Second Project Report

## Project Title

Real-time Anomaly Detection in Financial Transactions

## Authors and Team

* Author 1: Haozhen Guo
* Author 2: Yang Liu

Team name: Black Thym

## Executive Summary

### Decisions to be impacted

Based on our research, anomaly detection algorithms play a crucial role in shaping business decisions across three key areas:

* Personal Savings Protection: By identifying abnormal transactions, banks and other financial institutions can proactively monitor and block suspicious or high-risk activities, thereby safeguarding individual savings and account security.
* Risk Management: Anomaly detection enables financial institutions to recognize unusual patterns within portfolio management, allowing for timely adjustments to investment strategies, and ultimately enhancing risk management and financial performance.
* Anti-Money Laundering (AML): A major application of anomaly detection is in identifying irregular transaction behaviors that may indicate potential money laundering activities, helping institutions comply with regulations and prevent financial crime.

### Business Value

* Enhanced Security and Customer Trust
* Cost Reduction through Automated Monitoring
* Scalable Risk Management Solutions
* Proactive Fraud Prevention for Industry Growth

### Data Assets

Our fraud detection dataset was collected from Kaggle and provided from Vesta Corporation, a leader in e-commerce payment solutions. The dataset was split into two files: “train\_identity.csv” and “train\_transaction.csv”, both of which can be joined through the common and unique key TransactionID. Our goal is to establish machine learning methods to identify fraudulent transactions, which was labeled as isFraud in “train\_transaction.csv”, using a wide range of features. Here are some detailed information about each dataset:

“train\_transaction.csv” : Contains the majority of data related to transactions.

* TransactionDT: timedelta from a given reference datetime (not an actual timestamp)
* TransactionAMT: transaction payment amount in USD
* ProductCD: product code, the product for each transaction
* card1 - card6: payment card information, such as card type, card category, issue bank, country, etc.
* addr: address
* dist: distance
* P\_ and (R\_\_) emaildomain: purchaser and recipient email domain
* C1-C14: counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked.
* D1-D15: timedelta, such as days between previous transaction, etc.
* M1-M9: match, such as names on card and address, etc.
* Vxxx: Vesta engineered rich features, including ranking, counting, and other entity relations.

“Train\_identity.csv”: Contains the identity information variables associated with transactions, such as network connection information (IP, ISP, etc) and digital signature (UA/os, etc). However, the field names were masked and pairwise dictionaries would not be provided for privacy protection and contract agreement.

For this project, we will benchmark machine learning models on a large-scale dataset about real-world e-commerce financial transactions. We wish to build up a better method to detect the fraud in transactions, and further achieve saving personal assets with higher accuracy and efficiency.

Addison Howard, Bernadette Bouchon-Meunier, IEEE CIS, inversion, John Lei, Lynn@Vesta, Marcus2010, and Prof. Hussein Abbass. IEEE-CIS Fraud Detection. https://kaggle.com/competitions/ieee-fraud-detection, 2019. Kaggle.

## Data Preprocessing

### Correlation with the target column

Before dropping columns with a high percentage of NaN or dominated values, we first calculate the correlation between each column and the target column (“isFraud”). This helps us identify features that may strongly correlate with the target when a transaction is fraudulent. After performing the calculations, the highest absolute correlation value is 0.396, indicating that no feature has a strong correlation with the target column.

### Drop Nan and dominated features

We use two methods to analyze the distribution of NaN values in our dataset. First, we create a table that records each feature's data type, percentage of null values, and the number of unique values, which helps us evaluate the characteristics of each feature. Next, we calculate the proportion of NaN values to determine an appropriate threshold for handling missing data. The following table shows the proportion of NaN values:

| **NaN Percentage Interval** | **Number of Features** |
| --- | --- |
| [0.00, 0.05] | 244 |
| (0.05, 0.10] | 2 |
| (0.10, 0.15] | 1 |
| (0.15, 0.40] | 0 |
| (0.40, 0.45] | 50 |
| (0.45, 0.50] | 8 |
| (0.50, 0.55] | 70 |
| (0.55, 0.65] | 23 |
| (0.65, 0.70] | 1 |
| (0.70, 0.75] | 2 |
| (0.75, 0.95] | 3 |
| (0.95, 1.00] | 30 |
| Total | 434 |

### From the table, it is evident that when the NaN percentage is in the range [0, 0.15], we retain half of the features, which is sufficient for model training. Additionally, there are no features with NaN percentages between 0.15 and 0.40. A 15% threshold for NaN values is significantly more manageable compared to 40%, making it a preferable choice for data retention.

### Filling Nan

#### Correlation heatmap of Features with entire values

#### Simply filling

#### KNN filling

### Outlier detections

#### IQR detection

#### Z-score detection

#### DBScan/ Isolation

## Model Updates

### Models in Planning

### Machine Learning WorkFLow as MLM

## Source Code

// when finished the code, paste the github link.

## Next Steps

### Plan for Improvement

### Timeline of Next Steps