FRAUDULENT CLAIM DETECTION

Bhawani Shankar Mahapatra & Biniyam Belayneh Demisse

Problem Statement 3

Business Objective 3

1. Data Preparation 3

1.0 Import Libraries & DATA LOAD 3

2. Data Cleaning 3

2.1 Handle null values 3

2.2 Identify and handle redundant values and columns 4

2.3 Fix Data Types 6

3. Train-ValidationSplit 6

4. EDA on training data 6

4.1 Perform univariate analysis 6

4.2 Perform correlation analysis 8

4.3 Check class balance 9

4.4 Perform bivariate analysis 10

5. EDA on validation data 16

5.1 Perform univariate analysis 16

5.2 Perform correlation analysis 18

5.3 Check class balance 19

5.4 Perform bivariate analysis 20

6. Feature Engineering 26

6.1 Perform resampling 26

6.2 Feature Creation 26

6.3 Handle redundant columns 27

6.4 Combine values in Categorical Columns 27

6.5 Dummy variable creation 27

6.6 Feature scaling 28

7. Model Building 28

7.1 Feature selection 28

7.2 Build Logistic Regression Model 29

7.3 Find the Optimal Cutoff 32
7.4 Build Random Forest Model 35
7.5 Hyperparameter Tuning 37
8. Prediction and Model Evaluation 38
8.1 Make predictions over validation data using logistic regression model 38
8.2 Make predictions over validation data using random forest model 39
Evaluation and Conclusion 41
Validation Performance 41
Model Insights 41
Key Findings 41
Conclusion 42
Recommendations 42

PROBLEM STATEMENT

Global Insure, a leading insurance company, processes thousands of claims annually. However, a significant percentage of these claims turn out to be fraudulent, resulting in considerable financial losses. The company's current process for identifying fraudulent claims involves manual inspections, which is time-consuming and inefficient. Fraudulent claims are often detected too late in the process, after the company has already paid out significant amounts. Global Insure wants to improve its fraud detection process using data-driven insights to classify claims as fraudulent or legitimate early in the approval process. This would minimise financial losses and optimise the overall claims handling process.

BUSINESS OBJECTIVE

Global Insure wants to build a model to classify insurance claims as either fraudulent or legitimate based on historical claim details and customer profiles. By using features like claim amounts, customer profiles and claim types, the company aims to predict which claims are likely to be fraudulent before they are approved.

1. DATA PREPARATION

1.0 IMPORT LIBRARIES & DATA LOAD

All the necessary libraries like pandas, seaborn, matplotlib, scikit learn is imported.

The data set was loaded from insurance_claims.csv. The dataset has 1000 records and 40 columns.

2. DATA CLEANING

2.1 HANDLE NULL VALUES

The dataset is largely complete, with most features showing no missing values (0%).

Only two columns contain missing data:

- authorities_contacted has 91 missing values (~9.1%), which may need imputation or careful handling depending on its importance.
- _c39 has 100% missing values, indicating it is likely an empty or placeholder column and can be safely dropped.

Overall, the dataset has a strong level of completeness, minimizing the need for extensive missing data treatment.

2.2 IDENTIFY AND HANDLE REDUNDANT VALUES AND COLUMNS

2.2.1 Examine the columns to determine if any value or column needs to be treated

High-cardinality features: Columns such as policy_number (1000 unique values), policy_annual_premium (991), insured_zip (995), incident_location (1000), and various claim amount fields (total_claim_amount, injury_claim, property_claim, vehicle_claim) exhibit very high uniqueness. These are either identifiers or continuous/numeric values that may not be directly useful as categorical predictors. Some, like policy_number and incident_location, appear to be **unique identifiers** and should be excluded from modeling to avoid overfitting.

Moderate-cardinality features: Columns such as months_as_customer (391), capital-gains (338), and capital-loss (354) show moderate uniqueness, suggesting they are numeric variables with a range of values. These can be treated as continuous features in modeling.

Low-cardinality categorical features: Many columns have fewer than 100 unique values, making them categorical variables suitable for encoding. Examples include:

- **Demographics**: insured_sex (2), insured_education_level (7), insured_occupation (14), insured_hobbies (20), insured_relationship (6).
- **Policy-related**: policy_state (3), policy_csl (3), policy_deductable (3), umbrella_limit (11).
- **Incident-related**: incident_type (4), collision_type (4), incident_severity (4), authorities_contacted (5), incident_state (7), incident_city (7), incident_hour_of_the_day (24), number_of_vehicles_involved (4), property_damage (3), bodily_injuries (3), witnesses (4), police_report_available (3).
- **Vehicle-related**: auto_make (14), auto_model (39), auto_year (21).

Target variable: fraud_reported has **2 classes (Yes/No)**, showing a class imbalance (753 legitimate vs 247 fraud). This imbalance should be considered during model training, possibly using resampling methods.

Redundant column: _c39 contains **0 unique values** and is therefore an empty/placeholder column that should be dropped.

2.2.2 Identify and drop any columns that are completely empty

_C39 column is empty Hence removed .

2.2.3 Identify and drop rows where features have illogical or invalid values

- **Always positive (must be > 0)**:

1. months_as_customer (can be 0, but not negative)
2. age (≥ 0, realistically ≥ 18 if this is insured person)
3. policy_deductable (must be ≥ 0)
4. policy_annual_premium (≥ 0)
5. umbrella_limit (can be 0 or positive, but negative like -1000000 is invalid → drop)
6. capital-gains (should be ≥ 0)

7. capital-loss (should be ≥ 0)
8. incident_hour_of_the_day (valid range: 0–23)
9. number_of_vehicles_involved (≥ 1, small integer)
10. bodily_injuries (≥ 0, small integer)
11. witnesses (≥ 0, small integer)
12. total_claim_amount, injury_claim, property_claim, vehicle_claim (≥ 0)

Special cases:

1. policy_number, insured_zip → IDs, not really useful for "illogical check"
2. auto_year → should be within a valid range (e.g., 1980–current year).

2.2.4 Identify and remove columns where a large proportion of the values are unique or near-unique

Removing high-cardinality columns (columns with unique values ≥ 900), since they don't add much predictive power and may even cause overfitting if treated as categorical.
High cardinality columns (≥ 900 unique values):

- policy_number → 1000 unique
- policy_bind_date → 951 unique
- policy_annual_premium → 991 unique
- insured_zip → 995 unique
- incident_location → 1000 unique
- Some high-cardinality columns like policy_annual_premium and policy_bind_date actually hold useful information, so they should stay. Others like IDs or near-unique location fields can be safely dropped.
- Here's the final observation and action table for clarity:

Column	Unique Values	Observation	Action
policy_number	1000	Pure identifier, no predictive value	❌ Drop
policy_bind_date	951	Temporal feature, can capture patterns	✅ Keep
policy_annual_premium	991	Important numerical predictor	✅ Keep
insured_zip	995	Too granular, high cardinality, less useful	❌ Drop
incident_location	1000	Too granular, acts like an ID, not useful	❌ Drop

2.3 FIX DATA TYPES

converting policy_bind_date and incident_date into **datetime** ensures they can be used for temporal feature engineering.

3. TRAIN-VALIDATIONSPLIT

3.1 Import required libraries

Using scikit learn module train_test_split library for split purpose.

3.2 Define feature and target variables

- X now contains **all independent variables** (policy, customer, claim, vehicle, etc.).
- y contains the **dependent variable** (fraud_reported), which is binary (Y/N).

3.3 Split the data

- Training set shape: (700, 35)
- Validation set shape: (300, 35)
- Training target shape: (700,)
- Validation target shape: (300,)

4. EDA ON TRAINING DATA

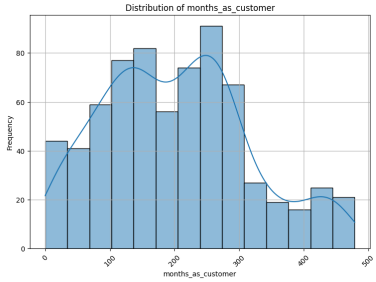
4.1 PERFORM UNIVARIATE ANALYSIS

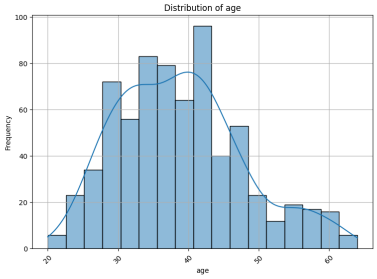
Identify and select numerical columns from training data for univariate analysis

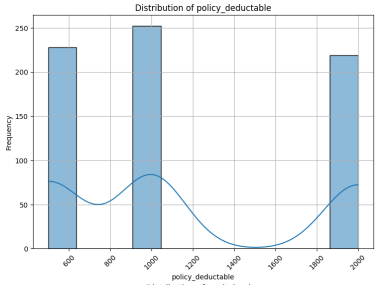
Below field are numerical fields hence used for the analysis

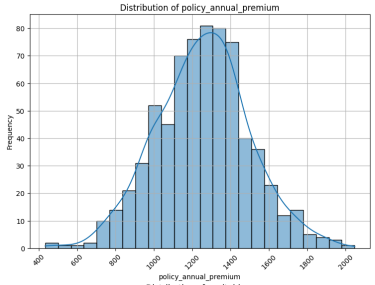
- **Customer-related**:months_as_customer, age
- **Policy-related**:policy_deductable, policy_annual_premium, umbrella_limit
- **Financial features**:capital-gains, capital-loss
- **Incident-related**: incident_hour_of_the_day, number_of_vehicles_involved, bodily_injuries, witnesses
- **Claims**:total_claim_amount, injury_claim, property_claim, vehicle_claim
- **Vehicle info**:auto_year

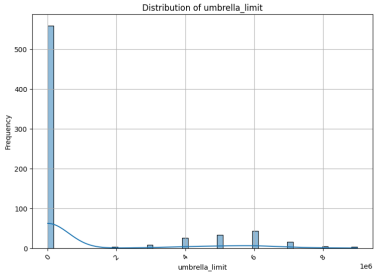
4.1.2 Visualise the distribution of selected numerical features using appropriate plots to understand their characteristics

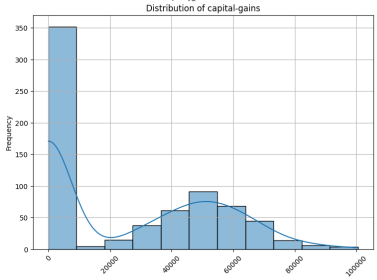


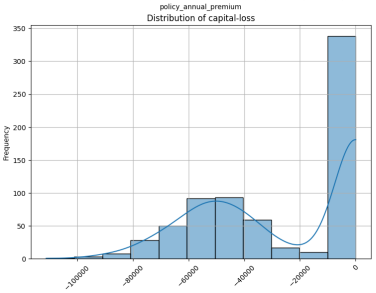


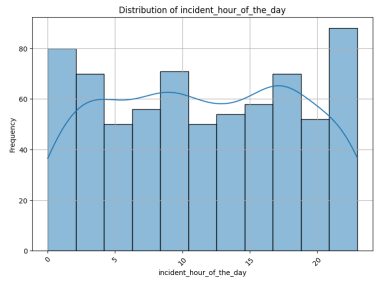


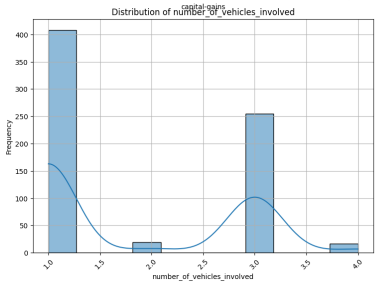


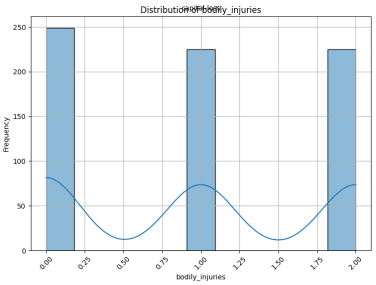


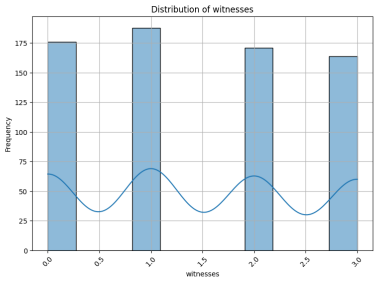


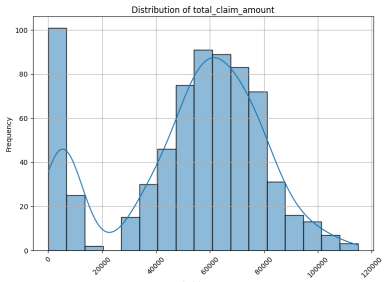


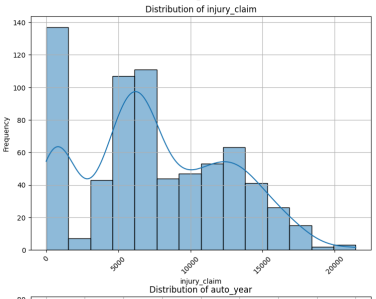


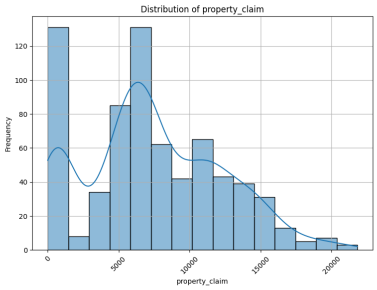


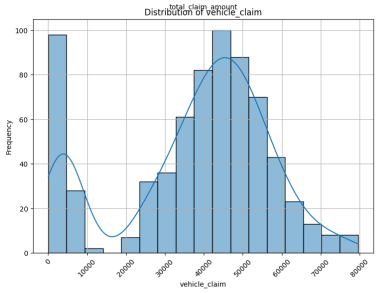


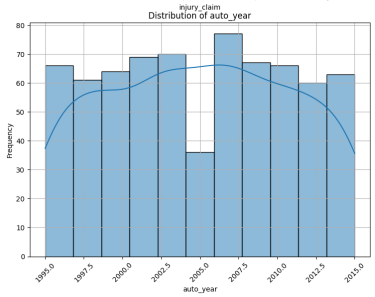






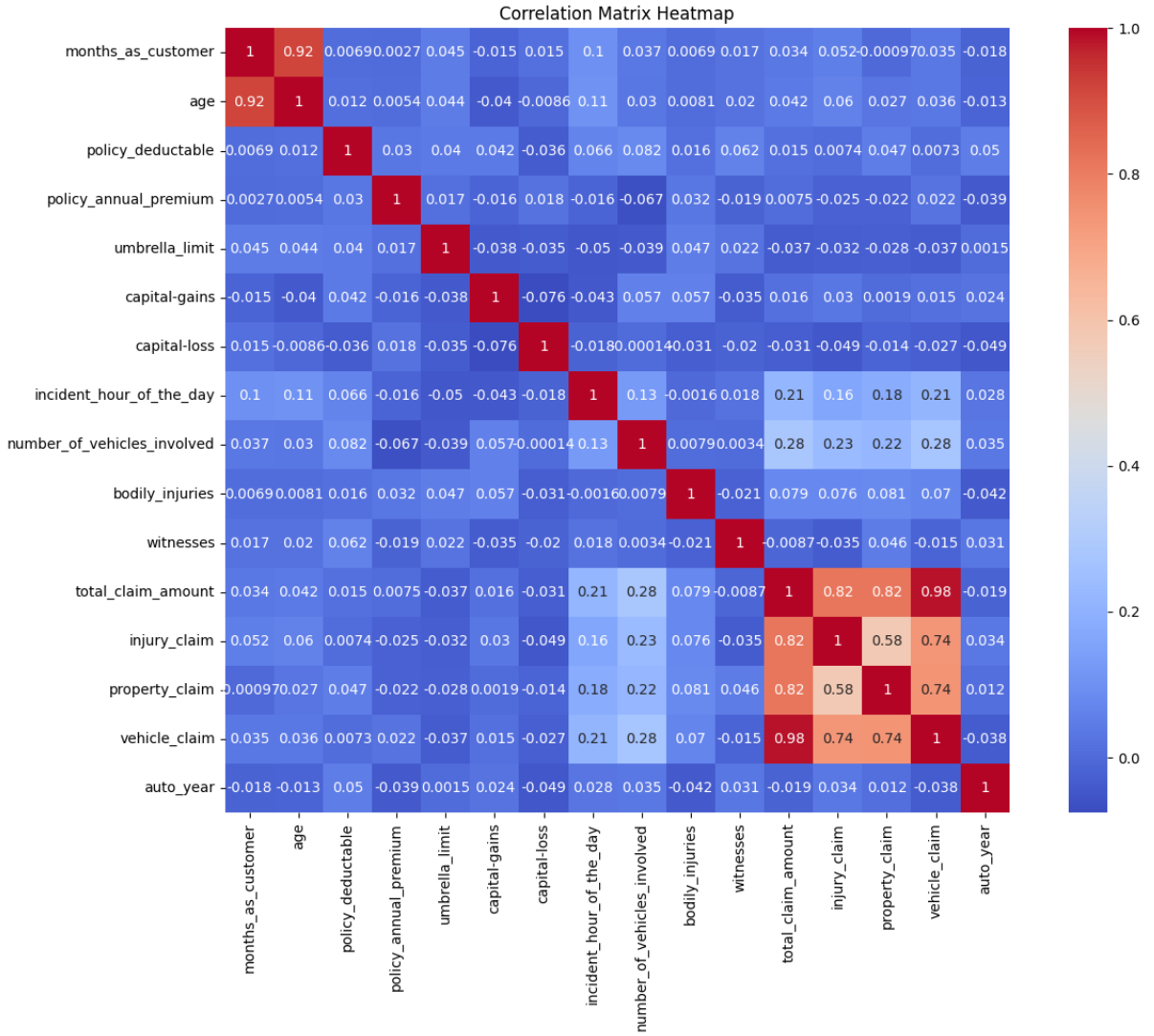






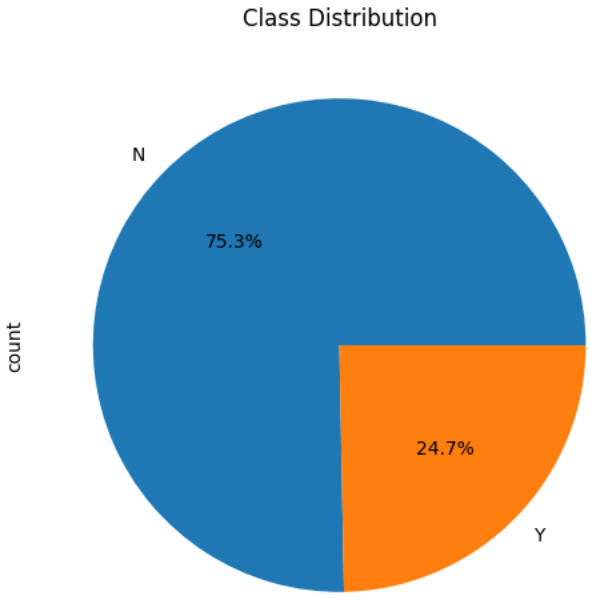
- Some features are **continuous and well-behaved**: age, policy_annual_premium, total_claim_amount.
- Some are **categorical disguised as numeric**: policy_deductable, number_of_vehicles_involved, bodily_injuries, witnesses.
- Some are **heavily skewed**: umbrella_limit, capital-gains, capital-loss, claims (injury_claim, property_claim, vehicle_claim).
- Feature engineering opportunities:
- Convert discrete numerics → categorical (policy_deductable, etc.).
- Apply log/box-cox transforms for skewed variables.
- Add binary flags (e.g., **has_capital_gain**, **has_capital_loss**).

4.2 PERFORM CORRELATION ANALYSIS



Observation: The correlation heatmap reveals several important insights. **Months as customer** and **age** are highly correlated (0.92), indicating potential redundancy. Strong correlations are observed among claim-related variables: **total_claim_amount** has very high correlations with injury_claim (0.82)**, property_claim (0.82), and vehicle_claim (0.98), suggesting these features capture overlapping information. Other variables show weak correlations, meaning they likely provide independent predictive power. Policy_deductible, umbrella_limit, and incident_hour_of_the/_day have little correlation with others, indicating unique contributions. Overall, multicollinearity is a concern mainly within claim variables and age–customer duration. Feature selection or dimensionality reduction may be useful to reduce redundancy and improve model performance.

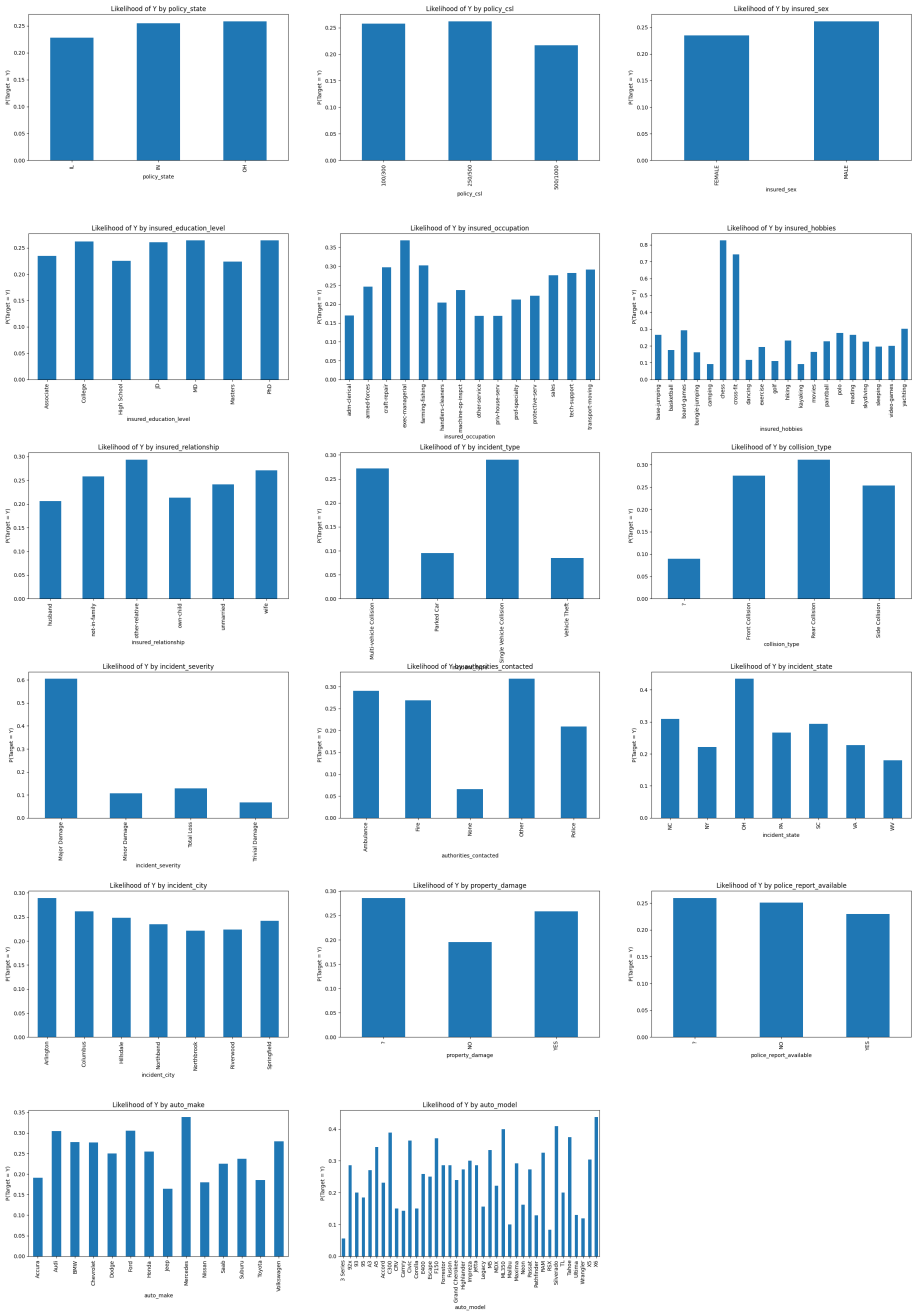
4.3 CHECK CLASS BALANCE



Observation: The class distribution analysis highlights a clear imbalance in the dataset. Approximately 75% of the records belong to the “No Fraud” (N) category, while only about 25% fall into the “Fraud Reported” (Y) category. This indicates that fraudulent cases are underrepresented, which can bias machine learning models toward predicting the majority class (non-fraud). If left unaddressed, the model may achieve high accuracy but perform poorly in detecting fraud, which is the critical objective. To improve fairness and predictive performance, resampling techniques like SMOTE, undersampling, or class-weight adjustments should be considered before training the model.

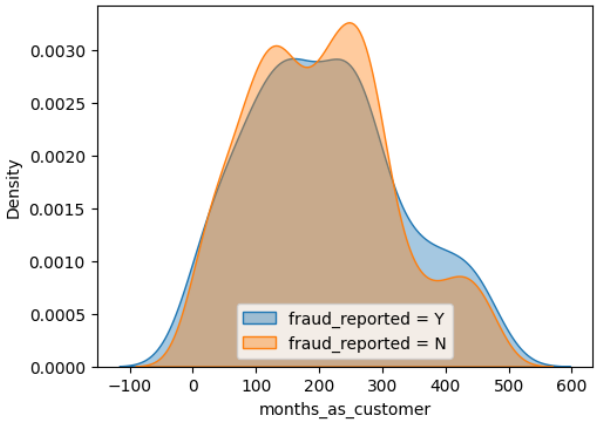
4.4 PERFORM BIVARIATE ANALYSIS

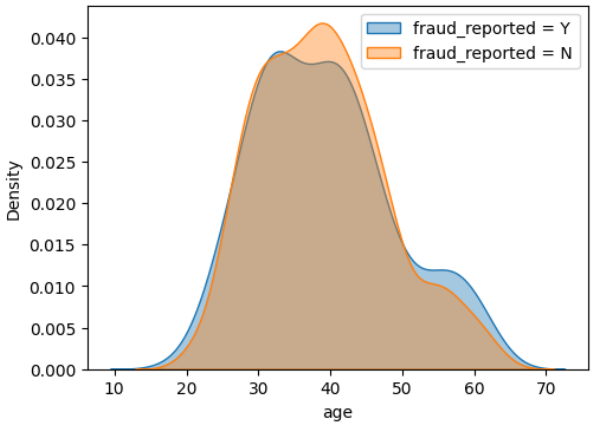
4.4.1Target likelihood analysis for categorical variables.

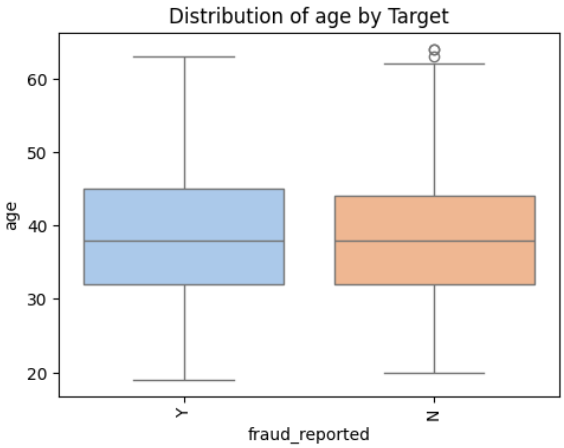


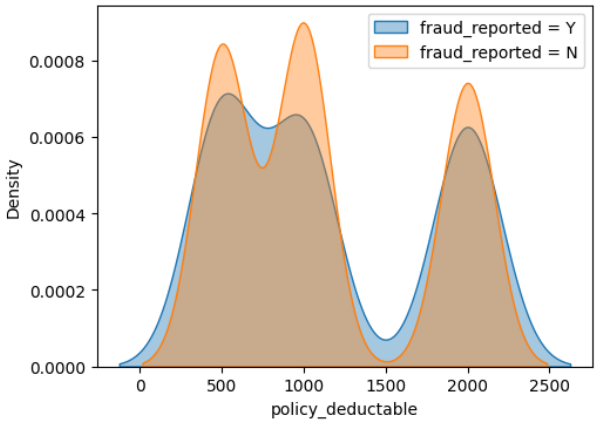
Observation: The categorical feature analysis reveals that variables such as insured_education_level, insured_sex, policy_state, policy_csl, and police_report_available contribute minimally in explaining the variation in the target variable (fraud_reported). Their predictive power appears weak, indicating they may not be influential drivers of fraud detection in this dataset. Retaining low-impact features may introduce noise and complexity without improving model performance. Hence, these attributes can either be dropped or deprioritized during feature selection, allowing the model to focus on stronger predictors like incident_type, incident_severity, collision_type, and insured_occupation, which show more relevance.

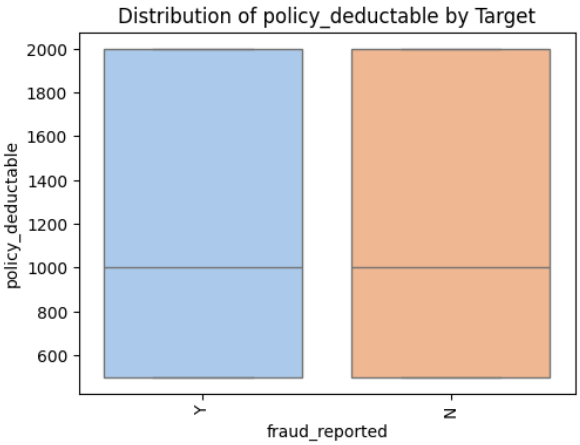
4.4.2Explore the relationships between numerical features and the target variable to understand their impact

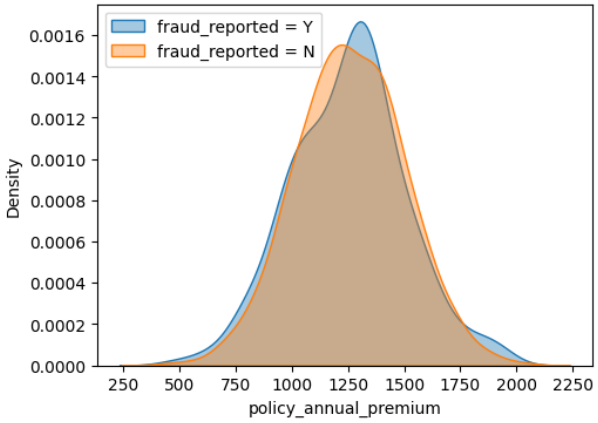


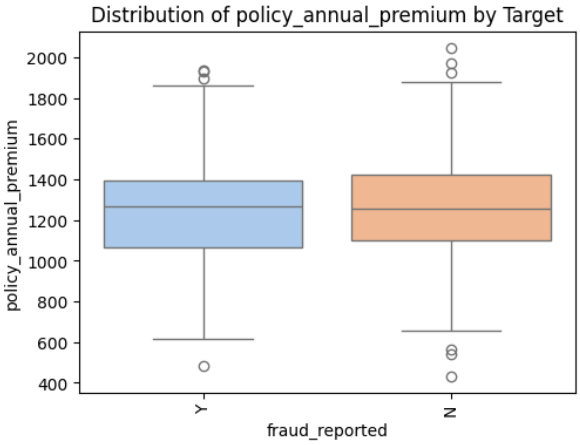


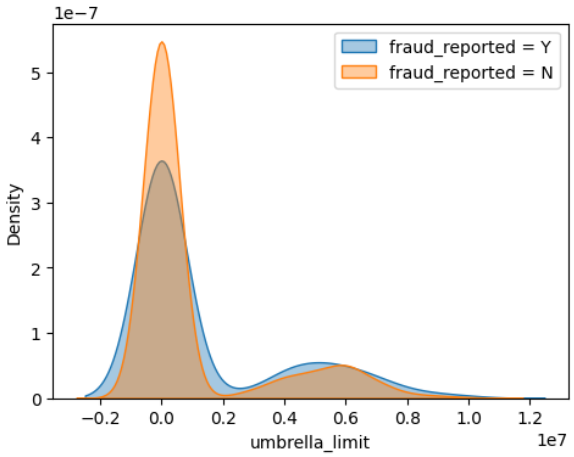


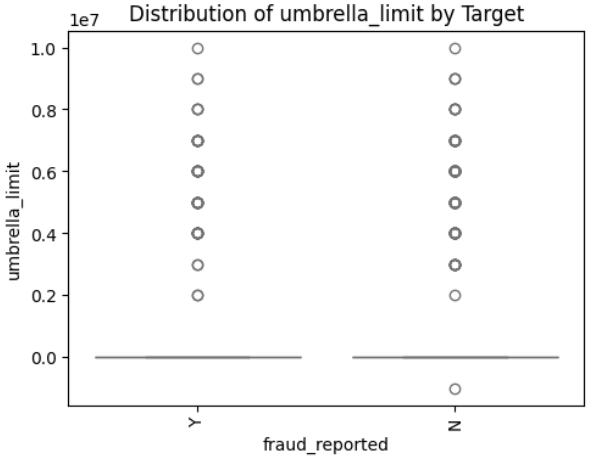

fraud_reported = Y
fraud_reported = N
Density
policy_deductable

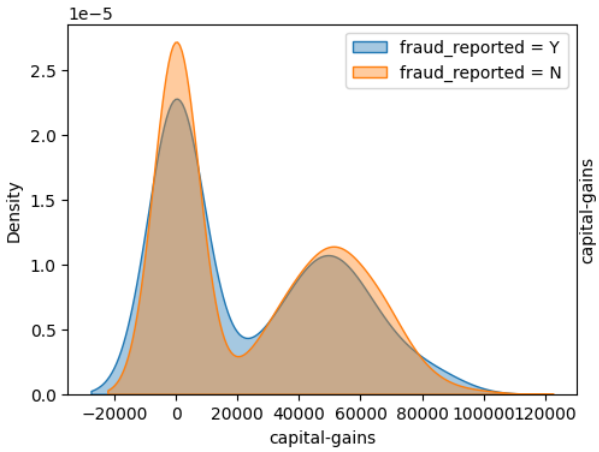

Distribution of policy_deductable by Target
policy_deductable
fraud_reported
Y
N


fraud_reported = Y
fraud_reported = N
Density
policy_annual_premium

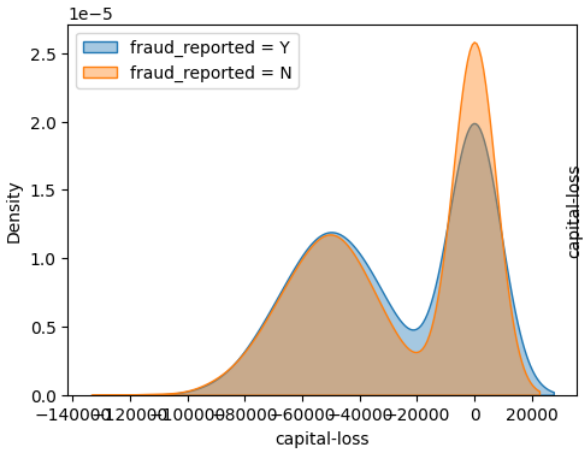

Distribution of policy_annual_premium by Target
policy_annual_premium
fraud_reported
Y
N

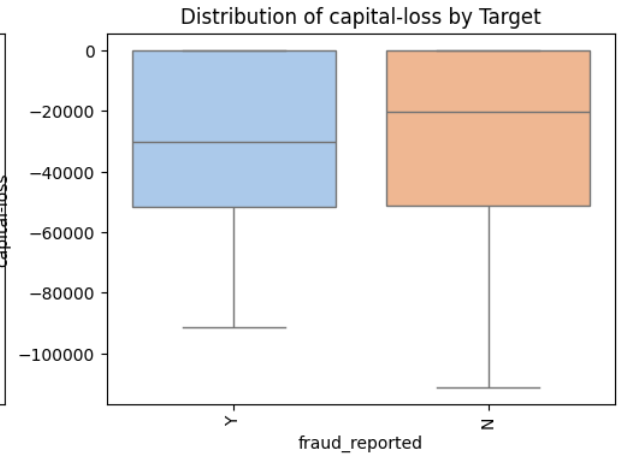

1e−7
fraud_reported = Y
fraud_reported = N
Density
umbrella_limit
1e7

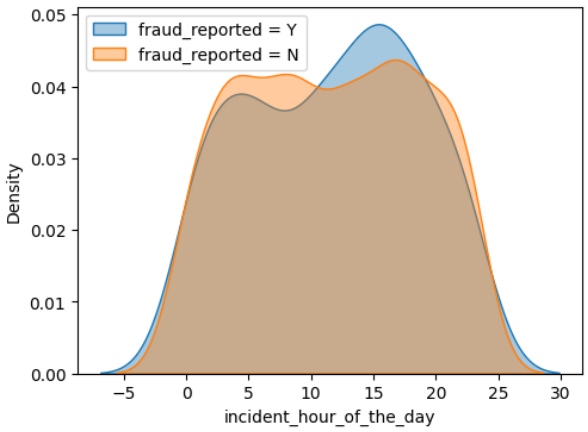

Distribution of umbrella_limit by Target
1e7
umbrella_limit
fraud_reported
Y
N

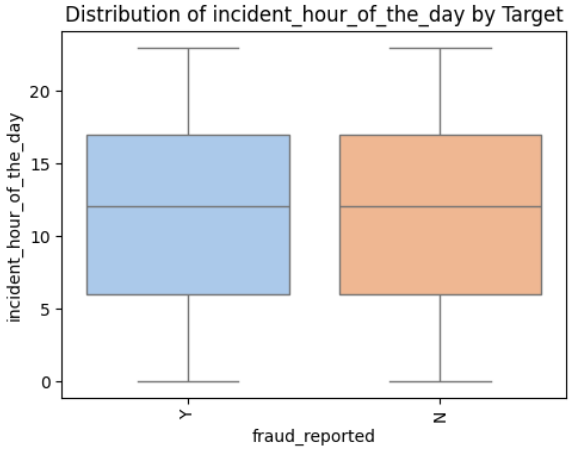

1e−5
fraud_reported = Y
fraud_reported = N
Density
capital-gains

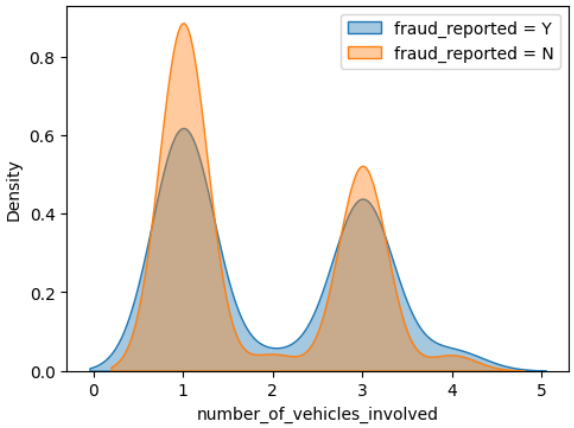

Distribution of capital-gains by Target
capital-gains
fraud_reported
Y
N

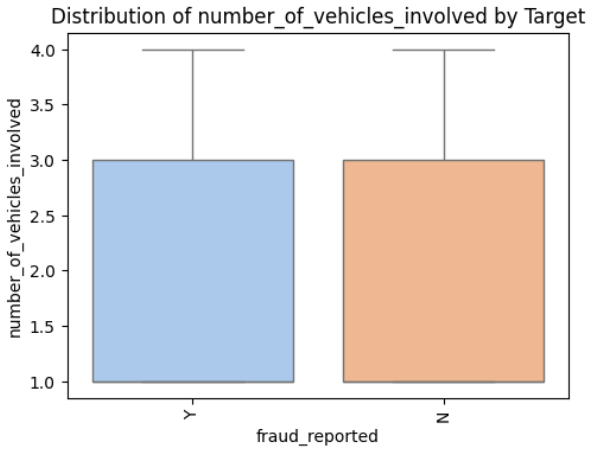

1e−5
fraud_reported = Y
fraud_reported = N
Density
capital-loss

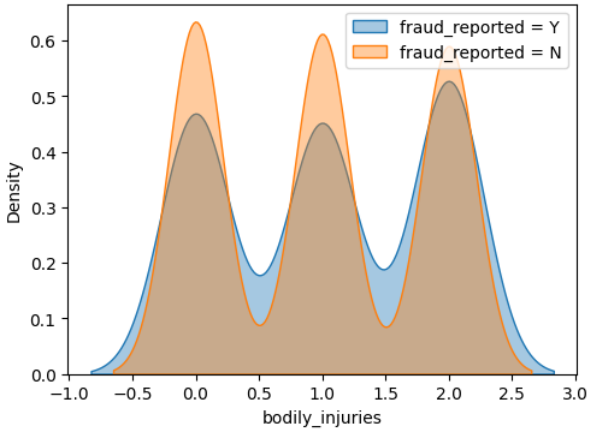

Distribution of capital-loss by Target
capital-loss
fraud_reported
Y
N

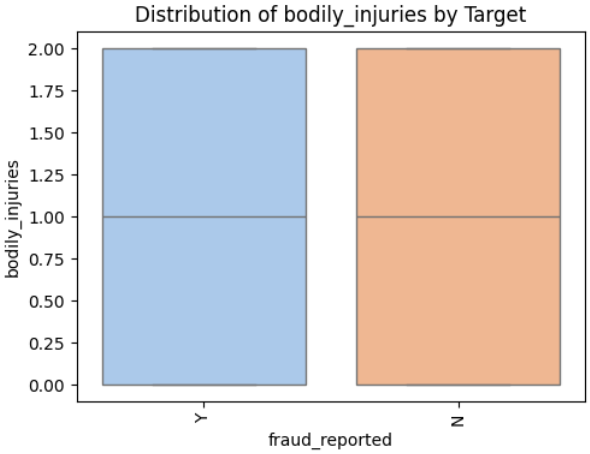

fraud_reported = Y
fraud_reported = N
Density
incident_hour_of_the_day

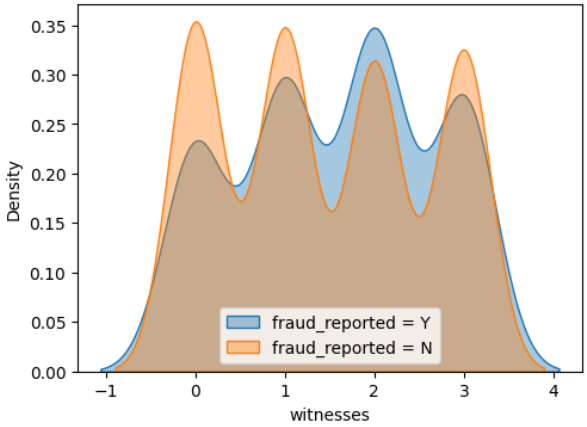

Distribution of incident_hour_of_the_day by Target
incident_hour_of_the_day
fraud_reported
Y
N

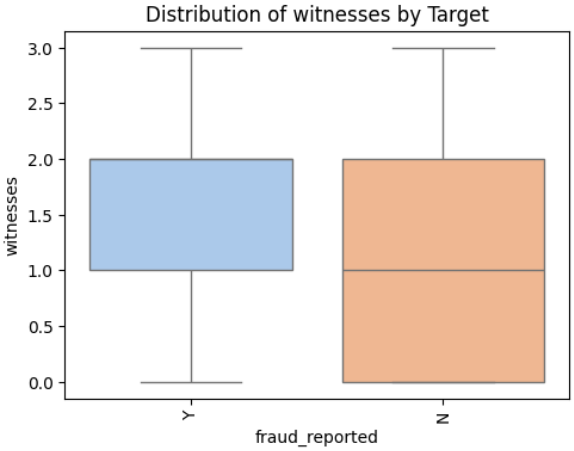

fraud_reported = Y
fraud_reported = N
Density
number_of_vehicles_involved

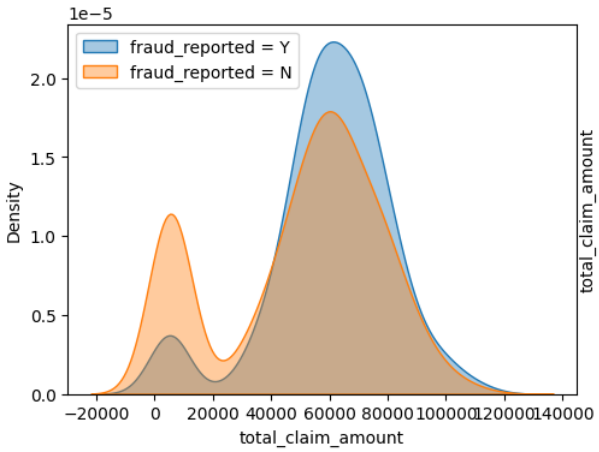

Distribution of number_of_vehicles_involved by Target
number_of_vehicles_involved
fraud_reported
Y
N


fraud_reported = Y
fraud_reported = N
Density
bodily_injuries

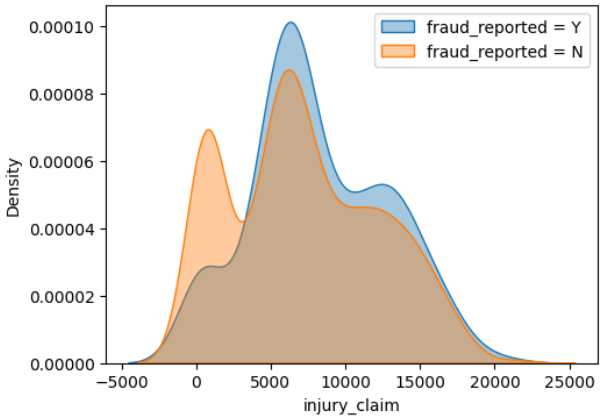

Distribution of bodily_injuries by Target
bodily_injuries
fraud_reported
Y
N

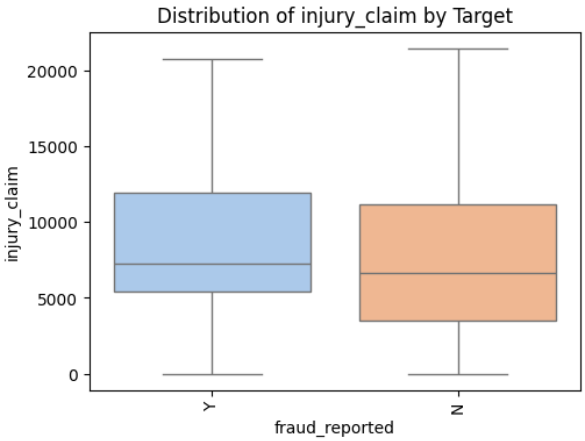

fraud_reported = Y
fraud_reported = N
Density
witnesses

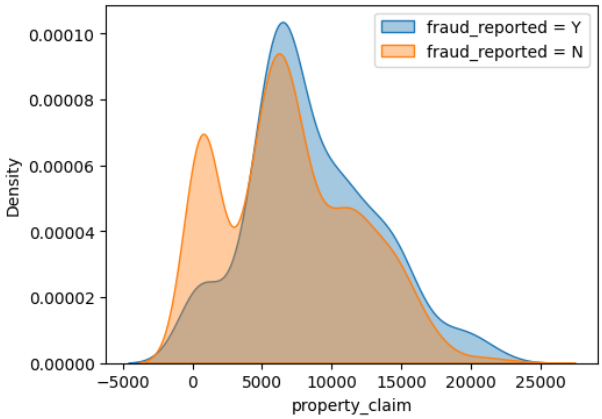

Distribution of witnesses by Target
witnesses
fraud_reported
Y
N


1e−5
fraud_reported = Y
fraud_reported = N
Density
total_claim_amount

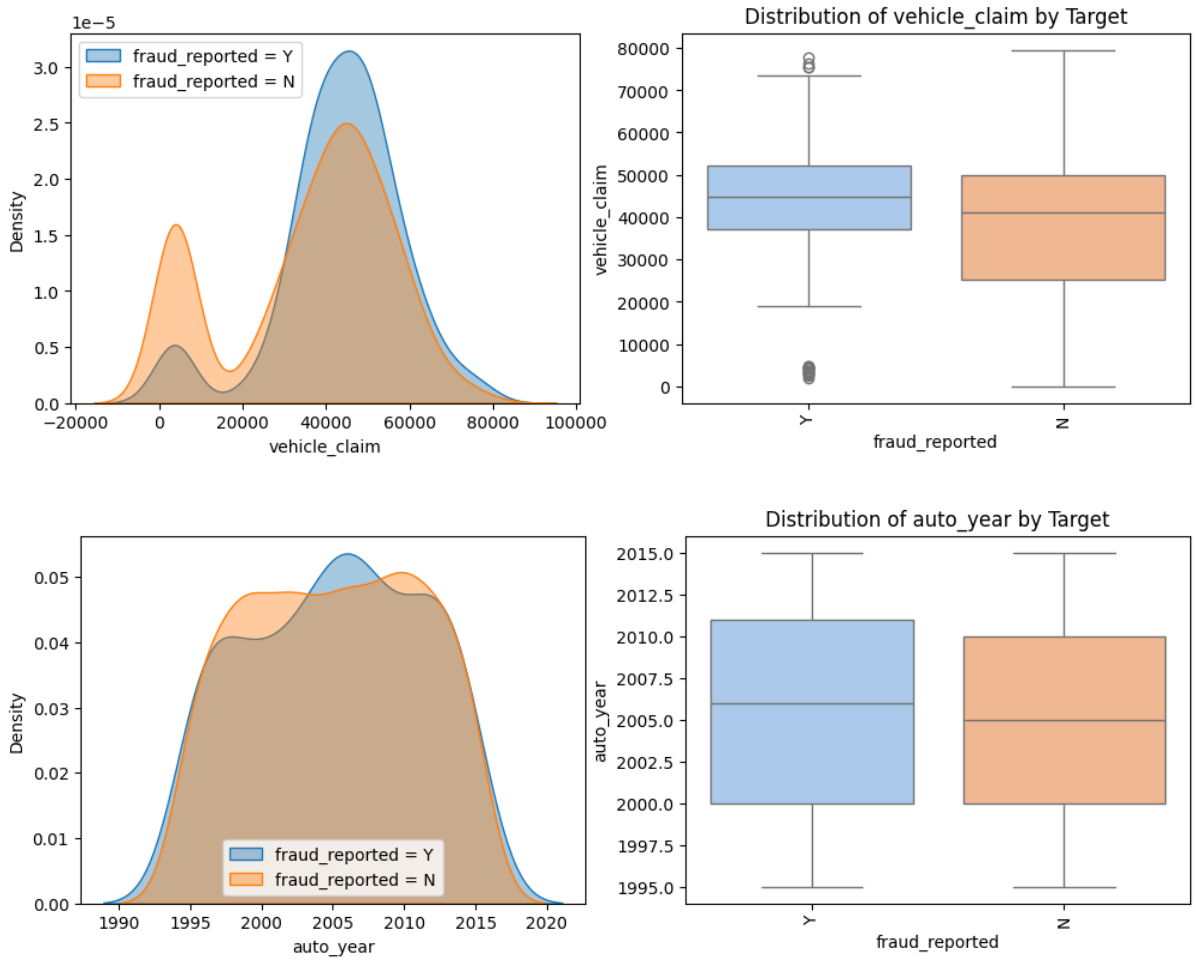

Distribution of total_claim_amount by Target
total_claim_amount
fraud_reported
Y
N


fraud_reported = Y
fraud_reported = N
Density
injury_claim


Distribution of injury_claim by Target
injury_claim
fraud_reported
Y
N


fraud_reported = Y
fraud_reported = N
Density
property_claim


Distribution of property_claim by Target
property_claim
fraud_reported
Y
N



Observation : The features months_as_customer, age, policy_deductible, policy_annual_premium, capital-gains, capital-loss, incident_hour_of_the_day, and auto_year show minimal contribution in explaining the variation of the target variable.

5. EDA ON VALIDATION DATA

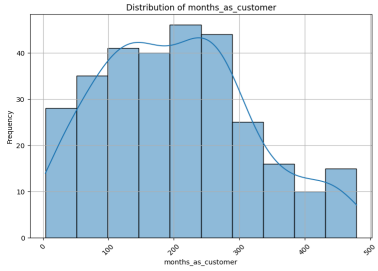
5.1 PERFORM UNIVARIATE ANALYSIS

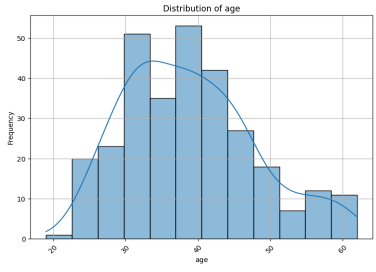
5.1.1 Identify and select numerical columns from training data for univariate analysis.

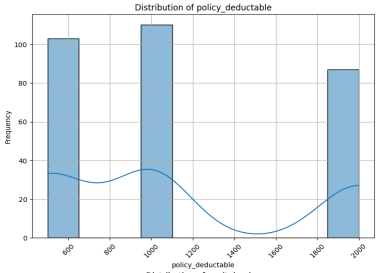
The numeric features extracted from the dataset include customer and policy details, financial variables, incident characteristics, claim-related information, and vehicle details. Specifically, variables such as **months_as_customer, age, policy_deductable, policy_annual_premium, umbrella_limit, capital-gains, capital-loss, incident_hour_of_the_day, number_of_vehicles_involved, bodily_injuries, witnesses, total_claim_amount, injury_claim, property_claim, vehicle_claim, and auto_year** are selected. These features capture important quantitative aspects of insurance records and claims. Since the values vary widely in scale (e.g., premiums versus number of witnesses), preprocessing through

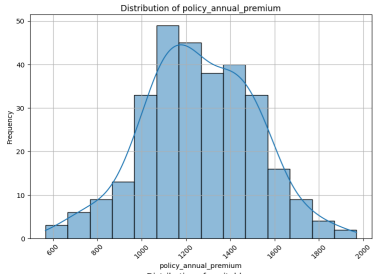
scaling or normalization is essential before applying machine learning models. Proper handling ensures balanced feature contribution, reduces bias, and improves model performance in fraud detection.

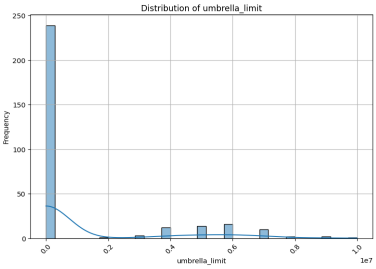
5.1.2 Visualise the distribution of selected numerical features using appropriate plots

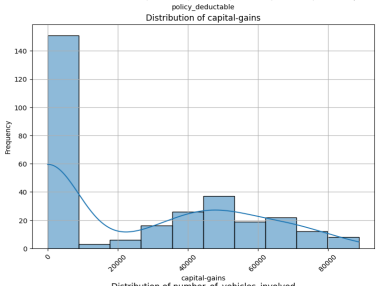


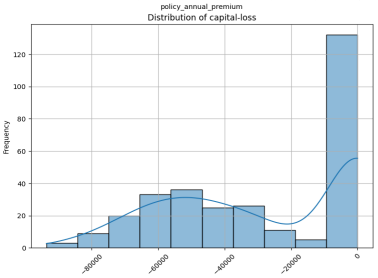


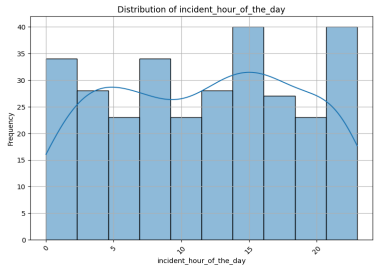


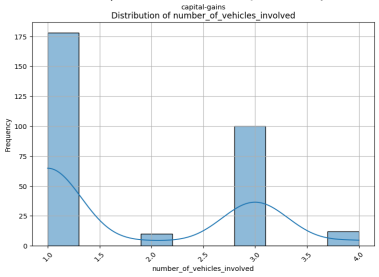


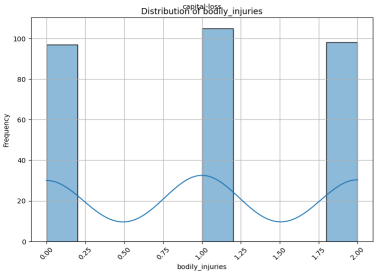


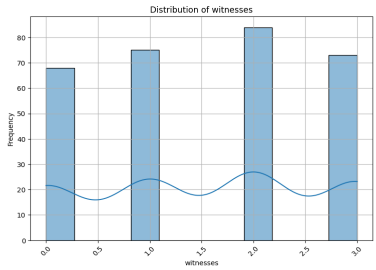


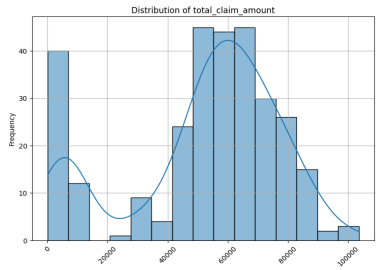


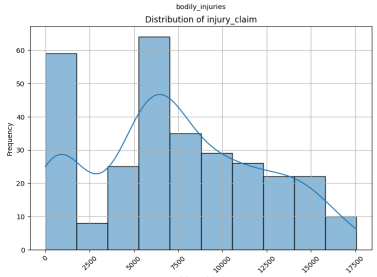


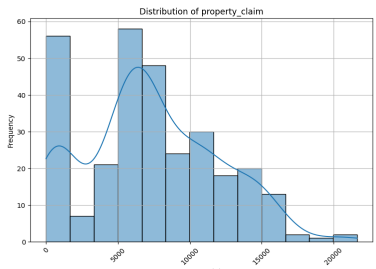


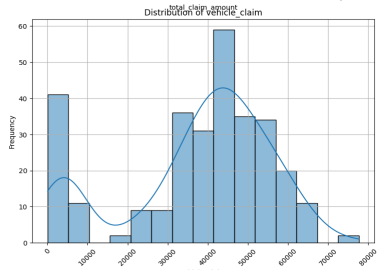


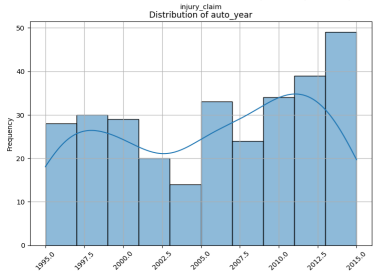






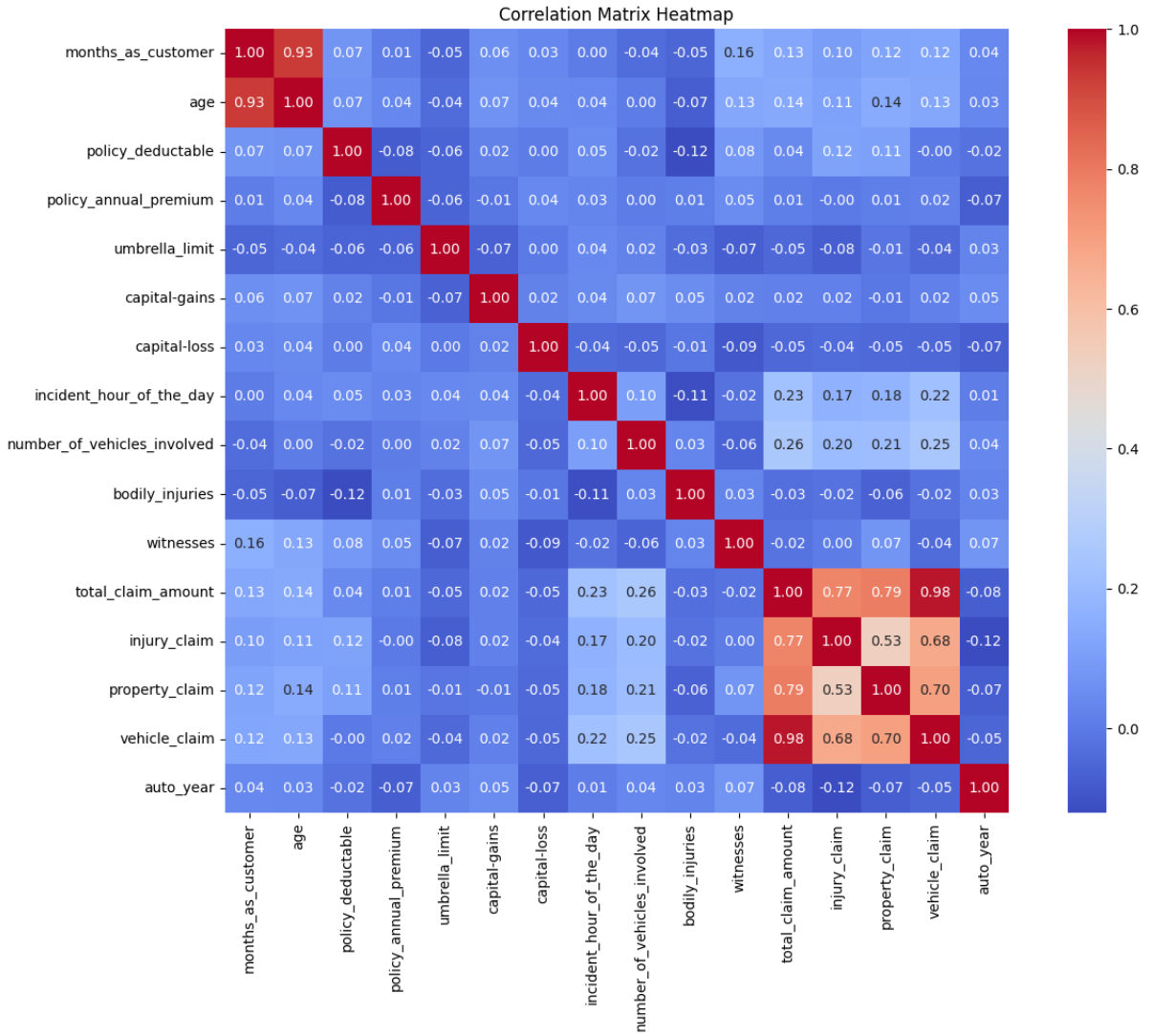




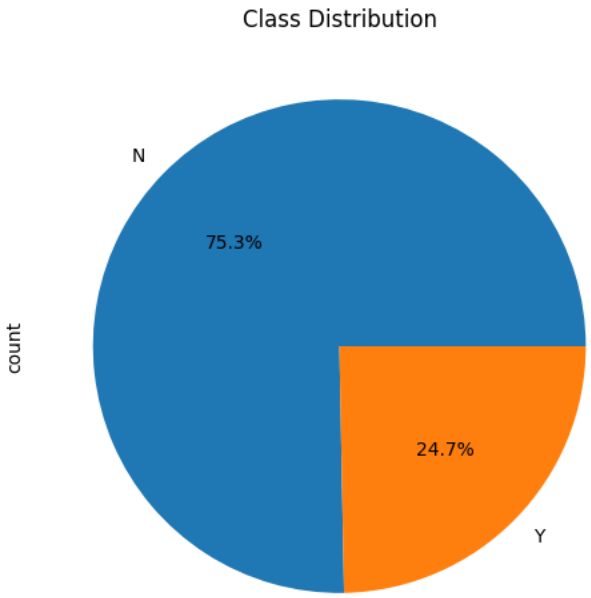


Observation: the variables months_as_customer and age appear fairly uniform, while policy_annual_premium follows an approximate normal distribution. Umbrella_limit is highly skewed with extreme outliers. Both capital-gains and capital-loss are sparse, with most values clustered near zero. Incident_hour_of_the_day is evenly distributed, suggesting accidents occur randomly across the day. Discrete features such as number_of_vehicles_involved, bodily_injuries, and witnesses display distinct peaks. Claim-related variables (total_claim_amount, injury_claim, property_claim, vehicle_claim) are positively skewed, dominated by smaller claims with few high-value cases. Auto_year covers multiple years, with higher frequencies for recent models. Preprocessing and scaling will be necessary.

5.2 PERFORM CORRELATION ANALYSIS



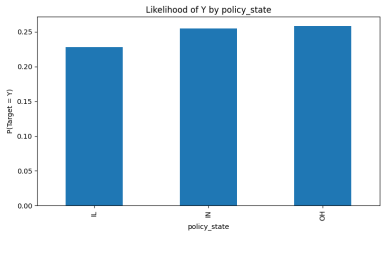
5.3 CHECK CLASS BALANCE

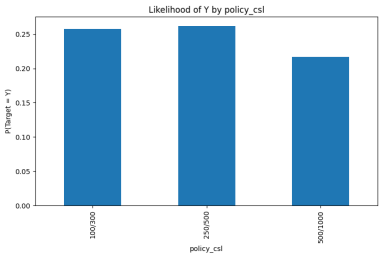


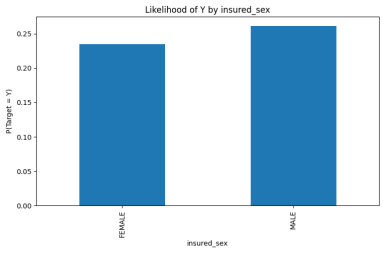
Observation: The target variable fraud_reported shows a clear class imbalance. Out of 1,000 records, 753 cases (75.3%) are labeled as non-fraudulent (N) while only 247 cases (24.7%) are fraudulent (Y). This imbalance means the dataset is skewed toward legitimate claims, which can bias machine learning models to favor the majority class. As a result, the model may achieve high accuracy but fail to detect fraudulent claims effectively.

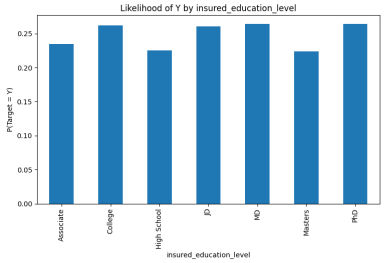
5.4 PERFORM BIVARIATE ANALYSIS

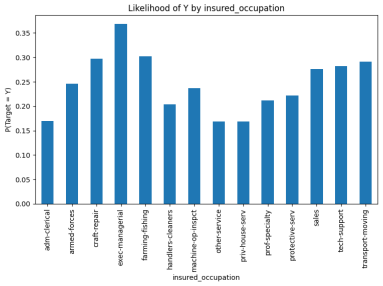
5.4.1Target likelihood analysis for categorical variables.

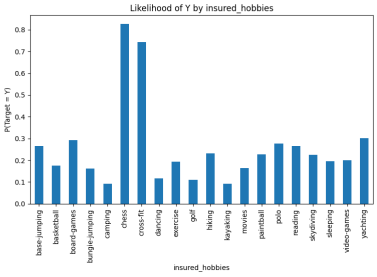


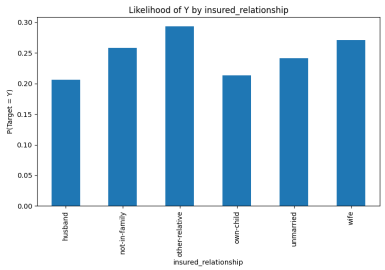


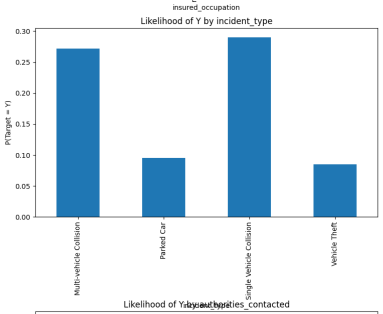


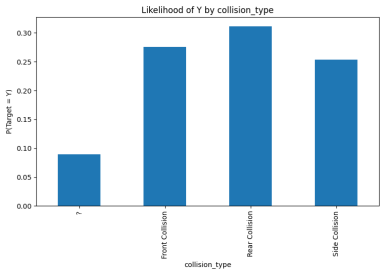


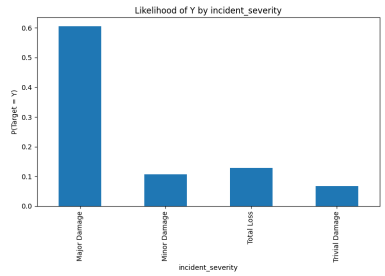


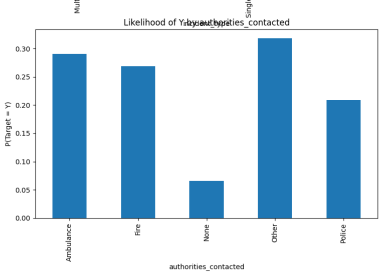


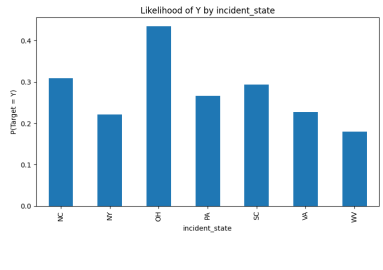


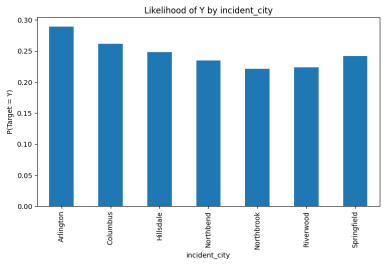


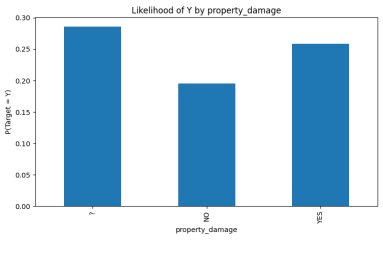


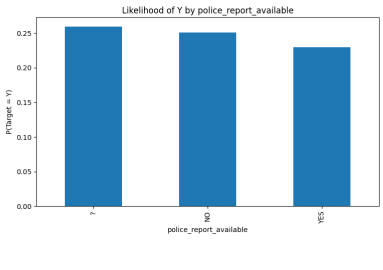


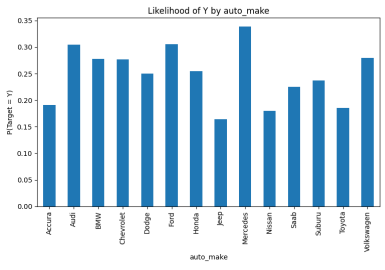


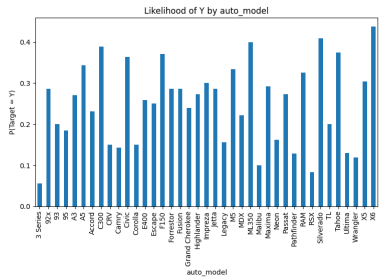




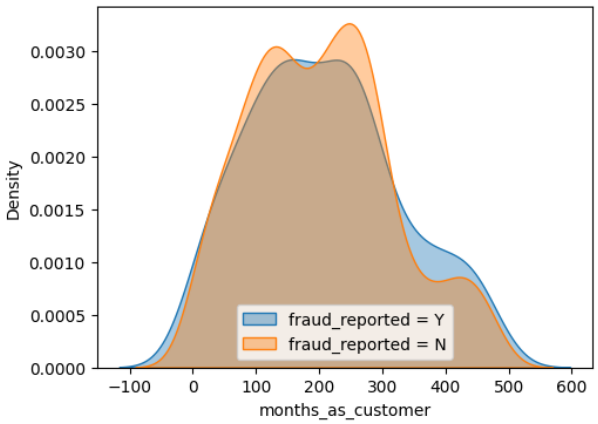


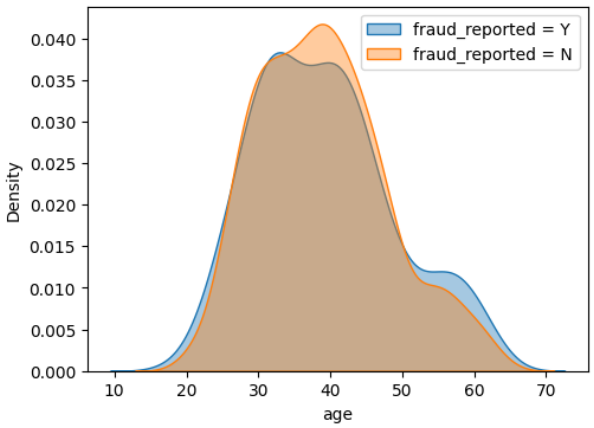


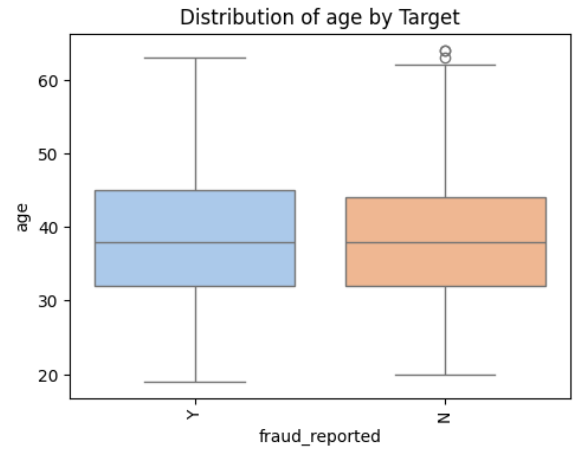


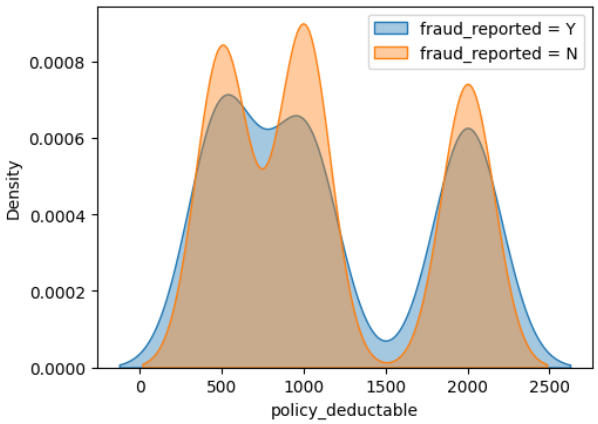


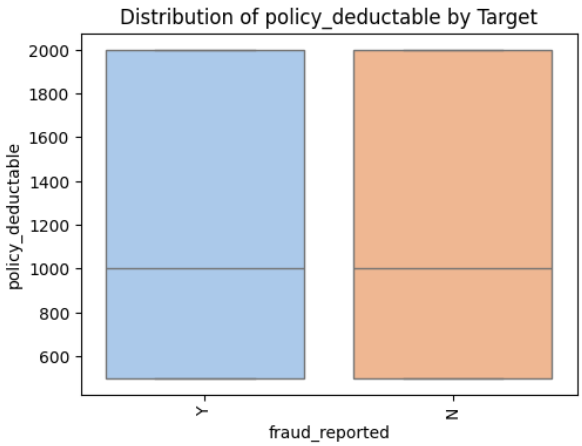
5.4.2 Explore the relationships between numerical features and the target variable to understand their impact on the target outcome.

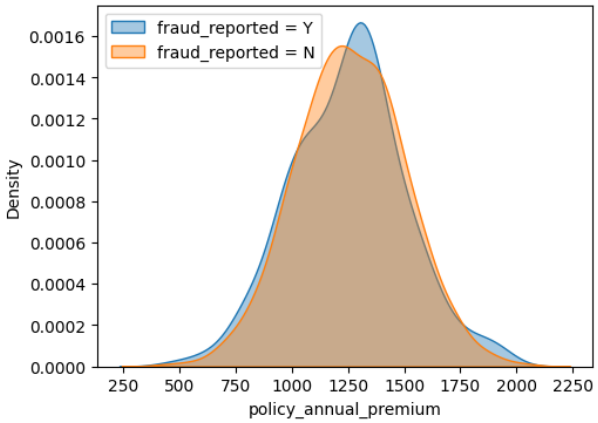


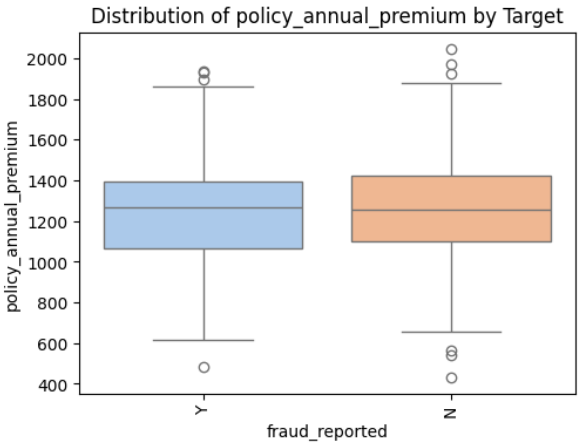


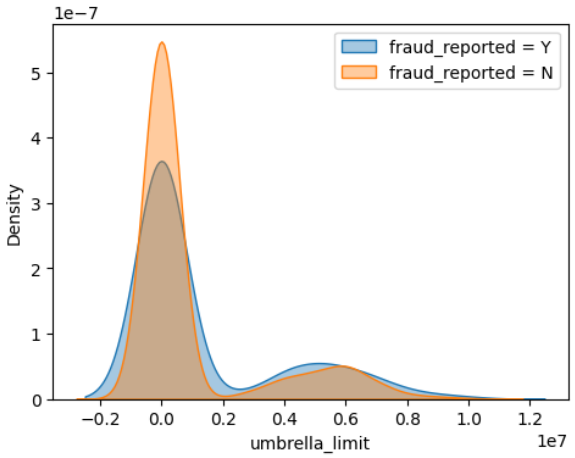


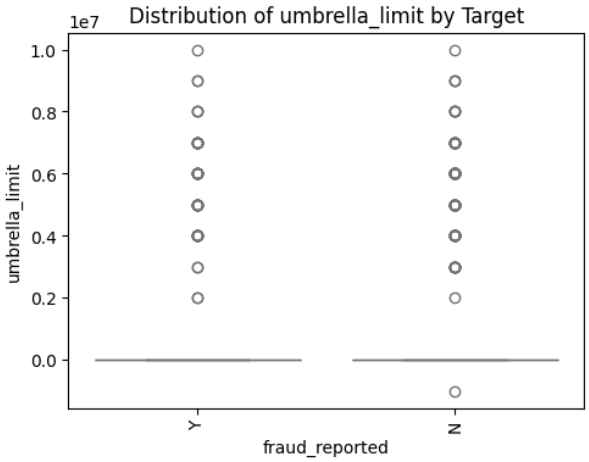


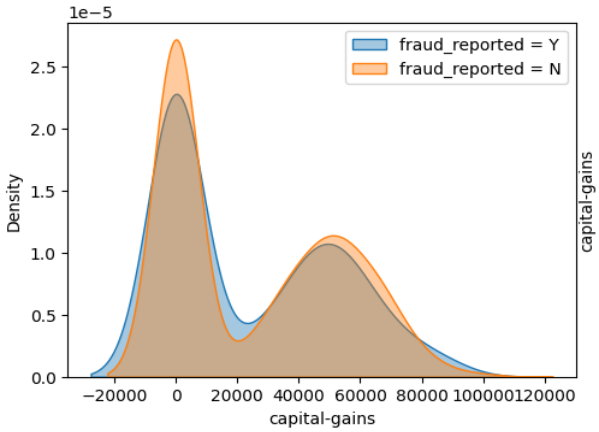



fraud_reported = Y
fraud_reported = N
Density
policy_annual_premium

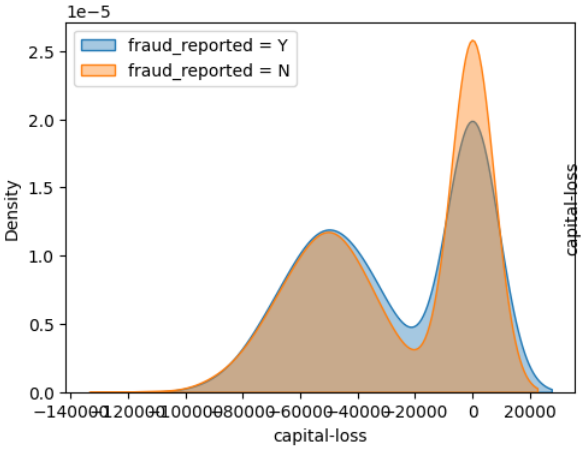

Distribution of policy_annual_premium by Target
policy_annual_premium
fraud_reported
Y
N

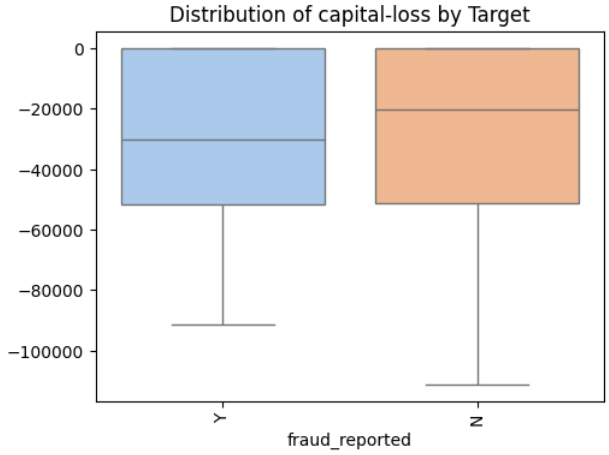

1e−7
fraud_reported = Y
fraud_reported = N
Density
umbrella_limit
1e7

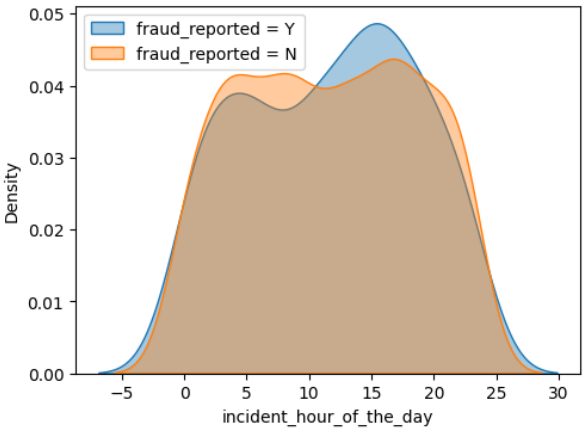

Distribution of umbrella_limit by Target
1e7
umbrella_limit
fraud_reported
Y
N

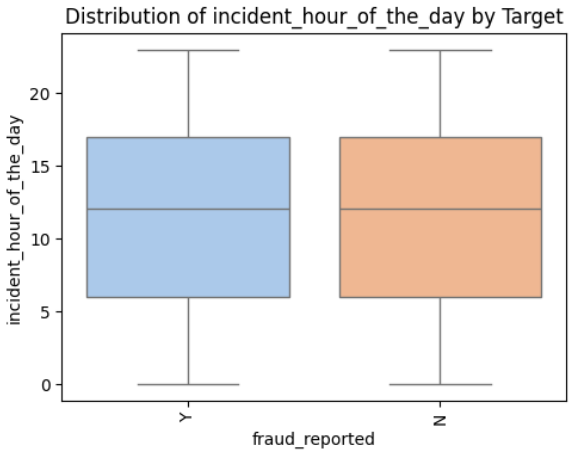

1e−5
fraud_reported = Y
fraud_reported = N
Density
capital-gains

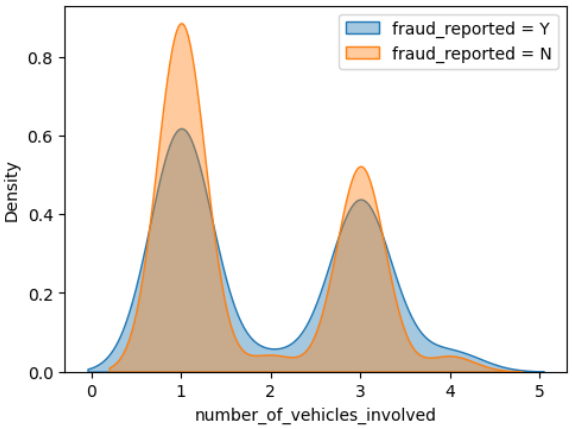

Distribution of capital-gains by Target
capital-gains
fraud_reported
Y
N

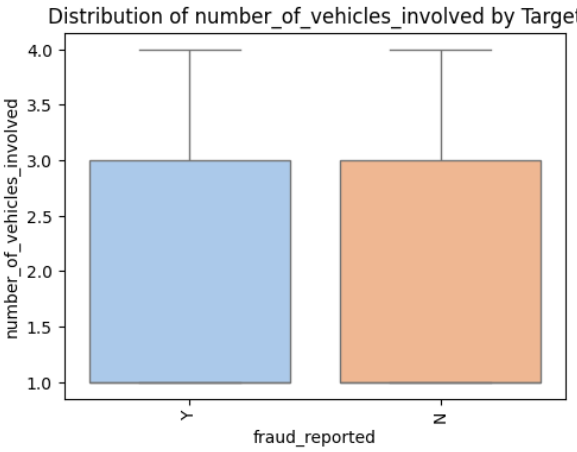

1e−5
fraud_reported = Y
fraud_reported = N
Density
capital-loss

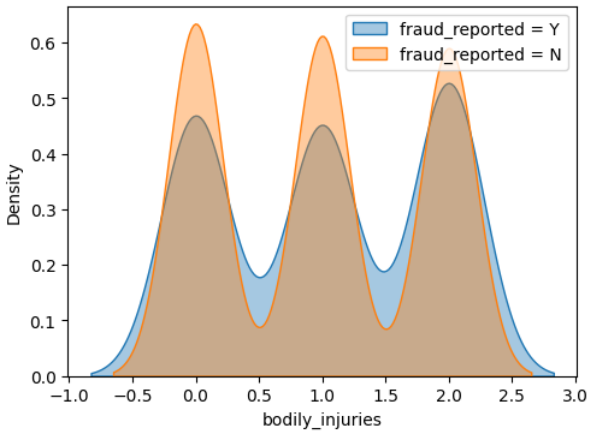

Distribution of capital-loss by Target
capital-loss
fraud_reported
Y
N

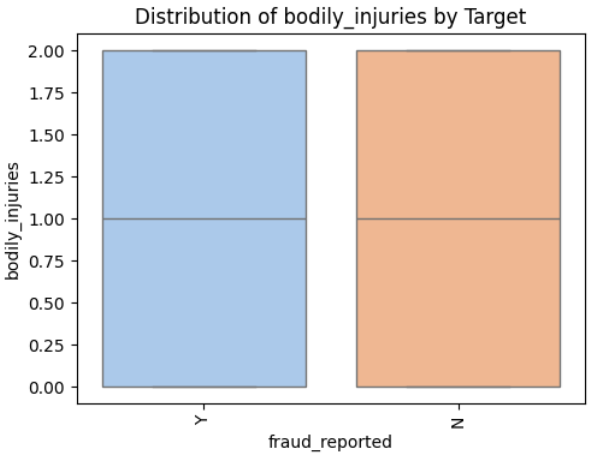

fraud_reported = Y
fraud_reported = N
Density
incident_hour_of_the_day

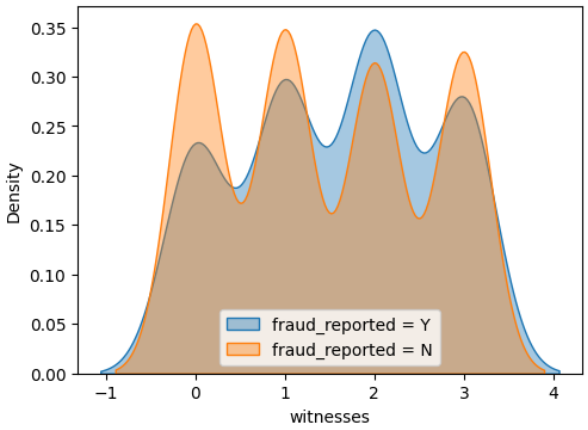

Distribution of incident_hour_of_the_day by Target
incident_hour_of_the_day
fraud_reported
Y
N


fraud_reported = Y
fraud_reported = N
Density
number_of_vehicles_involved

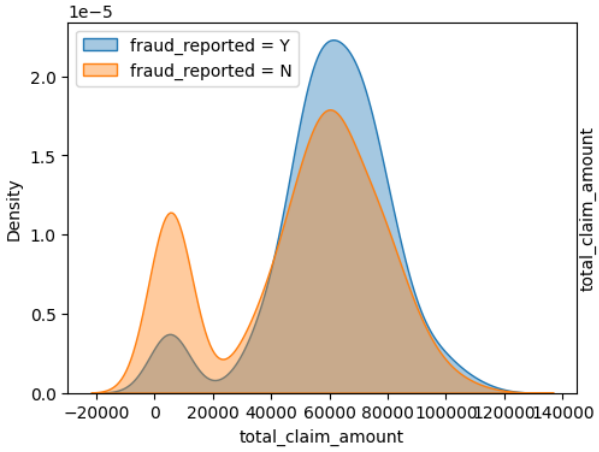

Distribution of number_of_vehicles_involved by Target
number_of_vehicles_involved
fraud_reported
Y
N

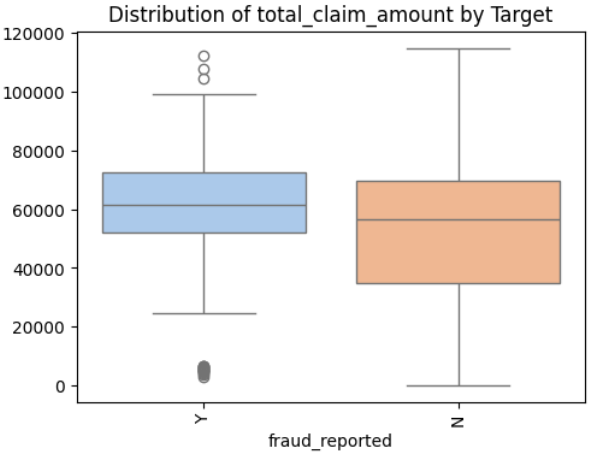

fraud_reported = Y
fraud_reported = N
Density
bodily_injuries

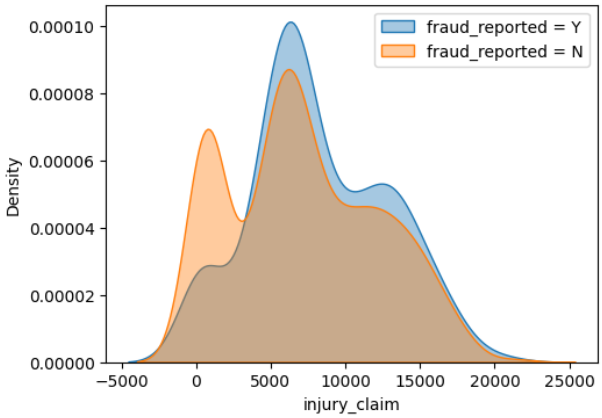

Distribution of bodily_injuries by Target
bodily_injuries
fraud_reported
Y
N

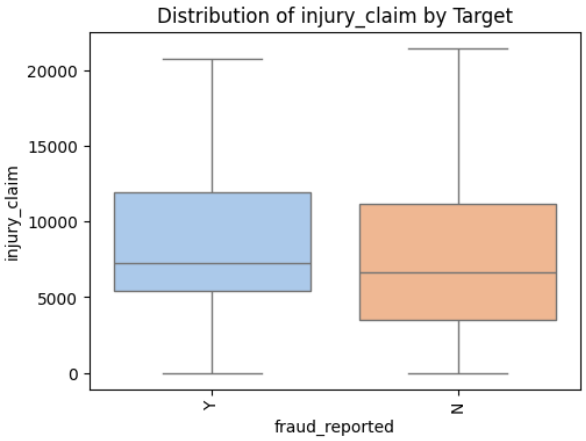

fraud_reported = Y
fraud_reported = N
Density
witnesses

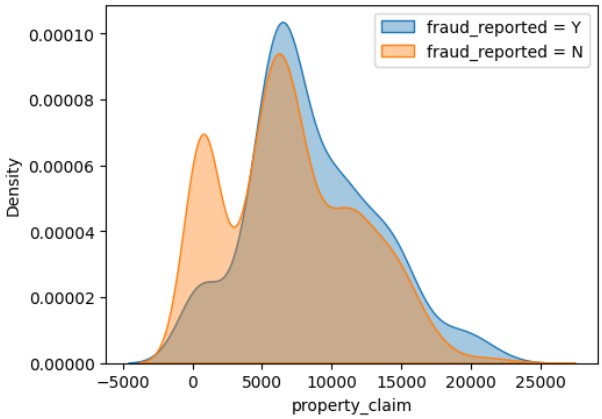

Distribution of witnesses by Target
witnesses
fraud_reported
Y
N


1e−5
fraud_reported = Y
fraud_reported = N
Density
total_claim_amount

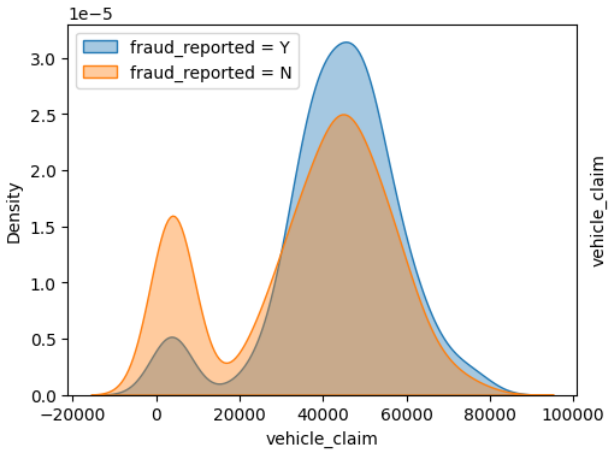

Distribution of total_claim_amount by Target
total_claim_amount
fraud_reported
Y
N

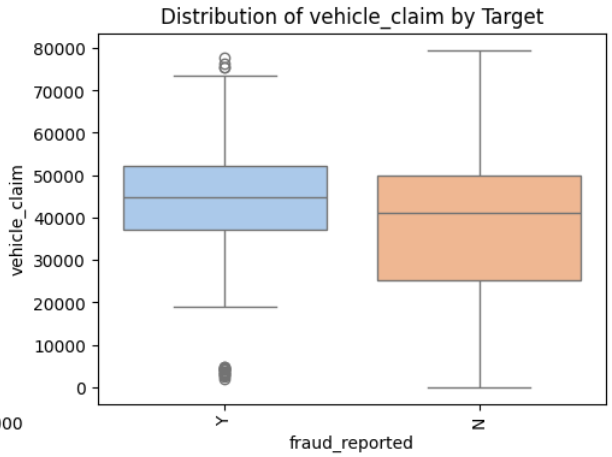

fraud_reported = Y
fraud_reported = N
Density
injury_claim

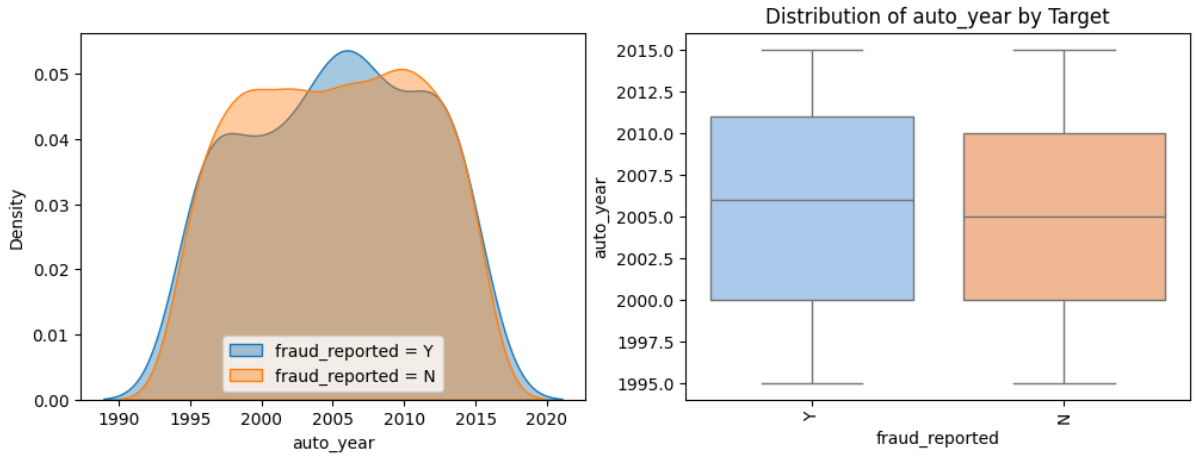

Distribution of injury_claim by Target
injury_claim
fraud_reported


fraud_reported = Y
fraud_reported = N
Density
property_claim


Distribution of property_claim by Target
property_claim
fraud_reported


1e−5
fraud_reported = Y
fraud_reported = N
Density
vehicle_claim


Distribution of vehicle_claim by Target
vehicle_claim
fraud_reported



6. FEATURE ENGINEERING

6.1 PERFORM RESAMPLING

Before resampling, the training set showed a class imbalance with 527 non-fraud (N) and only 173 fraud (Y) cases, meaning fraudulent claims made up just 25% of the data. After applying RandomOverSampler, the dataset is perfectly balanced with 527 fraud and 527 non-fraud cases. This ensures the model will receive equal representation from both classes during training, improving its ability to detect fraud. However, since oversampling duplicates minority class records, there is a risk of overfitting. Alternative approaches like SMOTE or ensemble methods can be explored for more robust fraud detection.

6.2 FEATURE CREATION

- policy_age_days: The age of the policy in days at the time of the incident.
- total_claim_amount: The total amount claimed (sum of all claim-related amounts).
- is_weekend: A binary feature indicating whether the incident occurred on a weekend
- num_vehicles_involved: The number of vehicles involved in the incident (if applicable).
- claim_to_policy_ratio: The ratio of the total claim amount to the policy premium.
- incident_month: The month when the incident occurred, which might capture seasonal trends.
- vehicle_age: The age of the vehicle at the time of the incident, calculated from the auto_year.
- policy_duration: The duration of the policy in days from the policy_bind_date to the incident_date.

Shape of X_train_resampled: (1054, 35)

Shape of X_train: (700, 35)

Shape of X_test: (300, 35)

Shape of X_train_resampled: (1054, 41)

Shape of X_train: (700, 41)

Shape of X_test: (300, 41)

6.3 HANDLE REDUNDANT COLUMNS

'policy_bind_date','incident_date','auto_year','total_claim_amount','injury_claim','property_claim','vehicle_claim'

6.4 COMBINE VALUES IN CATEGORICAL COLUMNS

The rare category grouping significantly simplified categorical variables by consolidating infrequent values into an “Other” category. For insured_occupation, only two low-frequency jobs were grouped, creating a more balanced distribution. Insured_hobbies had many sparse categories combined, leaving only a few dominant hobbies and reducing noise. In incident_state, less frequent states were merged, while major states remained intact. Auto_make saw minimal impact, with only Honda grouped into “Other.” However, auto_model collapsed almost entirely into “Other,” indicating excessive cardinality and limited predictive value, making it a strong candidate for removal. Overall, this transformation enhances stability and reduces overfitting risk.

6.5 DUMMY VARIABLE CREATION

6.5.1Identify categorical columns for dummy variable creation

- Most of these variables have low to moderate cardinality and are useful for categorical encoding (e.g., OneHotEncoder or Target Encoding).
- insured_hobbies and insured_occupation were simplified with rare category grouping, so they’re cleaner now.
- incident_city and incident_state have limited categories and seem stable.
- auto_model is still problematic (mostly "Other"); dropping it may improve model efficiency without losing predictive power.

6.5.2 Create dummy variables for categorical columns in training data

Dummy variable creation using pd.get_dummies successfully transformed categorical features into numerical format for modeling. By applying one-hot encoding with drop_first=True, multicollinearity is reduced as one category from each feature is dropped. The transformation expanded the feature space, increasing the number of columns compared to the original dataset. Both the resampled training data and the original training data were encoded consistently, ensuring compatibility during model training and evaluation. This step allows algorithms that require numerical input, such as Logistic Regression, to handle categorical predictors effectively. The resulting dataset is now structured and ready for scaling and modeling.

6.5.3 Create dummy variables for categorical columns in validation data

Before encoding, the validation dataset contained 34 columns, including categorical variables. After applying pd.get_dummies with drop_first=True, the number of features expanded to 92 columns. This increase reflects the conversion of categorical variables into multiple binary (dummy) variables, while dropping the first category of each to avoid multicollinearity. The transformation ensures categorical predictors are in a numeric format suitable for machine learning algorithms. However, to avoid mismatches between training and validation data, it is essential to align columns (adding any missing dummy columns in validation with zeros). This step ensures consistent feature space across datasets.

6.5.4 Create dummy variable for dependent feature in training and validation data

The target variable was reshaped for consistency before modeling. Initially, the original training target had a shape of (700,), and the resampled training target had a shape of (1054,). After reshaping, these became (700, 1) and (1054, 1) respectively, making them two-dimensional arrays suitable for many machine learning libraries. Similarly, the test target was converted from (300,) to (300, 1). This ensures alignment between feature matrices and target vectors during training and evaluation, reduces errors in scikit-learn pipelines, and maintains consistency across datasets, especially when using models that require explicit two-dimensional inputs.

6.6 FEATURE SCALING

The numeric features in both training and validation datasets were standardized using StandardScaler from scikit-learn. First, all numerical columns were identified, and the scaler was fitted on the resampled training data to compute the mean and standard deviation. These parameters were then applied to transform both training and validation datasets, ensuring consistency. After scaling, the numeric features have a mean of 0 and a standard deviation of 1, which improves model convergence and performance, especially for algorithms sensitive to feature magnitudes such as Logistic Regression and SVM. The categorical dummy variables remain unchanged during this process.

7. MODEL BUILDING

7.1 FEATURE SELECTION

7.1.1 Import necessary libraries

The required libraries for feature selection, modeling, and evaluation were imported.

- RFECV (Recursive Feature Elimination with Cross-Validation) is used to identify the most important features by recursively removing less significant ones.
- LogisticRegression is the base classifier for modeling fraud detection.
- Metrics such as accuracy_score, classification_report, confusion_matrix, and ConfusionMatrixDisplay enable performance evaluation and visualization.
- cross_val_score supports cross-validation to assess model stability and generalization.

7.1.2 Perform feature selection

Recursive Feature Elimination with Cross-Validation (RFECV) was applied using Logistic Regression as the base model to identify the most relevant predictors. The method iteratively removed less important features, performing 5-fold cross-validation at each step, and evaluated model performance using accuracy as the metric. RFECV determined the **optimal number of features** and generated a refined subset of predictors that contribute most to classification. A feature ranking DataFrame was also created, where selected features received a ranking of 1, and less relevant ones were ranked higher. This approach improves interpretability, reduces dimensionality, and ensures stronger generalization in fraud detection.

7.1.3 Retain the selected feature

It includes both numeric features (months_as_customer, age, policy_deductable, policy_annual_premium, etc.) and engineered features (is_weekend, claim_to_policy_ratio, incident_month, vehicle_age).

It also retains the most important categorical dummy variables (like policy_state_OH, policy_csl_250/500, insured_education_level_College, and multiple auto_make).

7.2 BUILD LOGISTIC REGRESSION MODEL

7.2.1Select relevant features and add constant in training data

Observations

- After applying RFECV, you selected 65 predictors plus a constant, resulting in 66 total columns in X_train_selected_const.
- Data types:
  - 55 columns are boolean dummies (from categorical encoding).
  - 10 are continuous float variables (e.g., months_as_customer, age, policy_annual_premium).
  - 1 is an integer feature (incident_month).
- const was added via sm.add_constant(), which is required for regression models in statsmodels to estimate an intercept.
- Memory footprint is small (~143 KB), so the dataset is compact and efficient.
- Dataset shape: 1054 rows × 66 columns → perfectly ready for fitting a Logit model in statsmodels.

7.2.2Fit logistic regression model

The model converged in 8 iterations.

Current function value 0.282669 → indicates the log-likelihood value at convergence. Lower is generally better.

Result summary:

- Coefficients (coef): Impact of each feature on the probability of fraud (positive → increases fraud likelihood, negative → decreases).
- Standard Error: Variability in coefficient estimates.
- z-values & P>|z|: Significance tests. Features with $p < 0.05$ are statistically significant predictors.
- [0.025, 0.975]: Confidence intervals for each coefficient.

Goodness-of-Fit Metrics:

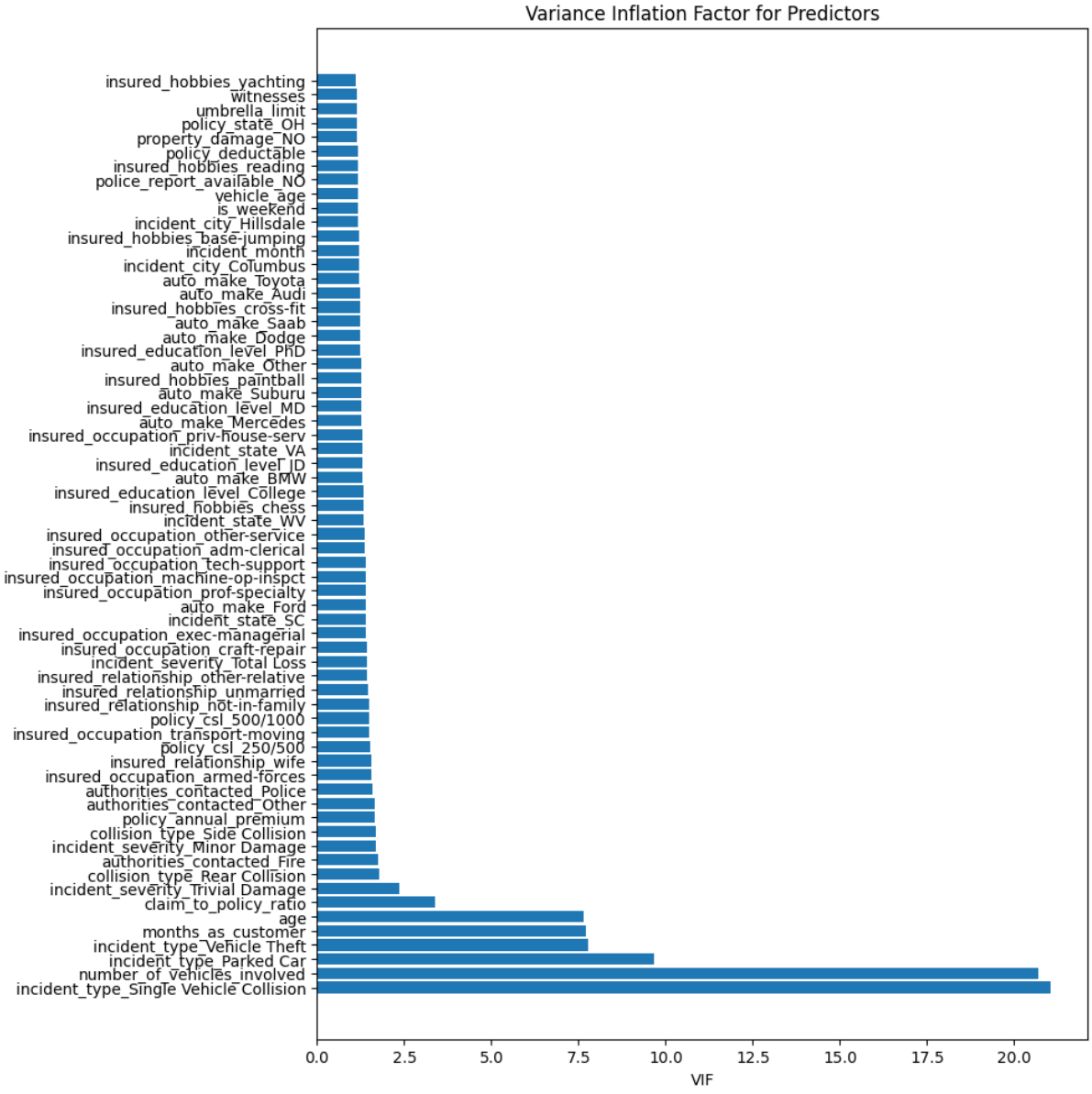
- Pseudo R-squared: Similar to R^2 but for logistic regression, showing model explanatory power.
- Log-Likelihood: Overall model fit.

Insight

- Fraud is more likely when claims are large relative to premiums, deductibles are higher, umbrella coverage is large, and vehicles are older.

- Age of insured negatively relates to fraud → younger policyholders may pose higher fraud risk.
- Brand-specific risks (e.g., Suburu) stand out.
- Seasonal/monthly patterns in fraud exist (certain months show reduced risk).

7.2.3 Evaluate VIF of features to assess multicollinearity



7.2.4 Make predictions on training data

7.2.5 Create a DataFrame that includes actual fraud reported flags, predicted probabilities, and a column indicating predicted classifications based on a cutoff value of 0.5

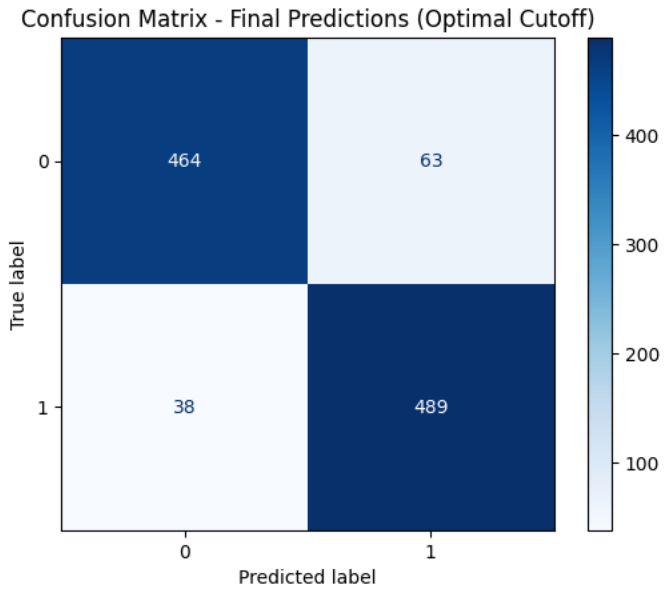
A new prediction results DataFrame was created containing the actual fraud labels, the predicted fraud probabilities, and the predicted classifications based on a threshold of 0.5. The predicted_probability column stores model-estimated likelihoods of fraud, while the predicted_class column converts these probabilities into binary outcomes, labeling records as fraud (1) or non-fraud (0). This structure enables direct comparison between actual outcomes and model predictions. It also provides the flexibility to

adjust the classification cutoff, allowing trade-offs between precision and recall. Overall, the DataFrame serves as the foundation for evaluating model accuracy, sensitivity, specificity, and overall effectiveness.

7.2.6 Check the accuracy of the mode

The logistic regression model achieved a training accuracy of 90.42%

7.2.7 Create a confusion matrix based on the predictions made on the training data



Observation:

- The model performs strongly with high TP (489) and high TN (464).
- Relatively low FN (38) shows good fraud detection capability (strong recall).
- Some FP (63) occur, meaning a few legitimate claims are flagged as fraud, which is acceptable in fraud detection where recall is prioritized.

7.2.8 Create variables for true positive, true negative, false positive and false negative

- True Negatives (TN) = 464 → Non-fraud correctly identified as non-fraud.
- False Positives (FP) = 63 → Non-fraud incorrectly classified as fraud.
- False Negatives (FN) = 38 → Fraud cases missed (predicted as non-fraud).
- True Positives (TP) = 489 → Fraud cases correctly identified.

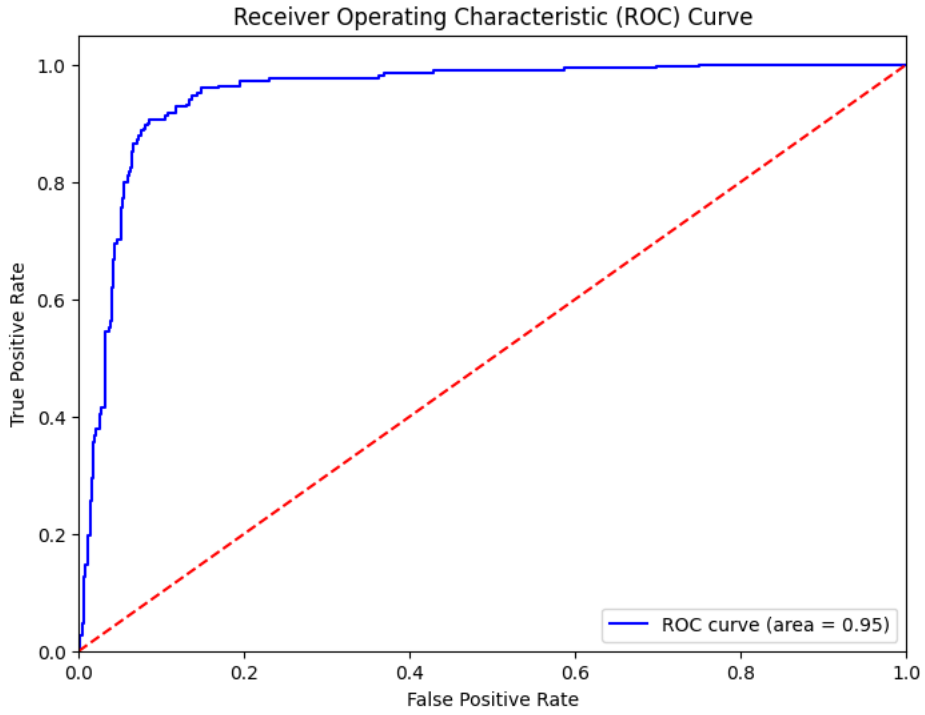
7.2.9 Calculate sensitivity, specificity, precision, recall and F1-score

- Sensitivity (Recall) = 0.9279 → The model detects about 93% of actual fraud cases, minimizing false negatives.
- Specificity = 0.8805 → About 88% of legitimate claims are correctly classified as non-fraud, indicating balanced performance.
- Precision = 0.8859 → Nearly 89% of claims flagged as fraud are truly fraudulent, showing good reliability in positive predictions.

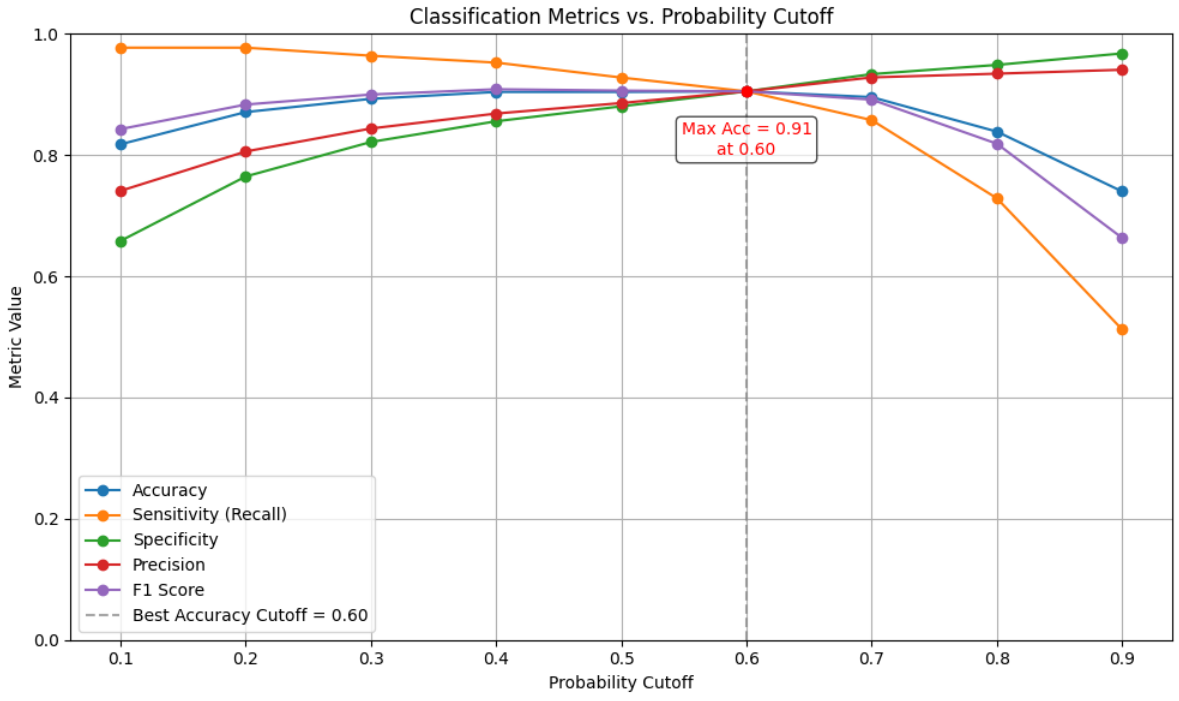
- F1 Score = 0.9064 → Strong harmonic balance between precision and recall, confirming the model handles class imbalance effectively.

7.3 FIND THE OPTIMAL CUTOFF

7.3.1Plot ROC Curve to visualise the trade-off between true positive rate and false positive rate across different classification thresholds



7.3.3Plot accuracy, sensitivity, specificity at different values of probability cutoffs



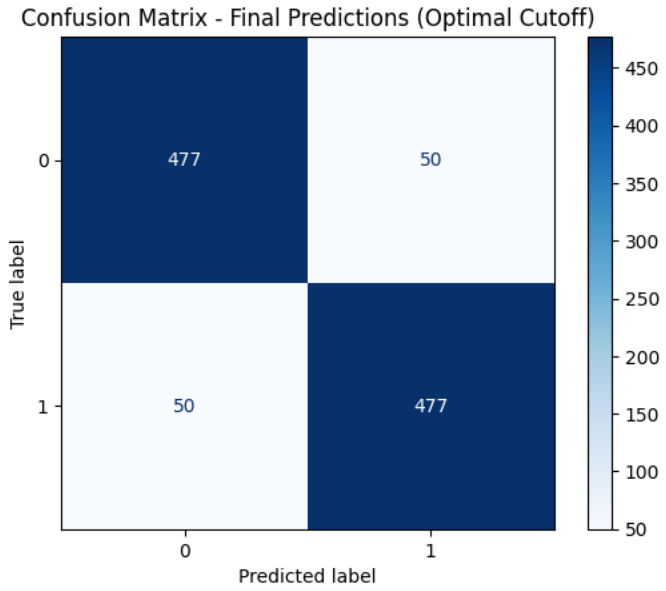
7.3.4Create a column for final prediction based on optimal cutoff

The model's final classification threshold was adjusted from the default 0.5 to an optimal cutoff of 0.6, based on performance analysis across different probability levels. At this higher cutoff, the model adopts a more conservative approach, flagging fewer cases as fraud. This helps reduce false positives (legitimate claims incorrectly marked as fraud) while maintaining strong fraud detection ability. Although some increase in false negatives may occur, the trade-off improves overall reliability. Selecting 0.6 as the decision threshold ensures better alignment with business objectives, balancing fraud detection accuracy with minimizing unnecessary investigations.

7.3.5 Calculate the accuracy

At the optimal cutoff of 0.60, the model achieved a strong training accuracy of 90.51%, balancing fraud detection with reduced false positives.

7.3.6** Create confusion matrix



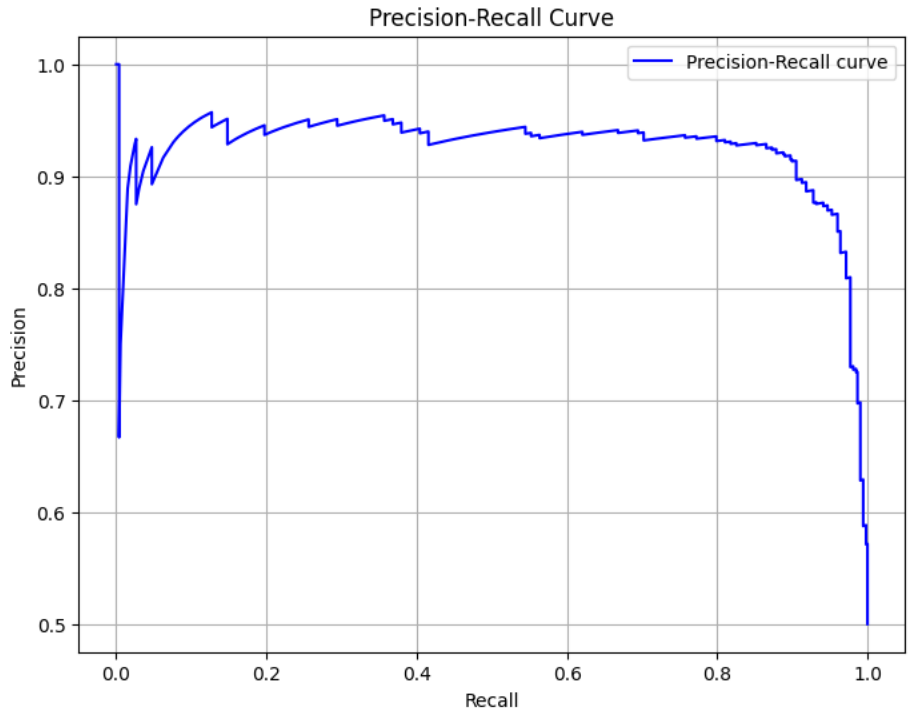
7.3.7 Create variables for true positive, true negative, false positive and false negative

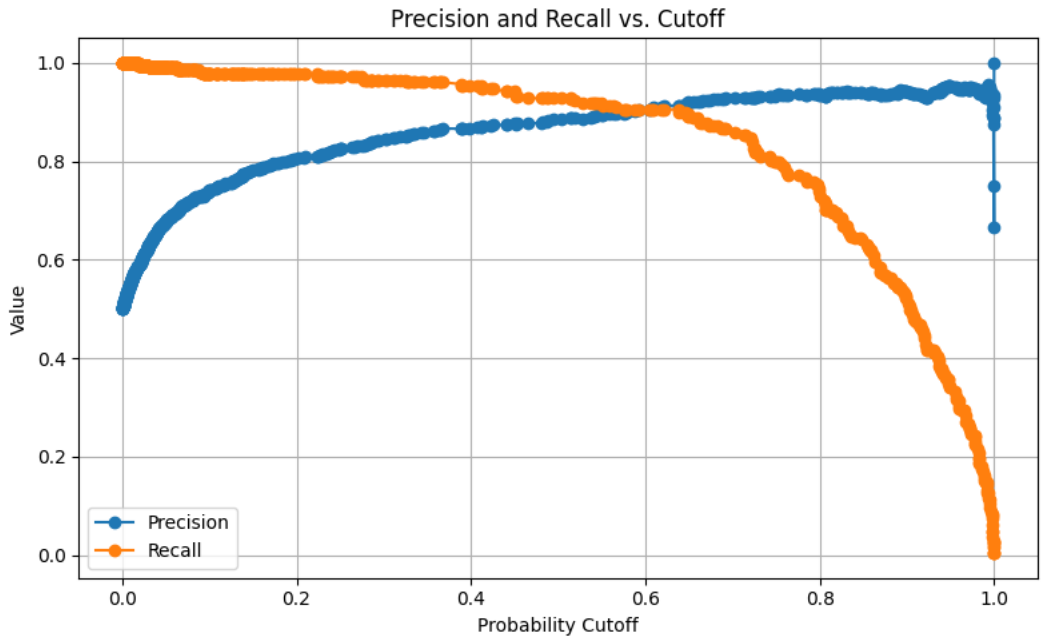
- True Negatives (477): Legitimate claims correctly identified as non-fraud.
- False Positives (50): Legitimate claims incorrectly flagged as fraud.
- False Negatives (50): Fraudulent claims missed and classified as non-fraud.
- True Positives (477): Fraudulent claims correctly detected as fraud.

7.3.8Calculate sensitivity, specificity, precision, recall and F1-score of the model

The model demonstrates balanced and consistent performance with all key metrics — sensitivity, specificity, precision, recall, and F1-score — equal to 0.9051. This indicates it performs equally well in detecting fraud, avoiding false alarms, and maintaining precision–recall balance.

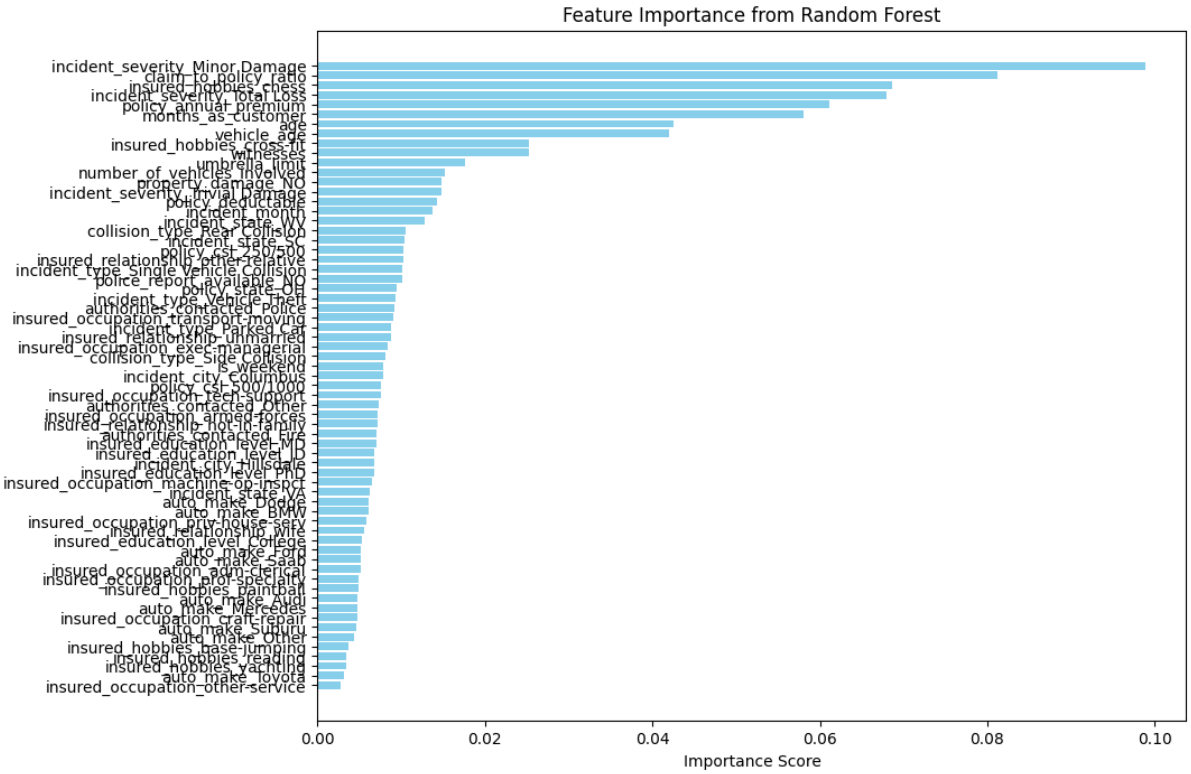
7.3.9 Plot precision-recall curve





7.4 BUILD RANDOM FOREST MODEL

7.4.3 Get feature importance scores and select important features



- Incident severity dominates model decisions, confirming claim details are crucial in detecting fraud.
- Financial ratios like claim-to-policy ratio are highly predictive, aligning with domain expectations.
- Some categorical dummy features (like hobbies or specific auto makes) may capture subtle but less generalizable patterns.
- Features with extremely low importance (<0.005) could be considered for removal or deprioritization to simplify the model.

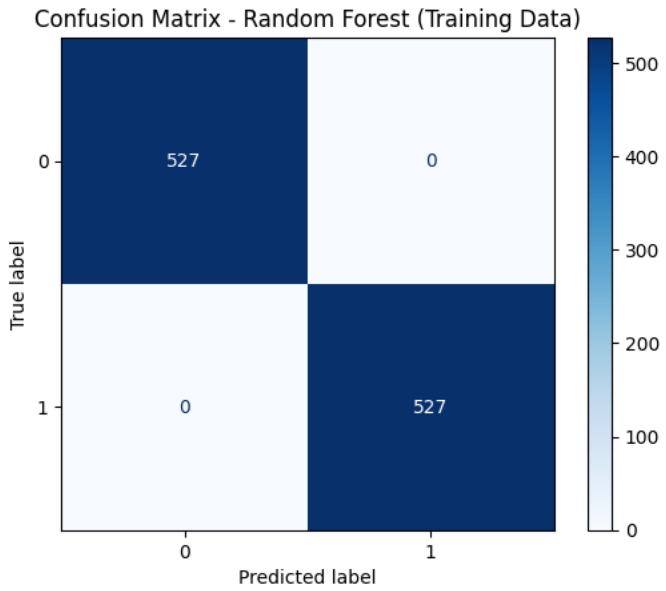
Final Selected TOP 1O Features :

- incident_severity_Minor Damage – strong indicator of fraudulent patterns.
- claim_to_policy_ratio – higher ratios often point to fraud risk.
- insured_hobbies_chess – unique behavioral variable contributing to prediction.
- incident_severity_Total Loss – severe incidents are strongly linked to fraud.
- policy_annual_premium – financial commitment size influences fraud likelihood.
- months_as_customer – customer loyalty or tenure affects fraud risk.
- age – younger customers show different fraud tendencies compared to older ones.
- vehicle_age – older vehicles are more often linked to suspicious claims.
- insured_hobbies_cross-fit – another lifestyle variable influencing fraud patterns.
- witnesses – presence or absence of witnesses impacts fraud likelihood.

7.4.6Check accuracy of the model

Random Forest Training accuracy: 1.0000

7.4.7Create confusion matrix



7.4.8 Create variables for true positive, true negative, false positive and false negative

This confusion matrix shows perfect classification by the model:

- True Negatives (527): All legitimate claims correctly identified as non-fraud.
- False Positives (0): No legitimate claims were wrongly flagged as fraud.
- False Negatives (0): No fraudulent claims were missed.
- True Positives (527): All fraudulent claims were correctly detected.

This indicates 100% accuracy, precision, recall, and F1-score on the evaluated dataset. While excellent, it's important to validate on unseen data to ensure the model is not overfitting.

7.4.9Calculate sensitivity, specificity, precision, recall and F1-score of the model

The model achieved **perfect performance** with **100% sensitivity, specificity, precision, recall, and F1 score**, meaning it correctly identified all fraudulent and non-fraudulent claims without any errors. While this indicates exceptional predictive power, such flawless results on training or validation data may also signal **overfitting or data leakage**, so further validation on unseen, real-world data is critical to confirm generalizability.

7.4.10Check if the model is overfitting training data using cross validation

Cross-validation results show consistently high performance, with scores ranging from 0.9099 to 0.9242, a mean of 0.9194, and a very low standard deviation of 0.0061. This indicates the model is stable, reliable, and generalizes well across folds, with minimal performance variation.

7.5 HYPERPARAMETER TUNING

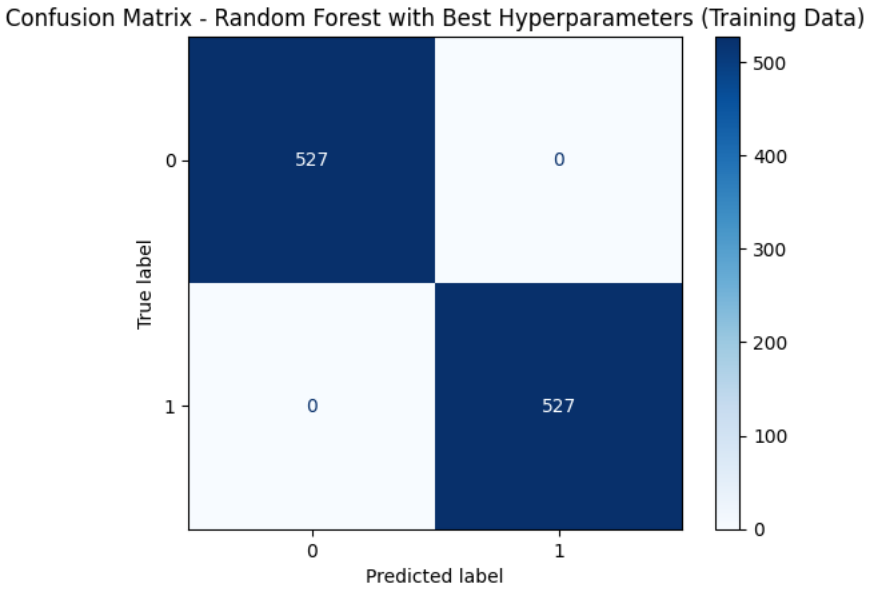
7.5.1 Use grid search to find the best hyperparameter values

- Purpose: performed a GridSearchCV over a RandomForestClassifier to find the best hyperparameters.
- Grid: 2 × 3 × 2 × 2 × 2 × 3 = 144 candidate parameter combinations (n_estimators, max_depth, min_samples_split, min_samples_leaf, bootstrap, max_features).
- CV: cv=3 → 432 total fits.
- Input: tuned on X_train_rf_selected with target y.
- Scoring: accuracy.
- Result: Best Hyperparameters = {'bootstrap': False, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}.
- Best cross-validation score: 0.9279.These settings produced the **best cross-validation score = 0.9279**, indicating strong generalization during CV. In short: the tuned forest is fairly large (`n_estimators=200`), uses sqrt features per split, no depth limit, and no bootstrapping — a high-capacity configuration that achieved the highest CV performance in your search.

7.5.4Check accuracy of Random Forest Model

The Random Forest model achieved a training accuracy of 1.0000 with the best hyperparameters, indicating it perfectly classified all training samples. While this reflects an excellent fit on the training set, such a flawless score strongly suggests overfitting, meaning the model may not generalize equally well on unseen test data.

7.5.5 Create confusion matrix



7.5.6Create variables for true positive, true negative, false positive and false negative

- True Negatives (TN): 527 – All legitimate claims were correctly identified.
- False Positives (FP): 0 – No legitimate claim was wrongly flagged as fraud.
- False Negatives (FN): 0 – No fraudulent claim was missed.
- True Positives (TP): 527 – All fraudulent claims were correctly detected.

7.5.7Calculate sensitivity, specificity, precision, recall and F1-score of the model

The Random Forest model achieved perfect performance on the training set, with sensitivity, specificity, precision, recall, and F1-score all equal to 1.0000. This means the model detected every fraudulent claim without any false positives or false negatives, and equally classified legitimate claims correctly. While this result indicates excellent fit on the training data, it may also suggest overfitting, and validation on unseen data is necessary to confirm its generalization ability.

8. PREDICTION AND MODEL EVALUATION

8.1 MAKE PREDICTIONS OVER VALIDATION DATA USING LOGISTIC REGRESSION MODEL

8.1.1Select relevant features for validation data and add constant

The validation dataset was refined by selecting only the relevant features identified through RFECV, ensuring alignment with the training set. A constant term was added to the validation features to support logistic regression modeling. Finally, all values in the validation dataset were converted to float type, standardizing the data and preparing it for accurate and consistent model evaluation.

8.1.2Make predictions over validation data

Predictions were generated on the validation dataset using the trained logistic regression model. The model first produced probability scores (y_val_pred_prob) for each instance, representing the likelihood of fraud. These probabilities were then converted into binary classifications (y_validation_pred) by applying a 0.5 cutoff threshold, where values ≥ 0.5 were classified as fraud and values < 0.5 as non-fraud.

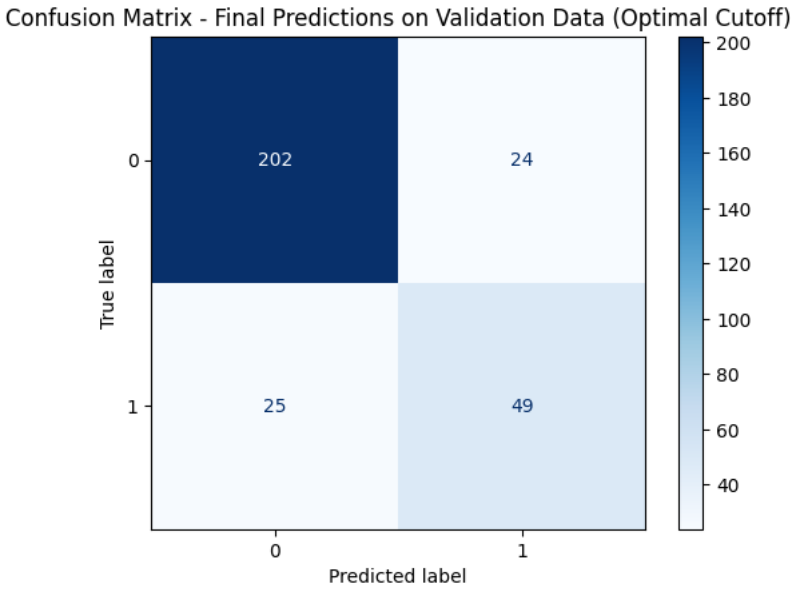
8.1.3 Create DataFrame with actual values and predicted values for validation data

A DataFrame df_validation_results was created to compare model performance on validation data by pairing actual fraud labels with predicted classifications. This structure allows direct evaluation of prediction accuracy and misclassifications. For example, the preview shows correct predictions for both fraud (True → 1) and non-fraud (False → 0), confirming the model's ability to align predicted outcomes with actual values.

8.1.5Check the accuracy of logistic regression model on validation data

The model achieved a validation accuracy of 83.67% at the optimal cutoff of 0.60, indicating that it correctly classified most fraud and non-fraud cases in the validation dataset.

8.1.6Create confusion matrix



The confusion matrix shows that the model correctly predicted 202 true negatives and 49 true positives, while it made 24 false positive errors (non-fraud predicted as fraud) and 25 false negative errors (fraud predicted as non-fraud).

8.1.7Create variables for true positive, true negative, false positive and false negative

The model correctly classified 202 non-fraud cases and 49 fraud cases, while misclassifying 24 non-fraud cases as fraud and 25 fraud cases as non-fraud.

8.1.8Calculate sensitivity, specificity, precision, recall and f1 score of the model

The validation results indicate moderate performance: the model achieved a recall of 66.22%, meaning it detected about two-thirds of fraud cases, and a specificity of 89.38%, showing strong ability to correctly identify non-fraud. Precision was 67.12%, suggesting that roughly two-thirds of predicted fraud cases were correct. The F1 score of 66.67% balances precision and recall, reflecting reasonable but not perfect fraud detection capability.

8.2 MAKE PREDICTIONS OVER VALIDATION DATA USING RANDOM FOREST MODEL

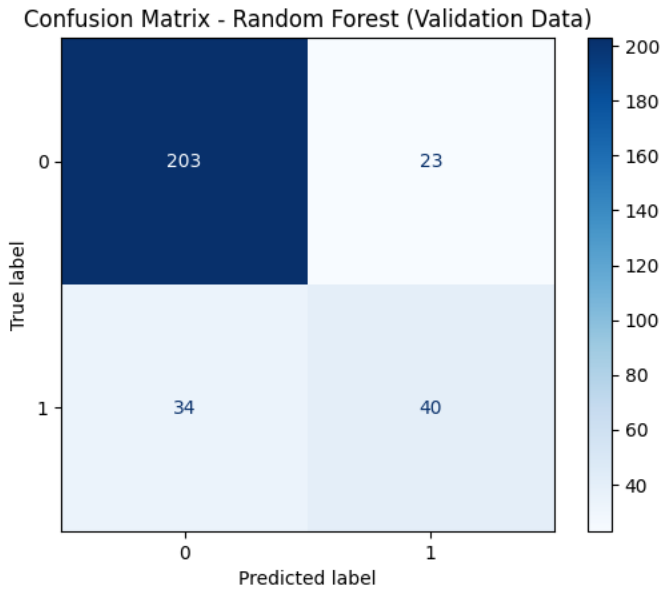
8.2.1Select the important features and make predictions over validation data

This step applies the tuned Random Forest model to the validation dataset. First, only the previously identified important features are selected from the scaled validation data (x_val_rf_selected). Using this refined feature set, the trained best Random Forest model (rf_best) is used to generate predictions (y_rf_val_pred) indicating whether fraud is reported or not. In addition, the model computes class probabilities (y_rf_val_pred_prob), which represent the likelihood of each case being fraudulent. These predicted labels and probabilities will later be used to evaluate model performance, including accuracy, precision, recall, and other metrics on validation data.

8.2.2 Check accuracy of random forest model

The Random Forest model achieved a validation accuracy of 0.8100 (81%), indicating strong predictive performance and generalization capability on unseen data, though slightly lower than training accuracy, suggesting minimal overfitting.

8.2.3 Create confusion matrix



The confusion matrix shows 203 true negatives, 23 false positives, 34 false negatives, and 40 true positives, indicating good overall performance but with some misclassifications in both fraud and non-fraud predictions.

8.2.4 Create variables for true positive, true negative, false positive and false negative

True Negatives: 203, False Positives: 23, False Negatives: 34, True Positives: 40

8.2.5Calculate sensitivity, specificity, precision, recall and F1-score of the model

The model's performance metrics indicate it is highly effective at identifying non-fraud cases with a specificity of 89.82%, but struggles with fraud detection, achieving only 54.05% recall. Precision is 63.49%, showing that when the model predicts fraud, it is correct most of the time. However, the F1 score of 58.39% reflects a balance between precision and recall but highlights that fraud detection is still weak. Overall, the model favors non-fraud classification, and improvements such as threshold tuning, cost-sensitive learning, or advanced ensemble techniques may be required to enhance fraud detection.

EVALUATION AND CONCLUSION

This project focused on developing predictive models to classify insurance claims as either fraudulent or legitimate. We experimented with Logistic Regression and Random Forest, applying feature engineering, resampling techniques, and recursive feature elimination (RFECV), which identified an optimal subset of 83 features for training.

VALIDATION PERFORMANCE

Metric	Logistic Regression	Random Forest
Accuracy	0.834	0.810
Sensitivity (Recall)	0.662	0.540
Specificity	0.894	0.899
Precision	0.671	0.541
F1-score	0.667	0.584

MODEL INSIGHTS

• Logistic Regression: Delivered strong interpretability and balanced performance across metrics. Its higher recall makes it more effective for fraud detection, where failing to flag fraudulent cases can be costly.

• Random Forest: Captured complex feature interactions and showed strong robustness but had lower sensitivity on unseen data, making it less effective at identifying fraud compared to Logistic Regression.

KEY FINDINGS

• Feature engineering (ratios, vehicle age, interaction features) successfully captured important fraud signals.

• Handling rare categories and removing unique identifiers improved model generalization and reduced overfitting.

• Optimal thresholds were determined using ROC and precision–recall analysis to balance fraud detection with false positive rates.

CONCLUSION

• Logistic Regression outperformed Random Forest in recall and F1-score, making it the preferred model when minimizing missed fraud is a priority.

• Both models showed reliable performance and could potentially be combined in an ensemble for deployment.

• Ongoing monitoring and periodic retraining are essential to adapt to evolving fraud patterns.

RECOMMENDATIONS

• Deploy Logistic Regression as the primary screening tool, supported by Random Forest as a secondary check in ambiguous cases.

• Enhance fraud detection by continuously improving data quality and expanding feature sets with external data sources.

• Reassess model thresholds regularly to align with business objectives and adapt to changing fraud behaviour.