Ethereum Fraud Detection

Problem Description

With the rising popularity of cryptocurrency, fraud cases have surged, posing significant risks to individuals and organizations. Due to the evolving and complex nature of fraud methods, detecting fraudulent transactions on cryptocurrency networks like Ethereum remains a challenge. This project aims to leverage both traditional machine learning and neural networks to analyze transaction patterns and detect fraudulent activities. By developing reliable predictive models, we hope to aid in identifying suspicious transactions, reducing financial losses, and assisting law enforcement in preventing cryptocurrency fraud.

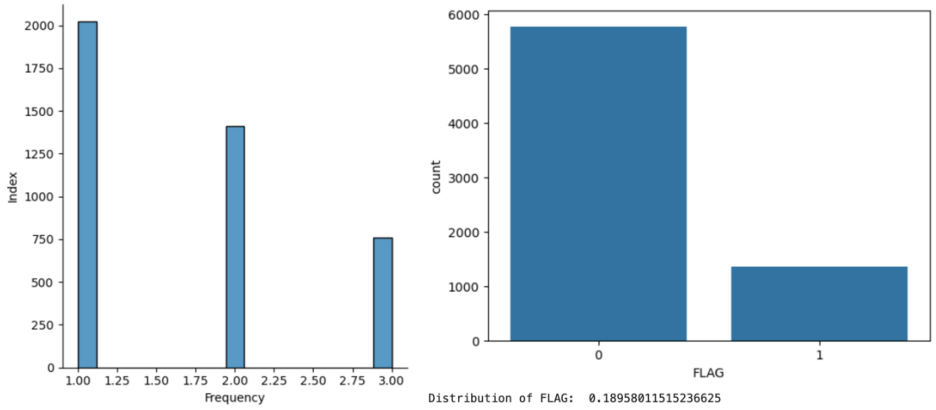
Dataset Overview

- **Dataset Size:** 9,842 rows and 51 columns.
- **Data Source:** https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset/data
- **Target Variable:** `FLAG` (0 for valid transactions, 1 for fraudulent transactions).
- **Feature Selection:**
    - **Transaction Timing:** Avg min between sent/received transactions, time difference between first and last transaction.
    - **Transaction Volume & Value:** Total Ether sent/received, min/max/avg transaction values.
    - **ERC20 Token Transactions:** Total ERC20 transactions, values of ERC20 transfers.
    - **Interaction Patterns:** Unique addresses interacted with (sent/received).

After initial analysis, we removed non-informative attributes such as `Unnamed: 0`, `Index`, and `Address` to reduce noise.

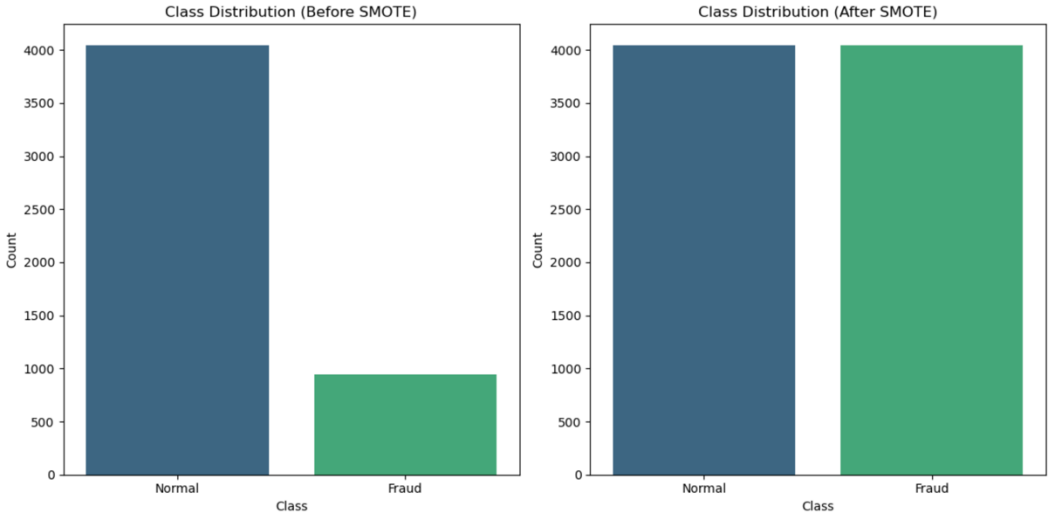
Methodology

- **Data Preprocessing**
  By checking the unique value of each field, we find that Unnamed: 0 and Address have almost no duplicates, so they are removed. It is worth mentioning that we see that the index in the material is not unique, and it cannot be used as an index. Moreover, using “LabelEncoder” to transfer the “dtype “of columns from ‘object’ to number, and the distribution of 1 in flag is 0.189.



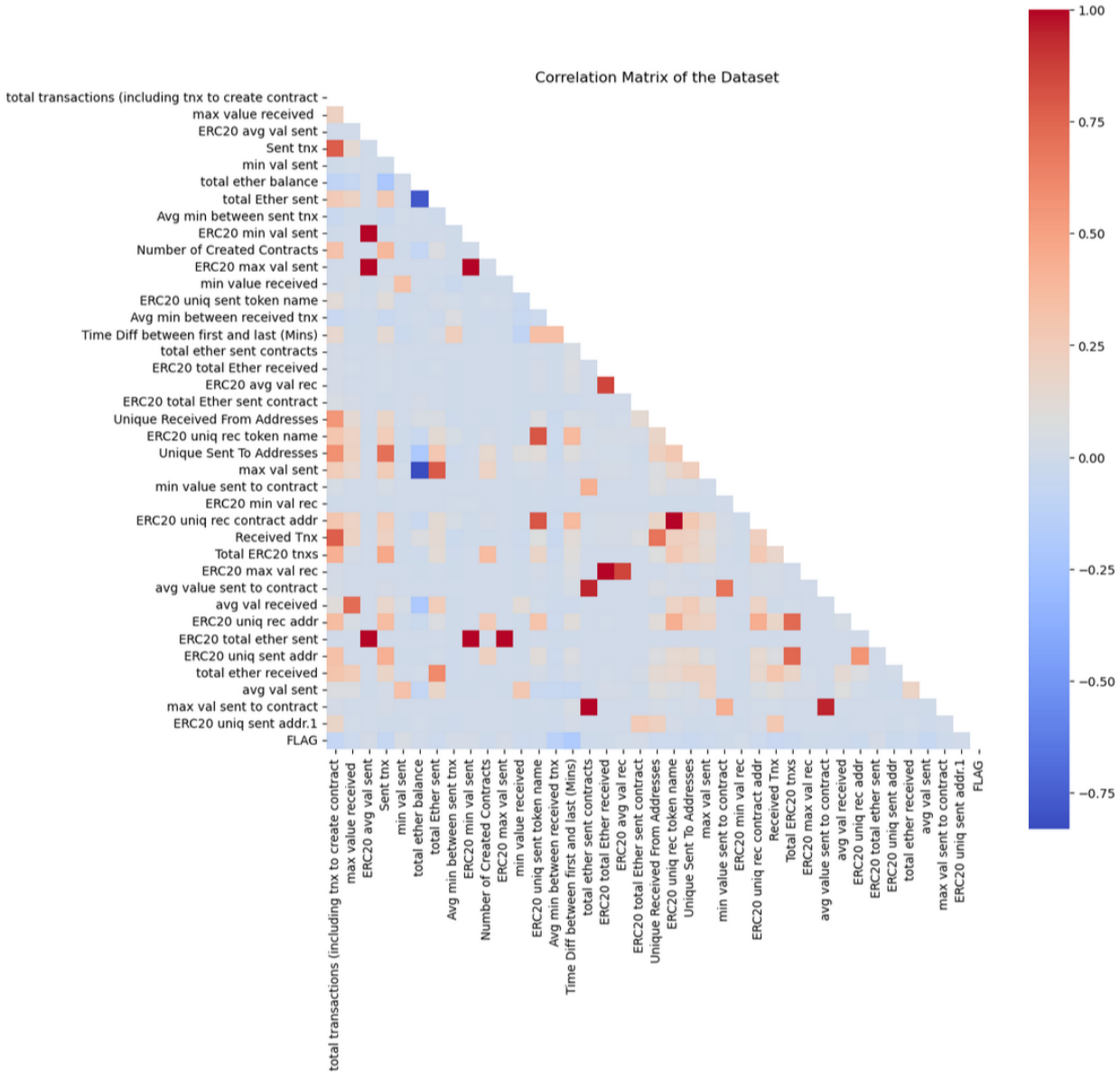
- **Data Balancing**

  Split the data into 70% training data and 30% test data and use SMOTE to generate synthetic samples for the minority class.



- **Exploratory Data Analysis**

  Through Correlation Matrix of the Dataset, we can find that the FLAG feature (which likely indicates fraudulent transactions) has weak correlations with most other features. However, it's quite common in fraud detection problem, it is typically identified based on a combination of factors rather than a single variable.

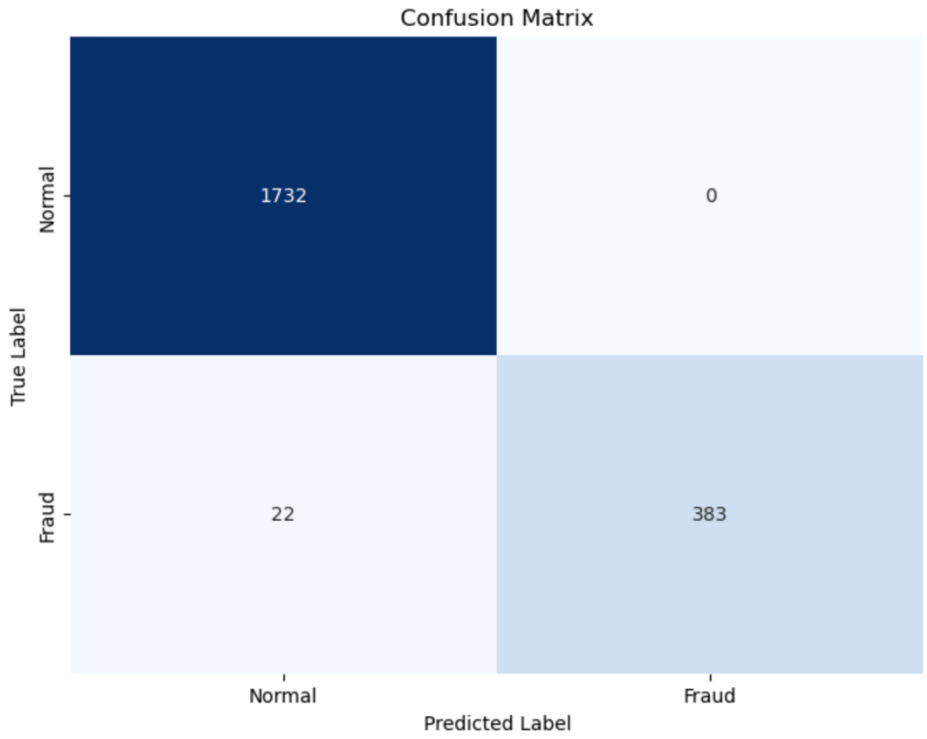


To solve the binary classification problem, we applied multiple machine learning models as well as a Fully Connected Neural Network (FCN). The dataset was preprocessed, balanced using **SMOTE**, and split into training (70%) and testing (30%) sets.

- ## Traditional Machine Learning Models

1. **Random Forest:**
    - An ensemble learning method combining multiple decision trees.
    - Utilized Bootstrap Aggregation (Bagging) to generate multiple datasets.
    - Model trained using `RandomForestClassifier` with `StratifiedKFold` cross-validation.

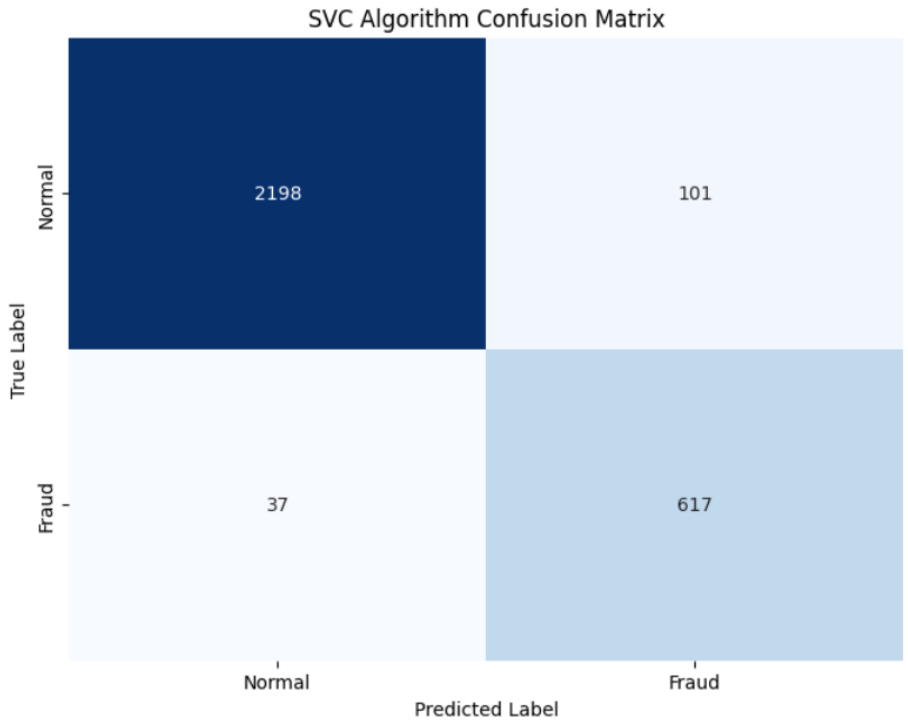
```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1732
           1       1.00      0.95      0.97       405

    accuracy                           0.99      2137
   macro avg       0.99      0.97      0.98      2137
weighted avg       0.99      0.99      0.99      2137
```



2. **Support Vector Classifier (SVC):**
   - Finds an optimal hyperplane for classification.
   - Applied kernel transformation for non-linear separation.
   - Used `LabelEncoder` for categorical data transformation.

SVC Algorithm Classification Report:

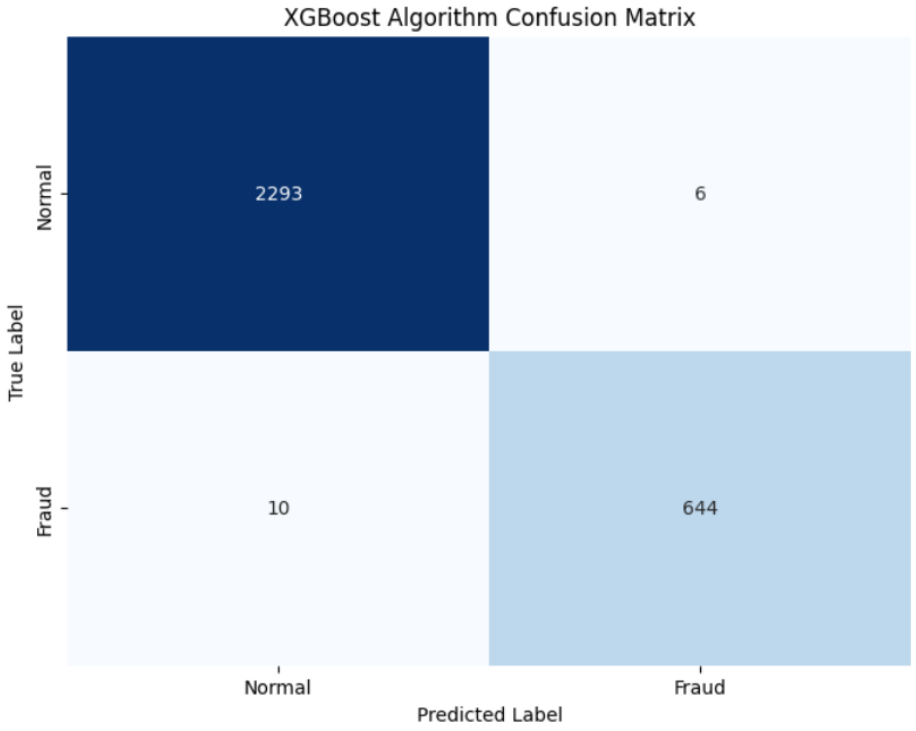
	precision	recall	f1-score	support
0	0.98	0.96	0.97	2299
1	0.86	0.94	0.90	654
accuracy			0.95	2953
macro avg	0.92	0.95	0.93	2953
weighted avg	0.96	0.95	0.95	2953



3. **XGBoost:**
   - Gradient boosting-based ensemble method.
   - Applied boosting techniques to improve prediction accuracy.
   - Evaluated using `eval_metric=error` for binary classification.

XGBoost Algorithm Classification Report:

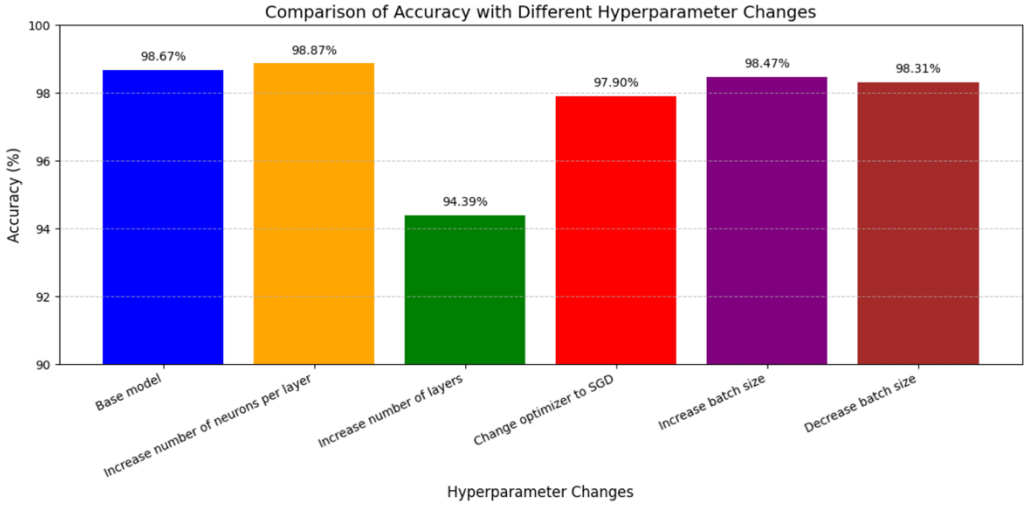
	precision	recall	f1-score	support
0	1.00	1.00	1.00	2299
1	0.99	0.98	0.99	654
accuracy			0.99	2953
macro avg	0.99	0.99	0.99	2953
weighted avg	0.99	0.99	0.99	2953

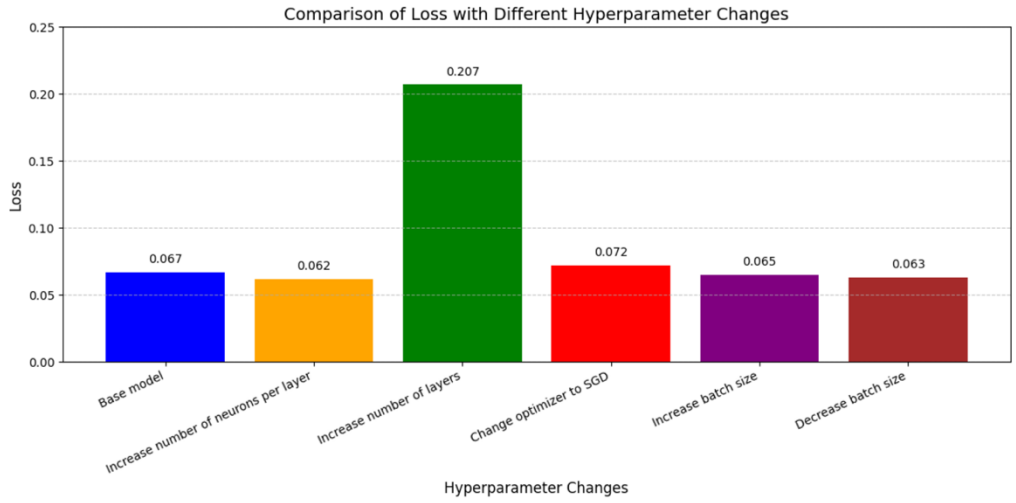


- **Neural Network Model (Fully Connected Network - FCN)**

  - **Architecture:**
    - Input layer with 4 neurons.
    - Five hidden layers with 4 neurons using ReLU activation.
    - Output layer with a sigmoid activation function.
  - **Loss Function:** Binary Cross-Entropy.
  - **Optimizer:** Adam (with comparison to SGD in tuning experiments).
  - **Training Parameters:** Batch size of 32, trained over 10 epochs.
  - **Hyperparameter Tuning:**
    - **Base Model:** The initial FCN model consisted of five hidden layers, each containing four neurons, trained with a batch size of 32 over 10 epochs.
    - **Tuning Process:** We experimented with different values for the number of neurons per layer, the number of layers, optimizer type, batch size, and dropout rate.
    - **Key Findings:**
      - Increasing the number of neurons slightly improved performance, but had minimal impact overall.
      - Increasing the number of layers beyond five led to performance degradation due to overfitting.
      - Changing the optimizer from Adam to SGD resulted in a small drop in accuracy.
      - Batch size adjustments had a negligible effect.
      - Introducing a dropout rate of 0.2 significantly reduced overfitting and improved generalization.

- **Final Model Configuration:** The best-performing model consisted of five hidden layers, each with 128 neurons, trained with Adam optimizer, a batch size of 256, and a dropout rate of 0.2.





Evaluation and Results

All models were evaluated using **k-fold cross-validation** and measured for **accuracy** and **loss**.

Model	Accuracy	Observations
Random Forest	98.97%	Excellent fraud detection, only 22 misclassifications.
SVC	95.00%	Higher false positive rate (101 normal transactions misclassified).

Model	Accuracy	Observations
XGBoost	99.00%	Most effective traditional model with low misclassification.
FCN (Tuned)	98.72%	Number of layer: 5 Number of neurons: 128 Batch size: 256 Optimizer: Adam Dropout rate: 0.2 Loss: 0.0469

Findings:

- **XGBoost outperformed all traditional models**, achieving the highest accuracy.
- **FCN performed competitively**, demonstrating that deep learning can capture complex patterns.
- **Hyperparameter tuning played a crucial role** in optimizing FCN performance.
- **SMOTE successfully balanced the dataset**, preventing the models from favoring the majority class.

Discussion and Future Work

- While **Random Forest and XGBoost** performed well, deep learning approaches like **FCN offer flexibility** in capturing subtle fraud patterns.
- **Tuning hyperparameters** (e.g., number of layers, batch size, and dropout) is crucial in deep learning models.
- **Feature engineering** could improve model performance by incorporating additional transaction attributes.
- Future work could explore **Graph Neural Networks (GNNs)** or **Recurrent Neural Networks (RNNs)** to analyze sequential transaction data

---

GitHub Link: https://github.com/Rita94105/Ethereum-Fraud-Detection