Blockchain Fraud Detection

Predicting fraud on Ethereum blockchain.

Name	Student Number
Thabo Mashabela	2468512
Tristen Haverly	1864691
Andile Thabethe	2092108
Riekert Holder	2517888
Rushil Patel	1679175

INTRODUCTION

What is Blockchain?

A blockchain is a distributed database that is shared among the nodes of a computer network. Similar to a database, a blockchain stores information electronically in digital format. Blockchains are best known for their role in cryptocurrency systems, such as Bitcoin (in our example Ethereum), for maintaining a secure and decentralised record of transactions. The innovation with a blockchain is that it guarantees the fidelity and security of a record of data and generates trust without the need for a trusted third party.

A database usually structures its data into tables, whereas a blockchain, as its name implies, structures its data into chunks (blocks) that are strung together. This data structure inherently makes an irreversible timeline of data when implemented in a decentralised nature. When a block is filled, it is set in stone and becomes a part of this timeline. Each block in the chain is given an exact timestamp when it is added to the chain.

How Does Blockchain Work

The goal of blockchain is to allow digital information to be recorded and distributed, but not edited. In this way, a blockchain is the foundation for immutable ledgers, or records of transactions that cannot be altered, deleted, or destroyed. Therefore, blockchains are also known as a distributed ledger technology (DLT).

Blockchain Decentralization

Imagine that a company owns a server farm with 10,000 computers used to maintain a database holding all its client's account information. This company owns a warehouse building that contains all these computers under one roof and has full control of each of these computers and all of the information contained within them. This, however, provides a single point of failure. What happens if the electricity at that location goes out? What if its Internet connection is severed? What if it burns to the ground? What if a bad actor erases everything with a single keystroke? In any case, the data is lost or corrupted.

What a blockchain does is to allow the data held in that database to be spread out among several network nodes at various locations. This not only creates redundancy but also maintains the fidelity of the data stored therein—if somebody tries to alter a record at one instance of the database, the

other nodes would not be altered and thus would prevent a bad actor from doing so. If one user tampers with Bitcoin's record of transactions, all other nodes would cross-reference each other and easily pinpoint the node with the incorrect information. This system helps to establish an exact and transparent order of events. This way, no single node within the network can alter information held within it.

Because of this, the information and history (such as of transactions of a cryptocurrency) are irreversible. Such a record could be a list of transactions (such as with a cryptocurrency), but it also is possible for a blockchain to hold a variety of other information like legal contracts, state identifications, or a company's product inventory.

Transparency

Because of the decentralised nature of Bitcoin's blockchain, all transactions can be transparently viewed by either having a personal node or using blockchain explorers that allow anyone to see transactions occurring live. Each node has its own copy of the chain that gets updated as fresh blocks are confirmed and added. This means that if you wanted to, you could track Bitcoin wherever it goes.

For example, exchanges have been hacked in the past, where those who kept Bitcoin on the exchange lost everything. While the hacker may be entirely anonymous, the Bitcoins that they extracted are easily traceable. If the Bitcoins stolen in some of these hacks were to be moved or spent somewhere, it would be known.

Of course, the records stored in the Bitcoin blockchain (as well as most others) are encrypted. This means that only the owner of a record can decrypts it to reveal their identity (using a public-private key pair). As a result, users of blockchains can remain anonymous while preserving transparency.

Is Blockchain Secure?

Blockchain technology achieves decentralised security and trust in several ways. To begin with, new blocks are always stored linearly and chronologically. That is, they are always added to the "end" of the blockchain. After a block has been added to the end of the blockchain, it is extremely difficult to go back and alter the contents of the block unless a majority of the network has reached a consensus to do so. That's because each block contains its own hash, along with the hash of the block before it, as well as the previously mentioned timestamp. Hash codes are created by a mathematical function that turns digital information into a string of numbers and letters. If that information is edited in any way, then the hash code changes as well.

Let's say that a hacker, who also runs a node on a blockchain network, wants to alter a blockchain and steal cryptocurrency from everyone else. If they were to alter their own single copy, it would no longer align with everyone else's copy. When everyone else cross-references their copies against each other, they would see this one copy stand out, and that hacker's version of the chain would be cast away as illegitimate.

Succeeding with such a hack would require that the hacker simultaneously control and alter 51% or more of the copies of the blockchain so that their new copy becomes the majority copy and, thus, the agreed-upon chain. Such an attack would also require an immense amount of money and resources, as they would need to redo all the blocks because they would now have different timestamps and hash codes.

Due to the size of many cryptocurrency networks and how fast they are growing, the cost to pull off such a feat probably would be insurmountable. This would be not only extremely expensive but also likely fruitless. Doing such a thing would not go unnoticed, as network members would see such drastic alterations to the blockchain. The network members would then hard fork off to a new version of the chain that has not been affected. This would cause the attacked version of the token to plummet in value, making the attack ultimately pointless, as the bad actor has control of a worthless asset. The same would occur if the bad actor were to attack the new fork of Bitcoin. It is built this way so that taking part in the network is far more economically incentivized than attacking it.

<u>Key Takeaways:</u>

- Blockchain is a type of shared database that differs from a typical database in the way that it stores information; blockchains store data in blocks that are then linked together via cryptography.
- As new data comes in, it is entered into a fresh block. Once the block is filled with data, it is chained onto the previous block, which makes the data chained together in chronological order.
- Different types of information can be stored on a blockchain, but the most common use so far has been as a ledger for transactions.
- In Bitcoin's case, blockchain is used in a decentralised way so that no single person or group has control—rather, all users collectively retain control.
- Decentralised blockchains are immutable, which means that the data entered is irreversible. For Bitcoin, this means that transactions are permanently recorded and viewable to anyone.

What is Ethereum?

Ethereum is a technology for building apps and organisations, holding assets, transacting and communicating without being controlled by a central authority. There is no need to hand over all your personal details to use Ethereum - you keep control of your own data and what is being shared. Ethereum has its own cryptocurrency, Ether, which is used to pay for certain activities on the Ethereum network.

What is the difference between Ethereum and Bitcoin?

Launched in 2015, Ethereum builds on Bitcoin's innovation, with some big differences. Both let you use digital money without payment providers or banks. But Ethereum is programmable, so you can also build and deploy decentralised applications on its network.

Ethereum being programmable means that you can build apps that use the blockchain to store data or control what your app can do. This results in a general purpose blockchain that can be programmed to do anything. As there is no limit to what Ethereum can do, it allows for great innovation to happen on the Ethereum network.

While Bitcoin is only a payment network, Ethereum is more like a marketplace of financial services, games, social networks, and other apps that respect your privacy and cannot censor you.

Aim of our project:

A blockchain is a type of distributed ledger technology where transactions are made visible and transparent via a shared digital ledger. Blockchain has long been hailed by futurists as a means of preventing fraud. Although blockchain technology is designed to be secure, there are two factors that contribute to its security flaws: the security features of the technology that supports it and the nature of its users.

With this prospect, the aim is to predict fraud on the Ethereum blockchain with high accuracy. This will assist in implementing measures for improved security.

THE DATA

Data Collection and Importance

1. https://www.kaggle.com/datasets/gescobero/ethereum-fraud-dataset
2. https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset

The two datasets utilised have been sourced from Kaggle. The datasets contain 34 and 51 features, respectively, which will be merged based on features that are present in both datasets. There are 22 000 entries in total.

The combined dataset resulted in 20 features, of which 19 are independent variables and only 1 dependent variable. The variable flag is our dependent variable and contains only two values, 0 (meaning no fraud detected) and 1 (meaning detection of fraud). Therefore, we shall test our data on multiple classification models to estimate/predict the probability of whether a transaction is fraud or not.

All the duplicates were removed from the combined dataset. This resulted in a total of 20 features including the dependent variable and 21862 entries in total.

Sources of Bias

What are biases in ML and AI:
Bias in data is an error that occurs when certain elements of a dataset are overweight or overrepresented. Biased datasets don't accurately represent ML model's use cases, which leads to skewed outcomes, systematic prejudice, and low accuracy.
Often, the erroneous result discriminates against a specific group or groups of people. For example, data bias reflects prejudice against age, race, culture, or sexual orientation. In a world where AI systems are increasingly used everywhere, the danger of bias lies in amplifying discrimination.

It takes a lot of training data for machine learning models to produce viable results. If you want to perform advanced operations (such as text, image, or video recognition), you need millions of data points. Poor or incomplete data as well as biased data collection & analysis methods will result in inaccurate predictions because the quality of the outputs is determined by the quality of the inputs.

Types of bias:

1. Selection bias.

In data science, selection bias occurs when you have data that aren’t properly randomized. If your dataset isn’t properly randomized, it means the sample isn’t representative - it doesn’t truly reflect the analysed population.

2. Overfitting and underfitting the data.

In machine learning, overfitting occurs when a model is trained with so much data that it begins to learn from the noise and inaccurate data entries in the data set. Machine learning models have trouble predicting new data based on the training data because this noise cannot be applied to new data. When a machine learning model fails to capture the underlying trend of the data (because it is too simple), underfitting occurs. In this case, it indicates that the model or algorithm is not fitting the data well enough.

The following bias was identified in our dataset. We notice a bias in our dataset where there are more samples with no fraud than those in which fraud was detected.

To fix the imbalance in the class we used SMOTE-NC to artificially generate data points. This will create a balance for the classification models to come up with better predictions and improve accuracy. We also used SMOTE-NC technique to make sure that there was no overfitting on our models.

Data Pre-processing

In the next steps we calculate and investigate the descriptive statistics, distribution of feature and prediction vectors. We also check the relationship between different features as well as the relationship between features and prediction vectors. We shall then check the health of data which include checking for missing values and skewness of data.
The visualisation technique used for our data is box plot. A box plot is a graphical summary of data that is based on a five number summary. A key to the development of a box plot is the computation of the median and the quartiles, Q1 and Q3. Box plots provide a way to identify outliers.

The data was checked for skewness to determine the distribution on all the features. The results showed that most features were positively skewed. Then data transformation must be done to train models for predictions.
The method chosen for transformation is box cox. A box cox transformation is a statistical technique that transforms a target variable so that your data closely resembles a normal distribution. Transforming our variables can improve the predictive power of our models because transformations can cut away white noise. The transformation was applied to all the independent variable features in the dataset.

The box cox function in the SciPy library is used for transformation calculations, this will choose the optimal value of lambda for the best approximation for the normal distribution. We found it to be the most effective to better the data distribution so that outliers have minimal impact on our model. We can see that we get less skewed data after applying box cox, with only 3 features that are still skewed compared to 18 skewed features before transforming.

Another common practice is to perform some type of scaling on numerical features. Applying scaling doesn't change the shape of each feature's distribution but ensures that each feature is treated equally when applying supervised learners. We dropped the categorical variable (address) and

standardised the numerical variables. The data now have the mean of 0 and the standard deviation of 1 and can be used for modelling.

DATA ANALYSIS AND EXPLORATION

Sanity Checks

Sanity checks are done to ensure that the developer has applied some form of rationality while creating the software. Sanity checks is a technique for testing software and doing a quick examination of the characteristics of the release to ensure that the same mistakes are not further repeated. Sanity testing or sanity checks is a subsection of regression tests that is focused on one or a few zones of functionality. The main criteria are to rule out the obvious fake results and not to catch every conceivable error.

General Sanity Checking Steps

1. Take a random sample of the data.
2. Check for data type mismatches, variations in how values are entered, and missing values.

   Effective downstream analysis requires consistency. You can't easily understand the relationships between events if some of those events have dates formatted as "yyyy/MM/dd" and some events have dates formatted as "dd/MM/yyyy". Similarly, some systems may store prices in strings ($1,000) while some systems store prices in decimals (1000.00).**3.**

3. Look for duplicate records and outliers.

   Both duplicate records and outliers can distort your analysis, so you often need to assess the overall quality of the dataset and determine if it contains any duplicate records or outliers. Duplicate records can also be problematic for users who construct dashboards that combine similar data from multiple source systems. Making sure the data used in the dashboard is crucial, especially since any inaccurate or duplicated data in the final dashboard could have a direct impact on decisions. By performing a duplicate records sanity check at the beginning of the data wrangling process, you will be able to discover if your source systems contain repeated primary keys and take remedial action to remove those duplicates.

4. Assess the data distribution for each column.

   Sometimes the data in a column may look good on the surface—there are no obvious data type mismatches, duplicate records, outliers, or null values—but if you examine the distribution of data in the column, you notice gaps or a values distribution that doesn't make logical sense. An odd data distribution may indicate a larger data quality issue that you need to investigate maybe some data points were not recorded, or the data was not encoded correctly.

Sanity checks used in our project include:

1. Taking random samples of data:

   What does this mean?

   i. Often, the datasets that you're working with are too large to easily assess as a complete source. Looking only at a consecutive sample of that data. By generating a random sample over the entirety of the dataset, you get a more accurate picture of the full dataset.

   ii. At any point when you are developing, you can collect a new sample. Random samples can be useful when you are working with unknown or new data sources, and you want to confirm that the data remains consistent throughout the dataset.

   iii. Examining a random sample of our data during a sanity test allowed us to surface this potential data quality issue and then take steps to address it.

   How did we perform this?

   iv. We printed out two separate but similar datasets using data_1.head() and data_2.head()

```
data_1.head()
```

	address	flag	minTimeBetweenSentTnx	maxTimeBetweenSentTnx	avgTimeBetweenSentTnx	minTimeBetweenRecTnx	maxTimeBetweenRecTr
0	0xd0cc2b24980cbcca47ef755da88b220a82291407	1	0.0	2387389.0	5.807655e+04	89.0	1.501076e+
1	0x4cdc1cba0aeb5539f2e0ba158281e67e0e54a9b1	1	0.0	0.0	0.000000e+00	3021091.0	1.502802e+
2	0x00e01a648ff41346cdeb873182383333d2184dd1	1	37.0	25112882.0	1.710279e+06	0.0	1.500397e+
3	0x858457daa7e087ad74cdeeceab8419079bc2ca03	1	0.0	642460.0	1.576106e+04	0.0	1.500037e+
4	0x240e125c20a4cc84bd6e7f8d1fd07aff4c06d43d	1	0.0	0.0	0.000000e+00	3894.0	1.500402e+

5 rows × 34 columns

```
data_2.head()
```

	Unnamed: 0	Index	Address	FLAG	Avg min between sent tnx	Avg min between received tnx	Time Diff between first and last (Mins)	Sent tnx	Received Tnx	Number of Created Contracts	...	ERC20 min val sent	ERC20 max val sent	ERC20 avg va sen
0	0	1	0x0000927775ac7d0d59eaad8fee3d10ac6c805e8	0	844.26	1093.71	704785.63	721	89	0	...	0.000000	1.683100e+07	271779.92000
1	1	2	0x0002b44ddb1476db43c868bd494422ee4c136fed	0	12709.07	2958.44	1218216.73	94	8	0	...	2.260809	2.260809e+00	2.26080
2	2	3	0x0002bda54cb772d040f779e88eb453cac0daa244	0	246194.54	2434.02	516729.30	2	10	0	...	0.000000	0.000000e+00	0.00000
3	3	4	0x00038e6ba2fd5c09aedb96697c8d7b8fa6632e5e	0	10219.60	15785.09	397555.90	25	9	0	...	100.000000	9.029231e+03	3804.07689
4	4	5	0x00062d1dd1afb6fb02540ddad9cdebfe568e0d89	0	36.61	10707.77	382472.42	4598	20	1	...	0.000000	4.500000e+04	13726.65922

5 rows × 51 columns

2. Datatypes, completeness Information

   Using the Pandas "info" function, in addition to the data-type information for the dataset, we can look at counts of available records/missing records.

```
df.info()
```

```
Int64Index: 21987 entries, 0 to 9840
Data columns (total 20 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   address                   21987 non-null  object 
 1   flag                      21987 non-null  int64  
 2   avgTimeBetweenSentTnx     21987 non-null  float64
 3   avgTimeBetweenRecTnx      21987 non-null  float64
 4   sentTransactions          21987 non-null  int64  
 5   receivedTransactions      21987 non-null  int64  
 6   createdContracts          21987 non-null  int64  
 7   numUniqSentAddress        21987 non-null  int64  
 8   numUniqRecAddress         21987 non-null  int64  
 9   minValSent                21987 non-null  float64
 10  maxValSent                21987 non-null  float64
 11  avgValSent                21987 non-null  float64
 12  minValReceived            21987 non-null  float64
 13  maxValReceived            21987 non-null  float64
 14  avgValReceived            21987 non-null  float64
 15  totalTransactions         21987 non-null  int64  
 16  totalEtherSent            21987 non-null  float64
 17  totalEtherReceived        21987 non-null  float64
 18  totalEtherSentContracts   21987 non-null  float64
 19  totalEtherBalance         21987 non-null  float64
dtypes: float64(12), int64(7), object(1)
memory usage: 3.5+ MB
```

```
df.isnull().sum() # LOOK IF THERE IS ANY MISSING VALUES
```

```
address                   0
flag                      0
avgTimeBetweenSentTnx     0
avgTimeBetweenRecTnx      0
sentTransactions          0
receivedTransactions      0
createdContracts          0
numUniqSentAddress        0
numUniqRecAddress         0
minValSent                0
maxValSent                0
avgValSent                0
minValReceived            0
maxValReceived            0
avgValReceived            0
totalTransactions         0
totalEtherSent            0
totalEtherReceived        0
totalEtherSentContracts   0
totalEtherBalance         0
dtype: int64
```

3. Descriptive Statistics
    a. Using df.describe() we are able to gain a deeper understanding of the data and take more specific analysis of our data.

	flag	avgTimeBetweenSentTnx	avgTimeBetweenRecTnx	sentTransactions	receivedTransactions	createdContracts	numUniqSentAddress	numUniqRecAddress	minV
count	21987.000000	2.198700e+04	2.198700e+04	21987.000000	21987.000000	21987.000000	21987.000000	21987.000000	21987.(
mean	0.333333	2.017807e+06	8.849104e+07	337.691409	448.022786	2.641788	66.613408	167.290535	5.(
std	0.471415	3.068441e+07	1.905694e+08	1496.171520	1807.664973	122.922835	496.873942	868.424851	206.1
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.(
25%	0.000000	0.000000e+00	1.020000e+00	1.000000	2.000000	0.000000	1.000000	1.000000	0.(
50%	0.000000	2.171600e+02	1.033698e+04	3.000000	8.000000	0.000000	2.000000	3.000000	0.(
75%	1.000000	3.269873e+04	6.604927e+07	32.000000	42.000000	0.000000	8.000000	10.000000	0.(
max	1.000000	8.291299e+08	8.309932e+08	10000.000000	10000.000000	9995.000000	9950.000000	9999.000000	25533.(

b. From the above description of our data, we can also make a guess at the skewness of the data at this stage by looking at the difference between the means and medians of numerical features

c. Using df.duplicated().sum() and df.drop_duplicates(inplace=**True**) we can look for and remove any duplicates in our dataset

Sanity testing is a form of validating data during the development process, it helps in quickly identifying the defects in the core functionality and can be carried out in less time, in a cost-efficient way. No documentation is required and if any defects are found in the project it is rejected which in turn helps to save time and efforts for the execution of regression tests.

Sanity tests also give quick status about the release and plan the next tasks accordingly. After reading the above article we hope that you have understood sanity check meaning and have a clear idea about sanity check software. It might be easy to overlook sanity checks but doing so can have a real impact on your data. Without taking care to understand the complete contents of your data, checking for mismatched or duplicate data, and assessing the distribution across each column, you'll end up with a skewed analysis—which might be more difficult to untangle than running a sanity test in the first place.

Training, validation, and testing data

Using validation and test sets is very important as validation and test sets will improve the model's capacity to generalise to fresh, untested data. The data was split into training, validation, and testing subsets. 75% of the data was used as the training set, 15% for the validation set and for the testing set 10% of the data was utilised.

The stratify parameter allows us to keep the same number of classes in the train and test sets as there are in the full original dataset. The y labels were stratified due to the imbalance in the dataset. An imbalanced dataset refers to data samples from one or more classes that significantly outnumber the samples from the rest of the classes in the dataset. The stratify parameter was used to make sure that the imbalance class is split equally. We wanted to preserve the dataset proportions for better prediction and reproducibility of results.

Synthetic Minority Over-sampling Technique for Nominal and Continuous (SMOTE-NC) is a great tool to generate synthetic data to oversample a minority class in an imbalanced dataset. We used SMOTE-NC to artificially generate data points for the training set to fix the imbalance noticed earlier.

Modelling

Initially we intended to use just one model which was Logistic Regression, but as we explored with our data, we finalised on using 7 different models: Logistic Regression, K-Nearest-Neighbours, Naïve Bayes, Support Vector Machines, Decision Trees, XGBoost and Random Forest Tree.

Results of Models

The evaluation metrics used were:

- *Precision*: indicates out of all positive predictions, how many are actually positive. It is defined as a ratio of correct positive predictions to overall positive predictions.
- ***Recall***: indicates out of all actually positive values, how many are predicted positive. It is a ratio of correct positive predictions to the overall number of positive instances in the dataset.
- *F1-Score*: is defined as the harmonic mean of precision and recall
- *Accuracy*: defined as the percentage of correct predictions made by our classification model.

For our problem, we are trying to detect if a transaction is fraudulent or not, therefore we will focus on the recall, since in a real-world scenario it would be safer to have a non-fraudulent transaction be classified as fraudulent then a fraudulent transaction be classified as non-fraudulent.

Table with the models trained and their accuracy and recall for both the positive and negative class:

Models	**Recall [0]**	**Recall [1]**	**Accuracy**
Logistic Regression	82%	77%	81%
KNN	89%	86%	88%
Naïve Bayes	19%	96%	45%
SVM	87%	87%	87%

Decision Tree	91%	86%	89%
XGBoost	95%	90%	93%
Random Forest Tree	94%	88%	92%

Top model Hyperparameter Tuning

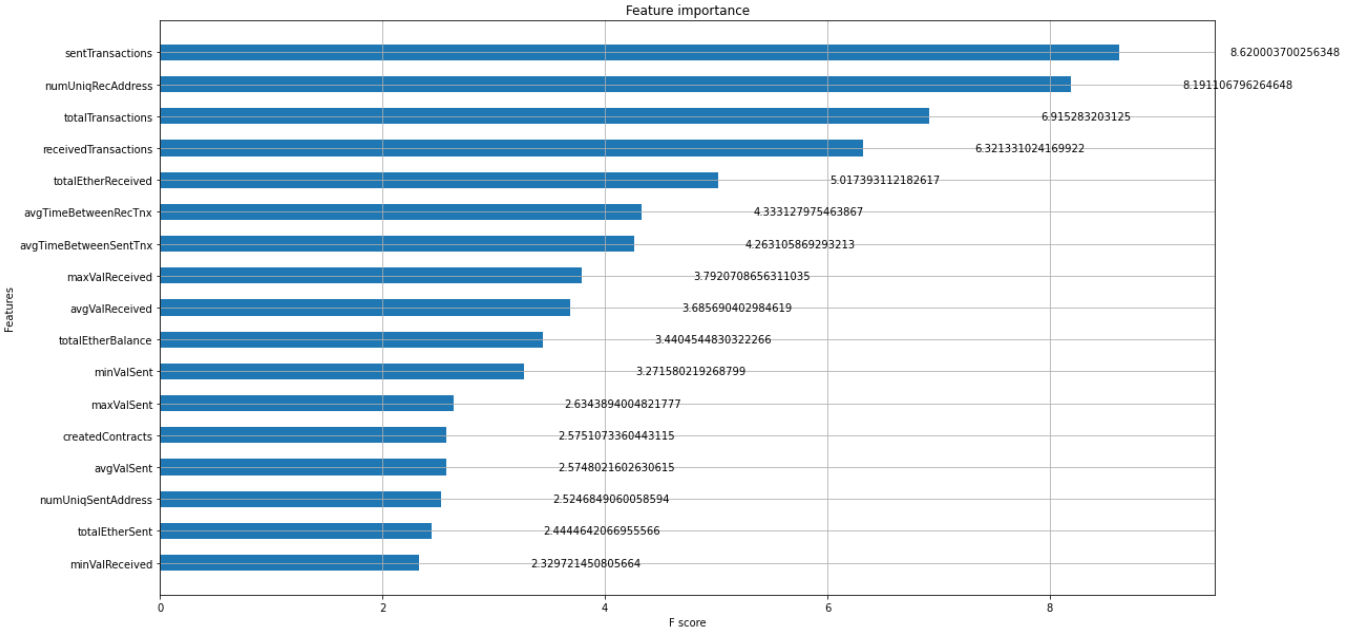
As noticed in the table of results, our top performing model was XGBoost in terms of recall and accuracy. To further our results, we tune our parameters for the XGBoost model. We used GridSearchCV which runs through all the different parameters that are fed into the parameter grid and produces the best combination of parameters, based on a scoring metric of your choice (accuracy, f1, etc). After tuning we achieved an accuracy of **95%.** The list of hyperparameters is below:

Hyperparameter	**Optimal value**
learning_rate	0.2
max_depth	15
min_child_weight	1
gamma	0.2
colsample_bytree	0.5

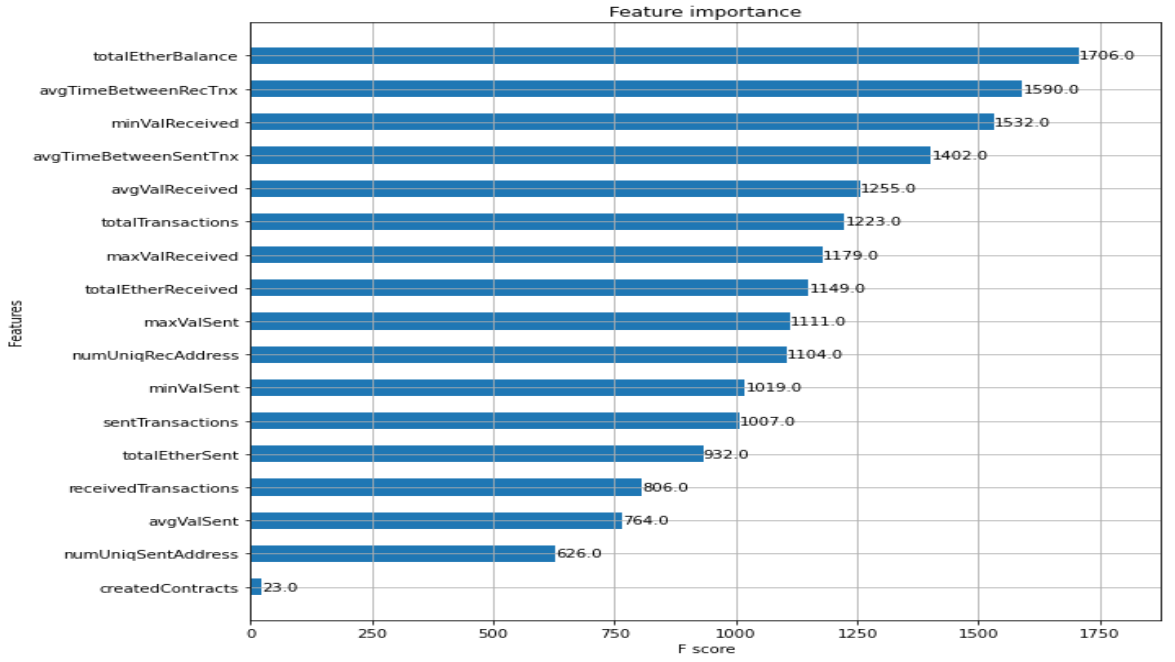
Feature Importance and Selection

Xgboost has multiple types of feature importance options that can be computed in various ways. If you design the model using an API similar to Scikit-Learn, the default type is called *gain*. There are three methods to measure feature_importances in xgboost. They are:

- weight: The total number of times this feature was used to split the data across all trees.
- Cover: The number of times a feature is used to split the data across all trees weighted by the number of training data points that go through those splits.
- Gain: The average loss reduction gained when using this feature for splitting in trees.



We used *gain* in the project and the model says when it used sentTransactions, the loss on average was reduced by 8.6%, see image below.



When we consider *weight*,the model says that it is used totalEtherBalance 1706 times to split the data across the trees and was viewed as most important.

Scikit-learn allows for feature selection using feature importance scores. The SelectFromModel class is used for this, which can take a model and turn a dataset into a subset with chosen features. This class is capable of accepting a trained model, such as one that was applied to the full training set. It can then choose which features to select using a threshold. This threshold is used to consistently

choose the same features on the training dataset and the test dataset when the transform() function is called on the SelectFromModel instance.In the notebook we first train and then evaluate an XGBoost model on the entire training dataset.

We then wrap the model in a SelectFromModel instance using the feature importances calculated from the training dataset. Using this, we choose features from the training dataset, train a model using the chosen subset of features, and then assess the model on the test dataset using the same feature selection method.We then experimented with various thresholds for classifying features according to their relevance. In particular, the feature importance of each input variable enables us to test every feature subset in turn, starting with the most significant feature and working our way down to the least important feature.

Below were our results.

```
Thresh=0.000, n=18, Accuracy: 93.42%
Thresh=0.032, n=17, Accuracy: 93.42%
Thresh=0.034, n=16, Accuracy: 93.46%
Thresh=0.035, n=15, Accuracy: 93.37%
Thresh=0.035, n=14, Accuracy: 93.05%
Thresh=0.035, n=13, Accuracy: 93.37%
Thresh=0.036, n=12, Accuracy: 92.59%
Thresh=0.045, n=11, Accuracy: 93.28%
Thresh=0.047, n=10, Accuracy: 92.55%
Thresh=0.051, n=9, Accuracy: 91.18%
Thresh=0.052, n=8, Accuracy: 90.81%
Thresh=0.058, n=7, Accuracy: 91.40%
Thresh=0.059, n=6, Accuracy: 90.53%
Thresh=0.069, n=5, Accuracy: 86.51%
Thresh=0.087, n=4, Accuracy: 83.31%
Thresh=0.095, n=3, Accuracy: 82.90%
Thresh=0.112, n=2, Accuracy: 77.37%
Thresh=0.118, n=1, Accuracy: 63.05%
```

We can see that the performance of the model generally decreases with the number of selected features. On this problem there is a trade-off of features to test set accuracy and we could decide to take a less complex model (fewer attributes such as n=12) and accept a modest decrease in estimated accuracy from 93.42 down to 92.59%. This is likely to be a wash on such a small dataset but may be a more useful strategy on a larger dataset and using cross validation as the model evaluation scheme.

LIMITATIONS

Here are some limitations we faced:

- The amount of data - We were only able to find two datasets which had meaningful and useful data for this project. We had around 22 000 entries after combining our datasets, since our data was biased towards one class, we were able to increase the number of entries using SMOTE.
- Data from different sources - since both datasets were from different sources there were features that were not present in both datasets and so we had to remove them.
- Knowledge - we would've liked to use complex algorithms like anomaly or genetic algorithms but were limited by time and the ability to learn and understand these algorithms.
- Computing power - we originally wanted to do hyperparameter tuning on the top 3 models in our notebook, but this took way too long, and we didn't have the computing resources to speed up this process. Therefore, we only included hyperparameter tuning on our best model.

CONCLUSIONS

Credit card fraud is a major problem for businesses. These schemes can result in significant losses for both businesses and individuals. As a result, businesses spend an increasing amount of money researching and creating innovative strategies that will aid in the detection and avoidance of fraud. This project's main objective was to compare various data analytic and machine learning methods to detect transaction fraud. The XGBoost algorithm, which best classifies whether transactions are fraudulent or not, was shown to have the best results after comparison.

A variety of evaluation metrics, including recall, accuracy, and precision, were used to establish this. Recall with a high value is crucial for this kind of problem. Achieving substantial results and performance has proved to depend greatly on feature selection and dataset balancing. Further research on this problem could involve testing with anomaly detection or genetic algorithms, and the use of several stacked classifier types, in addition to extensive feature selection, can get superior outcomes.

REFERENCES

4 key steps to sanity checking your data (2022) *Trifacta*. Available at: https://www.trifacta.com/blog/4-key-steps-to-sanity-checking-your-data/.

8 types of data bias that can wreck your machine learning models (n.d) *Statice*. Available at: https://www.statice.ai/post/data-bias-types#:~:text=What%20are%20biases%20in%20machine%20learning%20and%20artificial%20intelligence%3F,systematic%20prejudice%2C%20and%20low%20accuracy.

Aguilar, F. (2019) *Smote-NC in ML categorization models FO imbalanced datasets*, *Medium*. Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/smote-nc-in-ml-categorization-models-fo-imbalanced-datasets-8adbdcf08c25.

Box-Cox Transformation and target variable: Explained (n.d) *BuiltIn*. Available at: https://builtin.com/data-science/box-cox-transformation-target-variable.

Carlos A. Jimenez HolmquistCarlos A. Jimenez Holmquist 24511 gold badge33 silver badges66 bronze badges *et al.* (1966) *Meaning of stratify parameter*, *Data Science Stack Exchange*. Available at: https://datascience.stackexchange.com/questions/40584/meaning-of-stratify-parameter#:~:text=The%20stratify%20parameter%20asks%20whether,stratify%20%3D%20True%20%2C%20with%20a%20.

Hayes, A. (2022) *Blockchain facts: What is it, how it works, and how it can be used*, *Investopedia*. Investopedia. Available at: https://www.investopedia.com/terms/b/blockchain.asp.

Pramoditha, R. (n.d) *Why do we need a validation set in addition to training and test sets ...*, *Why Do We Need a Validation Set in Addition to Training and Test Sets?* Available at: https://towardsdatascience.com/why-do-we-need-a-validation-set-in-addition-to-training-and-test-sets-5cf4a65550e0.

Ray, J. (2021) *Data sanity checks for model development*, *Medium*. Better ML. Available at: https://medium.com/better-ml/data-sanity-checks-for-model-development-c6beb5b0c8ce.

Remove duplicate rows based on a primary key (n.d) *TRIFACTA community*. Available at: https://community.trifacta.com/s/article/Remove-duplicate-rows-based-on-a-primary-key.

Sanity checks: A 6 step easy guide for beginners (2022) *Jigsaw Academy*. Available at: https://www.jigsawacademy.com/blogs/business-analytics/sanity-checks/.

What is Ethereum? (n.d) *ethereum.org*. Available at: https://ethereum.org/en/what-is-ethereum/.