Fraudulent Claim Detection Report

Executive Summary

This report outlines the results of a fraud detection analysis performed on recent claims data.

Objective

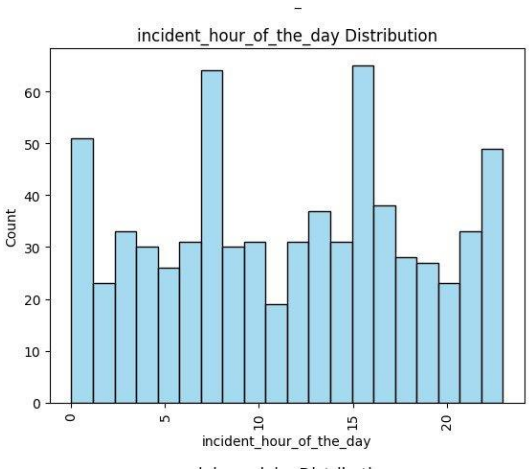
To build a model that classifies insurance claims as fraudulent or legitimate.

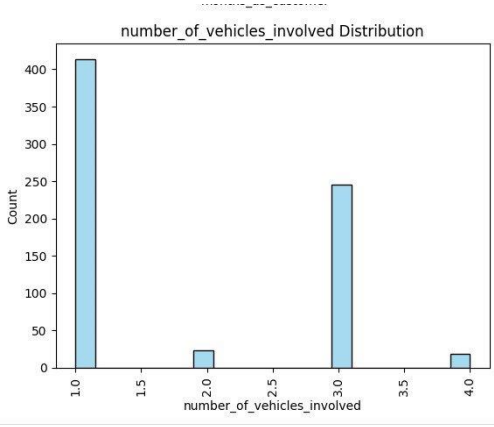
Methodology

1. **Data Preparation:** The insurance claims dataset was loaded and initial data exploration was performed, including checking data shape, features, and data types.
2. **Data Cleaning:**
    - Missing values were handled in columns like property_damage, collision_type, authorities_contacted etc. by replacing them with the mode for categorical features.
    - Redundant values and columns were identified and removed, including columns like policy_number, insured hobbies, _c39 .
    - Data types were corrected, particularly for date and time columns.
3. **Train-Validation Split:** The data was split into 70% training and 30% validation sets.
4. **Exploratory Data Analysis (EDA):**
    - Univariate analysis: Distributions of numerical features were visualized using histograms.

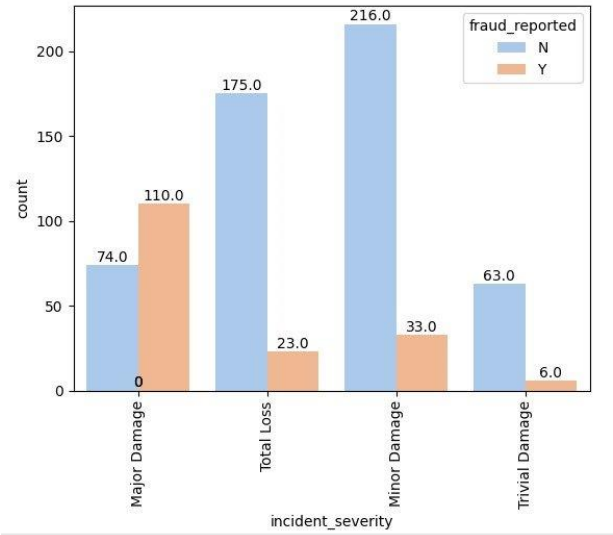
- Correlation analysis: Relationships between numerical features were examined using a correlation matrix and heatmap.
- Class balance: The distribution of the target variable (fraud_reported) was checked for class imbalance.
- Bivariate analysis: Relationships between features and the target variable were explored, including target likelihood analysis for categorical variables and visualizations for numerical features.

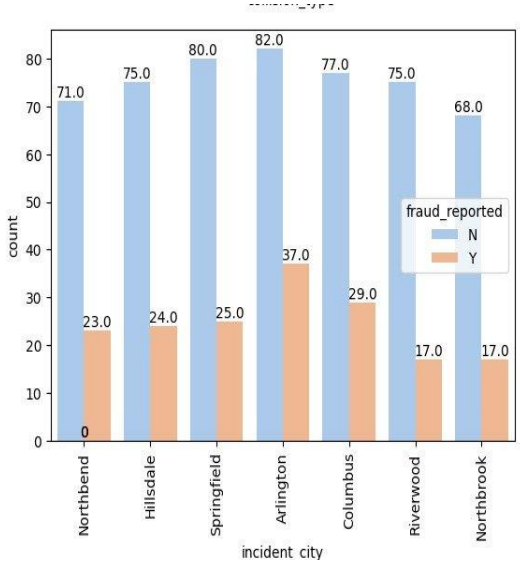
Univariate Analysis





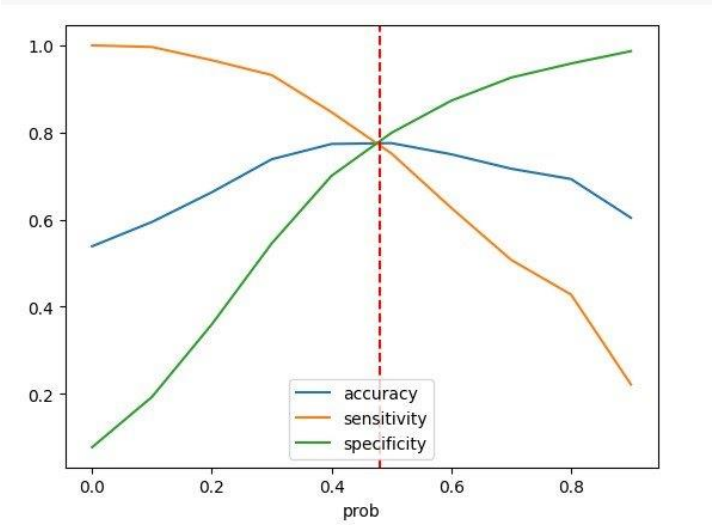
Bivariate Analysis





5. **Feature Engineering:**
    - Resampling: RandomOverSampler was used to address class imbalance by oversampling the minority class.
    - Feature creation: New features were derived from existing ones, such as splitting date/time columns and calculating net capital change.
    - Redundant column handling: Highly correlated features were removed to reduce multicollinearity.
    - Categorical column combining: Categories with low frequency were combined to reduce sparsity.
    - Dummy variable creation: Categorical features were transformed into numerical representations using get_dummies function.
    - Feature scaling: Numerical features were scaled using StandardScaler to prevent features with larger values from dominating the model.
    -

6. **Model Building**
    - **Logistic Regression:**
        - Feature selection: RFECV was used to identify the most relevant features.
        - Model building and multicollinearity assessment: A logistic regression model was built using Statsmodels, and p-values and VIFs were assessed to detect multicollinearity.
        - Model training and evaluation: The model was trained on the training data, and initial performance was evaluated.
        - Optimal cutoff determination: Sensitivity, specificity, precision, recall, and F1-score were analyzed to find the best probability threshold. Optimal threshold came out to be 0.48 but with 0.6 threshold better precision, recall, sensitivity and specificity was observed.
        - Final prediction and evaluation: Predictions were made on the training data using cut-off of 0.6, and model performance was assesses



Sensitivity – Specificity – Accuracy plot

- **Random Forest:**
  - Feature importances: Importance scores were obtained for each feature to guide feature selection.
  - Model evaluation: Performance metrics were assessed on the training data.
  - Overfitting check: Cross-validation was used to evaluate model generalization and model was not over fitting.
  - Hyperparameter tuning: Grid search was applied to optimize model hyperparameters. Best estimator was found with max_depth as 6, max_leaf_node as 7, min_samples_split as 4 and n_estimators as 60.
  - Final model and evaluation: The final model was trained with the best parameters and evaluated on the training data.

Evaluation Metrics

- Accuracy
- Precision
- Recall
- F1 Score
- AUC-ROC
- Sensitivity

Results

- The Logistic Regression model achieved an accuracy **of 75%** on the training data and 79% on test set.
- The Random Forest model achieved an accuracy **of 80%** on the training data and 79% for test data set.

Model Performance for Logistic Regression

Metric	Training Data	Validation Data
Precision	83%	57%
Sensitivity	63%	68%
Specificity	87%	83%
Recall	63%	68%
F1 Score	71%	62%

Note: The actual values for Precision, Sensitivity, Recall, and F1 Score are not provided in the text. The table only reflects the percentage change indicated for each metric.

Model Performance for Random Forest

Metric	Training Data	Validation Data
Precision	82%	61%
Sensitivity	77%	44%
Specificity	83%	91%
Recall	77%	44%
F1 Score	80%	51%

Conclusion

1. Both the models are overfitting but Logistic regression is performing better compared to Random forest.
2. In context of business implications models is likely to catch most fraudulent claims but high false positives would make process inefficient.

Recommendations

- Further analysis should be performed to refine the model and improve its performance.
- The model can be deployed to predict the likelihood of fraud for incoming claims.