# Machine Learning Applications to Combat Credit Card Fraud: An Analysis by Regularly Scheduled Programming

Will Greenwood
University of Tennessee, Knoxville
wgreenwo@vols.utk.edu

Maddie Gross
University of Tennessee, Knoxville
mgross10@vols.utk.edu

Cinzia Pacione
University of Tennessee, Knoxville
cpacione@vols.utk.edu

Ethan Weathers
University of Tennessee, Knoxville
dweathe5@vols.utk.edu

## I. Abstract

*Credit card fraud is an increasing threat to financial security, requiring robust methods to detect and prevent fraudulent transactions. In this study, we analyze a comprehensive dataset containing various transaction attributes, including time, location, category, and monetary value, to identify patterns associated with fraudulent behavior. We perform extensive data cleaning and exploratory data analysis (EDA) to visualize trends in fraudulent transactions across different features. Our analysis highlights key fraud indicators, such as transaction time, merchant category, and age demographics of fraud victims.*

## II. Introduction

One of the most important parts about having financial peace of mind is to make sure your finances are safe. Unfortunately, credit card fraud is a major threat to that financial security, and is happening more and more frequently by the day. Our goal was to look into cases of credit card fraud, using predictable trends and collections of fraudulent data to try and come up with solutions to try and reduce the amount of potential credit card fraud cases.

## III. Technical Approach

Our study analyzes the credit card fraud dataset using supervised classification models. Specifically, we evaluate and compare the performance of logistic regression and a baseline random forest implementation by Robin Rawat on Kaggle. Logistic regression was chosen for its simplicity and interpretability, while random forest was selected for its robustness and strong performance on imbalanced, nonlinear datasets. Model evaluation relied on classification reports, confusion matrices, and feature importance graphs.

We used NumPy and Pandas for general data processing. For visualizations, we utilized the Matplotlib and Seaborn libraries. The folium library was used to visualize location data. The Scikit-learn (Sklearn) library was extensively used for machine learning tasks and analysis. In particular, we used StandardScaler and LabelEncoder for preprocessing, and LogisticRegression and RandomForestClassifier for model implementation. We applied GridSearchCV to tune hyperparameters for Rawat's random forest model, primarily including the number of estimators (n_estimators), maximum depth (max_depth), and minimum samples per split (min_samples_split). Finally, evaluation metrics such as classification reports and confusion matrices were generated using Sklearn's built-in functions.

## IV. Dataset Description

The dataset we used for this project has a plethora of different categories for us to attempt to pull data from. For every transaction, it allows us to track the following for every purchase recorded in the dataset:

- Transaction date and time
- Merchant name
- Category of merchant
- Transaction amount
- Transaction number
- City of the credit card holder
- Population of credit card holder's city
- State of the credit card holder
- Latitude/Longitude of the credit card holder
- Occupation of credit card holder
- Credit card holder date of birth
- Was the transaction fraud? (Y/N)

### Data Cleaning and Feature Engineering

Using this extensive dataset, we first evaluated the best way to clean and tailor the data to our needs. We began by checking for null values, and to our pleasant surprise, found none. There were also no inconsistencies or duplicates that required removal. Scaling and normalization were performed

later during the machine learning implementation phase using appropriate tools from Scikit-learn.

Once basic cleaning was complete, we moved on to feature engineering. We started by converting the trans_date_trans_time column into a Pandas DateTime object for easier manipulation. From this, we derived several new time features:

- day_of_week – the day of the week the transaction occurred
- month – the month of the transaction
- hour – the hour the transaction occurred
- is_night – a boolean indicating whether the transaction occurred at night (between 10 PM and 6 AM)

We also created several additional features:

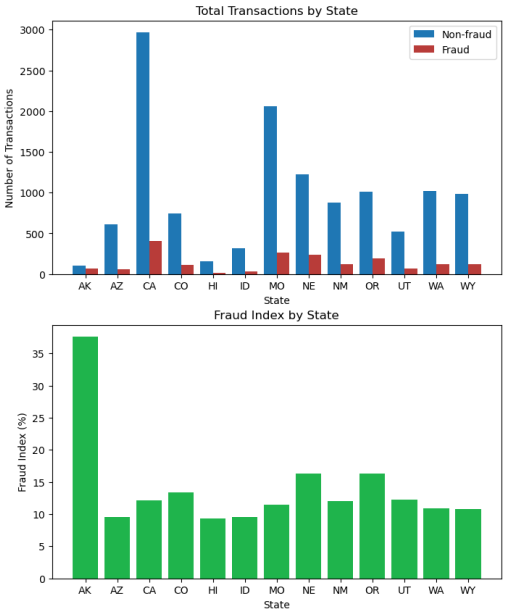
- dob – the date of birth of the cardholder, parsed for age calculation
- age – the age of the cardholder at the time of the transaction
- distance – the geographic distance between the cardholder's location and the merchant's location, calculated using the Haversine formula
- same_city – a boolean indicating whether the transaction occurred in the same city as the merchant

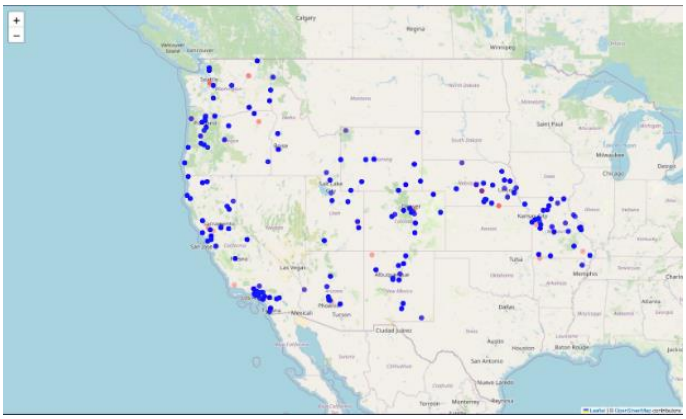
These engineered features provided additional context that helped improve model performance and interpretability.

## Exploratory Data Analysis

Once we felt that we had tuned and cleaned our dataset as needed, we opted to make different kinds of charts, graphs, and other visual aids to help us easily digest some of the highlights of the dataset. Below are some of the data analysis graphics we created:

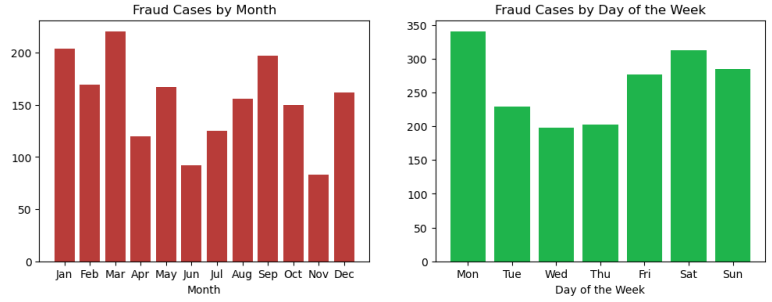
### Location Data



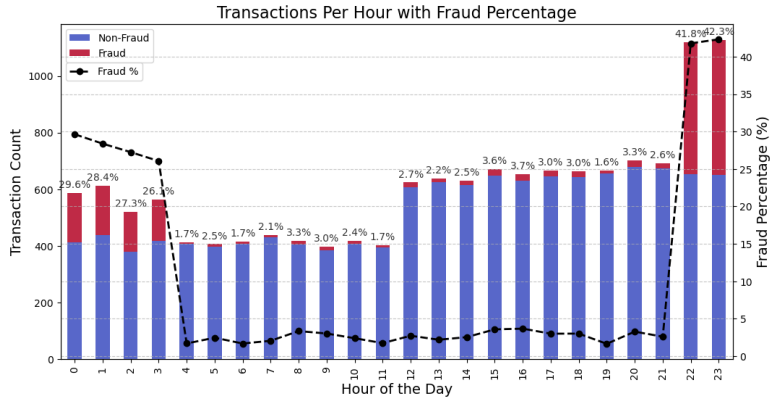



These 2 figures demonstrate the relationship between potentially fraudulent cases and their geographical location in the western United States.

### Time Data



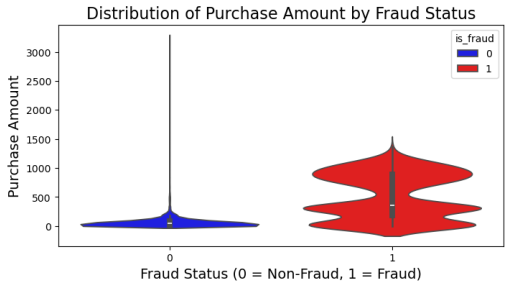

*[4] GeeksforGeeks was used for assistance in creating this histogram.*

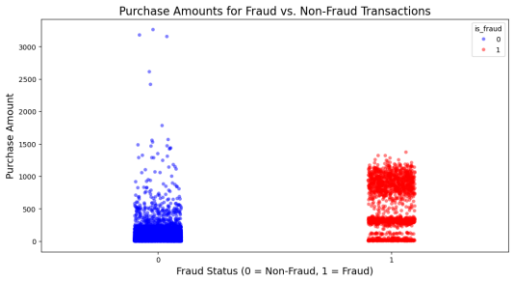



[3] ChatGPT: to make the two fraud categories stack on each other

These 2 figures simply demonstrate how many fraud cases occur with our time-based data from our samples in the dataset. It shows that during the evening, coming off the heels of a weekend, and during the first quarter of the year, the chance of fraudulent data might be higher.

Monetary Data


These binary graphs display how much the different recorded transactions cost. With fraudulent ones being on the low end of around <$1500.

These were some, but not all of the fiures in our exploratory data analysis. Please consult the Jupyter Notebook for the other data figures as well as additional insights into the dataset.

Another odd thing we noticed was that, with this particular dataset, less than 1% of all transactions are considered fraudulent. This is a very interesting thing to note, and will change the way we measure the success of our model moving forward.

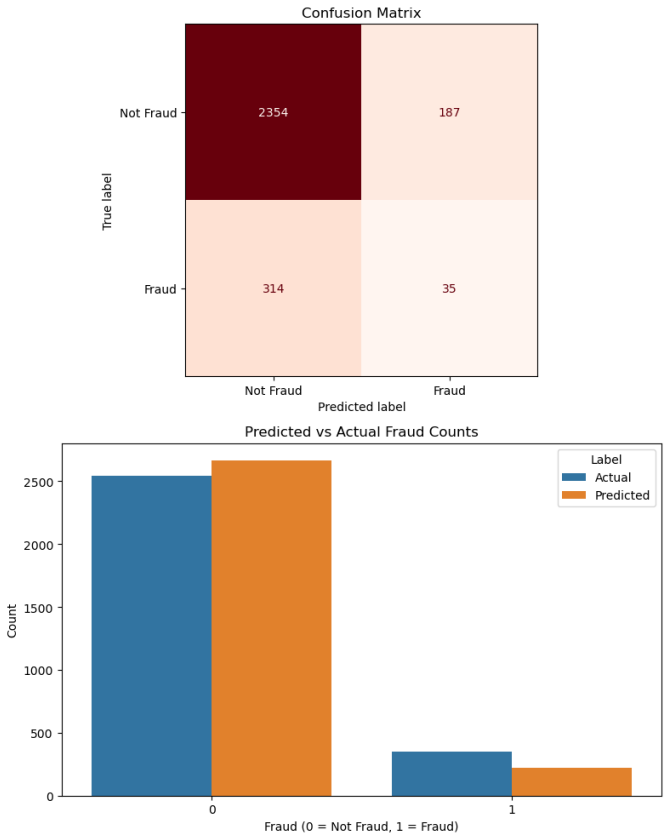
## V. Results and Discussion

Initially, we wanted to start with logistic regression, an extremely interpretable classifier. We ended up using the following features:

- Same_city
- Day_of_week
- Distance
- City_pop
- Age
- Is_night
- Hour
- Month
- amt

Before training the model, we used Scikit-Learn's 'StandardScalar()' to scale the data to our liking.

The model achieved 93% accuracy, with strong precision (93%) and recall (99%) for non-fraudulent cases. However, fraud recall is only 52%, meaning the model misses nearly half of all fraudulent transactions. This is a common issue when working with imbalanced datasets—the model learns to favor the majority class (non-fraud). Because logistic regression assumes linear relationships between features and the log-odds of the target, it may struggle to capture complex or nonlinear patterns in fraud behavior. Overall, logistic regression performs well for identifying non-fraud cases, but it struggles to detect fraud cases, which are rare but critically important. These conclusions can be drawn from the features below:

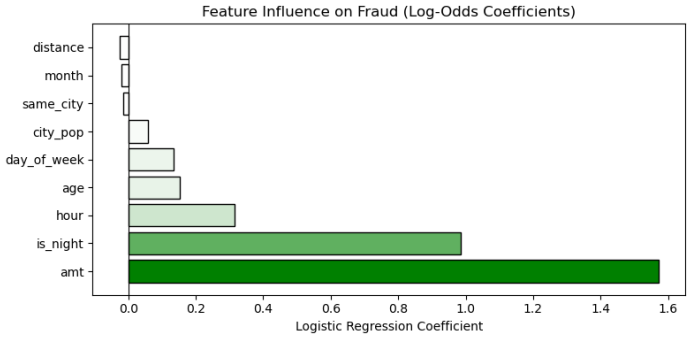



Confusion Matrix (left) and Histogram (right) measuring Actual vs Predicted fraud cases.

A key part of our analysis was to examine which features are more meaningful in predicting fraud. Below is a bar chart of the

log-odds coefficients learned by the model. In logistic regression:

- Positive coefficients increase the likelihood of fraud.
- Negative coefficients decrease the likelihood of fraud.
- Coefficients close to 0 suggest the feature has little influence.




Key takeaways from the plot:

- `amt` (transaction amount) has the strongest positive influence on predicting fraud (~1.57).
- `is_night` (night-time transaction) also has a large positive impact (~0.99).
- `same_city` and `distance` have small but meaningful negative coefficients, slightly reducing the likelihood of fraud.

Based off what we noticed with our initial views of patterns within our dataset, as well as our logistic regression baseline, we looked around at others who have also dipped their toes in experimenting with this data set as well. Upon initial glance, Rawat's Model that we found on Kaggle[1] seemed to fall in line with a lot of our goals and ambitions for this project. We used the same features as the logistic regression model, however, once we started playing around with it, we were pleasantly surprised.

The model achieved exceptionally high accuracy and strong recall. The precision-recall curve for our Random Forest model maintains high precision across a wide range of recall values, indicating that the model is able to identify most fraudulent transactions without flooding the system with false positives.

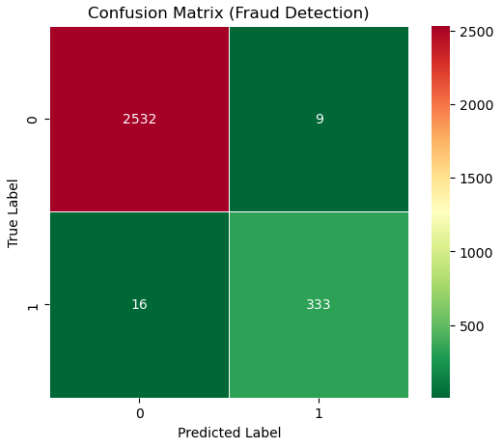
Compared to the earlier logistic regression model, which performed with solid precision and mediocre, the Random Forest classifier outperformed it in nearly every category:

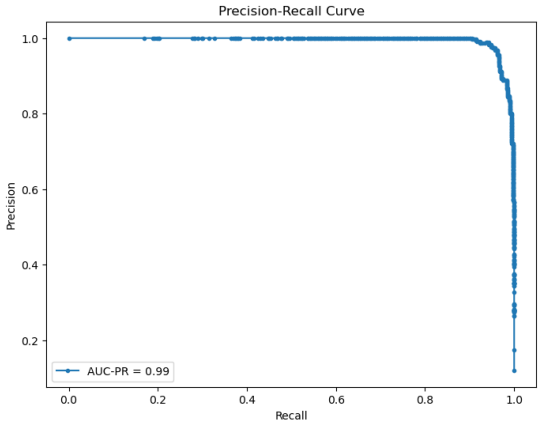
- Random Forest captured more fraudulent transactions (98% vs. logistic regression's 52%).

- Random Forest's accuracy exceeded 99.7%, whereas logistic regression was slightly lower.

- Unlike logistic regression, Random Forest is not limited to linear decision boundaries and can model intricate interactions between features—likely a contributing factor to its superior performance.

The following figures illustrate these conclusions:




The bar chart below shows the relative importance of each feature in the Random Forest model. Higher values indicate that a feature contributed more to the decision-making process when splitting the trees.

Key takeaways and generalization from the plot:

- `amt` (transaction Amount) – *47.7% importance:* The most predictive feature by far.
- `month` – *32.9% importance:* Strongly influential, potentially revealing seasonal fraud patterns.
- `hour` – *8.7% importance:* Time of day appears to somewhat affect fraud likelihood.

Compared to Logistic Regression, both models labled `amt` as the most important feature in classifying fraud, which makes intuitive sense since fraud often involves unusual amounts .The biggest differences are in the `is_night` and `month` features:

- Random forest deems `month` as highly important
- Logistic regression deems `is_night` as highly important

## VI. Conclusion

Logistic Regression achieved moderate performance, capturing general trends but struggling with the more nuanced patterns of fraud. It gave high accuracies and precisions, but a low recall of 52% for cases of fraud. This indicats that while it was cautious about labeling fraud, it missed many true cases.

Random Forest greatly outperformed Logistic Regression, achieving high marks in precision (97% fraud, 99% non-fraud), recall (95% fraud, >99% non-fraud), and accuracy (99%). This demonstrates its strength in detecting fraudulent transactions with fewer false positives, making it a much more effective tool for this task.

Logistic regression treats each feature independently and linearly. `is_night` was deemed an important feature by the logistic regression model. Since this feature consistently correlated with higher fraud, it recieved a strong positive coefficient, even if that relationship was weakly linear.

Random Forest, however, can capture more complex and nonlinear interactions between features. The low importance of `is_night` suggests that it may not have provided much additional value beyond what was already captured by other features or combinations — for instance, `hour` or `day_of_week`. Conversely, the model found strong splits using `month`, implying more subtle or indirect seasonal patterns that Logistic Regression couldn't pick up.

While the Logistic model's conclusions feel more intuitive, Random Forest's flexibility and significantly stronger performance give us more confidence in its assessment of feature significance.

## VII. Distribution of Work

Will

- EDA
    - Total Transactions by State
    - Fraud Index by State
    - Most Common Categories of Fraudulent Transactions
    - Fraud by Age
- Machine Learning
    - Data Cleaning and Feature Engineering
    - Logistic Regression implementation
- Analysis
    - Figure creation for analysis on Logistic Regression and Random Forest (e.g. Confusion Matrix, Feature Importance charts)

Maddie

- EDA
    - Transactions Per Hour with Fraud Percentage
    - Transactions Per Purchase Category with Fraud Percentage
    - Relationship Between Purchase Amount and Fraud
    - Distribution of Purchase Amount by Fraud Status
    - Purchase Amounts for Fraud vs Non-Fraud Transactions
- Analysis
    - Notebook organization
    - Notebook (and what would later be content for the final report analysis) analysis for each model and figure explanations
- Presentation
    - Organized presentation materials

Cinzia

- EDA
    - Fraud Cases United States Map
- Analysis
    - Notebook organization
    - Notebook (and what would later be content for the final report analysis) analysis for each model and figure explanations
- Presentation
    - Organized presentation materials
- Final Report
    - Secondary writer for final report

Ethan

- EDA
  - Fraud Cases by Month
  - Fraud Cases by Day of the Week
- Machine Learning
  - Random Forest
- Analysis
  - Figure creation for analysis on Random Forest (e.g. Confusion Matrix, Feature Importance chart)
- Final Report
  - Primary writer for the final report

## REFERENCES


[1] robinsinghrawat, "Credit_Card_fraud_analysis," *Kaggle.com*, Aug. 23, 2024. https://www.kaggle.com/code/robinsinghrawat/credit-card-fraud-analysis (accessed Mar. 30, 2025).

[2] "Python Machine Learning - Confusion Matrix," *www.w3schools.com*. https://www.w3schools.com/python/python_ml_confusion_matrix.asp

[3] OpenAI, "ChatGPT," OpenAI, Mar. 14, 2025. [Online]. Available: https://openai.com/chatgpt

[4] "Matplotlib Histograms," *www.w3schools.com*. https://www.w3schools.com/python/matplotlib_histograms.asp

[5] "Precision-Recall Curve | ML," *GeeksforGeeks*, Jul. 19, 2019. https://www.geeksforgeeks.org/precision-recall-curve-ml/