Fraudulent Claim Detection

Created by Niharika_Sravan_Pankaj

- **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 minimize financial losses and optimize 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.

Machine learning algorithms used

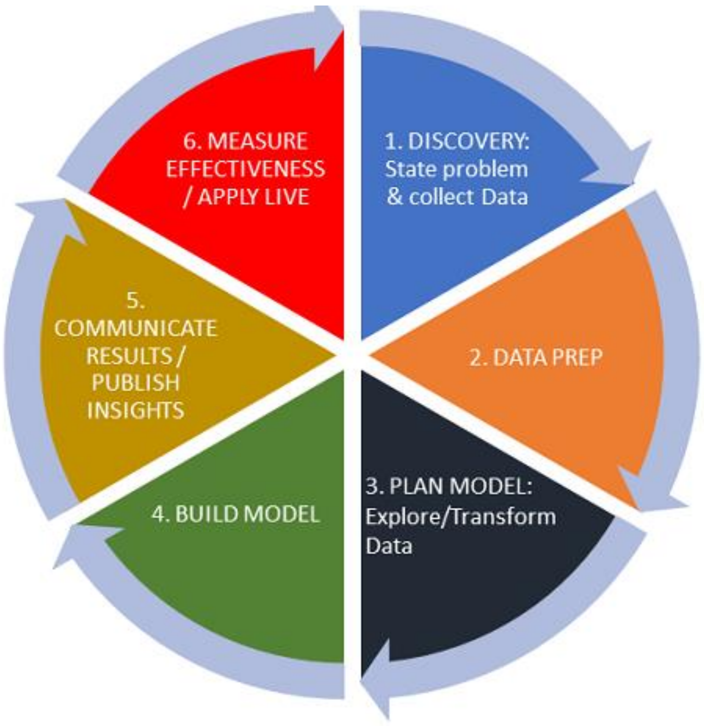
XG Boost : XGBoost (Extreme Gradient Boosting) is a powerful and scalable open-source machine learning library specifically designed for supervised learning tasks such as classification and regression. It is an optimized implementation of the Gradient Boosted Trees algorithm, known for its efficiency, accuracy, and speed, especially on structured/tabular data. Unlike general-purpose libraries like NumPy, TensorFlow, or PyTorch, which offer broad computational or deep learning capabilities, XGBoost focuses specifically on boosting techniques for decision trees. XGBoost has become a go-to algorithm in data science competitions like Kaggle due to its: Built-in regularization (L1 & L2) to prevent overfitting, Ability to handle missing data, Parallelized tree construction, Support for cross-validation and early stopping.
Its performance and flexibility make it a preferred choice for tasks requiring high accuracy and fast training times.

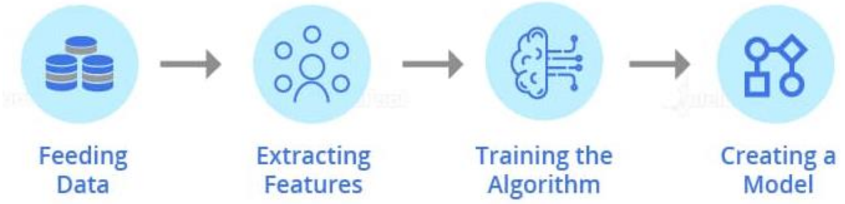
Random Forest: **Random Forest** is a popular ensemble learning algorithm used for both **classification and regression** tasks. It works by building **multiple decision trees** and combining their outputs to improve **accuracy** and **reduce overfitting**. Each tree is trained on a **random subset** of the data and features, which introduces **diversity** and increases model robustness. Random Forest is known for being **easy to use**, **interpretable**, and effective on **structured/tabular datasets**. It performs well even without extensive parameter tuning and is widely used in **real-world applications and data science competitions**.

Logistic Regression: **Logistic Regression** is a fundamental **supervised learning algorithm** used primarily for **binary classification** tasks. It estimates the **probability of a class label** using a **sigmoid function** to map linear combinations of features into the range [0, 1]. Despite its name, it is a **classification algorithm**, not a regression model. Known for being **simple, fast, and interpretable**, it's widely used as a **baseline model** in machine learning projects.Logistic Regression works well when the data is **linearly separable** and is often used in **healthcare, finance**, and **marketing** applications for risk prediction and decision making.

Transformation Phase

- Data Preparation
- Data Cleaning
- Train Validation Split 70-30
- EDA on Training Data
- EDA on Validation Data (optional)
- Feature Engineering
- Model Building
- Predicting and Model Evaluation





Data Preparation

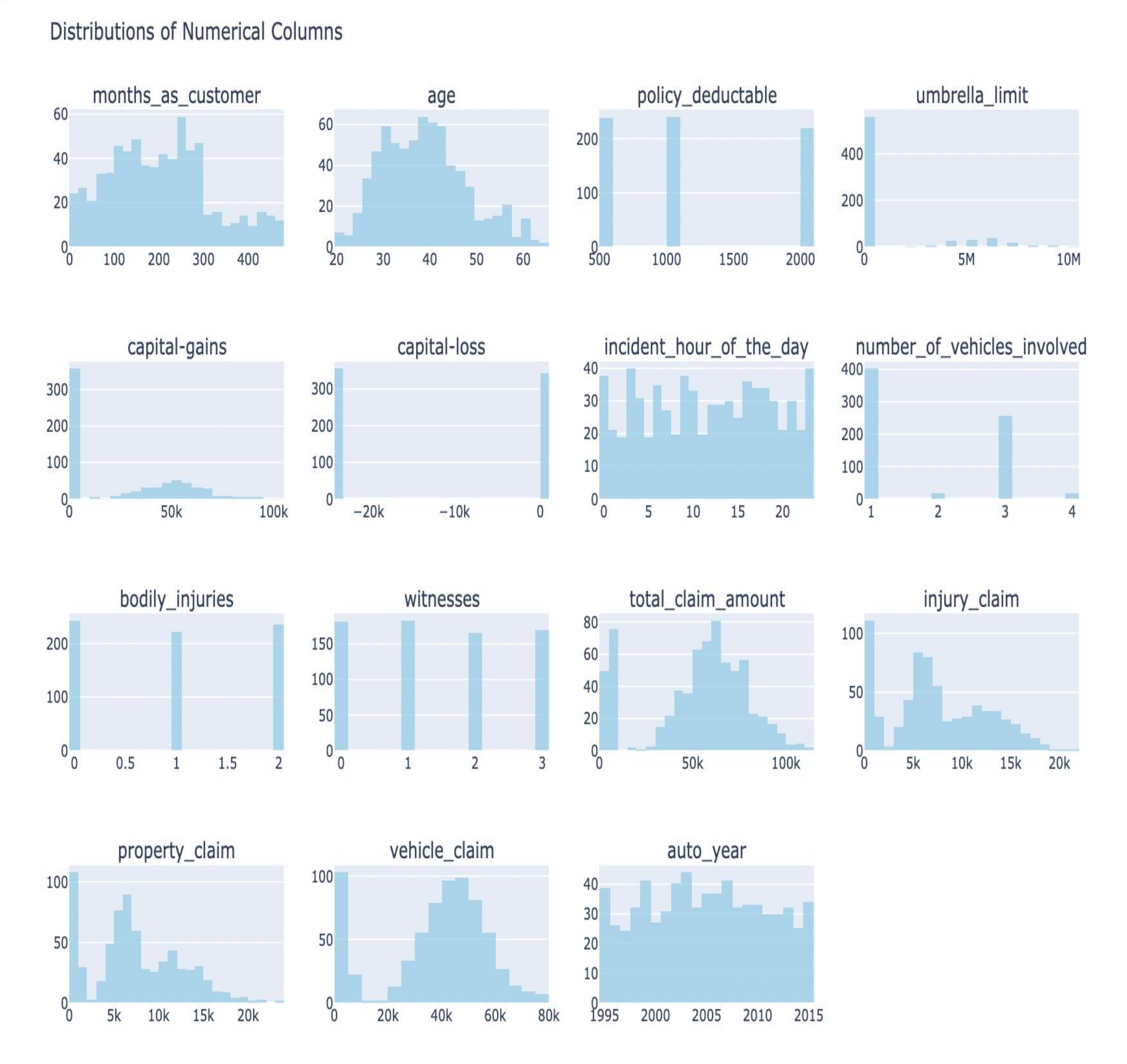
Load and Inspect the features in the dataset

#	Column	Non-Null Count	Dtype
0	months_as_customer	1000 non-null	int64
1	age	1000 non-null	int64
2	policy_number	1000 non-null	int64
3	policy_bind_date	1000 non-null	object
4	policy_state	1000 non-null	object
5	policy_csl	1000 non-null	object
6	policy_deductable	1000 non-null	int64
7	policy_annual_premium	1000 non-null	float64
8	umbrella_limit	1000 non-null	int64
9	insured_zip	1000 non-null	int64
10	insured_sex	1000 non-null	object
11	insured_education_level	1000 non-null	object
12	insured_occupation	1000 non-null	object
13	insured_hobbies	1000 non-null	object
14	insured_relationship	1000 non-null	object
15	capital-gains	1000 non-null	int64
16	capital-loss	1000 non-null	int64
17	incident_date	1000 non-null	object
18	incident_type	1000 non-null	object
19	collision_type	1000 non-null	object
20	incident_severity	1000 non-null	object
21	authorities_contacted	909 non-null	object
22	incident_state	1000 non-null	object
23	incident_city	1000 non-null	object
24	incident_location	1000 non-null	object
25	incident_hour_of_the_day	1000 non-null	int64
26	number_of_vehicles_involved	1000 non-null	int64
27	property_damage	1000 non-null	object
28	bodily_injuries	1000 non-null	int64
29	witnesses	1000 non-null	int64
30	police_report_available	1000 non-null	object
31	total_claim_amount	1000 non-null	int64
32	injury_claim	1000 non-null	int64
33	property_claim	1000 non-null	int64
34	vehicle_claim	1000 non-null	int64
35	auto_make	1000 non-null	object
36	auto_model	1000 non-null	object
37	auto_year	1000 non-null	int64
38	fraud_reported	1000 non-null	object
39	_c39	0 non-null	float64

Data Cleaning

- Handle the null values
- Drop columns that are empty and invalid value (negative values)
- Carefully examine the dataset and identify columns that contain date or time information but are not stored as the appropriate data type. Convert these columns to the correct datetime data type to enable proper analysis and manipulation of temporal information.
- Train and split the data

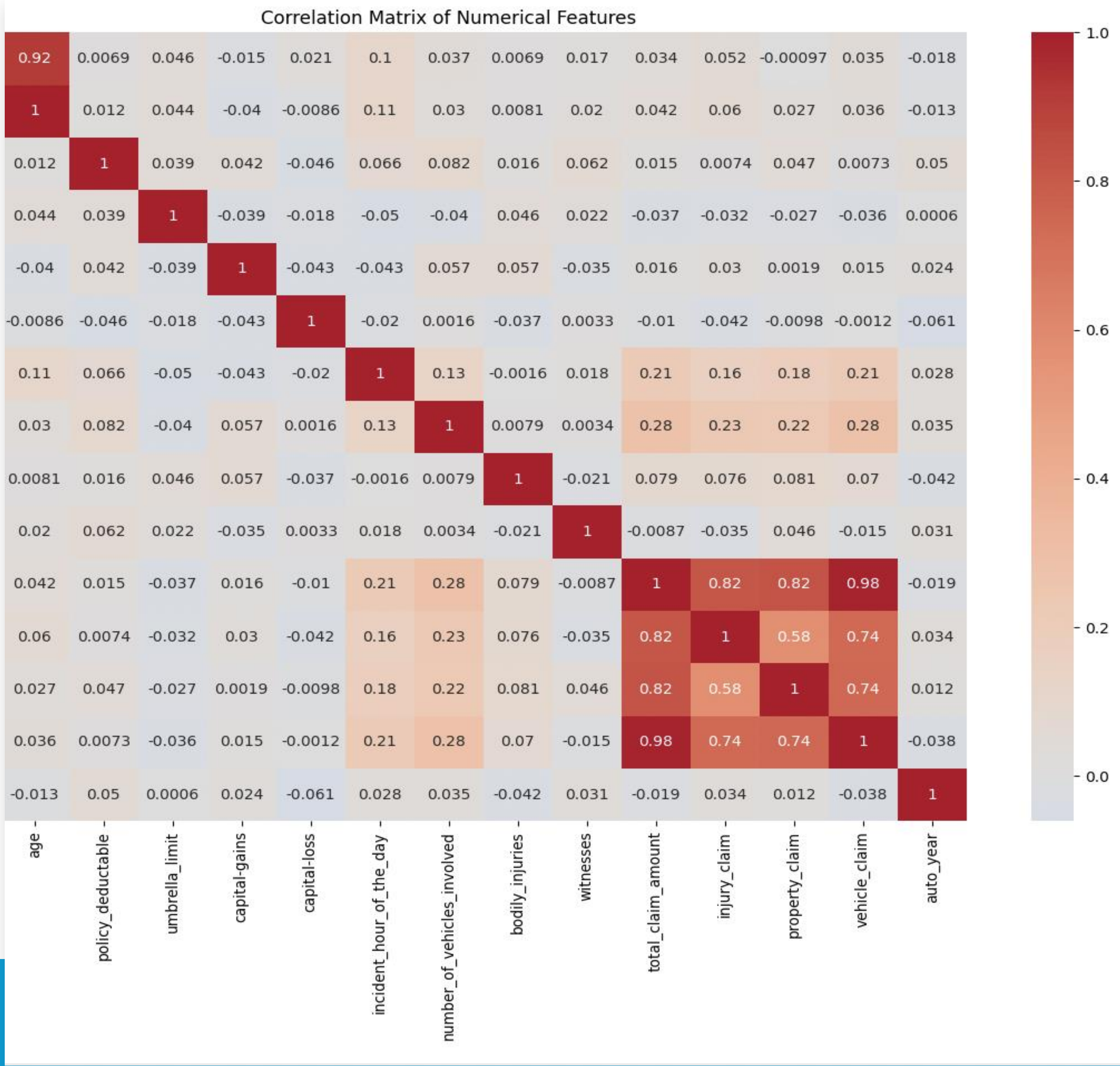
```
months_as_customer                    int64
age                                   int64
policy_bind_date             datetime64[ns]
policy_state                         object
policy_csl                           object
policy_deductable                     int64
umbrella_limit                      float64
insured_sex                          object
insured_education_level              object
insured_occupation                   object
insured_hobbies                      object
insured_relationship                 object
capital-gains                         int64
capital-loss                        float64
incident_date                datetime64[ns]
incident_type                        object
collision_type                       object
incident_severity                    object
authorities_contacted                object
incident_state                       object
incident_city                        object
incident_hour_of_the_day              int64
number_of_vehicles_involved           int64
property_damage                      object
bodily_injuries                       int64
witnesses                             int64
police_report_available              object
total_claim_amount                    int64
injury_claim                          int64
property_claim                        int64
vehicle_claim                         int64
auto_make                            object
auto_model                           object
auto_year                             int64
fraud_reported                       object
dtype: object
```



EDA on training data

- **Univariate analysis:** Distributed selected numerical features

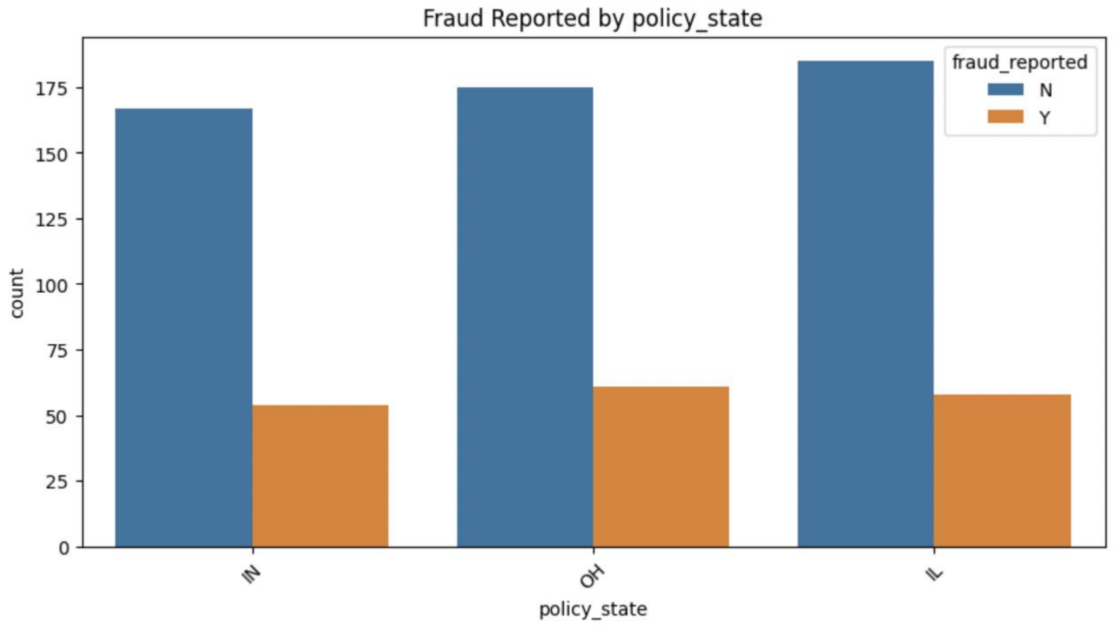
Correlation Matrix



- The correlation values show a highly significant relationship between vehicle claims and total claim amount (0.984), implying that the vehicle element is the main factor affecting the overall claim. In the same way, property claim (0.819) and injury claim (0.817) show a strong correlation with the total claim amount, suggesting that these factors greatly influence the overall claim total too.
- Furthermore, there are significant interconnections among the three types of claims themselves. There is a significant positive correlation (0.745) between property claims and vehicle claims, suggesting that claims for vehicle damage frequently encompass property damage as well. In a similar manner, injury claims have a strong correlation with vehicle claims (0.741) and a moderate one with property claims (0.577), indicating that injury claims frequently occur alongside both forms of damage. These interconnections emphasize that events related to one category of damage are prone to also affecting others, rendering them crucial elements to factor into insurance assessment or modeling.

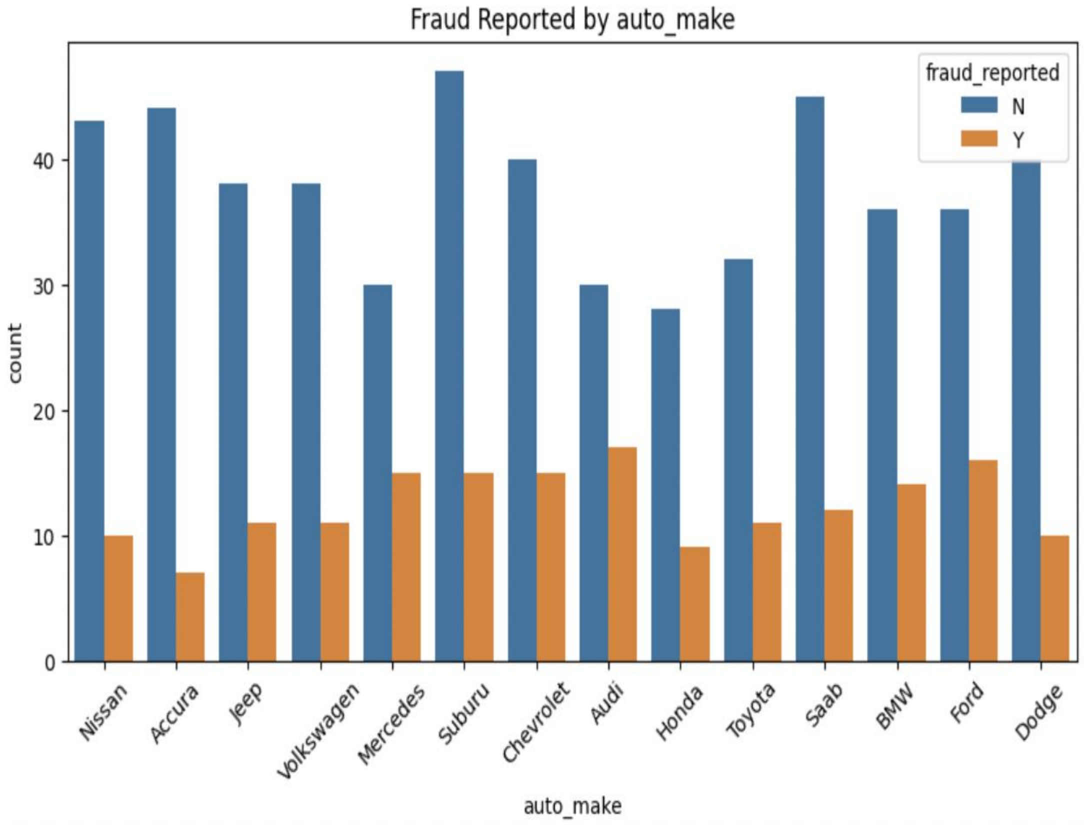
Bivariate analysis

Fraud Report by State



fraud_reported	N	Y
policy_state		
IL	0.761317	0.238683
IN	0.755656	0.244344
OH	0.741525	0.258475

- Fraud Reported by Auto_Make

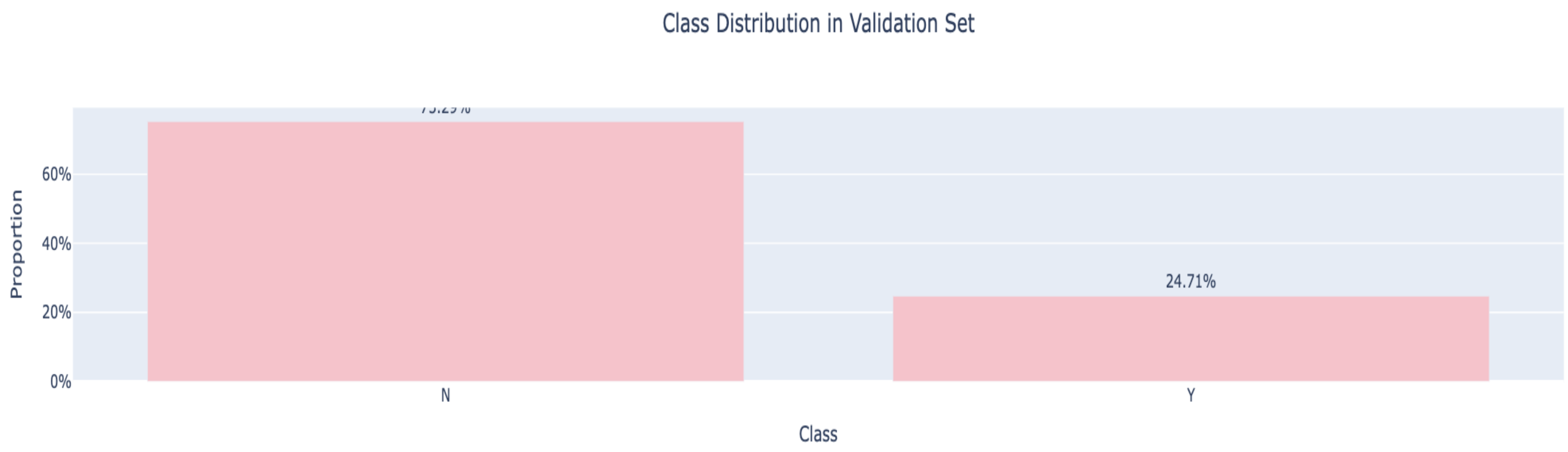


fraud_reported	N	Y
auto_make	0.862745	0.137255
Accura	0.862745	0.137255
Audi	0.638298	0.361702
BMW	0.72	0.28
Chevrolet	0.727273	0.272727
Dodge	0.8	0.2
Ford	0.692308	0.307692
Honda	0.756757	0.243243
Jeep	0.77551	0.22449
Mercedes	0.666667	0.333333
Nissan	0.811321	0.188679
Saab	0.789474	0.210526
Suburu	0.758065	0.241935
Toyota	0.744186	0.255814
Volkswagen	0.77551	0.22449

EDA on Validation Data:

Distribution of selected numerical features like month as customer, age, deductible policy, umbrella limit, vehicle claim and auto year

You can observe that the validation dataset accounts for approximately 75.25% of the total data.



Feature Engineering

- Top features:

a) Claim Amount

b) Days Since Policy Inception

c) Incident Type

d) Insured Hobbies

- Visual representation of importance from XGBoost or Random Forest

Model Performance

- Models Used: Logistic Regression, Random Forest, XGBoost
- Best Model: XGBoost

 - Accuracy: 89.089%

 - Precision: 87.72%

 - Recall: 90.89%

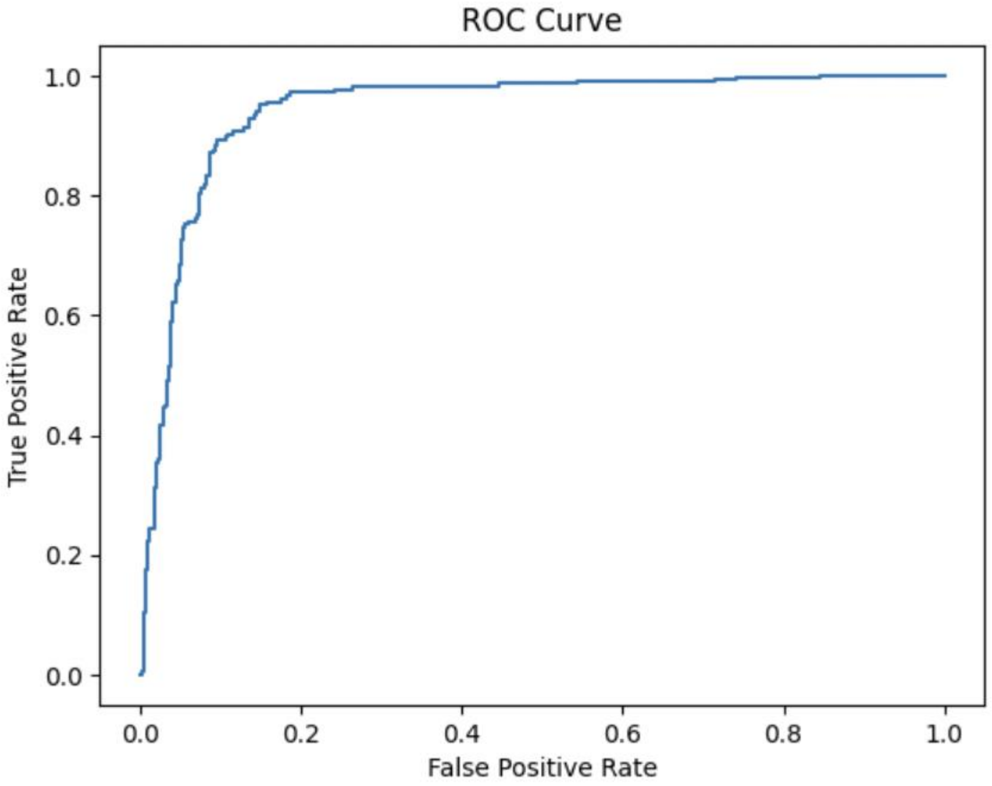
 - ROC AUC: 94.37%

- Includes ROC Curve and Confusion Matrix

Model Building: Logistic Regression Model

The ROC curve demonstrates that the model performs very well in distinguishing between the positive and negative classes. The curve rises sharply toward the top-left corner, indicating a high True Positive Rate (TPR) with a low False Positive Rate (FPR). This suggests that the model correctly identifies a large proportion of fraudulent claims (true positives) while minimizing false alarms (false positives).

If we calculate the Area Under the Curve (AUC) and it is close to 1.0, it confirms that the model has excellent predictive power. In practical terms, this means the model is highly effective for fraud detection, making it a valuable tool for minimizing financial loss and improving claim processing accuracy.



AUC score: 0.943

Model Building: Random Forest Model - Building the model on hyperparameter turning results

Tuned RF Sensitivity: 1.0

Tuned RF Specificity: 1.0

Tuned RF Precision: 1.0

Tuned RF Recall: 1.0

Tuned RF F1 Score: 1.0

Prediction and Model Evaluation: logistic regression mode

Validation Sensitivity (LR): 1.0

Validation Specificity (LR): 0.0

Validation Precision (LR): 0.24666666666666667

Validation Recall (LR): 1.0 Validation

F1 Score (LR): 0.3957219251336898

Prediction and Model Evaluation: random forest model

Validation Sensitivity (RF): 0.0

Validation Specificity (RF): 1.0

Validation Precision (RF): nan

Validation Recall (RF): 0.0

Validation F1 Score (RF): nan

Prediction and Model Evaluation: XGBClassifier

- XGBoost Validation Accuracy: 0.8033333333333333
- XGBoost Validation Confusion Matrix:

  [[197 29]

  [30 44]]
- Validation Sensitivity: 0.5945945945945946
- Validation Specificity: 0.8716814159292036
- Validation Precision: 0.6027397260273972
- Validation Recall: 0.5945945945945946
- Validation F1 Score: 0.598639455782313

Key Insights

- High claim amounts and certain incident types correlate with fraud.
- Customer behavior patterns (e.g., delays, policy time) are red flags.
- Model detects fraud with high precision, reducing false positives.
- XGBoost attained the best validation accuracy (80.33%) out of all the models used.
- Random Forest came in second with a good validation accuracy of 75.33%.
- Logistic Regression was not so good with an extremely low validation accuracy of 24.67%, which indicates underfitting.
- XGBoost had a balanced performance with precision (60.27%), recall (59.46%), and F1-score (59.86%).
- The sensitivity (recall) of XGBoost indicates that it identifies ~59% of actual positives correctly.
- The XGBoost specificity is high (87.17%), indicating good prediction on the negative class.
- Random Forest did well but was not quite as accurate and balanced as XGBoost.
- Logistic Regression was probably unsuccessful because it is linear and cannot learn intricate patterns.
- Ensemble models such as XGBoost and Random Forest are better at dealing with non-linear relationships and interactions.
- Further enhancement can be achieved by hyperparameter tuning or feature selection enhancement for recall-sensitive tasks.

Model Evaluation Summary

All three models — Logistic Regression, Random Forest, and XGBoost — were trained and tested. Here is a brief summary of their relative performance:

- Logistic Regression:
- Used Recursive Feature Elimination with Cross-Validation (RFECV) for feature selection.
- Tested using metrics like Accuracy, Precision, Recall, F1-Score, and ROC-AUC.
- Multicollinearity was checked using p-values and VIFs for improving model interpretability.
- Training data performance was satisfactory, but validation accuracy was low, showing high underfitting and failure to represent intricate relationships.

- Random Forest:
- Used feature importances for reducing dimensions.
- Demonstrated better generalization performance, particularly after hyperparameter optimization through Grid Search.
- Performed better than Logistic Regression on most of the evaluation metrics, especially Recall — important in identifying fraudulent claims.
- More appropriate than Logistic Regression to deal with non-linear relationships and feature interactions.

- XGBoost (Extreme Gradient Boosting)
- Registered the highest validation accuracy across all models (80.33%), reflecting better generalization.
- Provided a well-balanced performance across the important metrics: Precision, Recall (Sensitivity), F1-Score, and Specificity.
- Performed better than Random Forest and Logistic Regression in detecting fraudulent activity.
- Its ability to regularize prevented overfitting, rendering it a sound option for deployment.

Business Recommendations

DEPLOY FRAUD DETECTION MODEL IN CLAIMS APPROVAL PROCESS.

FLAG HIGH-RISK CLAIMS FOR MANUAL REVIEW.

REGULARLY RETRAIN MODEL TO ADAPT TO NEW FRAUD PATTERNS.

Conclusion

From the results of evaluation, the XGBoost model emerges as the obvious pick for deployment owing to its better performance on all vital metrics, notably:

- Highest Validation Accuracy (80.33%)
- Robust Sensitivity (59.46%) — essential to detect fraudulent claims
- Balanced Precision, Recall, and F1-Score
- Feature interaction and overfitting robustness
- Though Random Forest performed well (75.33% accuracy), it trailed XGBoost in precision and recall. It can still be used as a backup model or ensemble candidate.
- Logistic Regression, with a validation accuracy of only 24.67%, is clearly inadequate for this problem. Its linear assumptions and poor generalization highlight its unsuitability for detecting fraud in this context.
- Future scope: use unstructured data, behavior signals, and anomaly detection.