Detecting Deception: A Deep Dive into AI Models for Insurance Fraud

Zhiyan Chen
Department of Statistics and Actuarial Sciences
Western University
London, Canada
zche264@uwo.ca

Zhui Geng
Department of Statistics and Actuarial Sciences
Western University
London, Canada
zgeng23@uwo.ca

Jackie Niu
Department of Statistics and Actuarial Sciences
Western University
London, Canada
jniu56@uwo.ca

Wen Yang
Department of Statistics and Actuarial Sciences
Western University
London, Canada
wyang393@uwo.ca

Yuxuan Zhou
Department of Statistics and Actuarial Sciences
Western University
London, Canada
yzho553@uwo.ca

***Abstract*—Insurance fraud poses a severe financial burden on the industry, necessitating the need for automated detection mechanisms. In this study, a publicly available dataset is leveraged to explore machine learning approaches for insurance fraud classification. Techniques such as Logistic Regression, Random Forest, and Support Vector Machines were applied to uncover patterns in the data. Parameters for each model were varied to optimize performance, and results were evaluated using metrics such as F1-score and AUC. The Logistic Regression classifier, tuned via grid search, demonstrated the best performance. The findings demonstrate the effectiveness of various techniques in achieving superior classification performance, providing insights into the applicability of different algorithms for binary classification tasks. This work highlights the balance between model complexity, parameter tuning, and classification accuracy in a real-world dataset.**

***Index Terms*—Insurance Fraud Detection, Classification Model, Supervised Learning, Feature Engineering**

I. Introduction

Insurance fraud poses a significant financial burden to the industry, resulting in enormous losses each year. In the United States alone, insurance fraud costs over $308 billion annually [1]. As fraudulent claims become increasingly sophisticated, traditional rule-based detection systems struggle to keep pace with evolving tactics. This challenge necessitates the adoption of data-driven machine learning approaches, which can automatically identify complex patterns within large datasets to facilitate more informed decision-making.

This project addresses the problem of detecting fraudulent claims by developing a reproducible machine learning framework. This workflow integrates data preprocessing, model training, and performance evaluation. Multiple supervised classification models, such as Logistic Regression, Support Vector Machines (SVM), and XGBoost, were implemented and fine-tuned to determine the most effective approach for distinguishing fraudulent from legitimate claims.

In practice, such systems support claims adjusters, specialized fraud investigation units (SIU) [2], and risk professionals by enabling effective claim triage. This allows for the prioritization of high-risk cases, ensuring resources are focused on claims requiring detailed review.

Key contributions of this work include a comparison of widely used classification models for fraud detection. Additionally, a detailed discussion analyzes the specific performance drivers of each model in this context. The framework is designed for flexibility and intuitiveness, allowing for the seamless integration and testing of additional custom solutions by other users.

The report begins by reviewing prior relevant literature, followed by a detailed description of the dataset and preprocessing methodology. It subsequently outlines the system architecture, feature analysis, and modelling approach, including hyperparameter tuning and evaluation metrics. Finally, experimental results are presented, concluding with key insights and recommendations for future research directions.

II. Related Work

Fraud detection has been widely studied across financial and insurance domains, with traditional machine-learning models forming the foundation of early research. Logistic Regression has historically been favoured due to its interpretability and ability to handle high-dimensional tabular data. For instance, Dal Pozzolo et al. showed that Logistic Regression can achieve competitive performance when paired with appropriate sampling strategies on imbalanced fraud datasets [3]. However, its linear decision boundary limits its ability to capture complex nonlinear interactions commonly present in insurance claim data.

Tree-based ensemble methods, particularly Random Forest and gradient boosted trees, were later introduced to overcome such limitations. Several studies in auto-insurance fraud found that Random Forest outperforms linear models by modelling nonlinear dependencies among policyholder demographics, vehicle characteristics, and claim attributes. XGBoost and LightGBM further improved performance by introducing more flexible boosting frameworks. Although these models often achieve higher accuracy and AUC, they can be sensitive to hyperparameter choices and offer limited interpretability, which may hinder adoption in regulated insurance environments.

SVMs have also been applied to fraud detection due to their ability to maximize margins in high-dimensional feature spaces. Prior research shows that SVM can perform well in moderately imbalanced datasets, but computational cost increases sharply with dataset size, and model performance depends heavily on kernel selection [4].

More recent work has investigated deep-learning architecture and advanced resampling strategies for fraud detection. A survey by Ahmed et al. summarizes how convolutional and recurrent Neural Networks, when combined with cost-sensitive losses, can outperform traditional models [5]. Since fraud datasets are typically highly imbalanced, several authors have proposed using generative models for oversampling.

A. Limitations and Gaps in Prior Work

Despite extensive research, several limitations remain:

- Lack of standardized evaluation across model families, as many studies examine only a small number of algorithms, leading to inconsistent cross-model comparisons.
- Limited consideration of practical constraints such as interpretability. In regulated industries like insurance, transparency is essential, yet many high-performing models (e.g., XGBoost, Neural Networks) offer limited explainability.
- Insufficient focus on pipeline-level reproducibility, with many studies reporting results without a unified preprocessing and evaluation framework.

B. How This Project Addresses These Gaps

This project contributes to the literature by:

- Comparing multiple fundamentally different classification models, including Logistic Regression, Random Forest, XGBoost, SVM, and a Neural Network, within a unified experimental framework.
- Applying standardized preprocessing, hyperparameter tuning, and evaluation metrics to enable direct and fair cross-model comparisons.
- Examining trade-offs among accuracy, ROC-AUC, PR-AUC, and interpretability to highlight the strengths and limitations of each approach for insurance fraud detection.
- Ensuring reproducibility through a modular, consistent code pipeline.

Together, this approach not only benchmarks model performance but also clarifies when and why certain algorithms are more suitable for real-world insurance fraud detection tasks.

III. Methodology

A. System Architecture

To ensure reproducibility, modularity, and efficient experimentation, a unified machine-learning architecture was designed to be structured around four components.

- Data Preprocessing Module (*src/preprocess.py*) – Handles data loading, type correction, imputation, scaling, and encoding.
- Model Training Scripts (*src/models/*.py*) – Each model - Logistic Regression, Random Forest, XGBoost, SVM, and Neural Network - has an independent training script but shares the same preprocessing pipeline.
- Utility Module (*src/utils.py*) – Includes helper functions for saving trained models, logging performance to a central JSONL file, and generating leaderboard summaries.
- Leaderboard Generator (*leaderboard.py*) – Aggregates model performance across experiments and produces a ranked comparison table.

This architecture ensures that all models use the exact same data processing, enabling fair comparison across classifiers.

B. Data Preprocessing

The data processing procedure was carried out as follows:

- Load: Read the raw Excel worksheet (*Worksheet in Case Study question 2.xlsx*, sheet 0) into a pandas DataFrame.
- Missing Values: Replace the placeholder "?" with NaN. For categorical fields where missingness is informative for fraud detection, impute with the label "Unknown": `collision_type`, `property_damage`, `police_report_available`, `authorities_contacted`.
- Leakage/ID Removal: Drop near-unique identifiers and free-text or date fields to mitigate target leakage, high cardinality, and spurious memorization: `policy_number`, `policy_bind_date`, `incident_date`, `incident_location`, `insured_zip`.
- Encoding: One-hot encode all remaining categorical variables (drop first level to reduce collinearity). Preserve the target as a binary label (`fraud_reported`); if the dataset is pre-encoded, use `y` instead.
- Scaling: Standardize numeric features using `StandardScaler` (zero mean, unit variance) to stabilize optimization and ensure comparable feature magnitudes. This benefits gradient- and distance-based models (Logistic/Elastic-Net, SVM, KNN, NN), while tree-based models (RF, XGBoost, CatBoost) are scale-invariant.
- Split: Use a stratified $80/20$ train–test split with `random state` $= 42$ to preserve class proportions. The $80/20$ ratio was chosen to maximize training data while maintaining stable evaluation performance.
- Artifacts: Output $X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}}$ with all preprocessing steps applied consistently and reproducibly.

C. Algorithm Design

Several classification models were implemented to capture diverse decision boundaries and learning mechanisms relevant to insurance fraud detection. Hyperparameters were selected based on best practices for tabular data, the properties of fraud datasets (e.g., class imbalance), and computational considerations. Each model is described in detail, including its rationale, chosen hyperparameters, and the optimal settings identified during training.

1) Logistic Regression: The Logistic Regression model is a linear classifier that estimates the probability of an input belonging to a specific class using the Logistic (Sigmoid) function. By modelling the log-odds of fraud as a linear combination of features, it provides insight into which variables most strongly influence predictions. While primarily designed for binary classification, the model can also be extended to multiclass problems using multinomial Logistic Regression [6].

From a probabilistic perspective, Logistic Regression models the relationship between features and the binary outcome via the log-odds of the positive class. The key assumption is that these log-odds are a linear function of the input features:

$$\log\left(\frac{P(Y=1 \mid X=x)}{P(Y=0 \mid X=x)}\right) = \beta_0 + \beta^\top x \tag{1}$$

where β_0 is the intercept and β is the vector of coefficients.

Each coefficient β_j indicates how a one-unit change in feature j affects the log-odds of fraud: positive values increase the likelihood, and negative values decrease it. Applying the logistic function maps these log-odds to probabilities between 0 and 1:

$$P(Y=1 \mid X=x) = \frac{1}{1+\exp(-(\beta_0+\beta^\top x))} \tag{2}$$

The resulting S-shaped curve in Figure 1 is steepest near 0.5, meaning small changes in key features around the decision boundary can noticeably alter the predicted fraud probability. This linear relationship provides a clear, interpretable view of how policy and claim attributes influence fraud risk.

Logistic Regression is a discriminative linear model that estimates $P(Y \mid X)$ directly. Its interpretable coefficients and calibrated probabilities make it a reliable baseline for fraud detection and other tabular classification tasks [3], [6].

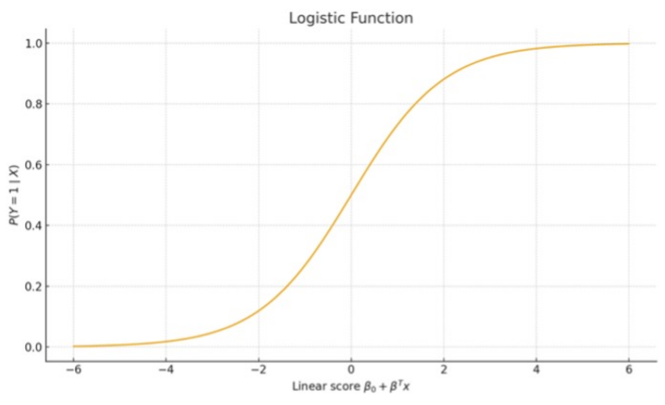

Fig. 1. Logistic function mapping the linear score to fraud probability.

Logistic Regression is a discriminative linear model that directly estimates the conditional probability $P(Y \mid X)$ for binary outcomes. Because it provides both interpretable coefficients and calibrated class probabilities, it is widely used as a baseline method for fraud detection and other tabular classification tasks [3], [6].

2) K-Nearest Neighbours: The K-Nearest Neighbours (KNN) is a simple yet effective non-parametric classification method [7]. It predicts the class label of a new observation by examining the labels of its nearest neighbours in the training set. The intuition assumes that observations that are close in feature space are likely to share similar class outcomes. Given a test observation x_0, the algorithms begin by specifying a positive integer K – the number of observations in the neighbourhood of x_0 to consider. Then, the Euclidean distance between x_0 and each training observation x_i is computed to measure similarity in the feature space.

The K observations with the smallest distances form the neighbourhood, which is then used to estimate the conditional probability of class j as:

$$\hat{P}(Y=j \mid X=x_0) = \frac{1}{K}\sum_{i\in N_0} I(y_i=j) \tag{3}$$

where I(·) is the indicator function.

The predicted class label for the test observation is then assigned according to the class with the highest estimated probability:

$$\hat{y}_0 = \arg\max_j \hat{P}(Y=j \mid X=x_0). \tag{4}$$

This process ensures that the predicted class corresponds to the majority label among the selected neighbours [8].

KNN is flexible and particularly suitable here as a small and well-structured dataset, allowing the algorithm to capture local patterns effectively with minimal tuning [7].

3) Neural Network: The Neural Network model was implemented using a multilayer perceptron architecture. A neural network is an effective choice for binary fraud detection, as fraud data is often high-dimensional, nonlinear, and highly imbalanced, conditions under which simpler linear models may struggle [9]. Since this project examines a dataset with these characteristics, a Neural Network would be a strong model.

4) Support Vector Machines: The Support Vector Machines (SVM) model is a powerful classification model that excels at handling nonlinear decision boundaries. SVM works by finding the optimal hyperplane that maximizes the margin between different classes in the feature space. To handle datasets that aren't linearly separable, SVM uses kernel functions like the Radial Basis Function (RBF) to transform the data into higher-dimensional spaces. This allows the classes to become more separable for effective classification. The chosen dataset contains both numerical and binary features, increasing the feature space's dimensions. SVM is particularly efficient at handling high-dimensional spaces, making it well-suited to capture the potential complex relationships between user data and insurance fraud detection [10].

5) Tree-Based Models: This study put more emphasis on designing tree-based models because they excel on structured tabular data, easily capture nonlinear interactions, and are robust to outliers and heterogeneous feature scales. There were three tree-based models explored: Random Forest, CatBoost, and XGBoost. Each model offers unique enhancements that influence learning behaviour and performance in a fraud detection task.

Random Forest was included as a baseline ensemble method that reduces variance by combining many bootstrapped decision trees, each trained on random subsets of features. Its predictive power depends heavily on tree complexity and feature-sampling strategy. Thus, a hyperparameter search in these areas was conducted.

CatBoost was also implemented in this project for its ability to handle categorical variables natively, as well as its strong performance on tabular data [11]. CatBoost also does not require one-hot encoding but rather converts categorical features through learned target statistics. Since the dataset contains many important categorical features, CatBoost's strength in this aspect makes it a strong model to consider. By tuning the depth, learning rate, and regularization strength, the configured model would balance performance with overfitting.

The last tree-based model considered was XGBoost. The model is most heavily used for datasets with class imbalance, nonlinear interactions, and noisy real-world features, which are all characteristics of the fraud dataset used in this project.

These three tree-based models provide a comparative perspective on ensemble learning for fraud detection. Random Forest's robust variance-reducing methods, CatBoost's unique handling of categorical variables, and XGBoost's strength in class imbalances all provide unique solutions to complex binary classification problems.

D. Model Training

For the sake of consistency, all scripts follow a unified pipeline implemented using the `scikit-learn` framework. Each training script begins by loading data through a helper function `process_data()` and defines a model-specific `param_grid` containing the hyperparameters for grid search. Model tuning is performed using `GridSearchCV` with five-fold cross-validation, which systematically evaluates all parameter combinations to identify the optimal configuration. This automatic process ensures model performance on the indicated metric while keeping a good robustness and balancing of the final model.

1) Logistic Regression: Since Logistic Regression is sensitive to model specification, it is tuned using a comprehensive hyperparameter grid on top of the same `scikit-learn` pipeline as the other models. The grid varied the penalty term (L1, L2, elastic net, or no regularization), the choice of solver (lbfgs, newton-cg, sag, saga), the regularization strength C, the elastic-net mixing parameter L1_ratio, the optimization tolerance, and the class_weight option to address class imbalance. Using `GridSearchCV` with five-fold stratified cross-validation and accuracy as the scoring metric, this grid was designed to evaluate how different regularization strategies and optimization settings influence predictive performance in the high-dimensional feature space created by one-hot encoding. The configuration that achieved the strongest cross-validated performance, while remaining stable, interpretable, and well-generalized, was then selected as the final Logistic Regression model.

This search was implemented using `scikit-learn`'s `GridSearchCV` with five-fold stratified cross-validation on the training data, using accuracy as the selection metric. Each candidate pipeline combined the shared preprocessing steps with a Logistic Regression model under a specific choice of penalty, solver, and regularization strength. This procedure ensured that every configuration was evaluated under the same data conditions and that the selected model reflected stable performance across different splits of the training set. Table I displays the selected configuration corresponds to an L1-regularized Logistic Regression with balanced class weights.

TABLE I
OPTIMAL HYPERPARAMETERS FOR LOGISTIC REGRESSION MODEL

Parameter	**Value**
penalty	L1
C	0.1
class_weight	balanced
solver	saga
l1_ratio	0
tol	1×10^{-4}

The hyperparameters in Table I lead to a sparse set of non-zero coefficients, because the L1 penalty drives many weights exactly to zero. The `saga` solver is well-suited to this setting, as it efficiently handles the large, sparse design matrices produced by one-hot encoding. At the same time, using `class_weight = ``balanced''` increases the influence of the minority fraud class during training. As a result, the final Logistic Regression model is both computationally efficient and interpretable, with its non-zero coefficients highlighting the features that are most important for distinguishing fraudulent from non-fraudulent claims.

2) K-Nearest Neighbors: As mentioned above, the training pipeline uses `scikit-learn` and a consistent structure where data is loaded through process_data(), and most models tune hyperparameters using `GridSearchCV` with five-fold cross-validation to ensure robust parameter selection.

However, the KNN classifier was trained by selecting its primary hyperparameter, the optimal number of neighbours K, through empirical evaluation. Values from 1 to 14 were evaluated using the training set, and accuracy on the test set was recorded for each model. The resulting accuracy curve seen in Figure 2 shows that performance increases initially and stabilizes around larger K values, with the highest accuracy achieved at $K = 11$. This was confirmed programmatically by selecting the value of K corresponding to the maximum test accuracy.

$K = 11$ was then used to retrain the final model on the full training set for subsequent evaluation.

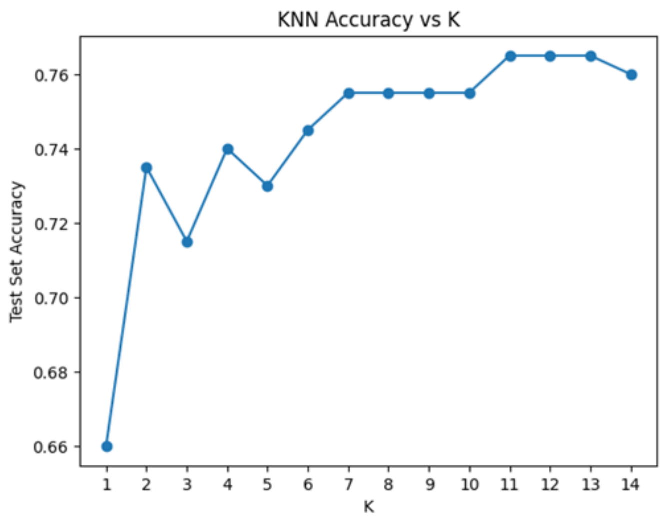

Fig. 2. KNN accuracy vs. K

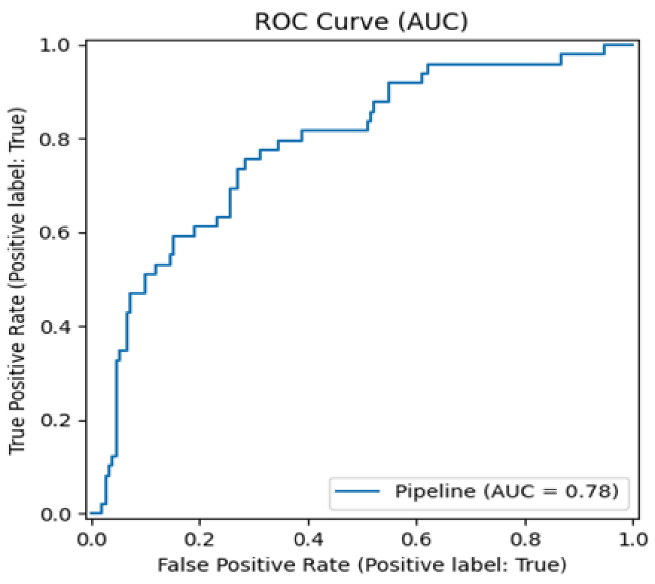

Fig. 3. Neural Network ROC curve

3) Neural Networks: Since Neural Networks are highly sensitive to architectural and optimization choices, a broad grid search was used. Different hidden-layer structures were tested, from shallow to deep networks with different numbers of neurons per layer. This allowed the model to explore varying representational capacities. Regularization strength and learning rate initialization were also tuned to test convergence dynamics. The types of activation functions and solvers were both tested to compare nonlinearities. The best Neural Network that achieves the highest cross-validated performance has the parameters displayed in Table II.

TABLE II
OPTIMAL HYPERPARAMETERS FOR NEURAL NETWORK MODEL

Parameter	**Value**
mlp_activation	tanh
mlp_alpha	0.01
mlp_hidden_layer_sizes	[128]
mlp_learning_rate	adaptive
mlp_learning_rate_init	1e-05
mlp_solver	lbfgs

Although a deep search on a large parameter grid has been performed, the result is still not satisfying. This is not beyond expectations because the dataset is too small for a Neural Network model. That can also be observed from the result of the grid search, which gives a small single layer with a size of only 128. That indicates that the optimal model still performs badly for the dataset.

To have a better understanding of the optimal Neural Network model, the ROC curve is shown in Figure 3.

The confusion matrix summarizes the model's classification performance by comparing predicted labels against true outcomes. In addition to class-level errors, the ROC curve (Figure 3) assesses the model's capability to discriminate between fraudulent and non-fraudulent claims across different decision thresholds. The curve demonstrates that the Neural Network achieves an AUC of approximately 0.78, indicating reasonably strong separability between the two classes.

4) Support Vector Machine: Within the common training pipeline, the Support Vector Machine classifier was implemented using `scikit-learn`'s SVC and tuned by means of a structured hyperparameter search. The search focused on the regularization parameter C and the kernel coefficient γ, which together control how closely the decision boundary follows the training data and how quickly it can vary in feature space. Concretely, this work considered $C \in \{0.1, 1, 10, 100, 1000\}$ and $\gamma \in \{1, 0.1, 0.01, 0.001, 0.0001\}$, with the kernel fixed to RBF. `GridSearchCV` with three-fold cross-validation was used to evaluate all combinations of these values on the training split, using ROC-AUC as the scoring metric so that models were selected based on their ability to separate fraudulent from non-fraudulent claims over the full range of decision thresholds.

After the grid search, the best hyperparameter configuration was extracted from the search object and used to fit a final SVM model on the entire training set, this time with `probability=True` to enable calibrated probability estimates for subsequent evaluation. The resulting optimal settings are summarized in Table III. A contour plot displayed in Figure 4 also proves the optimal C and γ values.

TABLE III
OPTIMAL HYPERPARAMETERS FOR THE SVM MODEL

Parameter	**Value**
kernel	rbf
C	1000
gamma	0.0001

These hyperparameters correspond to an RBF SVM with a relatively large C and small γ. A large C places a higher penalty on misclassified training examples, encouraging the

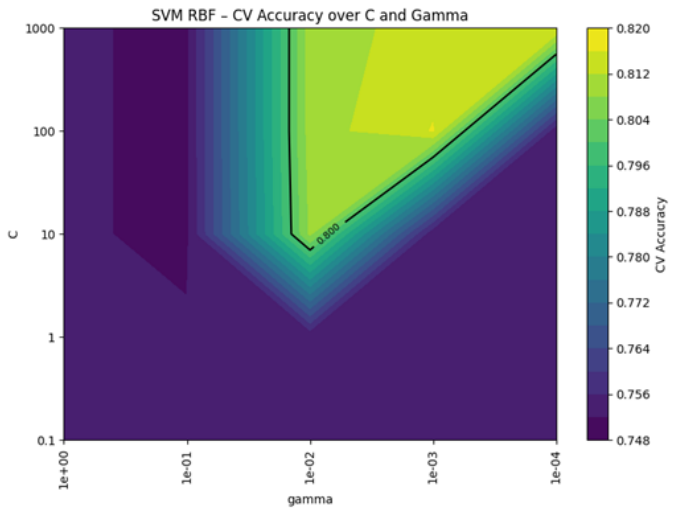

Fig. 4. Contour plot of C and gamma values for SVM model

model to fit them more closely, while a small γ yields a smoother decision surface that changes gradually across the feature space. Together, these choices allow the SVM to model nonlinear fraud patterns while keeping the decision boundary relatively smooth, which helps reduce overfitting to noise in the training data.

5) Tree-Based Models: The XGBoost model, serving as the baseline of the tree-based models, was experimented with using a grid search over a wide range of hyperparameters. The hyperparameters tested included the number of boosting rounds (between 100 to 300), tree depth (3 to 6), and subsampling (0.8 to 1), which help prevent the model from over- or underfitting while optimizing computational cost. Additionally, the learning rate (0.01 to 0.1), L2 regularization strength (1 to 10), and gamma (0 to 1) were grid searched, controlling the gradient updates and allowing for stronger generalization. The grid search results are displayed in Table IV

TABLE IV
OPTIMAL HYPERPARAMETERS FOR THE XGBOOST MODEL

Parameter	**Value**
n_estimators	200
max_depth	3
subsample	0.8
learning_rate	0.1
reg_lambda	1
gamma	1

With these hyperparameters, it suggests that shallow decision trees were enough to capture the feature interactions, implying that the patterns in insurance fraud are driven by low-order interactions rather than high-order. With 200 boosting rounds, a subsampling rate of 0.8, and a learning rate of 0.1, the model was able to learn efficiently through relatively larger steps rather than slow increments. The L2 regularization weight of 1 provided enough weight shrinkage such that extreme leaf values were prevented without sacrificing model flexibility.

6) Evaluation Metrics: To compare models under class imbalance and different error costs, standard classification metrics: accuracy, precision, recall, F1-score, ROC curves, ROC-AUC, and confusion matrices were considered. Let TP, FP, TN, and FN denote the numbers of true positives, false positives, true negatives, and false negatives in the confusion matrix. [12]

Accuracy measures overall correctness and serves as a simple check that a model does better than trivial baselines.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

However, on imbalanced datasets, it is dominated by the majority class and therefore cannot be used alone.

Precision focuses on the reliability of positive predictions,

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6}$$

indicating the proportion of claims flagged as fraud that are actually fraudulent. Precision is emphasized because fraud investigators have limited review capacity, and low precision would result in excessive false positives.

Recall (or sensitivity) is defined as

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

and measures how many truly fraudulent claims are captured by the model. In fraud detection, missing fraud is usually more costly than investigating an extra honest claim, so recall directly reflects the risk of undetected fraud.

The F1-score is the harmonic mean of precision and recall,

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

which becomes large only when both precision and recall are reasonably high. The F1-score was used as an overall summary metric to compare models when both false positives and false negatives are important, without assigning a strong preference to either precision or recall.

To evaluate discrimination across different classification thresholds, the Receiver Operating Characteristic (ROC) curve was also examined, which plots the true-positive rate against the false-positive rate as the threshold varies, and its area under the curve (ROC-AUC). ROC-AUC summarizes how well a model ranks fraudulent claims higher than non-fraudulent claims independently of a fixed threshold.

Finally, confusion matrices are reported to make the error patterns easier to interpret. They show the absolute numbers of false positives and false negatives for each model, which helps determine whether a classifier mainly tends to flag too many honest claims or to miss a substantial number of fraudulent ones.

IV. RESULTS

A. Comparison of Metrics

After the models were implemented and tested, the key metrics were tracked and stored in a table, displayed below

TABLE V
MODEL PERFORMANCE COMPARISON

Model	Accuracy	Precision	Recall	F1-Score	ROC
Logistic Regression	**0.840**	0.635	**0.816**	**0.714**	0.835
KNN	0.765	0.625	0.102	0.175	0.541
SVM	0.820	**0.659**	0.551	0.600	0.823
Neural Network	0.795	0.595	0.510	0.549	0.783
XGBoost	0.765	0.529	0.367	0.434	0.809
Random Forest	0.830	0.632	0.735	0.679	0.817
CatBoost	0.820	0.618	0.694	0.654	**0.858**

as Table V, with the best-performing model in each metric in bold.

Overall, it is observed that the Logistic Regression model achieved the best balance of performance. It delivered the highest accuracy, recall, and F1-score, along with a competitive ROC-AUC score. Despite being the simplest model, it was still very effective at identifying insurance fraud. Having a higher recall score is particularly important, as failing to identify fraud cases is often more expensive than falsely flagging legitimate claims.

CatBoost stood out in having the best ROC-AUC score and strong performance in other metrics. With the best ROC-AUC score, it was the best model to distinguish between fraudulent and legitimate insurance claims across all classification thresholds. With competitive precision and recall, CatBoost is also considered strong by performing well under class imbalance and benefiting from its native ability to handle categorical features.

Random Forest, SVM, and the Neural Network all had competitive performance relative to expectations. With both the Random Forest and CatBoost performing well, it confirms that tree models are well-suited for this study. SVM had the highest precision of the models, even if its overall performance shows that, although nonlinear interactions may exist, the relationships are still mainly linear. The Neural Network's relatively lower performance suggests that deep learning did not provide advantages beyond what ensemble tree models can in this context.

XGBoost and KNN were the most underperforming models. While the XGBoost model had a competitive ROC-AUC score, the other metrics do not show good generalization. The especially low recall shows that it struggles to capture fraudulent cases, possibly due to limited search on tree depth and regularization strength. The KNN model had the lowest metrics across the board, amplifying the limitation of distance-based models in sparse, high-dimensional data from one-hot encoding.

B. Confusion Matrix Analysis

A confusion matrix was generated for each model. This further investigation will allow us to visualize which aspects models struggled with, as well as the trade-offs between false positives and false negatives.

With the Logistic Regression model performing the best, it is important to show its strengths and weaknesses. This confusion matrix is displayed in Figure 5.

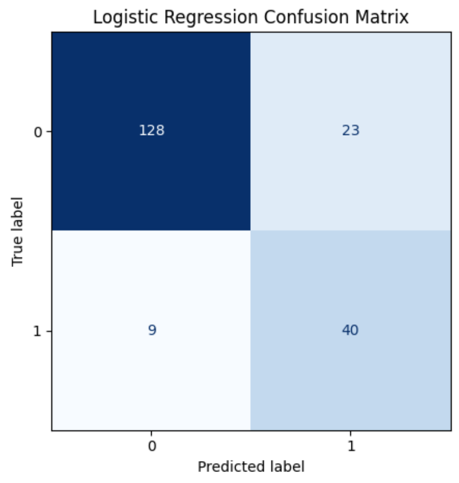

Fig. 5. Logistic Regression confusion matrix

From Figure 5, it is observed that the model performs very strongly on identifying true labels. However, it does seem to struggle with identifying a higher number of false positives due to its aggressive fraud detection.

The CatBoost model, with the highest ROC-AUC score, also gives opportunities for analysis. The confusion matrix is shown in Figure 6.

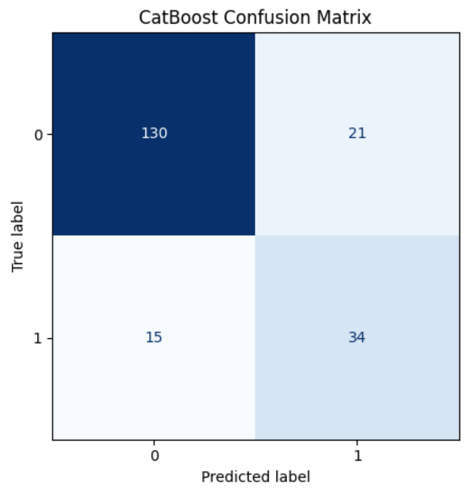

Fig. 6. CatBoost confusion matrix

From Figure 6, it is clear that the model does a marginally better job at identifying non-fraudulent cases. However, it struggles much more with identifying fraudulent cases, where roughly 30% of fraud cases were misclassified as false negatives. The Random Forest model had nearly identical confusion matrices. The XGBoost model had even worse performance on identifying true labels, with over 60% of fraudulent cases being misclassified. These tree-based models provided relatively strong detection overall, but with poor performance on identifying true positive cases, the models seem to be less desirable than the linear regression model.

The nonlinear models, KNN, SVM, Neural Network, had a variety of performances. The confusion matrix of the SVM model is provided in Figure 7.

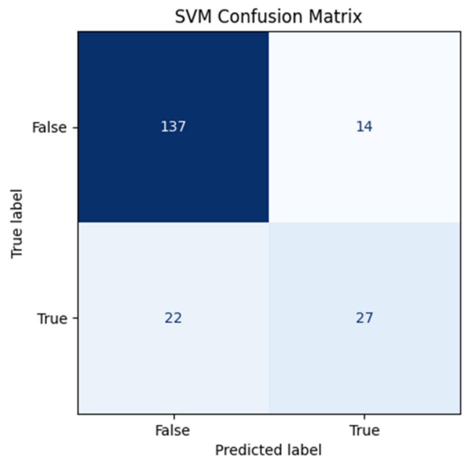

Fig. 7. SVM confusion matrix

From the confusion matrix, the SVM model surprisingly had the best performance at detecting non-fraudulent claims, at over 90%. Unfortunately, its ability to detect true positives was worse than CatBoost and Logistic Regression, where nearly half of fraudulent claims were misclassified. The Neural Network and KNN models had even more extreme cases of these issues.

There are multiple perspectives to consider when comparing false negatives and false positives. Having many false negatives would result in high costs, as many fraudulent claims could be missed, losing business's net revenue. However, having many false positives also reduces customer experience, as many legitimate claims would be denied as fraud. This could lose a business's reputation for turning away many customers, potentially losing out on future customers due to a poor perception. Although there are arguments for both cases, reducing the number of false negatives should be a higher priority, as the costs from fraudulent claims would be more detrimental to a business financially.

From the confusion matrices, it is evident that the Logistic Regression model had the strongest performance when detecting true positives, although its ability to classify true negatives is slightly worse compared to other models. However, with the highest F1-score and a competitive ROC-AUC, the Logistic Regression model would be the recommended deployment model. This combination of high efficiency and accurate flagging of fraudulent claims makes it the ideal model from a business standpoint.

V. Conclusion

A. Key Findings

Among the evaluated algorithms, **Logistic Regression** proved to be the optimal model for deployment with the highest Recall (0.816) and F1-score. Although CatBoost delivered the best overall ROC-AUC score (0.858), it failed to detect nearly 30% of fraudulent claims. In this domain, Logistic Regression's superior sensitivity is critical for minimizing the financial impact of missed fraud (False Negatives). Despite their architectural differences, both top-tier models converged on `incident_severity` and `insured_hobbies` as the most predictive features, indicating that fraud risk is consistently driven by accident details and policyholder lifestyle. Among the remaining models, Random Forest, SVM and Neural Network were second-tier models with lower fraud detection rates, while XGBoost and KNN proved unsuitable on the sparse, encoded data.

B. Future Work

Although the current framework demonstrates strong performance, several extensions could enhance robustness and practical relevance.

First, hyperparameter tuning could be expanded beyond the manual grids used in this study. Bayesian optimization tools such as Optuna can efficiently explore larger and continuous hyperparameter spaces, particularly for models such as CatBoost, XGBoost, and Neural Networks, allowing classifiers to better adapt to the dataset while controlling overfitting.

Second, dimensionality reduction and feature selection could simplify models and improve generalization. Techniques such as recursive feature elimination, regularization-based feature ranking, or principal component analysis (PCA) can remove redundant or weak predictors, resulting in faster training and inference and more stable parameter estimates without sacrificing accuracy.

Third, validating the models on additional claim datasets is important. The current analysis relies on a single public dataset, whereas insurers face evolving fraud patterns and distribution shifts. Testing on hold-out periods, different lines of business, or proprietary data would assess robustness and calibration and enable cost-sensitive evaluation in terms of expected financial loss or savings.

Finally, implementing model stacking represents a promising avenue. Combining Logistic Regression with tree-based models such as Random Forest, CatBoost, and XGBoost could balance the strengths of multiple learners and potentially improve F1-score and ROC-AUC, as demonstrated in recent automobile insurance studies [13].

References

[1] D. Bailey, "The rising tide of insurance fraud: an estimated $308B problem." Insurance News — InsuranceNewsNet, Jan. 03, 2025. https://insurancenewsnet.com/innarticle/the-rising-tide-of-insurance-fraud-an-estimated-308b-problem

[2] Insurance Training Center, "Special Investigative Unit (SIU)," 2025. https://insurancetrainingcenter.com/resource/special-investigative-unit-siu

[3] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, "Learned lessons in credit card fraud detection from a practitioner perspective," Expert Systems with Applications, vol. 41, no. 10, pp. 4915–4928, Aug. 2014, doi: https://doi.org/10.1016/j.eswa.2014.02.026.

[4] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: https://doi.org/10.1007/BF00994018.

[5] SamanehSorournejad, Z. Zojaji, Atani, Reza Ebrahimi, and Monadjemi, Amir Hassan, “A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective,” arXiv.org, 2016. https://arxiv.org/abs/1611.06439

[6] F. Lee, “Logistic Regression,” Ibm.com, May 14, 2025. https://www.ibm.com/think/topics/Logistic-regression

[7] E. Kavlakoglu, “What is the k-nearest neighbors (KNN) algorithm?,” Ibm.com, Oct. 04, 2021. https://www.ibm.com/think/topics/knn

[8] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, An Introduction to Statistical Learning: With Applications in Python, ch. 2, pp. 36–39. New York, NY, USA: Springer Nature, 2023.

[9] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and Jascha Sohl-Dickstein, “Sensitivity and Generalization in Neural Networks: an Empirical Study,” arXiv (Cornell University), Jan. 2018, doi: https://doi.org/10.48550/arxiv.1802.08760.

[10] E. Kavlakoglu, “What are SVMs?,” Ibm.com, Dec 12, 2023. www.ibm.com/think/topics/support-vector-machine

[11] A. V. Dorogush, V. Ershov, and A. Gulin, “CatBoost: gradient boosting with categorical features support,” arXiv:1810.11363 [cs, stat], Oct. 2018, Available: https://arxiv.org/abs/1810.11363

[12] A. C. Müller and S. Guido, Introduction to Machine Learning with Python: A Guide for Data Scientists, ch. 5, pp. 279–303. Sebastopol, CA, USA: O’Reilly Media, 2016.

[13] Ö, Özaltın and Ö. K. Erdemir, “Detecting automobile insurance fraud using a novel penalty-driven feature selection method with particle swarm optimization and machine learning classifiers,” *Scientific Reports*, vol. 15, no. 1, 2025, doi: https://doi.org/10.1038/s41598-025-25700-2.