IDM PROJECT – REPORT

NABEEL AHMED – 18024

SHAHMIR SHAHZAD – 17873

ABDULLAH AHMED – 18092

“Travel Insurance Claim Prediction”

Project Description

Travel insurance is a type of insurance that covers the costs and losses associated with travel. It is a useful protection for those traveling domestically or internationally. Many companies selling tickets or travel packages, give consumers the option to purchase travel insurance, also known as traveler's insurance. Insurance policies may cover damage to personal property, rented equipment, such as rental cars, or even the cost of paying a ransom.

Problem Statement

The insurance company has collected the data of previous travel insurance buyers. During the current vacations, **the company wants to know who will claim their travel insurance and who will not**. The company has chosen us to apply our Data Mining and Machine Learning knowledge and provide them with the most suitable model that achieves this goal.

Objective

We are responsible for building a machine learning model for the insurance company to predict if the insurance buyer will claim their travel insurance or not.

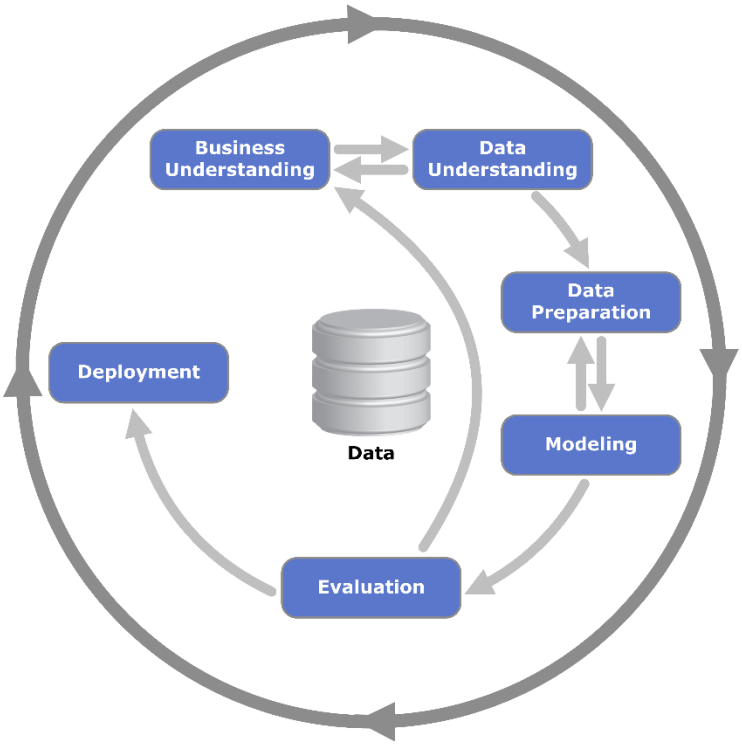
Description of Data

There are **11 columns** in the dataset. Some of them are mentioned below:

- Duration: Travel duration
- Destination: Travel destination
- Agency: Agency Name
- Commission: Commission on the insurance
- Age: Age of the insurance buyer
- Gender: Gender of the insurance buyer

- Agency Type: What is the type of the agency?
- Distribution Channel: offline/online
- Product Name: Name of the insurance plan
- Net Sales
- Claim: If the insurance is claimed or not (the target variable), 0 = not claimed, 1 = claimed

CRISP- DM MODEL



We will follow the Crisp-DM Model to understand the flow of our entire Project. The first thing which follows in Crisp-DM is Business Understanding.

Business Understanding

As already stated above that we were required to create a model which can predict based on certain attributes that whether a person **will claim Travel Insurance or Not.**

Data Understanding

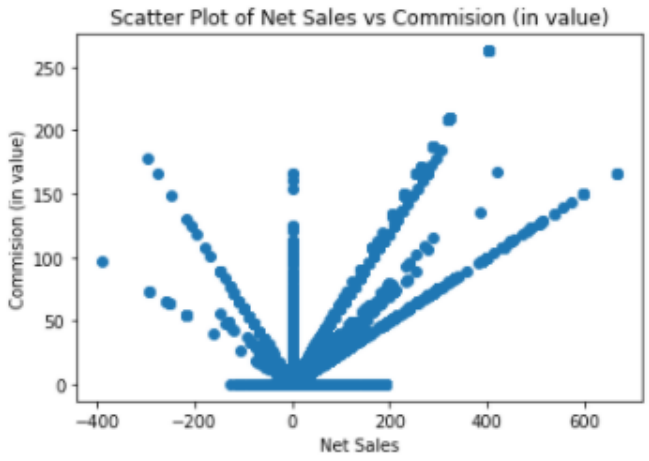
On inspecting the target Variables (Predictor), we can clearly see it is an **Imbalance Class Problem.**

Claim

0	47552
1	708

Then we tried to draw some relationships and understand based on some scatter plots which are described below:

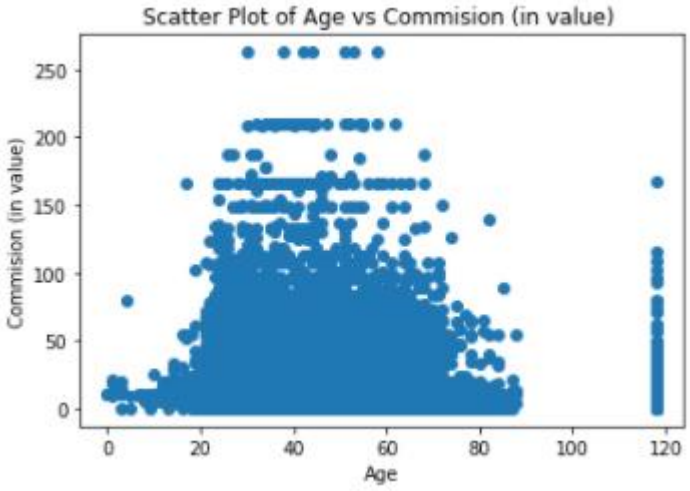
Scatter Plot of Net Sales vs Commission (in value)



On inspecting Net-Sales vs Commission it was found that there were some negative values for Net Sales which might indicate that Agency was going under loss.

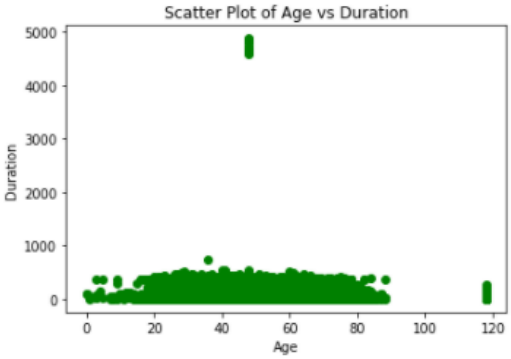
Commissions range from 0-250, while Net Sales range from 400 - 600.

Scatter Plot of Age vs Commission (in value)



From the Scatter plot of Commission and Age, we can see people mostly between 20-80 are getting Commission which ranges from 0-120(thousand). However, we can also see people whose age is around 120 are also getting commissions on insurance.

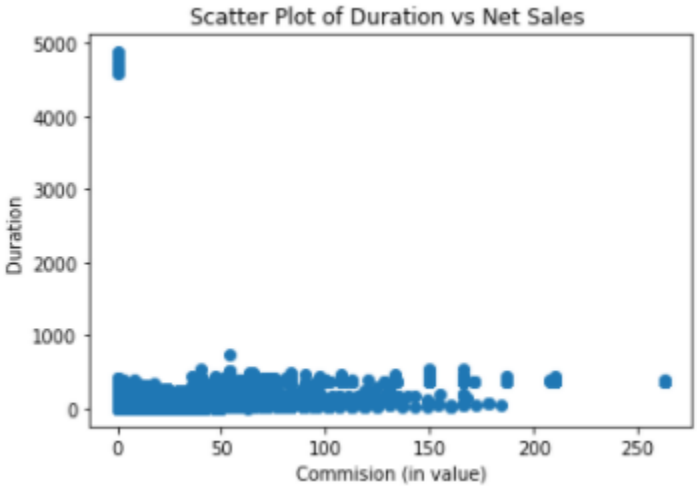
Scatter Plot of Age vs Duration



From the scatter plot of Age and Duration we can clearly see that between Age 20 - 80, mostly people have travel Duration in between 0 - 500 units.

- Some outliers are also seen as people with age about 120 have been observed to have some Travel Duration.
- Similarly, those who are about 50 have been observed to spend most time travelling, about 4700 - 5000 units.

Scatter Plot of Commission (in value) vs Duration

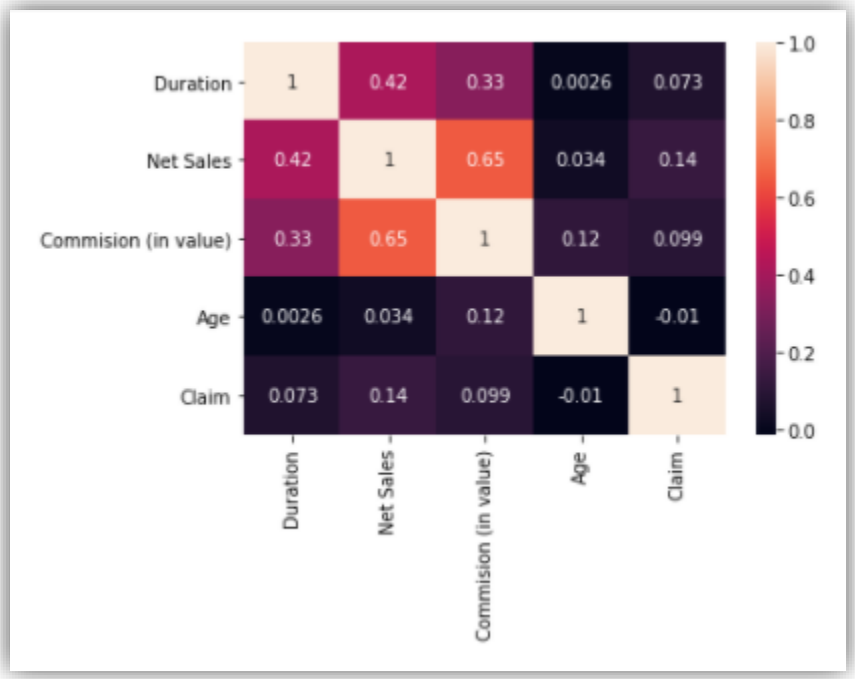


The relationship between Commission and Duration mostly remains constant. As seen in the previous plot, some people with ages around 50 have spent most of their time travelling but have got 0 commission on insurance.

More Insights from Data

- On further inspecting the dataset, it was observed that most people travel to Singapore, about 10068 people. The second most visited country was Thailand (4509) followed by Malaysia (4479).
- On inspecting the values in each column, it was also found that majority of the people didn't specified their Gender. There are about **34361** instances where Gender hasn't been specified. This therefore needs to be dealt with during Data Preparation.
- While inspecting Travel Insurance plan (**Product Name**) attribute, it was found that there are about 26 Insurance Plan being offered of which **"Travel Cruise Protect Family"** have just 1 instance, therefore it may be safe to remove this record from our dataset during **Data preparation** as we do not have enough data about this Travel Insurance plan.
- Moreover, In Duration there are 3 instances where we can see negative values which doesn't make sense as duration can't be negative. Therefore, it must also be dealt while Data Preparation.

- Finally, understanding and inspecting Correlation between different variables, it was observed that there was not any highly corelated variables. Below is the heatmap of Correlation:



Data Preparation

Now we have seen what our Data is like and what steps needed to be taken while Data Preparation. Let's move ahead with our Crisp-DM Cycle and prepare the data based on the insights we have gathered.

```
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Agency                48260 non-null  object
 1   Agency Type           48260 non-null  object
 2   Distribution Channel  48260 non-null  object
 3   Product Name          48260 non-null  object
 4   Duration              48260 non-null  int64
 5   Destination           48260 non-null  object
 6   Net Sales             48260 non-null  float64
 7   Commision (in value)  48260 non-null  float64
 8   Gender                13899 non-null  object
 9   Age                   48260 non-null  int64
 10  Claim                 48260 non-null  int64
```

From the description of the variables/attributes of our Dataset we can clearly see that leaving target variable(claim) aside we have **4 numerical variables** and **6 categorical variables** which needs to be encoded before we process into our model. But before encoding categorical variables, let's first handle missing values.

As discovered in data preparation, that there was only one column (Gender) that had missing values as many people didn't specified their gender in Travel insurance form, so we decided to deal with that first. There can be several ways to deal with missing values in this case. The simplest being dropping all NA records, but since this would result in great deal of Data loss as about **34361 records are with missing Gender values**, we decided to add another category of "Not specified" replacing NA in order to preserve maximum records.

After filling NA values in **Gender** column with **"Not Specified"** below is the table which proportion of each category.

Not Specified	34361
M	7137
F	6762

- Then, we handled negative duration values by replacing it with mean of Duration column. After handling this we move on to deal with that Insurance plan which had just one

occurrence as discovered from the findings while Data Understanding. So, we removed that single record and now we were left with just 25 different Travel Insurance Plans.

- Lastly based on some human insights and keeping the objective in mind we decided to drop 'Distribution Channel', 'Destination', 'Agency Type' from our data as they seemed less relevant with the task at hand. Below is the summary of the columns we will deal with while modelling.

```
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Agency                48259 non-null  object
 1   Product Name          48259 non-null  object
 2   Duration              48259 non-null  int64
 3   Net Sales             48259 non-null  float64
 4   Commision (in value)  48259 non-null  float64
 5   Gender                48259 non-null  object
 6   Age                   48259 non-null  int64
 7   Claim                 48259 non-null  int64
```

Handling Categorical Variable

Finally, after successfully dealing with all messy data, now it was time to handle **categorical variables** as we can't directly feed these variables to our model.

There are multiple strategies to deal with categorical attributes. The one which we chose was to make dummy variables and to one hot encode it.

We encode the categorical variables by making a function and using **get_dummies** method of pandas. This returns processed Data Frame which has encoded(one-hot) all categorical variables.

```
Data columns (total 48 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Duration                                    48259 non-null  int64
 1   Net Sales                                   48259 non-null  float64
 2   Commision (in value)                        48259 non-null  float64
 3   Age                                         48259 non-null  int64
 4   Agency_ADM                                  48259 non-null  uint8
 5   Agency_ART                                  48259 non-null  uint8
 6   Agency_C2B                                  48259 non-null  uint8
 7   Agency_CBH                                  48259 non-null  uint8
 8   Agency_CCR                                  48259 non-null  uint8
 9   Agency_CSR                                  48259 non-null  uint8
 10  Agency_CWT                                  48259 non-null  uint8
 11  Agency_EPX                                  48259 non-null  uint8
 12  Agency_JWT                                  48259 non-null  uint8
 13  Agency_JZI                                  48259 non-null  uint8
 14  Agency_KML                                  48259 non-null  uint8
 15  Agency_LWC                                  48259 non-null  uint8
 16  Agency_RAB                                  48259 non-null  uint8
 17  Agency_SSI                                  48259 non-null  uint8
 18  Agency_TST                                  48259 non-null  uint8
 19  Agency_TTW                                  48259 non-null  uint8
 20  Product Name_1 way Comprehensive Plan       48259 non-null  uint8
 21  Product Name_2 way Comprehensive Plan       48259 non-null  uint8
 22  Product Name_24 Protect                     48259 non-null  uint8
 23  Product Name_Annual Gold Plan               48259 non-null  uint8
 24  Product Name_Annual Silver Plan             48259 non-null  uint8
 25  Product Name_Annual Travel Protect Gold     48259 non-null  uint8
 26  Product Name_Annual Travel Protect Platinum 48259 non-null  uint8
 27  Product Name_Annual Travel Protect Silver   48259 non-null  uint8
 28  Product Name_Basic Plan                     48259 non-null  uint8
 29  Product Name_Bronze Plan                    48259 non-null  uint8
 30  Product Name_Cancellation Plan              48259 non-null  uint8
 31  Product Name_Child Comprehensive Plan       48259 non-null  uint8
 32  Product Name_Comprehensive Plan             48259 non-null  uint8
 33  Product Name_Gold Plan                      48259 non-null  uint8
 34  Product Name_Individual Comprehensive Plan  48259 non-null  uint8
 35  Product Name_Premier Plan                   48259 non-null  uint8
 36  Product Name_Rental Vehicle Excess Insurance 48259 non-null uint8
 37  Product Name_Silver Plan                    48259 non-null  uint8
 38  Product Name_Single Trip Travel Protect Gold 48259 non-null uint8
 39  Product Name_Single Trip Travel Protect Platinum 48259 non-null uint8
 40  Product Name_Single Trip Travel Protect Silver 48259 non-null uint8
 41  Product Name_Spouse or Parents Comprehensive Plan 48259 non-null uint8
 42  Product Name_Ticket Protector               48259 non-null  uint8
 43  Product Name_Travel Cruise Protect          48259 non-null  uint8
 44  Product Name_Value Plan                     48259 non-null  uint8
 45  Gender_F                                    48259 non-null  uint8
 46  Gender_M                                    48259 non-null  uint8
 47  Gender_Not Specified                        48259 non-null  uint8
```

We now had **48 attributes including target variable (Claim).**

- After handling categorical variables, we separated dependent and independent variables.

```
X = processed_df.drop(['Claim',], axis=1)
Y = processed_df['Claim']
```

- The shape of independent and dependent variables is as the following:

  X.shape = (48259, 48)

  Y.shape = (48259,)

Imbalance Class Problem

As we have seen that like any other real-life datasets, we have the problem of imbalance class problem, it was no different with this dataset as well, which needs our immediate attention before we proceed on to Modelling Step. This is an essential step and a very important one as well otherwise our model will be of no good than a random guess. Since the majority of the records have been of those who didn't claim **Travel Insurance** and only few people did. So, despite of our cleaning data our final model would still be giving most predictions as **Not Claim** as it was given majority this type of Data.

So, in order to deal with this biasness, we make use of one of the sampling strategies i.e., **oversampling** less frequent records to the most frequent one. This would eliminate our model from being bias towards just a single type of category.

Before oversampling we need to partition our Dataset into train and test as it only makes sense to oversample our training records not testing records.

For partitioning, we split our dataset into 80-20%. After splitting, we oversample our train dataset in order to have the same records for both classes in our **target variable.**

Now we are all set for passing our processed Data from different models noting their accuracies and evaluating which model would best be suited for this purpose.

Modelling

<u>INITIAL WORKING ON THE DATA SET (BEFORE DATA PREPARATION) TO SEE WHERE WE STAND AND WHAT WE NEED TO DO.</u>

1. Data Preparation: For the initial run, we needed to convert the categorical data into binary variables in order for our models to process the model. For that we created **dummy variables** using pandas.

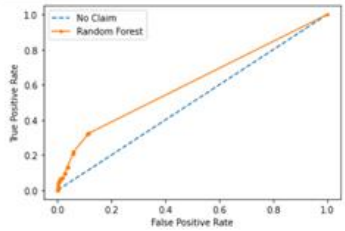
2. Initial Modelling:

a. **Random Forest:** We applied Random Forest as an initial algorithm to get started with our hunt for an optimized and ideal classification model with any enhancements that we do in our data.

   i. Accuracy: The accuracy, as expected was turning out to be a good number that is **98.1%** without any significant work done on the data set since it already is an **imbalance** class problem.

   ii. ROC Curve: However, to get a real insight into our model's performance, we created the ROC curve with the **probabilities** and **predictions** we got from this model.

```
In [51]: # plot the roc curve for the model
         pyplot.plot(default_fpr, default_tpr, linestyle='--', label='No Claim')
         pyplot.plot(rf_fpr, rf_tpr, marker='.', label='Random Forest')
         # axis labels
         pyplot.xlabel('False Positive Rate')
         pyplot.ylabel('True Positive Rate')
         # show the legend
         pyplot.legend()
         # show the plot
         pyplot.show()
```



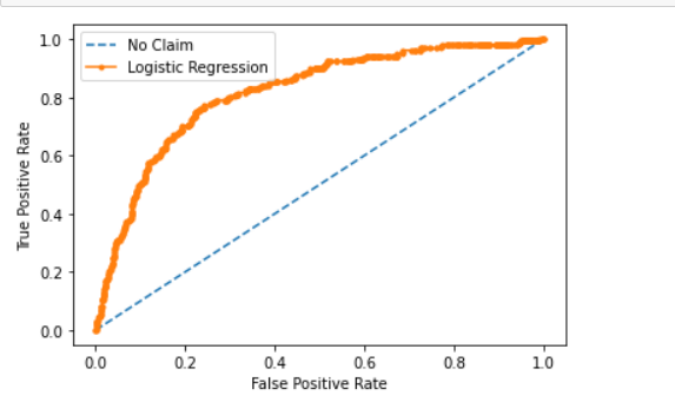
   iii. Accuracy of ROC Curve with Random Forest: **0.6**

This ROC curve gives us an actual idea of our performance with the Random Forest Classifier. We will give a try with some other classification algorithms as well and then hop back to the **Data Preparation** step and see if any making any changes in the data helps us in any way.

b. **Logistic Regression:**

   i. Accuracy: Logistic Regression gives us an accuracy of **98.4** % on the same dataset.

   ii. ROC Curve: An **improvement** in performance can be seen with logistic regression as you can see the curve moving up towards the (0,1) on the graph.



iii. The accuracy of the ROC curve climbed to **0.817**.

c. **Multinomial NB:** Use of this algorithm identifies a **problem** in our dataset that is the **presence of negative values** in the dataset in 'Duration' column.

```
D:\Anaconda\lib\site-packages\sklearn\utils\validation.py in check_non_negative(X, whom)
   1045
   1046     if X_min < 0:
-> 1047         raise ValueError("Negative values in data passed to %s" % whom)
   1048
   1049

ValueError: Negative values in data passed to MultinomialNB (input X)
```

When we think, logically too it is not possible for a duration to be in negative number, hence we feel a need to deal with these negative values as well. A hindrance was caused due to the current condition of our dataset.

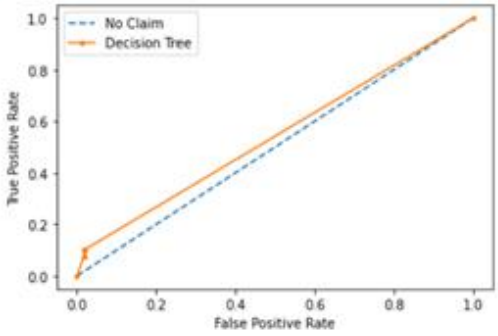
d. **Decision Trees:**

i. Accuracy: In terms of accuracy, it achieved **97%**, which is better than that of Random Forest.

ii. ROC Curve: But as we know that this accuracy could be deceiving, we can see that **despite greater accuracy, decision trees performance degrades here in terms of ROC when compared to Random Forest.**

```
In [64]: dt_fpr, dt_tpr, _ = roc_curve(y_test, dt_probs)

         # plot the roc curve for the Decision Tree Classifier
         pyplot.plot(default_fpr, default_tpr, linestyle='--', label='No Claim')
         pyplot.plot(dt_fpr, dt_tpr, marker='.', label='Decision Tree')
         # axis labels
         pyplot.xlabel('False Positive Rate')
         pyplot.ylabel('True Positive Rate')
         # show the legend
         pyplot.legend()
         # show the plot
         pyplot.show()
```

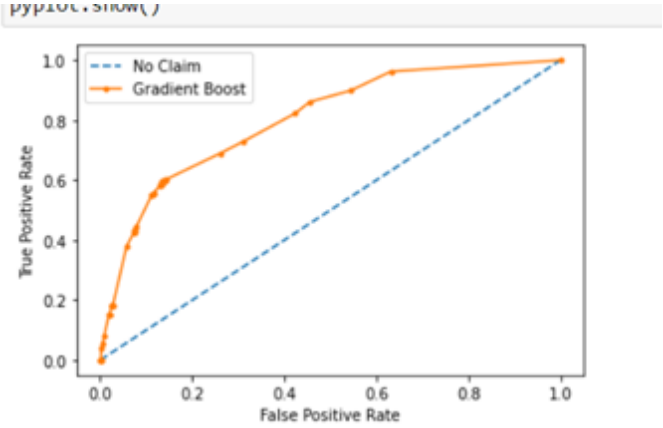


iii. ROC Accuracy: 0.541.

e. Gradient Boost:

i. Accuracy: results in 98.3% accuracy (almost similar to Logistic Regression).

ii. ROC Curve: Unlike Decision Trees or Random Forest, Gradient boost performs well in handling this imbalance class problem which is evident from its ROC curve.



iii. ROC Accuracy: **0.801.**

We need to work on our dataset before finalizing a model since there are several things in our data which are hindering the training and testing of our models. Hence, we deal with the challenges of preparing our data and make it clean before selecting the most appropriate model.

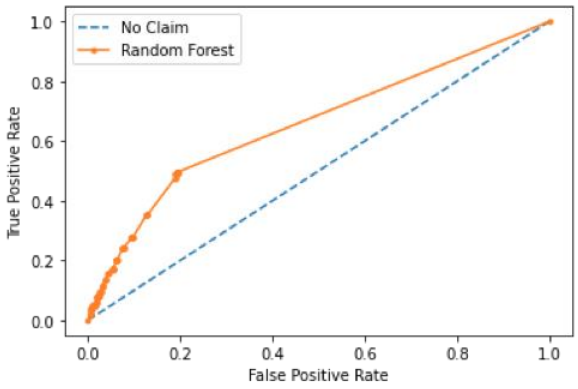
<u>WORKING ON THE DATA SET (AFTER DATA PREPARATION) TO SEE WHERE WE STAND AND WHAT WE NEED TO DO.</u>

Now after data preparation, we applied the same algorithms again to see how has data preparation changed the performance of these algorithms. We have kept all the parameters the same so that we can compare the effects of data preparation with fairness. The results of data preparation were as follows:

f. Random Forest:

i. Accuracy: The accuracy decreased from **98.1% to 96.25%.**

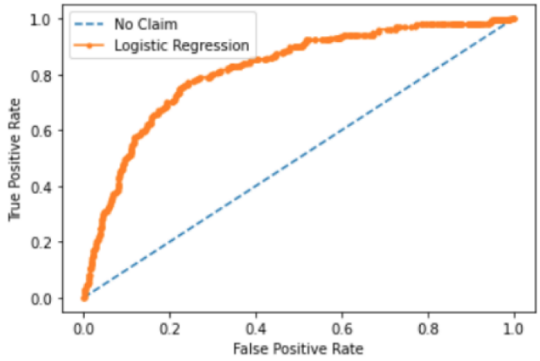
ii. The accuracy of the ROC Curve with Random Forest was **0.656,** which shows an increase (originally 0.6)**.**



g. Logistic Regression:

i. Accuracy: The accuracy of Logistic Regression decreased from **98.4%** to **92.14%.**

ii. The accuracy of the ROC curve also dropped from **0.817 to 0.767**.



h. Multinomial NB: Use of this algorithm identifies a **problem** in our dataset that is the **presence of negative values** in the dataset in 'Duration' column.

When we think, logically too, it is not possible for a duration to be in negative number, hence we feel a need to deal with these negative values as well. A hindrance was caused due to the current condition of our dataset.

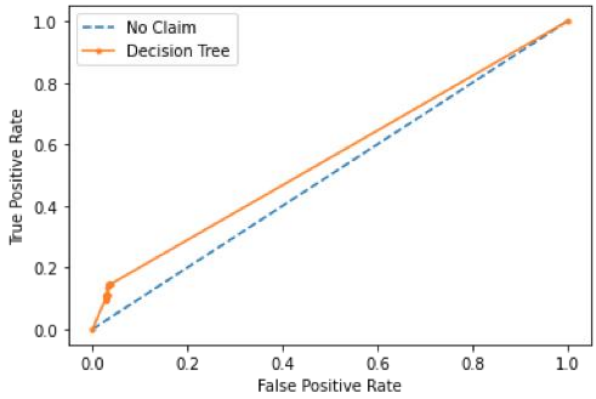
```
D:\Anaconda\lib\site-packages\sklearn\utils\validation.py in check_non_negative(X, whom)
   1045
   1046     if X_min < 0:
-> 1047         raise ValueError("Negative values in data passed to %s" % whom)
   1048
   1049

ValueError: Negative values in data passed to MultinomialNB (input X)
```

i. Decision Trees:

i. Accuracy: In terms of accuracy, it achieved **95.6%**, which is again a drop (originally 97%).

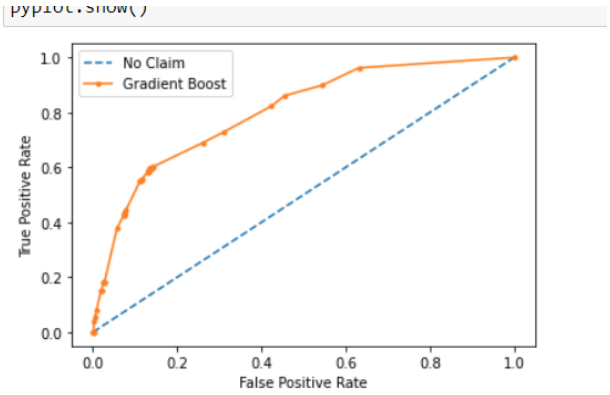
ii. ROC Accuracy increased from **0.541** to **0.554**.



j. Gradient Boost:

i. Accuracy: In terms of accuracy, it achieved 89.5%, which is again a drop (originally 98.3%).

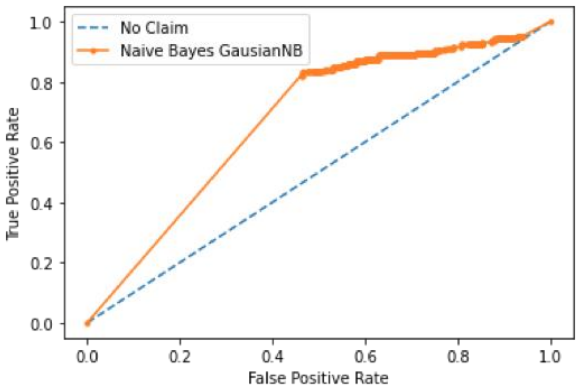
ii. ROC Accuracy: **0.779,** which is a decrease.



f. Gaussian NB:

i) Accuracy: Results in **24.3%** accuracy.

ii) ROC Accuracy: **0.675**

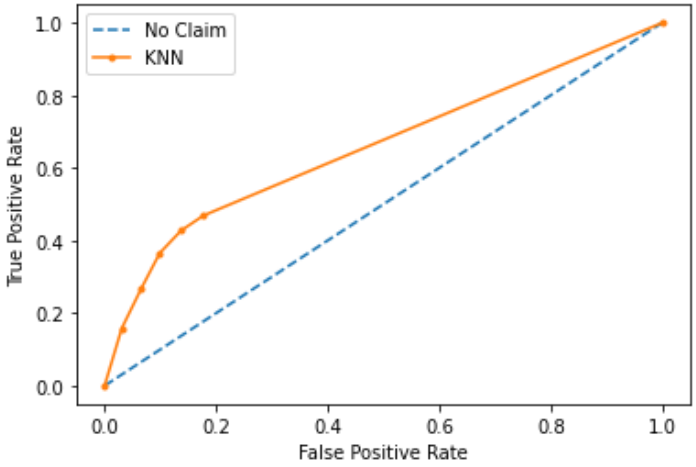


g. K-Nearest Neighbor:

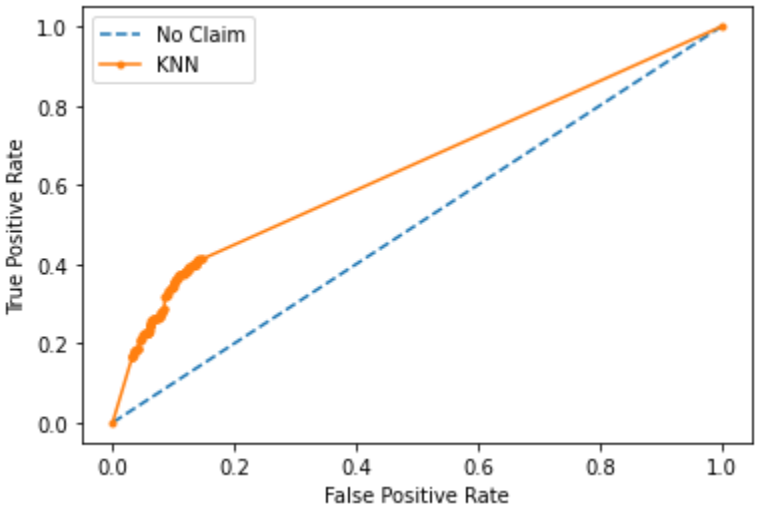
Using **k=5**

i) Accuracy: Results in **89.29%** accuracy.

ii) ROC Accuracy: **0.655.**

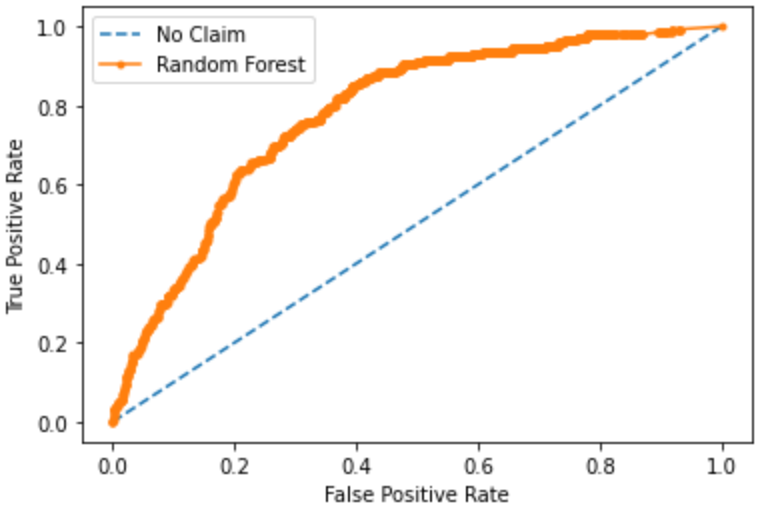


We also tried using weighted KNN. The accuracy moved to **90.57%** and ROC accuracy moved to **0.641.**

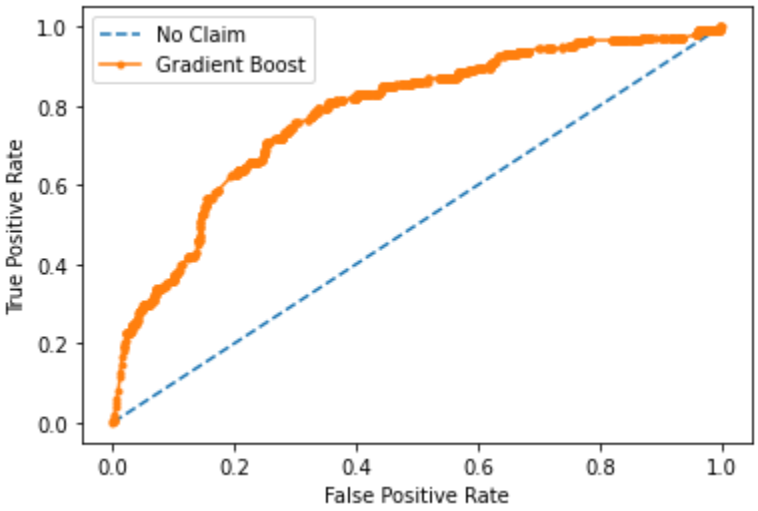


3. Evaluation: So far, we can see that upon data preparation, the ROC accuracy has dropped. Even though the accuracy has dropped, we still needed to clean the data for better understanding and better analysis. Due to data preparation, we had more accurate results. From the models that we were able to test so far, after data preparation, we can see that **Logistic Regression, Gradient boost and Random forest have the highest ROC accuracy.** From now on, we used these three models and tried to optimize each of the three. Our focus was to obtain the best possible model.

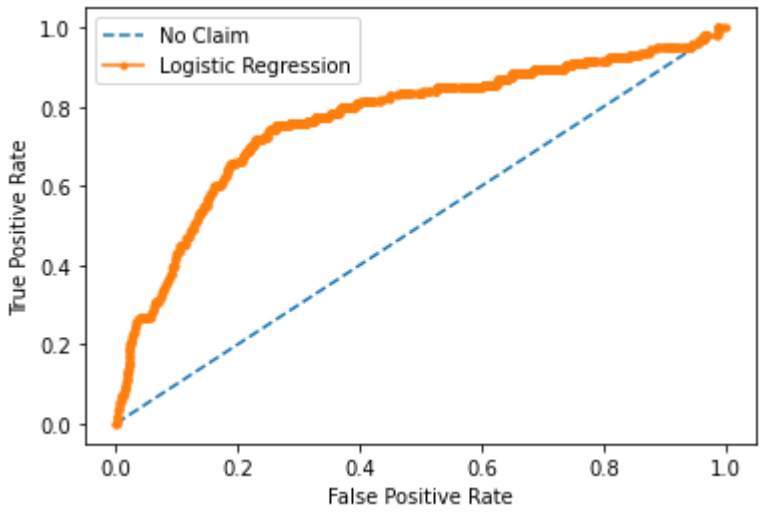
Firstly, we tried to optimize the Random Forest Model. We tried using different number of trees and also tried using different tree depths. **We concluded the best random forest model is the one with 50 trees and a max depth of 20. With this, the ROC accuracy was coming out to be 0.78.**



Next, we tried to optimize the Gradient Boost Model. We tried using different n estimators, learning rate and max depths. **We concluded the best Gradient Boost model is the one with n_estimator=100, max depth of 1 and learning rate of 1.0. With this, the ROC accuracy was coming out to be 0.779.**



Lastly, we tried to optimize the Logistic Regression Model. The changes did not have that much effect on accuracy. **ROC accuracy remained the same as before i.e. 0.767.**



Since Random Forest had the most accuracy, we chose **Random forest as the winning model.**

Evaluation

Initial Modelling: From the models that we have been able to test so far, Logistic Regression seems to be the most suitable one due to its efficiency of ROC curve which gives us an idea that it can be a decent choice of algorithm for imbalance class problem.

Final Modelling (with refined data):

- Though we observed a decrease in accuracy of the models after working on data (for e.g Random Forest's accuracy dropped to 96.25% from 98.1%, our effort on data preparation resulted in an **increase in ROC accuracy** in some cases as we can see the case Random Forest again.
- Incorporating all the steps we performed, **Gradient boost** emerges as the most suitable algorithm with an ROC curve accuracy of **0.779**.
- Generally, a mixed trend of accuracies increasing and decreasing after applying various data preparation techniques at various points can be seen in our experiments. Along with

the primary objective of classifying as accurately as possible, it is more convenient to deal and work on a processed form of data. Hence, we have decided that **Random Forest with 50 trees and a max depth of 20,** and accepting the changes made to data would be the **best combination** to proceed with as a result of this project.

Findings

Limitations: Though we preferred on using/giving cleaned data to our finalized model, but as a consequence of that due to some reasons or probably because of the nature of the data, performance of some models have deteriorated than what it was on uncleaned data.

Moreover, the accuracy of our evaluation metric, that is ROC curve, seems to have no significant leap from this point as if it has become saturated. There might be other possible hyperparameter tunings of these algorithms that could achieve a better accuracy as well, but the ones that we used are decent enough to be performed on even a personal system in a reasonable time without much difference in accuracy.

Advice to this travel agency: Record more details about the customer. Add more attributes when collecting data, for example **Marital Status, Financial Status or Occupation** of the customers could be helpful too in deriving conclusions about factors impacting results.

The agency can make sure they note complete information of every customer (A lot of missing data which could be important).

Considerations for Deployment

- For deploying this model we must use same predictor variables which we have used for training in order to get accurate results.
- We should only allow user to choose an existing insurance plan not a new one as our model is only trained for past Insurance Plans, so might perform poorly on any unseen data.
- Similarly, since we might have different preferences for traveling every year and the attributes of people might change and many new Travel Insurance plans might be introduced over the year, hence it is advisable to retrain the model after a year.

- Therefore, expected expiry date of the model is 1 year. And then the entire model-pipeline might have to be repeated with Crisp-DM Cycle.

Python Notebooks Link

Data Understanding & Preparation: Click on the link here by Nabeel Ahmed.

Initial Model Evaluation on Original Dataset: Click on the link here by Shahmir Shahzad.

Model Evaluation on Refined Dataset: Click on the links Naive Bayes , Logistic Regression , Gradient Boosting , Decision Tree , Random Forest by Abdullah Ahmed.