Fraud Detection
In Credit Card Transactions

Andrew Gelbard, Alex Walsh, Vladimir Kirichenko

Problem Statement

- Number of U.S. adults who have been victims of credit credit fraud: 127 million (Security.org)
- Percentage of U.S. fraud reports that involved financial losses in 2021: 26 percent (FTC)
- Develop a classification model to predict credit card fraud with goal of reducing identity theft and financial losses of U.S. citizens

Our data: we used a 'random generator' to generate a set of credit card data

How our random generator[1] works to create transactions:

1. There are pre-populated sets of Merchants, Customers, Transaction Categories

2. There are pre-populated customer profiles with behavior patterns by demographics (*e.g. females in rural areas age 25-50*)

3. User inputs (1) number of customers and (2) time period, and based on these parameters program generates transaction patterns using a combination of python build-in 'rand' function and the 'Faker' library (e.g. names, GPS coordinates, SSNs)

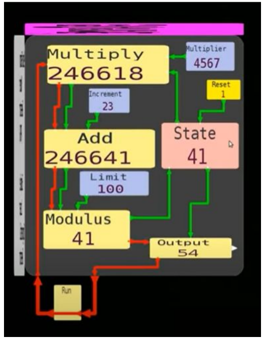
Sample Profile


```
▼ avg_transactions_per_day:
    min: 1
    max: 4
▼ date_wt:
  ▶ day_of_week:
  ▶ time_of_year:
  ▶ year:
▼ categories_wt:
    gas_transport: 225
    grocery_net: 15
    grocery_pos: 150
    misc_net: 100
    misc_pos: 65
    shopping_net: 125
    shopping_pos: 110
    entertainment: 120
    food_dining: 90
    health_fitness: 125
    home: 150
    kids_pets: 150
    personal_care: 120
    travel: 60
▼ categories_amt:
  ▼ gas_transport:
      mean: 60
      stdev: 15
```

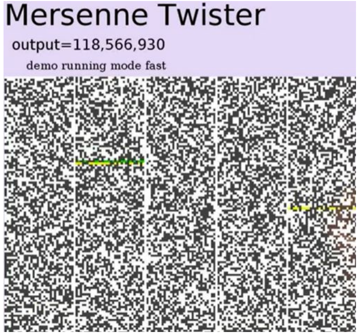

1. https://github.com/namebrandon/Sparkov_Data_Generation

Deep dive: how do random generators work in general?

1 **Pseudo random generator**; essentially a formula that starts off with a seed, applies a function, and then replaces a seed.

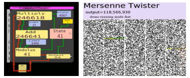


Old linear feedback shift register



Mersenne Twister

2 **True random generator,** which incorporates a pseudo random generator plus sources of entropy from external world

Pseudo random generator 

+

True source of randomness to adjust the seed, such as:

- Keyboard clicks
- Time between clicks
- Mouse movement
- Weather changes

Key Points For Consideration

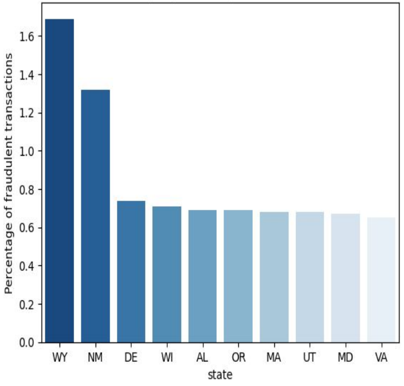
- In your random generation, are the assumptions you make about distributions life-like?[1]
- Are you missing patterns you might in real life in your simulation (hint: you probably are)[2]
- Are you aware of how your random generator works and its potential limitations or lack of reproducibility?

There is no perfect simulation tool, but we can understand the limitations and take them into account

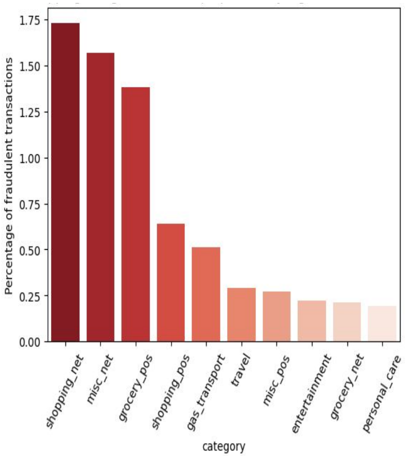
1. E.g. for the fraud detection model; are the standard deviations for key variables accurate? Are GPS coordinates random? Is there missed cyclicality in spending for Holidays? 2. E.g. city-level or zip-code level variation. Credit scores of individuals, etc.; there is more richness of data in real life vs. any human created simulation

Four Key Trends from EDA

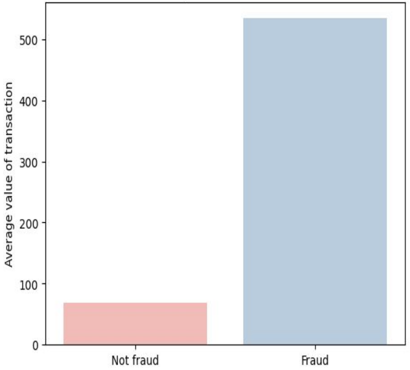
1 WY and NM have higher % of fraudulent transactions



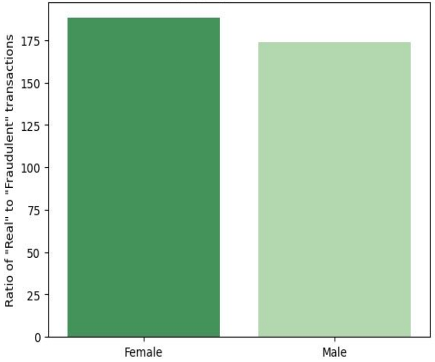
2 Certain categories have higher % fraudulent transact.



3 Fraudulent transactions tend to have higher $ amts.



4 Males have slightly higher ratio of fraudulent transact.



Feature Engineering

- Different consumers have different spending habits
- What can be considered fraudulent for some users may be very standard for others
- Features need to be calculated for each card number individually
- Features follow either normal or von Mises (circular) distributions

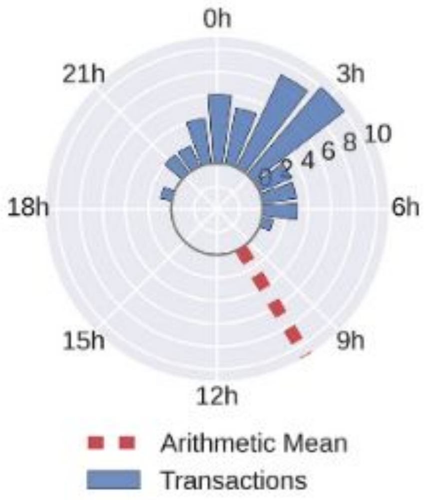


image credit:
https://albahnsen.github.io/files/Feature%20Engineering%20Strategies%20for%20Credit%20Card%20Fraud%20Detection_published.pdf

Rolling Features

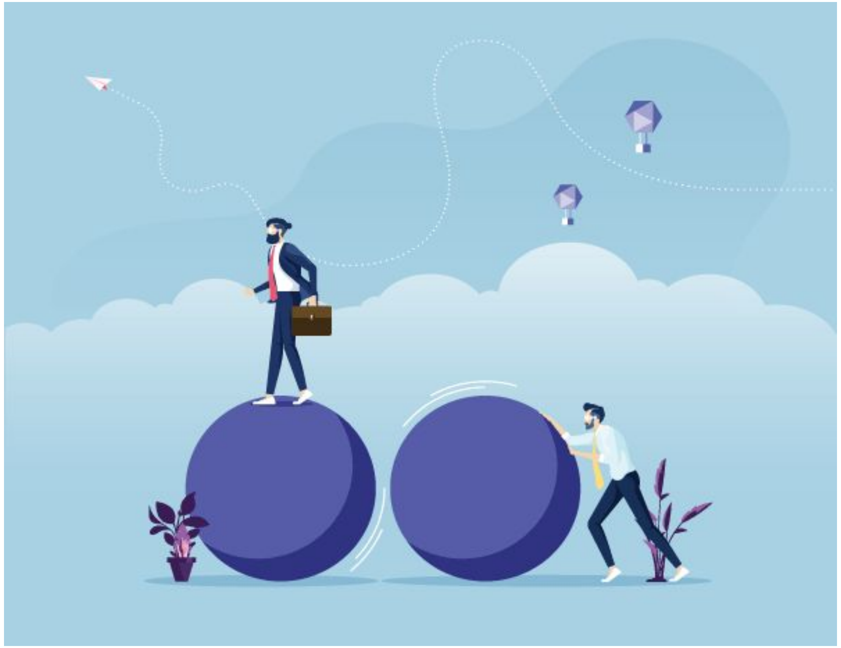
- Credit card users' spending habits are not set in stone, and can change suddenly and drastically

- These changes are not necessarily fraudulent, but they will appear suspicious at first

- Features need to be designed so that spending habits prior to such changes will be "forgotten" over time

Rolling Features (continued)

- Features can be calculated over a period of time or a number of transactions
- Multiple different calculation window sizes are used to reflect short and long term changes

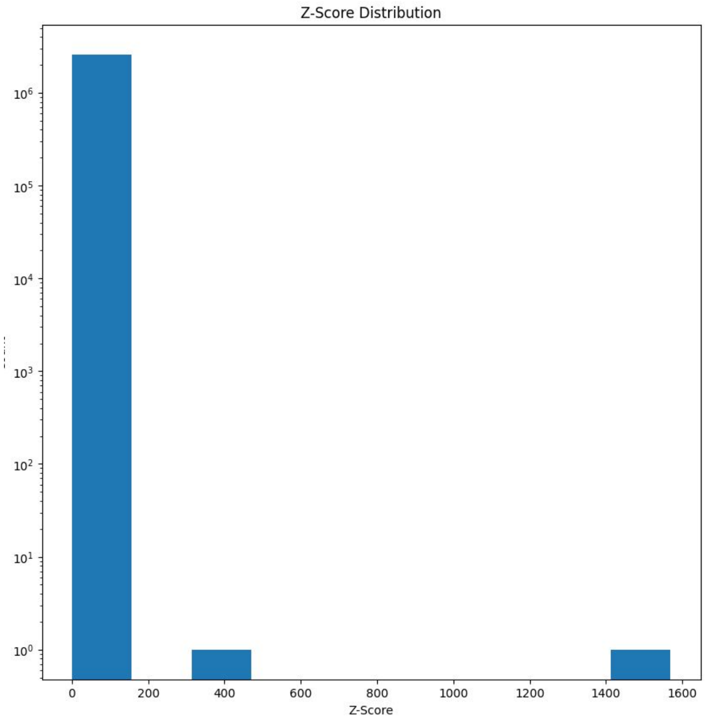


Statistical Anomaly Detection

- A common practice to identify anomalous points is to calculate the probability of eat point and flag the examples with the lowest probability

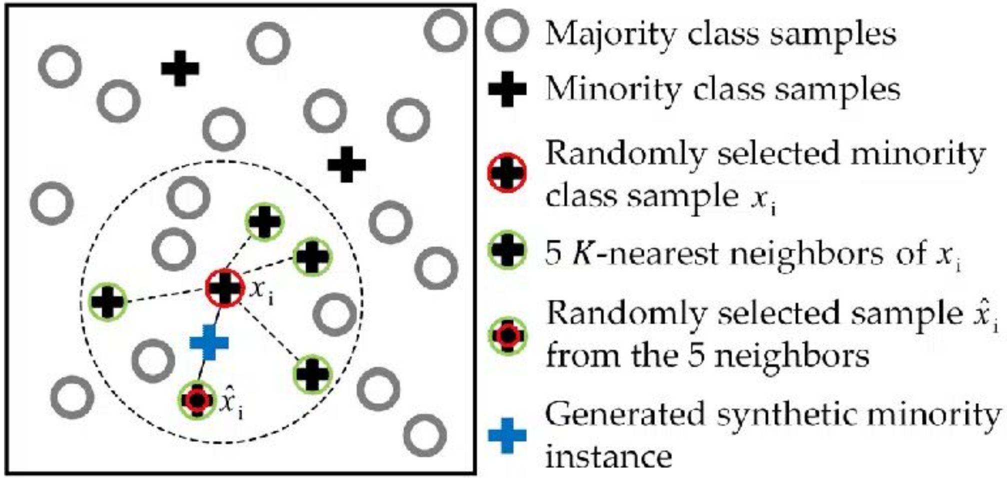
- Most likely due to the large number of features created, the dataset did not yield a usable distribution for this

- The features were created with supervised learning in mind, so while concerning this did not prove to be problematic



Challenges of Imbalanced Classes

SMOTE - Synthetic Minority Oversampling Technique

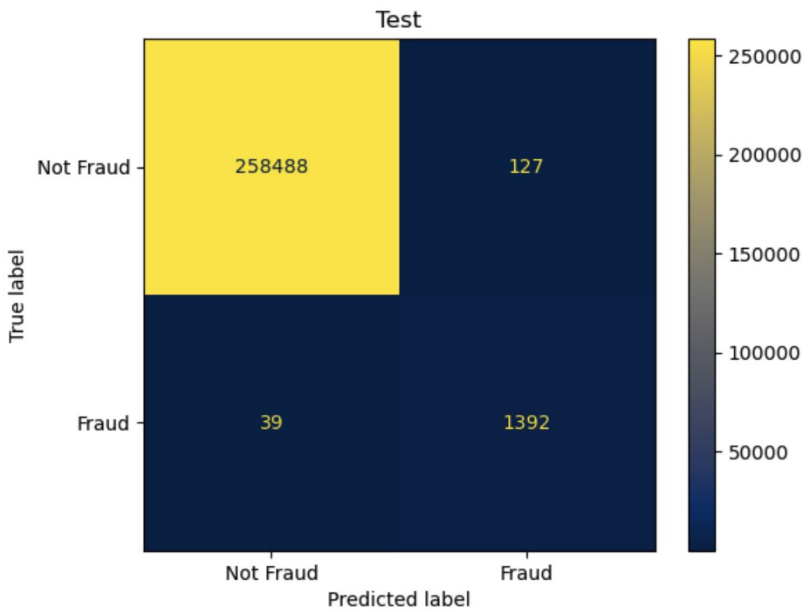


- 99.5 percent of the data is not fraud.
- Accuracy is not a great measure of model performance
- Precision and Recall are more informative

image source: https://rikunert.com/smote_explained

Final Model Results

Random Forest Confusion matrix



- Random Forest, AdaBoost, Logistic Regression
- implemented SMOTE to address imbalanced classes
- 0.97 recall, 0.92 precision, 0.94 f1-score on test data fraud predictions
- customers should expect only 1 in every 2000 transactions to labeled fraud that is actually not fraud

Summary and Recommendations

Important Features

- Multiple different calculation window sizes are used to reflect short and long term changes of spending habits

Next Steps

- Expand modeling to test more hyperparameters and classification algorithms
- Figure out flaw in probability distribution

Considerations

- Social considerations of detecting fraud
- Applications and limitations of the model

Questions?