Bitcoin Mining Pool Classifier

Saanika Gupta*
Department of Computer Science and Engineering
Dr. Shyama Prasad Mukherjee International Institute of Information Technology, Naya Raipur (IIIT-NR)

The report is structured as follows: In Section 1, I have cited the different libraries used along with their version. In Section 2, I have given a brief introduction about the Bitcoin network and mining, and the task that my model will perform. In Section 3, I have presented an overview of how to retrieve the data and model architecture of the implemented models for the task. Section 4 demonstrates different methods and parameters tried which didn't result in optimal results. In Section 5, a brief comparison between artificial neural network (ANN) and random forest (RF) for this task is done and also demonstrated the importance of each feature and the change in accuracy with feature reduction. In Section 6, results are shown and conclusion is drawn in Section 7.

1 Libraries Used

- BigQuery 1.12.1 (1)
- Keras 2.2.4 (2)
- Tensorflow 1.14.0 (3)
- Scikit-Learn 0.21.3 (4)
- Matplotlib 3.0.3 (5)
- NumPy 1.17.0 (6)
- Pandas 0.23.4 (7)

2 Introduction

Bitcoin is a decentralized digital currency (cryptocurrency) which leverages the Blockchain to store transactions in a distributed manner in order to mitigate against flaws in the financial industry. A transaction is a transfer of the value of Bitcoin that is broadcasted to the network and collected into blocks. A transaction typically references the previous transaction outputs as new transaction inputs and dedicates all input Bitcoin values to new outputs. Transactions are not encrypted, so one can browse and view any transaction that has ever been collected in a block. A coinbase transaction or generation transaction is a special type of bitcoin transaction that can only be made by a miner. This sort of transaction has no inputs, and one is created with each fresh block that is being mined on the network. In other words, this is a transaction that rewards a miner with a block reward for his job. Block reward is the reward that miner gets for successfully mining a block.

A block is a container data structure. In the Bitcoin world, a block currently contains $1,903$ transactions on an average (8). The average size of a block currently 0.93 MB (9). If the size of a block is increased, it would enable more transactions to be processed per second.

The mining of Bitcoin is what makes it possible for the blockchain to be decentralized and secure. By verifying transactions and preventing the network from being hijacked, miners do the work of making sure that Bitcoin is able to function as a plausible store of value at all. There are now countless cryptocurrencies in circulation, and miners can choose to mine virtually any of them. And if enough

*Third Year Undergraduate Student in CSE Branch

of them decide to start mining something different because some other blockchain is more profitable to mine, then it could take a very long time for transactions to be added to the blockchain and validated, which would end up making BTC impractical to use. As more users are actively transacting with cryptocurrencies, its value (and, therefore, its price) increases, hence both the mining fees and mining rewards become more valuable. This keeps miners sticking around rather than leaving and posing an existential threat to Bitcoin.

In this approach, my model will predict that whether the block is mined by an individual miner or a mining pool from the retrieved subset of the real-time dataset.

3 Methodology

3.1 Loading data from BigQuery

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 30 columns):
is_miner                    10000 non-null bool
address                     10000 non-null object
output_month_min            10000 non-null int64
output_month_max            10000 non-null int64
input_month_min             10000 non-null int64
input_month_max             10000 non-null int64
output_active_time          10000 non-null int64
input_active_time           10000 non-null int64
io_max_lag                  10000 non-null int64
io_min_lag                  10000 non-null int64
output_active_months        10000 non-null int64
total_tx_output_count       10000 non-null int64
total_tx_output_value       10000 non-null object
mean_tx_output_value        10000 non-null object
stddev_tx_output_value      10000 non-null float64
total_output_tx             10000 non-null int64
mean_monthly_output_value   10000 non-null object
mean_monthly_output_count   10000 non-null float64
input_active_months         10000 non-null int64
total_tx_input_count        10000 non-null int64
total_tx_input_value        10000 non-null object
mean_tx_input_value         10000 non-null object
stddev_tx_input_value       10000 non-null float64
total_input_tx              10000 non-null int64
mean_monthly_input_value    10000 non-null object
mean_monthly_input_count    10000 non-null float64
mean_output_idle_time       10000 non-null float64
stddev_output_idle_time     8087 non-null float64
mean_input_idle_time        10000 non-null float64
stddev_input_idle_time      7629 non-null float64
dtypes: bool(1), float64(8), int64(14), object(7)
memory usage: 2.2+ MB
```

Figure 1: DataFrame Overview

I have used the Bitcoin Blockchain dataset (10), which gives access to information about blockchain blocks and transactions. All historical data are in the 'bigquery-public-data:crypto_bitcoin dataset'. This dataset is updated every 10 minutes and contains four tables, namely 'blocks', 'inputs', 'outputs', and 'transactions'. I have used the BigQuery Python client library to query tables in this dataset in Kaggle Kernels. A subset of the whole dataset is retrieved keeping in mind the Kaggle's resource constraints. We know that the miner or mining pool can mine a block only if that miner/pool choose to identify themselves. They do this by inserting their name or other recognizable signature in the block's coinbase transaction (coinbase_param), which is allowed to contain arbitrary data. By checking the value of 'is_coinbase' column (True or False) and identifying the signature of the mining pool (signatures obtained from (11)) by using the SQL LIKE operator to find the pattern in the 'coinbase_param' column, the rows are grouped as either 'True' or 'False' (limit for each class is explicitly set to 5000 making sure that the dataset is balanced) for the created alias 'is_miner'. The

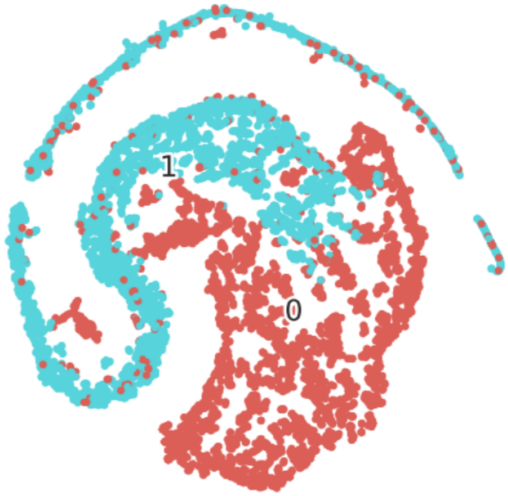

Figure 2: t-SNE plot of features

'is_miner' column is selected as the target column for modeling. The table rows were retrieved as pandas DataFrame. Overview of the retrieved dataset is provided in figure 1. To analyze further, the columns with null values and 'non-numeric' features were removed. The remaining twenty-six columns are the features fed as input to the model. A visualization of the features in 2D space using t-SNE (12), a dimensionality reduction method is shown in figure 2.

3.2 Artificial Neural Network

```
Index(['output_month_min', 'output_month_max', 'input_month_min',
       'input_month_max', 'output_active_time', 'input_active_time',
       'io_max_lag', 'io_min_lag', 'output_active_months',
       'total_tx_output_count', 'total_tx_output_value',
       'mean_tx_output_value', 'stddev_tx_output_value', 'total_output_tx',
       'mean_monthly_output_value', 'mean_monthly_output_count',
       'input_active_months', 'total_tx_input_count', 'total_tx_input_value',
       'mean_tx_input_value', 'stddev_tx_input_value', 'total_input_tx',
       'mean_monthly_input_value', 'mean_monthly_input_count',
       'mean_output_idle_time', 'mean_input_idle_time'],
      dtype='object')
```

Figure 3: Input Features

```
ann.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 26)                702
_________________________________________________________________
dense_2 (Dense)              (None, 11)                297
_________________________________________________________________
dropout_1 (Dropout)          (None, 11)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 72
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 14
=================================================================
Total params: 1,085
Trainable params: 1,085
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Artificial Neural Network Architecture

Table 1: Parameters of ANN

Model Parameters	
Optimiser	RMSProp
Loss	Binary Cross Entropy
Training Parameters	
Batch Size	500
Learning Rate	0.001
Epochs	270

Simple Feedforward Neural Network with two hidden layers is implemented for classifying 'mining pools' and 'individual miners' (not a mining pool). In total, twenty-six features (normalized) were fed as input to the artificial neural network as shown in figure 3. RMSprop optimizer and binary_crossentropy loss are used as the model parameters (mentioned in Table 1). Softmax function is used in the final layer and tanh activation is used in rest of the layers. One dropout layer is added between the two hidden layers. The model architecture is depicted in figure 4.

3.3 Random Forest Classifier

Table 2: Parameters of Random Forest

Hyperparameters	
n_estimators	100
min_samples_leaf	1
Model Parameters	
class_weight	Balanced

Random forest is an ensemble of decision trees. It fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy. A single decision tree is very sensitive to data variations and prone to overfitting. It can easily overfit to noise in the data. In this random forest implementation, classification between 'mining pools' and 'individual miners' (not a mining pool) Hence, random forest is more powerful as it overcomes this problem. In total, twenty-six features (non-normalized) were fed as input to the random forest model as shown in figure 3. In this random forest implementation, 100 trees are considered and 'class_weight' parameter set to 'balanced' as shown in Table 2. Rest of the parameters are set to default.

4 Different ideas tried

- Naive Bayes classifier is also implemented on the non-normalized dataset but it is giving only 91.35% accuracy.
- Different number of hidden layers and different hidden layer sizes were tried but weren't optimal. Non-normalized dataset was giving poor results for ANN.
- Different number of estimators and maximum depth for the random forest classifier were tried but weren't optimal.

5 Experiments

Out of the twenty-six features, top 3 features (according to the feature importance of the forest as shown in figure (5) are the most significant. This is also demonstrated in the figure (5) where one by one, the least significant feature according to feature importances of the forest is removed, and at point A, the slope suddenly decreased which means with every feature dropped from the input set, the change in the model accuracy is large and negative as shown in figure (6). We can see, that both random forest and artificial neural network perform similar but random forest marginally outperformed artificial neural network. Another advantage of random forest is it didn't require normalized inputs unlike artificial neural network. The plotted accuracies of both the models (RF and ANN) for different set of inputs is mentioned in Table 4.

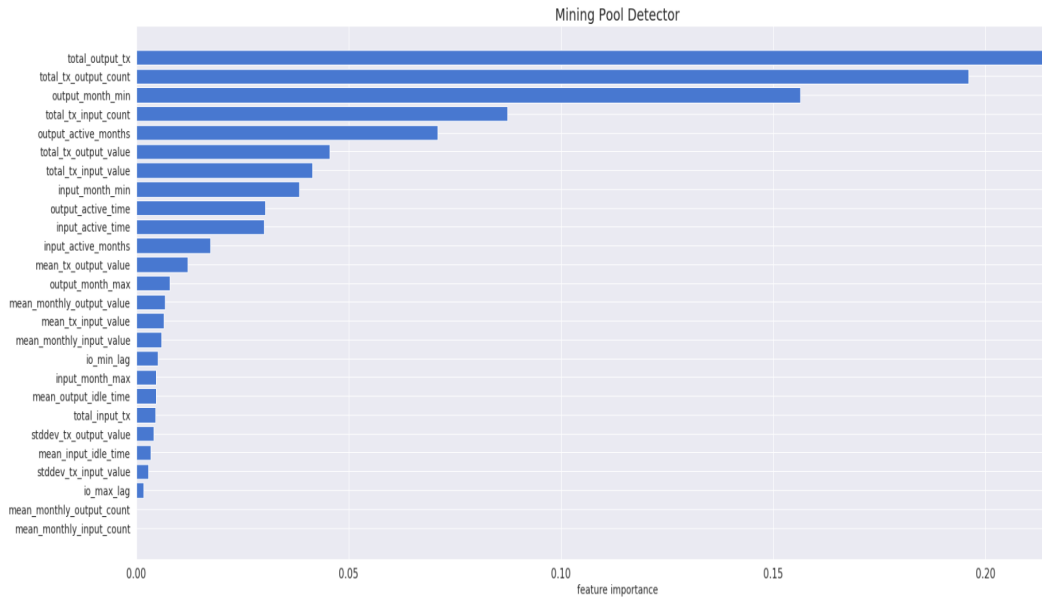

Figure 5: Feature Importance of the Forest

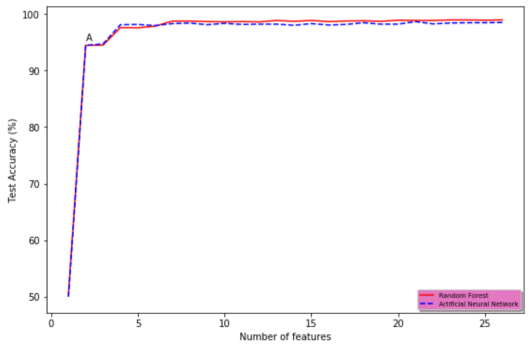

Figure 6: Random Forest vs Artificial Neural Networks Accuracy

6 Result Analysis

Confusion matrices of Random Forest Classifier, Artificial Neural Network, and Naive Bayes Classifier are shown in figure 7.

Precision and recall derived from the confusion matrix is shown in Table 5. From the confusion matrix, we can tell that for all the models (ANN, RF and Naive Bayes), prediction of class label '0' has lower accuracy than prediction of class label '1'. This is due to the fact that given a block, it is harder to predict that whether it is mined by an 'individual miner' or not. From Table 5 we can see that precision value of Random Forest is the highest whereas recall value of ANN and Naive Bayes is equal and more than that of Random Forest. In fact, there RF is only 0.001% lower than the others for recall which makes recall a poor choice as a factor to choose between the methods.

Among all the experiments conducted, the best results were obtained from the random forest classifier. The test accuracy of the three models (RF, ANN and Naive Bayes) for all the twenty-six input features is mentioned in Table 4.

Table 3: Random Forest vs Artificial Neural Networks Accuracy

Feature Dimension	Random Forest Accuracy (%)	ANN Accuracy (%)
26	98.5	**98.95**
25	98.45	98.9
24	98.45	98.95
23	98.4	98.95
22	98.25	98.85
21	**98.65**	98.85
20	98.2	98.9
19	98.2	98.7
18	98.45	98.8
17	98.15	98.75
16	98.05	98.65
15	98.3	98.85
14	98.0	98.7
13	98.2	98.85
12	98.2	98.6
11	98.15	98.65
10	98.35	98.6
9	98.1	98.65
8	98.4	98.75
7	98.3	98.74
6	97.95	97.85
5	98.15	97.55
4	98.1	97.6
3	94.7	94.5
2	94.45	94.45
1	49.65	50.35

Table 4: Accuracy Comparison between all the Implemented Models

Model	Best Accuracy (%)
Random Forest Classifier	**98.95**
Artificial Neural Network	98.65
Naive Bayes Classifier	91.35

7 Conclusion

I, first visualised the data points using the t-SNE plot. From the 2D visualisation of the 26 features, I saw that the data has high variance but indeed separable. The choice of shallow ANN over deep ANN is due to the intrinsic fact that deeper ANN classifies the dataset with high bias efficiently, but shallower ANN proves to be better for the dataset with high variance. From the above, result analysis, I see that Random Forest and ANN perform very good for the given dataset retrieved in the real time, however RF proves to be marginally better than ANN due to higher accuracy and precision. Recall cannot be a measure to choose between the methods itself because all the three methods have more or less same recall. In the real-life environment, a model should be accurate enough and precise enough. This trait of RF gives an edge over the shallow ANN. Moreover, I understand that as RF doesn't require any preprocessing as ANN does (Normalisation), this advantage of computation time also helps RF to faster than the ANN counterpart in the real life deployment of the models, where the data which is processed is in huge number.

Table 5: Precision and Recall for all the Implemented Models

Model	Precision	Recall
Random Forest Classifier	**0.984**	0.993
Artificial Neural Network	0.974	0.994
Naive Bayes Classifier	0.877	0.994

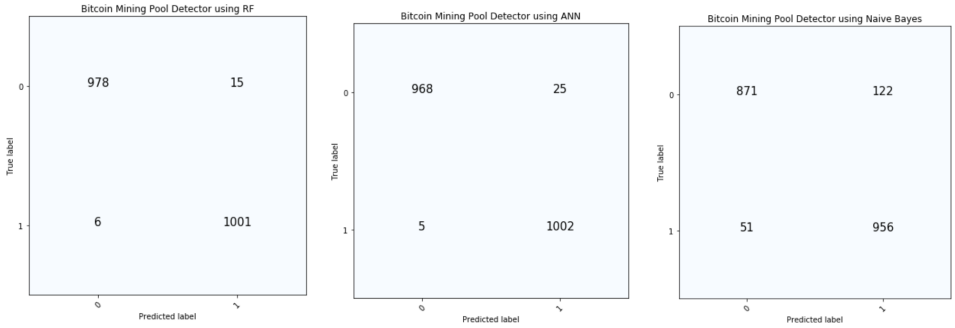

Figure 7: Confusion matrices of different methods. Note:Here, '0' means 'not a mining pool' and '1' means 'mining pool'.

References

[1] J. Tigani and S. Naidu, *Google BigQuery Analytics*. John Wiley & Sons, 2014.

[2] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, p. 90, 2007.

[6] T. Oliphant, "NumPy: A guide to NumPy," USA: Trelgol Publishing, 2006–, [Online; accessed <today>]. [Online]. Available: http://www.numpy.org/

[7] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51 – 56.

[8] "Average number of transactions per block," https://www.blockchain.com/charts/n-transactions-per-block, accessed: 2019-09-25.

[9] "Average block size," https://www.blockchain.com/charts/avg-block-size, accessed: 2019-09-25.

[10] "Bitcoin blockchain (complete live historical bitcoin blockchain data (bigquery))," https://www.kaggle.com/bigquery/bitcoin-blockchain, accessed: 2019-09-15.

[11] "Comparison of mining pools," https://en.bitcoin.it/wiki/Comparison_of_mining_pools, accessed: 2019-09-18.

[12] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html