Data Mining Project

Clustering, K-Means, CART, Random Forest & ANN

5/31/2020

Case Study 1

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

Data Set: bank_marketing_part1_Data.csv

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

Import all the libraries such as Numpy, Pandas, Matplotlibs that would help us in reading the dataset.

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
%matplotlib inline
```

Upload the dataset to jupyter Notebook:

```
bank=pd.read_csv("bank_marketing_part1_Data.csv")
```

Check the head and the tail of the data, to make sure all the data set is loaded correctly

```
bank.head(5)
```

	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

```
bank.tail(5)
```

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping
205	13.89	14.02	0.8880	5.439	3.199	3.986	4.738
206	16.77	15.62	0.8638	5.927	3.438	4.920	5.795
207	14.03	14.16	0.8796	5.438	3.201	1.717	5.001
208	16.12	15.00	0.9000	5.709	3.485	2.270	5.443
209	15.57	15.15	0.8527	5.920	3.231	2.640	5.879

The data loaded in the dataset has 210 rows and 7 columns. There are no duplicate rows available in the dataset.

We have to check the mean, median, standard deviation of the dataset in order see whether the data need to be scaled up further before proceeding to clustering and K Means.

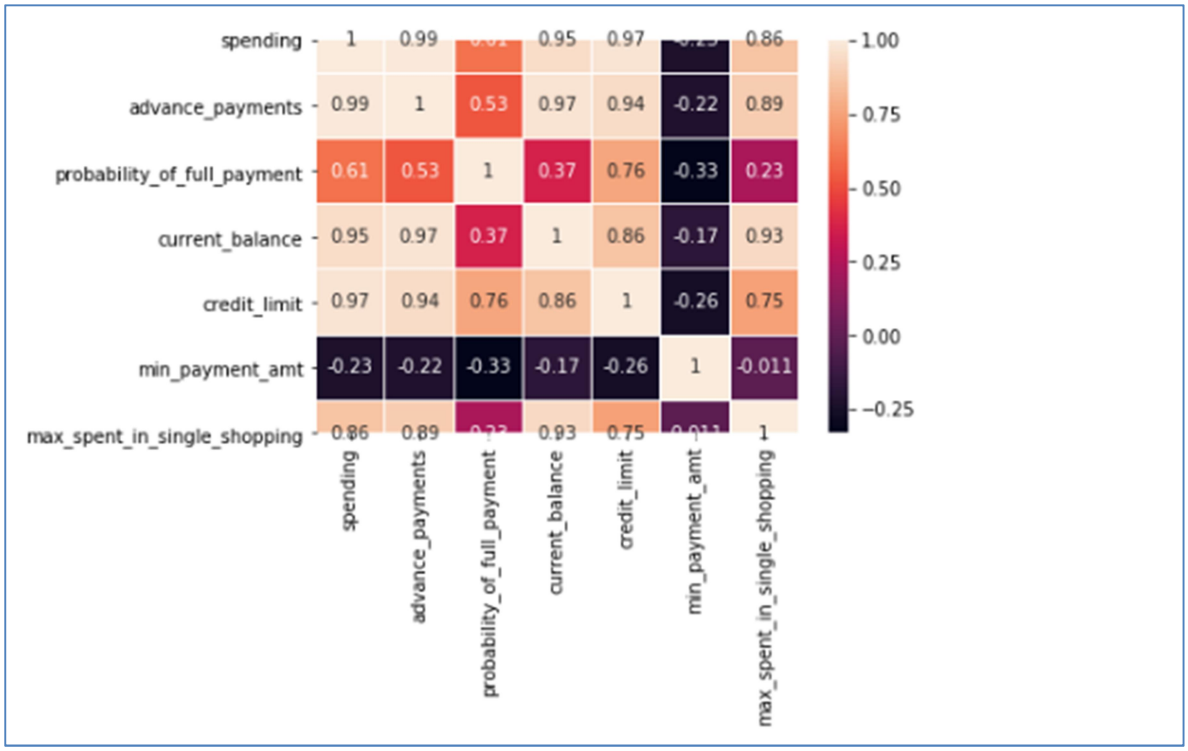
	count	mean	std	min	25%	50%	75%	max
spending	210.0	14.847524	2.909699	10.5900	12.27000	14.35500	17.305000	21.1800
advance_payments	210.0	14.559286	1.305959	12.4100	13.45000	14.32000	15.715000	17.2500
probability_of_full_payment	210.0	0.870999	0.023629	0.8081	0.85690	0.87345	0.887775	0.9183
current_balance	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

Check the correlation of the dataset using “Pearson’s” method.

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

We will be presenting the correlation in heat map and pair plot to analyse whether there is any correlation between the customer spending, advance payments, credit limit, etc.

Heat Map:



The darker the colour shows that there is lesser correlation and lighter the colour in the heat map shows that they are highly correlated. The data shows that spending of the customer is highly correlated with the advance payments options, the probability of paying in full, current balance, credit limit and their maximum spending on single shopping. There is less correlation between the spending and minimum payment amount.

We should check the dataset is having any outliers or not. If any outliers are found, that needs to be treated.

We could identify that there is one outlier in Minimum payment amount and one outlier detected on probability of full payment. Rest all the data are not having any outliers. After correcting the data for outliers, we could find there is no outliers present in the dataset.

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

We have to normalize the dataset before processing the K Means algorithm the data set should be scaled. In this dataset the spending and advance payment are showing high values, hence considering rest of the data, this data has to be scaled. K Means clustering is "isotropic" in all direction of space and hence it trends to produce more or less clusters. Standardization of data is recommended as the range of values in each feature will act as weight while determining the cluster.

To process the scaling we will be importing Standard Scaler from sklearn package.

	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

We have scaled the data and created a data frame out of the scaled data for processing clustering.

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

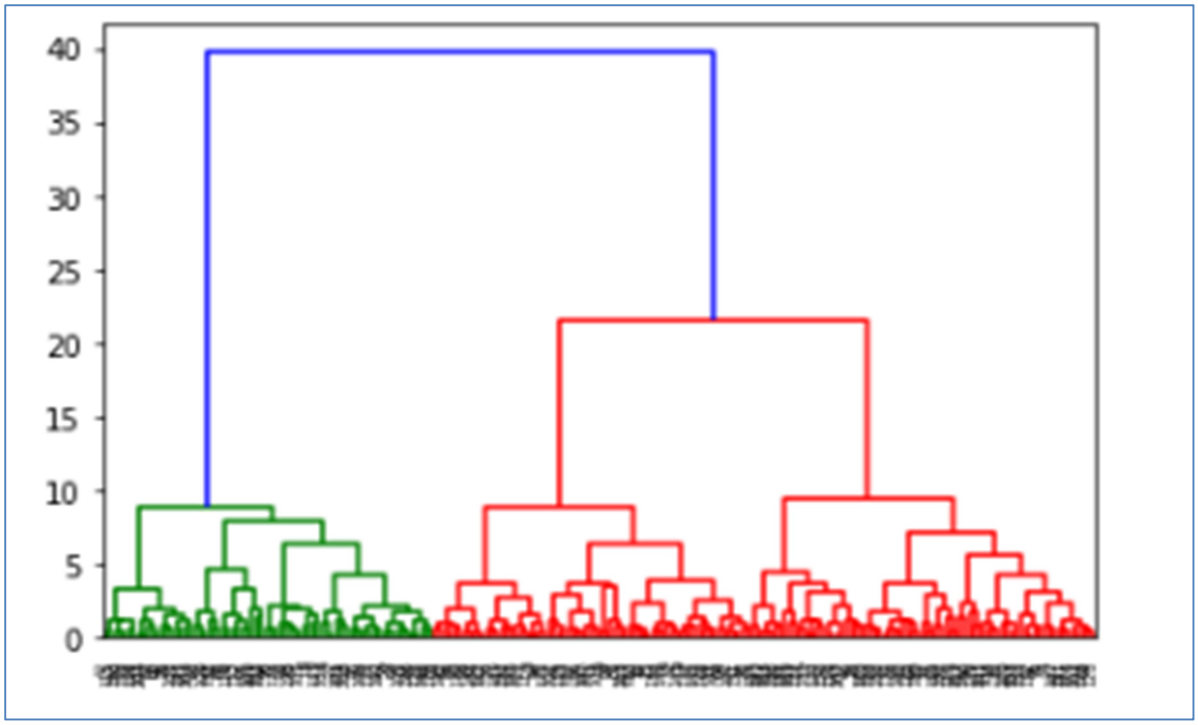
Hierarchical Clustering is an unsupervised clustering algorithm which involves in creating clusters that have predominant ordering from top to bottom. The algorithm groups similar objects into groups named as clusters. The end point is the subset of clusters.

Dendrogram is a type of tree diagram showing the relation between the different set of data

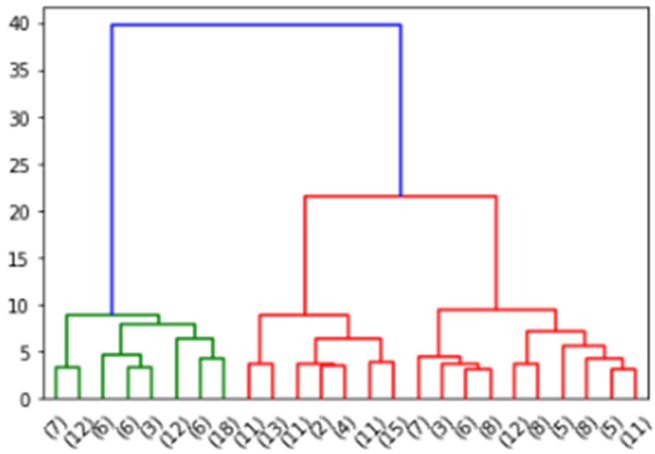
In order to perform the hierarchical clustering we have to import packages from scipy.

```
from scipy.cluster.hierarchy import dendrogram, linkage
```

We will be using the Linkages method as "Ward Method" on the scaled data for performing the cluster analysis.



From the Dendrogram that has generated have many leaves or clades in the tree. Hence we will be looking at the last 25 merges of the diagram. From that we could conclude with the optimum cluster that has been derived from the banking dataset.



We cut the Dendrogram tree with a horizontal line at the height where the line can traverse the maximum distance up and down without intersecting the merging point. Here we will be considering 3 as the number of clusters. The cluster has divided on the basis of spending of the customer.

	spending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	H_clusters
0	19.94	16.92	0.875200	6.675	3.763	3.252	6.550	1
1	15.99	14.89	0.906400	5.363	3.582	3.336	5.144	3
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	2
4	17.99	15.86	0.899200	5.890	3.694	2.068	5.837	1

We have two methods here with cluster that has criterion as “maxclust” and “distance”. We have created 2 clusters and we could find both the clusters created are true. We have added another column to the database as H_Cluster, where the whole data is divided under 3 clusters.

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

K-means clustering is a clustering algorithm that aims to partition n observations into k clusters.

There are 3 steps:

- Initialisation – K initial “means” (centroids) are generated at random
- Assignment – K clusters are created by associating each observation with the nearest centroid
- Update – The centroid of the clusters becomes the new mean

Assignment and Update are repeated iteratively until convergence

The end result is that the sum of squared errors is minimised between points and their respective centroids.

We will import the packages form sklearn.clusters.

```
from sklearn.cluster import KMeans
```

In order to calculate the K Means centroid, we will first check the inertia of the various clusters. Firstly we will check the inertia of the 3rd cluster, 4th cluster and 5th cluster.

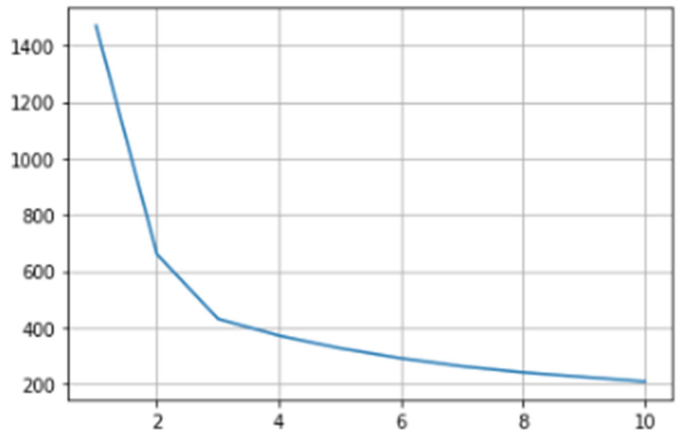
The inertia of 1st cluster id showing as 1470, the 2nd cluster shows 659, 3rd cluster shows 371, 4th cluster 326. We could find the inertia value doesn’t have any high difference after 3rd cluster.

WSS:

```
wss
```

```
[1470.0,
 659.1474009548498,
 430.2984817512229,
 371.0356644664014,
 326.06187488248025,
 290.7634026923419,
 261.97595012053785,
 240.68364375717326,
 224.3898166609323,
```

Elbow Curve:



The elbow curve shows that from first inertia to the 2nd inertia there is a drastic fall. After 3rd cluster the graph shows depletion

Silhouette Score:

Import the libraries' from sklearn packages for silhouette score.

```
from sklearn.metrics import silhouette_samples, silhouette_score
```

Considering the optimal cluster as 3, we will calculate the silhouette score for the dataset.

```
silhouette_score(scaled_df,labels)
```

```
0.4008059221522216
```

Silhouette sample_min

```
silhouette_samples(scaled_df,labels).min()
```

```
0.00276854112861622
```

Update the silhouette scores to the dataset

pending	advance_payments	probability_of_full_payment	current_balance	credit_limit	min_payment_amt	max_spent_in_single_shopping	Clus_kmeans	sil_width
19.94	16.92	0.875200	6.675	3.763	3.252	6.550	1	0.573278
15.99	14.89	0.906400	5.363	3.582	3.336	5.144	0	0.365564
18.95	16.42	0.882900	6.248	3.755	3.368	6.148	1	0.637092
10.83	12.96	0.810588	5.278	2.641	5.182	5.185	2	0.515595
17.99	15.86	0.899200	5.890	3.694	2.068	5.837	1	0.360972

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

The cluster has divided the data on the basis of spending and the probability of paying full.

- 1st cluster is bases on the spending falls in between 12 to 16 (in 1000s) where the advance paid by the customer falls in between 13 to 15. The probability of paying falls in between 85 to 91%.The credit limit of the customer falls in between 2.9 to 3.5. The current balance of these customers falls in between 4.9 to 5.9. These customers spend 4.6 to 5.8 in single window shopping.
- 2nd cluster is based on the spending of customer's falls in between 15 to 21(in 1000s) where the advance payment is 15 to 17. The probability of paying falls in between 84 to 91%.The current balance of the customers falls in between 5.7 to 6.6. The credit limit of the customers falls in between 3.3 to 4. These customers spend 5.4 to 6.5 (1000s) in single shopping.
- 3rd cluster is based on the customer whose spending falls in between 10 to 13 (in 1000s), where the advance paid by the customers in cash falls in between 12 to 13. The probability of the customers making full payments will fall 3in between 81 to 88%. The current balance of the customers falls in between 4.8 to 5.5. The credit limit of the customers falls in between 2.6 to 3.2. These customers spend 4.5 to 5.4 (1000s) in one shopping.

Strategies for all the three groups:

- The bank should gradually increase the credit limit of the cards for each customers falling in cluster 3 by 1%.
- They should introduce new promotional plans and products to those customers who falls under cluster 2
- The cluster 1 and cluster 3 spends high on single window purchases irrespective their probability of paying back is also high. The bank should be looking upon to these two clusters high for a better sales and promotional offeres.
- All this depends on the other variables like income and their nature of spending. This is even applied to those customers who are good customers rather than moving with bad payers.

****

Case Study 2:

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

Data set: insurance_part2_data-1.csv

Attribute Information:

1. Target: Claim Status (Claimed)
2. Code of tour firm (Agency_Code)
3. Type of tour insurance firms (Type)
4. Distribution channel of tour insurance agencies (Channel)
5. Name of the tour insurance products (Product)
6. Duration of the tour (Duration)
7. Destination of the tour (Destination)
8. Amount of sales of tour insurance policies (Sales)
9. The commission received for tour insurance firm (Commission)
10. Age of insured (Age)

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

Import all the relevant libraries' for the dataset.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
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
import warnings
warnings.filterwarnings("ignore")
```

Import the dataset

```
df=pd.read_csv("insurance_data.csv")
```

```
df.head()
```

	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

We do not require the column “Agency Code” for processing the CART, RF and ANN models. Hence we will drop the column from the dataset.

	Age	Type	Claimed	Commision	Channel	Duration	Sales	Product Name	Destination
0	48	Airlines	No	0.7	Online	7	2.51	Customised Plan	ASIA
1	36	Travel Agency	No	0.0	Online	34	20.00	Customised Plan	ASIA

The dataset that we are given is have 3000 rows and 10 columns. After dropping the Agency Code column we have 9 columns. Hence the shape of the dataset is showing 3000 rows and 9 Columns

There were no NAN values and Null Values in the dataset.

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

The data type of the dataset is having a combination of Object variables, Integer and Float variables. For processing the CART, RF and ANN we need to convert the Object data type to categorical variables.

```
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 9 columns):
Age             3000 non-null int64
Type            3000 non-null object
Claimed         3000 non-null object
Commision       3000 non-null float64
Channel         3000 non-null object
Duration        3000 non-null int64
Sales           3000 non-null float64
Product Name    3000 non-null object
Destination     3000 non-null object
dtypes: float64(2), int64(2), object(5)
memory usage: 211.1+ KB
```

```
for feature in df.columns:
    if df[feature].dtype == 'object':
        print('\n')
        print('feature:',feature)
        print(pd.Categorical(df[feature].unique()))
        print(pd.Categorical(df[feature].unique()).codes)
        df[feature] = pd.Categorical(df[feature]).codes
```

```
Int64Index: 2861 entries, 0 to 2999
Data columns (total 9 columns):
Age             2861 non-null int64
Type            2861 non-null int8
Claimed         2861 non-null int8
Commision       2861 non-null float64
Channel         2861 non-null int8
Duration        2861 non-null int64
Sales           2861 non-null float64
Product Name    2861 non-null int8
Destination     2861 non-null int8
dtypes: float64(2), int64(2), int8(5)
memory usage: 125.7 KB
```

We have to check any rows and columns are there in the dataset. We could find 139 rows are duplicated in the dataset. We will drop all the duplicated rows in the dataset.

```
print('Number of rows before discarding duplicates = %d' % (df.shape[0]))
df.drop_duplicates(subset = None, keep = 'first', inplace=True)
print('Number of rows after discarding duplicates = %d' % (df.shape[0]))
```

```
Number of rows before discarding duplicates = 3000
Number of rows after discarding duplicates = 2861
```

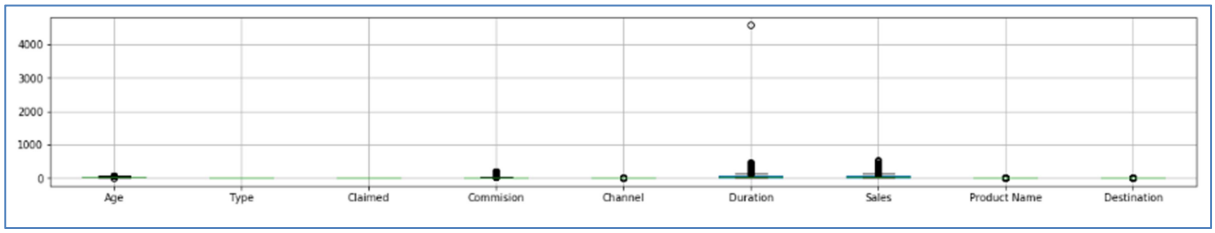
We will check the description of the dataset. We have to find the mean, Standard Deviation, Max, Min of the data set.

	count	mean	std	min	25%	50%	75%	max
Age	2861.0	38.204124	10.678106	8.0	31.0	36.00	43.00	84.00
Type	2861.0	0.597344	0.490518	0.0	0.0	1.00	1.00	1.00
Claimed	2861.0	0.319469	0.466352	0.0	0.0	0.00	1.00	1.00
Commision	2861.0	15.080996	25.826834	0.0	0.0	5.63	17.82	210.21
Channel	2861.0	0.983922	0.125799	0.0	1.0	1.00	1.00	1.00
Duration	2861.0	72.120238	135.977200	-1.0	12.0	28.00	66.00	4580.00
Sales	2861.0	61.757878	71.399740	0.0	20.0	33.50	69.30	539.00
Product Name	2861.0	1.666550	1.277822	0.0	1.0	2.00	2.00	4.00
Destination	2861.0	0.261797	0.586239	0.0	0.0	0.00	0.00	2.00

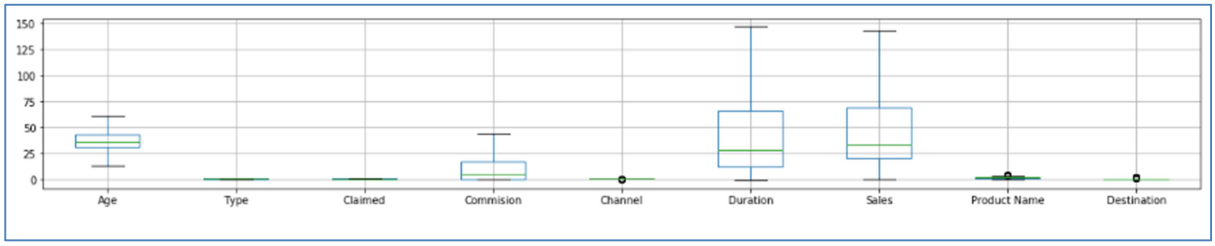
The data is not scaled and we have to scale the dataset. As the mean for some variables is more and the standard distribution is not 1. Hence we should use the “StandardScaler” package from sklearn to scale the data.

We will be checking the outliers of the dataset. We could find that there are many outliers in the dataset. Where treatment of outlier were not necessary for CART and Random Forest. Here for this dataset we are treating the outliers for a better result.

Box Plot:

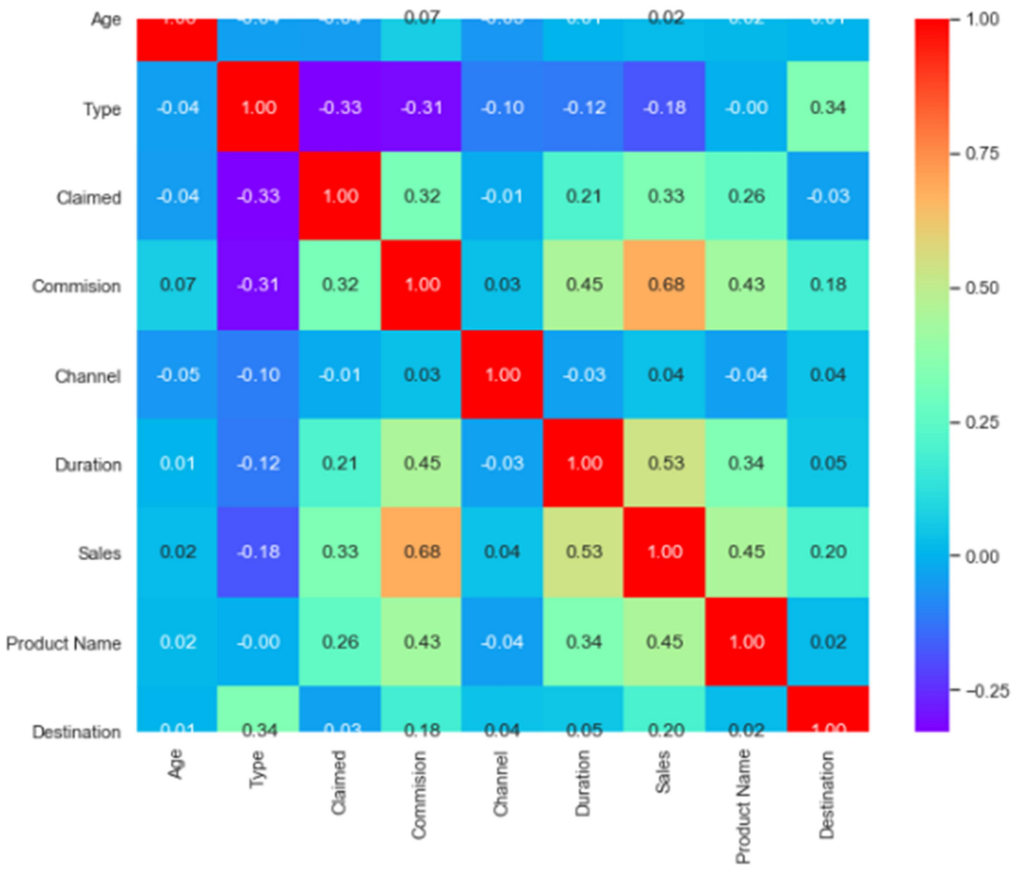


After treatment of outliers:



We will be checking the correlation of the dataset.

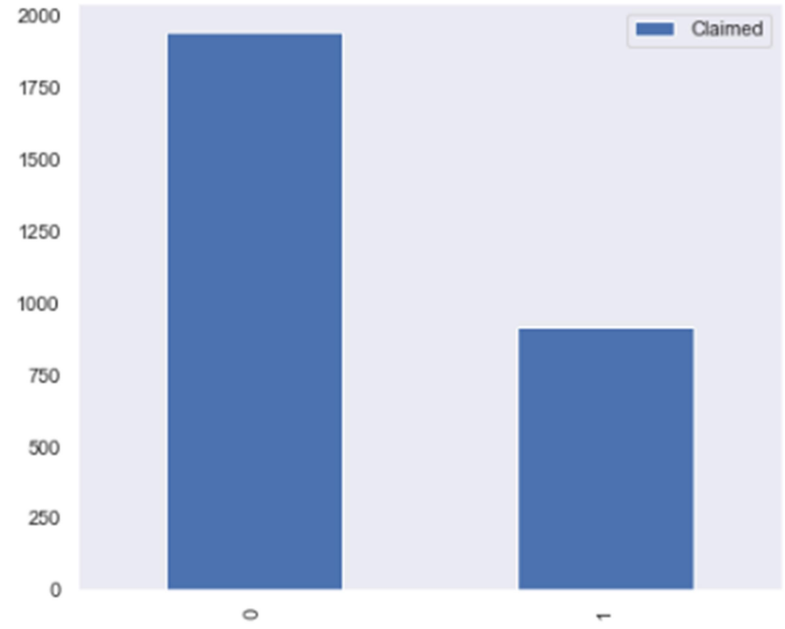
Heat Map:



The data set shows the Red as the highly correlated and purple as the least correlation. The majority of the data are showing positive correlated. Commission and the sales are highly correlated in the dataset.

Before splitting the data we will be checking the percentage of the Claims in the dataset.

```
Percentage of 0's 68.05 %
Percentage of 1's 31.95 %
```



Here 1 denotes that travel insurance is claimed and 0 indicates that travel insurance is not claimed.

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

We will split the dataset on with the target variable "Claimed". In order to split the dataset we will use the library "train test split" from sklearn package.

```
X = df.drop("Claimed", axis=1)
y = df.pop("Claimed")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
```

CART Model:

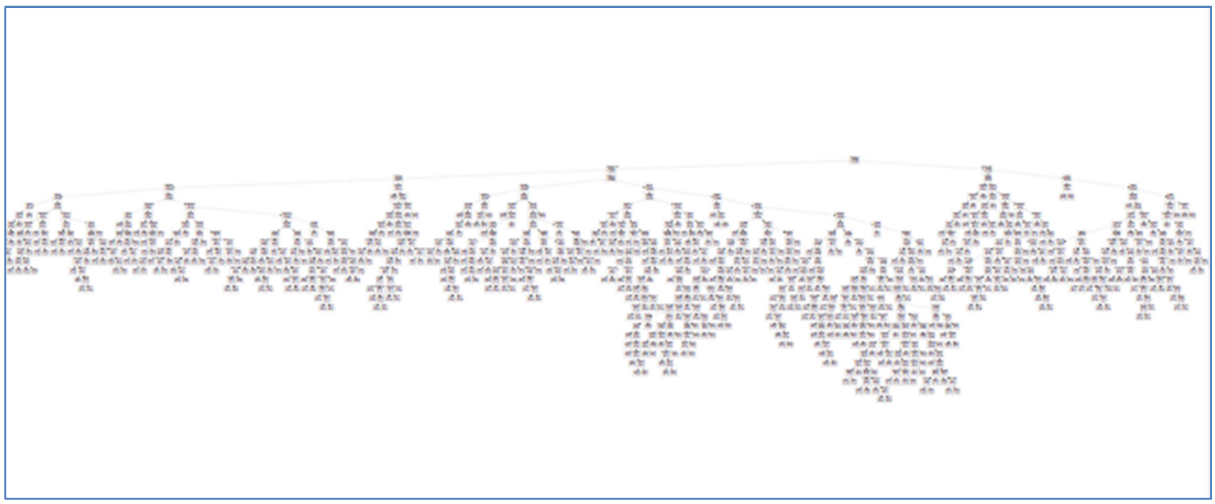
A **decision tree** is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (**decision** taken after computing all attributes)

In order to build the decision tree model we will be importing the library from skleran as Decision Tree Classifier. To construct the tree we will be using another package from sklearn named tree.

```
dtree.fit(X_train,y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

We will import the tree in order to view the depth of the tree. We have to identify how many splits have happened for the whole dataset. If the tree has grown further, then we have to use the pruning technique. The model uses Gini Index as the criterion.

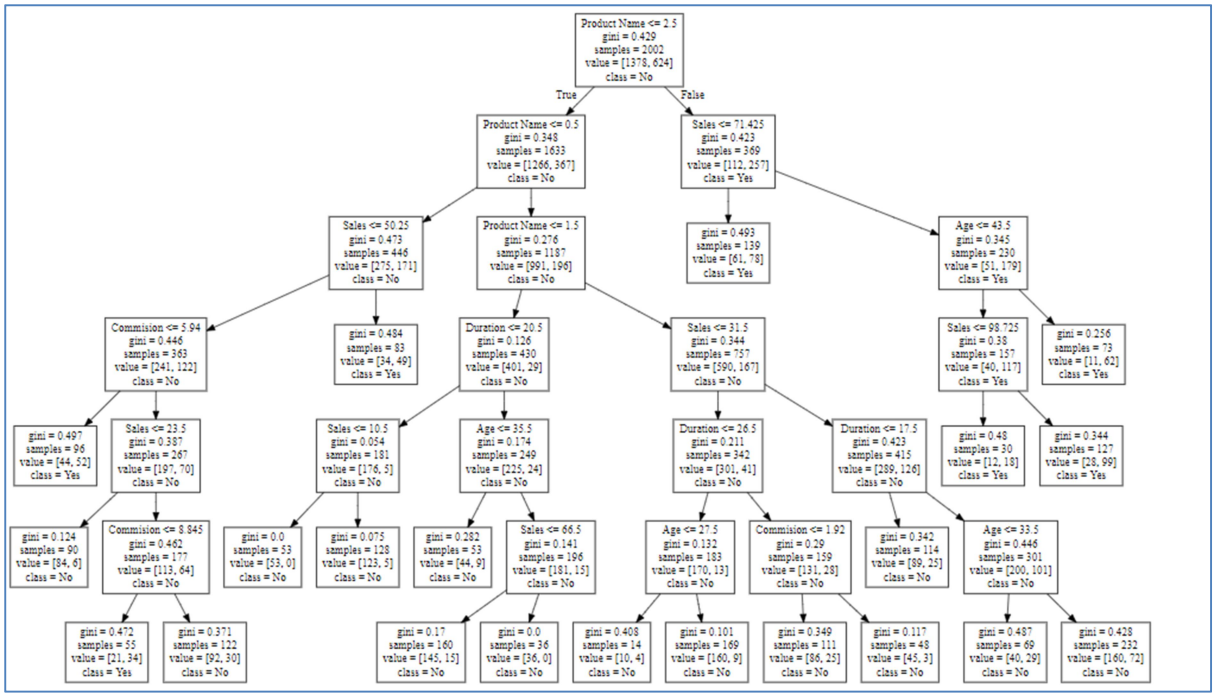


We have not used any pruning techniques for this graph. Hence we have to regularise the graphs by using pruning techniques.

```
reg_dtree = DecisionTreeClassifier(criterion = 'gini', max_depth = 6,min_samples_leaf=10,min_samples_split=150,random_state=0)
reg_dtree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=6,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=10, min_samples_split=150,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

We will be creating a dot file. The dot file data are copied and pated in online web graphs portal. We will be getting the graphs on the basis of pruning techniques given above.



Random Forest:

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds.

Import the library from SKLearn package for Random Forest Classifier.

```
from sklearn.ensemble import RandomForestClassifier
```

We will split the data into test and train with the target variable "Claimed".

Here we will decide how many number of trees to need to be constructed. In this case we are constructing 350 trees for this dataset.

We will check the Out of Bag data points for this dataset. The OOB Score is 73.6%

The Maximum level of the trees considering for the data set is 7

The total number of independent variables for this tree is 4

There will be 50 observations to be presented in the terminal notes.

The minimum sample splits will be 150

We will update all these in a Grid Search CV package to construct the Random Forest classifier.

```
param_grid = {
    'max_depth': [7],#10
    'max_features': [4],#6
    'min_samples_leaf': [50],#150
    'min_samples_split': [150],#550
    'n_estimators': [350]#50
}

rfcl = RandomForestClassifier()

grid_search = GridSearchCV(estimator = rfcl, param_grid = param_grid, cv = 5)
```

Fit the Test data and Train data on the grid search as X Train and Y Train.

```
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True, class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators='warn', n_jobs=None,
                                              oob_score=False,
                                              random_state=None, verbose=0,
                                              warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'max_depth': [7], 'max_features': [4],
                         'min_samples_leaf': [50], 'min_samples_split': [150],
                         'n_estimators': [350]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

We used different variable out of which the best fit are derived as below:

```
grid_search.best_params_
```

```
{'max_depth': 7,
 'max_features': 4,
 'min_samples_leaf': 50,
 'min_samples_split': 150,
 'n_estimators': 350}
```

The random forest classifier for the test and the train data are set as below:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=7, max_features=4, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=50, min_samples_split=150,
                       min_weight_fraction_leaf=0.0, n_estimators=350,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

We will predict the Train and test data.

```
ytrain_predict = best_grid.predict(X_train)
ytest_predict = best_grid.predict(X_test)
```

Artificial Neural Network:

A **neural network** is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. They interpret sensory data through a kind of machine perception, labelling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated. Neural networks can also extract features that are fed to other algorithms for clustering and classification; so you can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression.

The data set has to be scaled in before doing the artificial neural network. Hence we will be using the package named Standard Scaler from SK learn. We use the function Z score to scale the data.

```
X_Train=SC.fit_transform(X_train)
```

We will construct the best grid for the test and train data as below:

```
param_grid = {
    'hidden_layer_sizes': [350], #,50,100
    'max_iter': [2000], #5000,2500,1500
    'solver': ['sgd'], #sgd, adam
    'tol': [0.01],
}

nncl = MLPClassifier()

grid_search = GridSearchCV(estimator = nncl, param_grid = param_grid, cv = 3)
```

We tried using several other variable, where we are getting the best fit result by using the above constrains. Here the hidden layer is identified as 350, the maximum iter is 2000. The solver is considered as “Stochastic Gradient Decent” or sgd (“adam” as solver, where this was not providing an good recall value, hence changed to sgd).

Fit the Training and Test data for constructing the neural network model.

```
grid_search.fit(X_train, y_train)
grid_search.best_params_c
```

```
{'hidden_layer_sizes': 350, 'max_iter': 2000, 'solver': 'sgd', 'tol': 0.01}
```

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=350, learning_rate='constant',
              learning_rate_init=0.001, max_iter=2000, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=None, shuffle=True, solver='sgd', tol=0.01,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

Predict the test data and train data.

```
ytrain_predict = best_grid.predict(X_train)
ytest_predict = best_grid.predict(X_test)
```

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

1) Performance of Prediction on Train & Test data set for CART:

After finalizing the tree, the next step is to predict the Train and test data values.

```
ytrain_predict = reg_dtree.predict(X_train)
ytest_predict = reg_dtree.predict(X_test)
```

The classification report will share us the performance of the model. The performance of the model is split as below:

- Confusion Matrix
- Accuracy of the model
- Sensitivity of the model
- Precision of the model
- Specificity of the model.

	precision	recall	f1-score	support
0	0.76	0.75	0.76	569
1	0.52	0.54	0.53	290
accuracy			0.68	859
macro avg	0.64	0.65	0.65	859
weighted avg	0.68	0.68	0.68	859

- **Confusion Matrix**: summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Here for our dataset the training Confusion Matrix is as below:

```
confusion_matrix(y_train, ytrain_predict)

array([[1167,  211],
       [ 232,  392]], dtype=int64)
```

The True positive is 1167 and True Negative is 392.

Confusion Matrix for Test data is as below:

```
confusion_matrix(y_test, ytest_predict)

array([[481,  88],
       [117, 173]], dtype=int64)
```

The True Positive is 481 and True Negative is 173

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$

  If the accuracy is 99%, it is believed that the model is excellent. Here for the data set the accuracy of training and test are as below.

```
cart_train_acc=reg_dtree.score(X_train,y_train)
cart_train_acc

0.7787212787212787
```

The Accuracy for the training data is showing as 77.8% which we can say its good not excellent.

```
cart_test_acc=reg_dtree.score(X_test,y_test)
cart_test_acc

0.7613504074505238
```

The accuracy of the test data is showing as 76.1%. if compared training data is showing 1% better than the test data.

- Recall = $\frac{TP}{TP+FN}$

  Here the training data set shows a recall of 64% and Test data shows 56%

- Precision = $\frac{TP}{TP+FP}$

Here the training data shows precision of 63% and test data shows 61%

```
cart_metrics=classification_report(y_train, ytrain_predict,output_dict=True)
df=pd.DataFrame(cart_metrics).transpose()
cart_train_precision=round(df.loc["1"][1],2)
cart_train_recall=round(df.loc["1"][2],2)
cart_train_f1=round(df.loc["1"][0],2)
print ('cart_train_precision ',cart_train_precision)
print ('cart_train_recall ',cart_train_recall)
print ('cart_train_f1 ',cart_train_f1)
```

```
cart_train_precision  0.63
cart_train_recall  0.64
cart_train_f1  0.65
```

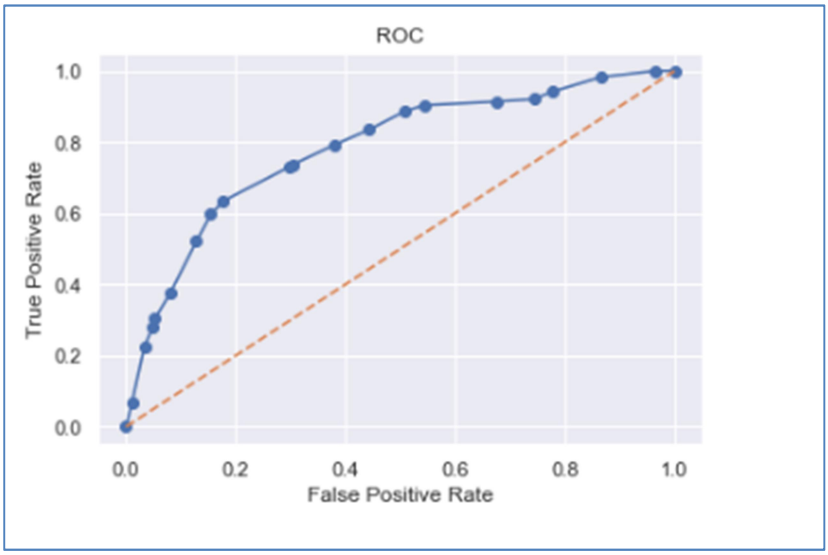
```
cart_metrics=classification_report(y_test, ytest_predict,output_dict=True)
df=pd.DataFrame(cart_metrics).transpose()
cart_test_precision=round(df.loc["1"][1],2)
cart_test_recall=round(df.loc["1"][2],2)
cart_test_f1=round(df.loc["1"][0],2)
print ('cart_test_precision ',cart_test_precision)
print ('cart_test_recall ',cart_test_recall)
print ('cart_test_f1 ',cart_test_f1)
```

```
cart_test_precision  0.61
cart_test_recall  0.56
cart_test_f1  0.52
```

ROC Curve and AUC Score for CART:

A **receiver operating characteristic curve**, or **ROC curve**, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

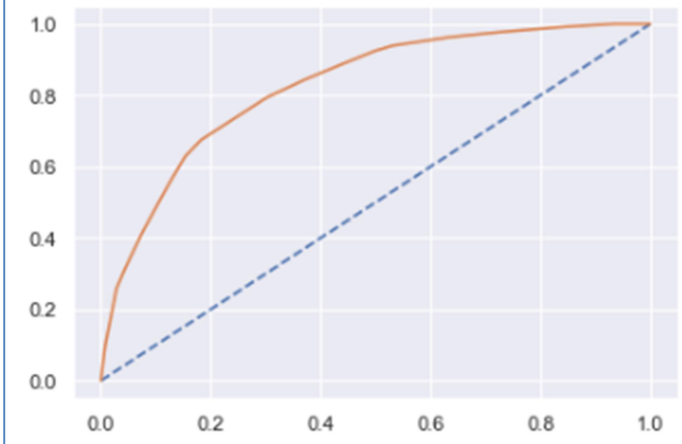
The below illustrated is the ROC curve for the CART Model retrieved from the dataset



AUC represents the probability that a random positive is positioned to the right of a random negative

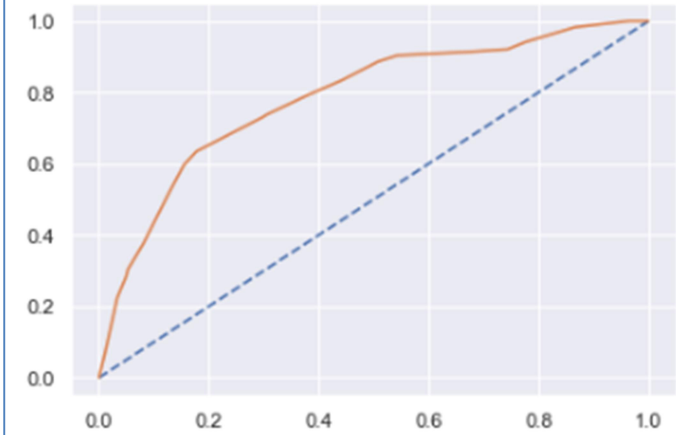
AUC Score for Training model:

```
AUC: 0.826

[<matplotlib.lines.Line2D at 0x17dfecd9f88>]
```



AUC Curve for the Test Model:

```
AUC: 0.784

[<matplotlib.lines.Line2D at 0x17dfeca70c8>]
```



2) Performance for Random Forest Model

After the model is constructed, we will predict the test and train variables as per the model.

```
ytrain_predict = best_grid.predict(X_train)
ytest_predict = best_grid.predict(X_test)
```

The performance of the model for the train data is checked on the basis of the confusion matrix, Recall, Precision and F1 score retrieved out of the model results.

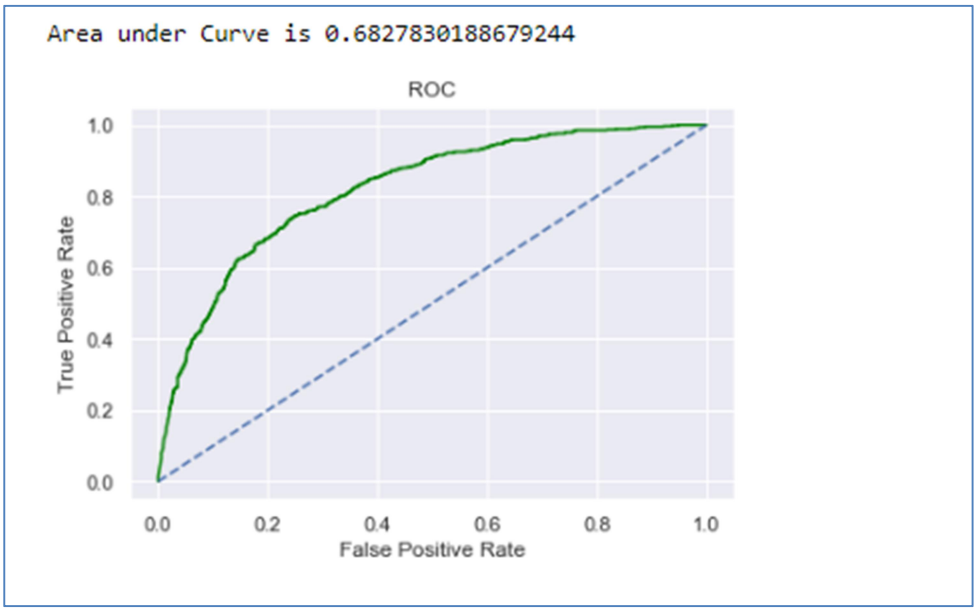
I. Accuracy of the Model is 76.9%
II. The classification report for the train data is as below. 0 is denotes as the No claims and 1 been claimed on travel insurance.

	precision	recall	f1-score	support
0	0.79	0.91	0.85	1378
1	0.70	0.45	0.55	624
accuracy			0.77	2002
macro avg	0.74	0.68	0.70	2002
weighted avg	0.76	0.77	0.75	2002

III. The performance of the Training model is represented as below: The precession is 45% , that shows a week model. Recall is 55%, an average result on the model and F1 score is 70%.

```
rf_train_precision  0.45
rf_train_recall  0.55
rf_train_f1  0.7
```

IV. The AUC score for the train data for random forest is 68%.



The performance of the model for the test data is checked on the basis of the confusion matrix, Recall, Precision and F1 score retrieved out of the model results.

I. The Accuracy of the test data is 73.9%
II. The confusion metrix shows that there are 516 true positives and 119 True negative variables in the test data.

```
array([[516,  53],
       [171, 119]], dtype=int64)
```

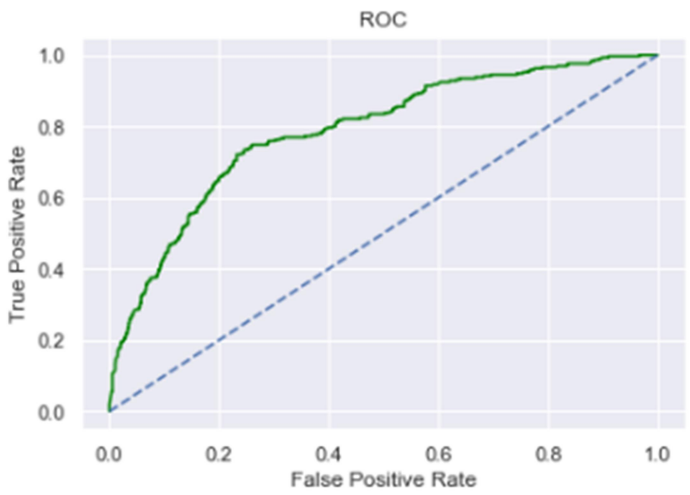
III. The performance of the test model is showing as below where the recall is 52% where the model is very week. The precision is also low with 41%.

	precision	recall	f1-score	support
0	0.75	0.91	0.82	569
1	0.69	0.41	0.52	290
accuracy			0.74	859
macro avg	0.72	0.66	0.67	859
weighted avg	0.73	0.74	0.72	859

```
rf_test_precision  0.41
rf_test_recall  0.52
rf_test_f1  0.69
```

IV. The AUC value and ROC curve for the Test model shows 65.8%. wider the model the stronger the model will be. Here the model is an average one.

```
Area under Curve is 0.6585994788194656
```



3) <u>Performance for Artificial Neural Network:</u>

After constructing the model we will predict the results for the test and training data

```
ytrain_predict = best_grid.predict(X_train)
ytest_predict = best_grid.predict(X_test)
```

The performances of the Training model for the artificial model are derived as below:

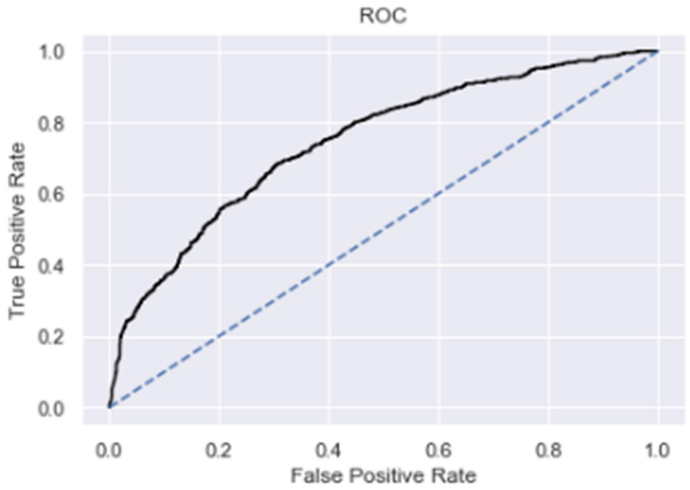
I. The accuracy of the Training data shows 69%

II. The performance of the training data shows that recall is 57 % which is low. The precision is 56% and the F1 score is 51. The performance of the neural network is very low compared to the other models.

```
nn_train_precision  0.65
nn_train_recall  0.57
nn_train_f1  0.51
```

III. The AUC curve and ROC graph for the training data is showing as 68%. The curve is steeper, that shows a weaker model.

```
Area under Curve is 0.6835238268021288
```



The performance of the test model derived from the neural network model is as below:

I. The accuracy of the model is 67.8%

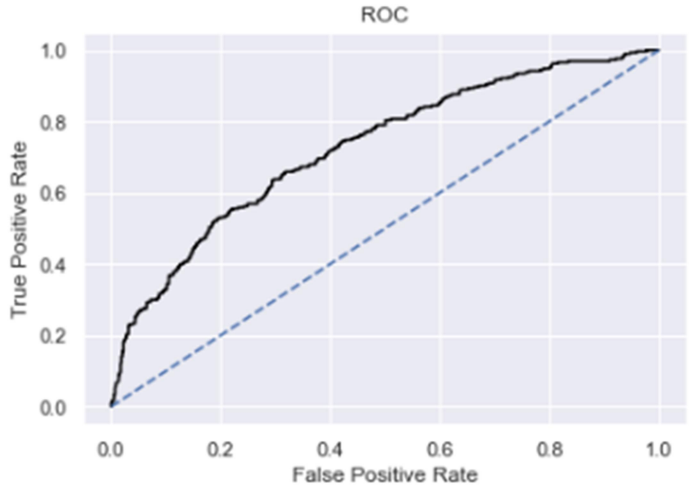
II. The performance of the test model tells that the recall is 56% and precision is 61 %. The F1 score is 52 %. This model is a weak model.

```
nn_test_precision  0.61
nn_test_recall  0.56
nn_test_f1  0.52
```

	precision	recall	f1-score	support
0	0.78	0.71	0.75	569
1	0.52	0.61	0.56	290
accuracy			0.68	859
macro avg	0.65	0.66	0.65	859
weighted avg	0.69	0.68	0.68	859

III. The AUC and ROC graph for the test data is showing a 66 % . The curve is steeper to the model is proving weaker.

```
Area under Curve is 0.6627840736925036
```



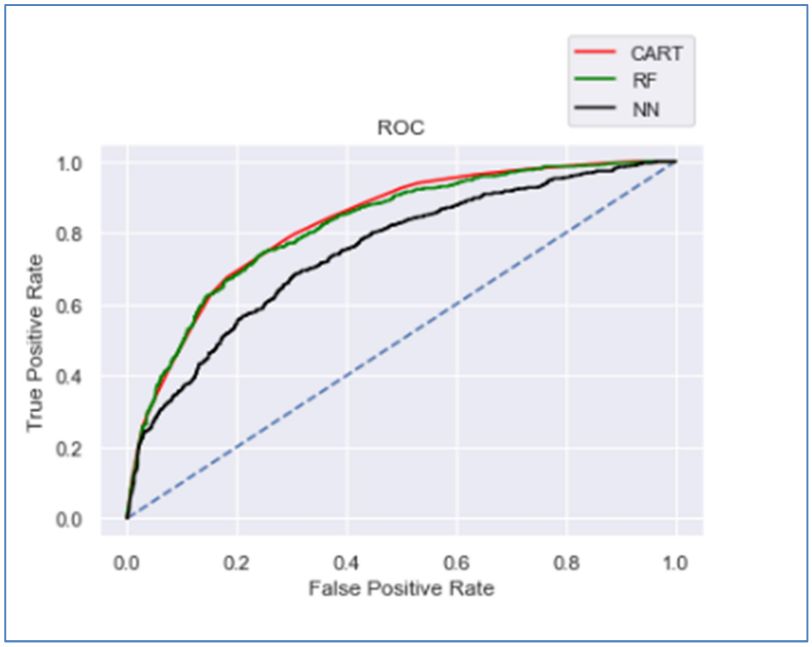
2.4 Final Model: Compare all the model and write an inference which model is best/optimized.

The comparison for the entire three models is as below:

	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.78	0.76	0.77	0.74	0.70	0.68
AUC	0.83	0.78	0.68	0.66	0.68	0.66
Recall	0.64	0.56	0.55	0.52	0.57	0.56
Precision	0.63	0.61	0.45	0.41	0.65	0.61
F1 Score	0.65	0.52	0.70	0.69	0.51	0.52

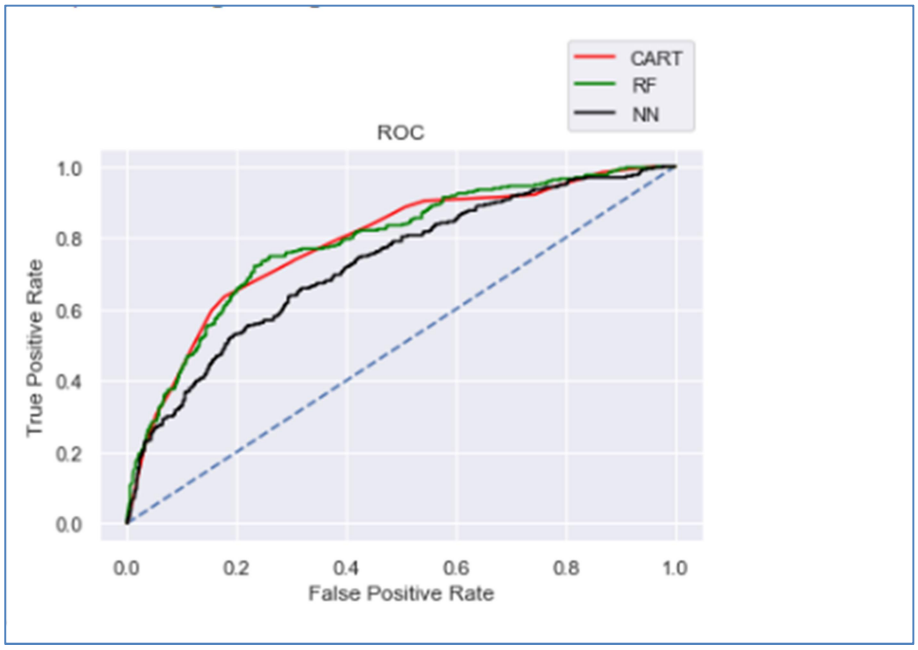
- Comparing the Accuracy out of three model “Decision Tree” and Random Forest model proves to be better. There is no significance difference in both the models.
- AUC score also shows better for CART. Random Forest and Neural Network shows same AUC scores.
- F1 Score shows better for Random Forest than Cart and ANN.
- The Precision is high in neural network compared with the other two models.

The ROC Curve for the Training Data:



The graphs shows that ROC curve for Training shows both Random forest and CART are graphed equally on the curve. For the training data we opt for the CART or Decision tree and Random Forest as the best model

ROC Curve for Test Data:



The graphs show that ROC curve for Test shows both Random forest is better among the three models. ANN AUC score really low, where the graph is showing weaker. Here we can either go for CART or decision tree and Random Forest as the best model. Both the models showing stronger ROC curve.

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

Compared to the three models, CART and Random forest are the strongest models. Here we could find the Product Name has a significant role in getting the insurance claimed or not. We could find the majority of the travellers have not claimed their insurance. Hence the business should increase the sale in the best product plan that would help the business in getting fewer claims.