Fraud Detection in Insurance Claims

Milestone: Project Report

Group 17

Gavin Dsa
Rohith Kumar Manjunatha

+1 (857) 869 - 5589
+1 (857) 334 - 3974

dsa.g@northeastern.edu

manjunatha.r@northeastern.edu

Percentage of contribution by Student1: 50%
Percentage of contribution by Student2: 50%

Submission Date: 1st May 2022

Problem Setting:

Insurance fraud detection is a challenging problem, given the variety of fraud patterns and small ratio of known frauds in typical samples. Machine learning techniques allow for improving predictive accuracy, enabling loss control units to achieve higher coverage with low false-positive rates. In this dataset, machine learning techniques for fraud detection are presented and their performance is examined.

Problem Definition:

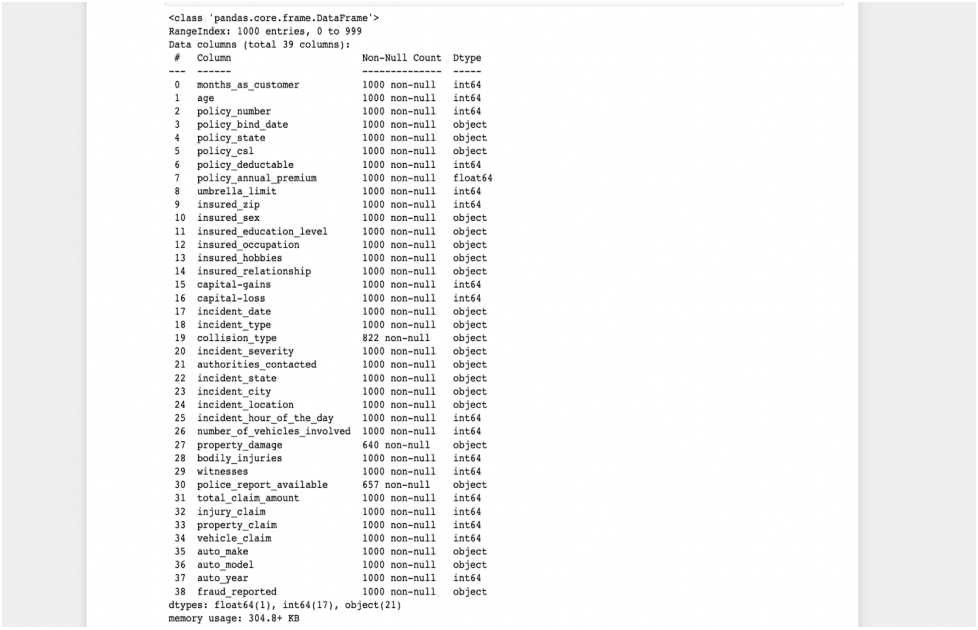
Since the U.S faces insurance fraud of about $80 billion every year, we are trying to investigate what kind of factors are important while predicting insurance fraud. This would help insurance companies since they wouldn't need to investigate every single incident and would only spend time and money on the ones that are more likely to be frauds. We are thus trying to build a classification model to tell us whether the incident is more likely to be a fraud or not.

Data Source:

We got this data from Kaggle, and the link is as follows:

https://www.kaggle.com/roshansharma/fraud-detection-in-insurance-claims/data

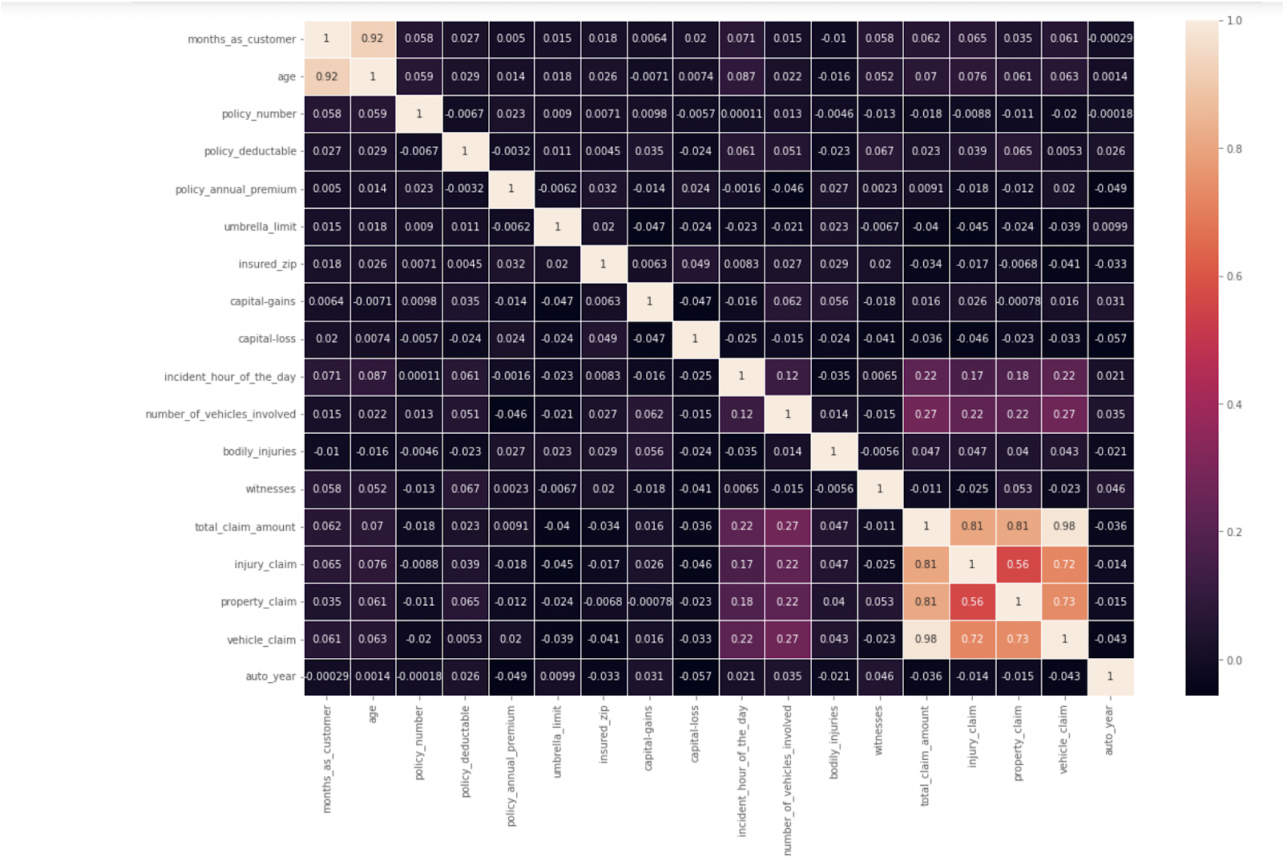
Data Description:



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 39 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   months_as_customer           1000 non-null   int64
 1   age                          1000 non-null   int64
 2   policy_number                1000 non-null   int64
 3   policy_bind_date             1000 non-null   object
 4   policy_state                 1000 non-null   object
 5   policy_csl                   1000 non-null   object
 6   policy_deductable            1000 non-null   int64
 7   policy_annual_premium        1000 non-null   float64
 8   umbrella_limit               1000 non-null   int64
 9   insured_zip                  1000 non-null   int64
 10  insured_sex                  1000 non-null   object
 11  insured_education_level      1000 non-null   object
 12  insured_occupation           1000 non-null   object
 13  insured_hobbies              1000 non-null   object
 14  insured_relationship         1000 non-null   object
 15  capital-gains                1000 non-null   int64
 16  capital-loss                 1000 non-null   int64
 17  incident_date                1000 non-null   object
 18  incident_type                1000 non-null   object
 19  collision_type               822 non-null    object
 20  incident_severity            1000 non-null   object
 21  authorities_contacted        1000 non-null   object
 22  incident_state               1000 non-null   object
 23  incident_city                1000 non-null   object
 24  incident_location            1000 non-null   object
 25  incident_hour_of_the_day     1000 non-null   int64
 26  number_of_vehicles_involved  1000 non-null   int64
 27  property_damage              640 non-null    object
 28  bodily_injuries              1000 non-null   int64
 29  witnesses                    1000 non-null   int64
 30  police_report_available      657 non-null    object
 31  total_claim_amount           1000 non-null   int64
 32  injury_claim                 1000 non-null   int64
 33  property_claim               1000 non-null   int64
 34  vehicle_claim                1000 non-null   int64
 35  auto_make                    1000 non-null   object
 36  auto_model                   1000 non-null   object
 37  auto_year                    1000 non-null   int64
 38  fraud_reported               1000 non-null   object
dtypes: float64(1), int64(17), object(21)
memory usage: 304.8+ KB
```

The data contains about 1000 rows and 39 columns with key attributes such as Age, Annual premium, policy deductible, umbrella limit, insured occupation, capital gains, capital loss, incident type, collision type, incident severity, witnesses, etc. Target variable (Y): Fraud reported (Yes/No) would be our target variable.

Data Exploration:

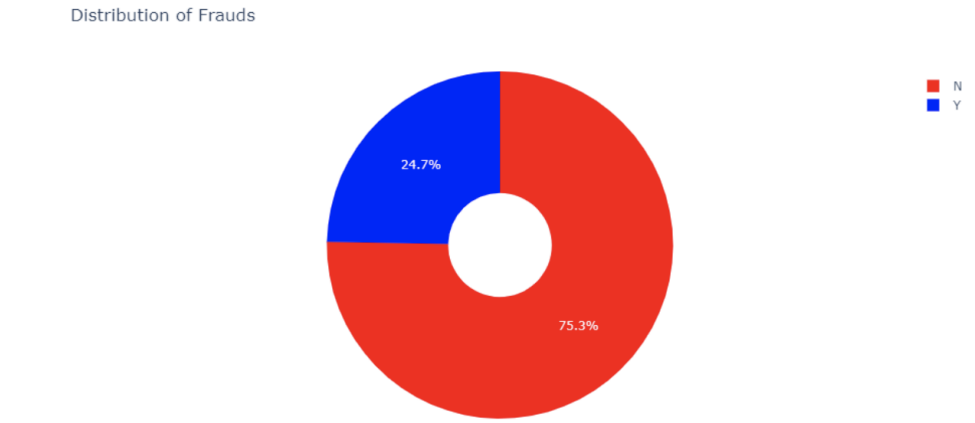


From this correlation plot, we can see that months_as_customers and age are highly correlated, so we can drop the 'Age' column. Similarly, even total_clam_amount, injury_claim, property_claim, vehicle_claim are highly correlated, and total claim is the sum of the other claims. Hence, we can drop that column too.

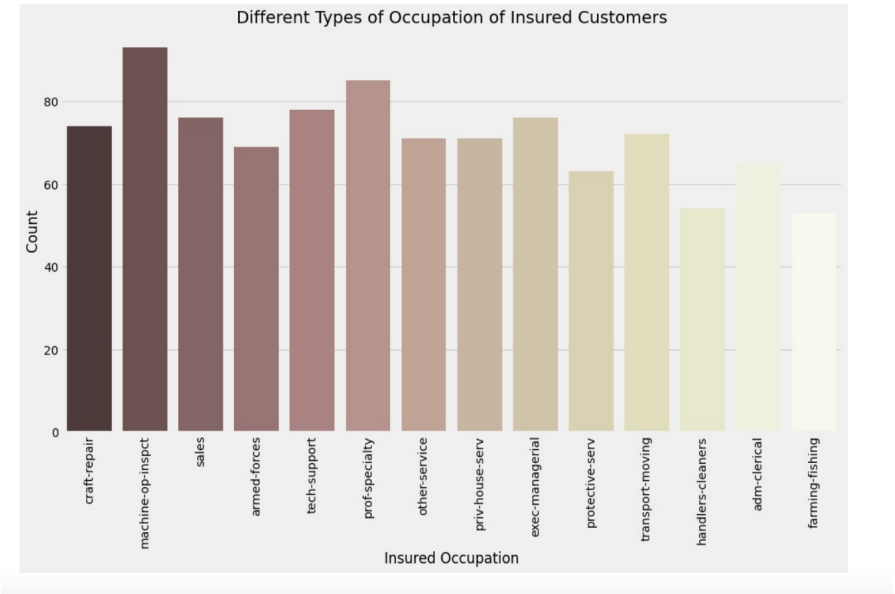
Distribution of Frauds:

There were 247 frauds and 753 non-frauds. We found that 24.7% of the data were frauds while 75.3% were non-fraudulent claims.

To find out the distribution of frauds:



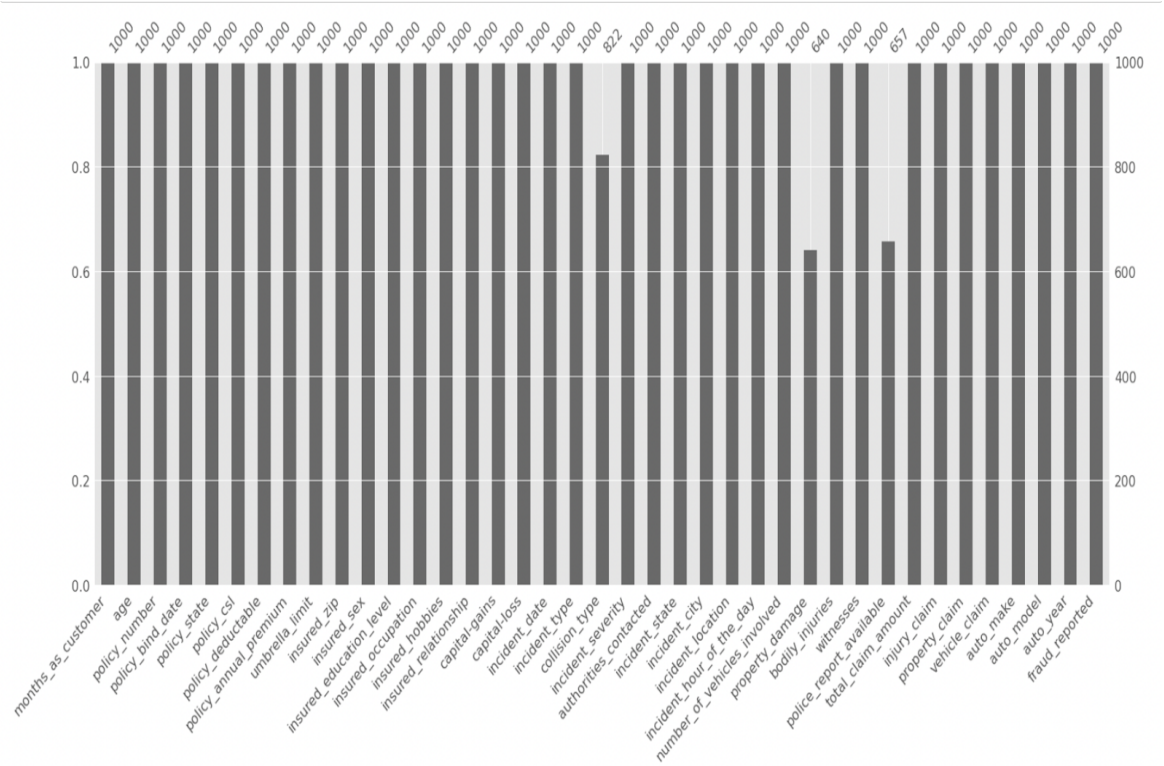
To study the different types of occupations that the insured customers had:



We can see that machine op inspectors are the most insured and farming- fishing are the least insured.

Data Preprocessing:

Firstly, we made the following plot to visualize the number of missing values in the dataset, in order to replace them with appropriate values.



Since the dataset had some '?' values, we have replaced them with 'NaN' values. We have also replaced the '?' values in collision type with the mode values (Since it occurs the most). We have replaced 'Property_damage' with the response 'No' wherever it is not available because the lack of a response is more likely to be in the case of no property damage. For the same reason, we have also filled 'Police_report_available' values with 'No' wherever it is not available.

To account for the scaling disparity, the dataset was standardized using the scikit-learn package's StandardScaler() function to guarantee that all predictor variables were given equal weight in terms of variability.

Model Exploration:

We split the dataset into 75% training set and 25% test set. The following models were used:

1. Support Vector Machine
2. K - Nearest Neighbors
3. Decision Tree
4. Random Forest

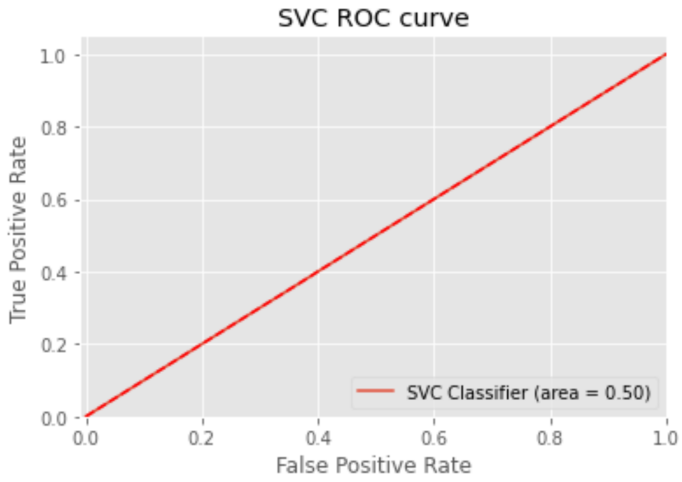
Support Vector Machine:

```
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report

svc_train_acc = accuracy_score(y_train, svc.predict(X_train))
svc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of the Support Vector Classifier is :␣
 ↪{svc_train_acc}")
print(f"Test accuracy of the Support Vector Classifier is : {svc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Training accuracy of the Support Vector Classifier is : 0.8293333333333334
Test accuracy of the Support Vector Classifier is : 0.752
[[188   0]
 [ 62   0]]
              precision    recall  f1-score   support

           N       0.75      1.00      0.86       188
           Y       0.00      0.00      0.00        62

    accuracy                           0.75       250
   macro avg       0.38      0.50      0.43       250
weighted avg       0.57      0.75      0.65       250
```

Accuracy is **0.75**

ROC Curve:



The area of the curve is 0.5, which means that SVC is not a very good model for this.

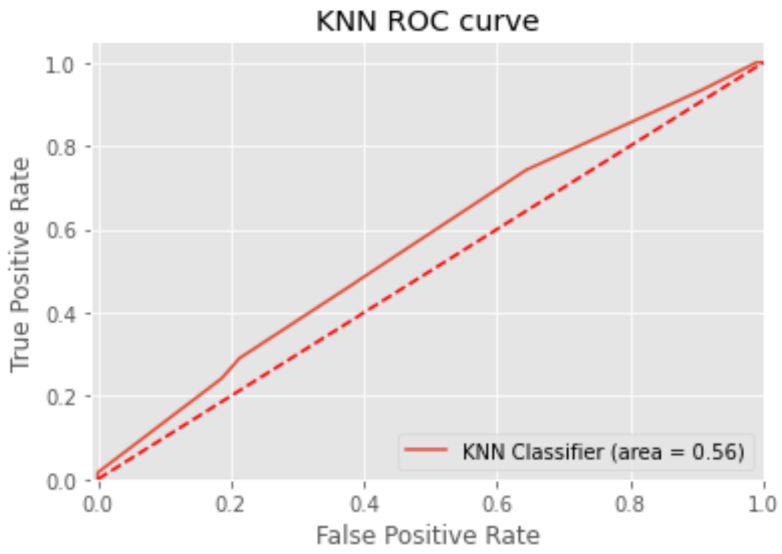
K - Nearest Neighbors:

```
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report

knn_train_acc = accuracy_score(y_train, knn.predict(X_train))
knn_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of KNN is : {knn_train_acc}")
print(f"Test accuracy of KNN is : {knn_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Training accuracy of KNN is : 0.7546666666666667
Test accuracy of KNN is : 0.752
[[188   0]
 [ 62   0]]
              precision    recall  f1-score   support

           N       0.75      1.00      0.86       188
           Y       0.00      0.00      0.00        62

    accuracy                           0.75       250
   macro avg       0.38      0.50      0.43       250
weighted avg       0.57      0.75      0.65       250
```

Accuracy is **0.75**

ROC Curve:



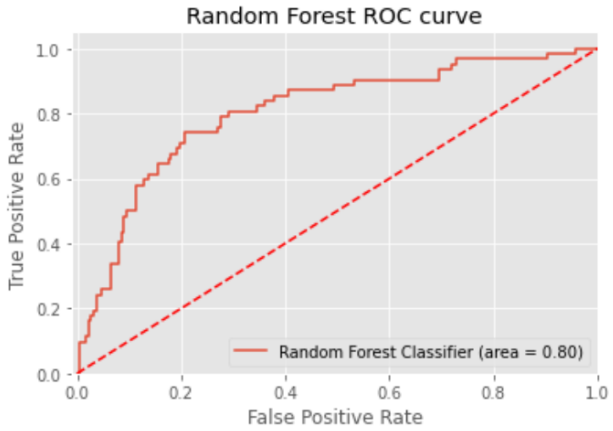
The area under the curve is 0.56, which is not a great value and hence this model does not work that well.

Random Forest:

```
Training accuracy of the Random Forest is : 0.956
Test accuracy of the Random Forest is : 0.768
[[186   2]
 [ 56   6]]
              precision    recall  f1-score   support

           N       0.77      0.99      0.87       188
           Y       0.75      0.10      0.17        62

    accuracy                           0.77       250
   macro avg       0.76      0.54      0.52       250
weighted avg       0.76      0.77      0.69       250
```

Accuracy is **0.77**

ROC Curve:



The area under this curve is significantly better than the previous two models and does a pretty decent job at classifying frauds.

Decision Tree:

```
dtc_train_acc = accuracy_score(y_train, dtc.predict(X_train))
dtc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of the Decision Tree is : {dtc_train_acc}")
print(f"Test accuracy of the Decision Tree is : {dtc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Training accuracy of the Decision Tree is : 1.0
Test accuracy of the Decision Tree is : 0.624
[[129  59]
 [ 35  27]]
              precision    recall  f1-score   support

           N       0.79      0.69      0.73       188
           Y       0.31      0.44      0.36        62

    accuracy                           0.62       250
   macro avg       0.55      0.56      0.55       250
weighted avg       0.67      0.62      0.64       250
```

Accuracy is **0.62**

Decision Tree (Hyper parameter tuning):

Since we got a pretty low accuracy for decision trees, we used Grid search to find the best parameters to train the model.

Using Grid search we got a Fitting of 5 folds for each of 512 candidates, totaling 2560 fits.

The best parameters were: 'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 5, 'min_samples_split': 2.

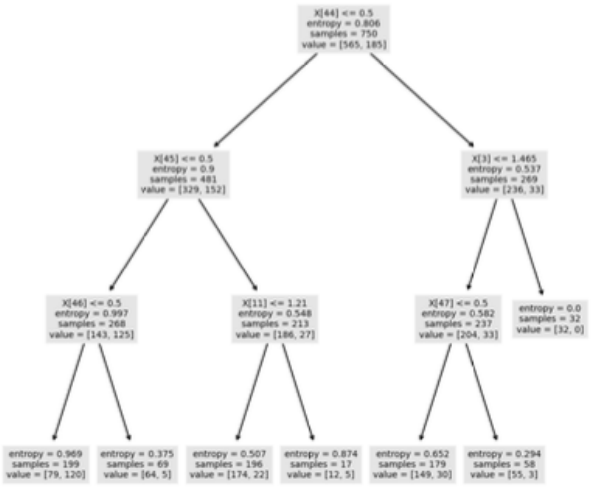
After training the new model, we obtained the following:

```
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report

dtc_train_acc = accuracy_score(y_train, dtc.predict(X_train))
dtc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of the Decision Tree is : {dtc_train_acc}")
print(f"Test accuracy of the Decision Tree is : {dtc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

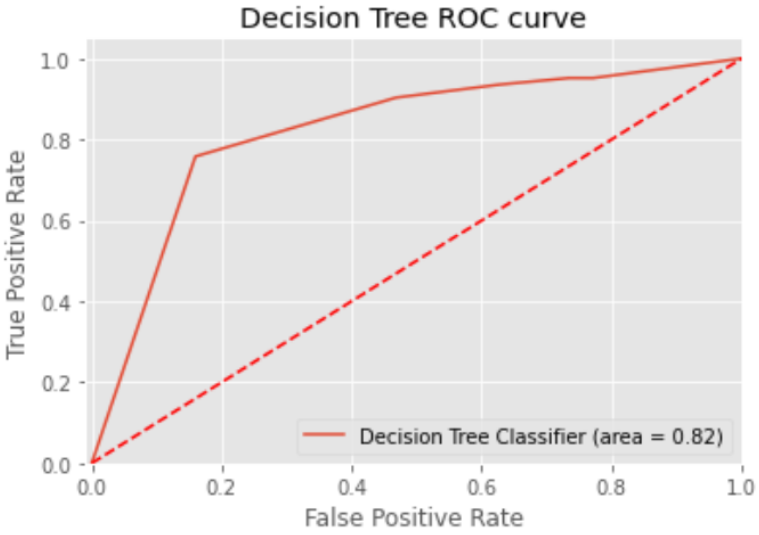
```
Training accuracy of the Decision Tree is : 0.808
Test accuracy of the Decision Tree is : 0.82
[[158  30]
 [ 15  47]]
              precision    recall  f1-score   support

           N       0.91      0.84      0.88       188
           Y       0.61      0.76      0.68        62

    accuracy                           0.82       250
   macro avg       0.76      0.80      0.78       250
weighted avg       0.84      0.82      0.83       250
```

Here we saw that the test accuracy has increased significantly to 0.82. Hence Grid search has helped us find the best parameters for our decision tree model.

Decision Tree Plot:

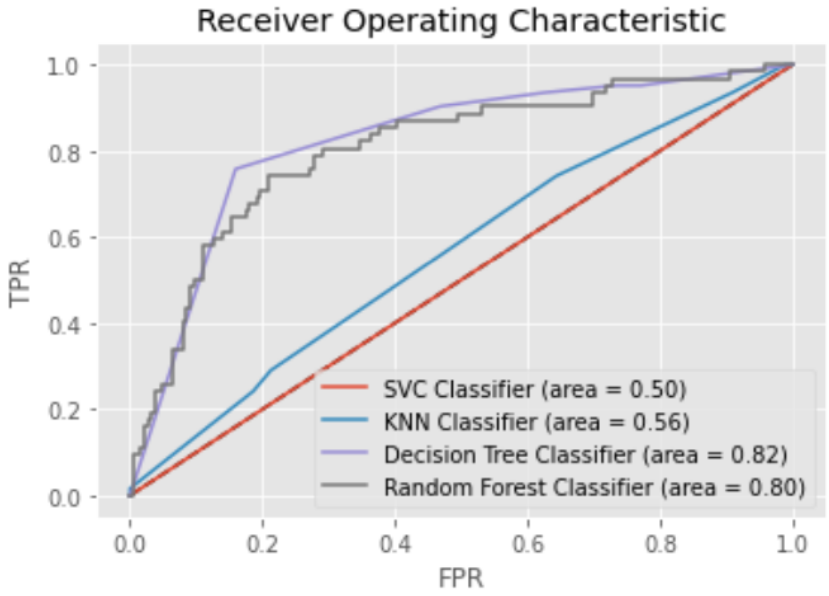


ROC Curve:



The area under the curve is 0.82 and is the best so far, hence this is our main model for classification.

Performance Evaluation:



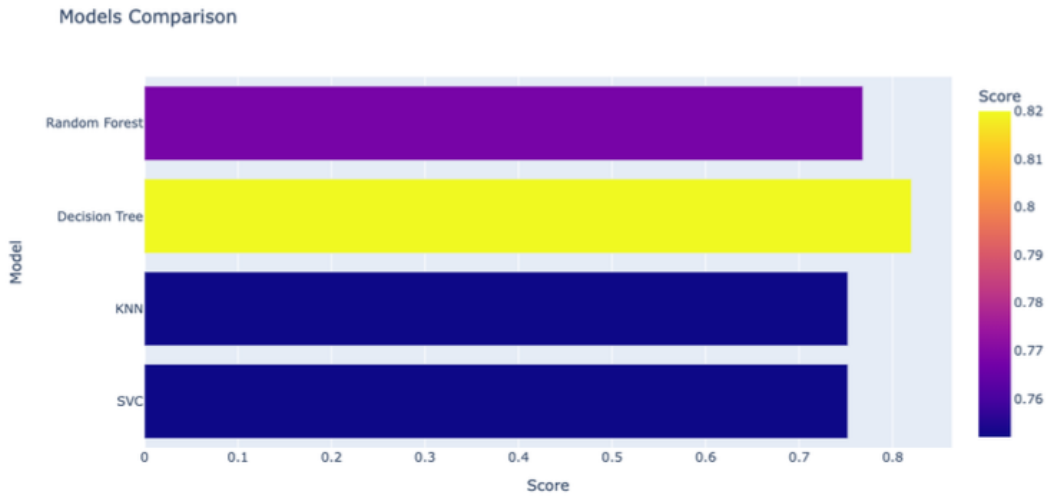
From the ROC plot comparison, we can see that Decision Trees is the best model.

Model	Score
Decision Tree	0.820
Random Forest	0.768
SVC	0.752
KNN	0.752

Model Comparison:

Classification Model	Training Accuracy	Test Accuracy
Support Vector Machine	0.8293	0.752
K-NN	0.7547	0.752
Random Forest	0.956	0.768
Decision Tree(Hyper parameter tuning)	0.808	0.82

When we look at the training accuracy and test accuracy of the models, we can clearly see that decision trees is the best model for classification of fraudulent insurance claims.



Project Results:

Our best model is Decision Trees with an accuracy score of 0.82. The model performed excellently. The model's F1 score and ROC AUC scores were the highest among the other models. In conclusion, the model was able to correctly distinguish between fraud claims and legit claims with high accuracy.

The study is not without limitations. Firstly, this study is restricted by its small sample size. Statistical models are more stable when datasets are larger. It also generalizes better as it takes a bigger proportion of the actual population.

Impact of the Project Outcome:

Instead of spending money on the investigation of fraudulent cases, insurance companies can leverage this classification model and only spend money on investigating cases that are classified as fraudulent by this model. This can save insurance companies a lot of money and will hence benefit their entire organization.