Ethereum Project

Binance Coin (BNB)

Prateek Sarna pxs180012; Shivani Mankotia sxm180018

INTRODUCTION

Our project primarily aims at working on a particular ERC20 token, namely Binance Coin or BNB. Here, we try to come up with our own statistical inference about BNB data set along with Token Price data set, BNB. In the first part of our project, we try to find how many times a particular user buys and sells a token. We also try to fit these observations in a discrete distribution type that best fits it. In addition, we create layers of transactions with increasing amounts, and find the number of transaction made on a particular date. We also compute the best correlation between the price of data and the layers of token amount that we created. In the second part of the project, we try to predict the price of token the next day by creating a multiple linear regression model. In this project report, we have explained our technique of solving the above mentioned problems with help of R code along with relevant outputs and graphs.

ETHEREUM

According to their website, “Ethereum is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference. These apps run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property.” In other words, Ethereum is trying to become a decentralized platform where anyone, from any part of the world can rent out some computational power and create decentralized applications, or Dapps, which can run top of this platform. Dapps are created using Smart Contracts, which are self executing contracts implemented in software code and get executed when certain conditions are met. [1]

ERC20 TOKENS

According to Wikipedia, ERC20 is a “list of rules that an Ethereum token has to implement, giving developers the ability to program how new tokens will function within the Ethereum ecosystem. The ERC-20 token standard became popular with crowdfunding companies working on initial coin offering (ICO) cases due to the simplicity of deployment, together with its potential for interoperability with other Ethereum token standards.” In simpler terms, ERC20 provides a set of rules and regulations that helps the creation of tokens by Ethereum based smart contracts. In ERC20, ERC expands to Ethereum Request for Comment and 20 is the number assigned to this request. A set of rules are followed that allows them to be shared , exchanged for other tokens and be transferred to a crypto-wallet. The ERC20 standard follows 6 rules which are mandatory and 3 rules which are optional. Mandatory rules include totalSupply, balanceOf, transfer, transferFrom, approve and allowance and optional rules include Token Name, Symbol and Decimal (up to 18). [2]

BINANCE COIN (BNB)

The token which we worked on was Binance Coin, which is a crypto-coin issued by Binance exchange and trades with the BNB symbol. Binance coin is among the many which run on the Ethereum blockchain and follows the ERC20 standard. It has a strict limit of maximum 200 million BNB tokens. Binance coin is expected to fuel the operations of Binance exchange and its ecosystem. It supports multiple utilities on the Binance ecosystem, which includes paying for trading fees, exchange fees, listing fees, and any other fees on the Binance exchange and can also be used to invest in certain ICOs that are listed through Binance's Launchpad program. Binance Coin had a market cap of around $1.4 billion during early April 2018. At present, it does not offer exchange against fiat currencies and exchange in BNB is possible only through crypto coins like bitcoin or ether tokens. Binance is said to become the native currency of the decentralized Binance exchange.[3]

VISUALIZATION OF DATA

Our data files contain two primary groups: token network edge files, and token price files. The Ethereum project is a blockchain platform, and our data comes from there. Although Ethereum started in 2015, most tokens have been created since 2016. As such, tokens have different starting dates, and their data starts from that initial date.

Token Network Edge file has the following row structure:
fromNodeID\toNodeID\tunixTime\ttokenAmount\r\n.

```
##        FromNodeID ToNodeID       Time  TokenAmount
## 202810    1586278  1612097 1519493206 1.000000e+17
## 199166    1527966  1608805 1519491082 1.000000e+17
## 5657            5  1447645 1523802246 9.773439e+18
## 353015    1722264  1722265 1503012178 5.000000e+09
## 216756    1541442  1622411 1519498268 1.000000e+17
## 310338          5  1683466 1513963590 4.990000e+20
## 233103    1484989  1639851 1519515454 1.000000e+17
## 130041    1543613  1546967 1519457988 1.000000e+17
## 264203     325160  1667311 1519542387 1.000000e+17
## 262248    1653052  1665466 1519540987 1.000000e+17
## 244428     956887  1649107 1519525512 1.000000e+17
## 63848     1468597  1492690 1519376857 1.000000e+17
## 224400    1609188  1631593 1519503191 1.000000e+17
## 11974     1450792        5 1522046942 1.999750e+21
## 85964      926041  1479068 1519399177 1.100000e+17
## 270314    1670362  1670359 1519547747 1.000000e+17
## 300175    1684996  1689516 1515077802 1.000000e+17
## 162668      40072    57174 1519475849 6.500000e+18
## 293330    1684150  1684397 1514703392 1.000000e+17
## 313165    1683119  1696319 1511687709 1.000000e+17
```

Figure I: 20 Sample points from Token Network Edge

We began by finding how many times a user buys and sells a token along with the discrete distribution type that would fit the data. Here, the columns 'FromNodeID' and 'ToNodeID' store the seller and the buyer IDs respectively.

Finding and Removing Outliers:
An outlier is a data point that is significantly different from other data points in a data set. We are removing outliers as we have a lot of data and our sample won't hurt by removing a questionable outlier. Also, outliers tend to affect mean of the data, however they have no significant effect on median. For our token BNB, there were no outliers as no amount was greater than the total amount of token BNB which was 'total_supply = 1978317006.32'.

I. Number of time a buyer buys a token and seller sells a token.

Selling of Token:
We first calculated the frequency of sellers, which came from the 'FromNodeID' column.
In order to calculate how many times a user sells a token, we again took the frequency of 'FromNodeID' column's frequency, i.e. we took frequency of the 'freq' column obtained in the first step and stored it as 'counts_sells'.

We plotted the frequency using the 'barplot' function and obtained the following bar graph for first 10 sample tokens.

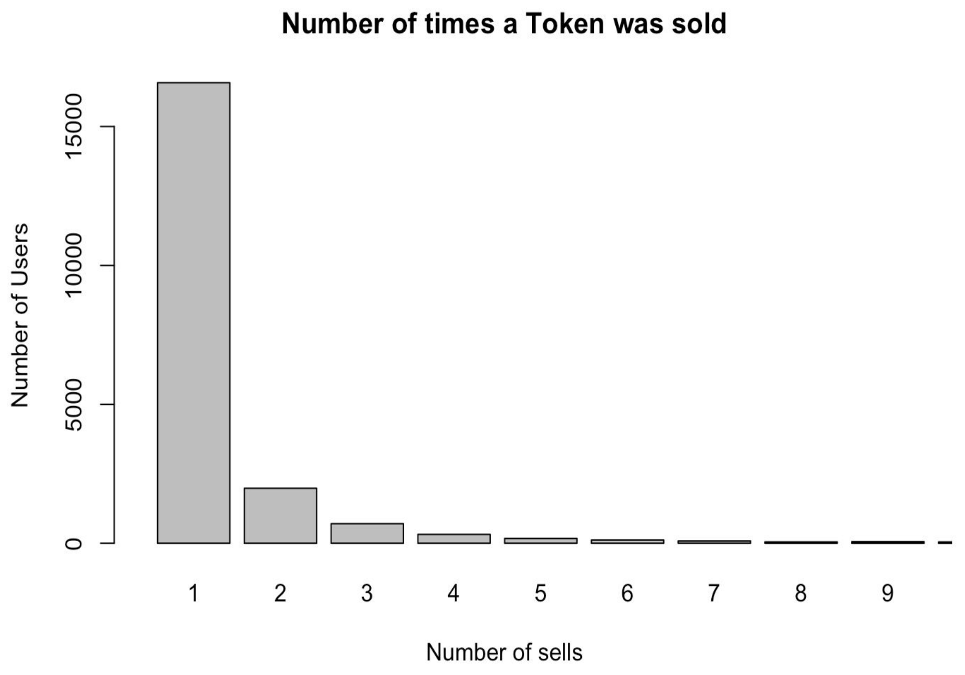


Figure II: Barplot of Token Sold

In the bar graph above, x-axis represents the tokens and y-axis represents the number of users who sold a particular token. From the first 10 samples of the tokens sold, we can see that Token 1 was sold the maximum number of times. This can also be treated as an outlier because it is hard to believe that a single token was sold 15000 times when others were barely sold 1000 times on an average.

Buying of Token:
To calculate the number of tokens bought, we followed the same procedure we used in *selling of tokens* using the ‘ToNodeID’ column.
We first calculated frequency of the ‘ToNodeID’ column.

We then took the frequency of ‘freq’ in the frequency table to calculate how many times a user bought a token and stored it as ‘count_buys’.
The bar graph obtained for number of tokens bought by a user shows Token 1 was bought the maximum number of times.

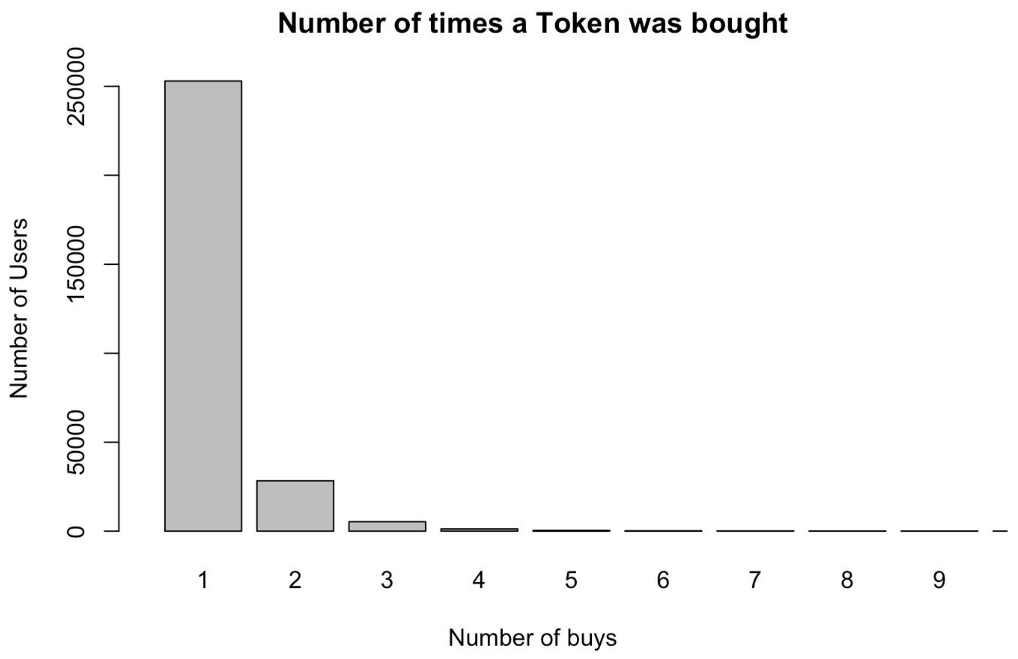


Figure III: BarPlot of Token Bought

Discrete Distribution:
To fit our above observations into a discrete distribution, we used “fitdistrplus” library of R. We converted the ‘count_sells’ and ‘count_buys’ into vectors and calculated the Mean, Variance and Standard Deviation to obtain an idea about the kind of distribution that would fit our data.

R Code for Selling of Token:

```
library(fitdistrplus)

#Distribution Types for Selling of Tokens

cat("Mean is: " , mean(count_sells_vector), "\n")
```

```
## Mean is:  59.17827
```

```
cat("Variance is: " , var(count_sells_vector), "\n")
```

```
## Variance is:  776720.4
```

```
cat("Standard Deviation is: " , sd(count_sells_vector), "\n")
```

```
## Standard Deviation is:  881.3174
```

Figure IV: R Code for selling of tokens

R Code for Buying of Token:

```
#Distribution Types for Buying of Tokens

cat("Mean is: " , mean(count_buy_vector), "\n")
```

```
## Mean is:  4987.241
```

```
cat("Variance is: " , var(count_buy_vector), "\n")
```

```
## Variance is:  1112241722
```

```
cat("Standard Deviation is: " , sd(count_buy_vector), "\n")
```

```
## Standard Deviation is:  33350.29
```

Figure V: R Code for buying of tokens

By looking at the above frequency graphs and statistics (mean, variance, standard deviation) for both buying and selling of the token, we narrowed down *Negative Binomial* and *Poisson Distributions* as the potential best fit discrete distributions for our data.

Negative Binomial Distribution:

"In probability theory and statistics, the negative binomial distribution is a discrete probability distribution of the number of successes in a sequence of independent and identically distributed Bernoulli trials before a specified (non-random) number of failures (denoted r) occurs. For example, if we define a 1 as failure, all non-1s as successes, and we throw a die repeatedly until 1 appears the third time (r = three failures), then the probability distribution of the number of non-1s that appeared will be a negative binomial distribution." - *Wikipedia*

Poisson Distribution:

"In probability theory and statistics, the poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known constant rate and independently of the time since the last event.The Poisson distribution can also be used for the number of events in other specified intervals such as distance, area or volume." - *Wikipedia*

Fitting Data in Negative Binomial Distribution:

Selling of Token:

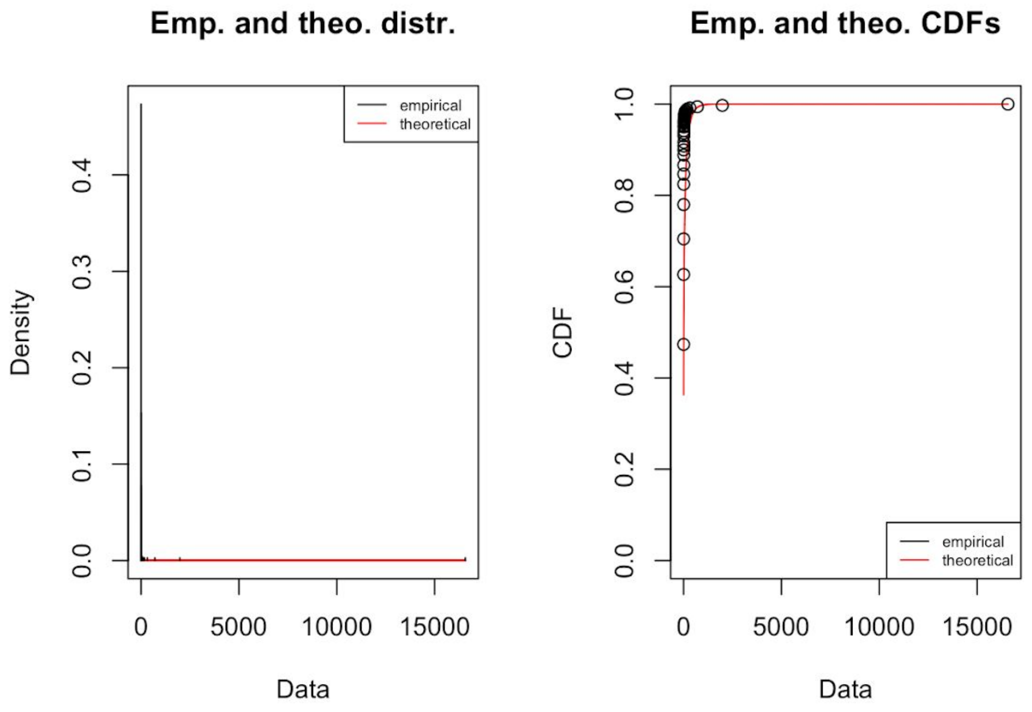


Figure VI: Fitting Data in Negative Binomial Distribution for Selling of Tokens

The following parameters were observed:

Fitting of the distribution ' nbiom ' by maximum likelihood

Parameters :

	Estimate	*Std. Error*
Size	0.2146788	0.01259763
Mu	59.1703560	6.75135597

Loglikelihood: -1293.966 AIC: 2591.932 BIC: 2599.698

Correlation matrix:

	Size	*mu*
Size	1.00000e+00	6.854487e-05
Mu	6.854487e-05	1.000000e+00

Buying of Token:

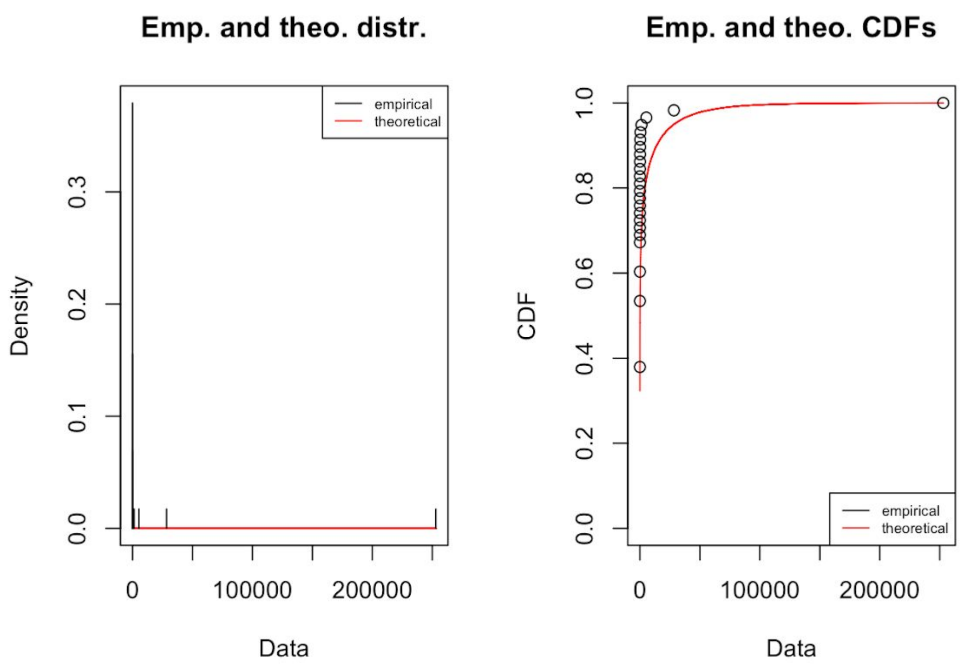


Figure VII: Fitting Data in Negative Binomial Distribution for Buying of Tokens

```
The following parameters were observed:
Fitting of the distribution ‘ poisson ‘ by maximum likelihood
Parameters :
        Estimate        Std. Error
Size    0.1160747       1.612189e-02
Mu      4989.2407547  1.875750e+03
Loglikelihood: -296.9747        AIC:    597.9494        BIC:    602.0702
Correlation matrix:
                Size            mu
Size    1.00000e+00     -0.0001405267
Mu      -0.0001405267 1.0000000000
```

Fitting data in Poisson Distribution:

Selling of Token:

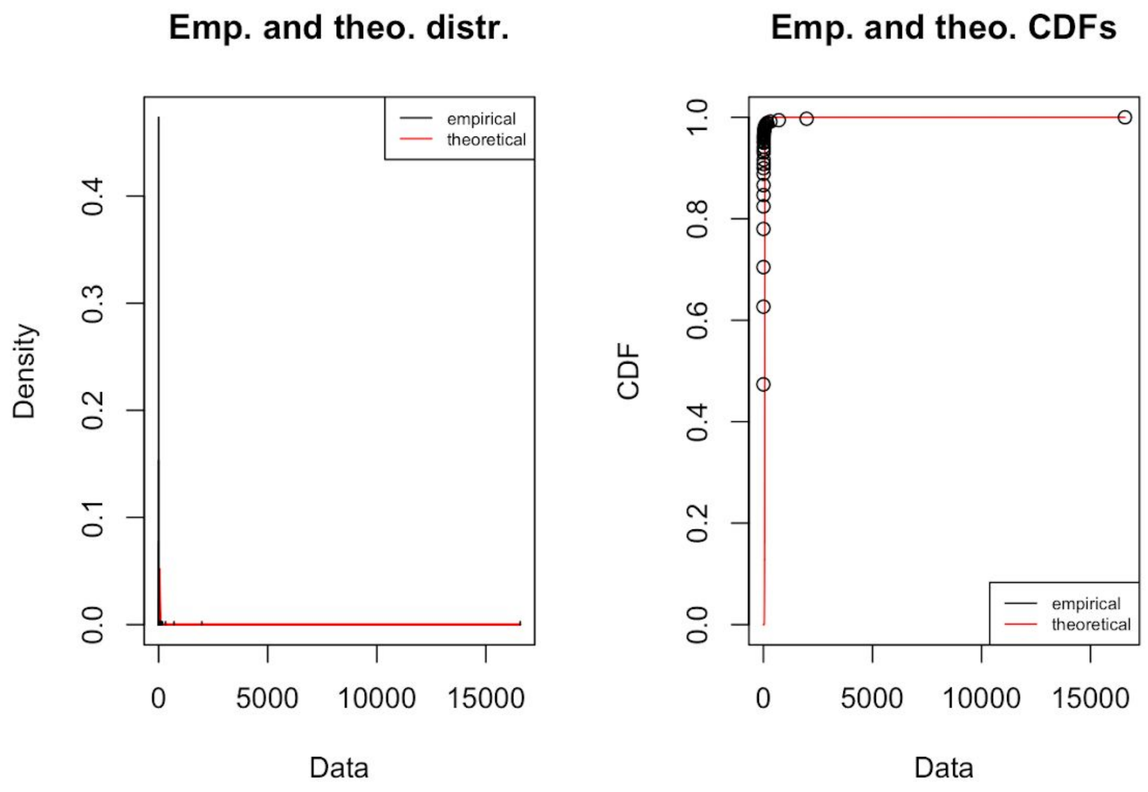


Figure VIII: Fitting Data in Poisson Distribution for Selling of Tokens

The following parameters were observed:

Fitting of the distribution ‘ poisson ‘ by maximum likelihood

Parameters :

	Estimate	*Std. Error*			
Lambda	59.17827	0.4060068			
Loglikelihood:	-100691.2	AIC:	201384.5	BIC:	201388.4

Buying of Token:

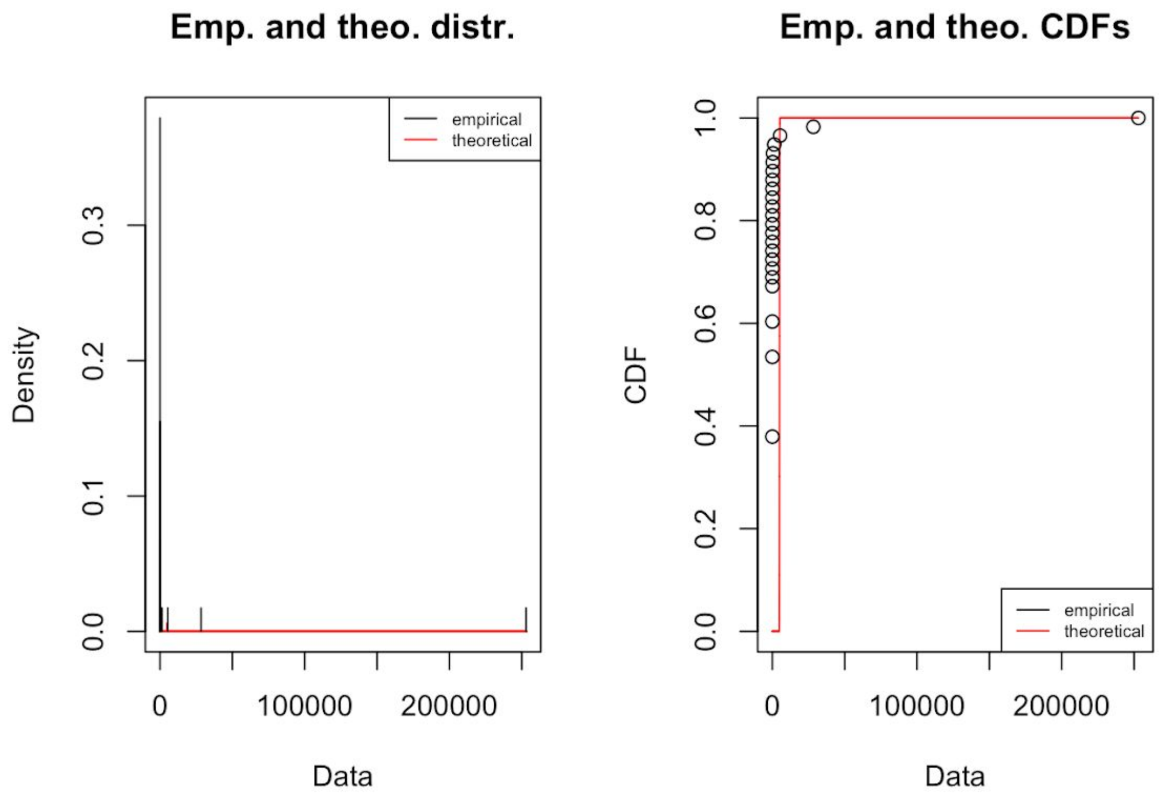


Figure IX: Fitting Data in Poisson Distribution for Buying of Tokens

The following parameters were observed:
Fitting of the distribution ‘ poisson ‘ by maximum likelihood
Parameters :

	Estimate	*Std. Error*			
Lambda	4987.241	9.256626			
Loglikelihood:	-1036783	AIC:	2073567	BIC:	2073569

II. Correlation between features

For this section of the project, our aim was to create layers of transactions with increasing amounts, find all transactions for a given day and finally find an algorithm to compute the correlation of price data with each of the layers. Here, we used 'bnb' as our Token Price file.

To construct layers, we plotted frequency table for 'TokenAmount' column from Token Network Edge file (BNB Token).
With the help of this frequency table, we plotted the graph for 'Frequency vs TokenAmount' which gave us a visual estimation of how concentrated the data was. We combined these clusters of data to create multiple layers.

- Frequency plot of entire 'TokenAmount' data.
- Frequency plot of most concentrated 'TokenAmount' data. This is the layer we have chosen to compute the correlation value. We also removed outliers above the count of 400.

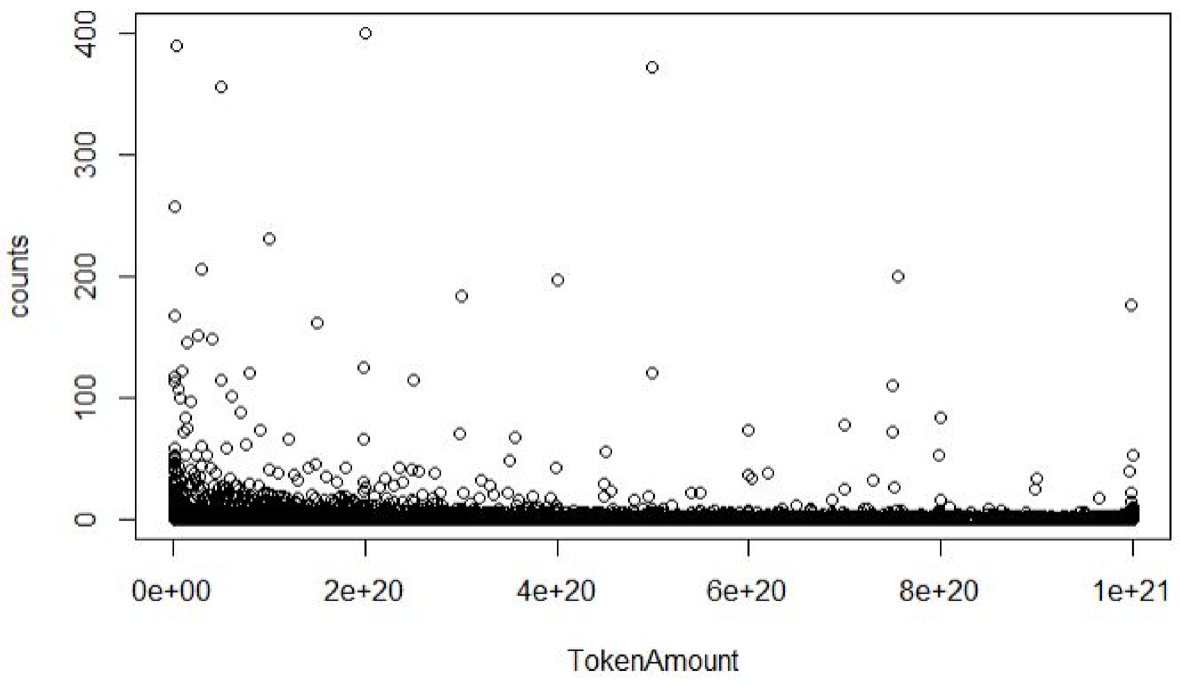


Figure X: Frequency plot of most concentrated 'TokenAmount' data

Hence, with the help of visual estimation of the frequency graph we constructed 28 layers and amongst these, we chose the following layer to compute the correlation of price data.

```
layer<- tokenBNB[which(tokenBNB$TokenAmount>1e+18 &
tokenBNB$TokenAmount<=1e+21),]
```

We chose the column 'High', which indicates the highest value of the tokenAmount for a particular day, to compute the correlation of price with our layer. To compute all the transactions done on a given day, we first converted unix time i.e values in the ‘Time’ column in our Token Price file to standard date format (mm/dd/yyy) using “anydate” library.

We then inner-joined the token prices table with the date-frequency table and plotted a graph between ‘High’ price and Date.

The following is the graph obtained for High vs date:

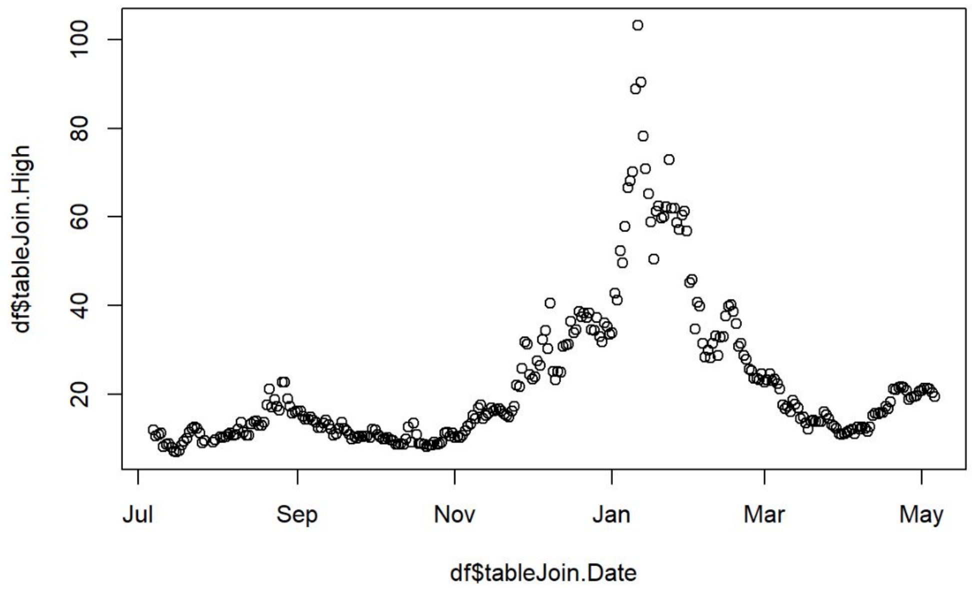


Figure XI: Frequency plot of High vs Date

Computation of Correlation:

To find the best correlation, we tested our data on all the three Correlation Coefficients: Pearson, Spearman and Kendall.

We used *cor.test()* function to computer the various correlations.

- *Pearson Correlation:*

```
#PEARSON cO-RELATION
cor.test(df$tableJoin.High, df$tableJoin.Frequency, method = "pearson")
```

Pearson’s product-moment correlation

Data: df$tableJoin.High and df$tabkeJoin.Frequency

T = 10.064, df = 299, p-value < 2.2e-16

Alternative hypothesis: true correlation is not equal to 0

95 percent confidence interval:

0.4134700   0.5829138

Sample estimates:

    Cor

    0.5030097

Price:High vs Date plot for Pearson Correlation:

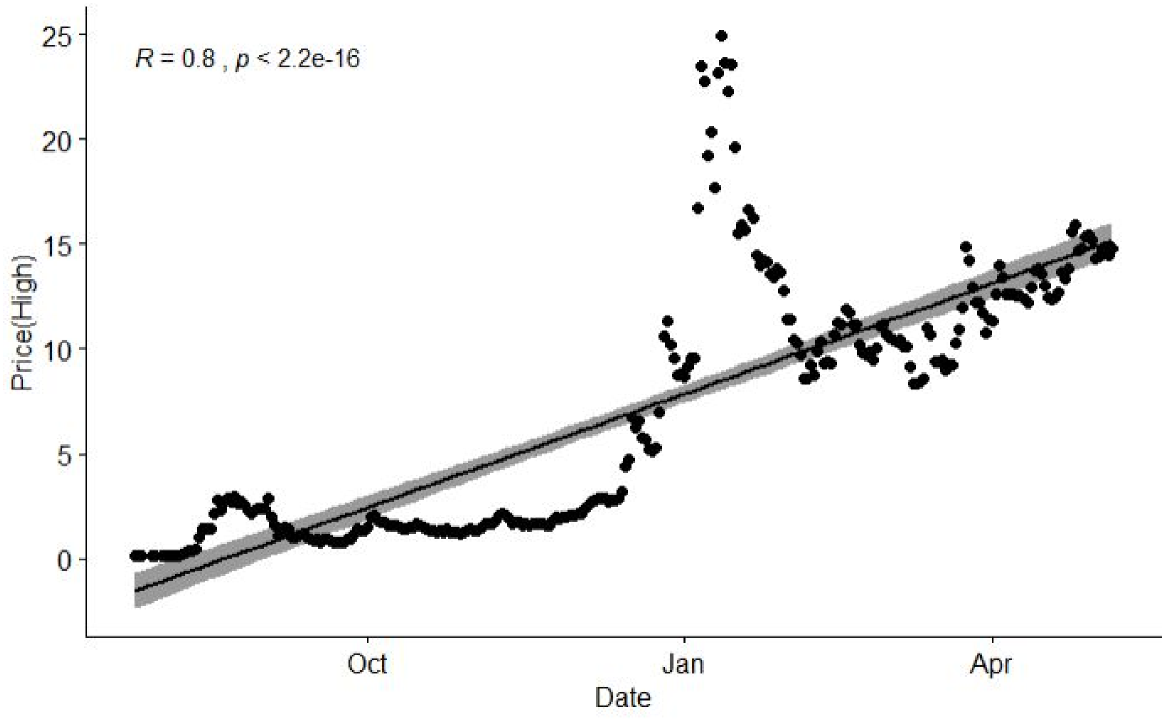


Figure XII: High vs Date plot for Pearson Correlation

- *Spearman Correlation:*

```
#SPEARMAN CO-RELATION
cor.test(df$tableJoin.High, df$tableJoin.Frequency, method = "spearman")
```

Spearman’s rank correlation rho

Data: df$tableJoin.High and df$tableJoin.Frequency

S = 1243100, p-value < 2.2e-16

Alternative hypothesis: true rho is not equal 0

Sample estimates:

Rho

0.7264976

Price:High vs Date plot for Spearman Correlation:

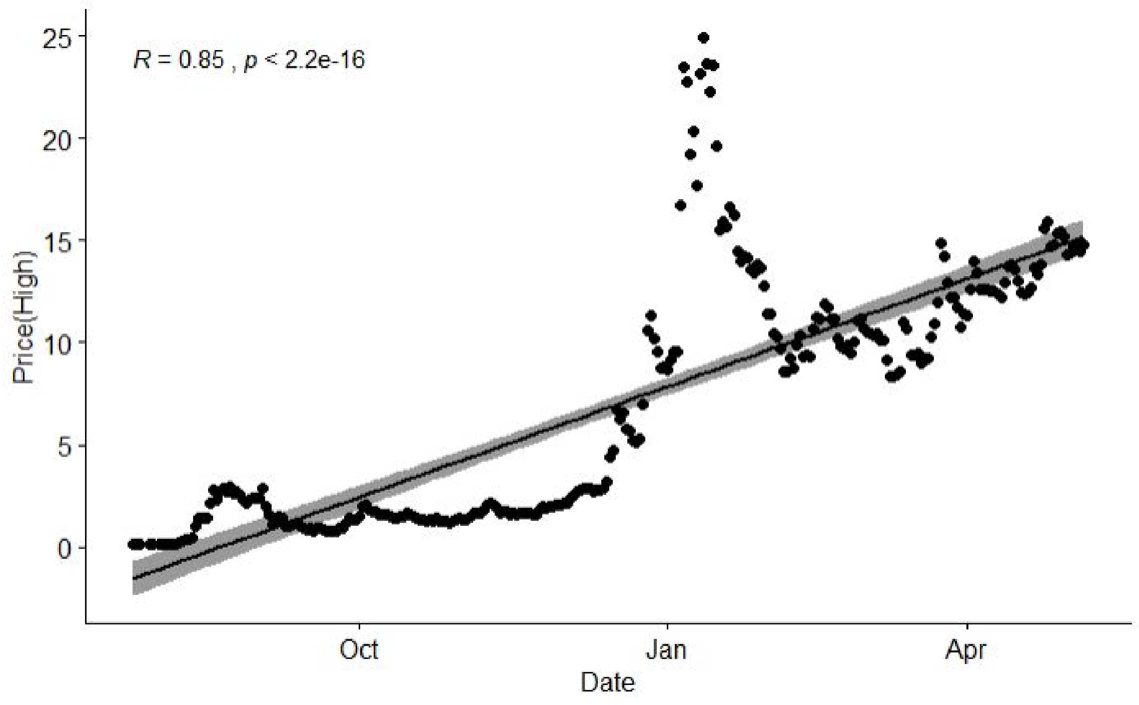


Figure XIII: High vs Date plot for Spearman Correlation

- *Kendall Correlation:*

```
#KENDALL C0-RELATION
cor.test(df$tableJoin.High, df$tableJoin.Frequency, method = "kendall")
```

Kendall’s rank correlation tau

Data: df$tableJoin.High abd df$tabkeJoin.Frequency

z = 13.517, p-value < 2.2e-16

Alternative hypothesis: true tau is not equal to 0

Sample estimates:

Tau

0.5234409

Price:High vs Date plot for Kendall Correlation:

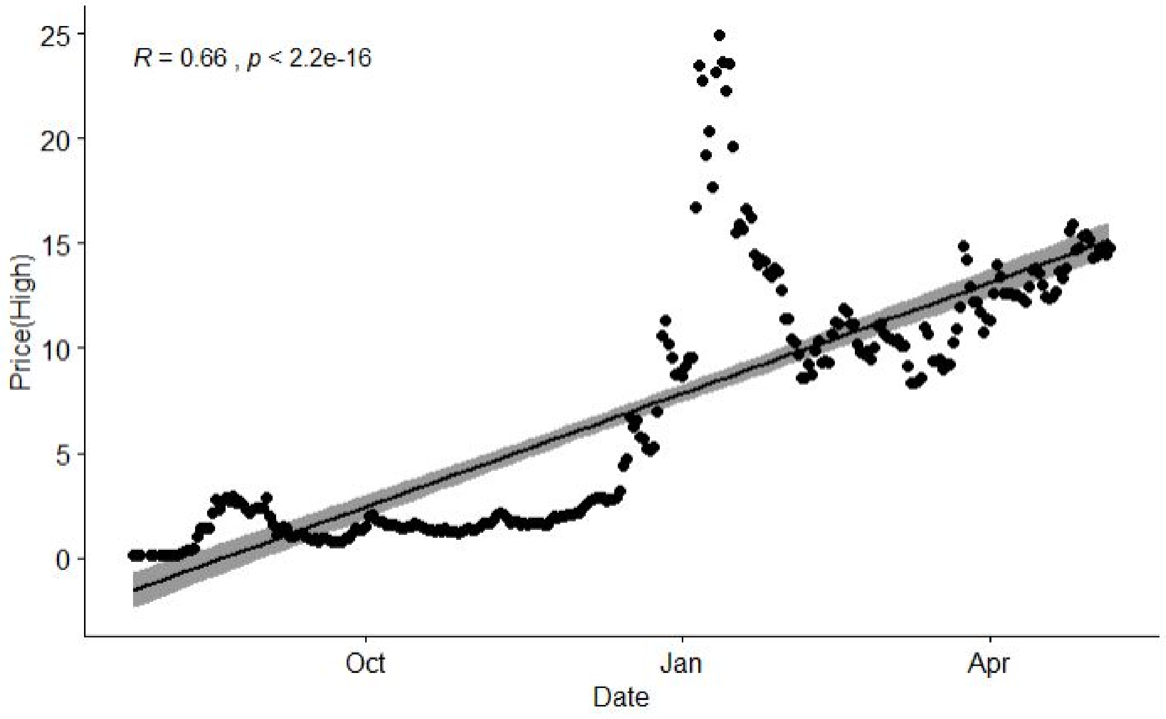


Figure XIV: High vs Date plot for Kendall Correlation

III. To predict price return on day 't' given the price on day 't-1' .

This part of the project requires us to predict the price return on day 't', given the price return on day 't-1'. We denote the price in dollar as P_t for day t, then the simple price return is given as:

$$(P_t - P_{t-1})/P_{t-1}$$

To predict values we need to find a relation between features and these relationships can be obtained via ***Regression Analysis.***

Regression Analysis: "In statistical modeling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed." *- Wikipedia*

Before we implement the regression models (linear or multilevel), we need to extract relevant features from the dataset. This process is called **Feature Engineering**. Excessive or redundant features may increase the runtime complexity of the predictive model which can affect the end results of the prediction.

The following approach was used to predict the return price using regression model:

We began by joining our two datasets i.e, the token graph (networdbnb) and the price (bnb). The next step was to extract features. We extracted four features:

- *Number of Unique Transactions on a given date*
- *Number of Unique Buyers on a given date*
- *Number of Unique Sellers on a given date*
- *Opening Price on a given date*

In the price return equation, we have used the closing price as the price parameter.

```
PriceReturn = (Close-lag(Close))/lag(close)
```

"R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination."[4]

We know that the R-squared value should lie between 0 and 1.

0% value indicates no variability of the response data around its mean, whereas, 100% value indicates all variability of the response data around its mean[4].

After calculating the price return, we started building our linear regression model. Starting off with only opening price as the regressor, we obtained the following results:

Multiple R-squared Value: 0.05984

Adjusted R-squared Value: 0.05971

Taking two regressors: opening price and number of unique buyers we obtained:

Multiple R-squared Value: 0.05989

Adjusted R-squared Value: 0.05963

There is a slight increase in the Multiple R-Squared Value.

Taking three regressors: opening price and number of unique transactions and buyers.

We obtain:

Multiple R-squared Value: 0.06044

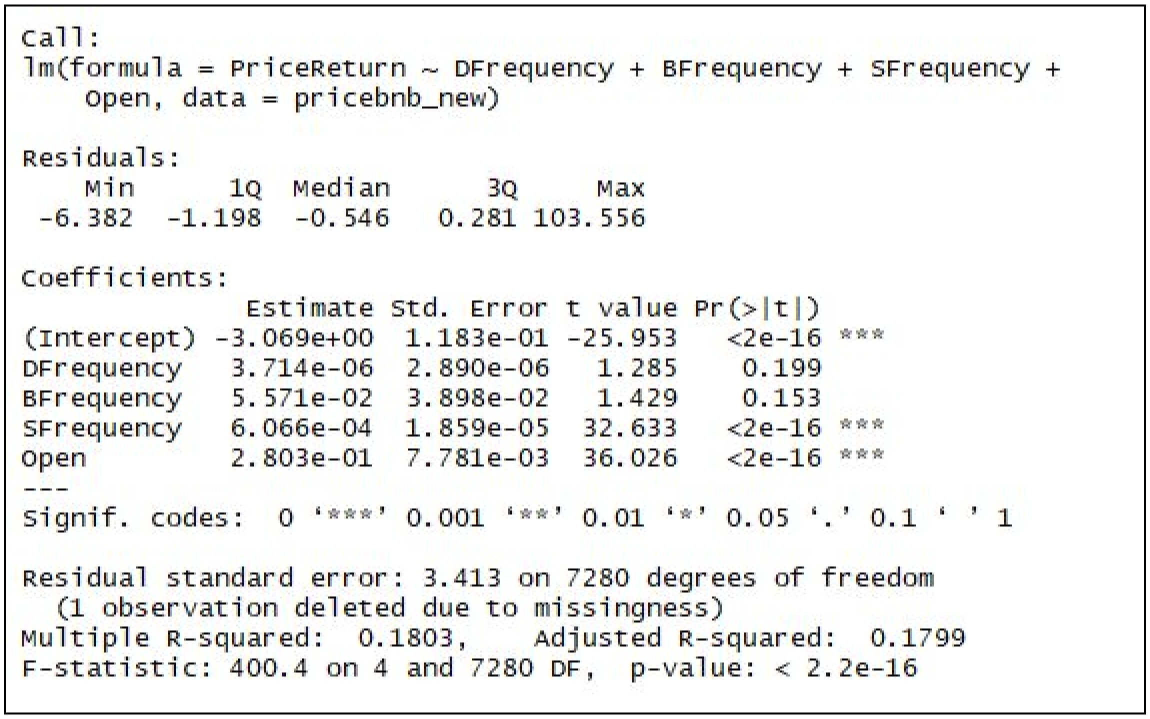
Adjusted R-squared Value: 0.06006

There is again an increase in values, though not by much.

When we use only one feature, the value obtained is very low. We observed that as we increase the number of regressors (features) in the model, the Multiple and Adjusted R-Squared value increases accordingly.

After trying out multiple combinations of regressors on our model, we obtained the highest Multiple and Adjusted R-Squared values of 0.1803 and 0.1799 respectively. The best combination of regressors we found is :

Open Price, Number of Unique Buyers, Sellers and Transactions.



```
Call:
lm(formula = PriceReturn ~ DFrequency + BFrequency + SFrequency +
    Open, data = pricebnb_new)

Residuals:
    Min      1Q  Median      3Q     Max
 -6.382  -1.198  -0.546   0.281 103.556

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.069e+00  1.183e-01 -25.953   <2e-16 ***
DFrequency   3.714e-06  2.890e-06   1.285    0.199
BFrequency   5.571e-02  3.898e-02   1.429    0.153
SFrequency   6.066e-04  1.859e-05  32.633   <2e-16 ***
Open         2.803e-01  7.781e-03  36.026   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.413 on 7280 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.1803,    Adjusted R-squared:  0.1799
F-statistic: 400.4 on 4 and 7280 DF,  p-value: < 2.2e-16
```

Figure XV: Linear Regression Model

Hence, our model can correctly determine the return price of the next day with approximately 18% accuracy.

Conclusion:

The following are the end results and inferences:

Part I: We observed that both the distributions fit our data pretty well. However, in Poisson distribution, mean and variance are both equal to lambda, which is not the observed case. Hence, we rejected Poisson and accepted Negative Binomial Distribution as best fit discrete distribution for our data.

Part II: The following are the results obtained:

Pearson : cor 0.5768044
Spearman: rho 0.8681768
Kendall: tau 0.6993524

According to above results, the Spearman Correlation gives us the highest coefficient value.

Part III: From the observations obtained, we can safely conclude that as we increase the number of regressors (features) in our model, the R-Squared value increases as well though not by much in some combinations, but when we used three regressors, there was a significant jump in the value. Since, we have obtained an R-square of 0.18 then we know that the variability of the Y values around the regression line is 1-0.18 times the original variance; in other words, we have explained 18% of the original variability, and are left with 82% residual variability. Ideally, we would like to explain most if not all of the original variability. The R-square value is an indicator of how well the model fits the data.

REFERENCES:

1. https://www.ethereum.org/
2. https://en.wikipedia.org/wiki/ERC-20
3. https://www.investopedia.com/terms/b/binance-coin-bnb.asp
4. http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit