ISYE 7406: Homework # 5 Report

Introduction:

The goal of this analysis is to build models that can accurately detect fraudulent transactions. Fraudulent transactions result in billions of dollars in losses each year, and detecting fraud can be challenging because fraudulent transactions are rare compared to legitimate ones. This makes it difficult for models to learn the patterns that separate fraud from legitimate purchases. I aim to compare the performance of different machine learning models to determine which one best detects fraudulent transactions.

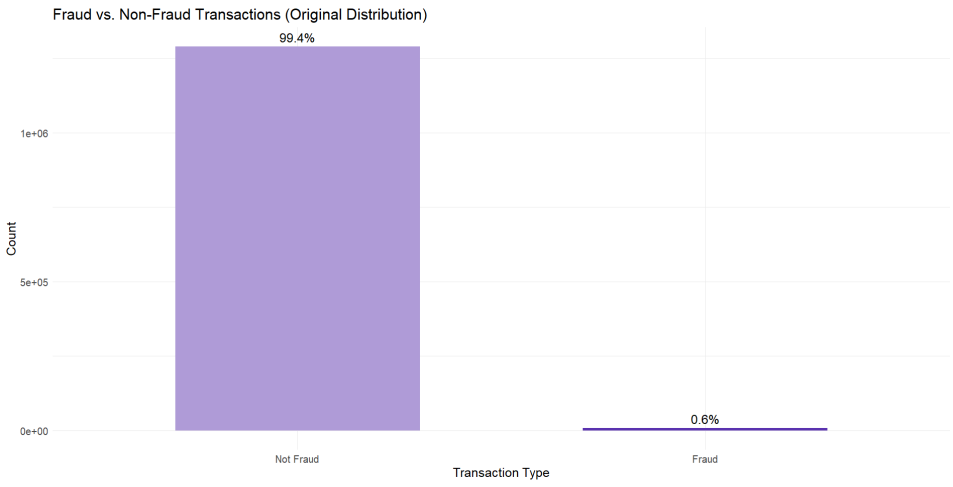
The dataset used in this analysis is the Credit Card Transactions dataset from Kaggle. It includes credit card transaction details from 1000 customers over the course of a year. It includes information like the transaction amount, location, time, purchase category, and cardholder details. However, only a small percentage of the transactions are fraudulent, making the dataset highly imbalanced.

The models in this analysis are evaluated based on key metrics like sensitivity, specificity, precision, and F1-score to find a balance between detecting fraud and minimizing false positives. The goal is to find the most effective approach for fraud detection. The results will give insights into which machine learning methods work best for this task and highlight their strengths and weaknesses.

Methodology:

During my exploratory analysis it became clear that the dataset was highly imbalanced. Fraudulent transactions made up only 0.5% of the data, while 99.5% were non-fraudulent transactions. This severe class imbalance can cause models to be biased towards the majority class, leading to misleadingly high accuracy. Accuracy alone is not a reliable metric for imbalanced datasets, so I focused on evaluation metrics like precision, recall, f1-score, and the confusion matrix to better assess model performance.

To address the imbalance, I experimented with different balancing techniques, including oversampling, undersampling, and a hybrid method. Oversampling generates synthetic samples of fraudulent transactions to balance the data, while undersampling reduces the number of majority class transactions to match the minority class. The hybrid sampling approach combines both methods to prevent information loss. ROSE package was used to implement these techniques. Ultimately I chose the hybrid approach as it provided the best balance.



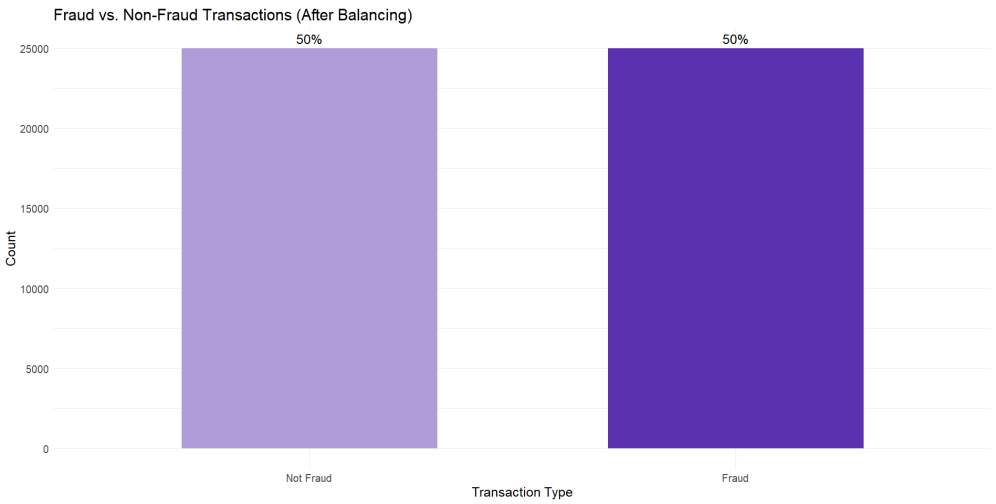


Figure 1-2. Distribution of Classes Before vs After Balancing.

Feature engineering played an important role in improving model performance. Some original features didn't seem meaningful on their own, so I created new variables to extract more insights. Age was calculated from the date of birth column, assuming that age could influence spending behavior. The transaction timestamp was split into two new features, the hour of the day and the day of the week, allowing models to detect potential time-based fraud patterns. Additionally, categorical variables like gender and state were numerically encoded so they could be used in machine learning models.

Since the dataset was so large, training models on the full dataset was computationally expensive. To make testing and tuning more efficient, I initially worked with a smaller subset of the data before tuning models on the full dataset. The data was split into a training set for model development and a test set for evaluation.

Several models were implemented and tested to compare their effectiveness in detecting fraud. Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) were included as simple linear classifiers that find a decision boundary based on statistical properties of the data. Naive Bayes, a probabilistic model that has the assumption of feature independence, was also tested. Logistic Regression was used to evaluate its ability to separate fraud and non-fraud cases. K-Nearest Neighbors (KNN) was included as a distance-based classifier, which predicts fraud based on the nearest transactions.

For the more complex models, I implemented Random Forest, an ensemble learning method that combines multiple decision trees to improve accuracy and reduce overfitting. Parameter tuning was performed by adjusting the number of trees, the number of predictors considered at each split, and the maximum depth of trees. Gradient Boosting Machines (GBM) were also tested. GBM builds trees sequentially and adjusts predictions at each step to minimize errors. Cross-validation was used to determine the optimal number of trees.

Each model was evaluated based on errors, sensitivity, specificity, precision, F1-score, and balanced accuracy. These metrics helped assess the trade-off between detecting fraud and minimizing false positives. After testing and tuning the models, the best-performing approach was identified based on its ability to balance fraud detection and overall accuracy.

Results:

The models were evaluated on its ability to detect fraudulent transactions while minimizing false positives. Below is the results table summarizing model performance.

Model	Train Error	Test Error	Sensitivity	Specificity	Precision	F1-Score	Balanced Accuracy
LDA	0.14684	0.0184	0.9831	0.5641	0.9983	0.9906	0.7736
Naive Bayes	0.16538	0.034	0.9664	0.5641	0.9982	0.982	0.7652
Logistic Regression	0.1471	0.0328	0.9687	0.5641	0.9982	0.9832	0.7664
KNN (k=3)	0.0236	0.0933	0.9043	0	0.9957	0.9478	0.4522
Random Forest	0.05088	0.0415	0.9506	0.8461	0.9994	0.9744	0.8984
Boosting	0.03458	0.0296	0.9677	0.9487	0.9998	0.9835	0.9582

Figure 3. Performance Table.

The models consistently highlighted certain features as the most important in predicting fraud. Transaction amount (amt) was by far the most influential variable across all models, especially in ensemble methods like Random Forest and Gradient Boosting. Transaction hour was also highly important, suggesting that fraudulent activity tends to follow certain time-based patterns. Category, which represents the type of transaction, also played a significant role, indicating that fraud is more common in certain types of purchases. Other moderately important features included age, unix time, and city population.

LDA and QDA had similar results, with low test errors but they struggled with specificity (~0.56), meaning they incorrectly flagged too many legitimate transactions as fraud. Balanced accuracy (~0.76-0.77) suggests they struggled to differentiate between fraud and non-fraud transactions effectively. This makes them unreliable in real-world fraud detection where false positives need to be minimized.

Naive Bayes also performed similarly to LDA but slightly worse with lower sensitivity. Since fraud detection involves correlated transaction attributes, Naive Bayes' assumption of feature independence likely contributed to its weaker performance. Logistic Regression performed moderately but lacked the precision and balance needed for fraud detection.

KNN performed poorly, with the highest testing error (0.0933), lowest balanced accuracy (0.4522), and very low specificity (=0). Specificity = 0, means it completely failed at correctly classifying any legitimate transactions, making it unsuitable for fraud detection. Multiple values of K were tested, and while K=3 had the best test error aside from K=1, the results at K=1 suggested overfitting. Even with K=3, KNN struggled with specificity and balanced accuracy, making it the weakest performer overall.

Random Forest showed strong overall performance but it initially didn't generalize well and overfitted before hyperparameter tuning. After tuning, it achieved a good balance between sensitivity and specificity, making it a strong choice for fraud detection. While it had slightly higher testing error and lower balanced accuracy than boosting, it still performed well, making it the second best model. Boosting (GBM) outperformed all other

models, achieving the lowest test error and showing the highest balanced accuracy while maintaining a good balance between specificity and sensitivity.

Conclusion:

The results highlight the challenges of fraud detection and the importance of choosing the right machine learning model. Since fraudulent transactions are rare, models need to be carefully trained to avoid bias toward legitimate transactions. Handling class imbalance was a critical step. Without balancing, models heavily favored predicting non-fraudulent transactions.

Gradient Boosting achieved the best results for fraud detection, with Random Forest as a close second. Random forest showed strong performance after tuning but still had slightly lower specificity. It required careful parameter tuning to avoid overfitting. required careful tuning to avoid overfitting. Adjusting class weights in Random Forest revealed a tradeoff—higher sensitivity detected more fraud cases but increased false positives, while higher specificity reduced false positives but missed fraudulent transactions. Finding the right balance depended on the goal and application, whether prioritizing fraud detection or minimizing disruptions for legitimate users.

Simpler models like LDA, QDA, and Naive Bayes struggled to balance sensitivity and specificity. These models detected fraud well but misclassified too many legitimate transactions, making them impractical for real-world use. Logistic Regression performed similarly to these models. KNN had the worst performance, suffering from high error rates and poor specificity.

Another major challenge was computational efficiency. The dataset was quite large, and training models, especially complex models like boosting, was time-consuming. The simpler models were computationally efficient, but lacked the precision needed for accurate fraud detection. Boosting followed by Random Forest had the highest precisions. Using a smaller subset helped with early testing, but running the full dataset significantly increased runtime, particularly for boosting and cross-validation. Managing computational costs while optimizing model performance was an ongoing tradeoff. My conclusion is Random Forest, while slightly worse than Boosting, is a strong alternative when computational efficiency is a concern.

Overall, this analysis demonstrates that ensemble methods are the best choice for fraud detection and outperform the simpler models. Future improvements could involve testing more advanced boosting techniques like XGBoost or LightGBM, experimenting with different feature engineering methods, and further tuning model parameters. In real-world applications, continuously updating models with new fraud patterns is essential to maintaining detection accuracy.