CSE C10 Data Intensive Computing
Project Phase 3

Name: Mali Pashupathi
Roll Number: 20211A05F8

Name: Kondakalla Bhoomika Reddy
Roll Number: 20211A05D6

Name: Kondaveeti Bhargav
Roll Number:20211A05D7

Ethereum Fraud Detection

Table of Content

Motivation:

Investors of all stripes have become interested in the cryptocurrency craze during the past few years. Additionally, it has drawn the interest of scammers. Most cryptocurrency scams try to deceive their victim into sending money to a hacked digital wallet. These targets are probably being persuaded by the enormous returns promised by the attackers through the use of specialized social engineering techniques like romance scams, email phishing, and even Ponzi schemes.

We plan to use the method of supervised and unsupervised learnings, in order to construct different types of classifiers to detect fraudulent Ethereum transactions, in order to put our security analyst talents to use in tackling real-world issues.

Problem Statement:

This is the problem which focuses on Ethereum transactions, and predict the transactions is fraudulent or not taking numerous inputs from the user.

Data Source:

The Ethereum Fraud Detection Dataset, an open-source, labeled dataset from Kaggle, contains over 10,000 samples. The dataset is hugely imbalanced.

Link to data source: Ethereum Fraud Detection Dataset | Kaggle

Training Models:

We have trained our data on different models, they are:

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. Gradient Boosting
5. XGBoost
6. AdaBoost
7. K - Nearest Neighbors
8. Support Vector Machines(SVM)

It is clear that, we need accurate results while giving the inputs. So, we have trained data using different models.

Model	Accuracy
Logistic Regression	81%
Decision Tree	96%
Random Forest	98%
Gradient Boosting	98%
XGBoost	99%
AdaBoost	96%
KNN	86%
SVM	70%

Model	AUC value
Logistic Regression	0.572
Decision Tree	0.957
Random Forest	0.975
Gradient Boosting	0.973
XGBoost	0.979
AdaBoost	0.963
KNN	0.865
SVM	0.686

Based on accuracy and AUC values, it is clear that XGBoost is giving better results. So, from here on we have decided to take our project ahead using XGBoost classifier.

The Final model: XGBoost classifier

XGBoost classifier's speed and performance are unparalleled and it consistently outperforms any other algorithms aimed at supervised learning tasks.

The Basic and main requirements for XGBoost to perform well are:

1. Numeric features should be scaled.
2. Categorical features should be encoded.

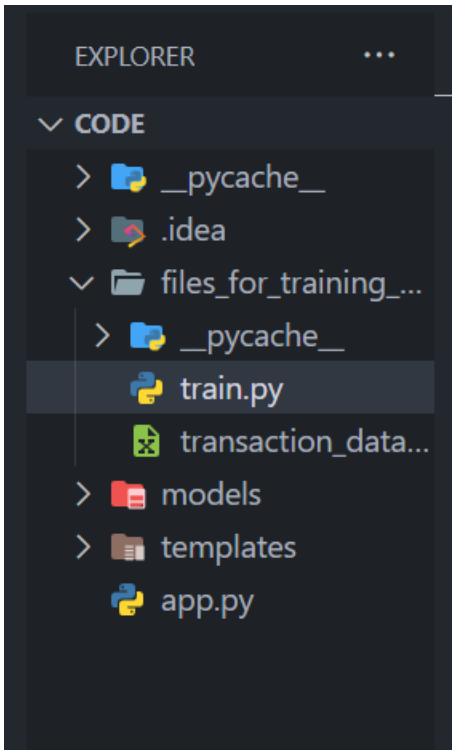
As an ensemble learning algorithm, XGBoost combines the output from numerous base learners to provide a prediction.

Working Instructions:

1. Extract the Zip file which contains the code.
2. Environment we have used to run the code for web application is **PyCharm**. (One can use any editor to run the code).
3. Libraries that need to be imported before executing the code are:

a. Numpy
b. Pandas
c. Sklearn
d. XGBoost

The structure of the file is:



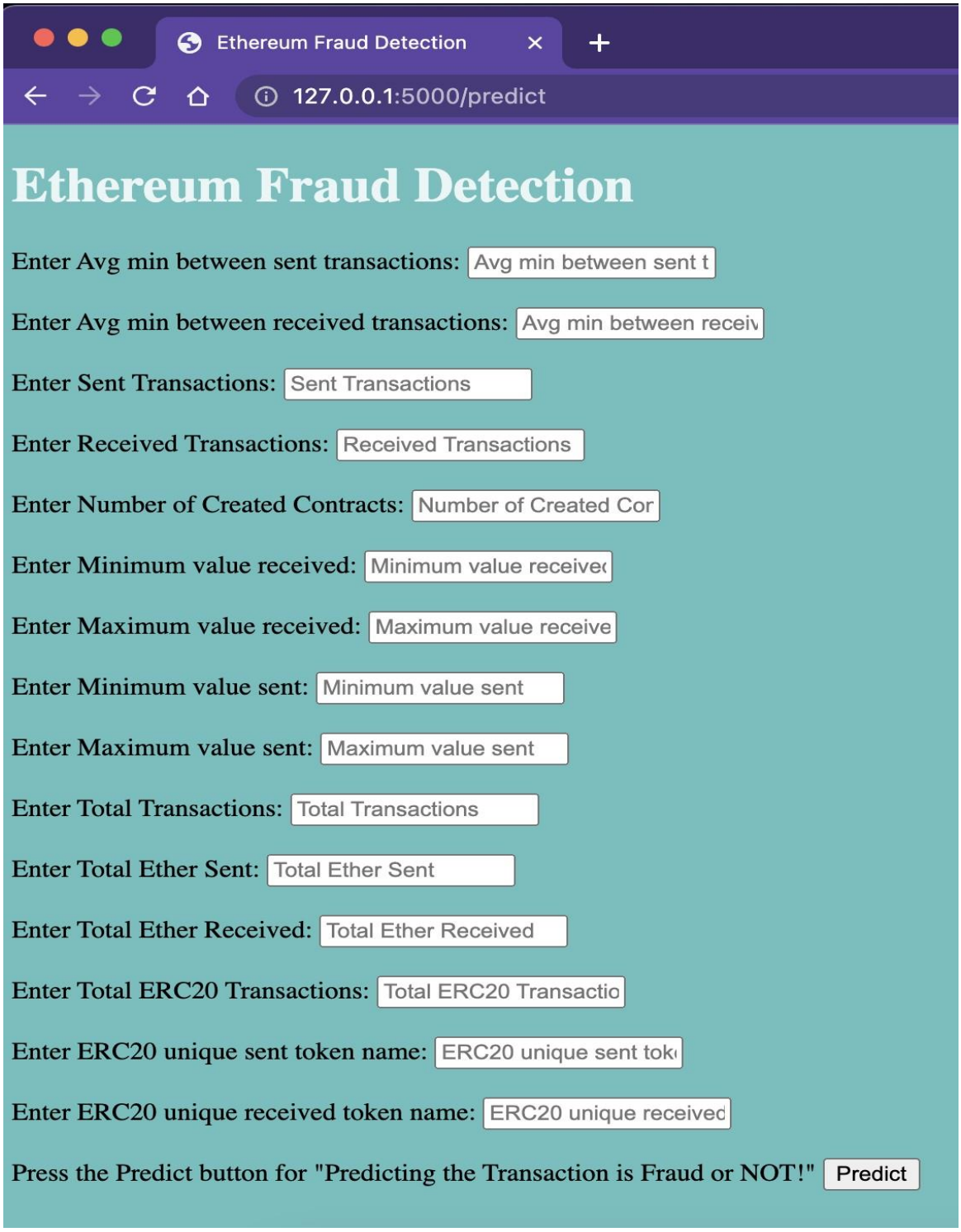
4. Open DICProject and then **files_for_training_model**
5. Our code resides in the above path with the name **train.py**
6. Open the terminal and then give the command **flask run**

The Terminal window looks like this:

```
(base) bharadwajvishnubhotla@Bharadwajs-MacBook-Air DICProject % flask run
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [02/Dec/2022 21:41:24] "GET / HTTP/1.1" 200 -
1
127.0.0.1 - - [02/Dec/2022 21:45:54] "POST /predict HTTP/1.1" 200 -
0
127.0.0.1 - - [03/Dec/2022 17:33:46] "POST /predict HTTP/1.1" 200 -
```

7. Copy the url (Ethereum Fraud Detection) and paste it in the browser.

After pasting the URL in the browser, the user a HTML page like this:



Here the end user is given the fields like Avg min between sent transactions, Avg min between received transactions, Sent transactions, Received transactions etc.. and in the background we have created a “train.py” file in which we choose the XGBoost Classifier for getting better result and this will run and predict the FRAUD status for the given coordinates.

Here is the code for the XGB classifier:

```
def predict(data):
    data.drop(['Unnamed: 0', 'Address', 'Index'], axis=1, inplace=True)
    col = data.select_dtypes(['object', 'category'])
    data.drop(col, axis=1, inplace=True)
    data[data.columns] = data[data.columns].apply(pd.to_numeric, errors='coerce')
    data.fillna(data.median(), inplace=True)
    X = data.drop('FLAG', axis=1)
    y = data['FLAG']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
    xgb = XGBClassifier()
    xgb.fit(X_train, y_train)
    print(xgb.score(X_train, y_train))
    pickle.dump(xgb, open("models/finalized_model.pkl", 'wb'))
    prediction_test = xgb.predict(X_test)
    return None
predict(data)

test_data = []
def test_model (test_data):
    model = pickle.load(open('models/finalized_model.pkl', 'rb'))
    prediction_test = model.predict(test_data)
    return prediction_test[0]
test_model(test_data)
```

8. The user needs to enter the values, and the press the Predict button.

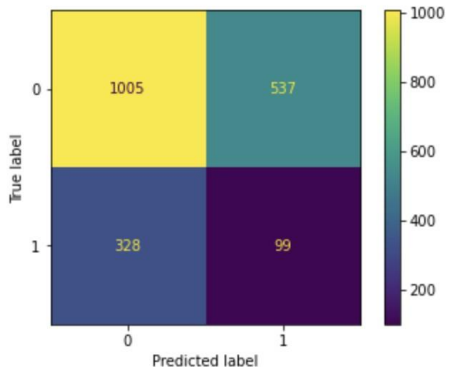
After pressing the Predict button the FRAUD status is displayed in the same web page, after the button. Status will be shown like the image below:

The Classification report of the model is as follows:

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

           0       0.99      0.99      0.99      1542
           1       0.97      0.97      0.97       427

    accuracy                           0.99      1969
   macro avg       0.98      0.98      0.98      1969
weighted avg       0.99      0.99      0.99      1969
```

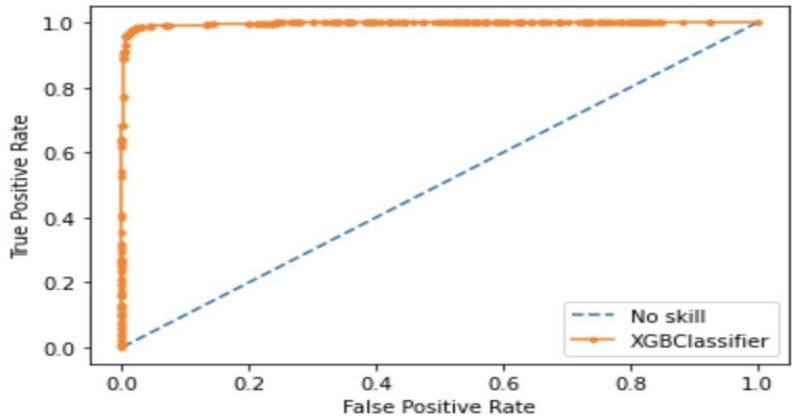
The Confusion matrix of the Classifier:

```
Confusion matrix of XGBoost classifier:
[[1529   13]
 [  14  413]]

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fa92be0e670>
```



The AUC value and ROC curve of the classifier:

```
AUC: 0.979
```



Analysis and Recommendations:

Generally by this type of application, people who are playing with Ethereum can get utmost benefit. i.e, one can easily differentiate between scammer and the real value provider. By being aware of this, frauds are likely to decline as more individuals switch to reliable ether providers and transactions become more deliberate now that everyone is watching.

To training our model we used the Ethereum Fraud detection dataset from Kaggle website which consists of 9841 records. Only the important features are used (the features shown in the web app), to find whether the transaction is legitimate or not.

Some of the useful recommendations are:

1. Adding a Country column so that people and Government of the country knows how many frauds are happening.
2. One can also try implementing this problem using the Artificial Neural Network.
3. The Multi-layer Perceptron model can be used to learn and decide which features should be combined in order to identify various interactions between them as needed as part of future research.
4. Can add a column on how likely Ethereum can get fraud with a link to other cryptocurrencies.

Reference:

1. **268 - How to deploy your trained machine learning model into a local web application? - YouTube**
2. **Flask in PyCharm Community Edition | by Mushtaque Ahmed | Medium**
3. **How to create a classification model using XGBoost in Python (practicaldatascience.co.uk)**