# Project Description

**By: Maverick Wong Ke Jun (16365445)**

## 1. Project Overview

In this project, we will be answering the research question “Which machine learning model provides the best approach to detect phishing activity on the Ethereum network?”

## 2. Dataset Used

For this project, the dataset used consists of Ethereum transaction records which can be downloaded from Kaggle

1. Kaggle Dataset 1
2. Kaggle Dataset 2

These Kaggle datasets were chosen as they contained columns with key fraud detection features like ‘address’, ‘FLAG’ and transaction values.

## 3. Exploratory Data Analysis

In this segment, I will discuss the idea behind the retained features, data cleaning and feature engineering process.

### 3.1 Retained Features

As the two datasets contained a different subset of features, I had to select features present in both datasets in order to merge them. Firstly, I inspected the individual dataframes to inform me on the empirical value of features to our goals and thus which to retain for model training. To achieve this, we will be looking at the characteristics of data that are labelled as fraudulent and identify if there is significant heterogeneity to the rest of the dataset.

### 3.2 Data Cleaning

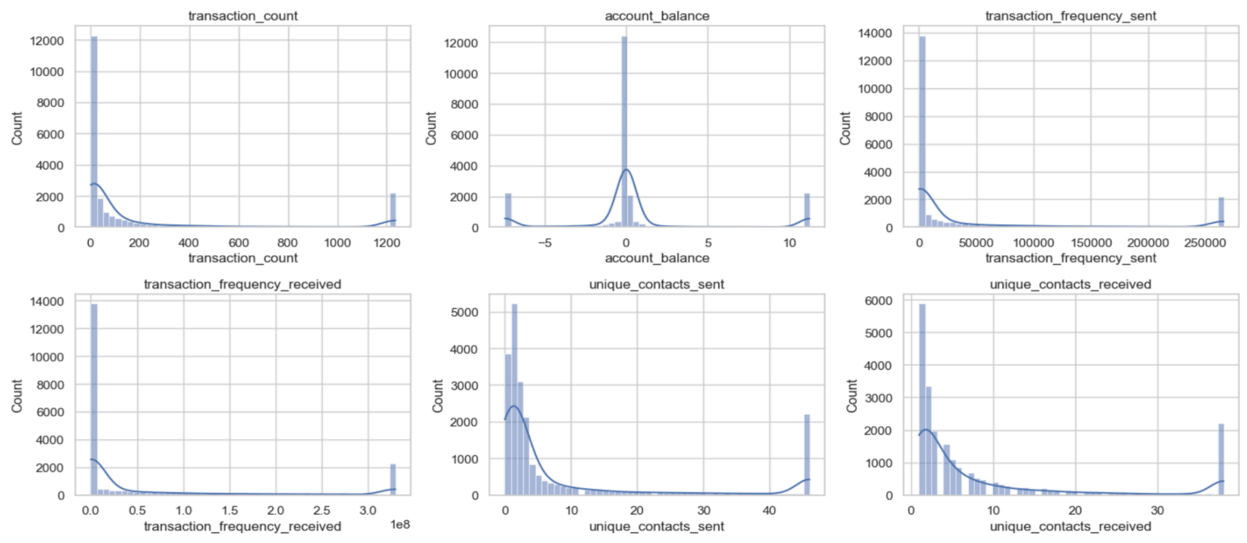
Since there were no null values present in the merged datasets, there was no need for any form of imputation. As for outliers, I decided to use winsorization as I deemed it most appropriate due to the following reasons. I believe that winsorization is the most appropriate way to handle the outliers in this case. Removing the outliers reduces the sample size which was something I want to avoid, whilst mean imputation seemed inappropriate as I felt these values are not measurement errors, thus I want to retain the heterogeneity through winsorization.

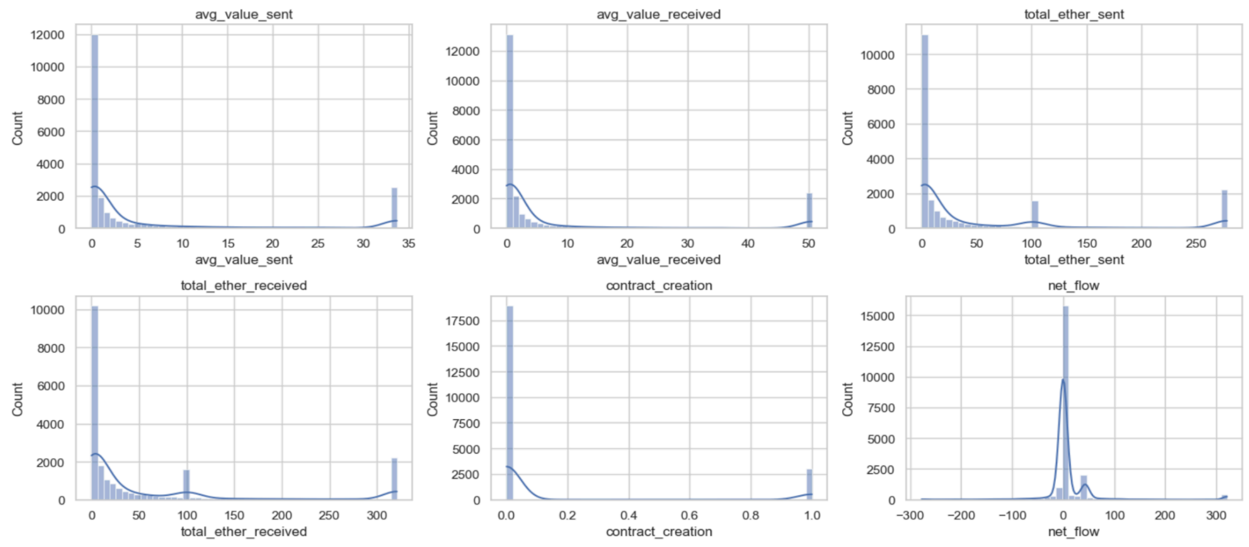
### 3.2 Feature Engineering

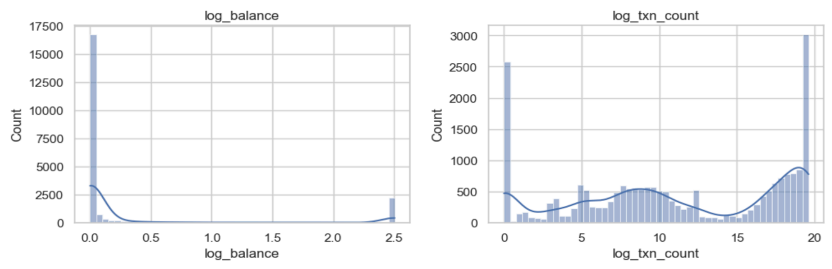
For feature engineering, I created features whilst ensuring that I wasn’t overdoing it or causing any data leakages. In the end, 5 new features were engineered, which are described in more detail in the notebook.

## 4. Data Validation

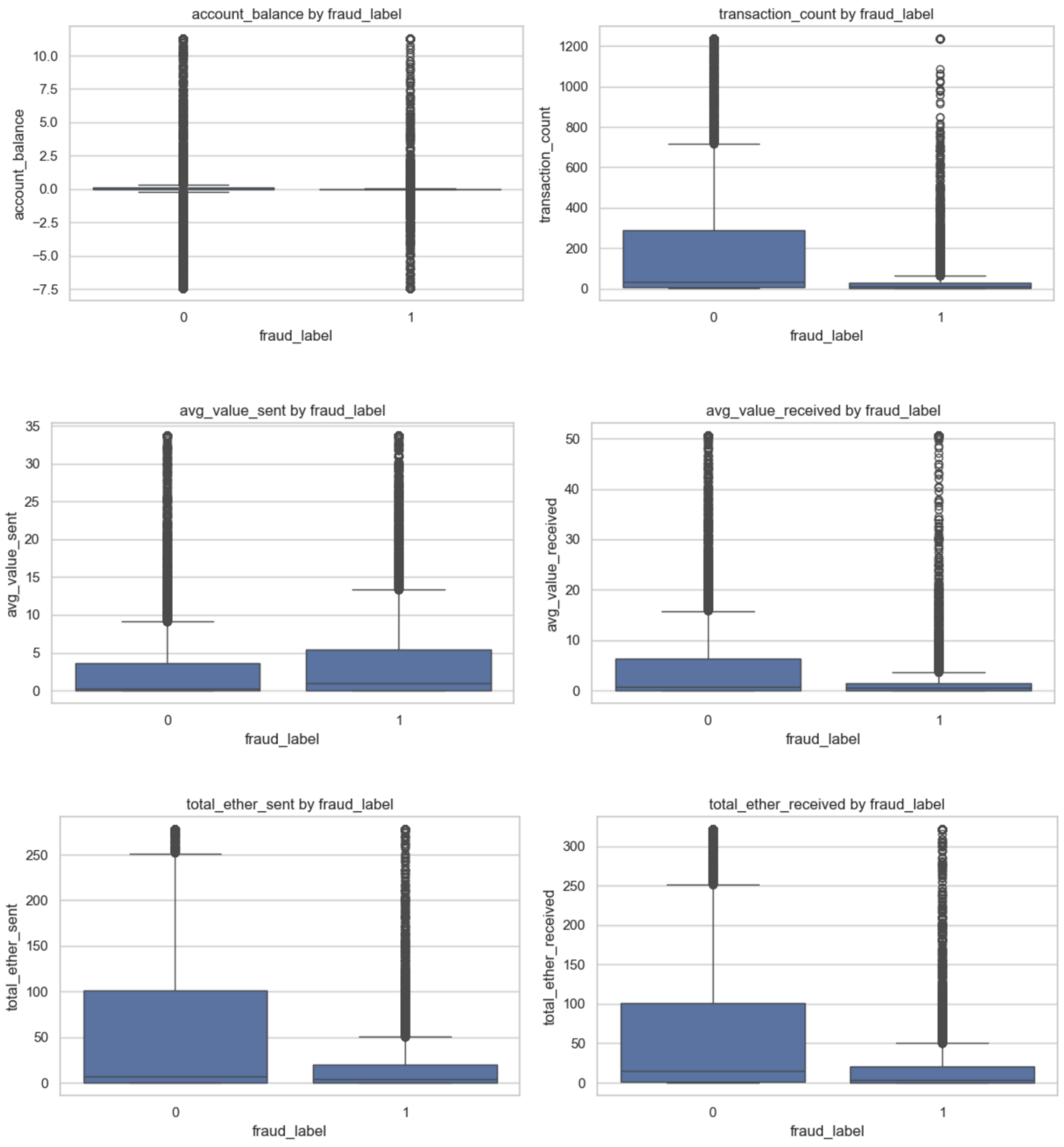
After the data cleaning, I created some plots to ensure that the data distribution was appropriate to proceed to model training. Below are the histogram plots for the distribution of each of the features.







I also plotted the boxplots for certain features separated by their fraud labels, the diagrams are displayed below.



## 5. Model Training and Hyper Parameter Tuning

For the model training, I will be training the models on the final dataset from the preprocessing and feature engineering. I will first do some baseline evaluations of the models using ROC AUC scores and Precision-Recall AUC scores. These metrics were chosen as ROC AUC scores are insensitive towards class imbalance while Precision-Recall AUC scores are sensitive towards class imbalance.

Furthermore, I also trained the models on a SMOTE balanced dataset to see which process results is a superior model. For this, only the train set was SMOTE balanced, whilst the validation and test sets were left unbalanced to ensure that the results are accurate of real world conditions.

It was found that the models trained on the imbalance train set had a better performance, which is unsurprising as there was no significant imbalance in the original train set, and balancing the train set through SMOTE likely resulted in more false positives instead of enhancing the models performance.

## 6. Results

After hyperparameter tuning, the best estimators were used and the models were subsequently evaluated on the validation set, which showed that XGBoost performed the best out of the 3 models, followed by Random Forest then Logistic Regression.

Logistic Regression performed the worst likely due to the linear nature of the model. Ethereum transactions exhibit non linear patterns, which causes it to struggle to capture complex interactions unless sufficient features are engineered. On the other hand, Random Forest and XGBoost can automatically model non-linearities and feature interactions which are critical in fraud detection. Furthermore, tree-based models are less sensitive to outliers as they split on thresholds instead of fitting coefficients. On the other hand, Logistic Regression can be pulled by extreme values, especially without careful scaling or regularization

XGBoost performed better than Random Forest, likely due to the ensemble effects. Random Forest uses an ensemble of many trees which helps to reduce variance and improve generalization. On the other hand, XGBoost uses gradient boosting which sequentially corrects mistakes and captures subtle patterns.