BINANCE TOKEN DATA ANALYSIS

PROJECT – 1

SUBMITTED
BY

MANOHAR KATAM (MXK164930)

SAIPRAVEEN VABBILISETTY (SXV165130)

OCTOBER 31, 2018

UNIVERSITY OF TEXAS DALLAS

Table of Contents

1. INTRODUCTION 2

1.1 About Ethereum and ERC20 Tokens 2

1.2 About Binance (BNB) Coin 2

2. MOTIVE OF THE PROJECT 2

3. PREPROCESSING THE DATA 3

3.1 Data Description 3

3.1.1 Token edge data 3

3.1.2 Price data 3

3.1.3 Token supply and sub-unit definitions 3

3.2 Checking for Outliers 3

3.3 Checking for NA values 5

3.4 R Packages Used 5

4. ANALYSIS AND CONCLUSIONS 6

4.1 Buyer and Seller Frequency Distributions 6

4.1.1 Estimating best distribution for selling a token 6

4.1.2 Estimating best distribution for buying a token 7

4.2 Number of layers selection and Finding Correlation between the token and price data 9

4.2.1 Number of Layers Estimation 9

4.2.2 Finding Pearson Correlation between Features and Price Data 9

5. CONCLUSION 14

REFERENCES 15

APPENDIX 16

1. INTRODUCTION

1.1 About Ethereum and ERC20 Tokens

The main motto of the blockchain technologies is to eliminate the centralized system of transactions. The biggest disadvantage of current financial system which follows a centralized system to authorize transactions is the single point of failure. This has become the bottleneck to the global finance system. Blockchain, with no central point of failure and secured using cryptography, applications are well protected against hacking attacks and fraudulent activities. Immutability, Security, Shield against Corruption and tamper are its salient features. Ethereum enables the development of potentially thousands of different applications all on one platform. Blockchain is going to be the electricity of the future financial systems.

In Wikipedia, Ethereum is defined as "An open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality. It supports a modified version of Nakamoto consensus via transaction-based state transitions".

Reference: *https://en.wikipedia.org/wiki/Ethereum*

Ethereum is one of the most popular cryptocurrencies after Bitcoin. ERC-20 is the benchmark/technical standard used for smart contracts on the blockchain platform for implementation. ERC stands for Ethereum Request for Comment and 20 is the number assigned for that request. These rules include how the tokens are transferred between addresses and how data is accessed with in each token.

Reference: *https://en.wikipedia.org/wiki/ERC-20*

1.2 About Binance (BNB) Coin

The network token we chose is Binance Coin (BNB). Binanace is one of the most popular cryptocurrency exchange. The total supply is nearly 200 million. With this cryptocurrency, one can pay a commission for transactions on the exchange. The company offers wide variety of discounts for the customers who opt to pay commission. As per the stats on 28th October,2018, Market cap of this coin is 1.26 billion dollars and volume are $24 million with a current price of 9.63 USD.

2. MOTIVE OF THE PROJECT

The main motive of this project is to find the distribution model of token amounts and understand the pricing pattern of the Ethereum token. The second part of the project deals with extracting features such as layers of transactions, median of all the transactions for a given day which are highly correlated with price data. These features could be used as independent variables to predict the dependent variable i.e., closing price of the network token.

3. PREPROCESSING THE DATA

3.1 Data Description

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.

3.1.1 Token edge data

Token edge files have this row structure: **fromNodeID\ttoNodeID\tunixTime\ttokenAmount\r\n**

This row implies that fromNodeID sold tokenAmount of the token to toNodeID at time unixTime. fromNodeID and toNodeID are people who invest in the token in real life; each investor can also use multiple addresses. Two addresses can sell/buy tokens multiple times with multiple amounts. For this reason, the network is considered a weighted, directed multi(edge) graph. Each token has a maximum token count maxt; you can think of maxt as the total circulating token amount.

3.1.2 Price data

Price files have no extensions, but they are text based. If you open them with a text editor (use notepad++ or similar), you will see this row structure: Date\tOpen\tHigh\tLow\tClose\tVolume\tMarketCap\r

The price data is taken from https://coinmarketcap.com/. Open and close are the prices of the specific token at the given date. Volume and MarketCap give total bought/sold tokens and market valuation at the date.

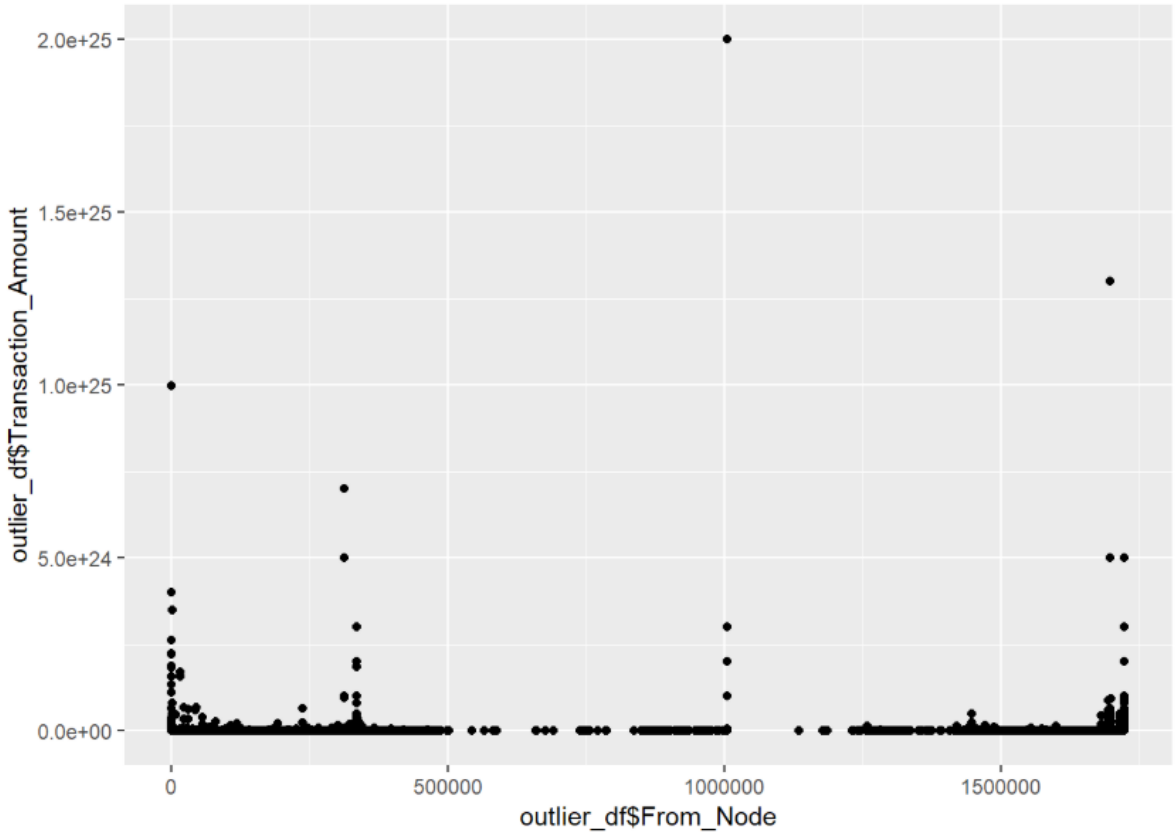
3.1.3 Token supply and sub-unit definitions

Token has a limited supply (i.e., token count, which can be found on coinmarketcap.com as circulating amount). Then each token may have sub-units. This idea comes from Bitcoin where subunits are called Satoshis, 1 Bitcoin $=10^8$ satoshis. Coin market cap gives the total supply, but not sub-units, which differ from token to token. Some tokens have 10^{18} sub-units. That means there can be numbers as big as $\text{totalAmount} * 10^{18}$.

3.2 Checking for Outliers

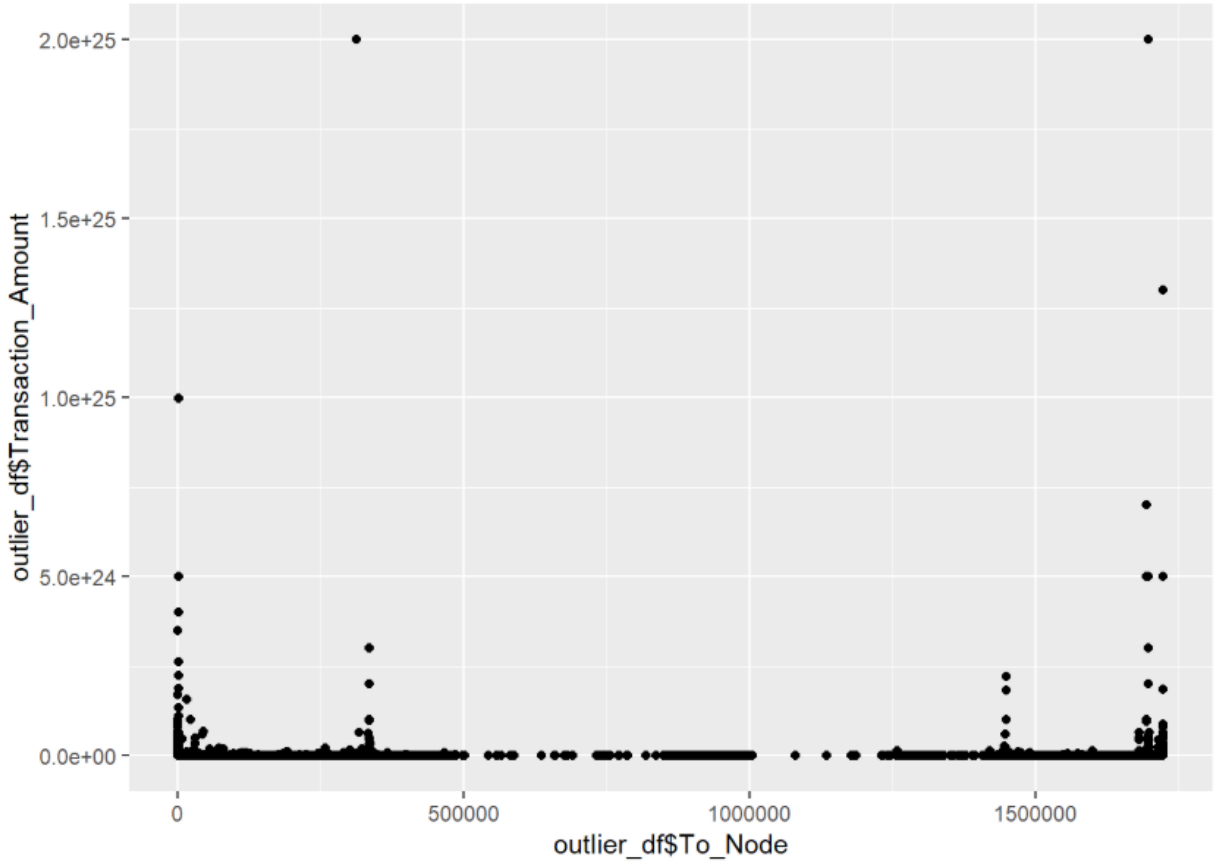
It is found in most of the articles that there can be some noise values in the dataset which would be later removed by the block, but this still has an entry in the dataset.

So, the initial steps of building our project involved in looking for outliers, to be precise looking for transactions whose token amount values are greater than 10^{26}.



Above is the graph for seller to transaction amount. We see that none of the points are outside the range of $1.92x10^{26}$. So, that we conclude here that there are no outliers for seller's vs the transaction amount.

Similarly, in the below graph of buyer's vs transaction amount too there are no outliers.



3.3 Checking for NA values

As another step in preprocessing we must check for NA or N/A values and remove them if we have enough data or we can impute them mean of our data or average of its previous and next value. This step-in preprocessing is called Imputation (i.e., imputing the missing or NA data with meaningful values)

To do this in R we have na.remove = TRUE.

3.4 R Packages Used

Before going any further, here we will describe the R packages we used.

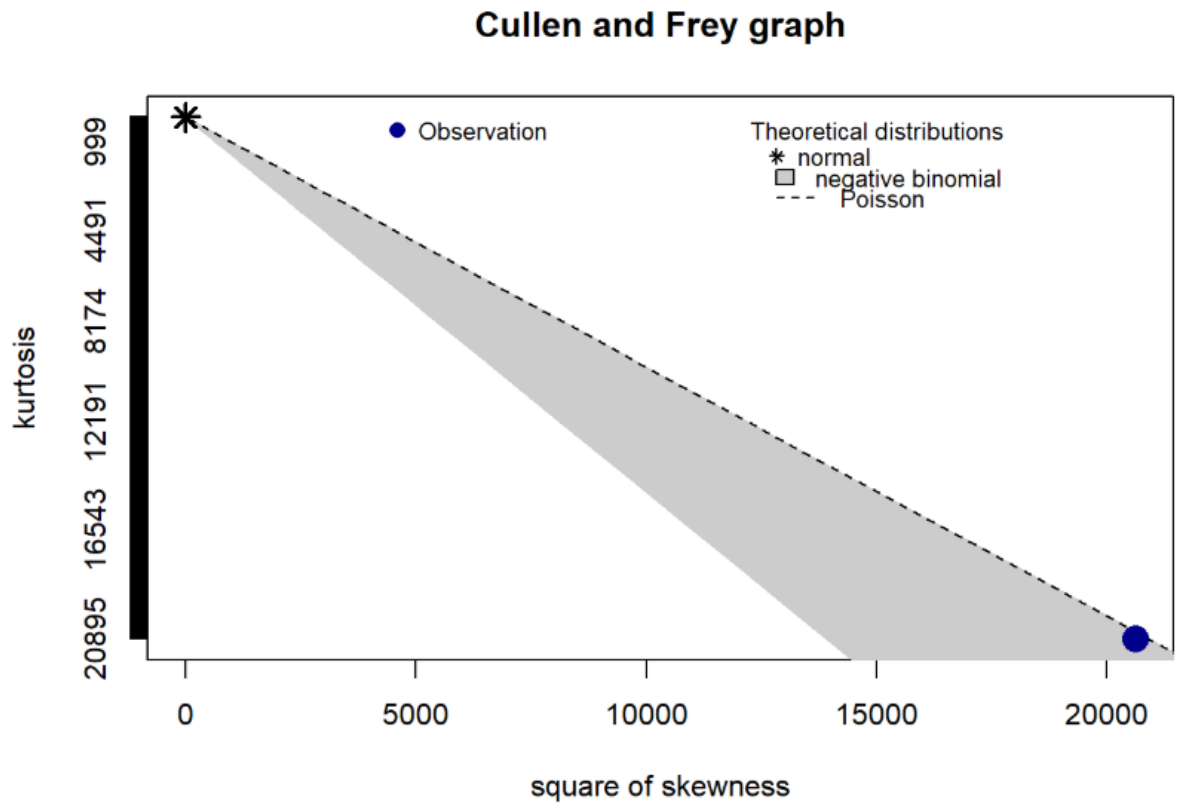
Package Name	Description
ggplot2	package to plot data visualizations
imputeTS	package to replace the "NA" values
plyr	package to implement count
dplyr	package for filtering the data
fitdistrplus	package to fit distribution

4. ANALYSIS AND CONCLUSIONS

4.1 Buyer and Seller Frequency Distributions

4.1.1 Estimating best distribution for selling a token

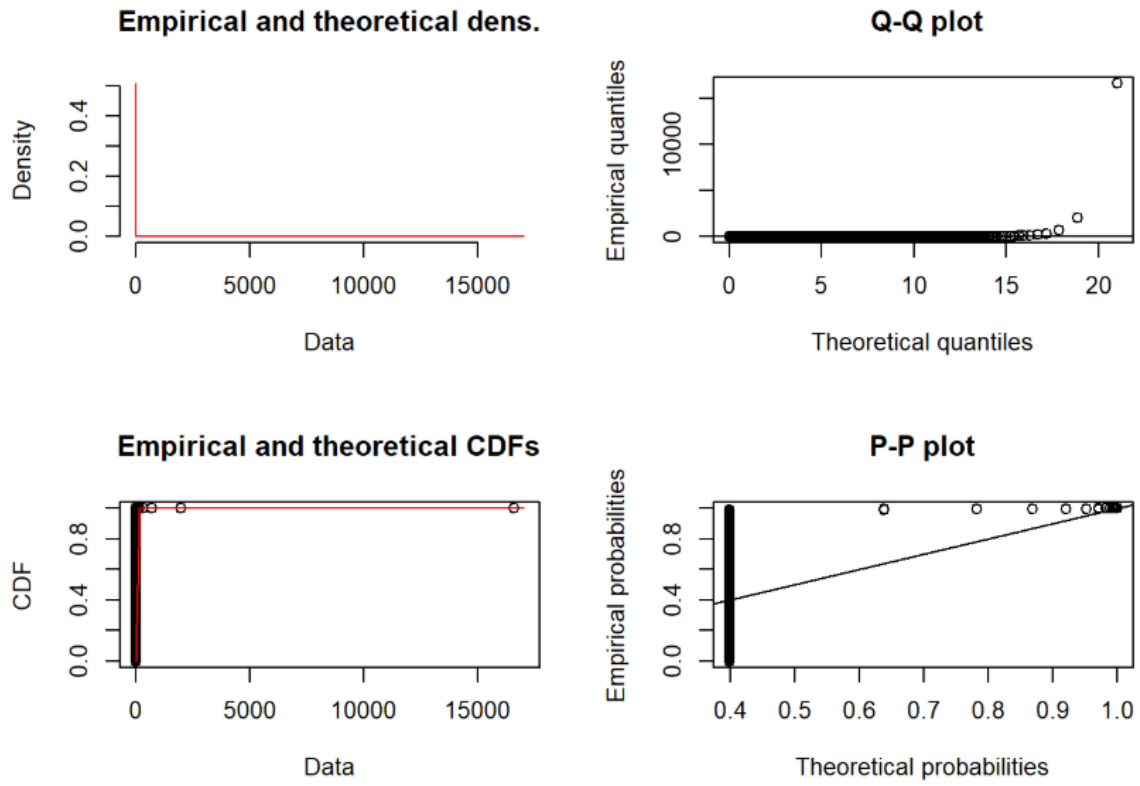
Best distribution for frequency of selling a token:



The above graph gives us an intuition that distribution of data is close to exponential, but Poisson distribution doesn't apply to our case. So the data is fit to Exponential distribution.

Estimated parameters:

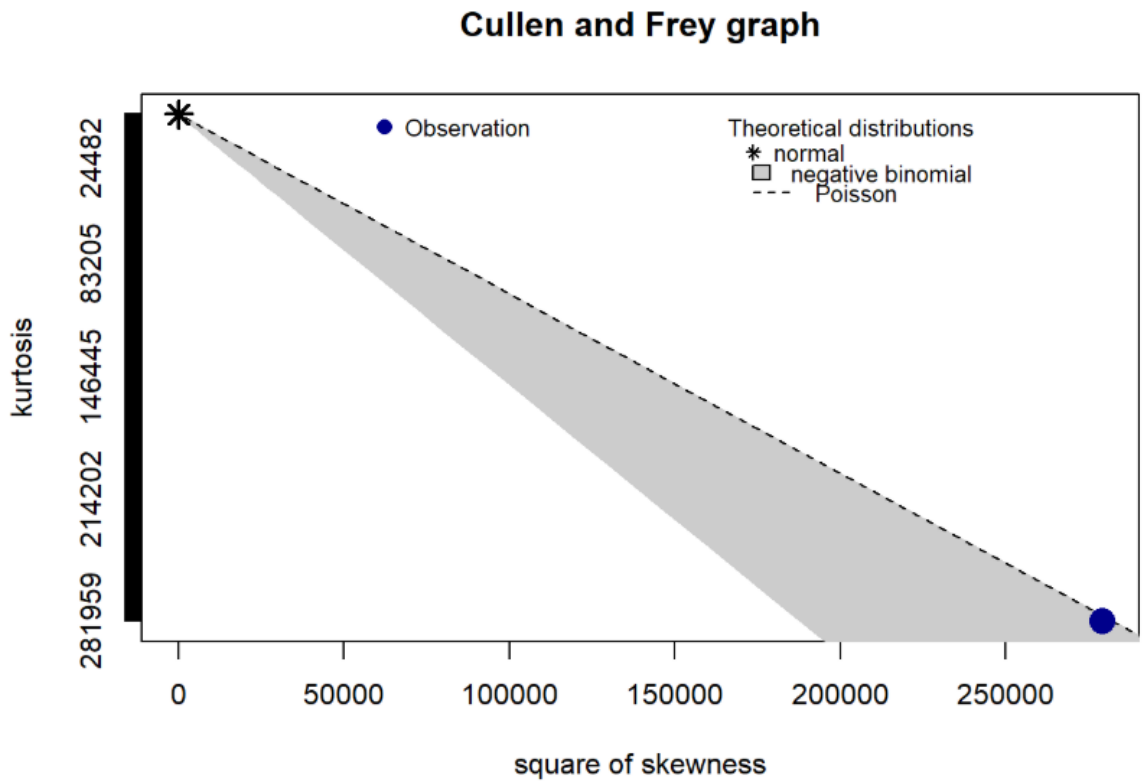
```
## summary statistics
## ------
## min:  1   max:  16575
## median:  1
## mean:  1.968041
## estimated sd:  113.7337
## estimated skewness:  143.5891
## estimated kurtosis:  20894.34
```



If we observe the QQ Plot, the theoretical and Empirical limits are almost the same and gives us an intuition how the data is distributed (exponential)

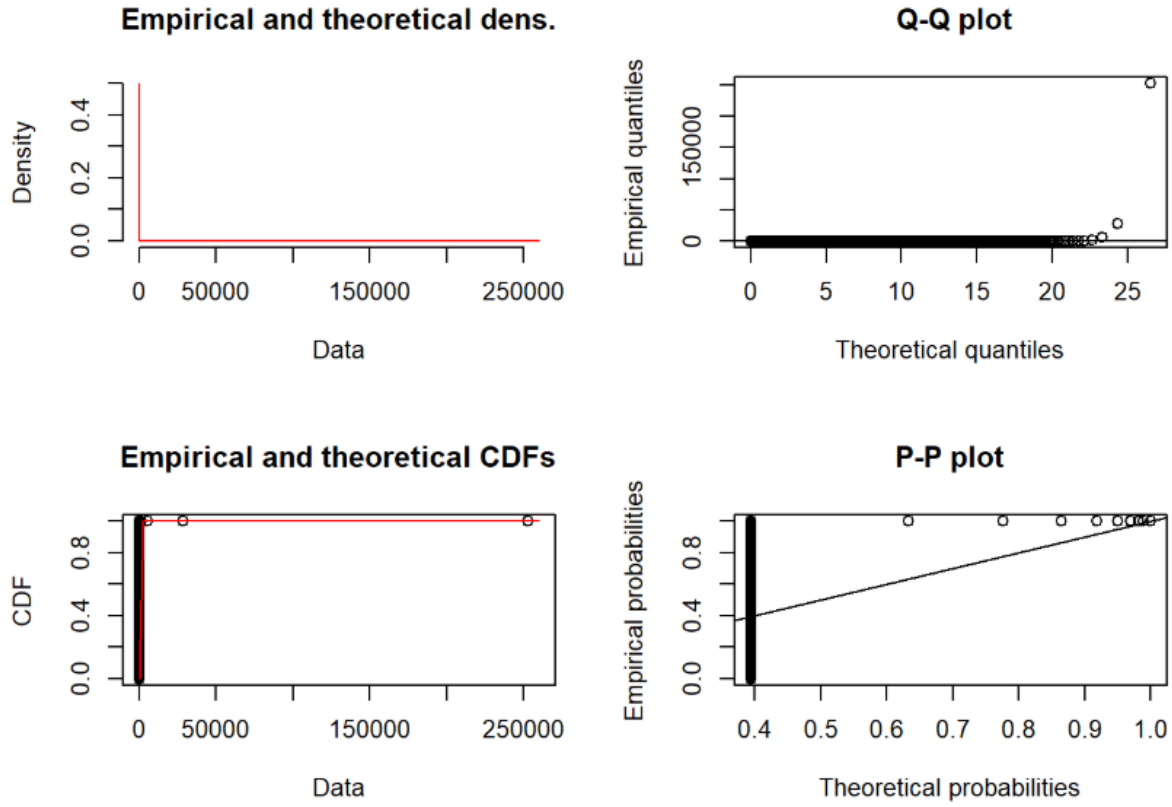
4.1.2 Estimating best distribution for buying a token

Best distribution for frequency of buying a token:



Estimated Parameters:

```
## summary statistics
## ------
## min:  1   max:  252994
## median:  1
## mean:  1.999661
## estimated sd:  473.4125
## estimated skewness:  528.2734
## estimated kurtosis:  281958.7
```



4.2 Number of layers selection and Finding Correlation between the token and price data

The main motive behind dividing into layers of transactions is to approximate the exponential distribution in each of the layers.

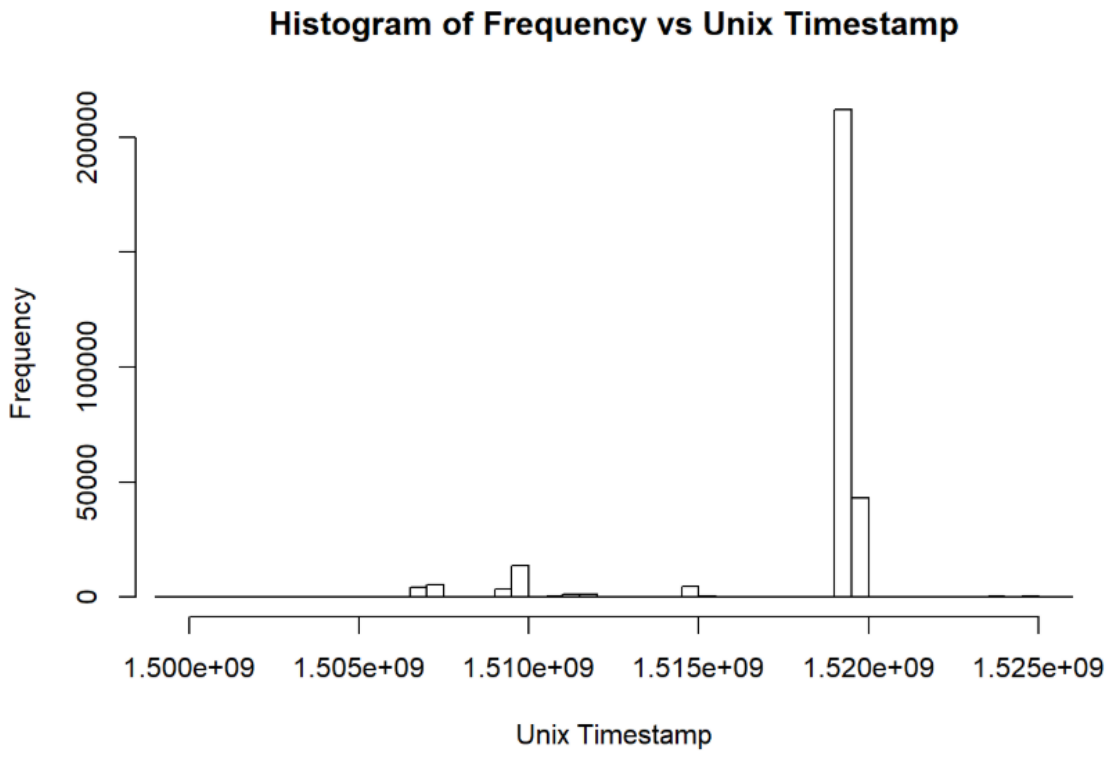
4.2.1 Number of Layers Estimation

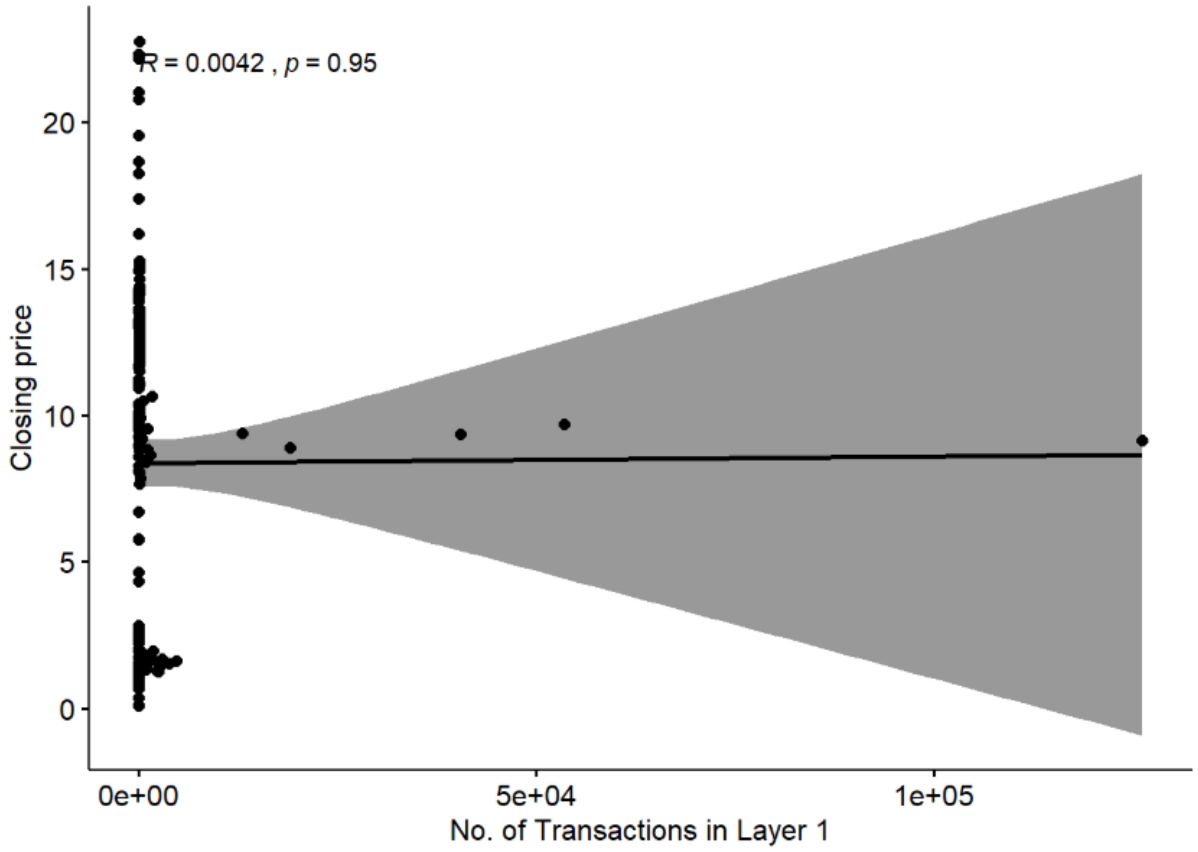
The total token data is divided into three layers based upon density of transactions. The first layer consists of transactions/token amounts less than 1 binance coin, second layer consists of transactions between 1 and 1000 binance coins and the last layer consists of transactions which are greater than 1000 binance coins. The model found that the closing price of the token depends upon the number of second layer transactions on that day.

4.2.2 Finding Pearson Correlation between Features and Price Data

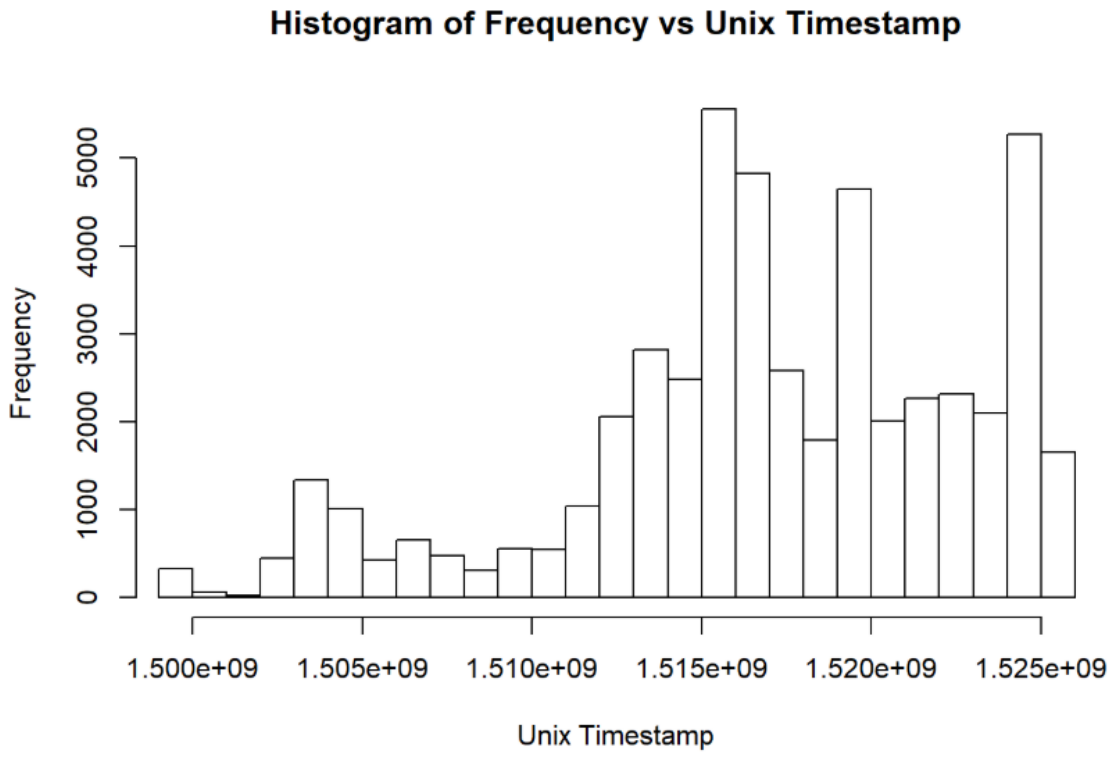
The important findings that are found on implementing this project are the token amount of Binance network token follow an exponential distribution, the closing price of the token for a given day is inversely correlated (negative correlated) to median of token amounts (normalized) and positively correlated (0.6) to number of layer two transactions for a given day.

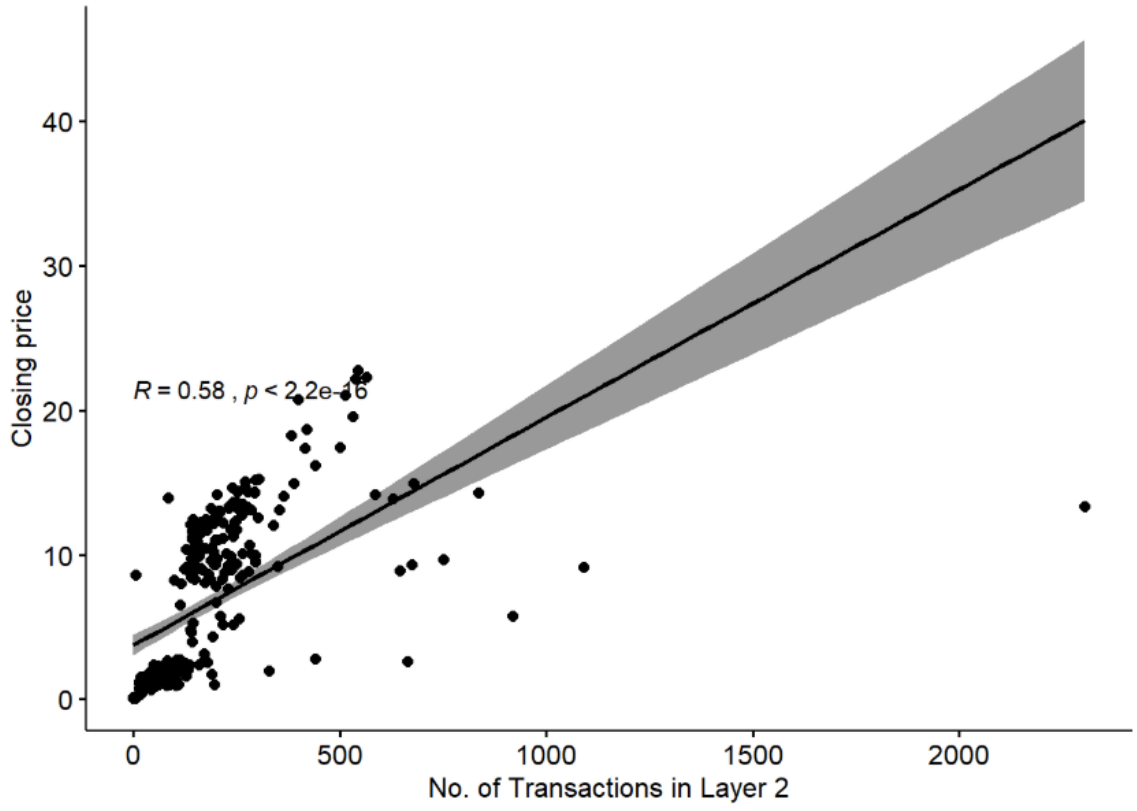
Layer1: Less than 0.817xtotal number of transactions



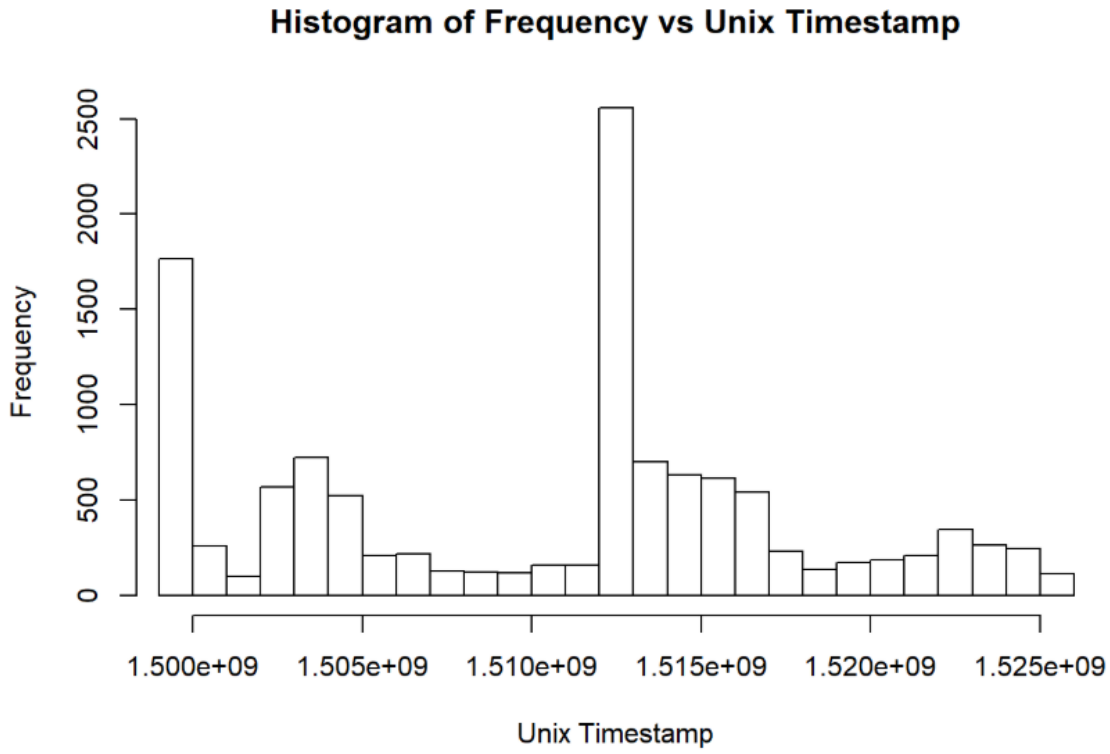


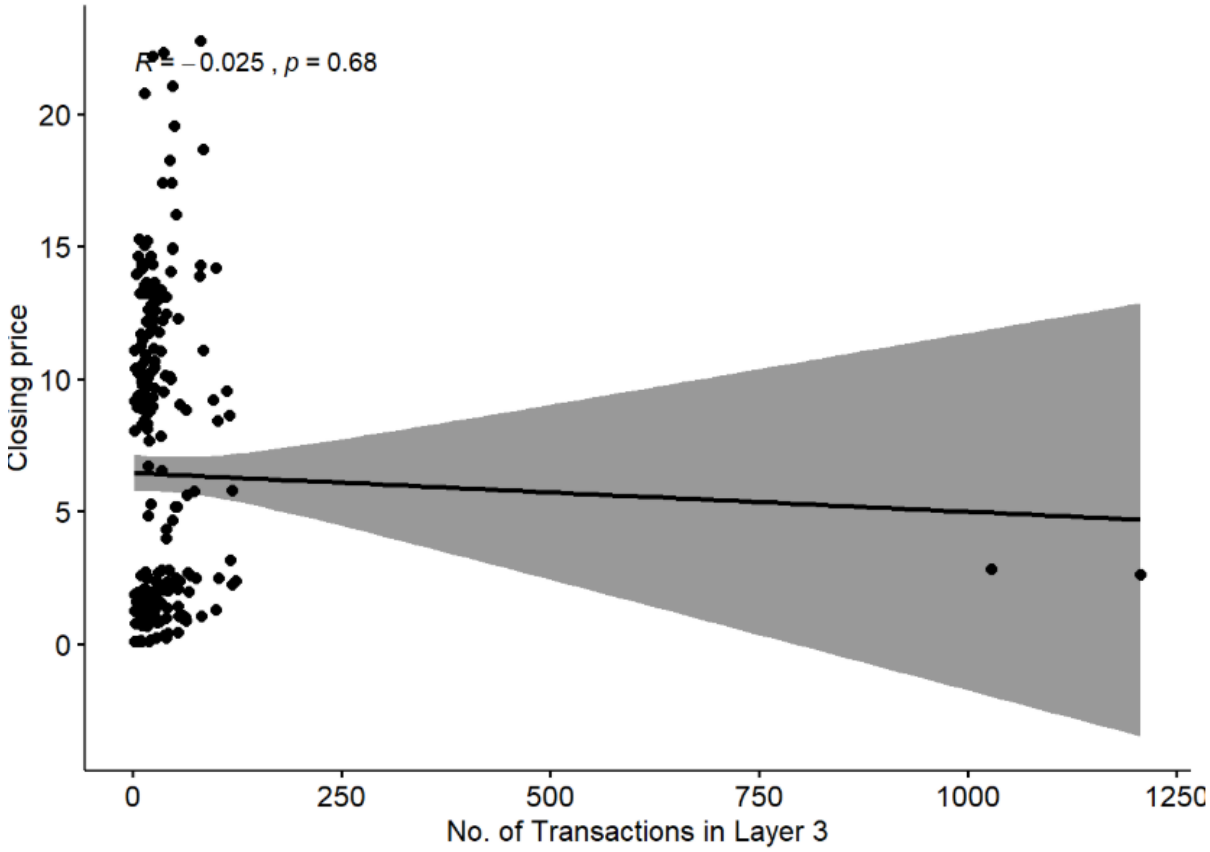
Layer2: 0.817xtotal number of transactions < n < 0.96xtotal number of transactions



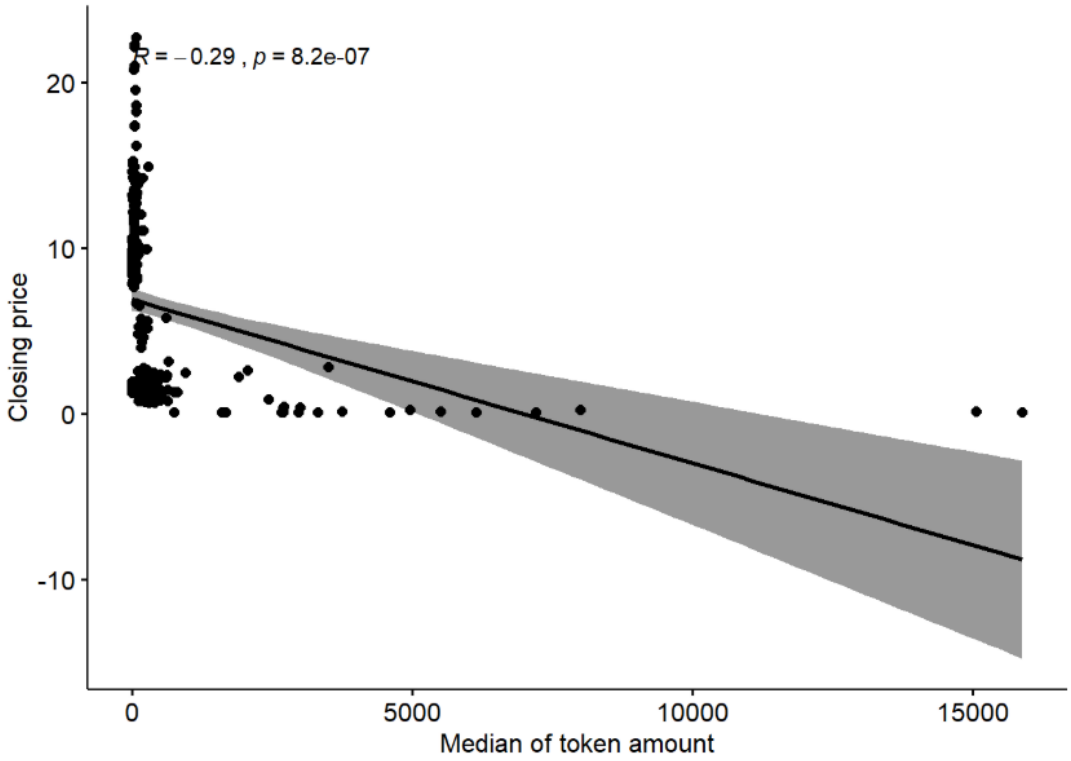


Layer3: Greater than 0.96xnumber of transactions

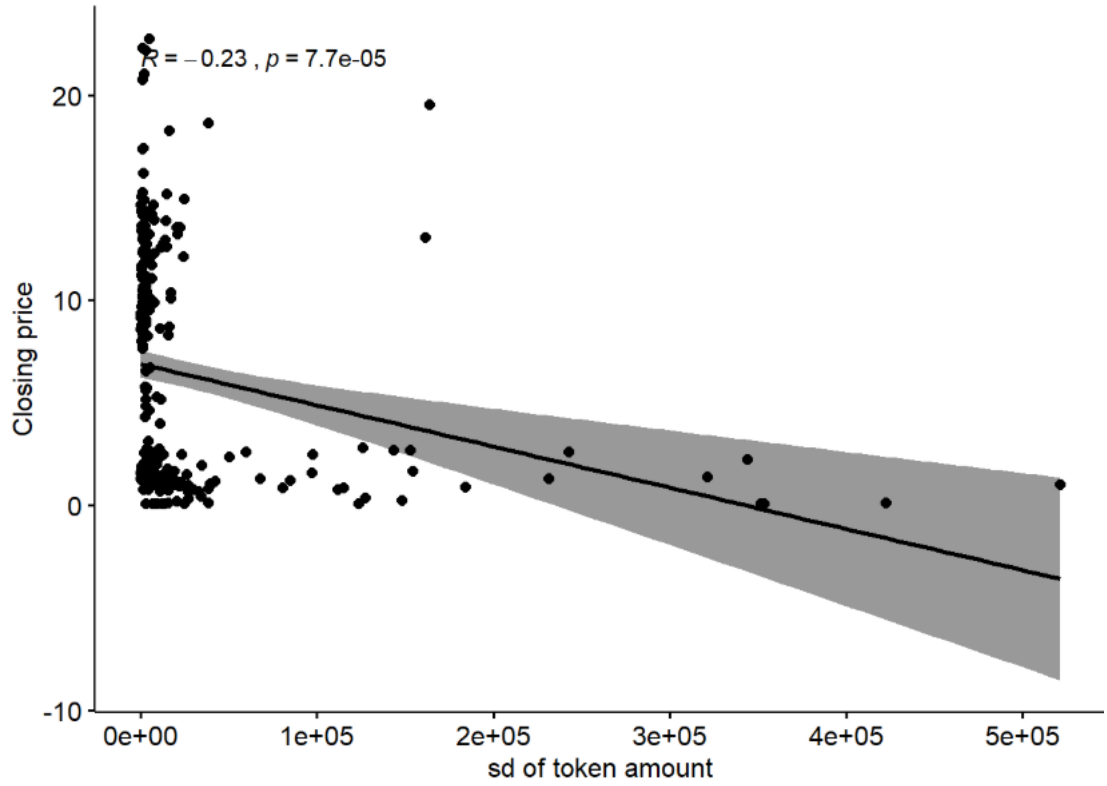




Correlation with Median of token amounts:



Correlation with standard deviation of token amounts:



Findings From above Layers:

The most significant features which contributes to prediction of closing token price from our experiments are

1.The number of Layer 2 transaction (1 to 1000 binance coins) with a positive correlation of 0.58.

2.The Median of token amounts for a given day with a negative correlation of -0.29

3.The Standard Deviation of token amounts for a given day with a negative correlation of -0.23

5. CONCLUSION

Frequency of Buys and Sells: We could see that from the above plots the token amounts are distributed exponentially. This is a feasible distribution because as the feature importance grows on increasing day by day, the token amounts being increased in the same way. For Example, a country 'X' has allowed crypto exchange as their official exchange there is a high chance the token amounts would increase.

Number layers and correlation: From the above experiments, results conclude that the number of layer-2 transactions, Median of token amounts, Standard deviation of token amounts for a given day could be the most significant features which contributes the maximum variance to the distribution. These results could be used in building a multi/polynomial regression model and could be used in predicting the price of Binance Closing price for a given day.

REFERENCES

1. *https://en.wikipedia.org/wiki/Ethereum*
2. *https://en.wikipedia.org/wiki/ERC-20*
3. https://coinmarketcap.com/currencies/binance-coin/historical-data/
4. https://www.binance.com/en
5. https://www.rdocumentation.org/packages/ggplot2/versions/3.1.0
6. http://www.di.fc.ul.pt/~jpn/r/distributions/fitting.html
7. https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

APPENDIX

Question 1: Find the distribution of how many times a user 1 - buys, 2 - sells a token. Which discrete distribution type fits these distributions best? Estimate distribution parameters.

Code:

```
---
title: "Project - 1: Buyer and Seller Distribution for Binanace Network Token"
author: "Saipraveen Vabbilisetty(sxv165130) Manohar Katam(mxk164930)"
date: "October 10, 2018"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

Binance Coin

Binanace is one of the top CryptoCurrency Exchanges, headed by Changpeng Zhao and a team distributed around the world.It can support 1,400,000 orders per second.Binance does not support physical currency and focuses purely on crypto-to-crypto.

```{r package to implement count}
library(plyr)
```

```{r package to fit distribution}
library(fitdistrplus)
```


```{r package to plot data visualizations}
library(ggplot2)
```


```{r package to replace the "NA" values}
library(imputeTS)
```

```{r package for filtering}
library(dplyr)
```
```

Reading Token Data

```{r}
setwd("C:\\Users\\makatam\\Desktop\\STATS\\Ethereum token graphs")
input_data <- read.delim('networkbnbTX.txt',sep = ' ',header = FALSE)
names(input_data)<- c("From_Node","To_Node","Unix_Time","Transaction_Amount")
head(input_data)
```

Estimating Outlier Data
```{r}
outlier_df <- as.data.frame(input_data)
threshold <- 10**26
totalSupply <- 1.92
outlier_df1<-filter(outlier_df,outlier_df$Transaction_Amount > totalSupply * threshold)
nrow(outlier_df1)
```

Scatter plots to check outlier data
```{r}
qplot(outlier_df$From_Node, outlier_df$Transaction_Amount)
```

```{r}
qplot(outlier_df$To_Node, outlier_df$Transaction_Amount)
```
Here we observe that there are no outliers in the data.

Converting the Token Data Vector to a data frame which increases the speed of data operations

```{r}
input_df<-data.frame(unclass(plyr :: count(input_data,vars = "From_Node")))
head(input_df)
```

Calculating Frequency of Frequencies for Sellers
```{r}
frequency_df<-data.frame(data.frame(table(unlist(input_df))))
names(frequency_df)<- c("Selling_Frequency","Frequency_of_Selling_Frequency")
summary(frequency_df)
qplot(frequency_df$Selling_Frequency,frequency_df$Frequency_of_Selling_Frequency)
```

Trying to find the best distribution for Seller Data

```{r}
descdist(frequency_df$Frequency_of_Selling_Frequency, discrete = TRUE)
fit <- fitdist(frequency_df$Frequency_of_Selling_Frequency, "exp")
plot(fit)
```

Calculating Frequencies for Buyers
```{r}
buyer_df<-data.frame(unclass(plyr :: count(input_data,vars = "To_Node")))
head(buyer_df)
```

Calculating Frequency of Frequencies
```{r}
buyer_df_freq<-data.frame(table(unlist(buyer_df)))
names(buyer_df_freq)<- c("Buying_Frequency","Frequency_of_Buying_Frequency")
summary(buyer_df_freq)
qplot(buyer_df_freq$Buying_Frequency,buyer_df_freq$Frequency_of_Buying_Frequency)
```

Fitting The Buyer Frequency Distribution
```{r}
descdist(buyer_df_freq$Frequency_of_Buying_Frequency, discrete = TRUE)
fit2 <- fitdist(buyer_df_freq$Frequency_of_Buying_Frequency, "exp")
plot(fit2)
```

Question 2: How can we create layers of transactions with increasing amounts? This descriptive statistic is similar to bin selection in histograms. For example, we could choose layer1 as those transactions that involve 0.01×maxt in amount. Find a good value for the number of layers and justify your choice.

Once you create layers, you can compute a feature in each layer. An example feature is the number of transactions, another one is the number of unique buyers. As each edge has a unix timestamp, it is easy to compute the edge time to a date. For example, 1294226315 is equivalent to 01/05/2011 @ 11:18am (UTC). See the website https://www.unixtimestamp.com/index.php for unix time conversion. R has functions to compute dates from unix time stamps as well. This way, for a given day you can find all layer transactions in that day.

For example, you can say on 10/12/2018 there were 25 transactions in layer 1. The price of token on that date was 3.2$. For each day in a token's history, you can then correlate price vs feature in time.

Find an algorithm to compute the correlation of price data with each of the layers (hint: start by looking at Pearson correlation).

Code:

---
title: "Project - 1"

```
author: "Saipraveen Vabbilisetty(sxv165130) Manohar Katam(mxk164930)"
date: "October 20, 2018"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

Reading Token Data

```{r}
# Loading all packages
# KernSmooth to find out number of bins
library(KernSmooth)
library(plyr)
library(dplyr)
library(ggplot2)

# For Date conversions
library(anytime)

# For plotting correlations
library(ggpubr)
```

```{r}
setwd("C:\\Users\\makatam\\Desktop\\STATS\\Ethereum token graphs")
input_data <- read.delim('networkbnbTX.txt',sep = ' ',header = FALSE)
names(input_data)<- c("From_Node","To_Node","Unix_Time","Transaction_Amount")
input_df<- as.data.frame(input_data,na.rm=TRUE)
head(input_df)
```

Dropping From Node and To Node as we don't need agrregate them
```{r}
drops <- c("From_Node","To_Node")
input_df<- input_df[ , !(names(input_df) %in% drops)]
```

Convert unix time stamp to date
```{r}
input_df$Unix_Time <- as.Date(as.POSIXct(as.numeric(input_df$Unix_Time), origin = '1970-01-01', tz = 'UTC'))
```
```

```

*# Load Price Data*
```{r}
setwd("C:\\Users\\makatam\\Desktop\\STATS\\tokenPrices")
price_data <- read.delim('binance.txt',sep = '\t',header = TRUE)
names(price_data)<- c("Date","Open","High","Low","Close","Volume","MarketCap")
price_df <- as.data.frame(price_data, na.rm=TRUE)
```

 # Read Date string (ex: Oct 10, 2018) to Date
```{r}
price_df$Date <- as.Date(price_df$Date, "%B %d,%Y")
```

 # Normalizing token amounts
```{r}
multiplier<-10**18
input_df$Transaction_Amount<-input_df$Transaction_Amount/multiplier
head(input_df)
```

 # Dividing the transactions into three layers
```{r}
threshold1 = 10**0
threshold2 = 10**3
threshold3 = 10**8
```

*# Layer 1: 0.817*total number of transactions*
```{r}
filter1<-filter(input_df,input_df$Transaction_Amount < threshold1)
summary(filter1)
```
```{r}
nrow(filter1)
```

 # Plotting the histogram to understand the distribution
```{r}
h1 <- dpih(as.numeric(as.POSIXct(filter1$Unix_Time, format="%Y-%m-%d")))
bins <- seq(min(as.numeric(as.POSIXct(filter1$Unix_Time, format="%Y-%m-%d"))),
        max(as.numeric(as.POSIXct(filter1$Unix_Time, format="%Y-%m-%d")))+h1,
        by=h1)
```
```

```
h1_plot<-hist(as.numeric(as.POSIXct(filter1$Unix_Time, format="%Y-%m-%d")),
          xlim = c(min(as.numeric(as.POSIXct(filter1$Unix_Time, format="%Y-%m-%d"))),
                max(as.numeric(as.POSIXct(filter1$Unix_Time, format="%Y-%m-%d")))),
          breaks=50, xlab = "Unix Timestamp", main = paste("Histogram of Frequency vs Unix Timestamp"))
```

Group by date and count number of transactions present in layer 1
```{r}
filter1_df = as.data.frame(plyr::count(anydate(filter1$Unix_Time)))
names(filter1_df)<-c('Date','Count1')
head(filter1_df)
```

Joining the price data with token data on Date
```{r}
combined_df1 <- inner_join(filter1_df,price_df,by=c('Date'))
head(combined_df1)
```

Calculating correlation
```{r}
ggscatter(combined_df1, x = "Count1", y = "Close",
        add = "reg.line", conf.int = TRUE,
        cor.coef = TRUE, cor.method = "pearson",
        xlab = "No. of Transactions in Layer 1", ylab = "Closing price")
res <- cor.test(combined_df1$Count1, combined_df1$Close,
                method = "pearson")
```

Layer 2: 0.817*total number of transactions < n < 0.96*total number of transactions
```{r}
filter2<-filter(input_df,(input_df$Transaction_Amount > threshold1 & input_df$Transaction_Amount < threshold2))
summary(filter2)
nrow(filter2)
```
```{r}
h2 <- dpih(as.numeric(as.POSIXct(filter2$Unix_Time, format="%Y-%m-%d")))
bins <- seq(min(as.numeric(as.POSIXct(filter2$Unix_Time, format="%Y-%m-%d"))),
          max(as.numeric(as.POSIXct(filter2$Unix_Time, format="%Y-%m-%d")))+h2,
          by=h2)
h2_plot<-hist(as.numeric(as.POSIXct(filter2$Unix_Time, format="%Y-%m-%d")),
           xlim = c(min(as.numeric(as.POSIXct(filter2$Unix_Time, format="%Y-%m-%d"))),
                 max(as.numeric(as.POSIXct(filter2$Unix_Time, format="%Y-%m-%d")))),
```

```
        breaks=30, xlab = "Unix Timestamp", main = paste("Histogram of Frequency vs Unix Timestamp"))
```

Group by date and count number of transactions present in layer 2
```{r}
filter2_df = as.data.frame(plyr::count(anydate(filter2$Unix_Time)))
names(filter2_df)<-c('Date','Count2')
```

Joining the price data with token data on Date
```{r}
combined_df2 <- inner_join(filter2_df,price_df,by=c('Date'))
head(combined_df2)
```

Calculating correlation
```{r}
ggscatter(combined_df2, x = "Count2", y = "Close",
      add = "reg.line", conf.int = TRUE,
      cor.coef = TRUE, cor.method = "pearson",
      xlab = "No. of Transactions in Layer 2", ylab = "Closing price")
res2 <- cor.test(combined_df2$Count2, combined_df2$Close,
            method = "pearson")
```

Layer 3: > 0.96* number of transactions
```{r}
filter3<-filter(input_df,(input_df$Transaction_Amount > threshold2 & input_df$Transaction_Amount < threshold3))
summary(filter3)
nrow(filter3)
```

```{r}
h3 <- dpih(as.numeric(as.POSIXct(filter3$Unix_Time, format="%Y-%m-%d")))
bins <- seq(min(as.numeric(as.POSIXct(filter3$Unix_Time, format="%Y-%m-%d"))),
        max(as.numeric(as.POSIXct(filter3$Unix_Time, format="%Y-%m-%d")))+h3,
        by=h3)
h3_plot<-hist(as.numeric(as.POSIXct(filter3$Unix_Time, format="%Y-%m-%d")),
         xlim = c(min(as.numeric(as.POSIXct(filter3$Unix_Time, format="%Y-%m-%d"))),
              max(as.numeric(as.POSIXct(filter3$Unix_Time, format="%Y-%m-%d")))),
         breaks=30, xlab = "Unix Timestamp", main = paste("Histogram of Frequency vs Unix Timestamp"))
```

```{r}
filter3_df = as.data.frame(plyr::count(anydate(filter3$Unix_Time)))
names(filter3_df)<-c('Date','Count3')
```

Joining the price data with token data on Date
```{r}
combined_df3 <- inner_join(filter3_df,price_df,by=c('Date'))
head(combined_df3)
```

Calculating correlation
```{r}
ggscatter(combined_df3, x = "Count3", y = "Close",
        add = "reg.line", conf.int = TRUE,
        cor.coef = TRUE, cor.method = "pearson",
        xlab = "No. of Transactions in Layer 3", ylab = "Closing price")
res3 <- cor.test(combined_df3$Count3, combined_df3$Close,
                 method = "pearson")
```

Groupby date and calculate the Median token amount
```{r}
input_median_df<- aggregate(input_df$Transaction_Amount,by=list(input_df$Unix_Time),median)
names(input_median_df)<-c("Date","Median")
input_median_df1<-as.data.frame(input_median_df)
```

```{r}
combined_df_median<-inner_join(input_median_df1,price_df,by=c('Date'))
ggscatter(combined_df_median, x = "Median", y = "Close",
        add = "reg.line", conf.int = TRUE,
        cor.coef = TRUE, cor.method = "pearson",
        xlab = "Median of token amount", ylab = "Closing price")
res_Median<-cor.test(combined_df_median$Median,combined_df_median$High,method="pearson")
```

Groupby date and calculate the standard deviation token amount
```{r}
input_sd_df<- aggregate(input_df$Transaction_Amount,by=list(input_df$Unix_Time),sd)
names(input_sd_df)<-c("Date","sd")
input_sd_df1<-as.data.frame(input_sd_df)
```

```{r}
combined_df_sd<-inner_join(input_sd_df1,price_df,by=c('Date'))
ggscatter(combined_df_sd, x = "sd", y = "Close",
        add = "reg.line", conf.int = TRUE,
        cor.coef = TRUE, cor.method = "pearson",
        xlab = "sd of token amount", ylab = "Closing price")
res_Median<-cor.test(combined_df_sd$sd,combined_df_sd$High,method="pearson")
```

END OF PROJECT REPORT