Data Mining Project CART + RF + ANN
From: 24st Jan,2021 to 23th Feb,2021
Group Members: Niranjan Dhavan, Sumedh Budukh & Sourabh Panigrahi

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.

Insurance Data

A data frame containing: Description

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)

Performing exploratory data analysis on the dataset. Showcasing some charts & graphs.

1. Loading the data set- We will be loading the" insurance_part2_data.csv" file using pandas library in python. For this, we will be using read_csv file.

2. The head function will tell you the top head records in the data set. By default, python shows you only the top 5 records, but we will check top 10 records.

3. The tail function will tell you the top tail records in the data set. By default, python shows you only the top 5 records, but we will check top 10 records for the totals/subtotals if any. The insurance_part2_data dataset doesn't contain any total/subtotals.

4. The shape attribute tells us a number of observations and variables we have in the data set. It is used to check the dimension of data. The Income data set has 3000 observations and 10 variables in the data set.

5. info() is used to check the Information about the data and the datatypes of each respective attribute.

Looking at the data in the head function and in info, we come to know that the variables comprise of float, object and integer data types. sklearn in Python does not take the input of object data types when building Classification Trees. So, we need to convert these variables into some numerical form. There are some object type variables (Agency_Code, Type, Claimed, Channel, Product Name, Destination) which has the object data types which we need to convert into numerical form. We shall perform onehot encoding for them all. Further there are no null values in dataset and the count is 3000 for all the variables.

6. The described method will help to see how data has been spread for numerical values. We can clearly see the minimum value, mean values, different percentile values, and maximum values for the Income data set.

	count	mean	std	min	25%	50%	75%	max
Age	3000	38.091	10.463518	8	32	36	42	84
Commision	3000	14.529203	25.481455	0	0	4.63	17.235	210.21
Duration	3000	70.001333	134.053313	-1	11	26.5	63	4580
Sales	3000	60.249913	70.733954	0	20	33	69	539

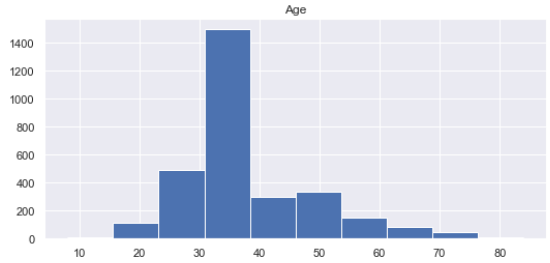
7. Check Proportion of observations in each of the target classes.

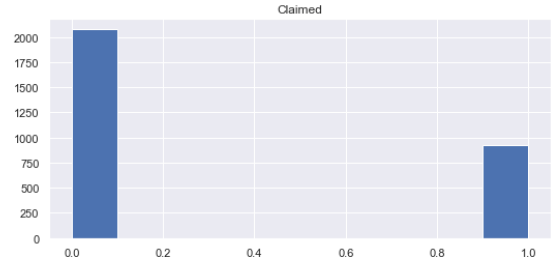
	Numbers	Percentage
0	2076	0.692
1	924	0.308

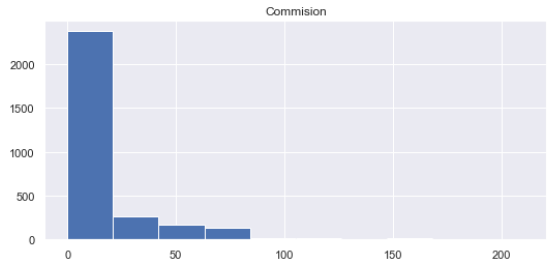
The Target variable seems to be not completely balanced data as 0 are approx. 70% and 1 are approx. 30%.

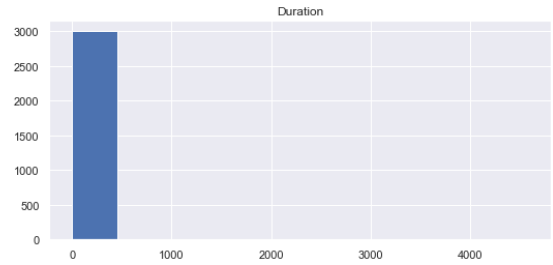
Since 70% & 30% is fairly balanced data. We Are not changing the threshold value / Probability value. We are going ahead considering it as 0.5 itself.

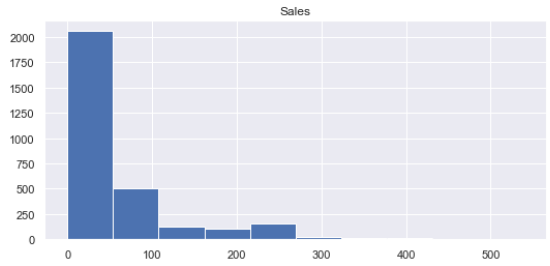
8. from pylab import rcParams Check whether the variables are normally distributed or not



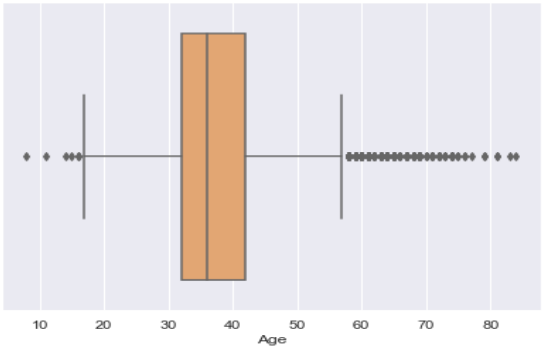


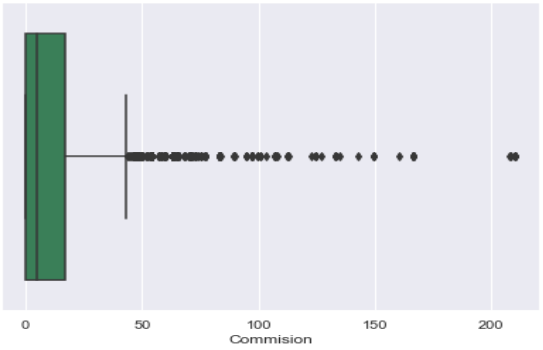


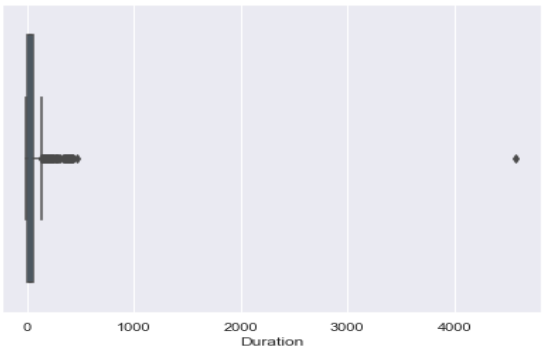


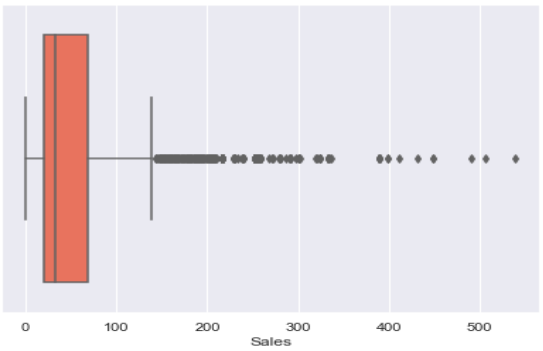


9. Visual Representation of data by Boxplot



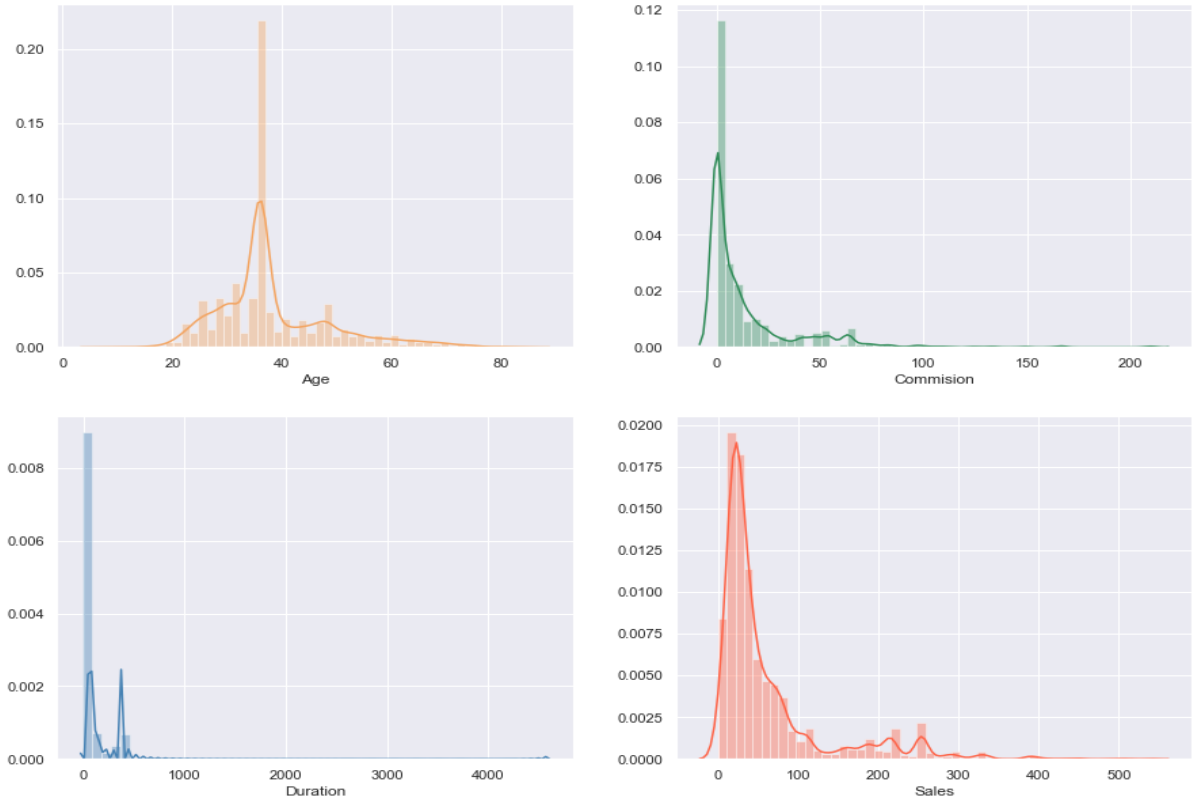






Inference - All the variables (i.e. Age, commission, duration & Sales) has the outliers.

10. Visual Representation of data by Dist. plot



Inference -

- The distribution for Age Variable seems to be not Normal. The distribution tends to be right skewed.
- The distribution for Commission also Variable seems to be not Normal. The distribution tends to be left skewed.
- The distribution for Duration Variable also seems to be not Normal. The distribution tends to be left skewed.
- The distribution for sales Variable also seems to be not Normal. The distribution tends to be Left skewed.

Formatting / Preparing the data to build the Models.

sklearn in Python does not take the input of object data types when building Classification Trees. So, we need to convert these variables into some numerical form. We shall perform onehot encoding for them all.

- Let us define the X (All independent variables) and Y (Dependent ~ "Claimed") variables**.**
- Encoding Categorical Features. As we have some object type variables (Agency_Code, Type, Claimed, Channel, Product Name, Destination) which has the object data types which we need to convert into numerical form. We are encoding so data.
- pd. get_dummies when applied to a column of categories where we have one category per observation will produce a new column (variable) for each unique categorical value. It will place a one in the column corresponding to the categorical value present for that observation. This is equivalent to one hot encoding.
- Post Encoding the Categorical Features. The head of data looks like

	Age	Commision	Duration	Sales	Agency_Code_C2B	Agency_Code_CWT	Agency_Code_EPX	Agency_Code_JZI	Type_Airlines	Type_Travel Agency
0	48	0.7	7	2.51	1	0	0	0	1	0
1	36	0	34	20	0	0	1	0	0	1
2	39	5.94	3	9.9	0	1	0	0	0	1
3	36	0	4	26	0	0	1	0	0	1
4	33	6.3	53	18	0	0	0	1	1	0

Channel_Offline	Channel_Online	Product Name_Bronze Plan	Product Name_Cancellation Plan	Product Name_Customised Plan	Product Name_Gold Plan	Product Name_Silver Plan	Destination_ASIA	Destination_Americas	Destination_EUROPE
0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0	1	0
0	1	0	1	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0

- Before building the model, we should split the data into Train and Test. We will thus build a model on the training data and use this model to predict on the test data.

- We will be doing a 70:30 split. 70% of the whole data will be used to train the data and then 30% of the data will be used for testing the model thus built.

- Importing train_test_split from sklearn.model_selection to splitting data into training and test set for independent attributes

Decision Tree Model

A decision Tree is one of most popular and effective supervised learning technique for classification problem that equally works well with both categorical and quantitative variables. It is a graphical representation of all the possible solution to a decision that is based on certain condition. In this algorithm, the training sample points are split into two or more sets based on the split condition over input variables.

Build a Preliminary tree

- Let us check importance of the variables in the Classification Tree. The importance of a feature or variable is computed as the (normalized) total reduction of the gini criterion brought by that feature. It is also known as the Gini importance.

Duration	0.254863
Sales	0.194513
Agency_Code_C2B	0.188265
Age	0.170710
Commision	0.096285
Product Name_Cancellation Plan	0.015495
Product Name_Customised Plan	0.012677
Product Name_Silver Plan	0.011399
Product Name_Bronze Plan	0.010276
Product Name_Gold Plan	0.009926
Destination_Americas	0.007818
Channel_Offline	0.007465
Destination_EUROPE	0.006234
Destination_ASIA	0.004914
Agency_Code_CWT	0.003748
Type_Travel Agency	0.002088
Type_Airlines	0.001915
Agency_Code_EPX	0.001408
Agency_Code_JZI	0.000000
Channel_Online	0.000000

- From the above output, we can see that 'Duration' is the most important variable followed by 'Sales' and so on.

- Let us take a look at the overall accuracy of the train and test data using the model that we just built.

Train Data	0.9948
Test Data	0.7022

- The accuracy on the Training Data is 99% and the accuracy on the Test Data is 70% which is lesser substantially. The model has surely been overfitted. Thus, we need to prune or regularize the tree.

Pruning/Regularizing the Tree Using Grid Search CV

- GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.

- What the command GridSearchCV does is build multiple models based on a range of parameters specified by us. Then it ultimately returns best parameters on the basis of which we can go ahead and build our model. Let us now see how to define the range of parameters for the GridSearchCV function.

- Lets import GridSearchCV from sklearn.model_selection

- Passing the inputs through Grid Search CV.

```
param_grid = {
'criterion':['entropy','gini'],
'max_depth': [5, 6, 7, 8, 9, 10],
'min_samples_leaf': [25, 40, 41, 42, 43, 44],
'min_samples_split': [150, 175, 200, 210, 220, 230, 240, 250, 260, 270]
}
```

- Now that we have defined a dictionary in Python with all these parameters, it is time to use these parameters to build the model and check for the best set of parameters. The above dictionary makes sure that all the different combinations of values are used to build Best CART models.

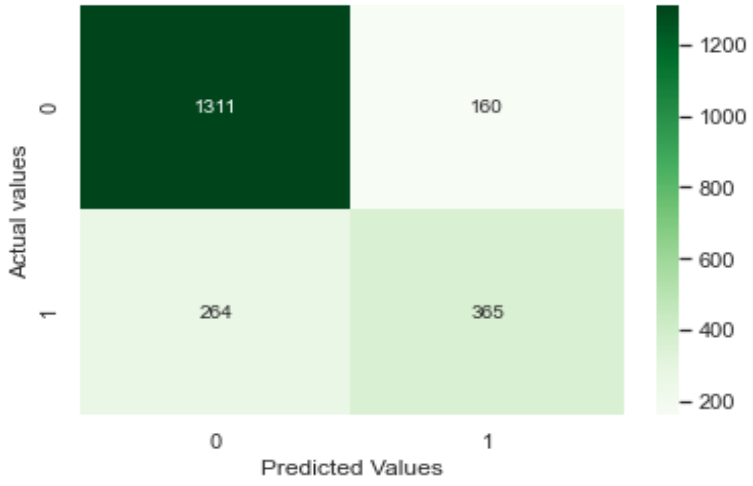
- Basis the inputs grid_search.best_params_ are

{'criterion': 'gini',
 'max_depth': 5,
 'min_samples_leaf': 25,
 'min_samples_split': 150}

Model Evaluation

- Lets import classification_report from sklearn.metrics.
- Let us first evaluate on the **training data**.
- We will start by checking the confusion matrix and then the classification report as well.
- Firstly, Train Accuracy for Cart Model is 0.7995238095238095.
- Confusion_matrix for train data.

  True Negative: 1311
  False Positives: 160
  False Negatives: 264
  True Positives: 365



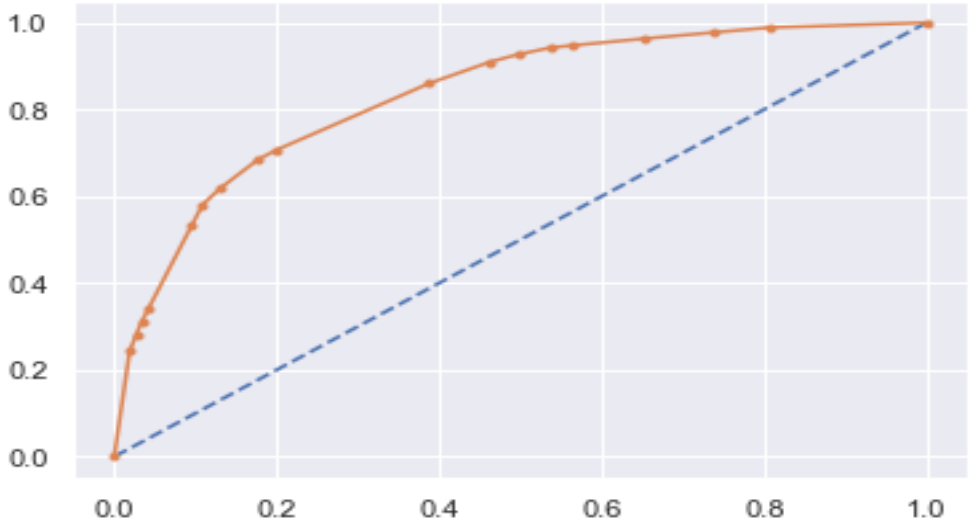
- Classification Report.

	precision	recall	f1-score	support
0	0.83	0.89	0.86	1471
1	0.70	0.58	0.63	629
accuracy			0.8	2100
macro avg	0.76	0.74	0.75	2100
weighted avg	0.79	0.80	0.79	2100

- We can see a 80% overall accuracy on the Training data.

- Now, let us check the Area Under Curve of the Receiver Operator Characteristic Curve. - AUC: 0.834

- AUC Curve



Test Model Evaluation

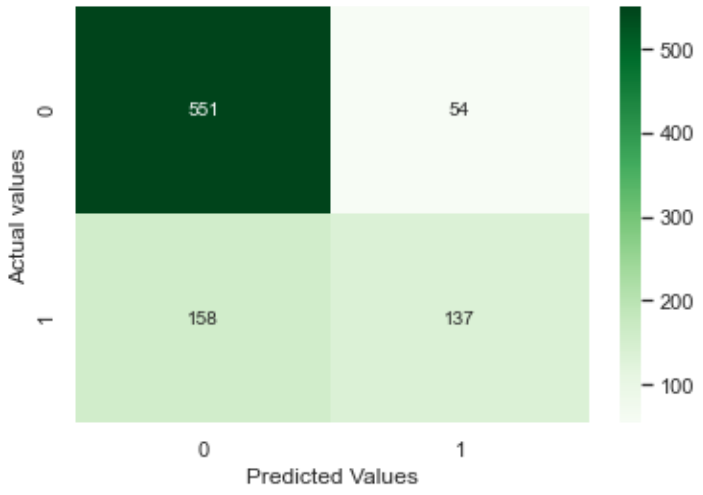
- Let us evaluate on the **test data**.

- We will start by checking the confusion matrix and then the classification report as well.

- Firstly, test Accuracy for Cart Model is 0.7755555555555556.

- Confusion_matrix for train data.

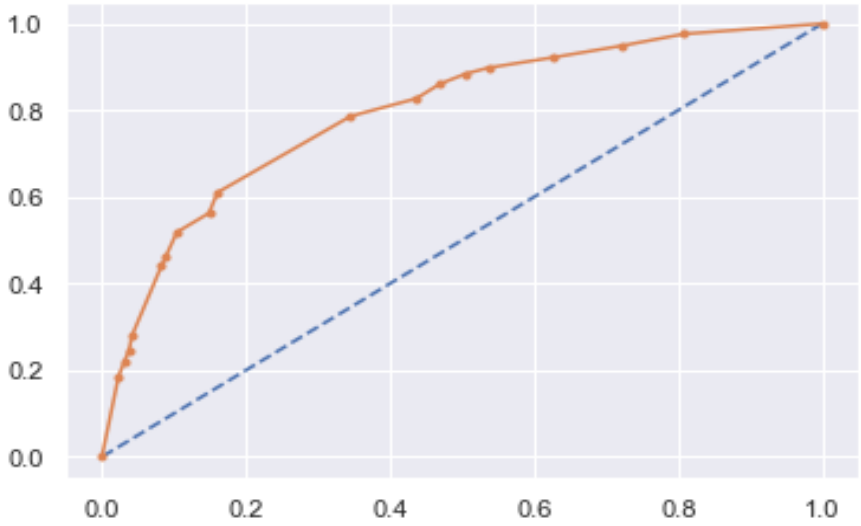
  True Negative: 551
  False Positives: 54
  False Negatives: 158
  True Positives: 137



- Classification Report.

	precision	recall	f1-score	support
0	0.78	0.91	0.86	605
1	0.72	0.46	0.56	295
accuracy			0.76	900
macro avg	0.75	0.69	0.70	900
weighted avg	0.76	0.76	0.75	900

- We can see a 78% overall accuracy on the Training data.
- Now, let us check the Area Under Curve of the Receiver Operator Characteristic Curve. - AUC: 0.796
- AUC Curve



Random Forest

Random forest is a classifier that evolves from decision trees. It actually consists of many decision trees. To classify a new instance, each decision tree provides a classification for input data; random forest collects the classifications and chooses the most voted prediction as the result. The input of each tree is sampled data from the original dataset. In addition, a subset of features is randomly selected from the optional features to grow the tree at each node. Each tree is grown without pruning. Essentially, random forest enables a large number of weak or weakly-correlated classifiers to form a strong classifier.

Building a basic Random Forest Model

- import RandomForestClassifier from sklearn.ensemble.
- Inputting the values for Basic RandomForestClassifier.

```
rfcl = RandomForestClassifier(criterion='gini',
                              random_state=1,
                                 oob_score=True,
                              max_features=4,
                               n_estimators=501)
```

- The Out of Bag score error score is 0.7542857142857143.
- The Out of Bag error rate is 24.571428571428566.
- Let us check importance of the variables in the Classification Tree. The importance of a feature or variable is computed as the (normalized) total reduction of the gini criterion brought by that feature. It is also known as the Gini importance.

Duration	0.254863
Sales	0.194513
Agency_Code_C2B	0.188265
Age	0.170710
Commision	0.096285
Product Name_Cancellation Plan	0.015495
Product Name_Customised Plan	0.012677
Product Name_Silver Plan	0.011399
Product Name_Bronze Plan	0.010276
Product Name_Gold Plan	0.009926
Destination_Americas	0.007818
Channel_Offline	0.007465
Destination_EUROPE	0.006234
Destination_ASIA	0.004914
Agency_Code_CWT	0.003748
Type_Travel Agency	0.002088
Type_Airlines	0.001915
Agency_Code_EPX	0.001408
Agency_Code_JZI	0.000000
Channel_Online	0.000000

- From the above output, we can see that 'Duration' is the most important variable followed by 'Sales' and so on.
- Let us take a look at the overall accuracy of the train and test data using the model that we just built.

Train Data	0.9948
Test Data	0.7578

- The accuracy on the Training Data is 99% and the accuracy on the Test Data is 75% which is lesser substantially. The model has surely been overfitted. We have allowed the Decision Trees inside the Random Forest algorithm to grow to their fullest. There is scope to regularize this particular Random Forest model to achieve comparative accuracy on both the Training and Test data.

Building Random Forest Using Grid Search CV

- GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.

- What the command GridSearchCV does is build multiple models based on a range of parameters specified by us. Then it ultimately returns best parameters on the basis of which we can go ahead and build our model. Let us now see how to define the range of parameters for the GridSearchCV function.

- Let's import GridSearchCV from sklearn.model_selection

- Passing the inputs through Grid Search CV.

```
param_grid = {
'max_depth': [5,7,8,9,10],
'max_features': [5,6,9,10],
'n_estimators': [301, 201,351,401,451,551]
}
rfcl = RandomForestClassifier(random_state=1)
grid_search = GridSearchCV(estimator = rfcl, param_grid = param_grid, cv = 3,n_jobs=-1)
#n_jobs=-1 allows paralle processing instead of serial processing
```

- Now that we have defined a Random Forest model let us build the model on the training data.

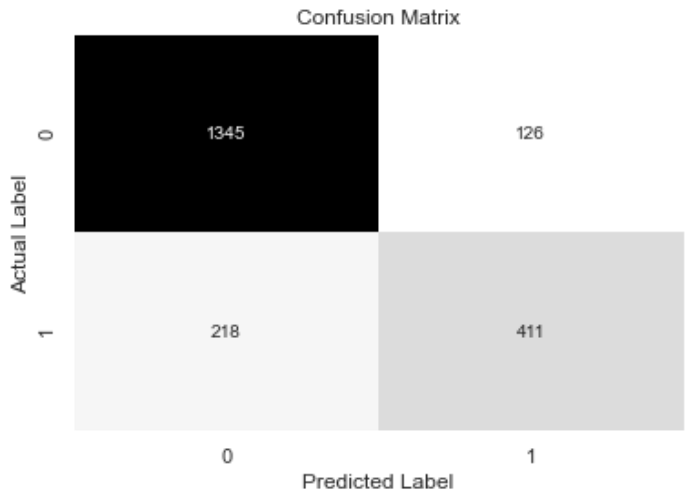
- Basis the inputs grid_search.best_params_ are

  {'max_depth': 7, 'max_features': 5, 'n_estimators': 401}

Model Evaluation

- Let's import classification_report from sklearn.metrics.
- Let us first evaluate on the training data.
- We will start by checking the confusion matrix and then the classification report as well.
- Firstly, Train Accuracy for Cart Model is 0.8361904761904762.
- Confusion_matrix for train data.

  True Negative: 1345
  False Positives: 126
  False Negatives: 218
  True Positives: 411

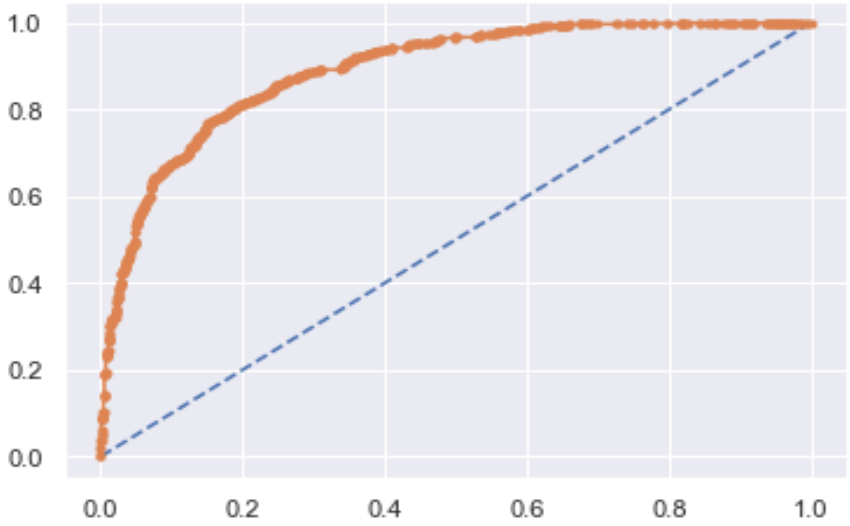


- Classification Report.

	precision	recall	f1-score	support
0	0.86	0.91	0.89	1471
1	0.77	0.65	0.70	629
accuracy			0.84	2100
macro avg	0.81	0.78	0.80	2100
weighted avg	0.83	0.84	0.83	2100

- We can see a 84% overall accuracy on the Training data.
- Now, let us check the Area Under Curve of the Receiver Operator Characteristic Curve. - AUC: 0.892

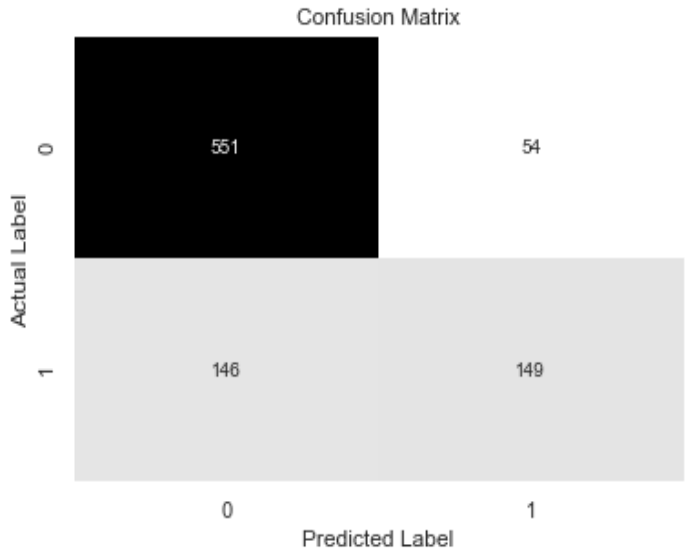
- AUC Curve



Test Model Evaluation

- Let us evaluate on the **test data**.
- We will start by checking the confusion matrix and then the classification report as well.
- Firstly, test Accuracy for Cart Model is 0.7777777777777778.
- Confusion_matrix for train data.

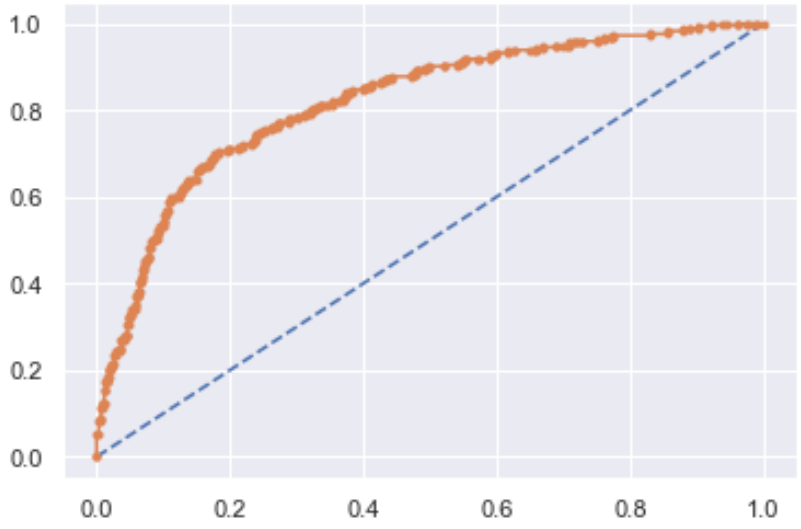
  True Negative: 551
  False Positives: 54
  False Negatives: 146
  True Positives: 149



- Classification Report.

	precision	recall	f1-score	support
0	0.79	0.91	0.85	605
1	0.73	0.51	0.60	295
accuracy			0.78	900
macro avg	0.76	0.71	0.72	900
weighted avg	0.77	0.78	0.757	900

- We can see a 78% overall accuracy on the Training data.
- Now, let us check the Area Under Curve of the Receiver Operator Characteristic Curve. - AUC: 0.82
- AUC Curve



MLP Classifier (Artificial Neural Network)

Artificial Neural networks (ANN) or neural networks are computational algorithms. It intended to simulate the behavior of biological systems composed of “neurons”. ANNs are computational models inspired by an animal’s central nervous systems. It is capable of machine learning as well as pattern recognition. These presented as systems of interconnected “neurons” which can compute values from inputs.

Build a Preliminary tree

- Before building the ANN model, we will go ahead and scale the data. Scaling the predictor variables helps in converging of the values. For computationally heavy algorithms, we can scale the data.
- Lets import StandardScaler from sklearn.preprocessing & scale the data.
- Here the Training & testing data is trained separately.
- Post scaling the data.

Building Artificial Neural Network Using Grid Search CV

- We will start by building a ANN Model and then we will go ahead and tune the various parameters of the ANN model.
- import MLPClassifier from sklearn.neural_network import MLPClassifier
- We are going to use the MLPClassifier function to build the ANN model. MLP stands for Multi Layer Perceptron.
- GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters.
- What the command GridSearchCV does is build multiple models based on a range of parameters specified by us. Then it ultimately returns best parameters on the basis of which we can go ahead and build our model. Let us now see how to define the range of parameters for the GridSearchCV function.

- Lets import GridSearchCV from sklearn.model_selection
- Passing the inputs through Grid Search CV.

```
param_grid = {
      'hidden_layer_sizes': [50,100,200],
   'activation': ['logistic', 'tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'learning_rate': ['adaptive','constant'],
    'tol': [0.01,0.001,0.0001],
    'max_iter' : [2500,3000,4000]
}
```

- Now that we have defined a dictionary in Python with all these parameters, it is time to use these parameters to build the model and check for the best set of parameters. The above dictionary makes sure that all the different combinations of values are used to build Best CART models.

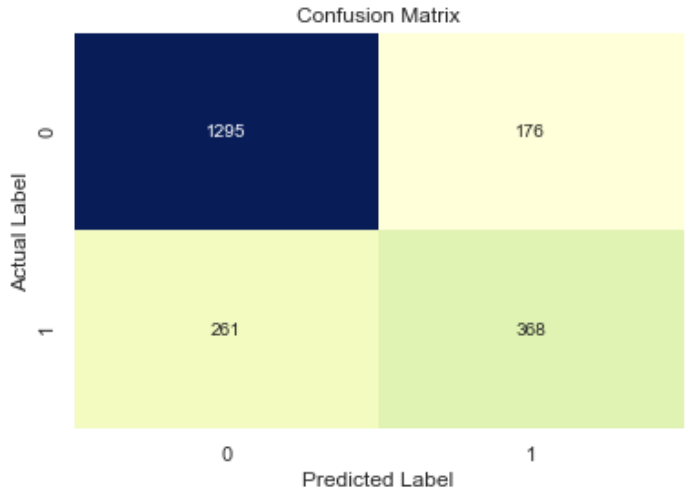
- Basis the inputs grid_search.best_params_ are

```
{'activation': 'relu',
 'hidden_layer_sizes': 200,
 'learning_rate': 'adaptive',
 'max_iter': 3000,
 'solver': 'adam',
 'tol': 0.001}
```

Model Evaluation

- Lets import classification_report from sklearn.metrics.
- Let us first evaluate on the **training data**.
- We will start by checking the confusion matrix and then the classification report as well.
- Firstly, Train Accuracy for Cart Model is 0.7919047619047619.
- Confusion_matrix for train data.

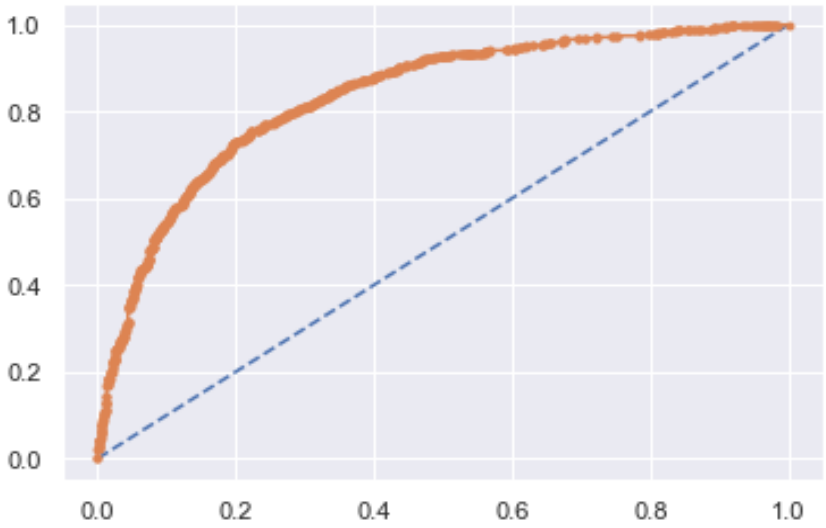
  True Negative: 1295
  False Positives: 176
  False Negatives: 261
  True Positives: 368



- Classification Report.

	precision	recall	f1-score	support
0	0.83	0.88	0.86	1471
1	0.68	0.59	0.63	629
accuracy			0.79	2100
macro avg	0.75	0.73	0.74	2100
weighted avg	0.79	0.79	0.79	2100

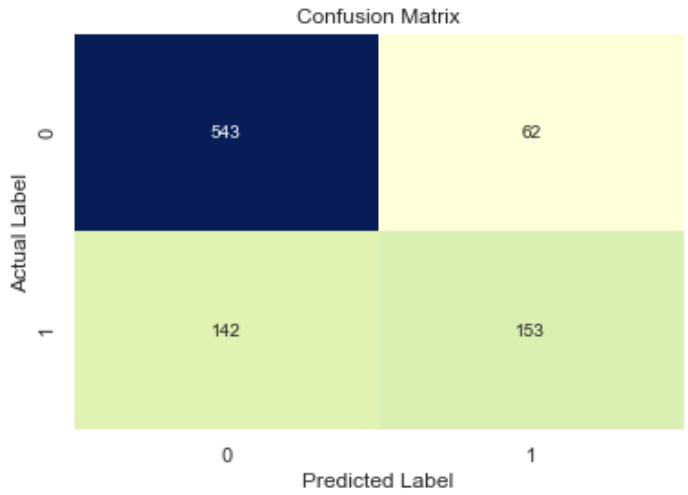
- We can see a 79% overall accuracy on the Training data.
- Now, let us check the Area Under Curve of the Receiver Operator Characteristic Curve. - AUC: 0.836
- AUC Curve



Test Model Evaluation

- Let us evaluate on the **test data**.
- We will start by checking the confusion matrix and then the classification report as well.
- Firstly, test Accuracy for Cart Model is 0.7733333333333333.
- Confusion_matrix for train data.

  True Negative: 543
  False Positives: 62
  False Negatives: 142
  True Positives: 153

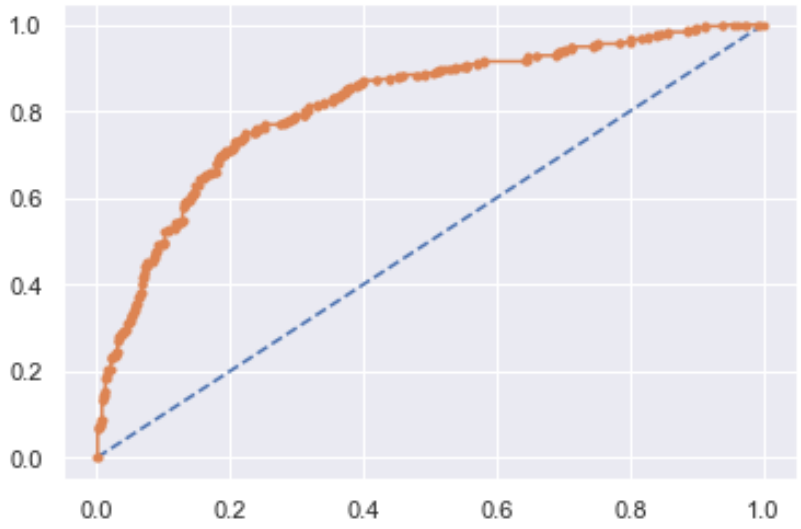


- Classification Report.

	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

- We can see a 77% overall accuracy on the Training data.
- Now, let us check the Area Under Curve of the Receiver Operator Characteristic Curve. - AUC: 0.815.

- AUC Curve



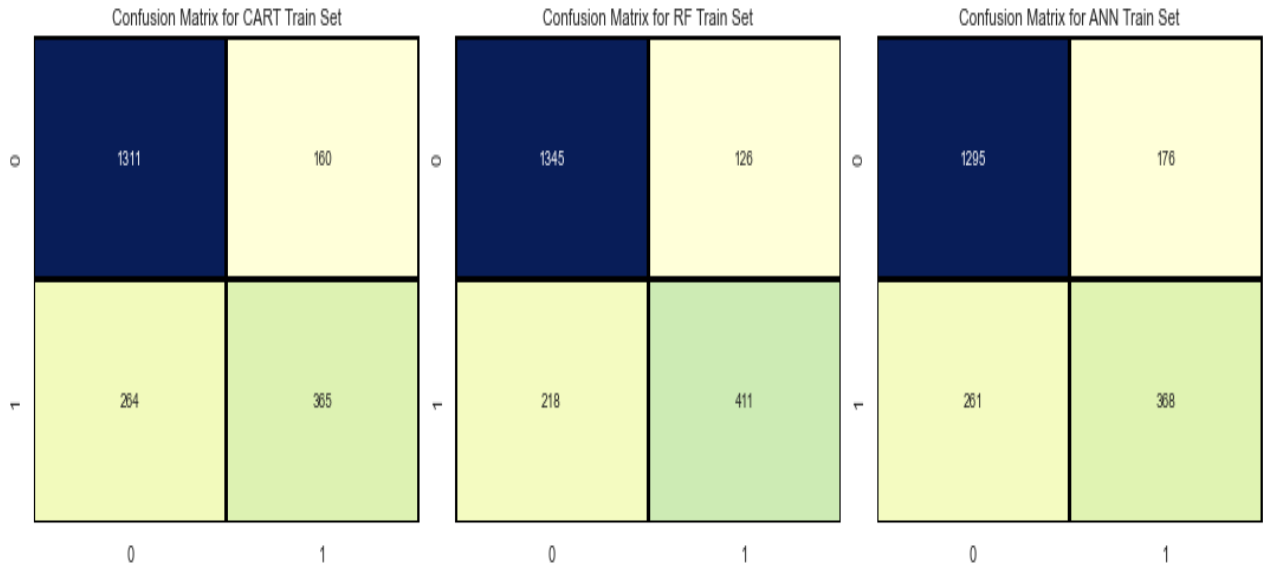
Model Comparison.

Comparing Accuracies from all the models for Train and Test Sets

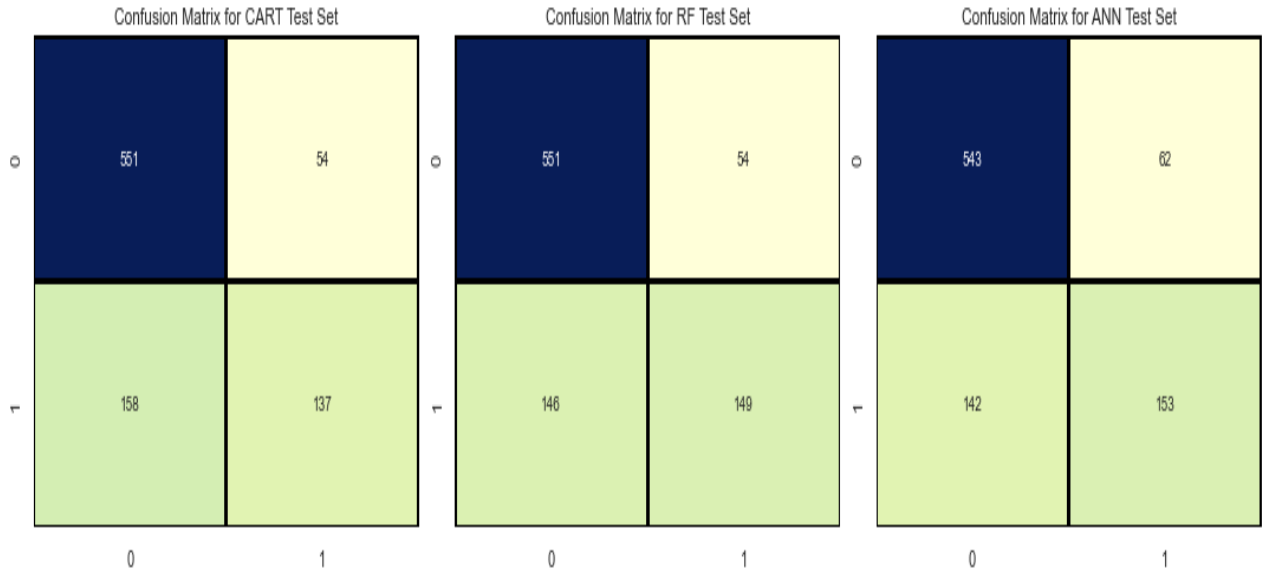
	CART	RF	ANN
Train-Accuracy	0.8	0.836	0.792
Test-Accuracy	0.776	0.778	0.773

Basis the above table we can infer that Random forest Model, has better accuracy, precision, recall & f1 score than other two Models (i.e. CART & Neural Network). Hence we can infer that Random Forest is the best model.

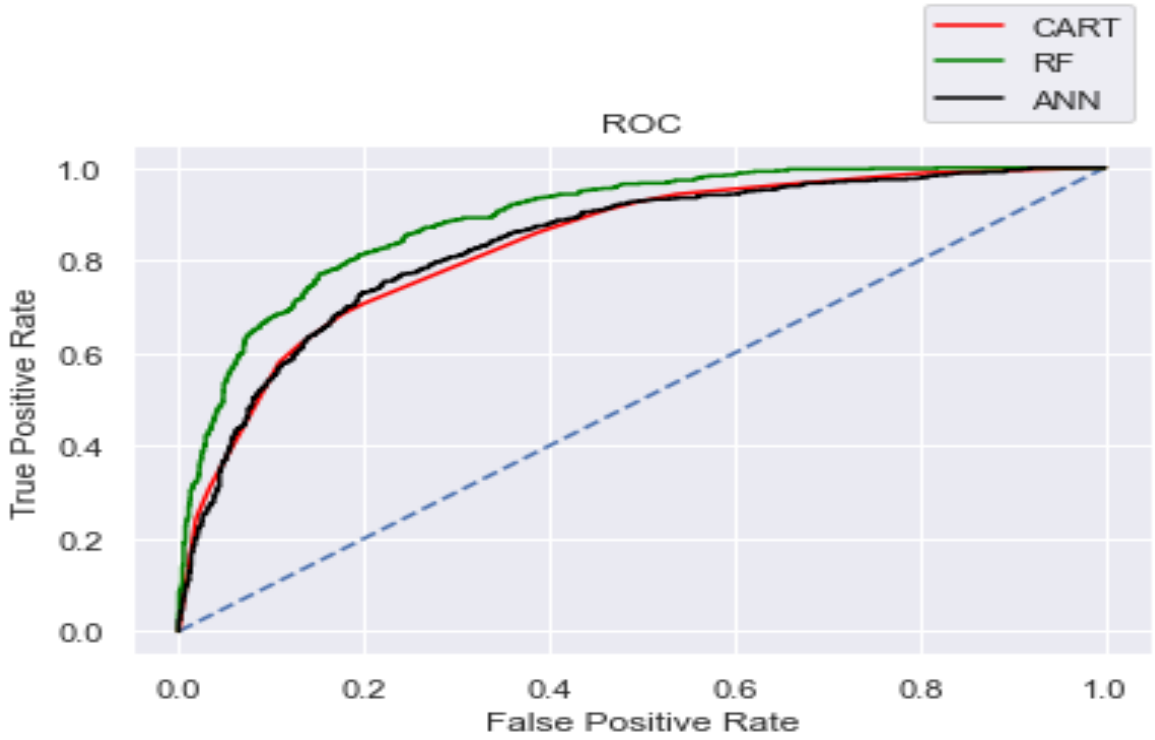
Comparing Confusion Matrices from All the models for the Train Set.



Comparing Confusion Matrices from All the models for the Test Set.



ROC Curve of all the models on the Training data



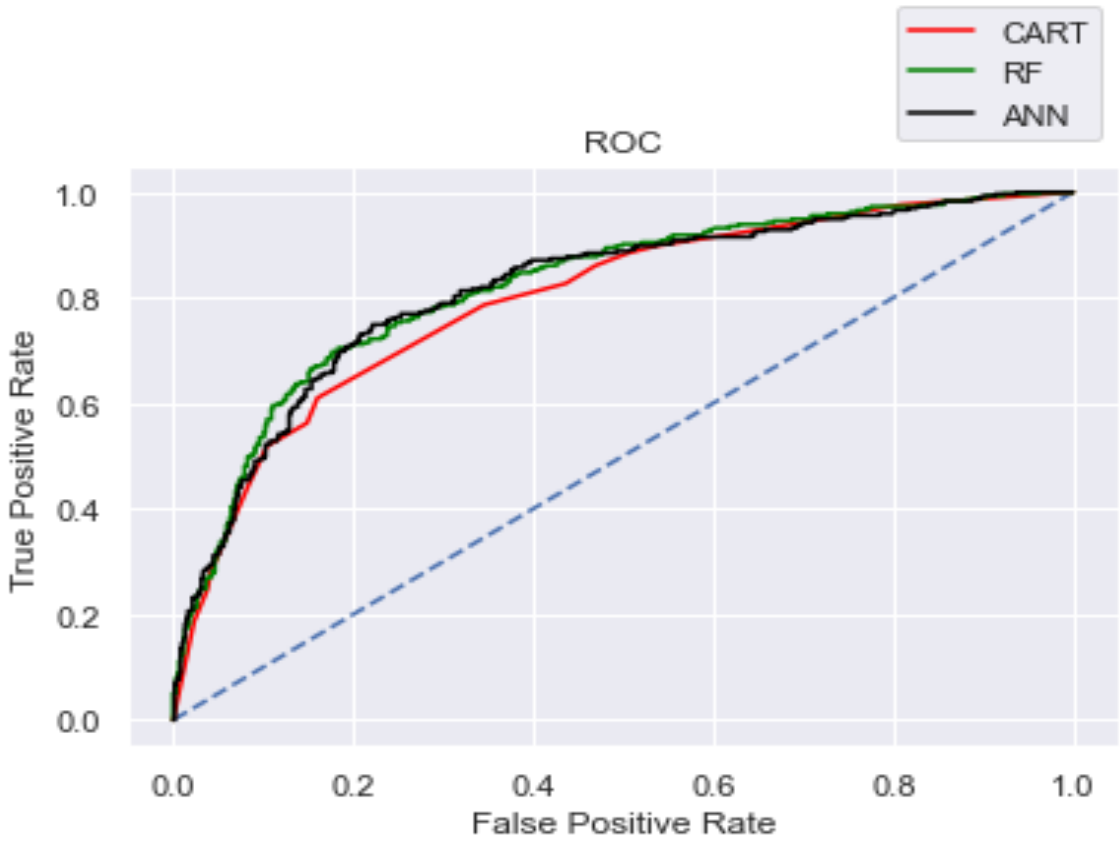
AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

Random forest has the highest AUC for train of all the models.

ROC Curve of all the models on the Testing data



AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

Random forest has the highest AUC for test of all the models.

Random forest has the highest AUC

Comparison of the performance metrics of all the models

	CART Train	CART Test	Random Forest Train	Random Forest Test	Neural Network Train	Neural Network Test
Accuracy	0.8	0.78	0.84	0.78	0.79	0.77
AUC	0.83	0.80	0.89	0.82	0.84	0.82
Recall	0.58	0.46	0.65	0.51	0.59	0.52
Precision	0.7	0.72	0.77	0.73	0.68	0.71
F1 Score	0.63	0.56	0.70	0.60	0.63	0.60

Based on above table we can infer that Random forest Model, has better accuracy, precision, recall & f1 score than other two Models (i.e. CART & Neural Network). The best model is Random Forest basis above table.

Recommendations

The insurance companies are tremendously interested in the prediction of the future. Accurate prediction gives a probability to decrease financial loss for the company. The insurers use rather complex methodologies for this purpose.

The major models are a decision tree, a random forest, artificial neural network a binary logistic regression, and a support vector machine. A great number of different variables are under analysis in this case.

The algorithms involve detection of relations between claims, implementation of high dimensionality to reach all the levels, detection of the missing observations, etc. In this way, the individual customer's portfolio is made.

Forecasting the upcoming claims helps to charge competitive premiums that are not too high and not too low. It also contributes to the improvement of the pricing models. This helps the insurance company to be one step ahead of its competitor.

Basis the dataset below points we noticed.

- Streamlining online experiences benefitted customers, leading to an increase in conversions, which subsequently raised profits.
- As per the data 90% of insurance is done by online channel.
- Basis data set it was identified that JZI has lowest sales.

The KPI's of insurance claims are:

• Reduce claims cycle time

• Increase customer satisfaction

• Combat fraud

• Optimize claims recovery

• Reduce claim handling costs Insights gained from data and AI-powered analytics could expand the boundaries of insurability, extend existing products, and give rise to new risk transfer solutions in areas like a non-damage business interruption and reputational damage.