AI-Assisted Financial Fraud Detection: Exploration of Model Architecture in Performance and Accuracy

Shriyan Dhannode

Abstract:

Multiple studies have reported that modern finance is moving to the digital front, which greatly increases the amount and complexity within financial systems; traditional methods of auditing transactions have decreased in efficiency because of this. In this study, we looked into the viability of using machine learning models for auditing. Five were tested (Random Forest, Decision Tree, Logistic Regression, Support Vector Machines, and Neural Networks) using the same synthetic financial fraud dataset, and the models were also threshold-tuned, data-balanced using the imbalanced-learn library, and optimized for speed. The dataset was cleaned of any erroneous data and prepped with encoding using the category_encoders library for one-hot encoding. The Decision Tree model performed the best with a 0.91 macro average and high recall and precision in both fraud and non-fraud categories. The rest of the models then go from best to worst performance in the following order: Random Forest, Logistic Regression, Neural Networks, and Support Vector Machines. It was noted that the tree-based models did better than the other types of models. Through the identification of tree-based models being much stronger than others at financial fraud detection, financial institutions can use this to protect money more easily and efficiently.

Introduction:

The digitization of monetary exchange has made making transactions much easier. This increased ease has caused digital transactions to skyrocket over the years. In fact, since 2012, the amount of cashless transfers has doubled (Bank for International Settlements, 2026). With the rising amount of transactions, manual auditing is unable to keep up with the overwhelming amount of fraud detection work, rendering it mostly obsolete. Banks have now switched to automatic monitoring methods.

Automatic auditing covers a few different types of audit methods. They include passive methods like transaction limits and active methods like different types of rule-based algorithms and data-driven algorithms (Bolton & Hand, 2002). Currently, rule-based algorithms are the best option as they are transparent and interpretable, something that is valued in a sector where one mistake could ruin a life. However, rule-based algorithms have their downsides. As time goes on, fraud gets more advanced, especially using bots and AI. Adding more rules to counter these methods eventually slows down the entire program. Data-driven algorithms and machine learning, however, lack this disadvantage (De Gasperis & Dino Facchini, 2025). Of course, there are other problems with using machine learning, like it not being transparent or interpretable at all, and the technology is still in an experimental stage.

However, the technology has since improved. Research has been done into machine learning algorithms, and it has shown that they can analyze larger datasets and adapt to new fraud

techniques (West & Bhattacharya, 2016). As such, this research was done to test the ability of certain machine learning algorithms to detect financial fraud at a commercial scale.

I believe that the non-linear models tested will perform better than the linear models, as fraud detection is not a linear occurrence.

Methods

Data Description

The dataset used was a collection of synthetically created transaction data (https://www.kaggle.com/datasets/aryan208/financial-transactions-dataset-for-fraud-detection). In the dataset, there are five million different datum with 18 categories: transaction details (ID, timestamp, sender/receiver accounts, amount, type (deposit, transfer, etc.)), behavioral features (time since last transaction, spending deviation score, velocity score, geo-anomaly score), metadata (location, device used, payment channel, IP address, device hash), fraud indicators (binary fraud label (is_fraud) and type of fraud (e.g., money laundering, account takeover)). The target class is_fraud is severely imbalanced to simulate real-world situations where fraud is rare.

Exploratory Data Analysis (EDA)

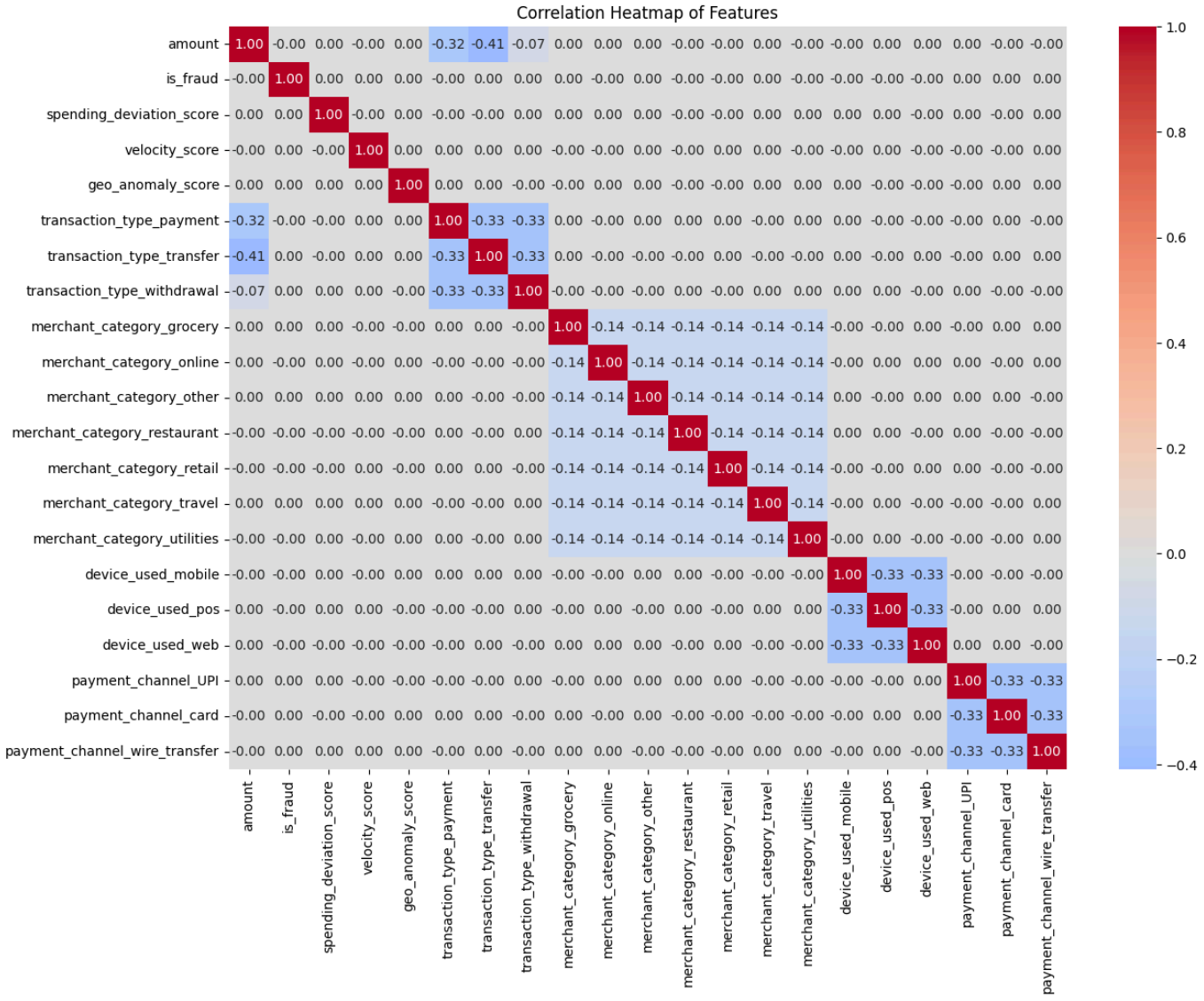
EDA was performed to understand the underlying data distributions and relationships between features. I created a custom data analysis script that validated the data in three areas: Missing Values, Extremities and Potential Wrong Data, and Correlation and Interaction Between Categories.

In the dataset, the only times a value was missing were in time_since_last_transaction when there was no last transaction and when the transaction was not fraud in the is_fraud category. Every other category had complete data.

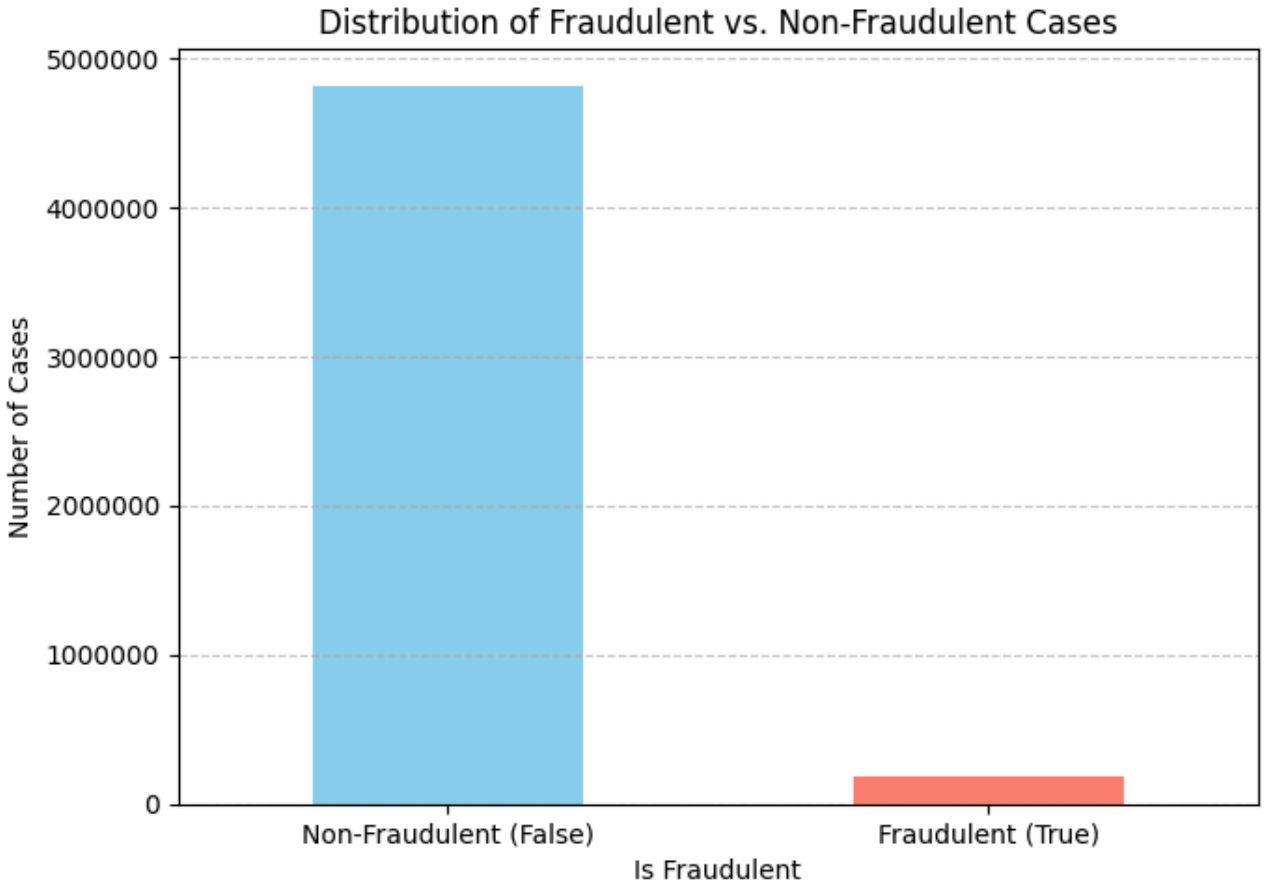
A problem was found with the time_since_last_transaction category. When checking for incorrect data, this category was found to have negative values despite storing time data. After further investigation, I concluded that when the features of the dataset were generated, the dataset may not have been sorted or that the data engineer may have used an incorrect subtraction direction. The rest of the categories had no missing data. There were extremities in the amount category of data. Multiple transactions were above the upper threshold of 1,219.87. However, 3.59% of these transactions were actually fraud. This value is similar to the total fraud percentage, suggesting that larger transactions do not correlate with fraud.

Absolutely no significant correlation (>0.7) was found between any of the categories and the is_fraud category. A generated Correlation Heatmap of the features showed correlation between some categories, such as transaction_type_transfer and amount, with a correlation coefficient of

-0.41, but the correlation was not enough to be significant



(Fig 1). However, the this_fraud class specifically has a very imbalanced set of data. In the histogram below, it can be seen that the non-fraudulent cases severely outweigh the fraudulent ones



(Fig 2).

Data Preprocessing

Preprocessing steps include categorically encoding string values to numbers using the category_encoder library, balancing of the data in the target category “is_fraud” using the imbalanced-learn library, and threshold tuning of certain models. Some of the values were Strings, so a string-to-integer converter was used to be able to train the model on those categories. Due to a high imbalance between the non-fraud and fraud instances, many of the models were only returning non-fraud in order to maximize accuracy and F1-score. Therefore, I used the imbalanced_learn library to properly use the data.

Model Selection

To examine a full range of machine learning algorithms, I diversified the types of algorithms tested in the experiment. I decided to test Logistic Regression, Decision Tree, Random Forest, Support Vector Machines, and Neural Network. The main reason for this was that I wanted to

test both linear and non-linear algorithms (ex: Logistic Regression vs Neural Network), and algorithms that range in transparency (ex: Decision Tree vs Random Forest). I worried less about training time and data amounts because enterprise-grade data centers could easily run these models quickly and use much more data than was used.

Evaluation Methods

To properly evaluate the accuracy of the algorithms, multiple performance measures were used to benchmark each one. I first used a classification report with precision, recall, accuracy, weighted and macro average, and an F1-score, each measuring the proportion of true positives among predicted frauds, the model's sensitivity to actual fraud cases, and balanced precision and recall for imbalanced data, respectively.

The ROC-AUC metric measured discrimination capability across all classification thresholds. Key visualizations included confusion matrices for each classifier to show prediction accuracy by class and ROC curves to evaluate sensitivity versus specificity. These visualizations provided insight into class separation and overall model discrimination performance, though no significant pattern deviations or data biases were identified. Each model was tested individually, with performance compared across metrics to determine which provided the best fraud detection reliability.

Code availability

Code can be found at https://github.com/boba-frog/Fraud-Detection/tree/main

Results

Compared to the rule-based algorithm, the AI models' performance varied drastically. Due to the imbalance of the is_fraud category, I will be using the macro average and not the weighted average.

The Decision Tree had the best performance, with a macro average of 0.91. Within the breakdown of the classification report, we can see that the model almost perfectly labelled all of the real transactions as real (False Recall: 0.99; Precision: 1.00) and labelled more than 75% of fraudulent transactions correctly (True Recall: 0.90; Precision: 0.78). The Accuracy Reflects that, showing an almost perfect label rate

Metric	False Class	True Class	Accuracy	Macro Avg	Weighted Avg
Precision	1.00	0.78	N/A	0.89	0.99
Recall	0.99	0.90	N/A	0.94	0.99
F1-score	0.99	0.84	0.99	0.91	0.99
Support	964,089	35,911	1,000,000	1,000,000	1,000,000

(Table 1). Additionally, it handled class imbalance effectively without excessive bias toward the

```
Confusion Matrix:
[[955081   9008]
 [  3700  32211]]
```

majority or minority class (Table 2).

The Random Forest model had a macro average of 0.23

The Random Forest Model had the highest recall (0.94). This means that the model was able to The Random Forest model achieved the highest recall (0.94), showing strong capability in identifying fraudulent transactions. However, it exhibited extremely low precision (0.04), resulting in a low F1-score (0.08). This behavior is likely due to the combination of ensemble averaging and SMOTE, which biases the model toward predicting the minority class more frequently

Metric	False Class	True Class	Accuracy	Macro Avg	Weighted Avg
Precision	0.99	0.04	N/A	0.52	0.96
Recall	0.23	0.94	N/A	0.59	0.26
F1-score	0.37	0.08	0.26	0.23	0.36
Support	964,089	35,911	1,000,000	1,000,000	1,000,000

(Table 3). As a result, the model overclassifies transactions as fraudulent, leading to a high false

```
Confusion Matrix:
[[221879 742210]
 [  2030  33881]]
```

positive rate (Table 4).

Neural Network (MLPClassifier)

The Neural Network (MLPClassifier) demonstrated similar behavior, with high recall (0.85) but very low precision (0.04), yielding an F1-score of 0.07. This outcome can be explained by the model's sensitivity to class imbalance and its tendency to overfit on extra synthetic minority samples generated through SMOTE

Metric	False Class	True Class	Accuracy	Macro Avg	Weighted Avg
Precision	0.96	0.04	N/A	0.50	0.93
Recall	0.44	0.56	N/A	0.50	0.44
F1-score	0.60	0.07	0.44	0.34	0.59
Support	964,089	35,911	1,000,000	1,000,000	1,000,000

(Table 5). Although the model has strong performance when modeling complex nonlinear patterns, a lack of

```
Confusion Matrix:
[[424676 539413]
 [ 15759  20152]]
```

(Table 6).

Logistic Regression had a moderate fraud recall (0.40). However, the fraud precision was very low (0.04). This shows that although the model was able to successfully detect 40% of all fraud, there were a huge number of false positives. On the other hand, the precision for non-fraudulent cases is 0.96, aligned with a recall of 0.60. This means that the model was able to somewhat accurately identify non-fraudulent transactions

Metric	False Class	True Class	Accuracy	Macro Avg	Weighted Avg
Precision	0.96	0.04	N/A	0.50	0.93
Recall	0.60	0.40	N/A	0.50	0.59
F1-score	0.74	0.07	0.59	0.40	0.71
Support	964,089	35,911	1,000,000	1,000,000	1,000,000

(Table 7). Furthermore, in the confusion matrix, we can see that the model predicted fraud

```
Confusion Matrix:

[[577594 386495]

 [ 21487 14424]]
```

wrong more than it did right. It did so a little (Table 8).

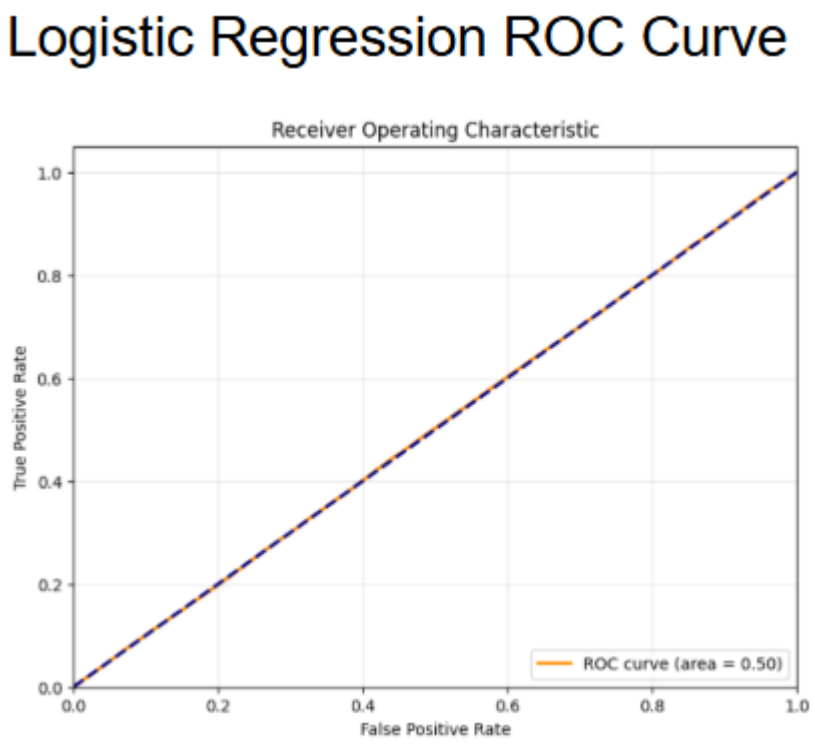
Support Vector Machine had a near-perfect false recall (0.97) but attempted to label almost none of the transactions as real. On the other hand, the model labeled nearly all transactions as fraud (1.00) and got only 4% right. It seems that the model was overly sensitive to mislabeling fraud, and therefore classified nearly all of them as such to avoid mistakes

Metric	False Class	True Class	Accuracy	Macro Avg	Weighted Avg
Precision	0.97	0.04	N/A	0.50	0.94
Recall	0.00	1.00	N/A	0.50	0.04
F1-score	0.00	0.07	0.04	0.04	0.01
Support	964,089	35,911	1,000,000	1,000,000	1,000,000

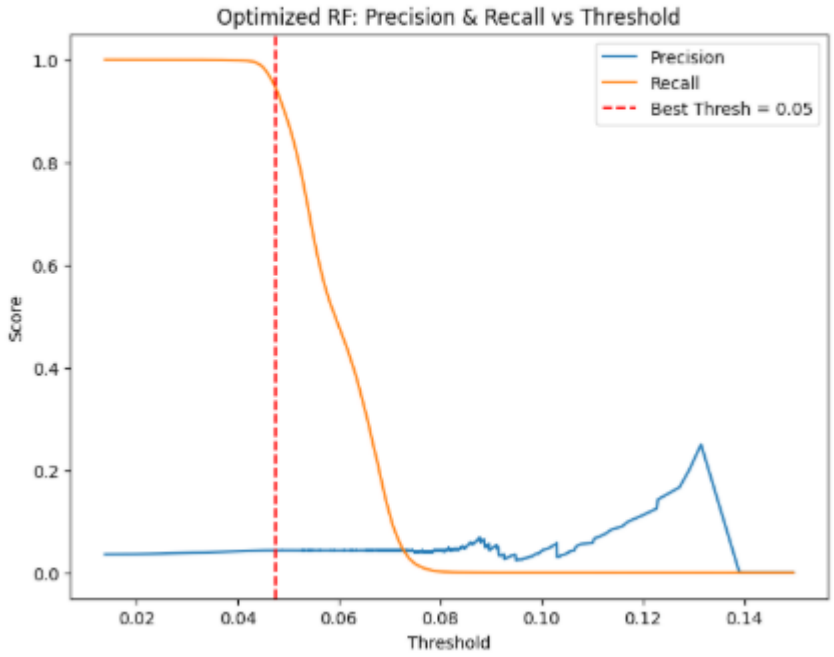
(Table 9). The confusion table furthers my thought that the model sacrificed balance for nearly 100% accuracy on catching fraud cases, mislabeling only 42 while mislabeling nearly the entirety of the non-fraudulent transactions

```
Confusion Matrix:
[[  1358 962731]
 [    42  35869]]
```

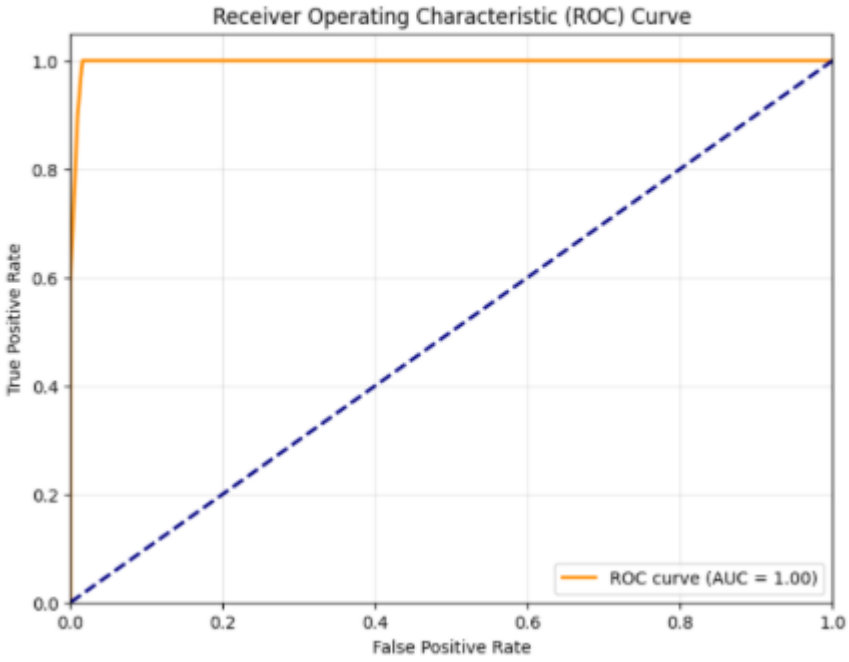
(Table 10). In the ROC-AUC curves below, it is clearly visible that the models that performed poorly were unable to be tuned properly for this task



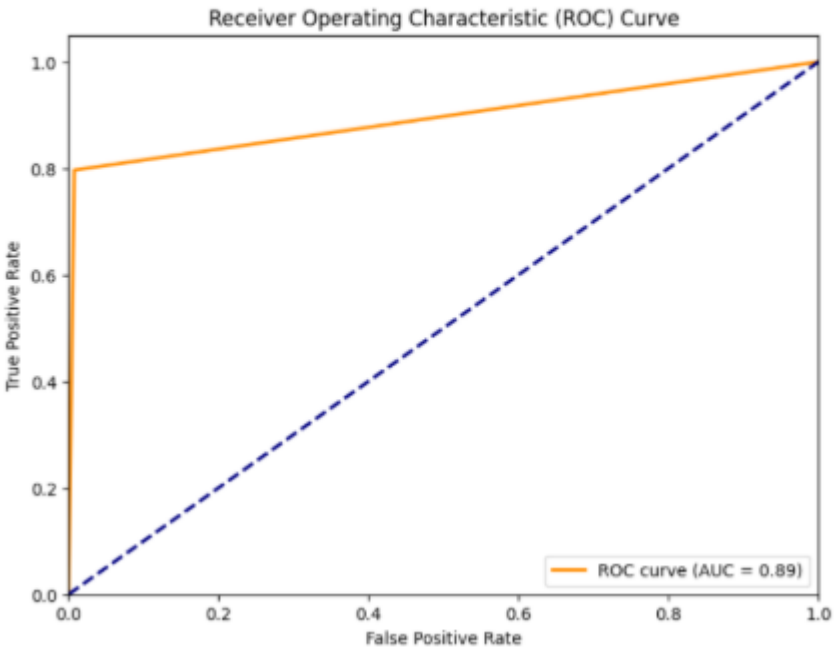
Random Forest ROC Curve



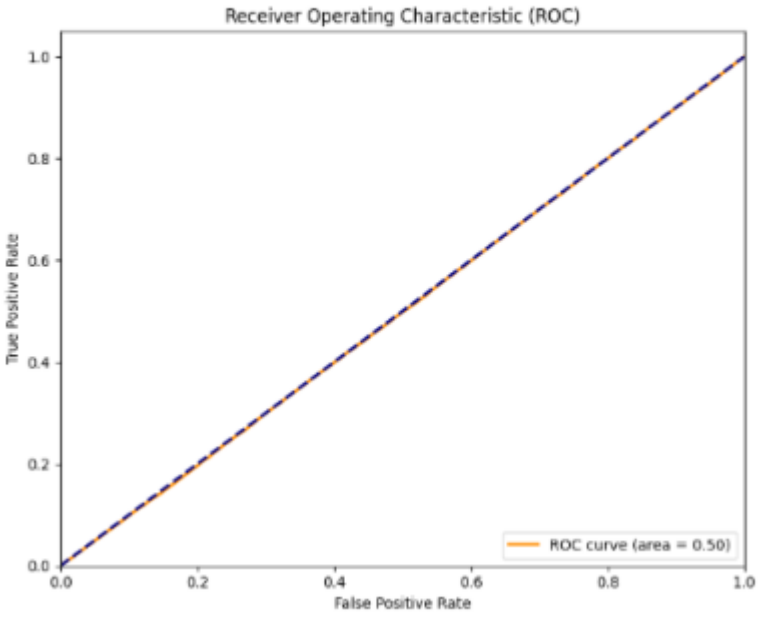
Decision Tree ROC Curve



Support Vector Machine ROC Curve



Neural Networks ROC Curve



(Fig 3).

Discussion

Overall, the results show that tree-based models like the Decision Tree and Random Forest had the best performance and that both linear or hard to tune models struggled with the task. Decision Tree clearly won with the average positive labelling rate above 90%. Because tree-based algorithms are great at capturing non-linear relationships, handle imbalanced data well, and are easier to interpret, they are the best option for fraud detection. In commercial use, a model that overlabels fraud is optimal, as the main goal of the algorithm is to capture all fraud. This model does not do that; yet, it is the best-performing algorithm out of the rest, so it is the most suited for fraud detection on this dataset in this study.

The Random Forest performed decently well for a similar reason. It is a tree-based algorithm that boasts similar perks to the Decision Tree. I wanted to note that it was challenging to tune the Random Forest to this specific dataset. There are several possible explanations for this. The algorithm may have overfitted the dataset, or the algorithm may have been biased toward non-fraud due to the imbalance of fraud and non-fraud transactions. If the problem was one of the first two reasons, then this model was not well suited to detect fraud using this dataset, as not only is it much harder to interpret, but using the Random Forest algorithm may require additional tuning compared to other models.

In fact, tuning the Neural Network and Support Vector Machine was troubling as well. Logistic Regression performed similarly to the Neural Network algorithm; both had 0.07 as their true class F1-score and similar stats around the board. This is contrary to my hypothesis because the Random Forest and Neural Network were expected to perform extremely well for detecting fraud. Overall, this is not because the models were not good for the task, but that these models contain more hyperparameters, making optimization computationally expensive on a large-scale dataset. Also, class imbalance is one of the hardest tasks, as there is usually a heavy class imbalance in real fraud datasets, which makes models favor getting only the non-fraud correct (Dal Pozzolo et al., 2015). Limited feature engineering and limited computational resources may have restricted their ability to capture fraudulent transaction patterns fully. This shows that model sensitivity is a major challenge within data-based algorithmic fraud detection. Support Vector Machine handles large datasets poorly, so I expected it to perform the worst no matter how much I tuned it.

Published research on financial fraud detection consistently reports that tree-based ensemble models like Decision Tree, XGBoost, and Gradient Boosting usually outperform both linear models and SVMs on real transactional data (West & Bhattacharya, 2016).

Different machine learning models perform differently for fraud detection, and the best model is not necessarily the most complex one. In this study, Decision Tree performed the best because it was able to identify patterns in the transaction data effectively. However, every model has tradeoffs between catching fraud and avoiding false alarms. The results show that machine learning can help with forensic auditing, but real-world implementation requires more testing and improvements.

Figure Legend

Figure 1:
“Correlation Heatmap of Features”

Figure 2:
“Distribution of Fraudulent Vs. Non-Fraudulent Cases Histogram”

Figure 3:
Figure 4:
(A: “Logistic Regression ROC Curve”)(B: “Random Forest ROC Curve”)(C: “Decision Tree ROC Curve”)(D: “Support Vector Machine ROC Curve”)(E: “Neural Network ROC Curve”)(Abbreviations: ROC = Receiver Operating Characteristic)

Table 1:
“Decision Tree Classification Report”

Table 2:
“Decision Tree Confusion Matrix”

Table 3:
“Random Forest Classification Report”

Table 4:
“Random Forest Confusion Matrix”

Table 5:
“Neural Network Classification Report”

Table 6:
“Neural Network Confusion Matrix”

Table 7:
“Logistic Regression Classification Report”

Table 8:
“Logistic Regression Confusion Matrix”

Table 9:
“Support Vector Machine Classification Report”

Table 10:
“Support Vector Machine Confusion Matrix”

Bibliography

1. Bank for International Settlements. (2026, April 27). Retail payments, currency and related indicators. Bank for International Settlements. Data.Bis.Org. https://data.bis.org/topics/CPMI_CT/tables-and-dashboards
2. Bolton, R. J., & Hand, D. J. (2002). Statistical Fraud Detection: A Review. *Statistical Science*, *17*(3), 235–255. Euclid. https://doi.org/10.1214/ss/1042727940
3. Brenner, L., Meyll, T., Stolper, O., & Walter, A. (2019). Consumer fraud victimization and financial well-being. *Journal of Economic Psychology*, *76*, 102243. ScienceDirect. https://doi.org/10.1016/j.joep.2019.102243
4. Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., & Bontempi, G. (2014). Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, *41*(10), 4915–4928. ScienceDirect. https://doi.org/10.1016/j.eswa.2014.02.026

5. De Gasperis, G., & Dino Facchini, S. (2025, September 19). A comparative study of rule-based and data-driven approaches in industrial monitoring. Arxiv. Arxiv.Org. https://arxiv.org/html/2509.15848v1#S4
6. Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2010). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, *50*(3), 559–569. ScienceDirect. https://doi.org/10.1016/j.dss.2010.08.006
7. West, J., & Bhattacharya, M. (2016). Intelligent financial fraud detection: a comprehensive review. Computers & Security, 57, 47–66. https://doi.org/10.1016/j.cose.2015.09.005