# Credit Card Fraud Classification with Machine Learning

Sage Gray
*University of Tennessee, Knoxville*
sgray38@vols.utk.edu

Dylan Woods
*University of Tennessee, Knoxville*
dwoods24@vols.utk.edu

Hayden Parsons
*University of Tennessee, Knoxville*
hparson5@vols.utk.edu

Kale Dodson
*University of Tennessee, Knoxville*
lsd728@vols.utk.edu

***Abstract*— Credit card fraud is a significant problem for users and banks. Any potentially fraudulent transaction requires efficient, accurate, and prompt identification by the company and puts stress on the user, as they realize their money may not be safe. This study applies machine learning techniques to historical data in an attempt to effectively classify a charge as fraudulent or not. Exploratory data analysis (EDA) is applied to the data to root out any erroneous entries and rectify them with an appropriate solution. The model is trained and then evaluated using a confusion matrix.**

***Keywords*— Logistic Regression, Credit Card Fraud, Machine Learning, Supervised Learning, Anomaly Detection, AdaBoosted Decision Stumps**

## I. Introduction

Banks use machine learning models to detect whether a transaction is fraudulent or legitimate[1]. Fraud detection is an important factor for consumers when they decide with whom they wish to bank, and customer retention is one of the most valuable factors a bank looks for when making policy and research decisions[2]. This paper looks to find efficient and accurate methods to identify and correctly classify fraudulent transactions.

Logistic Regression is a machine learning algorithm that calculates a probability based on a series of predictor values to create a binary classification. For the purposes of credit card fraud detection, logistic regression can take historical data and classify a transaction as “Fraudulent” or “Not Fraudulent”. This classification is dictated by the “Threshold” hyperparameter, a cutoff value between 0 and 1. If the calculated probability is above the threshold, the model will classify the transaction as fraudulent, otherwise, it will classify the transaction as not fraudulent.

## II. Dataset

This data comes from the Kaggle dataset “Credit Card Fraud data”[1]. It includes the columns for transaction date and time, merchant name, category of transaction, transaction cost, city of credit card holder, state of credit card holder, latitude location of purchase, longitude location of purchase, credit card holder’s city population, job of credit card holder, and the designator for if the transaction was fraudulent or not.

### *A. Data Analysis*

The data analysis started with using the cleaned data, which was done to make the dataset more compatible with machine learning. First, the data was made into a pairplot using the Seaborn Python API. This plot shows how all of the integral data interact with one another. This helps show how the data can be used later for the machine learning algorithm that is chosen. This type of plot was not able to show all of the features of the dataset because of the type of graphs it made. Because of this, we had to make more graphs so we were able to see how the rest of the features represented whether or not the sample was fraudulent. When looking at the rest of the features, we started to look for relations between purchase type, city, and even the merchant that made the sale.

### *B. Data Preprocessing*

The work began with identifying any problems that may have occurred during the data collection or entry steps of the data lifecycle. The errors found in these data include data that are incorrectly entered, duplicated, or incorrectly formatted. These issues were rectified in the first phase of this project. Two entries in the "is_fraud" column had values that looked to be an entry accidentally including both a date and an actual value. The date aspect was removed, then the entire column was converted to an integer value to allow the model to access and interpret the values more easily. Duplicate entries were dropped from the dataframe, and no values of NA were found, so the information was ready to be modeled.

## C. *Visualizations*

First, the data was made into a pairplot using the Seaborn Python API. This plot shows how all of the integral data interact with one another. The plot uses a coloring of blue and orange to show whether the specific sample was found to be fraudulent or not. This type of plot was not able to capture all of the features that the dataset has. After the pairplot, we created smaller subsets of the whole dataset to be able to easily create graphs that represent the rest of the useful features of the dataset. First, we looked at the relationship between the type of transaction and fraud, creating a bar graph that shows the number of fraud and non-fraud samples per transaction category, as can be seen below in Fig 1. Then, we made step graphs that show the relationship between cities, jobs, and merchants. There were too many different jobs, merchants, and cities, so we showed the top 10 of each with the highest amount of fraud, which can be seen in Fig 2 below. Each graph also has the number of non-fraud to see the comparison to make sure we do not have too many outliers in the dataset.

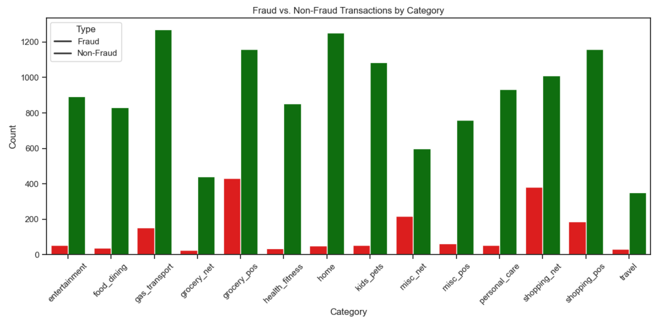


Fig. 1 Plotting Fraud Count per Category

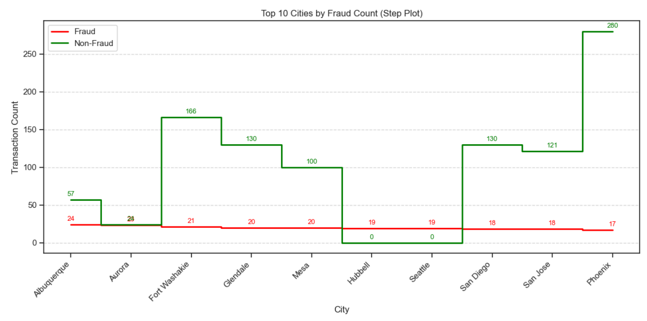


Fig. 2 Step Plot of Fraud Count and Non-Fraud Count per City

## III. Technical Approach

The model used for this problem was Logistic Regression, classifying each transaction as Fraudulent or Not Fraudulent based on the total cost of each transaction. The data was split into train and test groups of sizes 11,506 and 2877 respectively.

The Python library "scipy" was used to find an optimal θ of [-3.35203114  0.00752193] on the training dataset. Then, a series of thresholds between 0.1 and 1.0 were tested via a confusion matrix to determine the most accurate classifications of the model. The old adage goes "better safe than sorry", so the model that had the smallest number of false negatives was chosen to be the most successful model. This threshold was found to be 0.1 with amounts less than or equal to $153.52 being classified as "Not Fraudulent" and values higher than that as "Fraudulent", with a classification accuracy of 84.4% when predicting Fraudulent and a 92.33% accuracy when predicting Not Fraudulent. The regression plot can be seen below in Fig 3, and its corresponding confusion matrix can be found in Fig 4. This model is proven to be more accurate than the naive model, which assumes that all values are the majority. The naive model had an accuracy of 87.61% when predicting Not Fraudulent and was found to be 12.39% when predicting Fraudulent. When identifying a model that is superior to the baseline, one that has a lower percentage of false negatives will be more important than one that misclassifies non-fraudulent transactions. Incorrectly identifying a transaction as

"Non-Fraudulent" would be a serious blow to customer satisfaction and retention, and would likely provoke a customer to spend a significant amount of time on the phone, and would cost the bank time and resources to identify and correct the fraudulent charge(s). The model that surpasses the logistic regression model should be able to classify more accurately, and these incorrect classifications should lean more towards false positives than false negatives.

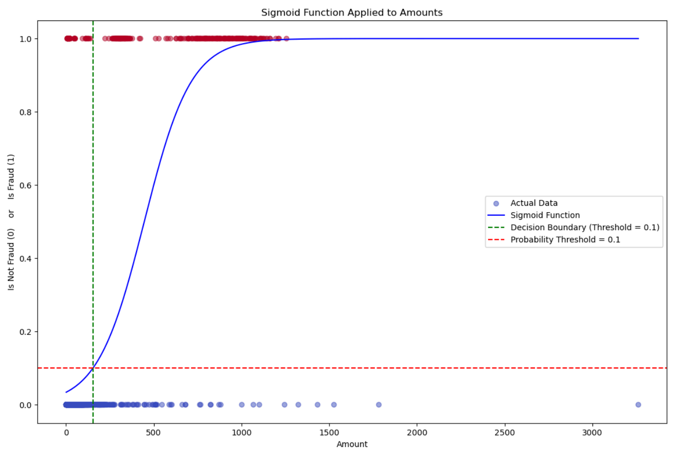


Fig. 3 The Sigmoidal Classifier at Threshold = 0.1

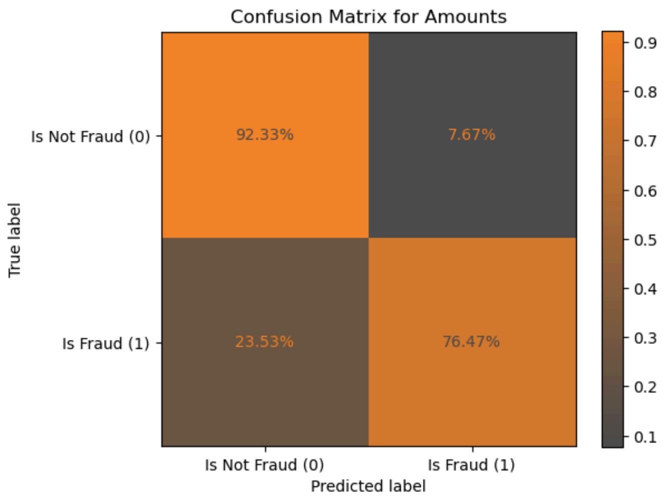


Fig. 4 The Confusion Matrix for Threshold = 0.1

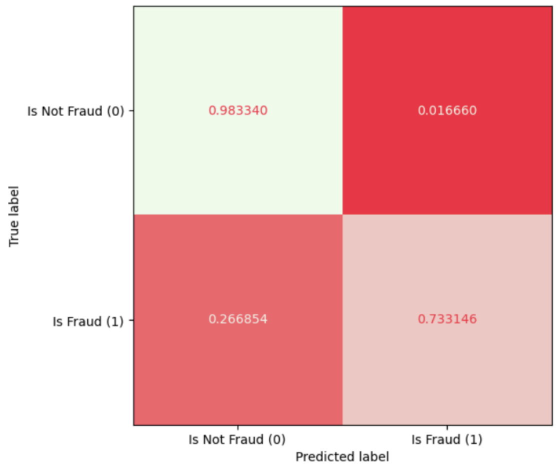
Three advanced models were used to try and increase the true positive and false positive values:

- AdaBoosted Decision Stumps, which uses a series of binary classifications and repeats the process after emphasizing correcting the wrong classifications.
- Bagging Classification, a method that randomly samples, with repetition, the original dataset to use the Wisdom of the Crowd to lower erroneous classifications.
- Random Forest, an extension of bagging that uses decision trees and sampling with repetition to classify subsets, then refers back to holdout data to double-check its results.

A grid-search cross-validation approach was used to increase the accuracy of each of these models. The grid search was used to be able to find the best hyperparameters for each of the models we tried. To find these optimal hyperparameters, an exhaustive search is done for all the possible values of the parameter.

## IV. Results & Discussion

After running cross-validation with 5 folds, the decision stump model came out on top with a 95% accuracy. Figure 6 shows the Receiver Operating Characteristic (ROC) curve for our AdaBoosted decision stump model. The precision of this model, defined as the number of true positives out of all classified positives, was 96%. The recall of this model, defined as the number of true positive classifications out of the total number of actual positives, was found to be 98%. Last, the F-1 score, essentially the tradeoff between recall and precision, was 97%. The last metric used to evaluate the accuracy of the model was ROC/AUC, or Receiver Operating Characteristic/Area Under the Curve. This was found to be 94%, indicating that 94% of the time, this model will accurately classify a charge as fraudulent when provided with new information. For our AdaBoosted decision stump model, we performed extensive hyperparameter tuning using grid search with 5-fold cross-validation. We explored a range of parameter combinations, focusing on three key hyperparameters that significantly influence model performance: n_estimators, learning_rate, and the algorithm used. First, we set up n_estimators to be these possible values (50, 100, and 200), which controls the number of weak learners in the ensemble, with higher values offering better performance but at the cost of computation time. Secondly, we looked at different learning rate values such as 0.5,1, and 1.5. The learning rate determines how much each weak learner contributes to the final prediction. Thirdly, we compared two boosting algorithms: SAMME and SAMME.R. The most optimal parameters were found to be n_estimators = 200, learning_rate = 1, and the SAMME.R algorithm.



Receiver Operating Characteristic (ROC) Curve

Fig. 5 The Confusion Matrix for AdaBoosted decision stump

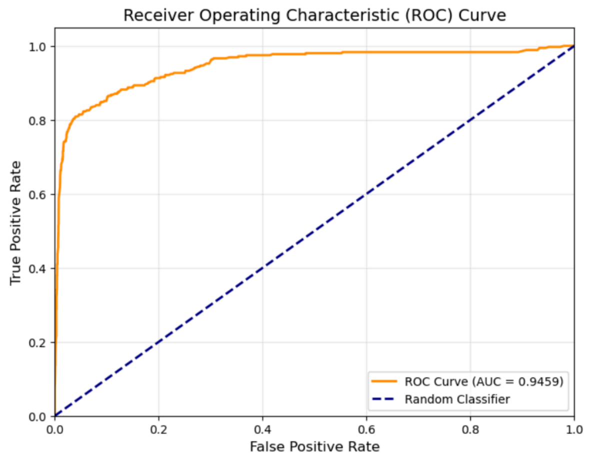


Fig. 6 The ROC curve for the AdaBoosted decision stump

## V. Conclusion

After we trained and evaluated multiple machine learning algorithms, we were able to pick the best model for classifying credit fraud. We were able to find that an AdaBoosted decision stump algorithm is the most accurate way to classify credit fraud from the samples that we had. We also went through and tried many different models that were not as good as logistic regression or were not as good as the decision stump. We tried using a random forest model, a random forest with 5-fold cross-validation, a random forest with grid-search, an AdaBoosted decision stump with grid-search, a random forest with bagging classification, and a random forest with bagging and a grid-search. None of these different algorithms and data processing techniques were able to beat AdaBoosted decision stumps, but some were able to get similar accuracy scores with worse percentages of false positives and/or false negatives. With this model, we had true negatives 98% of the time, while having true positives 73% of the time. With accuracies being this high, we have a lot of confidence in our model to correctly classify fraud in unseen data.

## VI. Distribution of Work

Sage cleaned the data to prepare it for logistic regression and classification and formatted the paper to the IEEE Conference template[4]. He also created the GitHub repository to allow synchronous viewing and editing of code and resources. Additionally, Sage organized the tasks and delegated them to the other group members, and assisted in writing and editing the paper. He also wrote the code for the new models and their cross-validations, and edited the midterm report to meet the specifications of the feedback for the final paper. Finally, he added around half of the new material for the final paper and proofread some of the other groups' additions.

Hayden wrote the majority of the logistic regression classification code.

Kale wrote the code for visualizing the data and the corresponding section on data visualization.

Dylan worked on the classification code and feature engineering, as well as helping write and proofread the paper.

## References

[1] (2025) "Why banks are using advanced analytics and faster fraud detection", Biztech Magazine. [Online]. Available: https://biztechmagazine.com/article/2023/07/why-banks-are-using-advanced-analytics-faster-fraud-detection

[2] (2025) "New FICO Survey: Americans Value Financial Fraud Prevention More Than Banking Customer Experience" Available: https://www.businesswire.com/news/home/20221206005199/en/New-FICO-Survey-Americans-Value-Financial-Fraud-Prevention-More-Than-Banking-Customer-Experience

[3] (2025) Kaggle dataset of credit card fraud and predictors. [Online]. Available: https://www.kaggle.com/datasets/neharoychoudhury/credit-card-fraud-data/data

[4] (2025) The IEEE website on Manuscript Templates. [Online]. Available: https://www.ieee.org/conferences/publishing/templates.html