**Ethereum Gas Price Prediction**

| Aryan GD Singh | Rupanshoo Saxena | Pragya Singh | Saatvik Bhatnagar |
| --- | --- | --- | --- |
| 2019459 | 2019096 | 2017305 | 2019097 |
| aryan19459@iiitd.ac.in | rupanshoo19096@iiitd.ac.in | pragya17305@iiitd.ac.in | saatvik19097@iiitd.ac.in |

**1. Motivation**

ETH is a cryptocurrency. It is digital money that is global and decentralized, secured by cryptography. Ethereum is a technology that allows the transaction of the cryptocurrency Ether for a small fee.   
In Ethereum, the user should purchase a number of gases for reaching a transaction consensus. Gas Limit restricts the execution time and the memory cost of the transaction.   
Generally, the higher the gas price, the less time spent on reaching a consensus of the transaction. In a block, transaction gas prices vary greatly, generating a gas price that is reasonable in terms of both the price and the time spent on reaching consensus is of utmost importance.  
Through this project, we aim to find the lowest reasonable gas price for a block using features influencing it.

**2. Introduction**

In Ethereum, Gas refers to the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network. Since each Ethereum transaction requires computational resources to execute, each transaction requires a fee. Reaching a transaction consensus costs a certain number of gas. They are purchased by users at their self-defined gas prices. Generally, the higher the gas price, the shorter the time is spent on reaching consensus,thus speeding the transactions. The exact price of the gas is determined by supply and demand between the network's miners, who can decline to process a transaction if the gas price does not meet their threshold, and users of the network. Since the transaction gas prices still vary greatly in a block, generating a reasonable price that can make a trade-off between the consensus time and the gas's cost is of great significance.

**3. Literature Survey**

1. The paper [1] uses a regressive approach to build the prediction model on the basis of various features of a transaction, this would form the basis of our model.
2. The paper [2] uses a model different from the widely used Geth algorithm, this provides us a different perspective on our model and provides other features for inclusion in our model
3. The paper [4] uses multiple models - binomial classification algorithms and regression models. Different models are used to see how different assumptions can affect the model’s performance.

**4. Dataset**

The datasets used were taken from the Ethereum blockchain using GCP’s BigQuery API - Transactions and Blocks.

The two main entities which affect the gas price are blocks and transactions. We used the BigQuery API to join the blocks and transactions dataset and get the relevant features.

**4.1 Description**

**4.1.1 Transaction Dataset:**

It consists of 1.1 million transactions that took place from 00:00:02 UTC to 23:59:38 UTC, July 3, 2021. The total number of unique blocks processed during this time is 6357.

The dataset consists of 11 attributes:

- transaction details like hash, transaction index

- sender and receiver addresses

- block details like block hash, block number

- value of the transaction

- gas and gas price

- other details like nonce, input

**4.1.2 Block Dataset:**

The dataset consists of 200,000 entries and 18 attributes:

- number, hash

- parent details like parent\_hash, sha3\_uncles (combined hash of all siblings of a parent)

- transaction details like transaction\_root, transaction\_count,

- state\_root, miner, recipients\_root

- gas\_limit, gas\_used

- difficulty, total\_difficulty

- other details like nonce, size, extra\_data, timestamp

**4.1.3 Final Dataset:**

We combined both these datasets to obtain our required dataset. The SQL query used to obtain the same is:

*SELECT*

*BLOCKS.number, BLOCKS.difficulty, BLOCKS.total\_difficulty, BLOCKS.size, BLOCKS.gas\_limit, BLOCKS.transaction\_count, TRANSACTIONS.value, TRANSACTIONS.gas, TRANSACTIONS.gas\_price*

*FROM*

*`bigquery-public-data.crypto\_ethereum.blocks` AS BLOCKS*

*INNER JOIN*

*`bigquery-public-data.crypto\_ethereum.transactions` AS TRANSACTIONS*

*ON*

*BLOCKS.number = TRANSACTIONS.block\_number*

*WHERE*

*DATE(BLOCKS.timestamp) = '2021-07-03'*

The final dataset consists of 12 attributes:

-number: the block number

- difficulty: Integer of difficulty of the block

- total difficulty: Integer of the total difficulty of the chain until this block.

- size: the size of the block

- value: amount of ETH to transfer from sender to recipient.

- gas: the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network.

- gas\_price: the price we are paying for every unit of gas used in the transaction.

- gas\_limit: the max amount of gas units that can be consumed by the transaction.

- transaction\_count: the number of transactions in the block.

The attributes that affect gas price are:

tx\_gas\_limit: it affects whether a transaction would be mined or not

difficulty: it affects the ease of mining blocks, thus an increase in difficulty would lead to higher gas prices. It has a positive correlation with gas price

**4.2 Dropping Columns**

The following columns were dropped from the transaction dataset while forming the final dataset since they were unrelated to the prediction of gas price:

- hash, transaction index

- sender and receiver addresses

- block hash, block number

- nonce, input.

The following columns were dropped from the block dataset while forming the final dataset since they were unrelated to the prediction of gas price:

- number, hash

- parent\_hash, sha3\_uncles

- transaction\_root, transaction\_count

- state\_root, miner, recipients\_root

- nonce, extra\_date, timestamp

The following columns were dropped from the Final dataset:

- total\_difficulty: it is the integer of total difficulty of the chain until this block. We later realized that this feature doesn’t play an important role in determining gas price.

- value: while training the models, we noticed that there were big values in this attribute that didn’t really play a huge role in training.

- gasPrice: this was dropped because we are trying to predict min gas price we should pay so that it gets included in the block and this attribute denotes the transaction gasPrice which we won’t have access to.

**4.3 Feature Engineering**

**4.3.1 Handling NULL/Zero values**

The entries which had their gas\_price = 0 were removed from the table. Replacing it with mean values would not have been correct and would have hindered the predicting process.

**4.3.2 Feature addition**

Firstly, we grouped the data by number (block number). Next, for every block number, we find the minimum, mean and maximum gas prices and store these.

Next, for each block, we make 3 new features → past\_min, past\_mean, and past\_max.

past\_min: contains the mean of the minimum gas price for the past 5 blocks.

past\_mean: contains the mean of the mean gas price for the past 5 blocks.

past\_max: contains the mean of the maximum gas price for the past 5 blocks.

For each transaction, past\_min, past\_mean, and past\_max were appended based on the block number.

**4.3.3 Dataset Scaling**

We standardized the input data using StandardScalar so that the model is not skewed due to absolute values.

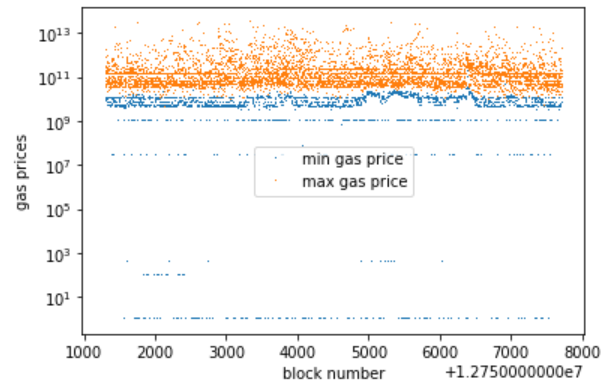
**4.4 Preparing Training and Testing Data**

We split the dataset in a 7:3 ratio using Sklearn’s train\_test\_split.

**4.5 Dataset Analysis**

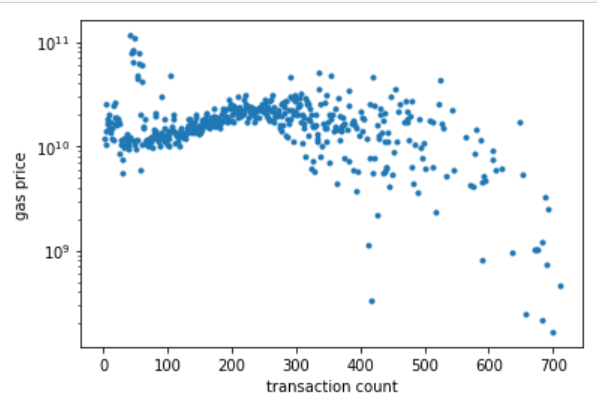
We conducted some operations on the data so as to get a better understanding of the dataset.

In Fig. 4.5.1, we can see that there exists a gap between the minimum and maximum gas price for a particular block, so we can infer that it is possible to predict an apt gas price within this range that will lead to our transaction being included in the block.

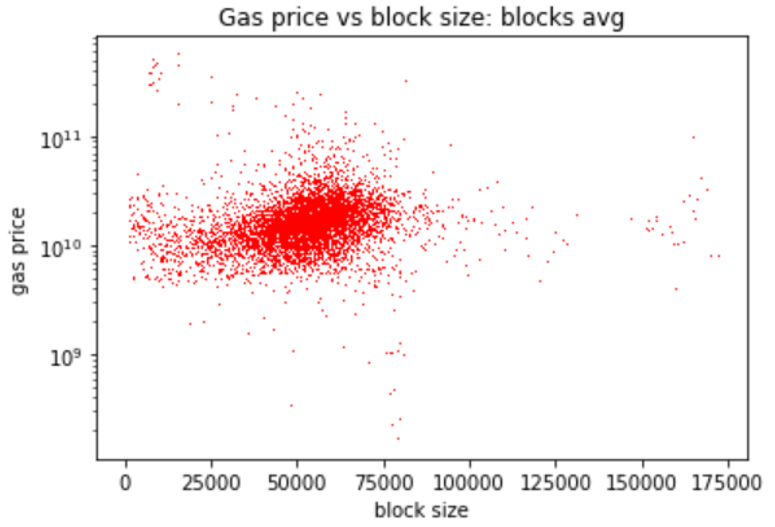


*Fig 4.5.1*

In Fig. 4.5.2 and 4.5.3, we can see that there does not exist a strong relationship between the block size and transaction count to the gas price, so we need to find a more complex linear relationship using Regression techniques that can accurately predict the gas price.

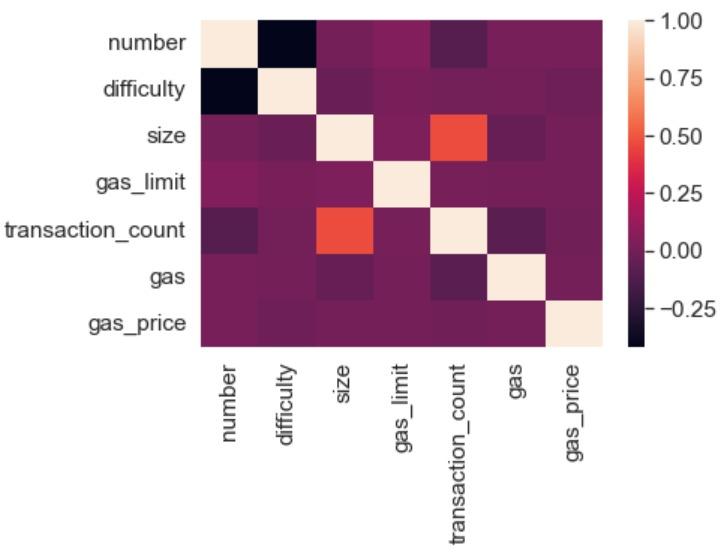


*Fig 4.5.2*



*Fig 4.5.3*

Fig 4.5.4 shows the correlation of different attributes of the final dataset with each other, lighter to darker color shows higher to lower correlation.

****

*Fig 4.5.4*

**5. Methodology**

**5.1 Models**

We trained our dataset on the following machine learning models:

**5.1.1 Linear Regression (Simple, Ridge, Lasso Regression)**

We used simple regression with default values, grid search using negative RMSE for both lasso and ridge regression.

**5.1.2 Random Forest**

Implemented default random forest model, grid search using negative RMSE.

**5.1.3 Neural Network**

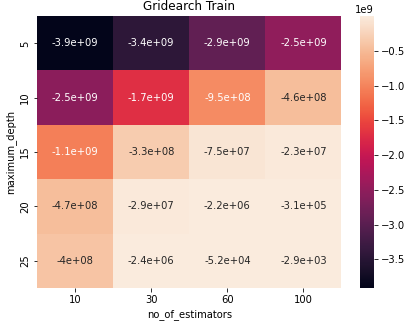
We used a neural network with 4 layers, the first 3 have activation ReLu, and the last has a linear activation function, loss -> MSE and metric -> MSE.

**5.1.4 ADA Boost**

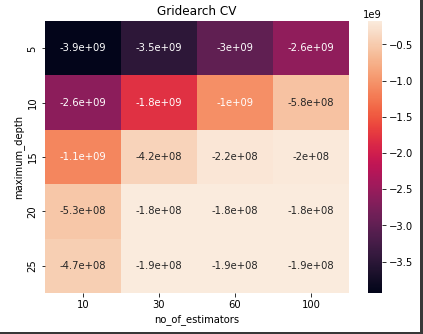
We used the default ADA boost model.

**5.1.5 XG Boost**

We implemented XG Boost with grid search.



*Fig 5.1: XG Boost Grid Search on training data*

**

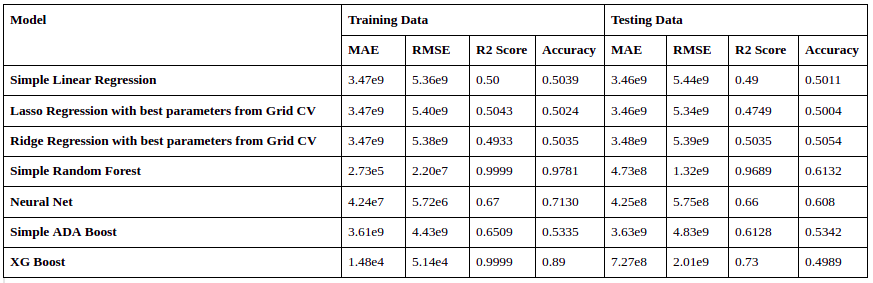
*Fig 5.2: XG Boost Grid Search on testing data*

For comparison among these models we are using the following criteria:

1. RMSE (Root Mean Square Error)
2. MAE (Mean Absolute Error)
3. R2 Score
4. Accuracy

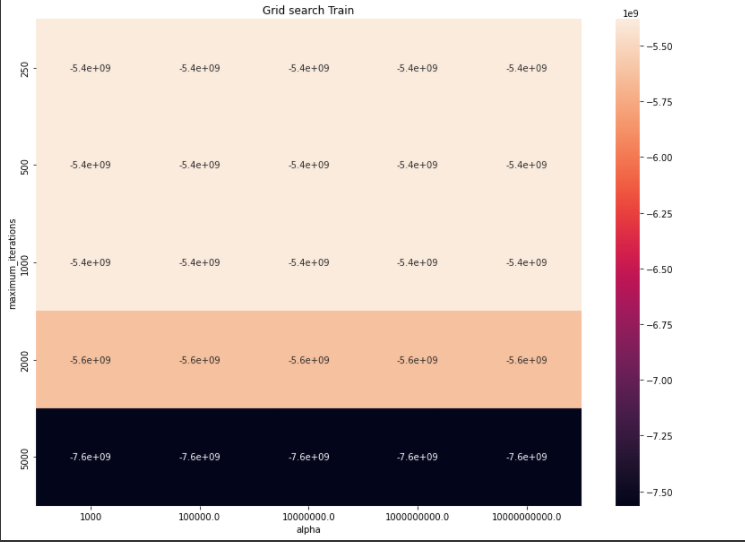
**6. Analysis and Results**

**6.1 Results**

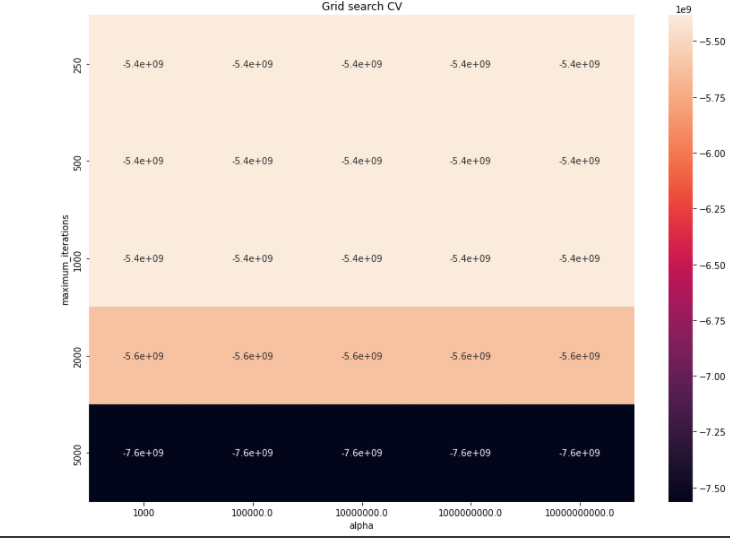
****

*Table 1: Results*

**6.2 Exploring Models**

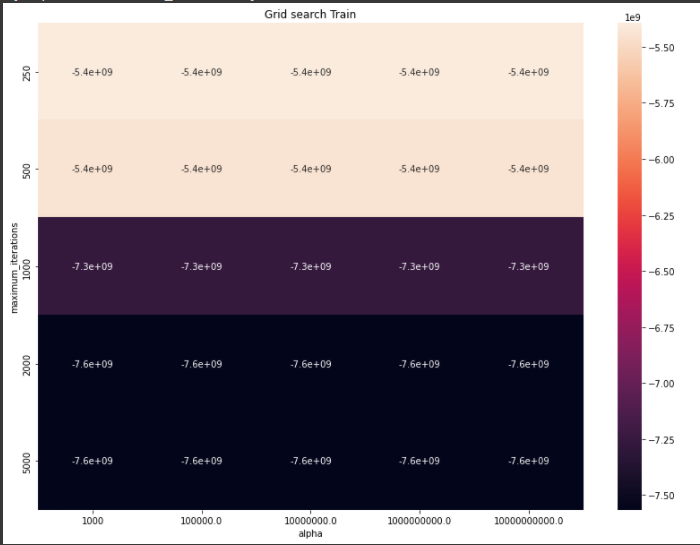
****

*Fig 6.2.1: Heat map of negative root mean squared error on training data while using Grid Search CV with Lasso Regressor as the parameter*



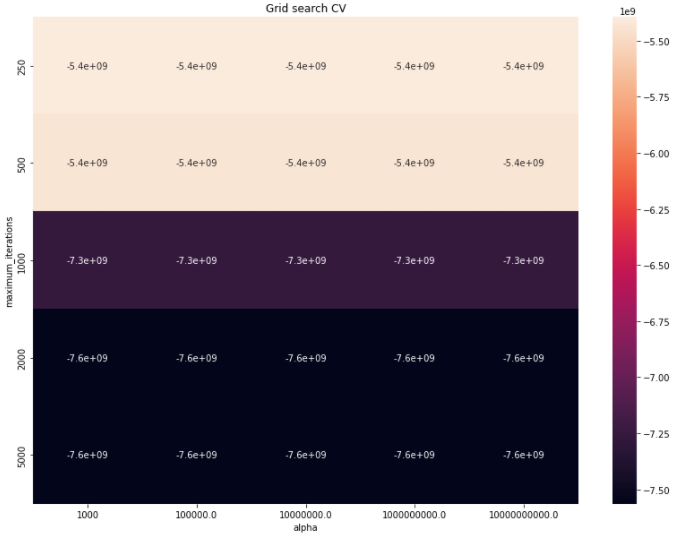
*Fig 6.2.2: Heat map of Negative root mean squared error of testing data while cross-validation using grid search with Lasso Regressor as the parameter.*

Figures 6.2.1 and 6.2.2 denote heat maps of negative root mean squared error using Grid Search CV with Lasso regressor as the parameter on Training and testing datasets.

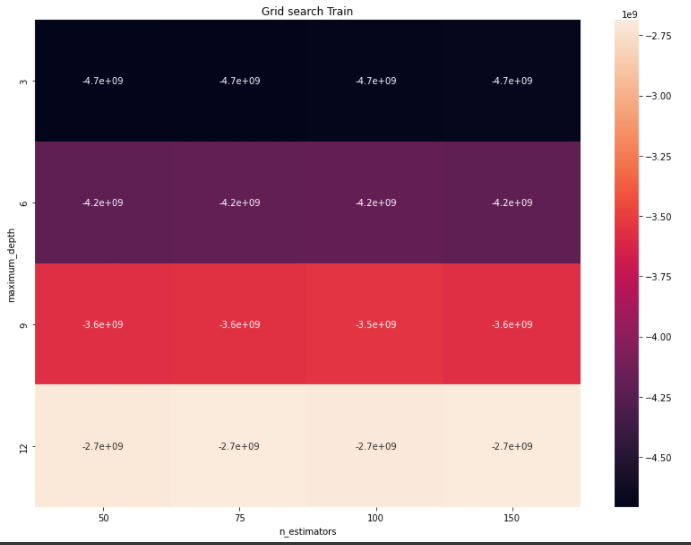
**

*Fig 6.2.3: Heat map of negative root mean squared error on training data while using Grid Search CV with Ridge Regressor as the parameter*

Figures 6.2.3 and 6.2.4 denote heat maps of negative root mean squared error using Grid Search CV with Ridge regressor as the parameter on Training and testing datasets.

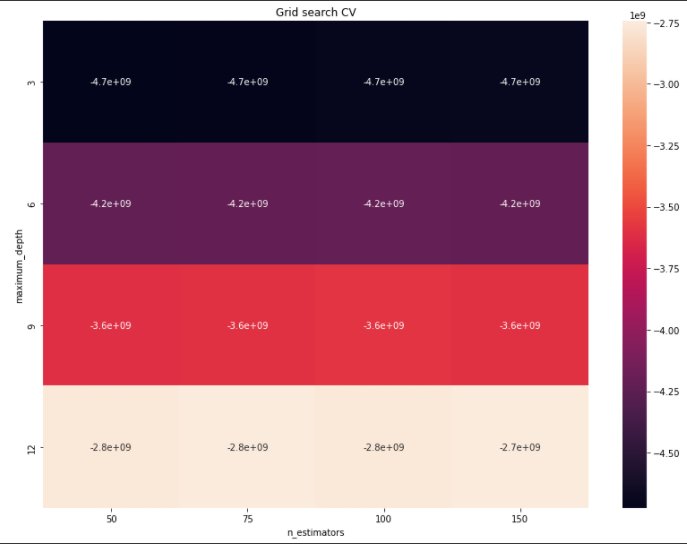
**

*Fig 6.2.4: Heat map of Negative root mean squared error of testing data while cross-validation using grid search with Ridge Regressor as the parameter.*

**

*Fig 6.2.5: Heat Map of Negative root mean squared error of training data while cross-validation using grid search with Random Forest*

Figures 6.2.5 and 6.2.6 denote heat maps of negative root mean squared error using Grid Search CV with Random Forest on Training and testing datasets.

**

*Fig 6.2.6: Heat Map of Negative root mean squared error of testing data while cross-validation using grid search with Random Forest*

In our analysis, we thoroughly examined and observed a few different Machine learning models (listed in section 5.1) to predict the gas price of ethereum. Linear Regression gave us low accuracy on training and testing data. The Random Forest and Neural network models perform the best with accuracy 0.6132 and 0.608 respectively.

Overall, Random Forest gave us the best accuracy.

**7. Conclusion**

We started by learning about Ethereum, how it functions, etc. We understood the importance of its application in today’s world and thus the need of predicting the lowest reasonable gas price for a block.

Afterward, we critically analyzed our data, which initially consisted of 2 different datasets for transactions and blocks. We studied their features and combined the two datasets, dropped columns that didn’t affect the prediction process, dropped rows with incomplete data, and added more features that could help us make predictions.

We trained some models like Linear Regression (Ridge and Lasso), Random Forest, Neural Network, ADA Boost, and XG Boost.

From our results, Random Forest performed the best. Neural Network also performed well followed by ADA Boost and Linear Regression.

**8. References**

[1] Fangxiao Liu, Xingya Wang\*, Zixin Li, Jiehui Xu, Yubin Gao. (2019). Effective GasPrice Prediction for Carrying Out Economical Ethereum Transaction.

[2] Sam M. Werner, Paul J. Pritz, and Daniel Perez. (2020). Step on the Gas? A Better Approach for Recommending the Ethereum Gas Price.

[3] Vinicius C. Oliveira, Julis Almeida Valadares, Jose Edurado A. Sousa, Alex Borges Vieira, Heder Soares Bernardino, Saulo Moraes Villela. (2021). Analyzing Transaction Confirmation in Ethereum Using Machine Learning Techniques.

[4] Matthew Chen, Neha Narwal, and Mila Schultz. Predicting Price Changes in Ethereum.