Credit Card Fraud Detection

Using metadata to detect fraudulent transactions

Brendan McDonald
McDonaBP@colorado.edu

Kevin Corboy
Kevin.Corboy@colorado.edu

ABSTRACT

Credit card fraud costs consumers and banks money every single day despite multiple systems in place to attempt to stop it. Rates of online purchases compared to purchases made in person are now at all time highs, and quick and reliable fraud detection has become paramount.

Using basic information about the transaction, customer, and merchant, we explored the data to learn more about underlying trends, and developed additional features to flag transactions with a higher probability of fraudulent activity. We achieved this using Python to mine, manipulate, visualize, and summarize our dataset of ~1.3M transactions.

Using a combined set of raw and feature engineered data points as its training baseline, our model was able to predict fraudulent transactions with up to 95% accuracy using a Random Forest approach.

PROBLEM STATEMENT/MOTIVATION

The credit card industry is responsible for processing billions of dollars in transactions on a daily basis, which makes timely and reliable detection of fraud paramount. An entire industry has sprung up to support the banking and credit card transaction markets in fraud detection, using all variants of regression analysis, decision trees, neural networks, and all types of AI/ML analysis to describe and make decisions using data.

- Non cash payments increased at a rate of 9.5 percent per year since 2018 with payments value reaching $128.51 trillion in 2021 [1]
- The value of card payments rose 10% annually since 2018 with card payments value reaching $9.43 trillion in 2021 [1]
- In person card fraud decreased due to adoption of embedded chips in cards [2]
- Remote fraud increased, accounting for $4.57 billion in 2016 and growing since then [2]
- In 2016, the fraud rate for credit cards alone was 14.9 “basis points” meaning for every $10,000 spent, $1.49 was fraudulent [2]

The primary motivation of this work is to determine if we can predict fraudulent transactions before they are executed using a limited number of data points available about the customer. By reducing the frequency of fraudulent transactions, costs to administer bank accounts and facilitate transactions are reduced, benefitting both corporation and consumer. Additionally, merchants can use this analysis to determine areas of high risk that impact their business, allowing them to make changes to their operations to reduce the probability of fraudulent customer activity.

SURVEY OF PRIOR LITERATURE

This dataset was found on Kaggle and there are a number of analysis notebook submissions from different developers available. Since the authors are unknown and the quality is variable, we are not referencing these analyses in our work. We have provided a review of some of the most popular Kaggle analyses that used this dataset. We also reviewed two additional studies related to credit card fraud analysis that rely on alternative datasets.

Kaggle Analysis With This Dataset:

Mohamed Alabasy

- Alabasy's work is primarily a descriptive analysis of the data
- They remove many of data categories prior to checking if they are significant predictors of fraud, including order and merchant location data, transaction date and time, and other potentially significant descriptors
- Their logistic regression results are not highly predictive of fraudulent transactions
- https://www.kaggle.com/code/mohamedalabasy/credit-card-transactions

Omar Wagih

- Wagih's work is a classification model using three separate techniques;
  - Logistic regression
  - K-nearest neighbors (KNN) classification
  - Decision tree classification
- They retain most of the data in preprocessing, only dropping a few fields that are incomplete and not required for the analysis
- Like the Alabasy analysis, the performance of the classification models is highly biased towards non-fraudulent transactions, and did not have a single case of accurately detecting true fraudulent transactions
- :https://www.kaggle.com/code/owm4096/credit-card-fraud-eda-w-ml

Shivam Chaurasia

- Chaurasia's work is similar to Alabasy's in that they remove a large number of fields prior to investigating for correlation
- Like the Alabasy analysis, the performance of the classification models is highly biased towards non-fraudulent transactions, but did have some success in detecting fraudulent transactions using random forest classification techniques
- The geospatial aspect of Chaurasia's analysis was interesting, and we intend to review this data in a different way to see if there is a variability in fraud based on distance
- https://www.kaggle.com/code/shivjichaurasia/credit-card-fraud-detection-eda-prediction

Other Studies on Card Fraud:

"Enhancing Credit Card Fraud Detection: An Ensemble Machine Learning Approach"

- This paper proposes an ensemble machine learning model that combines various algorithms (SVM, Random Forest, KNN, etc.) to tackle challenges w/ data imbalance and concept drift in fraud detection. This model significantly improved accuracy, recall, and F1-scores when compared to traditional machine learning approaches
- https://www.mdpi.com/2504-2289/8/1/6

"A systematic review of literature on credit card cyber fraud detection using machine and deep learning"

- This paper aligns well on credit card fraud detection, with an emphasis on the importance of accurate and timely fraud detection within the credit card industry. It highlights how the industry heavily relies on advanced techniques such as regression analysis, decision trees, neural networks, and various AI/ML methods to analyze data and improve fraud detection systems. The paper's focus on machine learning approaches could provide valuable insights for our project, as it explores the methods and models already in use for detecting fraud.
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10280638/

DATA SET

The dataset includes ~1.3M rows and 24 columns of detailed credit card transaction data. Each transaction

includes purchase information about time and value, customer and merchant details (age, location, market segment, demographic data), and a binary value indicating whether the transaction is fraudulent or not. The dataset spans ~18 months, which also allows for analysis of seasonality and year-over-year spending trends.

The uncompressed CSV file is 337 MB, which is larger than the Github upload limit, but easy to ingest in most other environments. The CU-furnished CSEL environment accepts the uncompressed CSV as an upload, or you can pull the data from the Kaggle link below. [3]

Within the file there are three primary data types represented: integer, floating point number, and string objects.

Integers: Transaction number, credit card number (hashed for privacy reasons), zip code, city population, Unix time, and a binary value for whether or not the transaction is fraudulent.

Floating point numbers: transaction amount, customer and merchant lat/long, merchant zip code

String objects: merchant name and category, customer address, DOB, job, a letter to identify gender (M or F), and a unique hexadecimal transaction identifier.

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	long	city_pop	job	dob
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	...	-81.1781	3495	Psychologist, counselling	1988-03-09
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	...	-118.2105	149	Special educational needs teacher	1978-06-21
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	...	-112.2620	4154	Nature conservation officer	1962-01-19
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	...	-112.1138	1939	Patent attorney	1967-01-12
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	...	-79.4629	99	Dance movement psychotherapist	1986-03-28

Phase 1 Work Completed

1. Loaded data into a suitable coding environment:
    a. Ingest of the dataset is as simple as reading the raw CSV file into a Pandas dataframe. The raw CSV file is too large to upload to Gitbhub in a single file, but theuniversity furnished CSEL coding environment is well suited to a medium sized dataset like this one.
    b. For more compute-intensive tasks like training and large extract/transform/load operations, local execution was managed using Anaconda to launch a Jupyter Lab environment, or Visual Studio
2. Summarized data and reviewed each column for data type, value range, and completeness
    a. 24 columns of data; merchant zip code is missing ~195K observations, otherwise all 1,296,675 rows of data are non-null in all columns (~99.6% “complete” data)
    b. Column data types are a mix of integer, string, and float values
    c. When read into a Pandas dataframe, the dataset occupies ~1.1GB of memory, which will serve as our starting point for data reduction measurement
    d. All columns are complete with no missing data except for merchant zip code, which was missing ~200K entries; this was not an issue since we have different location data that is complete for all merchants

Phase 2 Work Completed

3. Explored the data to review differences between fraudulent transactions and non-fraudulent transaction sets:
    a. Transaction time (month, week, day, and hour)
    b. Transaction value/amount
    c. Transaction distance between customer and merchant
    d. Merchant category
    e. Geospatial distribution and hotspot analysis
4. Added feature flags based on our findings above:

    a. Transactions between 10PM and 12AM were flagged as high risk; 12AM and 4AM medium risk, else low risk for fraud
    b. Transaction values between $250 and $400, $650 and $1200 were flagged as high risk
    c. Transactions within grocery point of sale, online shopping, shopping point of sale, gas/transport, and online misc were more likely to be fraudulent than any other category, and were marked accordingly
    d. Transactions over long distances had a slightly higher fraud rate than those over short distances, and were marked as non-local and appended with a distance between customer and merchant in kilometers
5. Reviewed correlations between our transactions flagged as fraudulent and the various input data and engineered features
    a. As expected from EDA, the strongest correlation was the time of day risk flag
    b. Additional weak correlations were observed, but this was particularly helpful for the purposes of pruning our dataset
6. Cleaned data to reduce memory requirements and computational overhead once training was initiated
    a. Determined that merchant zip code could ultimately be removed; merchant latitude and longitude are more useful for quick distance estimates without the need to perform an address geocode lookup
    b. Removed hashed credit card number data since it was encrypted and not useful for our analysis
    c. Removed hexadecimal transaction identifier in lieu of incrementing integer value, which requires fewer bits to represent based on the small size of our dataset
    d. Removed customer address information in lieu of customer latitude and longitude values that are easier to calculate distances
    e. Our final dataframe size was ~200MB, which represents an 82 percent reduction in memory usage
7. Prepare for analysis: A hefty effort was needed to clean and prepare the data for analysis. This included (but is not limited to) the following steps:

a. Encoding Categorical Variables: Fields such as merchant category, transaction type, and risk flags were label encoded for use in machine learning models. Transaction timestamps were transformed into actionable features such as transaction hour, day, month, and day of the week. Customer age at the time of the transaction was calculated by combining date of birth (dob) and transaction date and time (trans_date_trans_time).
b. Ex: Credit card numbers:

Since credit card numbers are random and issued by the bank/merchant, it is unlikely that the number itself is useful in determining fraud. We did a quick check to confirm that all the card numbers were indeed valid card numbers based on the Luhn algorithm, but as expected, there were no errors found. Removing the credit card number column saved ~10MB of memory.

We determined that despite passing the Luhn algorithm check, the credit card numbers were hashed and randomized Ended up not needing to do a Luhn check at all - the card numbers were all anonymized in the dataset ahead of time, so they were not useful and could be removed

c. Transaction Num:

Since the transaction Num column is a unique hexadecimal string, it is unlikely that the string value is useful for fraud detection. The primary use is to serve as a unique identifier for each transaction. However, using a hexadecimal string is unnecessary when we can instead use an integer value to identify each transaction. Additionally, our dataset already contains an incrementing integer value for each transaction (the column originally titled "Unnamed: 0" in the raw data). Removing the transaction number column frees up ~110 MB of memory.

Other columns like customer first, last, and street/address data were removed as they don't add anything unique/interesting that the other latitude longitude data doesn't provide. Removal of these columns resulted in a memory savings of ~254 MB.

d. Data reconciliation/completion

Merchant zip code: this is the only column that is not completely populated in the dataset. We will review initial correlations before making a determination on whether to fill the missing values, or remove the column entirely. We have a proof of concept to perform a geocode look-up of the missing zip code data using the latitude and longitude data that is available. A zip code can be useful for an approximation of distance between two points, but it is imperfect, and we can get a similar approximation using a great circle distance estimate between both sets of coordinates

Imputed missing values in merch_zipcode using the most frequent value strategy fills ~150k empty entries

We removed all address information for both customer and merchant, and instead opted to use the latitude and longitude data that was available for each to do distance lookups and calculations. Then it was confirmed no other columns had significant missing data.

Memory Optimization:

Removing unnecessary fields and converting data types resulted in a reduced memory footprint of ~50%, from around 1.1 GB to <550 MB.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 22 columns):
 #   Column                          Non-Null Count    Dtype
---  ------                          --------------    -----
 0   Unnamed: 0                      1296675 non-null  int64
 1   trans_date_trans_time           1296675 non-null  datetime64[ns]
 2   amt                             1296675 non-null  float64
 3   lat                             1296675 non-null  float64
 4   long                            1296675 non-null  float64
 5   city_pop                        1296675 non-null  int64
 6   merch_lat                       1296675 non-null  float64
 7   merch_long                      1296675 non-null  float64
 8   is_fraud                        1296675 non-null  int64
 9   transaction_year_month          1296675 non-null  period[M]
 10  transaction_month               1296675 non-null  int32
 11  transaction_weekday             1296675 non-null  int32
 12  hour_of_day                     1296675 non-null  int32
 13  minute_of_hour                  1296675 non-null  int32
 14  distance_km                     1296675 non-null  float64
 15  not_local                       1296675 non-null  int64
 16  transaction_value_risk_encoded  1296675 non-null  int64
 17  transaction_time_risk_encoded   1296675 non-null  int64
 18  category_risk                   1296675 non-null  int64
 19  transaction_year                1296675 non-null  int64
 20  age_at_transaction              1296675 non-null  int64
 21  gender_encoded                  1296675 non-null  int64
dtypes: datetime64[ns](1), float64(6), int32(4), int64(10), period[M](1)
memory usage: 197.9 MB
```

COMPLETE; Initial correlations investigated, training methods selected

4. Feature Engineering:

a. Geospatial Features

The distance between customer and merchant was calculated using geospatial coordinates, and transactions which were then categorized as local (<10 km), regional (<50 km), or long distance (>50 km). As expected, fraud rates increased slightly with distance from home.

b. Risk Indicators

Risk levels were encoded for transaction amounts ($250-$400 and $650-$1200 marked as high risk) and transaction timing (e.g., late-night hours as high risk).

c. Functions created :
    i. Created a feature by calculating the distance between customer and merchant locations/zips + set a threshold (50 km) to flag non local transactions as potentially fraudulent:

#function to add a not_local flag to determine if further distance causes more fraud --ie Fraud while Traveling

```
def calculate_distance(row):

    return np.sqrt((row['lat'] - row['merch_lat'])**2 + (row['long'] - row['merch_long'])**2)

df['not_local'] = df.apply(calculate_distance, axis=1).apply(lambda x: 1 if x > 50 else 0)
```

        a. Determines if fraud happens more frequently when people are away from home (ie traveling or maybe in places that are not typical)

2. Visualized transaction locations using a scatter geoplot for geospatial insights
    a. Random sample used for rendering
    b. Can see if there are hotspots of fraud
3. Data exploration completed:
    a. Reviewed how fraudulent transactions are distributed across the entire 18 month sample period, within the month, week, day, and hour time periods..
    b. Reviewed fraudulent transaction values to determine that fraudulent transactions are clustered in two distinct value ranges
    c. Investigated merchant category data to determine five categories that had higher rates of fraud than others, and flagged transactions in those categories
    d. Calculated distance between customer and merchant, classifying transactions into one of three categories based on their distance:
        i. local = < 10 km
        ii. regional = >10km, < 50 km
        iii. non-local => 50 km

COMPLETE: Model trained and tested

4. As expected in our early proposal phase, data hold-back was an important consideration due to significant class imbalances inherent in our fraud dataset. Ultimately we went with a randomized training and test split set of 80% to 20%, with the goal of including as many data points as possible in our training. In the future, we'd prefer to work with a training set that has a large sample of fraud transactions, or we could consider additional modifications to our existing training:
    a. Class weight adjustment is a method to increase the weights in minority class observations
    b. Evaluation metrics that may work for balanced datasets (like five number summary statistics) often are less useful with imbalanced datasets, so we will observe different metrics like the ROC Curve and F1 scores that can be used to assess datasets with significant size differences in classes
5. Data addition and augmentation is another area that we'd like to explore:
    a. There are some basic customer demographic data points included within the dataset that can be augmented with additional

demographics information that may be useful in flagging transactions.

b. There is also a subset of data that spans more than 12 months, which could prove useful for year-over-year trend analysis that is applicable for fraud detection purposes.

c. Customer segmentation and predictive modeling may support flagging mechanism for future buyer behaviors

Exploratory Data Analysis (EDA)

EDA showed unique patterns in fraudulent transactions compared to regular transactions:

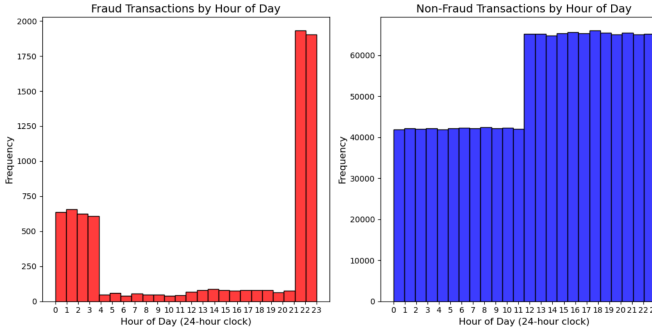
1. Transaction Time:

Fraudulent transactions occurred most frequently between 10 PM and midnight. Risk levels were assigned based on time categories:

High Risk: 10 PM–12 AM.

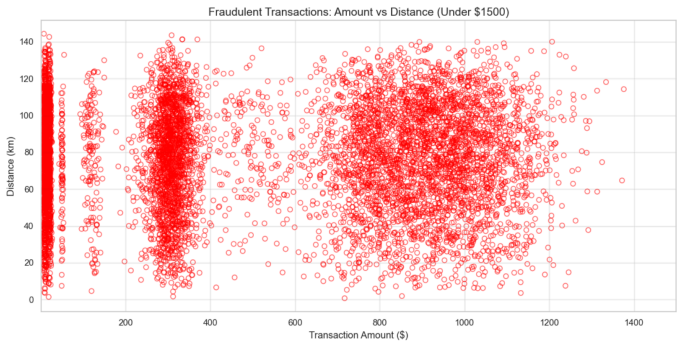
Medium Risk: 12 AM–4 AM.

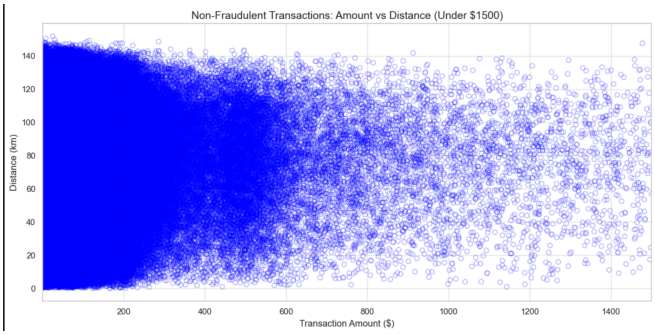
Nominal Risk: All other times



2. Transaction Value:

Fraud was more common in specific price ranges ($250-$400 and $650-$1200). These ranges were flagged as high risk, likely reflecting anomalies in consumer behavior.





3. Transaction Distance:

Fraud rates increased with the distance between the customer and merchant, particularly for long-distance transactions (>50 km). A `not_local` flag was implemented to highlight these cases.

4. Merchant Categories:

Fraudulent transactions were disproportionately concentrated in specific merchant categories, such as online shopping, grocery stores, gas/transport, and shopping point-of-sale.

Feature Engineering

A range of new features were engineered to improve the predictive power of the models:

- Distance Calculation: Using geospatial coordinates
- Temporal Features: Hourly risk encoding
- Categorical Risk Flags: Merchant category flags
- Dynamic Features: Transaction history trends and clustering in high risk periods

These features were validated through correlation analysis, demonstrating significant predictive potential for fraud detection.

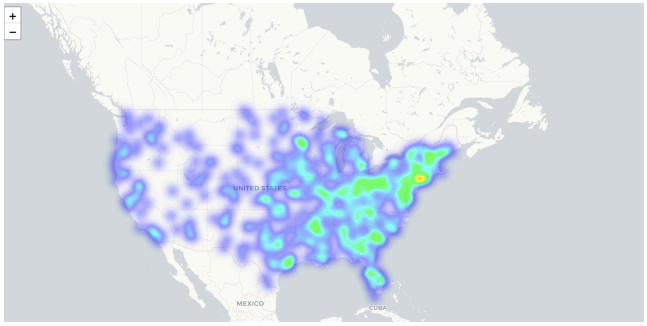
Evaluation Metrics:

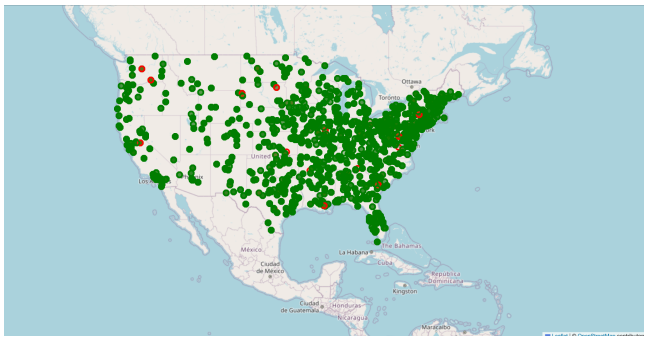
- Accuracy: Demonstrated the ability of the model to classify transactions correctly.
- Precision: High precision indicated that flagged transactions were genuinely fraudulent.
- Recall: High recall ensured minimal false negatives, reducing undetected fraud.
- F1 Score: Balanced precision and recall to reflect overall model performance.
- ROC-AUC: Showed the model's capacity to distinguish between fraudulent and non-fraudulent transactions.

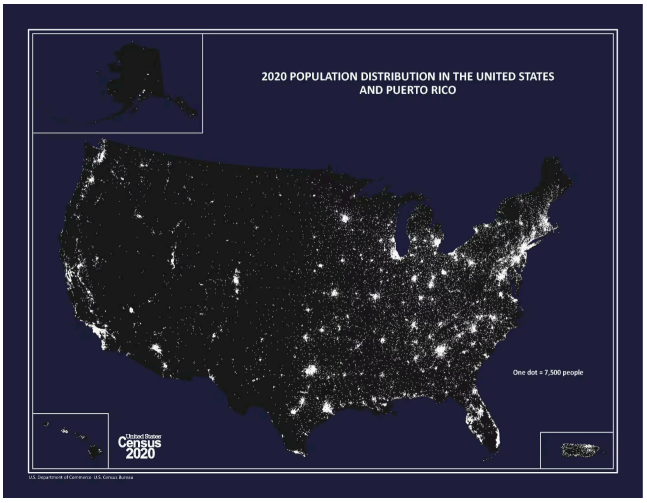
Geospatial and Temporal Analysis

Visualizations provided additional insights:

- Heatmaps: Highlighted fraud hotspots in densely populated areas.
- Temporal Trends: Revealed peak fraud times and seasonal variations in transaction patterns.
- Geospatial Maps: Demonstrated clustering of fraud incidents in specific urban centers.







EVALUATION METHODS

To evaluate our results, we will hold-back a subset of the data prior to performing our analysis. This subset will serve as our sample to conduct testing on once we have our model developed.

If our model is working, it should be able to detect fraudulent transactions using customer data provided at the point of purchase.

We can compare model estimates to the binary flag that tells us whether or not the transaction was fraudulent, and tune our model as needed.

Here are some metrics we intend to evaluate:

1. Accuracy: Evaluate how well (we would need to pick a scale) your model predicts both fraudulent transactions correctly. This could be misleading considering non fraud cases dominate most datasets (avgs above in the federal reserve numbers)
2. Precision: Calculate the ratio of true positives (correct fraud predictions) to all positive predictions (both true and false) – High precision would indicate flagged transactions are *truly* fraudulent i.e.not a false positive
3. Recall (Sensitivity): Calculate the ratio of true positives to **all** fraud cases (true positives + false negatives) – High recall means fewer false negatives.
4. F1 Score: Calculate the mean of precision and recall, providing a balance between these two metrics
5. ROC-AUC (Receiver Operating Characteristic - Area Under Curve): Evaluates a model's performance by plotting true positive rates against false positive rates – A high AUC score means the model distinguishes more easily between fraudulent and non-fraudulent transactions
    a. AUC score closer to 1 = better performance

Several machine learning models were tested, with the following results:

- Random Forest:
    - Performance: Achieved the highest accuracy (95%) and ROC-AUC score (0.9868).
    - Strengths: Effective handling of imbalanced datasets, robust against overfitting, and capable of leveraging the engineered features.

- XGBoost:
    - Performance: Delivered comparable accuracy (91%) and an ROC-AUC score of 0.9723.
    - Strengths: Faster training time and lower computational cost than Random Forest.

- Logistic Regression:
    - Performance: Lower accuracy (86%) and ROC-AUC (0.8310) due to limitations in capturing non-linear relationships.

- Strengths: Simple and interpretable, useful as a baseline model.

```
Random Forest:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99    257186
           1       0.98      0.99      0.99    258482

    accuracy                           0.99    515668
   macro avg       0.99      0.99      0.99    515668
weighted avg       0.99      0.99      0.99    515668

XGBoost:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96    257186
           1       0.96      0.96      0.96    258482

    accuracy                           0.96    515668
   macro avg       0.96      0.96      0.96    515668
weighted avg       0.96      0.96      0.96    515668

Logistic Regression:
              precision    recall  f1-score   support

           0       0.86      0.85      0.85    257186
           1       0.85      0.86      0.86    258482
...
   macro avg       0.85      0.85      0.85    515668
weighted avg       0.85      0.85      0.85    515668

ROC-AUC Scores: RF=0.9982733640831645, XGB=0.9937198556118954, LogReg=0.9307934322181903
```

Key Analysis

- Fraudulent transactions exhibit distinct patterns in terms of time, value, and location.
- Engineered features *significantly* enhanced model performance
- Random Forest emerged as the most effective model for this dataset.

Proposed Future Work

1. Enhanced Feature Engineering

Future engineering will focus on enhancing features by incorporating additional datasets to round out the information studied. Key insights from customer demographics, merchant reliability scores, and historical fraud patterns are all valuable sources that could provide valuable context and improve the predictive power of the models. Dynamic variables, like trends in transaction history, could also be explored to make models more responsive in real time fraud patterns rather than determining fraud after the fact. Additionally, a deeper dive into the trends during holidays and seasonal variations could offer insights into different temporal trends than the ones studied here.

Real time fraud detection will be another important area of development. Optimizing the models for real time prediction would significantly enhance the ability to prevent suspicious transactions before the sale has been made. Implementing active machine learning techniques will let the system improve its performance over time.

Finally, deploying the model into existing fraud detection systems would improve the system dramatically. By integrating insights into real world applications, engineers can proactively design security measures for higher risk merchant categories to further reduce fraud.

TOOLS

Our primary tool for this analysis was Python due to its easy of use and extensive support libraries and packages to complete our analysis:

- Pandas: open source data manipulation and analysis library that makes processing large datasets quick and easy using Python
    - https://pandas.pydata.org/
- Numpy: another opens source data manipulation and analysis library with Python
    - *https://numpy.org/doc/stable/index.html*
- SciKit Learn: *"Simple and efficient tools for predictive data analysis built on the Numpy package"*
    - https://scikit-learn.org/stable/
- MatPlotLib and Seaborn: Both of the libraries are useful for data visualization purposes
    - https://matplotlib.org/
    - https://seaborn.pydata.org/
- We will manage our analysis within an iPython Notebook so that we can easily

integrate markdown text and explain our work as we go

REFERENCES

[1] *Federal Reserve Payments Study (FRPS)* (no date) *Federal Reserve Board - Federal Reserve Payments Study (FRPS)*. Available at: https://www.federalreserve.gov/paymentsystems/fr-payments-study.htm (Accessed: 15 October 2024)

[2] *Changes in U.S. payments fraud from 2012 to 2016*. Available at: https://www.federalreserve.gov/publications/files/changes-in-us-payments-fraud-from-2012-to-2016-20181016.pdf (Accessed: 16 October 2024).

[3] *Credit Card Transactions Dataset*, *Kaggle*. Choksi, P. (2024) Available at: https://www.kaggle.com/datasets/priyamchoksi/credit-card-transactions-dataset?resource=download (Accessed: 15 October 2024).