Kaylynn Mosier
January 23, 2025

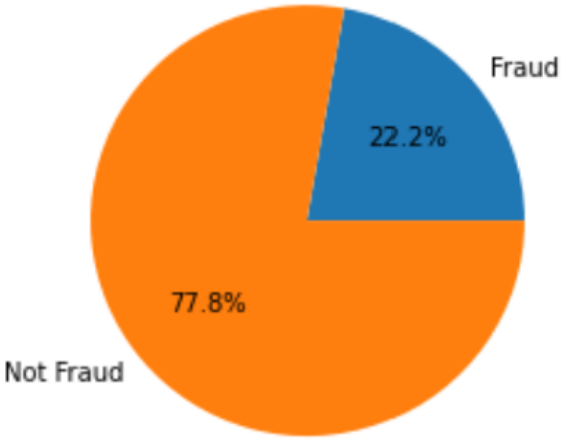
Fraud Detection on Ethereum Dataset

Background

Fraudulent transactions can result in tremendous loss of profits for any company. For those companies operating with online platforms, an accurate fraud detection system is essential to minimize this profit loss. Classification models can be built to quickly and accurately notify businesses of fraudulent transactional activity. The following research will be used to build a classification model aimed at reducing the number of fraud transactions allowed in a business's payment system to ultimately increase profitability of the company. The best model will be determined by weighing accuracy scores as well as type I vs type II error rate.

Data Explanation & Preparation

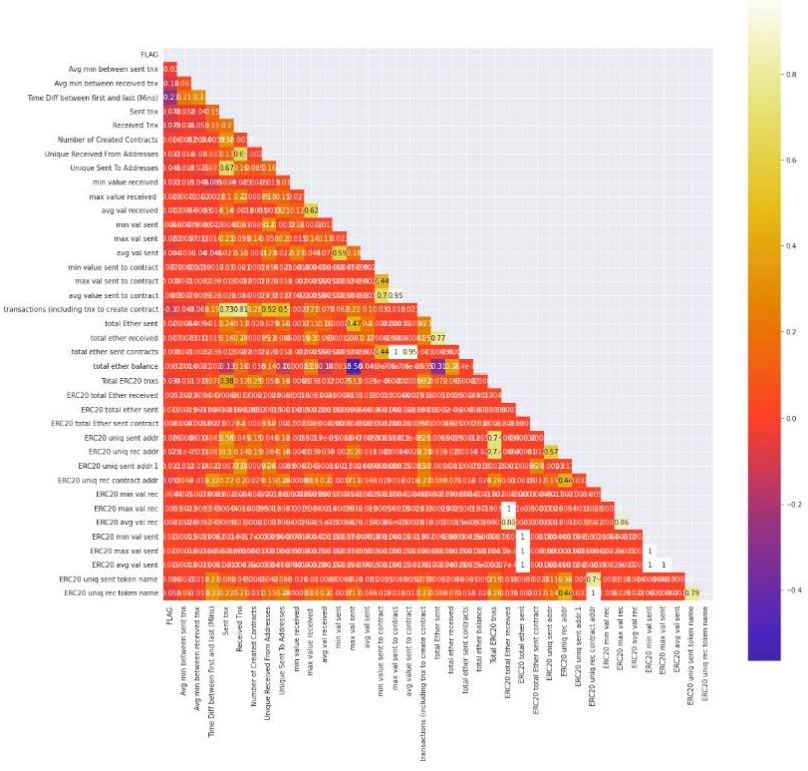
The dataset is composed of fraudulent and authentic transactions made over Ethereum, a cryptocurrency platform. It contains 49 columns of information pertaining to the transaction amount, time, location, and other relevant information stored in the process of making a transaction. Importantly, this dataset contains a column marking transactions as either fraud or not fraud so any model built from it will be supervised. However, this dataset has severely imbalanced classes, in which 22.2% of transactions are fraud and 77.8% are not fraudulent.



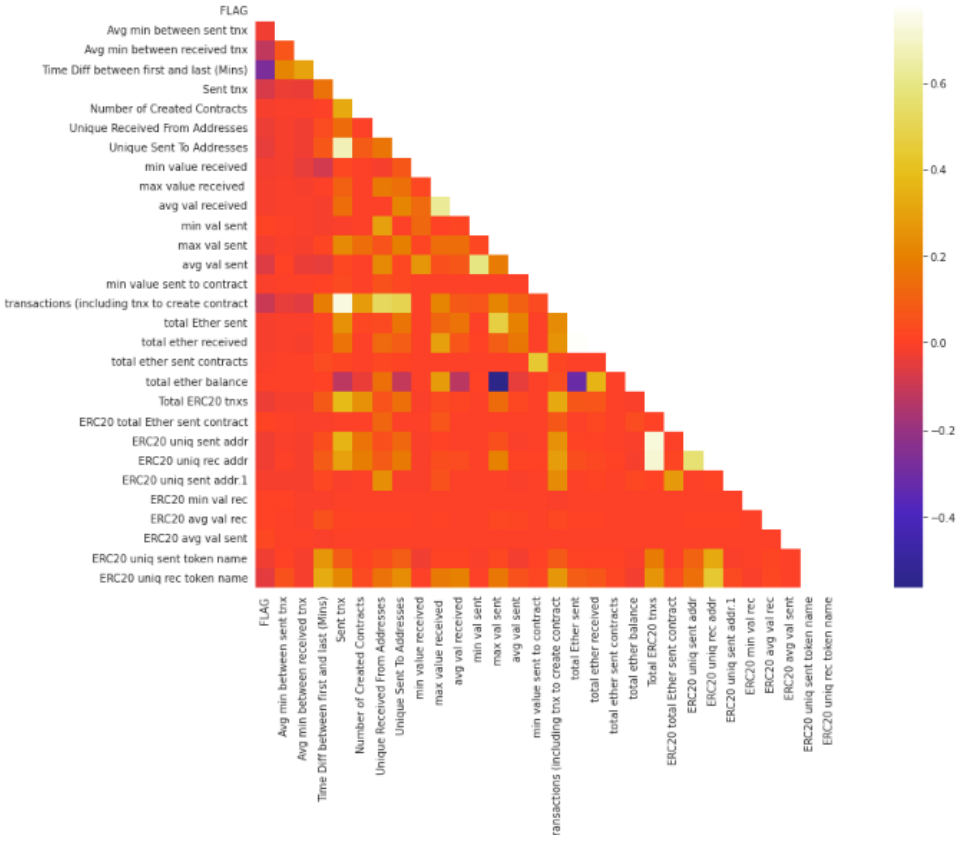
To begin preparing the data for modeling, I removed the three categorical columns. Generally, removing columns as the first step of pre-processing is not advised. However, transforming these categorical columns into numerical or dummy columns would have added to the already very large number of columns. Of the remaining columns, over 20 contained missing values. Rather than drop rows with missing values and reduce my sample size, I filled missing values with median of that column.

The next major step in preprocessing the data was understanding the correlation between each column with the target column. In this case, the target column is called FLAG and marks transactions as fraudulent or not fraudulent. After running a correlation analysis, it was revealed that no columns had a strong positive or negative correlation with FLAG. However, 7 columns resulted in an NaN value in the correlation table. Columns that do not have any correlation with the target variable will not increase the effectiveness of a model so these columns were dropped.

Due to the large number of columns, another concern in this dataset is collinearity, in which independent variables correlate with each other rather than with the target variable. To check for this, I plotted a correlation table of each column with all other columns in the dataset. Any columns that display collinearity, will have a high correlation. I removed one column from each pair of columns that displayed a correlation greater than

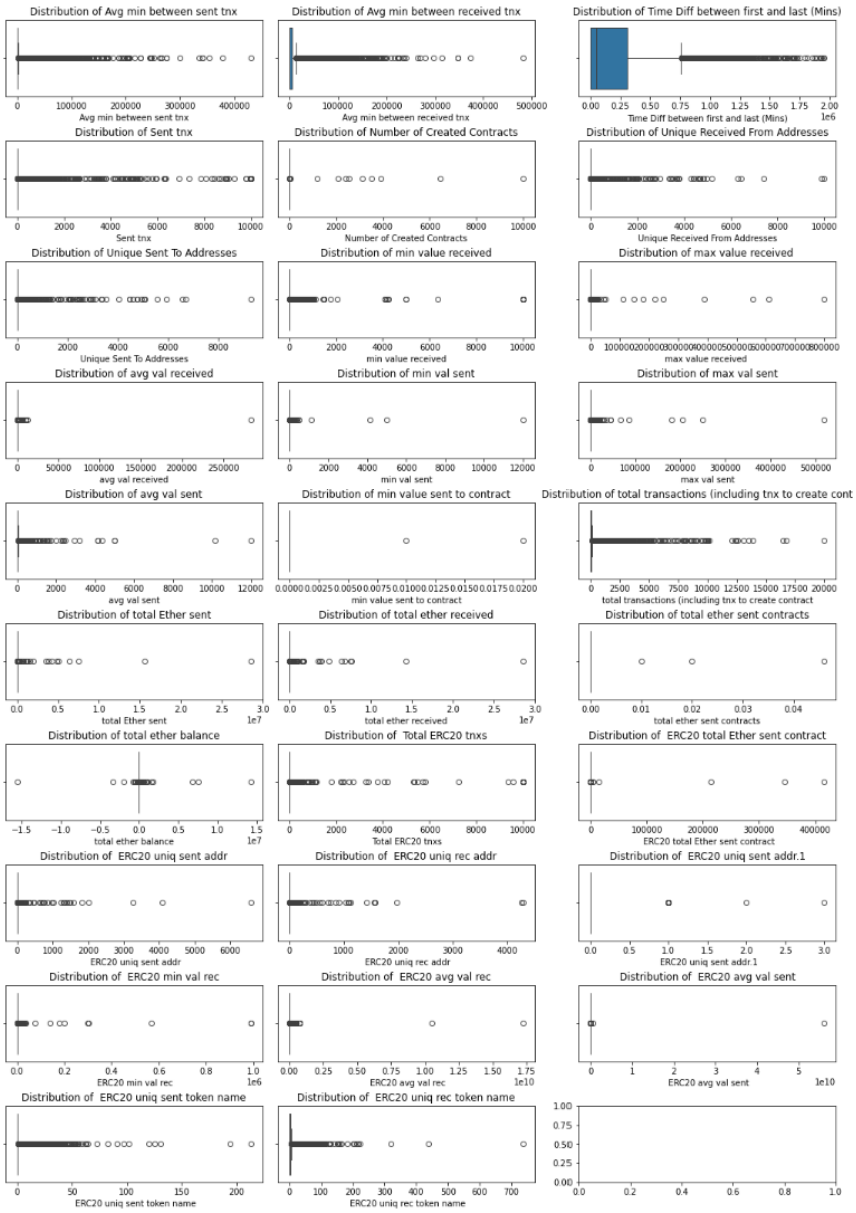


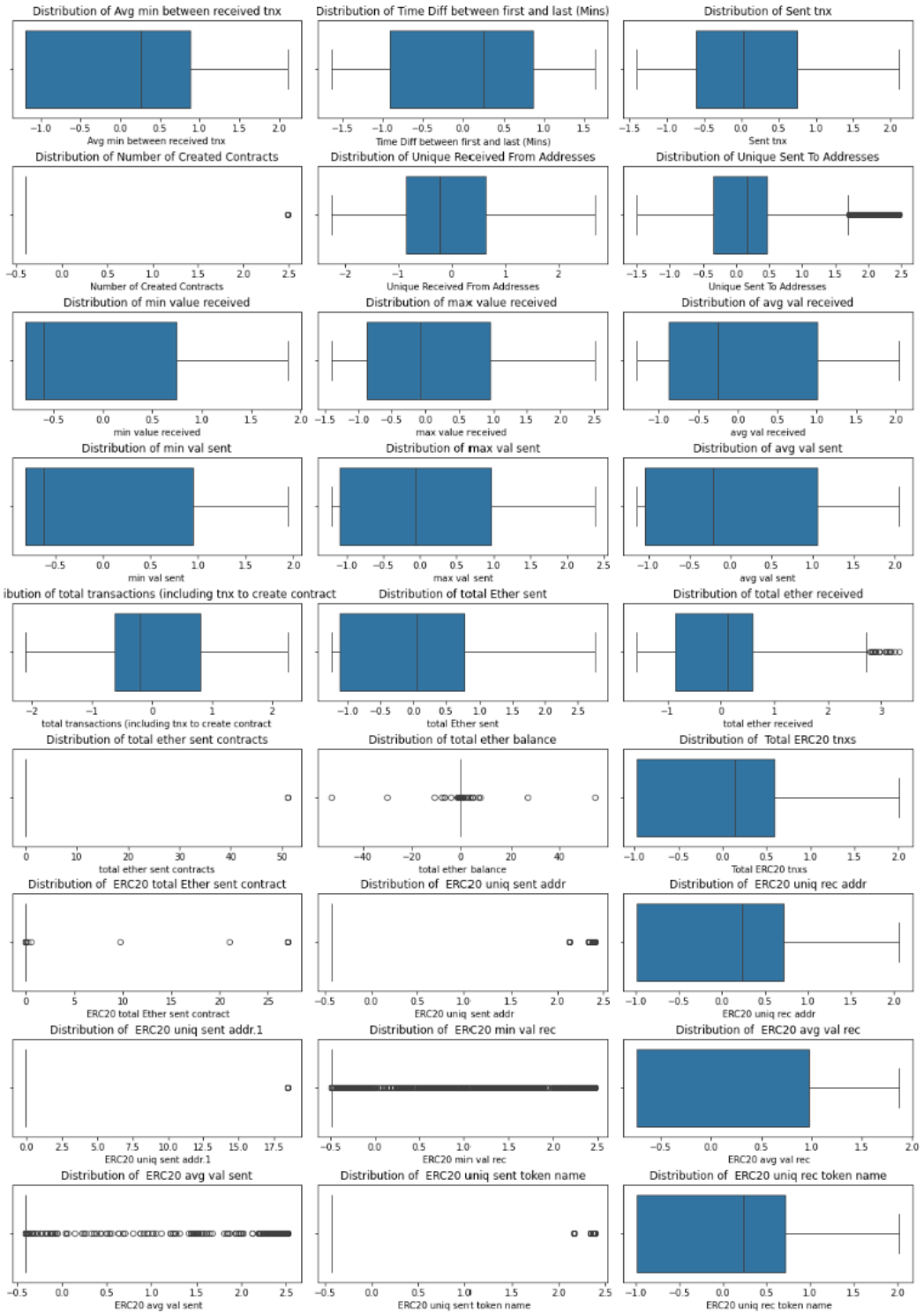
or equal to 0.80. The below correlation table displays the correlation values before and after removal of collinearity.



The next step in data preparation was feature engineering, in which raw data is transformed into features more suitable for modeling. To begin I calculated the variance of each attribute in the dataset. If an attribute has a variance of 0, all values are the same and it does not add anything to the predictive power of the model. In this dataset, no features had a variance of 0 so I was not able to eliminate any columns this way.

Next, I plotted a box plot of each feature in the dataset. This allowed me to better understand the spread and outliers in each. The boxplot of the feature ‘min value sent to contract’ revealed that this variable contained almost all observations at one point. This adds almost no additional information to the model, so I removed this column. Observing the rest of the boxplots revealed the majority of features had a large spread and numerous outliers. Additionally, some features ranged from 0-3 while others ranged from 0 to 8000. In modeling, features with significantly higher values would be weighted more than those with lower values, even if they are not actually more significant. To combat this, I applied the PowerTransformer from sklearn to the features which is used to make data have a more normal distribution. The below images display the box plots before and after normalization.





The final step in data the data preparation process was to deal with the imbalanced classes in the target variable. As shown above, the dataset contains 22% fraudulent and 78% not fraudulent transactions. For a binary classification problem, to avoid biasing the

model, the classes must be roughly 50/50 in the target variable. To help deal with this class imbalance, I applied SMOTE (synthetic minority over-sampling technique) from imblearn to the dataset. This technique generates synthetic samples for the minority class, in this case fraud transactions, until the classes are balanced. After oversampling, there are 6156 non-fraudulent transactions and 6157 fraudulent transactions.

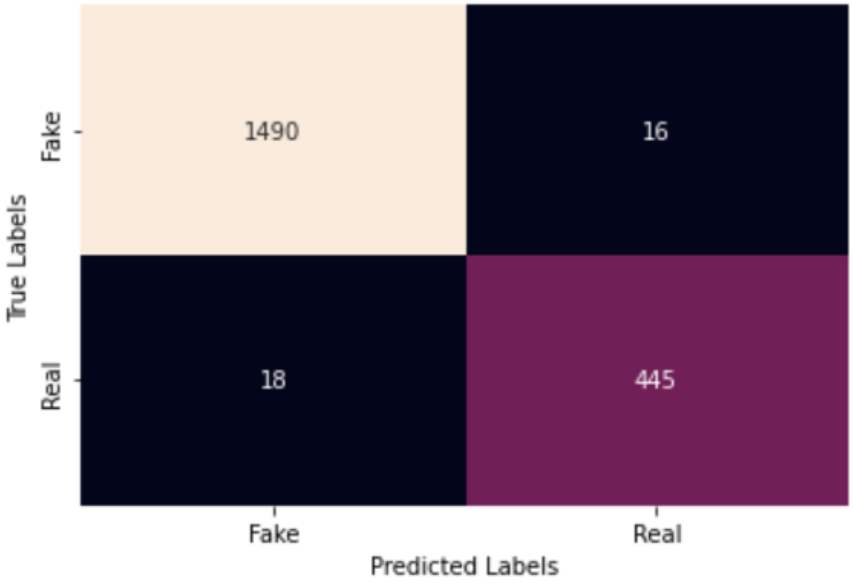
Methods & Analysis

The goal of this study is to classify transactions as either fraud or not fraud which requires a classification model. There are many classification models to choose from and making the right choice can be a difficult and time-consuming task. To aid in this decision, I employed a grid search to test logistic regression, random forest, decision tree, and K-neighbors at one time. Each of the models tested are good at binary classification problems, but it is difficult to determine which is the best without trying them all. A grid search allows simultaneous testing of each model to determine the best model based on the desired metric.

The grid search works by training each model on a labeled subset of the data called the training set. Each model is then evaluated against an unlabeled subset of the data, called the test set. Accuracy can then be found by comparing the predicted labels of each transaction in the test set with the actual labels. In this case, accuracy is the most important metric, because the best model will correctly mark transactions as fraudulent or not fraudulent most often. Therefore, the best model will be the one that produces the most accurate results. After implementing grid search, it was revealed that the best model was the random forest which produced a 98.2% accuracy.

Another challenge of this task is to balance Type I and Type II errors. Type I error occurs when a non-fraudulent transaction is labeled as fraudulent (false positive) while type II error occurs when a fraudulent transaction is labeled as non-fraudulent (false negative). There is a larger business impact in instances of false negatives. In this case a fraudulent transaction would be allowed to continue, and the business would have an immediate loss of profit. In model building, type II error must be reduced as much as

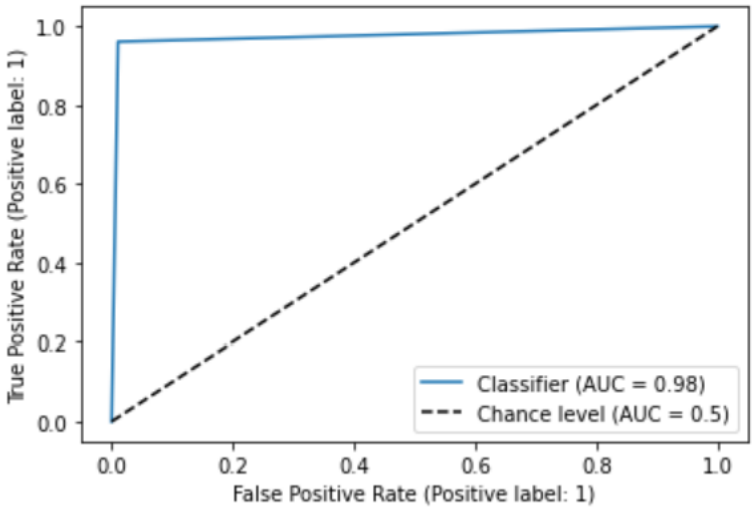
possible. In addition to evaluating models for accuracy, I evaluated their balance of type I and type II error by visualizing a confusion matrix. For the random forest model, there are 16 instances of type I error and 18 instances of type II error. Once again, this model is a good choice because there are less instances of type I error than type II error.



This first iteration of the random forest model has excellent accuracy and correctly balances type I and type II error. However, there still may be room for improvement by utilizing hyperparameter tuning. In this stage of model building, different hyperparameters of the model such as the number of estimators and maximum number of features can be tested to determine which produces the best model. Similarly to the initial model building stage, I employed a grid search to test each combination of parameters and find the model with the highest accuracy. However, in this instance, hyperparameter tuning did not improve the accuracy. It also did not change the type I or type II error rate.

The final model evaluation technique I used was to plot a receiver operating characteristic, ROC, plot. In this plot, the true positive rate is plotted against the false positive rate at different thresholds. The resulting plot shows how well the classifier predicts the class compared to a random classifier. In the below image, the dotted line represents a random, untrained classifier while the blue line represents the trained random

forest classifier created in previous steps. This model is a significantly better predictor of classes than a naïve model.



Conclusion

Fraud detection in transactions is an essential practice that should be employed by all businesses with an online platform. This specific research into using data on Ethereum transactions has shown that creating a model for this task can be achieved quickly with a relatively simple model while also having a high accuracy rate. Any businesses that employ a predictive model for this task must ensure there is a balance between type I and type II error, or they risk missing fraudulent transactions by assuming the model is always correct.

Assumptions

This research has not been done without assumptions about the data. To begin with, this data was initially retrieved from Kaggle, and I was unable to determine where the publisher gained the information. For that reason, this model cannot be readily applied to other Ethereum datasets without further verification of its original source.

Additionally, some assumptions were made during model building. For starters, features that had 'NaN' when calculating correlation with FLAG were removed from the

dataset because I assumed that meant they did not have a correlation with the target variable. However, this assumption may not be correct. During cleaning of the dataset, I filled all missing data with the median of that column. Although this is a common practice, it may not have been the best practice in this specific case. However, having not tested other options I cannot say whether that is true. Finally, during the model evaluation stage I assumed that any business employing this model would want a lower number of type II errors than type I errors, but this may not be the case. Any business who employs a model must evaluate their own business needs and determine what is best for their particular use case.

Limitations & Challenges

As mentioned previously, I cannot be sure that this dataset truly contains transactions from Ethereum which means the application of all learnings must be taken with a grain of salt. In addition, this dataset required extensive preparation which poses its own set of challenges. Anyone intending to perform the same tasks must have a strong understanding of the data cleaning process or they risk making improper decisions during cleaning. Each change made to the data during the cleaning process must be weighed by comparing the increase in efficacy of the model with the reduction in use of the raw data. In a large-scale deployment, these data cleaning steps might be much more time consuming.

Recommendations, Future Uses & Implementation

This research can be used to gain an understanding of the steps required to create a classification model for predicting fraudulent transactions in online purchasing platforms. Each dataset contains its own set of challenges and must be evaluated independently of this research if the same result is desired.

Implementation of this model is dependent on validation of the data source. If the data is truly real transactions from Ethereum, it is likely a good model to prevent fraud. However, if it is not, a new model would need to be built to determine fraud transactions on the Ethereum platform.

Ethical Considerations

I have tried to make only alterations to the data that were absolutely necessary to model building, so I feel data integrity has been maintained. As mentioned above, I do still have concerns about the source of the data and confirming these transactions are real transactions from Ethereum is essential before any deployment processes occur.

In model building, I chose to minimize type I errors to create a conservative model that captured the most possible fraudulent transactions. In practice, it may be preferable to have a better balance between type I and type II errors because the business risks alienating customers if they flag a transaction as fraud when it's a real transaction.

10 Questions from Audience

1. Why did you choose to test the models you did?
    - The models I chose are all known to be reliable classifiers for binary classification tasks.
2. Are there other models that could have been tested?
    - Yes, there are other models that could have been tested such as XGBoost. However, I achieved a highly accurate model in little time so I did not test other models.
3. In hyperparameter testing, you ran a rather expansive grid search which can be very time consuming. Is there a faster way to achieve the same result?
    - I could have done a random search rather than a full grid search. In this practice, only a few of the possible combinations would have been tested and it could have been quicker. However, since the accuracy of the model was already so high, I wanted a more extensive search to ensure no improvement could be made.
4. Why is type I error preferred to be lower than type II error?
    - Type I error should be lower because it will label transactions as fraud, even when they are not. This allows further investigation into transactions and

limits the number of actual fraudulent transactions that fall through the cracks.

5. You mention numerous times that you have concerns about the data source, why did you still choose to use it?
    - I chose to still use it because I found the same dataset in numerous different sources. This leads me to believe that it is a real (or not synthetic) dataset.
6. Why did the best model not have 100% accuracy?
    - No model is perfect, and achieving 100% accuracy would lead me to believe the model had been over fit to the data. So, with a 98% accuracy, we can be sure the model is correctly labeling most transactions without being over fit to the data and limiting it's application to new transactions.
7. What would be the next steps if Ethereum wanted to deploy this model?
    - I would personally test it on a larger dataset and check accuracy and error rates before deployment. If the business is satisfied with those values, deployment should proceed.
8. Why did the model not improve after hyperparameter tuning?
    - Likely because hyperparameter tuning could not further improve the model without overfitting it to the dataset.
9. Does it seem odd that 22.2% of transactions would be fraudulent in this dataset?
    - This does seem like a lot of fraudulent transactions for a normal online platform, however Ethereum is a platform for cryptocurrency and is a wildly unregulated platform, so I don't find that rate of fraud to be surprising.
10. Could you have built a model without resampling to balance the classes?
    - Yes, I could have. However, the model would be at a higher risk of overfitting to the larger class so there would be additional concerns about large scale model deployment.

References

Aliyev, V. (2021, January 3). *Ethereum Fraud Detection Dataset*. Kaggle. https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset/data

Chiticariucristian. (2022, October 5). *Fraud detection: Ethereum transactions*. Kaggle. https://www.kaggle.com/code/chiticariucristian/fraud-detection-ethereum-transactions

II, R. (Bob) R.-. (2023, March 20). *Ethereum Fraud Detection*. Medium. https://bobrupakroy.medium.com/ethereum-fraud-detection-ba4e1d8b262a

KumarI, A. (2024, September 8). *ROC Curve & AUC explained with python examples*. Analytics Yogi. https://vitalflux.com/roc-curve-auc-python-false-positive-true-positive-rate/

Powertransformer. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer.html

Smote#. SMOTE - Version 0.14.dev0. (n.d.). https://imbalanced-learn.org/dev/references/generated/imblearn.over_sampling.SMOTE.html

Tanjung, A. I. (2024, May 18). *Unveiling ethereum fraud: A machine learning approach for advanced analytics*. Medium. https://medium.com/@axelivandatanjung/unveiling-ethereum-fraud-a-machine-learning-approach-for-advanced-analytics-3d4bfcdf0d1c