Predicting the Hidden Costs Behind Ethereum Transactions

Jack Studiner-20164574, Saqib Ahmad-20047232, Chris Kingsland-20184320, Leo Toueg-20184320

Abstract—Cryptocurrency trades are decentralized. There is no central authority that oversees trades that take place within the blockchain. The record of all transactions is publicly stored and distributed by participating nodes. As such, when you transact a cryptocurrency such as Ethereum, it only involves you and the blockchain which verifies your transactions. There are distributed protocols that ensure the legitimacy of every transaction before being confirmed by the network, that involve dedicating hardware to solving complex computational problems. As a result, the confirmation of transactions requires immense computational power. However, the blockchain does not do this verification for free, it charges a fee to perform the verification. This is to cover energy costs for processing and validating transactions, as well as for the monetary incentives provided by the blockchain to block validators for dedicating their processing power. These fees can sometimes be very large, ranging from 10's to 1000's of USD, depending on the size of the transaction. These fees are called "Gas fees" and the price of "Gas" fluctuates based on many factors. People who trade cryptocurrencies frequently want to know when the price of gas will be lowest, so they can execute their transaction at a time which results in the most optimal transaction fees. Through our project, our team will look to help traders time their trades so they can pay a smaller gas fee, using machine learning techniques learned in CISC 351.

Index Terms—Ethereum, gas fee, regression,modelling

1 Introduction

Completing a transaction on the Ethereum blockchain is a 2 step process. First the sender must post the transaction to the blockchain. Then miners on the blockchain must validate the transaction. To encourage more verification nodes on the network, miners are given a rewards for every transaction they verify. This reward comes from the gas fees paid by the user initiating the transfer. The gas price varies every day. Consequently, it is the best interest of the person initiating the transaction to only post their transactions when the gas price is low.

The purpose of this paper is to build a regression model which accurately estimates the ether gas price for a given day. We approach this problem by collecting different variables relating to the Ethereum network, determining which variables are best at predicting gas fees, and testing various regression model to determine the optimal method to predict Ethereum gas price.

Our main contributions are that binning does not work well with daily Ethereum data. The reason being that there are only 2000 days since the Ethereum blockchain has been active. Binning reduces the information in the data too much. The best method to predict gas fees is the random forest regression model which has an accuracy of 90.23

2 Related Work

The following section provides a discussion for works related to the idea of prediction Ethereum gas prices

• *Jack Studiner, Saqib Ahmad, Leo Toueg, and Chris Kingsland are with School of Computing at Queen's University*
E-mails: 18jdas@queensu.ca, saqib.ahmad@queensu.ca, 16lsit@queensu.ca, 19cek2@queensu.ca

2.1 An Analysis of the Fees and Pending Time Correlation in Ethereum

In this report the authors attempt to determine whether there is a relationship between gas prices and the pending time of transactions. The authors attempted to build a regression model that predicted pending time of transaction using gas price and related variables, but failed to do so. The conclusion was that gas price does not affect the pending time of individual transactions. [1]

2.2 Workshop Summary: 2019 IEEE / ACM Second International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB 2019).

This paper attempts to determine whether there are factors which affect the gas price of the ethereum network. The conclusion was that variables which highlighted how much in use the Ethereum network was were correlated with the Ethereum gas price. Variables highlighted in the paper include how many pending transactions there are on the network and the number of miners on the network. [2]

2.3 The influence factors on Ethereum transaction fees.

This paper attempts to determine whether there are factors which affect the gas price of the ethereum network. The conclusion was that variables which highlighted how much in use the Ethereum network was were correlated with the Ethereum gas price. Variables highlighted in the paper include how many pending transactions there are on the network and the number of miners on the network.[3]

2.4 Gas consumption analysis of Ethereum blockchain transactions

The paper begins with a background section detailing the basics of how Ethereum works. It gives a brief overview of the mining process and where gas prices come into play. The aim of the paper is to examine the relationship between different parameters of the Ethereum blockchain and the associated gas consumption. The conducted research determines the level at which these variables have an impact on gas fees. Data was collected from 5000 transactions using the Etherscan blockchain API. A regression analysis was performed that showed "87.8 percent of the variability in the response variable (gas consumption) is due to the parameters used in this analysis" [4]. Several research projects have been conducted that are similar in nature, although this paper is one of the most recent and one of the few to use both correlation and regression in its analysis. [4]

3 Dataset

The outliers in our data representing unrealistic spikes and dips in some of our features were handled by displaying the max values along with the different standard deviations of that variable and if any variable's max value was seen to be far out of line from it was removed and replaced by the mean of that feature. We chose to sub in the mean values for outliers as the mean values won't affect the amount of information in that feature as it is the average and the average values do not contribute any predictive power. Another method to find the outliers was plotting box plots and visually and finding which features have values that stray away from the median value, these values were then manually removed and subbed in with the mean of that feature. Also to ensure uniformity we removed all null values in the dataset and replaced them with the mean value of that feature.

3.1 Feature Engineering

When gathering our data through the API we pulled every feature we could get data on, there was no pre screening or selection process of any kind we just took it all, resulting in our dataset being made up of 30 different features. But we don't want all of the features, we only want features that help predict future gas prices and contribute to our models accuracy. To find the features with high predictive power and discard the features with low to no predictive power, we used a technique called Granger causality. This technique is an econometric test used to verify the usefulness of one variable to predict another. A variable is said to: Granger-cause another variable if it is helpful for forecasting the other variable. And Fail to Granger-cause if it is not helpful for forecasting the other variable. We applied this technique on each feature, comparing it to the gas price feature to figure out which variables contain predictive power or not based on the p values calculated with the technique. It was found that the difficulty and Hashrate features fail to granger cause and contribute no predictive power towards the gas price. Therefore the 2 features were removed from our dataset. Next we explored our data visually creating box plots and graphs to see the correlation between different variables. Boxplots showed us the ranges of different variables and exposed which variables had outliers which get cut out during data preprocessing

The dataset comes from an Ethereum API known as etherscan.io. The dataset broadly contains 3 types of variables: market values (show the supply and demand in the Ethereum market), reward statistics (show the amount of rewards which are given to miners), and network stats (show the popularity and amount of use of the Ethereum network). The market value variables are as follows. Marketcap: the total marketcap of Etherum, supply: the amount of Ethereum tokens in circulation, price: the daily price of Etherum. The Reward variables are as follows. blockSize: The amount of data that miners have to verify before they can get their reward. blockTime: the amount of time miners have to spend verifying transactions before they can get their rewards, blockRewardsUncles: the amount of rewards miners get from uncle blocks, blockCountsandRewards: the total amount of rewards miners are getting for verifying transactions. The network variables are as follows. AddressCount: the number of Etherum addresses active on the network. dailyTransactions: The amount of transactions taking place on the network, gasPrice: The gas price of the day, gasLimit: the total amount of gas allowed to be used on the network, gasUsed: the amount of gas used from the maximum gas limit, dailyETHBurnt: the amount of Ethereum sent to invalid accounts. The dataset only contains 2,442 data points. This is because each datapoint corresponds a day since the inception of the Ethereum blockchain. Since the blockchain has only been functioning for 2,442 days this is the maximum number of datapoints that we can obtain. The descriptive stats of the variables are shown in Appendix 1

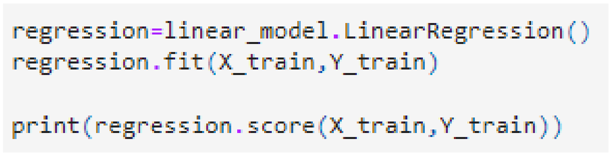
4 Methodology

With the data effectively preprocessed, a model can be constructed to predict gas prices from the other features. A regression approach is used because regression techniques perform well when trying to predict a numerical value such as gas price. Several models were considered, with the top three choices outlined in the next section.

4.1 Model Implementation

The first model is a linear regression. This is a basic machine learning approach to regression. It is mostly used to validate the regression problem and to set a baseline for more complex models to follow. For this Before training the model, the dataset is split into X and Y. X represents the independent features such as address count and market cap, while Y represents the dependant feature of gas price. The linear model is imported from sklearn and fit to the X and Y data.

The next approach uses an artificial neural network. The MLP Regression technique requires the data to be split into training and testing sets. A split of 80:20 train:test is used. This model has several hyper parameters and for the purposes of this project, the maximum iterations and hidden layer size parameters were changed. Ideal results were found when these values were 200 and 20 respectively.

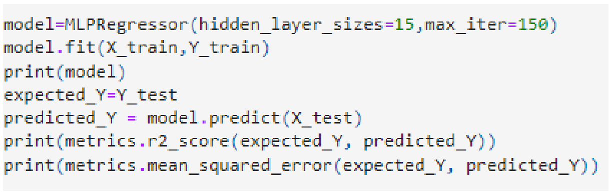


```
regression=linear_model.LinearRegression()
regression.fit(X_train,Y_train)

print(regression.score(X_train,Y_train))
```

Fig. 1. The code above represents the linear regression approach to the problem.

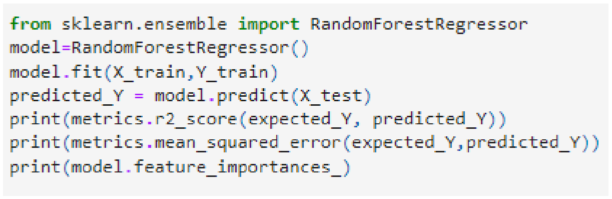
The data is also normalized prior to the training to achieve mean values close to 0. This is done to speed up the training process and allow for quicker convergence. The model is the MLPRegressor from sklearn. The training data is fit to the model. The trained model is used to make a prediction on the test data. The final approach is a Random Forest Regres-



```
model=MLPRegressor(hidden_layer_sizes=15,max_iter=150)
model.fit(X_train,Y_train)
print(model)
expected_Y=Y_test
predicted_Y = model.predict(X_test)
print(metrics.r2_score(expected_Y, predicted_Y))
print(metrics.mean_squared_error(expected_Y, predicted_Y))
```

Fig. 2. The code above represents the neural network approach to the problem.

sion. Random Forests inherently gives bias to features with the most predictive power. This results in more accurate predictions and is a major reason for its selection in this prediction problem. The random forest model uses a similar technique to the neural network approach with splitting the data into training and testing sets. A split of 90:10 train:test is used. The random forest model is the RandomForestRegressor from sklearn's ensemble learning package. The model is fit to the training data and subsequently tested with the test data. A feature importance matrix is also generated using the ensemble learning package.



```
from sklearn.ensemble import RandomForestRegressor
model=RandomForestRegressor()
model.fit(X_train,Y_train)
predicted_Y = model.predict(X_test)
print(metrics.r2_score(expected_Y, predicted_Y))
print(metrics.mean_squared_error(expected_Y,predicted_Y))
print(model.feature_importances_)
```

Fig. 3. The code above represents the random forest approach to the problem

5 Experiments and Results

For the linear regression approach, the model's performance is measured with the score function from sklearn. The model returned an accuracy score of 0.7153. This score represents an average performance at predicting gas prices, though it is a good indication that more complex models can better solve the prediction problem. The neural network approach underperformed when compared to team expectations and the simpler linear model. It is possible that an improper normalization of values is the reason behind this. The prediction accuracy returned a value of 0.5134 which is not ideal. This was accompanied by a mean square error of 0.223. Overall, the neural network model fails to accurately predict gas prices. The random forest approach returned very promising results. A prediction accuracy of 0.9023 and MSE of 0.03 represent an excellent performance at predicting gas prices.

5.1 Research Questions

The following section proposes research questions related to the performance of the team's approach at predicting Ethereum gas prices.

5.1.1 Would an ensemble technique work better then our approach?

Overall ensemble algorithms produce better results than any single model technique as it takes the results of many separate models and combines them to make one single prediction. Using an ensemble would also make the model more robust as it reduces the dispersion of single separate models and isn't impacted as much by outlier models which may return poor results to over fitting or simply just uninteresting sample for the test set.

5.1.2 Would normalizing features in the neural network between 0 and 100 produce a better result than normalizing where the mean is 0?

When we normalize between -1 and 1 we achieve our accuracy of 90.23. If we normalize between 0 and 100 it would give us a less granular normalization as values will be scaled down less and provide a larger range of values. Also since we cut out outliers the data will not be compressed and the "loss function is less likely to have very elongated valleys" which is a key characteristic that improves the performance of neural networks. Having a normalized feature tends to make the loss function more symmetrical making it easier to optimize as the gradients point towards the global minimum, thus increasing the accuracy of the neural network.

5.1.3 How would tuning hyperparameters benefit the result of our model?

In our existing model we did not adjust any of the hyper parameters. In our model we used a default minimum sample split of 2 which states the minimum number of observations in any given node to split it, the value of 2 resulted in a 90.23 accuracy. If we increased this parameter the results of our model may increase. Another hyperparameter that could be adjusted by trial and error to increase the overall accuracy of the model would be $\mathrm{n}_estimatorswhichwouldadjustthenumberoftreesintheforest.$

5.2 Threats to Validity

A threat to our validity might be the increments our data was collected, if we captured data at too large of a dataframe let's say every 24 hours. We might miss some important data that may affect our models prediction in a positive or negative way. A way it could negatively affect our model is if the time frame caused our dataset to be biased as it only captured the data late at night and didn't include data in other parts of the day. That missing data could give our model a more accurate prediction as opposed to the "narrow" data collected which could give a misleading result. Another threat to the validity of our model is overfitting, this threat stems off the same reasoning as the last point. The fact that the timeframe the data was collected doesn't paint the entire picture and only a fraction of the data's story is captured. When our model runs on this fraction of data it may pick up on noise or other characteristics that aren't spread throughout the dataset. This will cause the model to over fit to this noise and negatively impact the performance of the model. For example to make this clear, let's say our data was captured at 24 hour increments at 4am every day. The dataset would be skewed towards nighttime trading activity which is very different from daytime trading activity as in general there is a lot more volume on the blockchain during the day. So the model will over fit itself to the nighttime "noise" and as a result will have trouble accurately predicting gas prices during the day.

References

[1] Azevedo Sousa, José Eduardo et al. "An Analysis of the Fees and Pending Time Correlation in Ethereum." International journal of network management 31.3 (2021): n. pag. Web.

[2] Tonelli, Roberto et al. "Workshop Summary: 2019 IEEE / ACM Second International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB 2019)." Software engineering notes 44.3 (2019): 48–52. Web.

[3] "The influence factors on Ethereum transaction fees." [Online]. Available: https://hal.inria.fr/hal-02403098/document. [Accessed: 23-Apr-2022].

[4] M. M. A. Khan, H. M. A. Sarwar, and M. Awais, "Gas consumption analysis of Ethereum blockchain transactions," Concurrency and computation, vol. 34, no. 4, 2022, doi: 10.1002/cpe.6679.

6 Group Member Contributions

Group Member	Contributions
Jack	Methodology, 5.1 of Experiments, Editing, Formatting
Leo	Motivation, Introduction,Editing, Formatting
Saqib	Dataset
Chris	Data Preprocessing, Feature Engineering, Experiment/Result Questions, Threats to Validity.

7 Replication Package

https://github.com/JStudiner/CMPE351Project

8 Conclusion and Future Work

Based on the experiment, it has been concluded that the best method to predict Ethereum Gas Price is by using the Random Forest Regression model. The most important features in the regression model were pendingTransactions (The total amount of pending transaction on the Ethereum Network), marketcap (The total value of Ethereum), and addressCount (the total number of Ethereum wallets which are able to receive Ethereum). Additionally, we have learnt that binning does not work on the current Etherum dataset, this is because it has only been 2400 days since the Etherum network has been active. Given the limited size of the dataset, binning reduces the granularity of the data too much making it difficult for the model to understand what is happening.

For future work, we would consider using XGBoost and other advanced ensemble methods to see if they can improve upon the performance of random forest. Additionally, the current model only predict the average gas price of a day. It may be more useful to predict gas price over a shorter time frame (hour, minute, second). This is because some users might want to do a transaction within a 24 hour time frame and would like to know when the ideal time to do so is.