Insurance Fraud Detection

-Rijul Kumar

Insurance fraud is a deliberate deception perpetrated against or by an insurance company or agent for financial gain. Fraud may be committed at different points in the insurance transaction by applicants for insurance, policyholders, third-party claimants or professionals who provide services to claimants.

Problem Definition

Insurance fraud is a huge problem in the industry. It's difficult to identify fraud claims. Machine Learning is in a unique position to help the Auto Insurance industry with this problem.

In this project, we have a dataset which has the details of the insurance policy along with the customer details. It also has the details of the accident on the basis of which the claims have been made.

In this project, we will be working with some auto insurance data to demonstrate how we can create a predictive model that predicts if an insurance claim is fraudulent or not.

Some facts related to insurance frauds are :

- Insurance fraud steals at least $308.6 billion every year from American consumers. (Coalition Against Insurance Fraud is working to update this figure in 2022).
- Fraud occurs in about 10% of property-casualty insurance losses.
- Medicare fraud is estimated to cost $60 billion every year. (AARP 2018)
- US insurance fraud stats reveal that, in 2020, a form of fraud was present in 18% of insurance claims.

Some facts related to insurance fraud prevention are :

- The biennial study by the Coalition Against Insurance Fraud and SAS reviews how insurers use anti-fraud technology, their fraud detection strategies, and future plans to expand technology in this area. The 2021 study found that 80 percent of respondents currently use predictive modeling to detect fraud, up from 55 percent in 2018
- The insurance industry counts 7,000 organizations in the US alone. Roughly 85% of insurers have dedicated investigation teams.

- Key concerns of anti-fraud leaders include staffing and talent management, legal/regulatory environment and oversight, managing emerging fraud trends, and organization technology challenges

Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

EDA steps taken in project:

1. Getting insights :
    - used .describe() to get statistical summary of the dataframe

```
In [15]: #Statistical summary
         df.describe()
```

Out[15]:

	months_as_customer	age	policy_number	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains	capital-loss	incident_hou
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.000000	1000.000000	
mean	203.954000	38.948000	546238.648000	1136.000000	1256.406150	1.101000e+06	501214.488000	25126.100000	-26793.700000	
std	115.113174	9.140287	257063.005276	611.864673	244.167395	2.297407e+06	71701.610941	27872.187708	28104.096686	
min	0.000000	19.000000	100804.000000	500.000000	433.330000	-1.000000e+06	430104.000000	0.000000	-111100.000000	
25%	115.750000	32.000000	335980.250000	500.000000	1089.607500	0.000000e+00	448404.500000	0.000000	-51500.000000	
50%	199.500000	38.000000	533135.000000	1000.000000	1257.200000	0.000000e+00	466445.500000	0.000000	-23250.000000	
75%	276.250000	44.000000	759099.750000	2000.000000	1415.695000	0.000000e+00	603251.000000	51025.000000	0.000000	
max	479.000000	64.000000	999435.000000	2000.000000	2047.590000	1.000000e+07	620962.000000	100500.000000	0.000000	

    - used .shape to get the shape of the dataframe

```
In [16]: df.shape
Out[16]: (1000, 40)
```

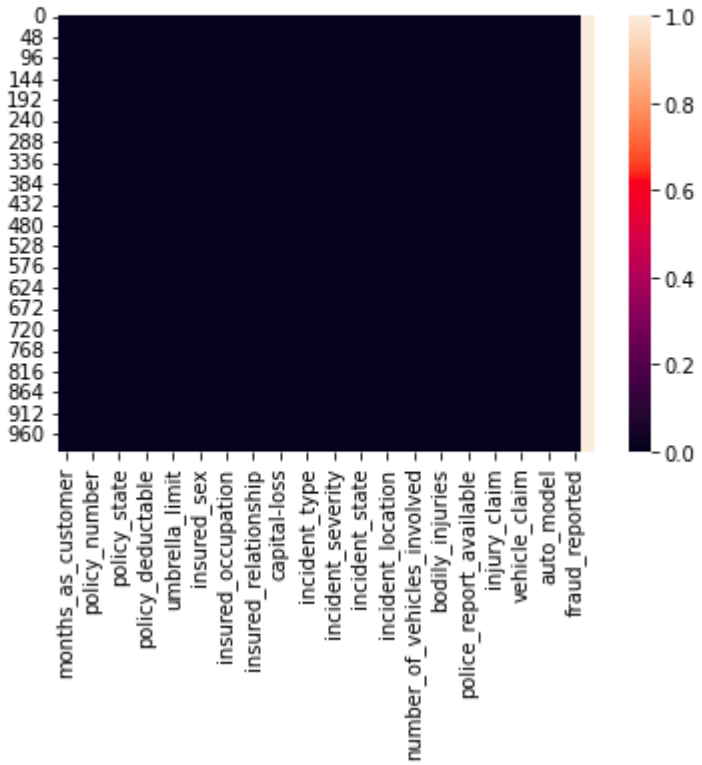
    - used .dtypes to get data types of each column of the dataframe

```
In [17]: #Column Data type
         df.dtypes
Out[17]: months_as_customer      int64
         age                     int64
         policy_number           int64
         policy_bind_date       object
         policy_state           object
         policy_csl             object
         policy_deductable       int64
         policy_annual_premium   float64
         umbrella_limit          int64
         insured_zip             int64
         insured_sex            object
```

2. Detecting NULL values :

- used Heatmap to get NULL values visually

In [18]:
```
#Checking null values using heatmap
sns.heatmap(df.isnull())
plt.show()
```



- used .isnull() to check NULL values in all columns

In [20]:
```
#number of null values in each column
df.isnull().sum()
```

Out[20]:
```
months_as_customer            0
age                           0
policy_number                 0
policy_bind_date              0
policy_state                  0
policy_csl                    0
policy_deductable             0
```

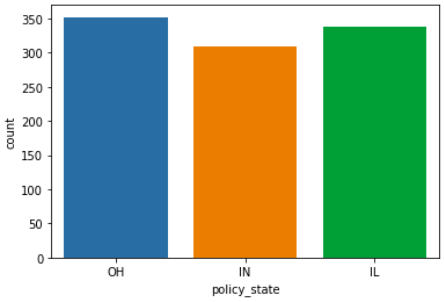
- Dropped '_C39' column as it is completely filled with NULL values

3. Data Visualization :

- Plotted Count Plots for Nominal data Column

```
In [25]: #Visualisation of policy_state data
         ax = sns.countplot(x='policy_state',data=df_visualisation_nominal)
         print(df_visualisation_nominal['policy_state'].value_counts())
```

```
OH    352
IL    338
IN    310
Name: policy_state, dtype: int64
```

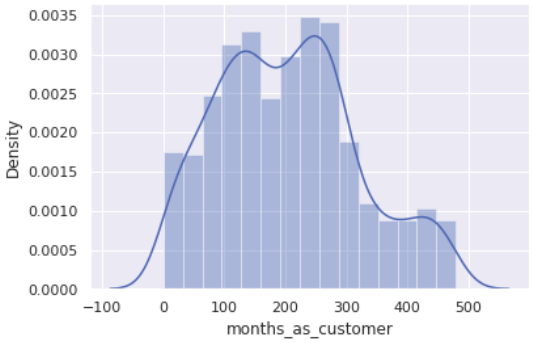


Highest number of Policy state is OH, then IL and finally IN

- Plotted Density plots for Continuous data Column

```
In [69]: #Visualisation of months_as_customer data
         sns.distplot(df_visualisation_continous['months_as_customer'])
```

```
Out[69]: <AxesSubplot:xlabel='months_as_customer', ylabel='Density'>
```



EDA Concluding Remarks

- Visualizing all the factors (independent variables) affecting the target variable (fraud_reported) helped a lot in understanding their contribution in affecting the target variable.
- Heatmaps are very useful in finding NULL values if a large number of NULL values are there in a column.
- Count plots are plotted for nominal data columns while density plots are plotted for continuous data columns

Pre-Processing Pipeline

- Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.
- Data preprocessing is a predominant step in machine learning to yield highly accurate and insightful results. Greater the quality of data, the greater is the reliability of the produced results.

Data Pre-Processing steps taken in project:

1. Encoded the Dataframe using Ordinal Encoder :

- Encoding categorical data is a process of converting categorical data into integer format so that the data with converted categorical values can be provided to the different models.
- We do this because most of the models require numbers as the data, not strings or anything else.

```
from sklearn.preprocessing import OrdinalEncoder
enc = OrdinalEncoder()
```

```
for i in df.columns:
    if df[i].dtypes == "object":
        df[i] = enc.fit_transform(df[i].values.reshape(-1,1))
```

2. Handling Correlations :

- Data and feature correlation is considered one important step in the feature selection phase of the data pre-processing.

- **Multicollinearity** happens when one predictor variable in a multiple regression model can be linearly predicted from the others with a high degree of accuracy. This can lead to skewed or misleading results.
- Decision trees and boosted **trees algorithms are immune to multicollinearity** by nature. When they decide to split, the tree will choose only one of the perfectly correlated features. However, other algorithms like Logistic Regression or Linear Regression are not immune to that problem and we should fix it before training the model.

  Steps involved for handling correlations in the project are :

  - Checked Correlations visually between different columns using heatmaps, bar plots and scatter plots.
  - Checked Correlations statistically between different columns using correlation matrix.
  - Confirmed Correlations using VIF.
  - Dropped columns (months_as_customer and vehicle_claim) which have high correlations with other columns (VIF > 10).

3. Handling Skewness :

- Data is skewed when its distribution curve is asymmetrical (as compared to a normal distribution curve that is perfectly symmetrical) and skewness is the measure of the asymmetry.
- More reliable predictions are made if the predictors and the target variable are normally distributed . Hence we need to remove skewness if it is there.
- However, **tree based models are not affected**.

  Steps involved for handling skewness in the project are :

  - Checked skewness of each column using .skew()
  - We took acceptable skewness range as (-0.65,+0.65) .
  - All the continuous columns were in the acceptable range (if it were not in acceptable range then we would have used yeo-johnson transformation).

4. Handling Outliers :

- Outlier is an observation in a given dataset that lies far from the rest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set.
- An outlier may occur due to the variability in the data, or due to experimental error/human error.
- 'Mean' is the only measure of central tendency that is affected by the outliers which in turn impacts Standard deviation.

Steps involved for handling outliers in the project are :

- Checked outliers of each column visually using Box-plot
- Checked outliers of each column statistically using z-score method
- Removed outliers from dataframe with a data loss of 2%

5. Up-sampling to remove Imbalance classification of Target variable(fraud_reported) :

- Imbalanced data refers to those types of datasets where the target class has an uneven distribution of observations, i.e one class label has a very high number of observations and the other has a very low number of observations.
- Real-world datasets can be highly imbalanced, which may affect performance of statistical algorithms or machine learning models.
- Imbalance in target data in case of classification model can be easily observed through visual representations (graphs) such as count plots.

Steps involved for upsampling in the project are :

- Used RandomOverSampler for up-sampling

In [114…

```
from collections import Counter
from imblearn.over_sampling import RandomOverSampler

print(Counter(target))

# Using oversampling
oversample = RandomOverSampler(sampling_strategy='minority')

feature_over, target_over = oversample.fit_resample(feature, target)

print(Counter(target_over))
```

```
Counter({0.0: 740, 1.0: 240})
Counter({1.0: 740, 0.0: 740})
```

Building Machine Learning Remarks

The purpose of this project is to identify the machine learning classification algorithm that is best-suited for the problem by comparing different classification algorithms and selecting the best-performing one.

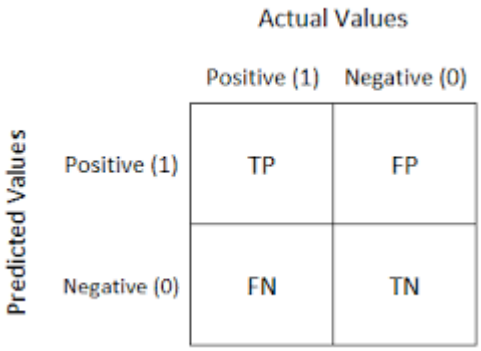
Some important terms used in model:

- **AUC-ROC Curve** :
    - When we need to check or visualize the performance of the multi-class classification problem, we use the AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve.
    - For a visual comparison of classification models, the ROC curve is utilized.
    - It illustrates the correlation between the false positive rate and the true positive rate. The accuracy of the model is determined by the area under the ROC curve.

- **Cross-Validation :**
    - The most prominent issue with most machine learning models is over-fitting. It is possible to check the model's overfitting with K-fold cross-validation.
    - With this technique, the data set is randomly divided into k equal-sized, mutually exclusive subsets. One is retained for testing, while the others are utilized for training the model. For each of the k folds, the same procedure is followed.

- **Confusion Matrix :**

- A confusion matrix is a table that is used to define the performance of a classification algorithm.
- A confusion matrix visualizes and summarizes the performance of a classification algorithm.



- **<u>Classification Report</u> :**
  - A Classification report is used to measure the quality of predictions from a classification algorithm.
  - The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives.

- **<u>Accuracy Score</u>** :
  - Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Steps taken for building model in project:

1. We defined a function ‘model’ using which we tried 4 different classification models – LogisticRegression(), SVC(), KNeighborsClassifier() and DecisionTreeClassifier()

```
def model_selection(algorithm_instance,features_train,target_train,features_test,target_test):
    algorithm_instance.fit(features_train,target_train)
    model_1_pred_train = algorithm_instance.predict(features_train)
    model_1_pred_test = algorithm_instance.predict(features_test)
    print("Accuracy for the training model : ",accuracy_score(target_train,model_1_pred_train))
    print("Accuracy for the testing model : ",accuracy_score(target_test,model_1_pred_test))
    print("Confusion matrix for model : \n",confusion_matrix(target_test,model_1_pred_test))
    print("Classification Report for train data : \n",classification_report(target_train,model_1_pred_train))
    print("Classification Report for test data : \n",classification_report(target_test,model_1_pred_test))

    Train_accuracy = accuracy_score(target_train,model_1_pred_train)
    Test_accuracy = accuracy_score(target_test,model_1_pred_test)

    for j in range(2,10):
        cv_score = cross_val_score(algorithm_instance,feature_over,target_over,cv=j)
        cv_mean = cv_score.mean()
        print("At cross fold " + str(j) + " the cv score is " + str(cv_mean) + " and accuracy score for training is " + str(Train_accurac
        print("\n")

    #Plotting auc_roc curve
    plt.figure(figsize=(8,6))

    # calculate roc curves
    lr_fpr, lr_tpr, _ = roc_curve(target_test, model_1_pred_test)
    lr_fpr1, lr_tpr1, _ = roc_curve(target_train, pred_train)

    plt.plot(lr_fpr, lr_tpr, marker='.', label=algorithm_instance, color='r')
    plt.plot(lr_fpr1, lr_tpr1, marker='*', label='No skill', color='b')

    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')

    plt.legend()
    plt.show()
```

2. General detail about each model :

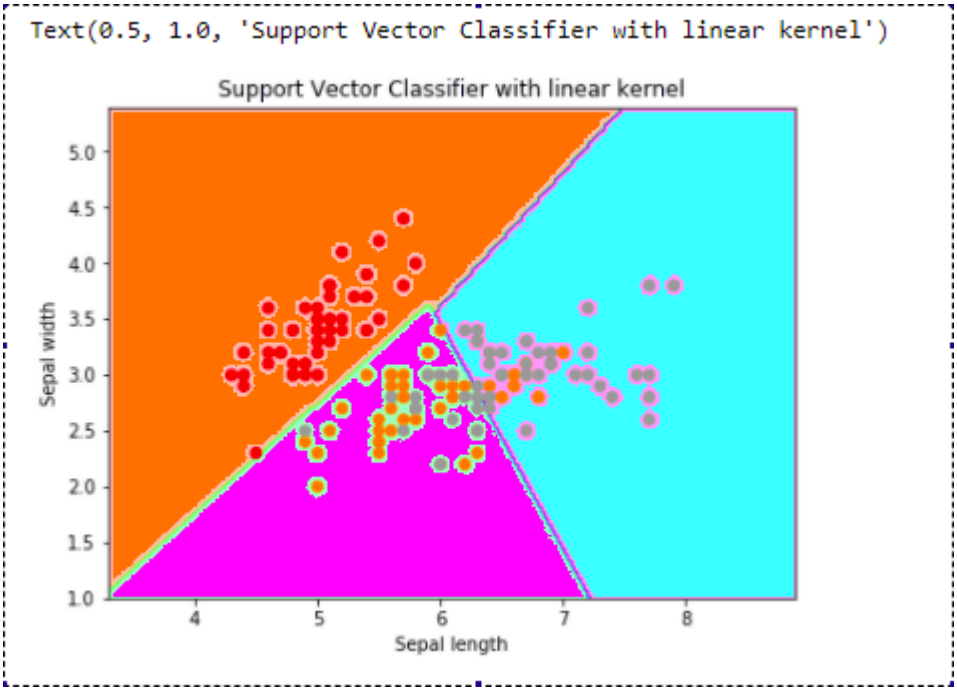
- **LogisticRegression()** :

    - Logistic regression aims to solve classification problems. It does this by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome.
    - Logistic regression is a fundamental classification technique. It belongs to the group of linear classifiers and is somewhat similar to linear regression.
    - Logistic regression is fast and relatively uncomplicated.

- **SVC()** :

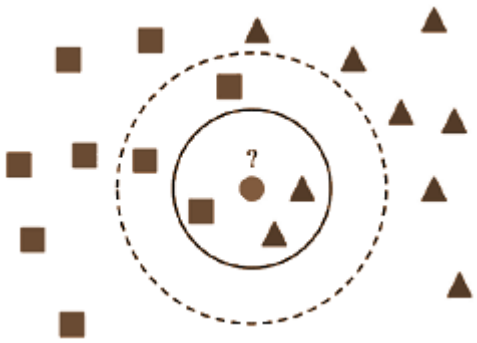
    - Support Vector Classification is a powerful yet flexible supervised machine learning methods used for classification.

- Support Vectors Classifier tries to find the best hyperplane to separate the different classes by maximizing the distance between sample points and the hyperplane.
- It handles the multiclass support according to one-vs-one scheme.

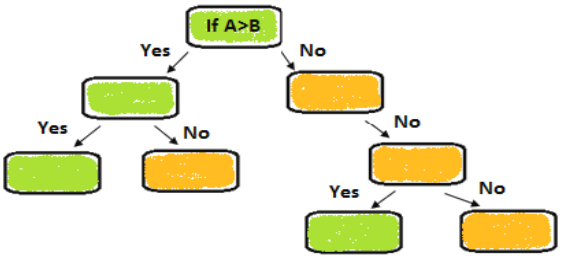


- **KNeighborsClassifier()** :

- KNeighborsClassifier is a supervised learning classifier which uses proximity to make classifications or predictions about the grouping of an individual data point.
- In KNeighborsClassifier, K is the number of nearest neighbors. The number of neighbors is the core deciding factor.
- KNeighborsClassifier is called lazy because it does no training at all when you supply the training data. At training time, all it is doing is storing the complete data set but it does not do any calculations at this point.
- The KNeighborsClassifier has no explicit training step and all the work happens during prediction.



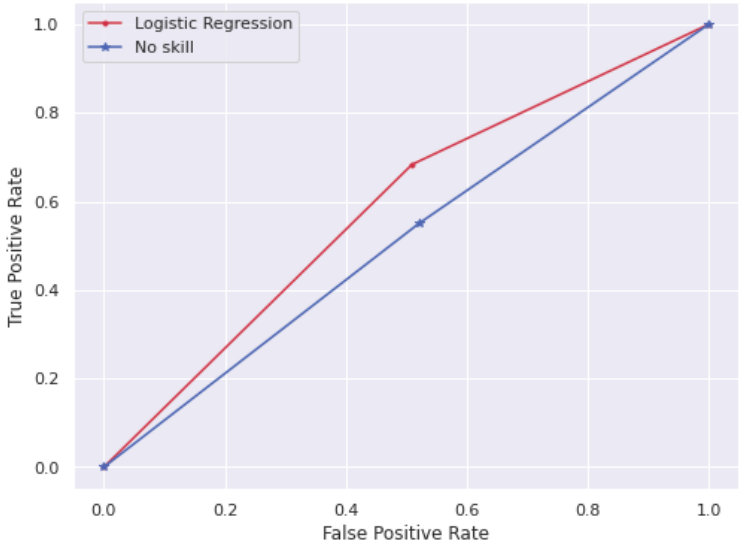
- **<u>DecisionTreeClassifier()</u> :**
    - DecisionTreeClassifier is a Supervised learning technique that is used for solving Classification problems.
    - It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
    - In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.



3. Results (statistically and graphically/AUC-ROC curve) which we got are as follows:

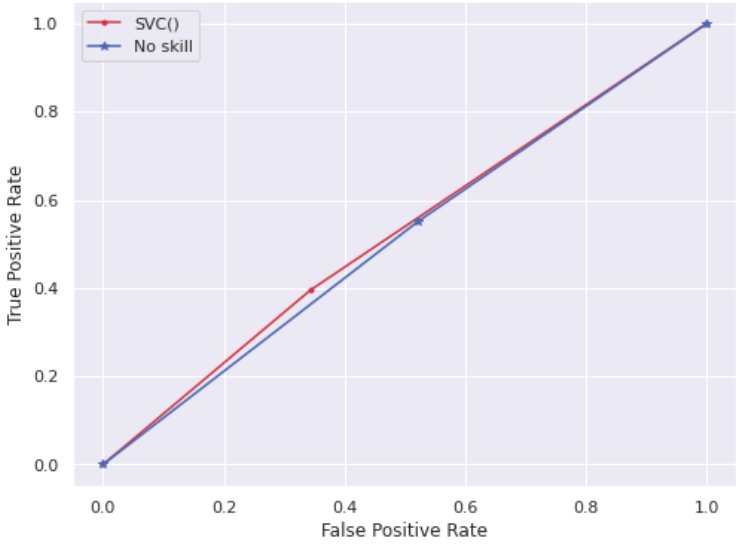
- **LogisticRegression()** :
  - At random state 7 the training accuracy is : 0.59375
  - At random state 7 the testing accuracy is : 0.581081081081081
  - At cross fold 8 the cv score is 0.5851351351351352 and accuracy score for training is 0.5152027027027027 and accuracy score for testing is 0.581081081081081

```
Confusion matrix for model :
 [[77 80]
 [44 95]]
Classification Report for train data :
              precision    recall  f1-score   support

         0.0       0.51      0.48      0.49       583
         1.0       0.52      0.55      0.54       601

    accuracy                           0.52      1184
   macro avg       0.51      0.51      0.51      1184
weighted avg       0.51      0.52      0.51      1184

Classification Report for test data :
              precision    recall  f1-score   support

         0.0       0.64      0.49      0.55       157
         1.0       0.54      0.68      0.61       139

    accuracy                           0.58       296
   macro avg       0.59      0.59      0.58       296
weighted avg       0.59      0.58      0.58       296
```



- **SVC()** :
  - Accuracy for the training model : 0.5278716216216216
  - Accuracy for the testing model : 0.5337837837837838
  - At cross fold 2 the cv score is 0.5175675675675675 and accuracy score for training is 0.5278716216216216 and accuracy score for testing is 0.5337837837837838

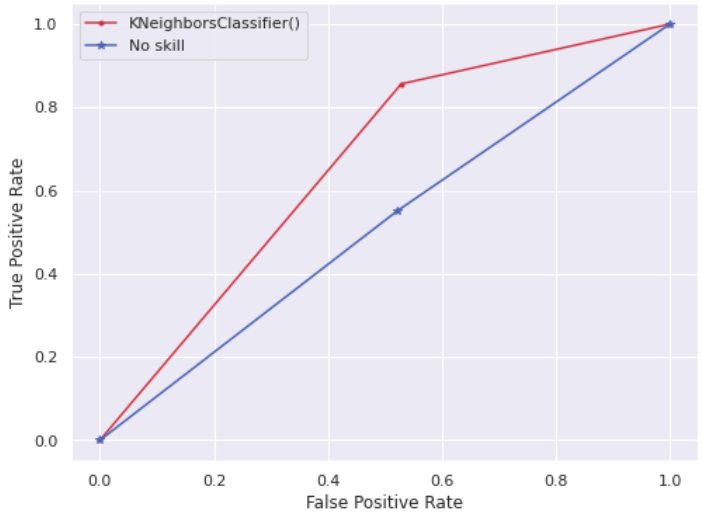
```
Confusion matrix for model :
 [[103  54]
 [ 84  55]]
Classification Report for train data :
              precision    recall  f1-score   support

         0.0       0.52      0.64      0.57       583
         1.0       0.55      0.42      0.47       601

    accuracy                           0.53      1184
   macro avg       0.53      0.53      0.52      1184
weighted avg       0.53      0.53      0.52      1184

Classification Report for test data :
              precision    recall  f1-score   support

         0.0       0.55      0.66      0.60       157
         1.0       0.50      0.40      0.44       139

    accuracy                           0.53       296
   macro avg       0.53      0.53      0.52       296
weighted avg       0.53      0.53      0.53       296
```



- **KNeighborsClassifier()** :

- Accuracy for the training model : 0.7989864864864865
- Accuracy for the testing model : 0.652027027027027
- At cross fold 2 the cv score is 0.6216216216216217 and accuracy score for training is 0.7989864864864865 and accuracy score for testing is 0.652027027027027

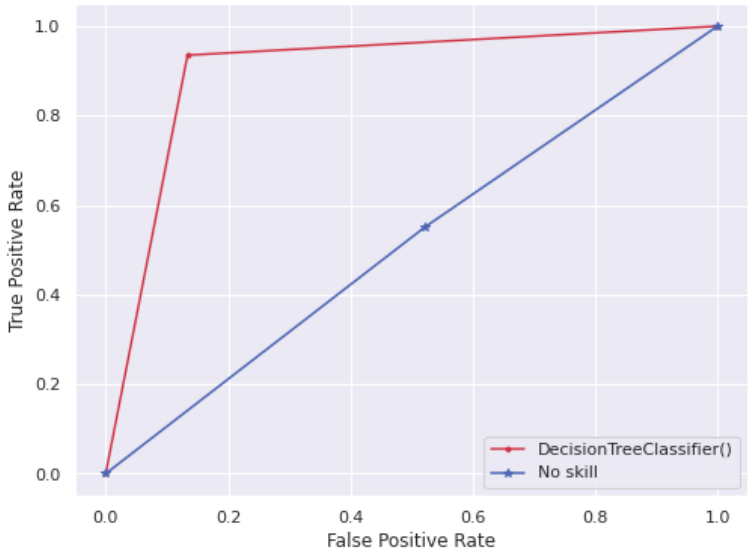
```
Confusion matrix for model :
 [[ 74  83]
 [ 20 119]]
Classification Report for train data :
              precision    recall  f1-score   support

         0.0       0.86      0.70      0.78       583
         1.0       0.76      0.89      0.82       601

    accuracy                           0.80      1184
   macro avg       0.81      0.80      0.80      1184
weighted avg       0.81      0.80      0.80      1184

Classification Report for test data :
              precision    recall  f1-score   support

         0.0       0.79      0.47      0.59       157
         1.0       0.59      0.86      0.70       139

    accuracy                           0.65       296
   macro avg       0.69      0.66      0.64       296
weighted avg       0.69      0.65      0.64       296
```



- **<u>DecisionTreeClassifier()</u> :**
  - Accuracy for the training model : 1.0

- Accuracy for the testing model : 0.8986486486486487
- At cross fold 2 the cv score is 0.8878378378378378 and accuracy score for training is 1.0 and accuracy score for testing is 0.8986486486486487

```
Confusion matrix for model :
 [[136  21]
 [  9 130]]
Classification Report for train data :
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       583
         1.0       1.00      1.00      1.00       601

    accuracy                           1.00      1184
   macro avg       1.00      1.00      1.00      1184
weighted avg       1.00      1.00      1.00      1184

Classification Report for test data :
              precision    recall  f1-score   support

         0.0       0.94      0.87      0.90       157
         1.0       0.86      0.94      0.90       139

    accuracy                           0.90       296
   macro avg       0.90      0.90      0.90       296
weighted avg       0.90      0.90      0.90       296
```



4. We selected DecisionTreeClassifier() as our main model due to best accuracy among the 4 models which can be seen statistically and graphically

5. Finally we tuned the Hyperparameter using GridSearchCV and got Accuracy score of **78.04%**

Conclusion Remarks

- Classification can be considered a standard supervised learning activity. It is a valuable strategy that we use while attempting to determine whether a specific example falls into a given category or not.
- We got to know that total_claim_amount and vehicle_claim as well as age and months_as_customer columns in the project are highly correlated.
- All the continuous columns were in the acceptable range for skewness otherwise we would have used yeo-johnson transformation to get their skewness in acceptable range.
- The best model out of 4 models is selected on the basis of area under the curve of AUC-ROC curve as well as classification report.
- After tuning the Hyperparameter using GridSearchCV and DecisionTreeClassifier, we got Accuracy score of **78.04%**