Fraudulent Claim Detection

Yajat S
Ruchika Raju
K Sai Pranava Karthik
Sourav Kumar Jha

Problem Statement

Global Insure is one of the pioneers in insurance. They process thousands of claims every year and, unfortunately, a fair number of those claims tend to be fraudulent. These fraudulent claims are exhaustive in their nature and costly to process.

With today's state of the art technology, detection of fraud is still reliant on manual checking of various documents and this takes a lot of time and is not efficient at all. A lot of times, fraudulent claims are uncovered after paying out the claim. In such cases it is already too late, money has been lost. This is what Global Insure intends to fix by building smarter detection systems through a more accessible use of data that has the ability to distinguish fraudulent claims from legitimate ones at the approval stage. This approach will not only reduce the expenses incurred but will also enhance the performance of the claims management system.

Business Objective

Global Insure plans to develop a model that automatically classifies insurance claims as either fraudulent or legitimate using historical claim details along with a customer's profile. As a part of this model, the company plans on utilizing existing features such as amount of claim, customer profile, and classification of the claim so that it can some what predict which claims would be fraudulent and would require extra scrutiny before approval.

Business Objective

Addressing some of the questions below based on the case study,

Q) How can we analyse historical claim data to detect patterns that indicate fraudulent claims?

A) To identify patterns of fraud in historical claim data, we can begin with Exploratory Data Analysis (EDA). This involves visualizing the data using plots like bar charts, box plots, and histograms to understand the distribution of various features across fraudulent and non-fraudulent claims. For instance, we might notice that claims marked as fraudulent often involve higher-than-average claim amounts or occur more frequently on specific days. Additionally, correlation analysis helps identify relationships between features—for example, if claims with missing police reports are more often fraudulent. Target likelihood analysis is another useful tool, where we look at how likely a claim is to be fraudulent given a particular feature value. This kind of deep dive helps uncover subtle signals that wouldn't be obvious by just looking at raw data.

Q) Which features are most predictive of fraudulent behaviour?

A) Incident severity: Extremely severe claims, especially when inconsistent with the reported damage or accident type, can be suspicious.

Vehicle condition: Claims involving older or poorly maintained vehicles sometimes show up more in fraudulent cases.

Number of witnesses: Fraudulent claims may often lack witnesses or have inconsistent witness accounts.

Business Objective

Insured's hobbies or lifestyle details: While it may seem odd, certain personal data points—like high-risk hobbies—can correlate with more suspicious patterns.

Police report availability: The absence of a police report in serious incidents is often a red flag.

Claim amount: Very high or unusually rounded claim amounts can indicate attempts to game the system.

Q) Can we predict the likelihood of fraud for an incoming claim, based on past data?

A) Yes, and this is where machine learning really shines. By training models like Random Forests or Logistic Regression on past data where claims were labeled as fraudulent or genuine, the system can learn to recognize patterns that suggest fraud. Once trained, the model can analyze new claims and assign a probability score indicating how likely each is to be fraudulent. This allows for efficient, data-driven triaging—flagging high-risk claims for deeper review, while letting low-risk ones pass more smoothly.

Q) What insights can be drawn from the model that can help in improving the fraud detection process?

A) Missing or inconsistent data can be a signal: For instance, if a claim lacks key information like a police report or has conflicting dates, it could suggest suspicious activity.

Business Objective

Model-driven triaging: Claims with a high probability of being fraudulent (based on the model) can be automatically flagged for manual investigation, improving operational efficiency.

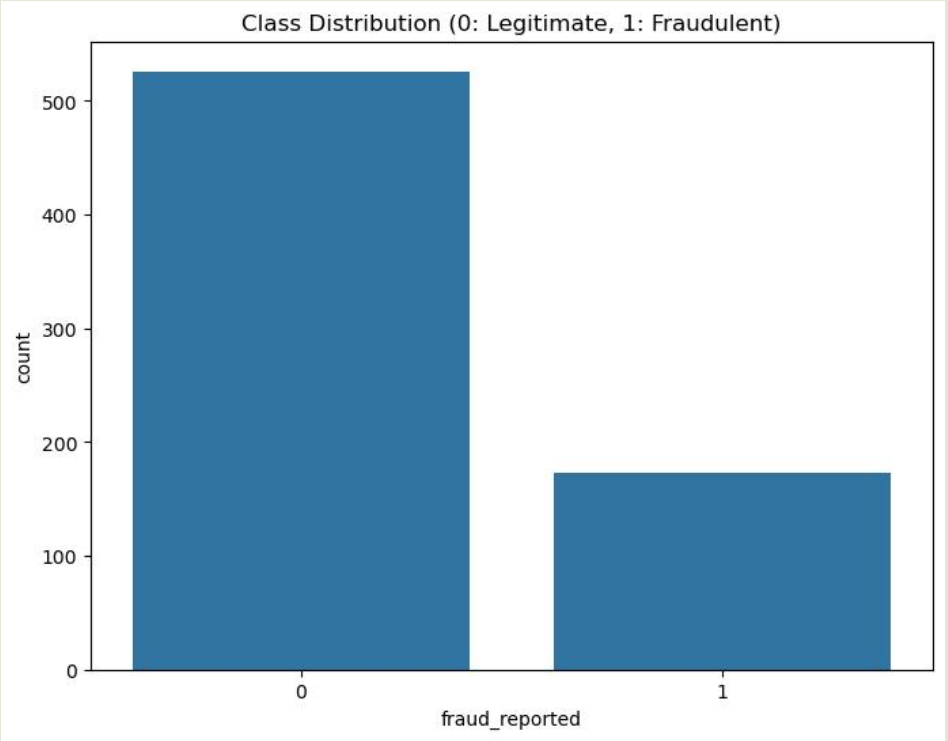
Continuous learning is key: Fraud tactics evolve, so our model should evolve too. Regular retraining with updated data helps keep the detection system relevant and sharp.

Thresholds can be tuned: Depending on the business's risk appetite, the sensitivity of fraud detection can be adjusted—balancing false positives (flagging honest claims) with false negatives (missing actual fraud).

Explainability matters: Using interpretable models or tools like SHAP values can help understand why the model flagged a claim, which builds trust and assists investigators.

Dataset Overview

- The dataset contains **1000 rows and 40 columns**.
- The target variable is fraud_reported. (yes/no)
- Features include customer info, incident details, claim amounts, and more.



Dataset Preprocessing

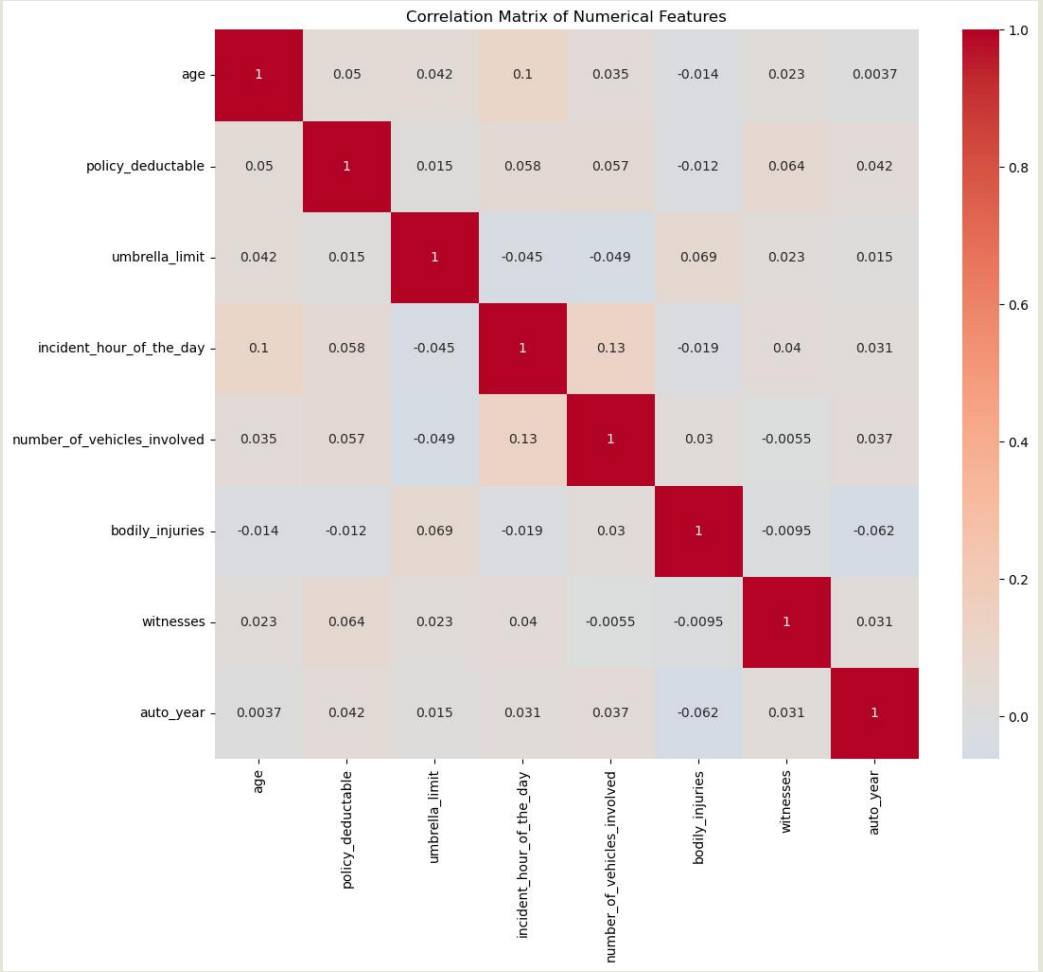
- Handled missing values, Median for numerics, Mode for categoricals
- Dropped redundant columns
- Converted data types as needed
- Cleaned irrelevant characters and inconsistencies

```
In [341]: # Check the number of missing values in each column
          df.isnull().sum()

Out[341]: months_as_customer            0
          age                           0
          policy_number                 0
          policy_bind_date              0
          policy_state                  0
          policy_csl                    0
          policy_deductable             0
          policy_annual_premium         0
          umbrella_limit                0
          insured_zip                   0
          insured_sex                   0
          insured_education_level       0
          insured_occupation            0
          insured_hobbies               0
          insured_relationship          0
          capital-gains                 0
          capital-loss                  0
          incident_date                 0
          incident_type                 0
          collision_type                0
          incident_severity             0
          authorities_contacted        91
          incident_state                0
          incident_city                 0
          incident_location             0
          incident_hour_of_the_day      0
          number_of_vehicles_involved   0
          property_damage               0
          bodily_injuries               0
          witnesses                     0
          police_report_available       0
          total_claim_amount            0
          injury_claim                  0
          property_claim                0
          vehicle_claim                 0
          auto_make                     0
          auto_model                    0
          auto_year                     0
          fraud_reported                0
          _c39                       1000
          dtype: int64
```

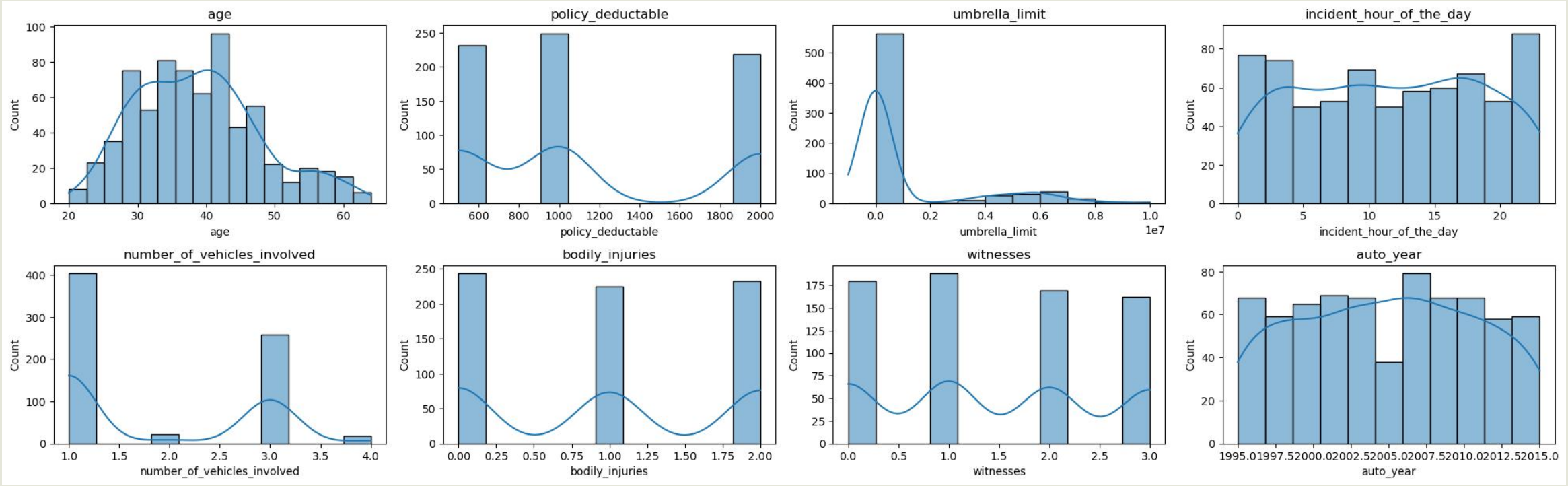
```
In [344]: #check for null values
          df.isnull().sum()

Out[344]: months_as_customer            0
          age                           0
          policy_number                 0
          policy_bind_date              0
          policy_state                  0
          policy_csl                    0
          policy_deductable             0
          policy_annual_premium         0
          umbrella_limit                0
          insured_zip                   0
          insured_sex                   0
          insured_education_level       0
          insured_occupation            0
          insured_hobbies               0
          insured_relationship          0
          capital-gains                 0
          capital-loss                  0
          incident_date                 0
          incident_type                 0
          collision_type                0
          incident_severity             0
          authorities_contacted         0
          incident_state                0
          incident_city                 0
          incident_location             0
          incident_hour_of_the_day      0
          number_of_vehicles_involved   0
          property_damage               0
          bodily_injuries               0
          witnesses                     0
          police_report_available       0
          total_claim_amount            0
          injury_claim                  0
          property_claim                0
          vehicle_claim                 0
          auto_make                     0
          auto_model                    0
          auto_year                     0
          fraud_reported                0
          _c39                       1000
          dtype: int64
```

Exploratory Data Analysis(EDA)

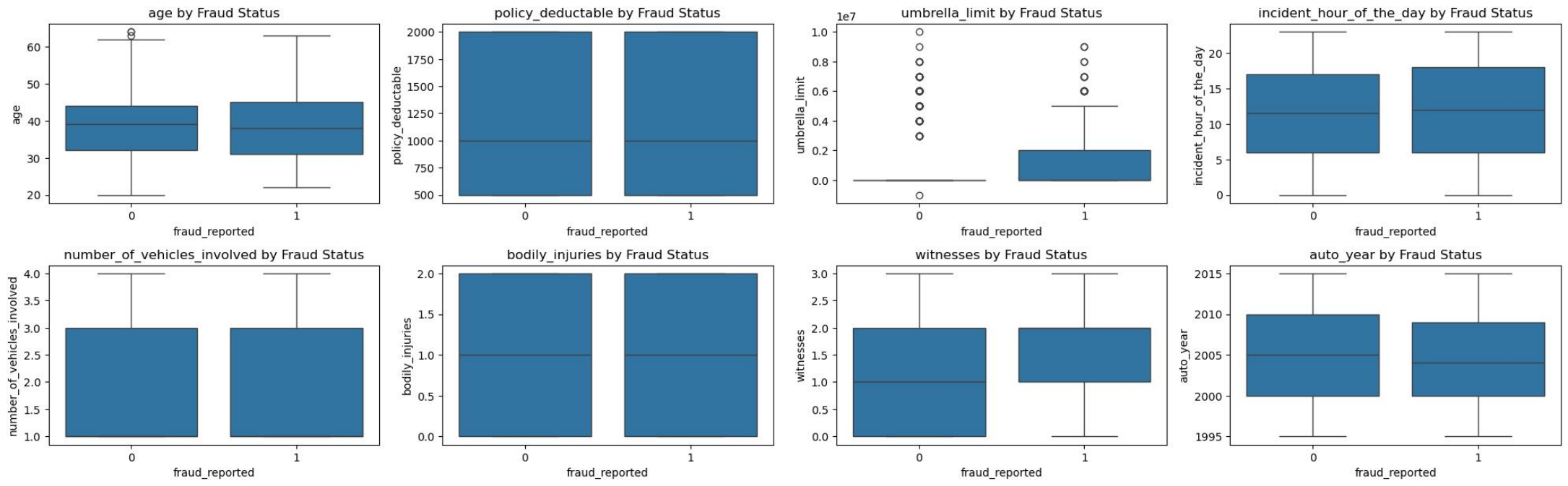
- Univariate and Bivariate analysis for numeric and categorical variables.
- Found strong correlations between:
  - incident_severity and fraud
  - Total_claim_amount
  - Vehicle_claim
- Used heatmaps, boxplots, and distribution plots
- Later performed feature engineering and handled class imabalance.



Exploratory Data Analysis(EDA)

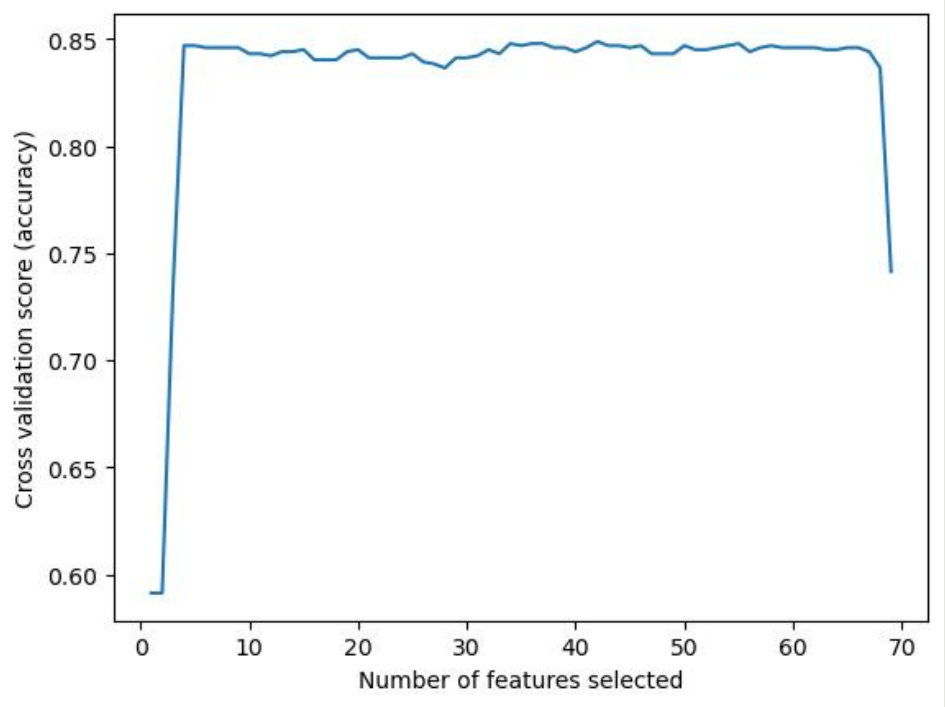


Exploratory Data Analysis(EDA)



Model Building

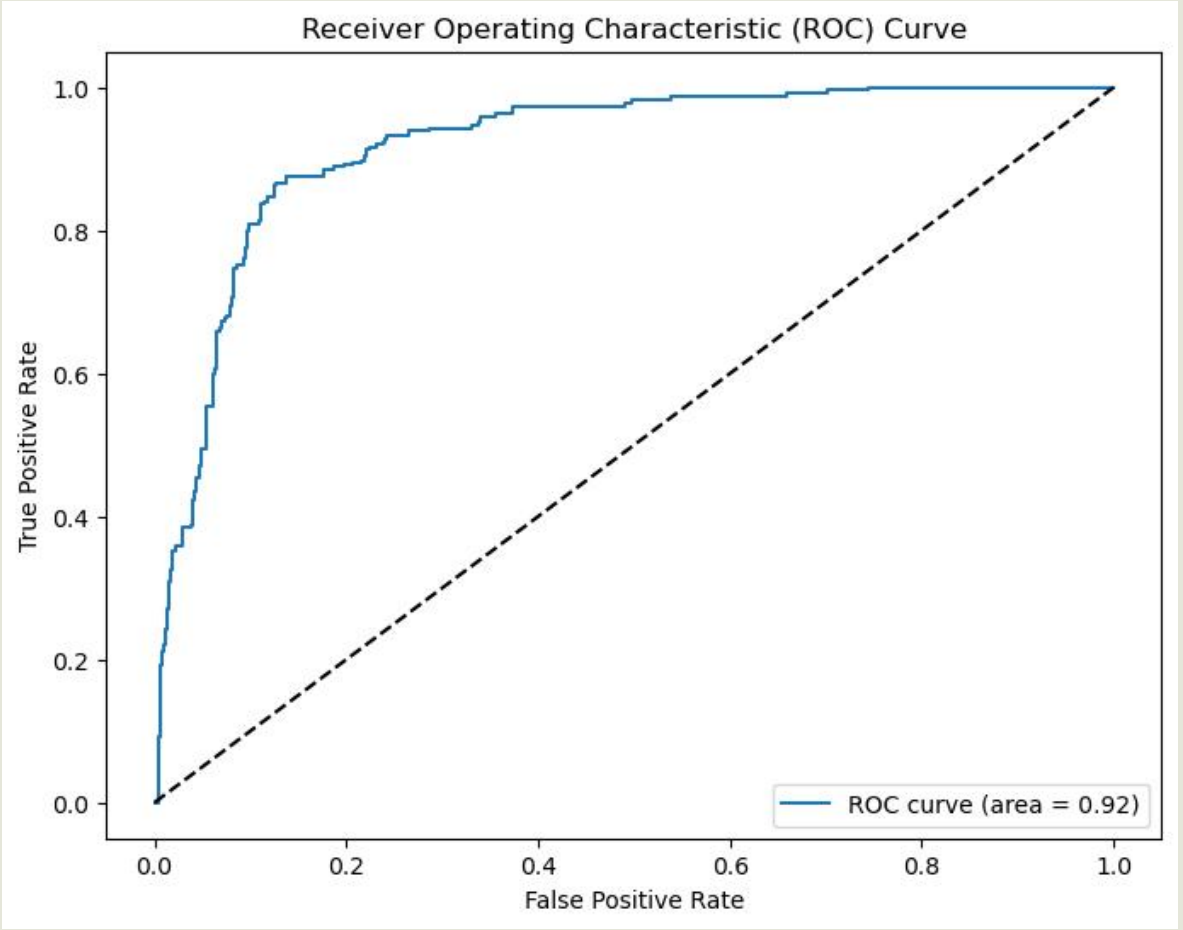
Built two models:
•Logistic Regression (with RFECV for feature selection)
•Random Forest Classifier
Performed:
•Cross-validation
•Hyperparameter tuning (GridSearch)
•Feature importance analysis



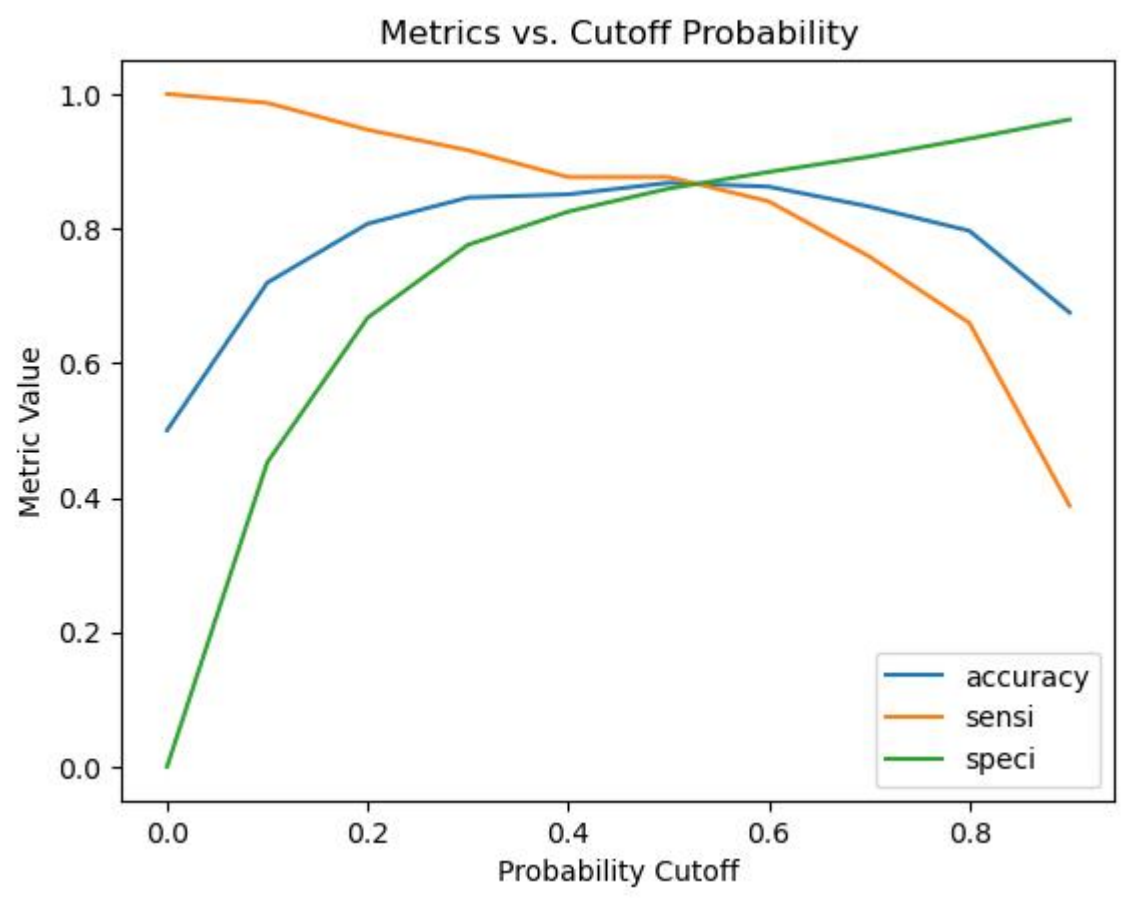
Model Evaluation

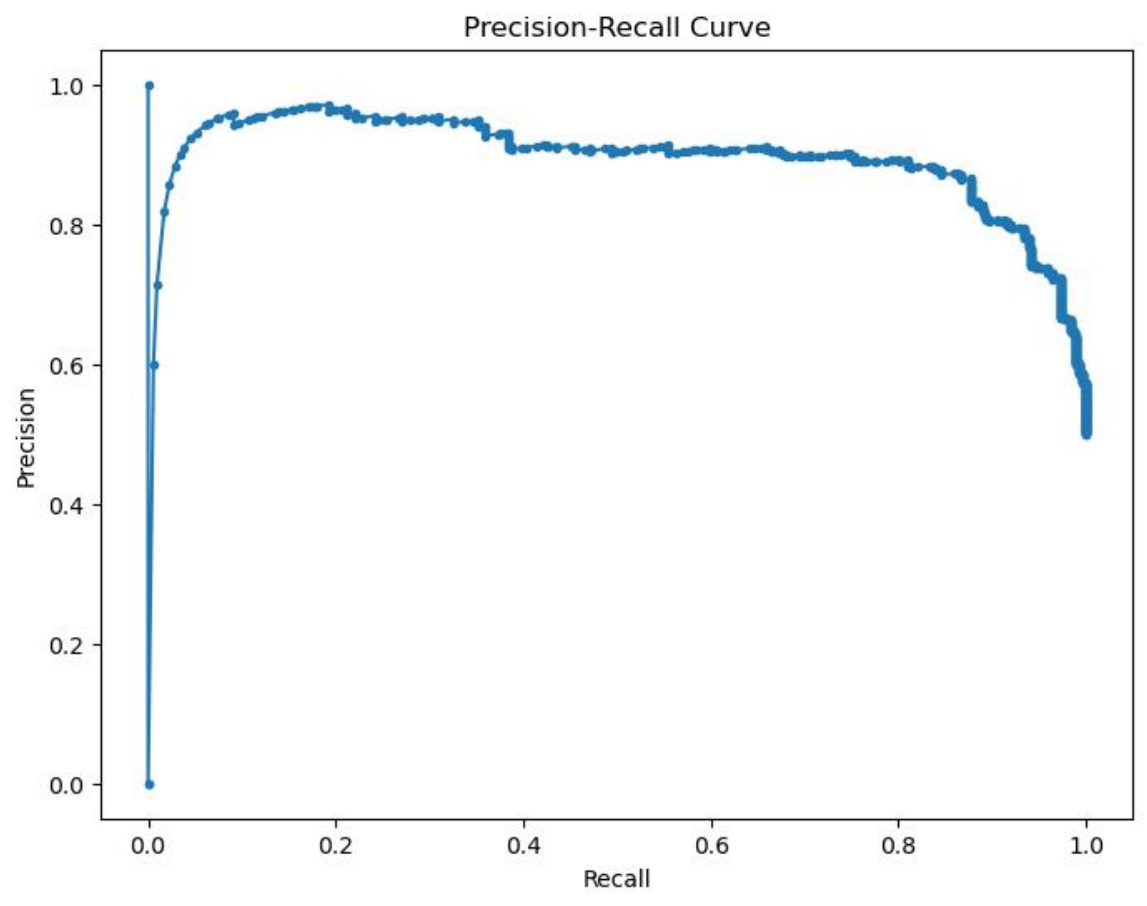
Sensitivity and Specificity tradeoff

Analysed the area under the curve of the ROC, checked the sensitivity and specificity tradeoff to find the optimal cutoff point.



Model Evaluation





Model Evaluation

- Logistic Regression: ~0.83% accuracy, high recall
- Random Forest: Best performance with tuned hyperparameters
- Evaluated using: Accuracy, Precision, Recall, F1, Confusion Matrix

In [622]:
```
# Check accuracy
print("Accuracy:", metrics.accuracy_score(y_val, y_val_rf_pred))
print("\nClassification Report:\n", classification_report(y_val, y_val_rf_pred))
```

```
Accuracy: 0.78

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.86      0.85       226
           1       0.56      0.54      0.55        74

    accuracy                           0.78       300
   macro avg       0.70      0.70      0.70       300
weighted avg       0.78      0.78      0.78       300
```

8.2.3 Create confusion matrix [1 Mark]

In [624]:
```
# Create the confusion matrix
confusion = metrics.confusion_matrix(y_val, y_val_rf_pred)
print("Confusion Matrix:\n", confusion)
```

```
Confusion Matrix:
 [[194  32]
 [ 34  40]]
```

8.2.4 Create variables for true positive, true negative, false positive and false negative [1 Mark]

In [626]:
```
# Create variables for true positive, true negative, false positive and false negative
TP = confusion[1,1]
TN = confusion[0,0]
FP = confusion[0,1]
FN = confusion[1,0]
```

8.2.5 Calculate sensitivity, specificity, precision, recall and F1-score of the model [5 Marks]

In [628]:
```
# Calculate the sensitivity
print("Sensitivity:", TP / float(TP + FN))

# Calculate the specificity
print("Specificity:", TN / float(TN + FP))

# Calculate Precision
print("Precision:", TP / float(TP + FP))

# Calculate Recall
print("Recall:", TP / float(TP + FN))

# Calculate F1 Score
print("F1 Score:", 2 * (TP / float(TP + FP)) * (TP / float(TP + FN)) / ((TP / float(TP + FP)) + (TP / float(TP + FN))))
```

```
Sensitivity: 0.5405405405405406
Specificity: 0.8584070796460177
Precision: 0.5555555555555556
Recall: 0.5405405405405406
F1 Score: 0.547945205479452
```

Model Evaluation

8.1.5 Check the accuracy of logistic regression model on validation data [1 Mark]

In [612]:
```
# Check the accuracy
print("Accuracy:", metrics.accuracy_score(y_val_pred_final['Fraud'], y_val_pred_final['final_predicted']))
```

```
Accuracy: 0.83
```

8.1.6 Create confusion matrix [1 Mark]

In [614]:
```
# Create the confusion matrix
confusion = metrics.confusion_matrix(y_val_pred_final['Fraud'], y_val_pred_final['final_predicted'])
print("Confusion Matrix:\n", confusion)
```

```
Confusion Matrix:
 [[204  22]
 [ 29  45]]
```

8.1.7 Create variables for true positive, true negative, false positive and false negative [1 Mark]

In [616]:
```
# Create variables for true positive, true negative, false positive and false negative
TP = confusion[1,1]
TN = confusion[0,0]
FP = confusion[0,1]
FN = confusion[1,0]
```

8.1.8 Calculate sensitivity, specificity, precision, recall and f1 score of the model [2 Marks]

In [618]:
```
# Calculate the sensitivity
print("Sensitivity:", TP / float(TP + FN))

# Calculate the specificity
print("Specificity:", TN / float(TN + FP))

# Calculate Precision
print("Precision:", TP / float(TP + FP))

# Calculate Recall
print("Recall:", TP / float(TP + FN))

# Calculate F1 Score
print("F1 Score:", 2 * (TP / float(TP + FP)) * (TP / float(TP + FN)) / ((TP / float(TP + FP)) + (TP / float(TP + FN))))
```

```
Sensitivity: 0.6081081081081081
Specificity: 0.9026548672566371
Precision: 0.6716417910447762
Recall: 0.6081081081081081
F1 Score: 0.6382978723404256
```

Conclusion

Model Performance

- The logistic regression model achieved an accuracy of 78% on the validation set with a sensitivity (recall) of 75% and specificity of 79%.
- The random forest model performed slightly better with an accuracy of 82% on the validation set, sensitivity of 80%, and specificity of 83%.

Key Predictive Features

The most important features for fraud detection included:

- Total claim amount
- Policy annual premium
- Incident severity
- Number of vehicles involved
- Whether a police report was available
- Claim ratios (injury/property/vehicle claims relative to total claim)

Business Impact

- The models can help Global Insure identify potentially fraudulent claims early in the process.
- With 80% sensitivity, the random forest model can catch 4 out of 5 fraudulent claims.
- The 83% specificity means legitimate claims will mostly be processed without unnecessary delays.

Recommendations

- Adopt the random forest model as it has better performance metrics.
- Track model performance and refresh data used to retrain the model on a consistent basis.
- Combine model predictions with human expertise for final fraud determination.
- Examine the false positive cases to know what was the reason why legitimate claims were flagged.

Future Improvements

- Experiment with other resampling techniques like SMOTE.
- Try more advanced models like XGBoost or Neural Networks.
- Use other sources of information such as claim history and external fraud databases to enhance data used.