC_SCORE and ROC_CURVE for CART**

```
roc_auc_score for train data for CART:  0.8434233009351976
```

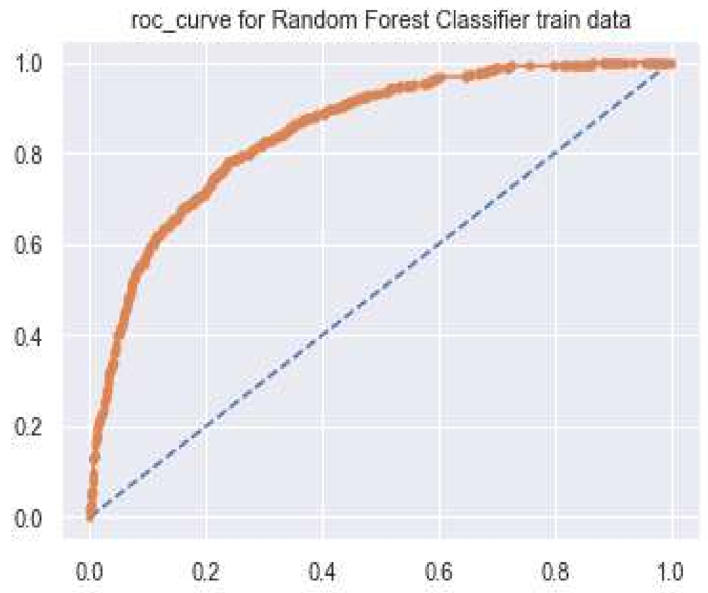


```
-------------------------------------------
roc_auc_score for test data for CART:  0.7995293458467572
```

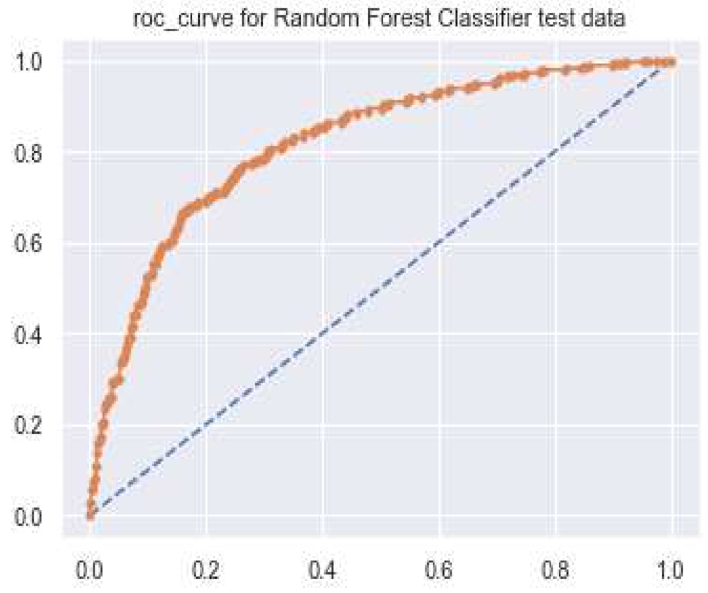


## ROC_AUC_SCORE and ROC_CURVE for RANDOM FOREST Classifier

```
roc_auc_score for train data for Random Forest Classifier:  0.8434233009351976
```

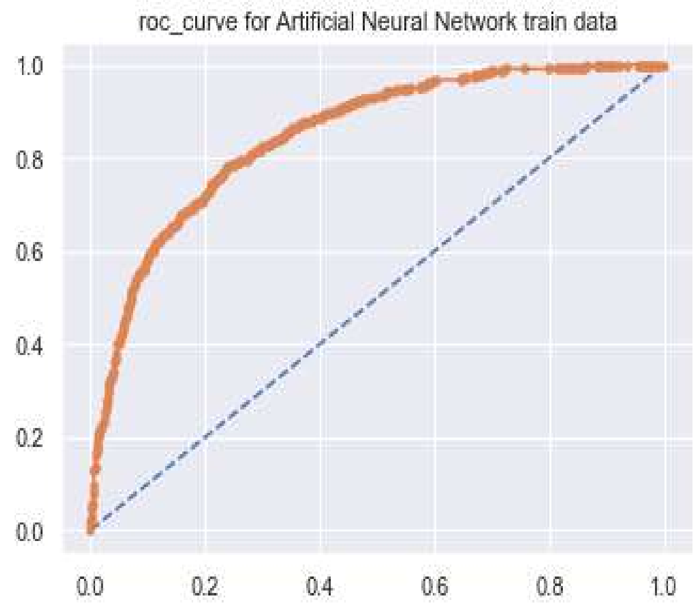


```
----------------------------------------------------
roc_auc_score for test data for Random Forest Classifier:  0.8200112060512676
```

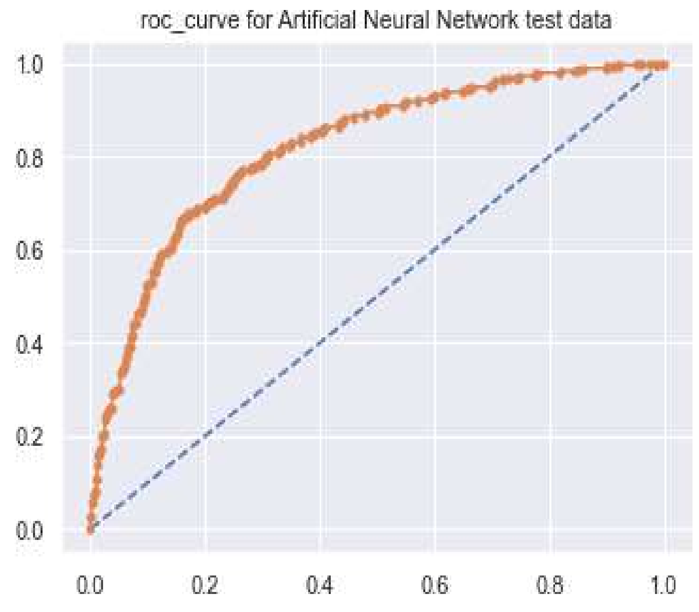


**ROC_AUC_SCORE and ROC_CURVE for Artificial Neural Network**

```
roc_auc_score for train data for Artificial Neural Network:  0.8502224782466314
```



```
---------------------------------------------
roc_auc_score for test data for Artificial Neural Network:  0.8200112060512676
```



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

### Comparison of AUC, Accuracy, Precision, Recall and F1 Score between all 3 models:

As we can see from the above calculations and classification report, below are some values that we got:

AUC, Accuracy, Precision, Recall and F1 Score for CART

**#train data**

cart_train_acc=0.80

cart_train_recall=0.62

cart_train_precision=0.69

cart_train_f1=0.65

**#test data**

cart_test_acc=0.77

cart_test_recall=0.52

cart_test_precision=0.71

cart_test_f1=0.60

AUC, Accuracy, Precision, Recall and F1 Score for RFCL

**#train data**

rf_train_acc=0.80

rf_train_recall=0.57

rf_train_precision=0.72

rf_train_f1=0.64

**#test data**

rf_test_acc=0.77

rf_test_recall=0.47

rf_test_precision=0.72

rf_test_f1=0.57

AUC, Accuracy, Precision, Recall and F1 Score for ANN

**#train data**

neural_train_acc=0.77

neural_train_recall=0.41

neural_train_precision=0.68

neural_train_f1=0.52

**#test data**

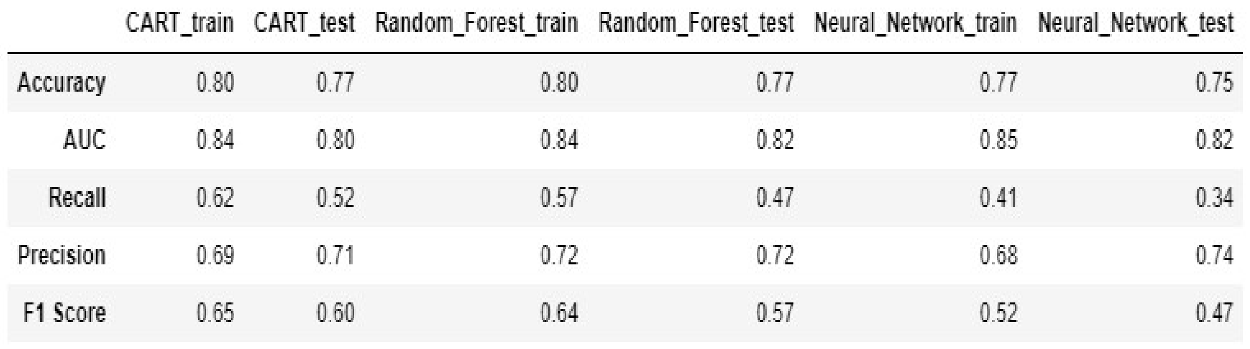
neural_test_acc=0.75

neural_test_recall=0.34

neural_test_precision=0.74

neural_test_f1=0.47

## Table for Comparison of all values:



| | CART_train | CART_test | Random_Forest_train | Random_Forest_test | Neural_Network_train | Neural_Network_test |
|---|---|---|---|---|---|---|
| Accuracy | 0.80 | 0.77 | 0.80 | 0.77 | 0.77 | 0.75 |
| AUC | 0.84 | 0.80 | 0.84 | 0.82 | 0.85 | 0.82 |
| Recall | 0.62 | 0.52 | 0.57 | 0.47 | 0.41 | 0.34 |
| Precision | 0.69 | 0.71 | 0.72 | 0.72 | 0.68 | 0.74 |
| F1 Score | 0.65 | 0.60 | 0.64 | 0.57 | 0.52 | 0.47 |

## Final Model:

I prefer to go with Random Forest Classifier Model because, based on above performance metrics, we can see that the accuracy, f1 score and AUC score is better than the CART and ANN based models. Due to high accuracy and better AUC score, we can select RF model confidently.

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

**Recommendations:**

1. As we can see that majority of the insurance is done through 'Online' channel because it is easy to get the benefits online instead of choosing Offline channel. Also, it is easy to process claim online, so, due to these reasons, there is increase in the claim process.
2. I have also observed that JZI agency has lesser number of sales, they need to boost up their sales by applying some kind of promotional strategies to attract the customers. They can also take help and support from other successful agencies or resources.
3. JZI agency need to add more number of working and active resources to increase their sales.
4. Agencies have more number of sales than Airlines, but number of insured is more by Airlines than agencies.
5. One thing we have observed in the dataset is, people opted for customized plan more than any other plans offered by agencies and airlines. By looking at this situation, I recommend that agencies should gather more information from the customers who took customized plan and ask them what they liked most and in which of the situations they felt to take the insurance plan.
6. If customers are buying insurance offline, we must make sure that the process is simple and smooth so that in future, the customers who claimed their insurance suggest others to take insurance from us. In this way we are not only making our active customers happy but also doubling our sales.
7. We can add-up few more extra risk covers which does happen generally with all kind of people, like loss of personal objects, damage due to the services provided while travelling.
8. We can opt for traditional ways to boost up the sales and also for making the process safe and secure.