Credit Card Fraud Detection Using Machine Learning

Jeena Mahajan Rhea Mehta Sanya Verma

Abstract

Detecting fraudulent credit card transactions is a critical yet challenging task due to the rarity of fraudulent transactions and the high cost of false positives. In this project, we explored various machine learning models to address this problem, leveraging a Kaggle dataset with features such as transaction amount, time, and metadata.

Our primary models include balanced logistic regression, Support Vector Machines (SVM), Random Forests, and neural networks. Among these, different methods exhibited different strengths and weaknesses. Overall, our baseline method - balanced logistic regression - performed exceedingly well and obtained the highest F1-score, when we explicitly performed undersampling to ensure a more balanced dataset of fraudulent (minority class) and legitimate (majority class) transactions. In terms of more advanced methods, the Random Forest implementation achieved the highest precision - 1.00 - but could not achieve a higher recall than the baseline, obtaining the second highest F1-score. On the other hand, the SVM achieved somewhat average recall and precision scores; the neural network achieved a higher recall score than both the SVM and Random Forest implementation, but obtained a much lower precision value than all the methods listed above, thereby attaining the lowest F1-score.

The varying performance of the advanced methods highlight the difficulties of classification problems like fraud detection, where each method had its own strengths that could contribute towards a better prediction if combined. In future work, our aim is to further optimize model architecture and explore ensemble methods to leverage the strengths of multiple models.

1. Introduction

In this project, our aim is to develop a machine learning system to detect fraudulent credit card transactions in real-time. The system takes a single transaction as input, characterized by features such as time, amount, and other relevant metadata, and provides a probability score indicating the likelihood of fraud.

Fraudulent transactions are rare events that make their detection more challenging but critical, as the accuracy of the system directly impacts a large number of people and financial institutions. This problem addresses the need for more robust fraud detection systems that must navigate the imbalance between the rarity of fraudulent activities and the high stakes involved in their accurate identification.

Since the midterm report, we created two new models - SVM and Random Forest implementation - and improved our baseline method through undersampling techniques. In terms of final progress, we now have findings based on baseline models like balanced logistic regression (with undersampling) and more advanced methods such as SVMs, neural networks, and Random Forests that are able to successfully parse the dataset to find trends. What makes this project especially hard is that the odds of fraud are one in many, and so there is a high chance of false positives or negatives.

We have written code for these models; we aimed to tackle class imbalances by exploring different models, balancing techniques, and hyperparameter values. Our findings are that the balanced logistic regression provides an F1-score of 0.9418 after we undersampled the majority class, the SVM has an F1-score of 0.8235, the Random Forests has an F1-score of 0.8642, and the neural network has an F1-score of 0.4561. While the baseline does achieve the highest F1-score overall, it is worth analyzing the results from the other models to determine the strengths and weaknesses of each, for a more practical implementation. Random Forests are the best choice for this project because they effectively balance precision and recall, providing a high F1-score while minimizing false positives and false negatives, which is crucial in the context of detecting rare fraudulent transactions.

2. Related Work

Fraud detection has been widely explored using various machine learning techniques. (Varmedja et al., 2019) evaluated Logistic Regression (LR), Naive Bayes (NB), Random Forests (RF), and Multilayer Perceptrons (MLPs). Among

these, RF achieved the highest precision of 99.96%, while MLP, optimized with Adam, reached 99.93% accuracy. Oversampling was employed to address class imbalance; however, rare fraudulent transactions still limited overall model reliability, and complex models like RF and MLP risk overfitting without proportional performance gains.

(Afriyie et al., 2023) further explored decision trees, LR, and RF, emphasizing trade-offs between interpretability and effectiveness. Decision trees were prone to overfitting, while LR struggled with nonlinearity, and RF faced challenges in interpretability due to its ensemble structure. These methods underscore the persistent difficulty of handling imbalanced datasets, particularly in fraud detection.

Neural network-based undersampling techniques offer a promising alternative. (Arefeen et al., 2019) proposed algorithms for undersampling majority class instances, achieving superior results in terms of AUC, F1, and G-mean scores compared to traditional resampling techniques. These approaches mitigate the bias introduced by class imbalance while maintaining robust classification performance.

(Reinders & Rosenhahn, 2024) propose Neural Random Forest Imitation, a method that uses imitation learning to train neural networks to replicate the decision boundaries of random forests. Unlike direct mappings, their approach generates parameter-efficient networks that generalize well, enabling end-to-end optimization and fine-tuning. Their work demonstrates state-of-the-art performance on benchmarks, particularly in low-data scenarios, with significantly fewer parameters.

Our final proposed method is a Random Forest implementation, which we carefully tune hyperparameters like tree depth and class weights to mitigate the issues of overfitting mentioned in papers like Varmedja et al. above. This use of Random Forests not only enhances the interpretability of the model (compared to more blackbox methods like neural networks) but also improves its robustness against overfitting, particularly in cases with limited or skewed data.

3. Dataset and Evaluation

For this project we will use the "Credit Card Fraud Detection" dataset from Kaggle, which contains anonymous credit card transactions labeled as fraudulent or legitimate. This dataset is commonly used in fraud detection projects and contains a good mix of features such as transaction amount, time, and some anonymous customer data.

Data Split

1. **Training Set**: 224,000 examples (70% of the total dataset)
2. **Validation Set**: 56,000 examples (20% of the total dataset)
3. **Test Set**: 28,480 examples (10% of the total dataset)

Class Distribution

- **Legitimate Transactions**: Approximately 99.83% of the dataset (284,315 examples)
- **Fraudulent Transactions**: Approximately 0.17% of the dataset (492 examples)

Given the imbalance in the dataset (fraudulent transactions make up less than 1% of the data), we use metrics such as precision, recall, and F1-score, taking both false positives and false negatives into account.

- **Precision**: Measures the proportion of true positives among all positive predictions. It is crucial in fraud detection, as it indicates how many of the predicted fraudulent transactions were actually fraud.
- **Recall**: Measures the proportion of true positives among all actual positive cases. High recall is essential in fraud detection to minimize the risk of missing fraudulent transactions.
- **F1-Score**: The harmonic mean of precision and recall, which balances the two metrics. It is particularly useful in imbalanced datasets, providing a single score that reflects both false positives and false negatives.

4. Methods

I. Baseline Method: Balanced Logistic Regression

As a baseline, we will start with balanced logistic regression, which is a simple yet effective model for binary classification and is used as a baseline to evaluate the performance of more complex models.

- **Inputs and Outputs**:
  - Inputs: Features derived from the dataset consisting of transaction attributes (e.g., amount, time).
  - Outputs: A probability score indicating the likelihood of a transaction being fraudulent.
- **Feature Creation**: Features include transaction-related attributes directly extracted from the dataset. The dataset is preprocessed by selecting an equal number of legitimate and fraudulent transactions to balance the data before training.
- **Class Balancing**: The dataset is highly imbalanced, with far fewer fraudulent transactions compared to legitimate ones. Here, we made significant changes since

the midterm report. Namely, in our final version, we balanced the class distribution by ensuring the dataset contains an equal number of legitimate and fraudulent transactions. This ensures the model does not become biased toward the majority class (legitimate transactions). To address this, we adopted undersampling to balance the class distribution in our final version. Specifically, we randomly sampled 492 legitimate transactions (equal to the number of fraudulent transactions) to match the size of the minority class, thereby ensuring a balanced dataset. This step minimizes the risk of the model being biased toward the majority class (legitimate transactions).

- **Training Process**: The training dataset is split into 80% training data and 20% testing data, maintaining the class distribution through stratified sampling. Logistic regression is then trained on the balanced dataset to predict the likelihood of fraudulent transactions. Transactions with probabilities above a threshold (default: 0.5) were classified as fraudulent. After training, the model was evaluated on the test set to assess its performance.

II. Advanced Methods

For our more advanced machine learning methods, however, we implemented a neural network, SVM, and random forests, which we aimed to use to better able to capture non-linear relationships and therefore more complex fraud analysis.

In the midterm report, we only had the neural network as an advanced method. However, due to the limited performance of the neural network, in the final report, we have created and tested two new models - SVM and Random Forests.

A. Neural Network

Neural Networks can be effective for binary classification problems like fraud detection, as the data often displays complex patterns and relationships that may not be linearly separable. Given their ability to learn features automatically, we explored the use of a neural network in fraud detection.

- **Inputs and Outputs**:
  - Inputs: Scaled feature vectors from the dataset.
  - Outputs: A probability score indicating whether a transaction is fraudulent.
- **Feature Creation**: Similar to the logistic regression model, features were derived from transaction data and normalized.
- **Hyperparameters**: To choose hyperparameters for the neural network, the dataset was first split into three sets: training set, validation set, and testing set. We identified three different hyperparameters as the focus of our experiments - number of epochs, batch size, and dropout rate. We experimented with multiple values for each to get the best one for our final model. To determine the optimal values for these hyperparameters, we trained models on different values for each hyperparameter, and evaluated each model's performance on the validation set. The hyperparameter values that produced the best F1-score for the validation set were chosen, and thereafter evaluated on the test set to produce test accuracy. The following breakdown shows the hyperparameter values determined by the experiment (as well as values for other hyperparameters that were not part of the experiment):
  - Hidden Layers: There are two hidden layers one of 128 neuron density and other of 64 neuron density.
  - Dropout Rate: Set at 30% to help reduce overfitting.
  - Batch Size: 32; the number of samples processed before the model's weights are updated.
  - Epochs: Number of times the learning algorithm will work through the entire training dataset (10 epochs).
  - Optimizer: Adam optimizer, chosen for its efficiency in handling sparse gradients.
- **Training Process**: The neural network was trained on the scaled training set, with class weights to handle imbalance. A validation split was used to monitor performance of hyperparameters. The model hyperparameters with the best validation F1-score was used to make predictions on the test set.

B. Support Vector Machine

Support Vector Machines (SVM) are particularly effective for binary classification problems and datasets with imbalanced classes, as they allow for class weighting to mitigate the dominance of the majority class. In this project, SVM was selected due to its robustness in handling high-dimensional data and its ability to focus on the minority (fraudulent) class through balanced class weights.

- **Inputs and Outputs**:
  - Inputs: The dataset was preprocessed by normalizing features using `StandardScaler`, ensuring uniform scaling for all features. It was also rebalanced using `RandomUnderSampler` to mitigate class imbalance, ensuring that the minority class (fraudulent transactions) is adequately represented.

  - Outputs: A binary classification result where 0 indicates a non-fraudulent transaction and 1 indicates a fraudulent transaction.

- **Feature Creation**: Features were normalized to ensure consistency in scale across all features. Undersampling was performed on the data, where the minority class (fraudulent transactions) was given more weight through rebalancing.

- **Hyperparameters**:
  - **Kernel:** Radial Basis Function (RBF), suitable for non-linear data.
  - **Class Weight:** `balanced`, which adjusts the weight of each class to compensate for the imbalanced distribution of classes.
  - **Random State:** 2, for reproducibility of results.
  - **Probability Estimation:** Enabled to allow for AUC-ROC scores and better model evaluation.

- **Training Process**:
  - The training data was split into features and target, with `StandardScaler` applied to ensure that all features had similar scaling.
  - SVM was trained using the undersampled dataset, making the classification task more balanced by compensating for the dominance of the majority class.
  - The model was trained with `class_weight = 'balanced'` to ensure that the model correctly prioritizes the detection of fraudulent transactions despite their lower frequency.

C. Final Method: Random Forest

Random Forest is an ensemble method that combines multiple decision trees to make predictions. Each tree is trained on a different bootstrapped sample of the data, adding diversity to the model. For classification tasks like fraud detection, the forest aggregates the predictions of all trees (majority vote) to decide the final output, making it more robust and accurate than a single tree. Since Random Forests can detect feature importance, they are particularly effective for fraud detection, as they can help determine which variables contribute the most towards detecting fraud.

- **Inputs and Outputs**:
  - Inputs: The dataset was preprocessed by normalizing the feature vectors from the dataset using StandardScaler, ensuring uniform scaling for all features.
  - Outputs: A binary classification result where 0 indicates a non-fraudulent transaction and 1 indicates a fraudulent transaction.

- **Feature Creation**: Features were derived from transaction data and normalized to ensure consistency in scale across all features.

- **Hyperparameters**: To choose hyperparameters for the Random Forest, similar to the neural network, the dataset was first split into three sets: training set, validation set, and testing set. We identified four different hyperparameters as the focus of our experiments - number of trees in the forest, maximum depth of each decision tree, minimum number of samples required to split an internal node, and minimum number of samples required in a leaf node. We experimented with multiple values for each to get the best one for our final model. To determine the optimal values for these hyperparameters, we trained models on different values for each hyperparameter, and evaluated each model's performance on the validation set. The hyperparameter values that produced the best F1-score for the validation set were chosen, and thereafter evaluated on the test set to produce test accuracy. The following breakdown shows the hyperparameter values determined by the experiment (as well as values for other hyperparameters that were not part of the experiment):
  - Number of Trees in the Forest: There are 500 trees in the forest. [This was key, because we needed to strike the balance model instability (too few trees) versus increasing the training time or likelihood of overfitting (too many trees).]
  - Maximum depth of each decision tree: The maximum depth of each decision tree is 20. (Shallower trees prevent overfitting but may underfit the data).
  - Minimum number of samples required to split an internal node: 2; Ensures splits occur only when there are enough samples to reduce overfitting.
  - Minimum number of samples required in a leaf node: 2; Prevents the creation of small, isolated nodes that could lead to overfitting.
  - Criterion: Gini, chosen for its efficiency in measuring impurity.
  - Class Weight: Balanced; Ensures the model accounts for imbalanced class distribution
  - Bootstrap: True; Introduces randomness to avoid overfitting.

- **Training Process**:
  - The training data was split into features and target, with `StandardScaler` applied to ensure that all features had similar scaling.
  - Class weights are computed to address class imbalance, assigning higher weights to the minority

(fraud) class, which are then passed on to the Random Forest classifier.

- A Random Forest is trained for each combination of the four hyperparameter values outlined in the section above (as the experiment) using the scaled training data. A validation split was used to monitor performance of hyperparameters. The model hyperparameters with the best validation F1-score was used to make predictions on the test set.

D. Why Does the Final Method Overcome Limitations of Neural Network (Method in Midterm Report)?

In the midterm report, our advanced method was a neural network. With its many layers and parameters, it is extremely prone to overfitting, especially with imbalanced datasets. On the other hand, Random Forests use randomness in bootstrapping and feature selection to prevent overfitting. Moreover, the main issue with the neural network was its incredibly low precision value. However, Random Forests are highly precise in classifying fraud because they create clear, rule-based splits in the feature space, reducing false positives. Finally, Random Forests, as ensembles of decision trees, while simpler, are more robust to noisy patterns compared to neural networks, which will help reduce overfitting overall.

5. Experiments

Table 1 summarizes the results from all our methods: balanced logistic regression, neural network, SVM, and Random Forests. Here, the balanced logistic regression method is our baseline, the neural network retains its performance from the midterm report, and the SVM and Random Forest are newly created and tested since:

Method	Precision	Recall	F1-Score
Logistic Balanced Regression	0.9780	0.9082	0.9418
Neural Network	0.3120	0.8478	0.4561
SVM	0.875	0.7778	0.8235
Random Forest	1.0	0.7609	0.8642

Table 1. Results for Four Methods

- **Description of Results/Comparison of Methods**:
  - Balanced Logistic Regression produces the highest F1-score overall, as well as extremely high scores on precision (0.9780) and recall (0.9082). However, it is likely overfitting as it takes a very small sample of the data. By undersampling the majority class, the model is trained on a much smaller subset of the data, which might not capture the full variability of the majority class. Logistic regression, being a linear model, might fit overly specific patterns in the reduced dataset, leading to overfitting. Overall, this results in massive data loss which is why despite the high scores, we chose not to go with this model as our final method, and retain it as a baseline.
  - Neural Network, in contrast, performs poorly across several metrics with a very low precision (0.3120) and the lowest F1-Score (0.4561). While its recall (0.8478) is reasonably high (highest amongst the advanced methods), the extremely low precision indicates that it generates a large number of false positives, reducing its overall reliability for fraud detection, as it can significantly increase the cost of investigations into false positives. Moreover, neural networks can overfit to the training data. This overfitting can lead to the network memorizing minority-class examples, which boosts recall but harms generalization to new data. Thus, though this was our primary advanced method in the midterm report, we chose not to proceed with this model as our final method.
  - SVM achieves the third highest F1-score overall, and decent precision (0.875) and recall (0.7778) scores, in comparison to the other advanced methods. Like Logistic Regression, SVM was trained on an undersampled dataset, which reduces data diversity. This can lead to overfitting to the reduced dataset and poor generalization to unseen data. Moreover, SVMs are computationally intensive, especially for large datasets, as it involves solving a quadratic optimization problem. Given its decent (but not optimal) performance compared to the other advanced methods in terms of F1-score as well, it was not selected as the final method.
  - **Final Method**: Random Forest achieves the second highest F1-score (highest among all advanced methods), with a perfect precision (1.0) and a recall score of (0.7609). With its high precision, it ensures there are no false positives, which is critical to avoid unnecessary costs in fraud investigation. Moreover, while it does achieve a slightly lower recall than the logistic balanced regression, SVM, and Neural Network, logistic balanced regression and SVM were undersampled while the Random Forest was not, which means that it preserves the diversity of the original dataset and is more likely to produce better results on new data. The Neural Network is likely prone to overfitting and has a much lower precision value, whereas the Random Forest is able to achieve a good balance between both to achieve the highest F1-score out

of all advanced methods. This is likely because it combines multiple decision trees trained on bootstrapped samples, reducing variance and mitigating overfitting through averaging, to capture complex, non-linear relationships. Given the fact that it has no undersampling bias, manages class imbalance by producing a competitive F1-score, and can scale better to large datasets without requiring extensive tuning, it is the most suitable model for our final method.

- **Hyperparameter Experiments**

I. Experiment Results for Neural Network

Table 3 summarizes the results from the experiments ran to select the hyperparameter values for batch size, number of epochs, and dropout rate for the neural network. As seen from the results, the hyperparameter values of 10 epochs, a batch size of 32, and a dropout rate of 0.4 produces the highest F1-score on the development set of 0.5268817204301075. Thus, for our "best" or final model, we select the latter hyperparameter values.

Epochs	Batch Size	Dropout	F1-Score
10	32	0.3	0.13868116927260368
10	32	0.4	0.5268817204301075
10	64	0.3	0.27146814404432135
10	64	0.4	0.23809523809523808
20	32	0.3	0.35125448028673834
20	32	0.4	0.2081165452653486
20	64	0.3	0.3498233215547703
20	64	0.4	0.3535714285714286

Table 2. Hyperparameter Experiments for Neural Network

II. Experiment Results for Random Forest

Table 3 summarizes the results from the experiments ran to select the hyperparameter values for the number of trees in the forest, maximum depth of each decision tree, minimum number of samples required to split an internal node, and minimum number of samples required in a leaf node. As seen from the results, the hyperparameter values of 500 trees in the forest, a max depth of 20 for each tree, 2 samples required to split an internal node, and 2 samples required in a leaf node produces the highest F1-score on the development set of 0.81553. Thus, for our "best" or final model, we select the latter hyperparameter values.

Trees	Depth	Samples Split	Samples Leaf	F1-Score
500	10	2	2	0.81278
500	10	2	4	0.80909
500	10	5	2	0.81278
500	10	5	4	0.80909
500	20	2	2	0.81553
500	20	2	4	0.80568
500	20	5	2	0.81553
500	20	5	4	0.80568
1000	10	2	2	0.80184
1000	10	2	4	0.81447
1000	10	5	2	0.80184
1000	10	5	4	0.81447
1000	20	2	2	0.81553
1000	20	2	4	0.80568
1000	20	5	2	0.81552
1000	20	5	4	0.80568

Table 3. Hyperparameter Experiments for Random Forest

6. Discussion

To perform manual error analysis, we evaluated the final model - Random Forest- with the best hyperparameters on the development set, and identified a random sample of misclassified examples from the model's predictions for the development set to study patterns. The random sample is given in Table 4 below:

Predicted Label	Actual Label	Error Type
0	1	False Negative
0	1	False Negative
0	1	False Negative
0	1	False Negative
0	1	False Negative
0	1	False Negative
0	1	False Negative
0	1	False Negative
0	1	False Negative

Table 4. Random Sample for Error Analysis

As seen in the random sample, there are only false negatives (when the predicted label is not fraud [0] but the actual label is fraud [1]). This helps explain the lower recall score for decision trees, compared to its precision score. Given the high precision score and lower recall value, these results align with the expected misclassified examples. Since fraud cases are relatively rare in the dataset, the model seems to be biased towards predicting 0 (non-fraudulent), leading to multiple false negatives. This is expected in imbalanced datasets, and it is possible that the model struggles to correctly classify fraudulent transactions as compared

to legitimate. The random forest model uses a combination of decision trees, each using different splits of the data. The misclassified examples could represent instances where the specific features of fraudulent transactions are not well represented in the training set or are similar to legitimate transactions. Thus, overall, it seems as though the issue of class imbalance still persists to some extent, hindering the ability of the Random Forest to identify harder-to-determine fraudulent transactions.

Thus, to improve the performance of the Random Forest, we will further focus on the issue of class imbalance highlighted by this random sample. We can explore more hyperparameter configurations and investigate additional or transformed features through feature engineering to improve the ability of the Random Forest to classify hard-to-identify fraudulent transactions. Thus, while the random forest is effective at handling some aspects of fraud detection (given its decent recall score), the model's ability to capture all fraudulent transactions could still be improved.

7. Conclusion

Our work highlights the challenges of detecting fraudulent credit card transactions, particularly when dealing with class imbalance. We implemented several models, including balanced logistic regression, neural networks, SVMs, and Random Forests, to address these challenges. Among these, balanced logistic regression served as a strong baseline with the highest F1-score, but it suffered from significant data loss due to undersampling. Neural networks demonstrated a high recall of 0.8478 likely due to overfitting, with their low precision (0.3120) resulting in many false positives and limiting their overall reliability. SVMs provided a decent precision and recall score likely due to undersampling, yet they did not outperform Random Forests or logistic regression. Random Forests provided excellent precision (1.0), showcasing their robustness in reducing false positives, and had moderate recall, highlighting their balance between precision and coverage.

Compared to our initial expectations, while we expected neural networks to excel due to their capacity to model complex patterns, they struggled with precision (0.3120) and were prone to overfitting, particularly with the imbalanced dataset. While balanced logistic regression secured the highest F1-score, their reliance on undersampling made them an unreliable choice for the final method. Overall, with their ability to balance precision and recall and obtain a solid F1-score without the use of undersampling, Random Forests made the best choice for our final method.

Additionally, we learned that simpler models, such as logistic regression, can still be highly effective in tasks with imbalanced datasets when combined with appropriate balancing techniques. However, their limitations, such as data loss during undersampling, can impact scalability and real-world application. Random Forests, by contrast, provided a more practical solution, offering high precision without sacrificing interpretability, making them more adaptable for future improvements and deployment in production systems. This highlights the importance of not only selecting models based on theoretical performance but also considering their practicality and reliability in real-world scenarios.

Code and Dataset Availability

The code and dataset for this project are available at the following link: Google Drive Folder.

References

Afriyie, J. K., Tawiah, K., Pels, W. A., Addai-Henne, S., Dwamena, H. A., Owiredu, E. O., Ayeh, S. A., and Eshun, J. A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions. *Decision Analytics Journal*, 6:100163, 2023. ISSN 2772-6622. doi: 10.1016/j.dajour.2023.100163. URL https://www.sciencedirect.com/science/article/pii/S2772662223000036.

Arefeen, M. A., Nimi, S. T., and Rahman, M. S. Neural network based undersampling techniques, 2019. URL https://arxiv.org/abs/1908.06487.

Reinders, C. and Rosenhahn, B. Neural random forest imitation. *arXiv preprint arXiv:1911.10829*, 2024. URL https://arxiv.org/abs/1911.10829.

Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M., and Anderla, A. Credit card fraud detection - machine learning methods. In *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp. 1–5, 2019. doi: 10.1109/INFOTEH.2019.8717766.