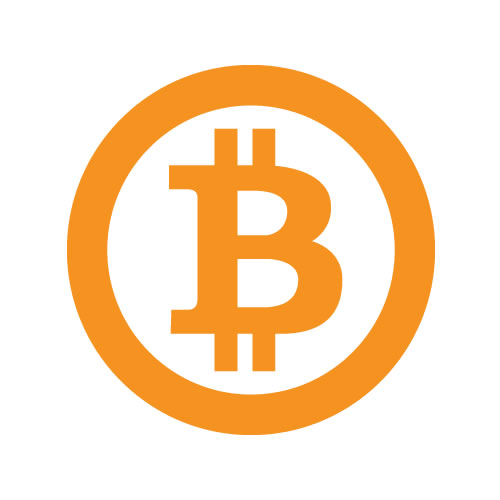
Bitcoin Price Prediction and Transaction Network Analysis

Applying machine learning models to predict the price of bitcoin and analyze the transaction network



**Northeastern University**

**Under the guidance of Prof. Srikant Krishnamurthy**



**Nehal Bhanushali**



**Pranjal Jain**



**Vasanti Mahajan**

# ABSTRACT

In this project, we attempt to apply machine-learning algorithms to predict rise and fall in price and further predict the price of bitcoin. We have also discovered some interesting finds through exploratory data analysis and anomalies through clustering on the graph database that was preprocessed to identify the unique ID that makes transactions with multiple public keys

# EXPLORATORY DATA ANALYSIS

Refer to the Interesting-Bitcoin-Transactions.ipynb in the github repository

**Data Downloaded from Numsight bitcoin Visualization tool**

Three files have been created as follows:

**1. User\_transactions**

FROM,TO,AMOUNT,TIMESTAMP

**2. Raw\_transactions**

FROM,TO,TIMESTAMP

**3. Transaction\_value**

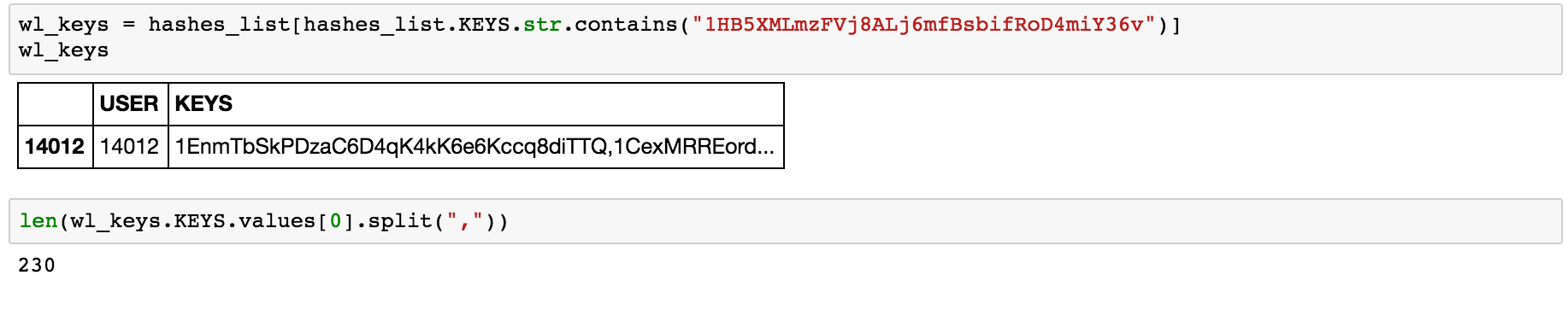
TX,TIMESTAMP,VALUE

We can now lookup transactions involving interesting accounts on our web-application to find networks. Here's some known addresses to get started:

* 1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa Genesis Address
* 1Ez69SnzzmePmZX3WpEzMKTrcBF2gpNQ55 US-Marshals Silk Road
* 1XPTgDRhN8RFnzniWCddobD9iKZatrvH4 Pizza Laszlo
* 14rE7Jqy4a6P27qWCCsngkUfBxtevZhPHB French Maid FBI
* 1HB5XMLmzFVj8ALj6mfBsbifRoD4miY36v Wikileaks
* 1FfmbHfnpaZjKFvyi1okTjJJusN455paPH FBI User
* 1EBHA1ckUWzNKN7BMfDwGTx6GKEbADUozX A very rich user

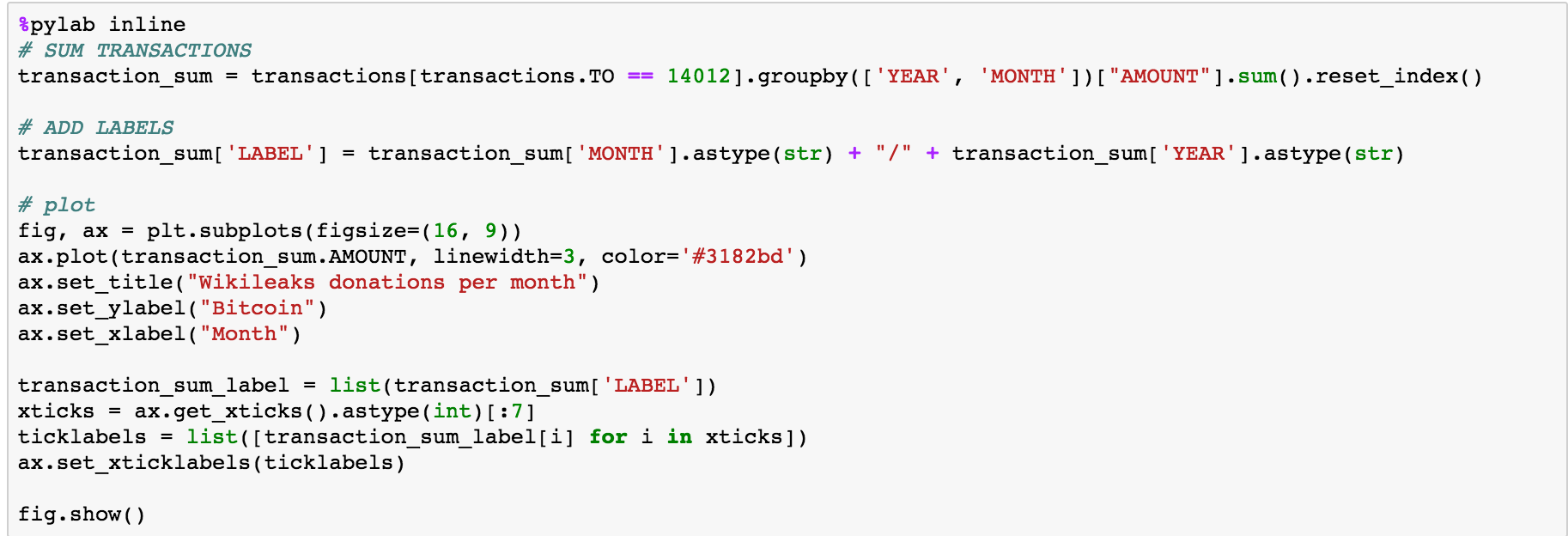
**1.WIKILEAKS**

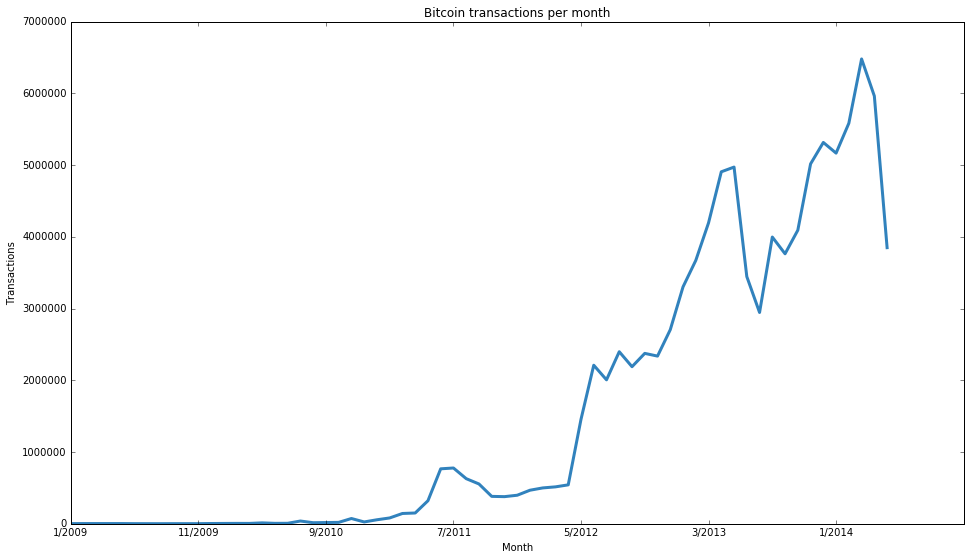
First, we have to lookup the user id for these hashes, e.g. for Wikileaks:



We can now see that Wikileaks seem to be using **230 different known public hashes for receiving and distributing their donations**. Let's now take a look at the total incoming and outgoing transactions:

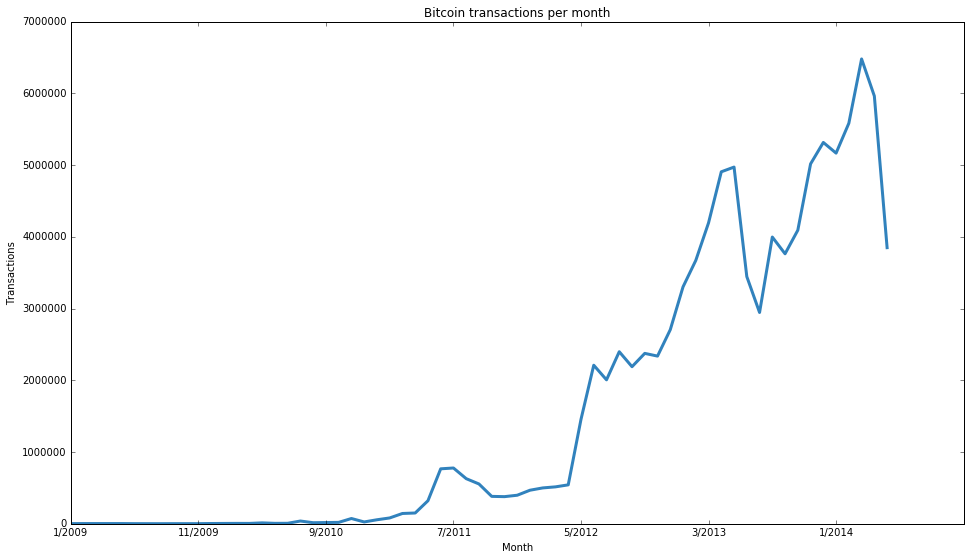
**PLOTTING DONATIONS COMING TO WIKILEAKS OVER THE TIME**





## **2.Aggregated Bitcoin analyses:**

Plot the number of Bitcoin transactions per month. Despite the bleak future of Bitcoin, the number of transactions experienced strong growth in the last years.



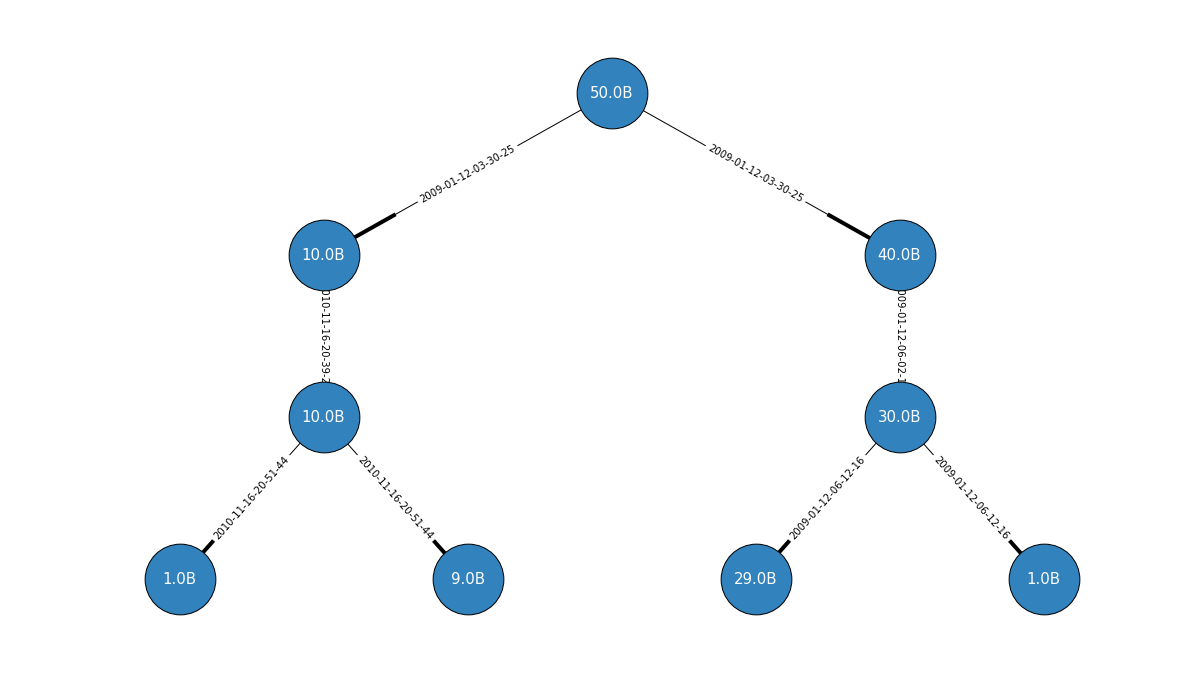
**Bitcoins sent between addresses**

A huge spike in September 2012 and then declined again.



## **The first Bitcoin transaction ever**

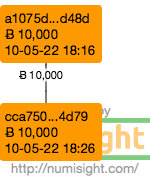
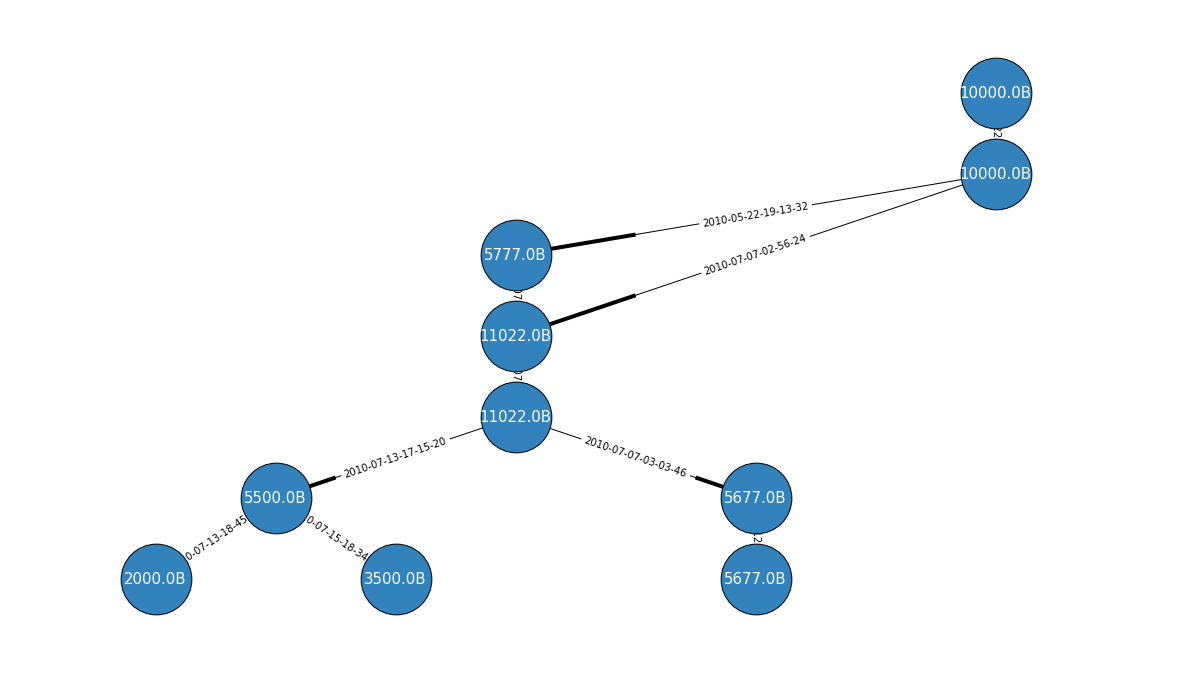
This is the first Bitcoin transaction where Satoshi Nakamoto sends Hal Finney 10 BTC and transfers the remaining 40 BTC back to his own address. This structural pattern is called "**spend and change**" because some amount of Bitcoins are spent in this transaction and the rest is returned to the original address.



The same plotted using **Numsight Bitcoin Explorer tool**

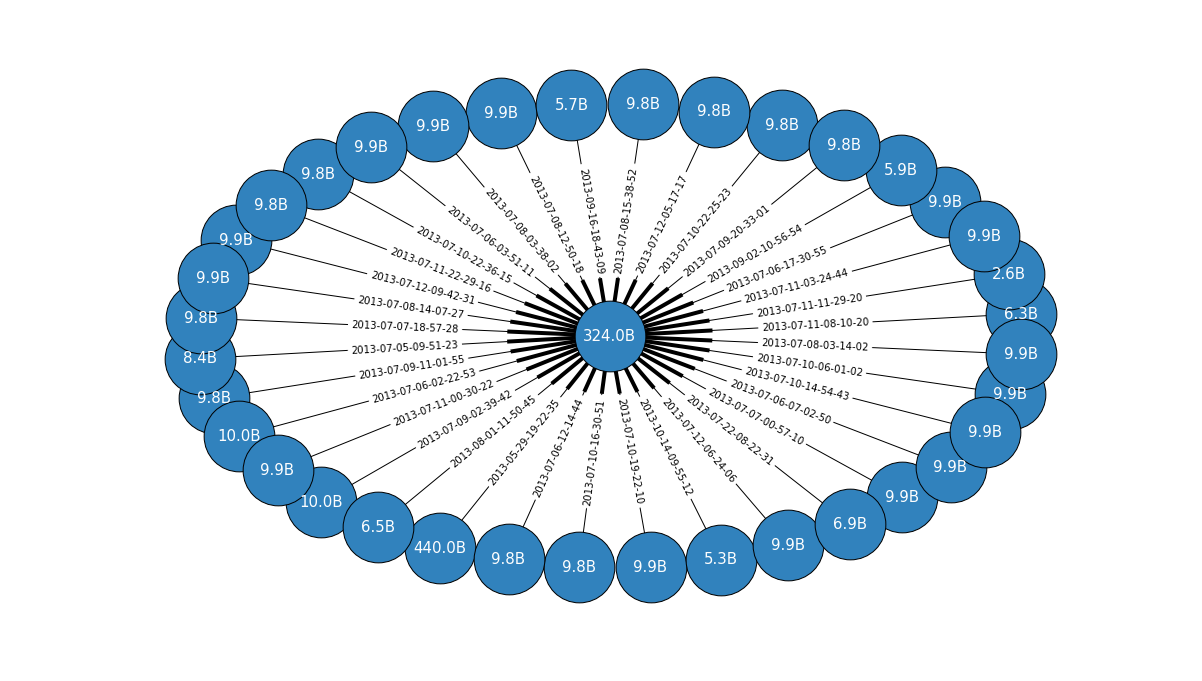
## **3.The Pizza Transaction**

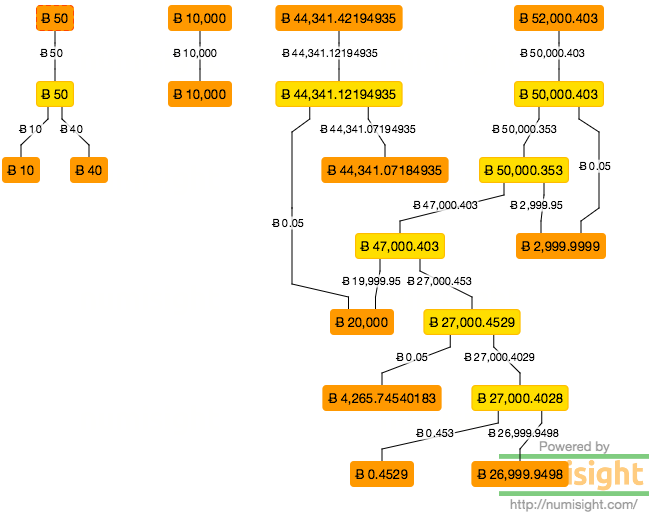
This is the famous pizza transaction, when Laszlo ordered a Pizza for 10,000 BTC in May 2010. This pizza money has been distributed from transaction to transaction, so parts of it are still in many Bitcoin accounts.



## **4.The Silk Road FBI seizures**

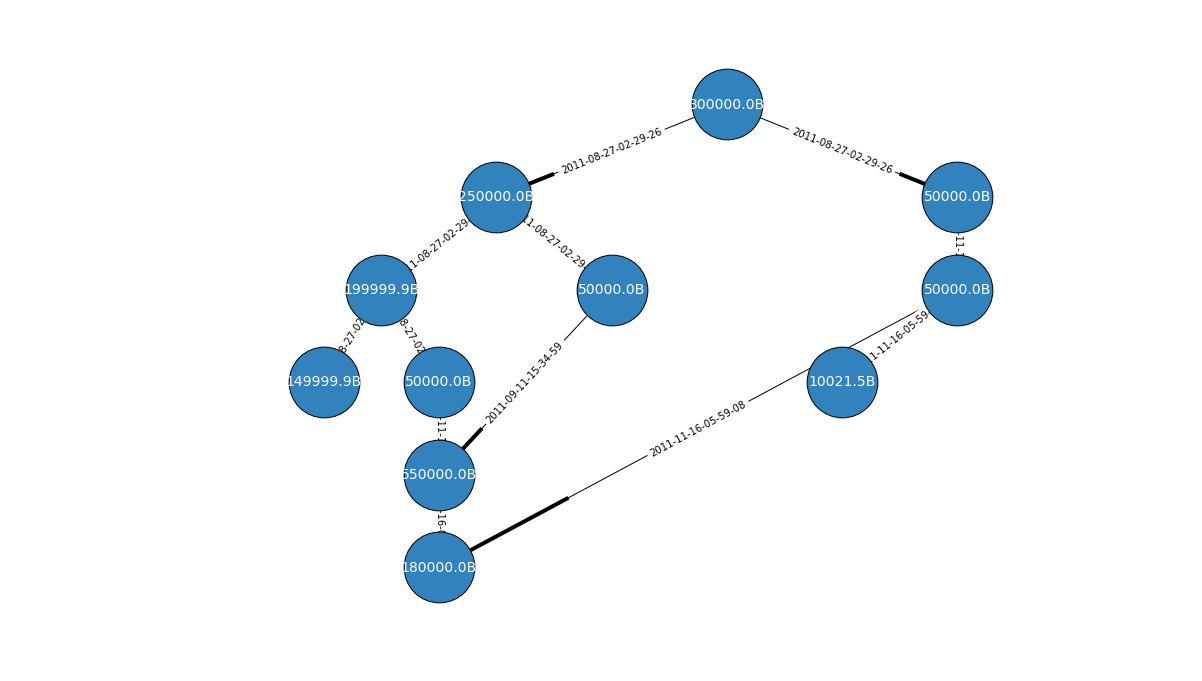
Here Bitcoins from multiple addresses are combined into a new (or a previously existing) address. The following example are the Silk Road seizures, where the FBI seized funds from Silk Road's Dread Pirate Roberts (DPR) and combined them into wallets of 324 Bitcoins each. This pattern is called **"consolidation".**

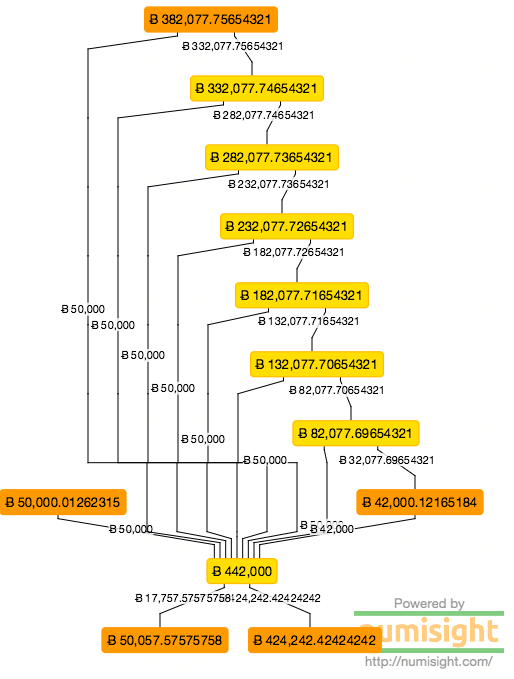




## **5.Transaction chains / "Peeling"**

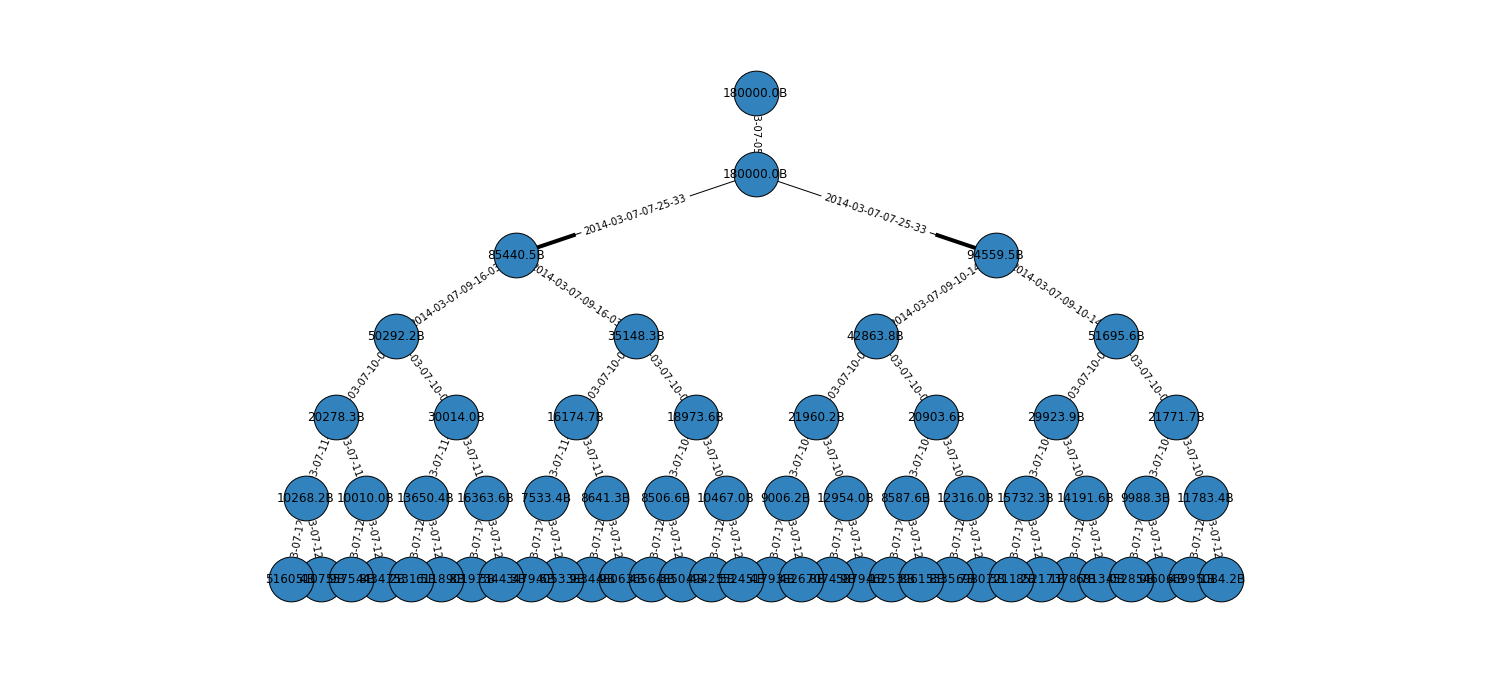
Very often from larger amounts of Bitcoins round amounts of Bitcoins (e.g. 50,000 BTC) are subsequently "peeled" off the original amount. These change addresses can also feed back into another peeling chain. The example is assumed to be a peeling chain controlled by Mt. Gox.





## **6.The 180,000 BTC Pyramid Scheme**

An interesting transaction where 180,000 BTC are subsequently split into smaller amounts

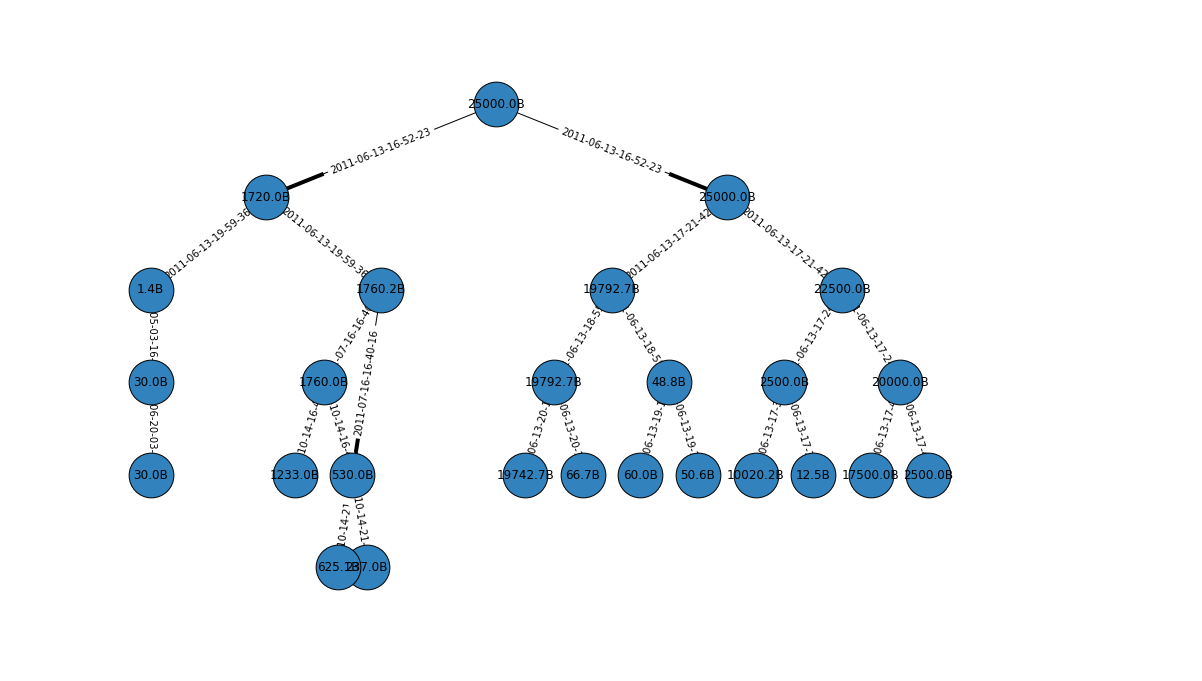


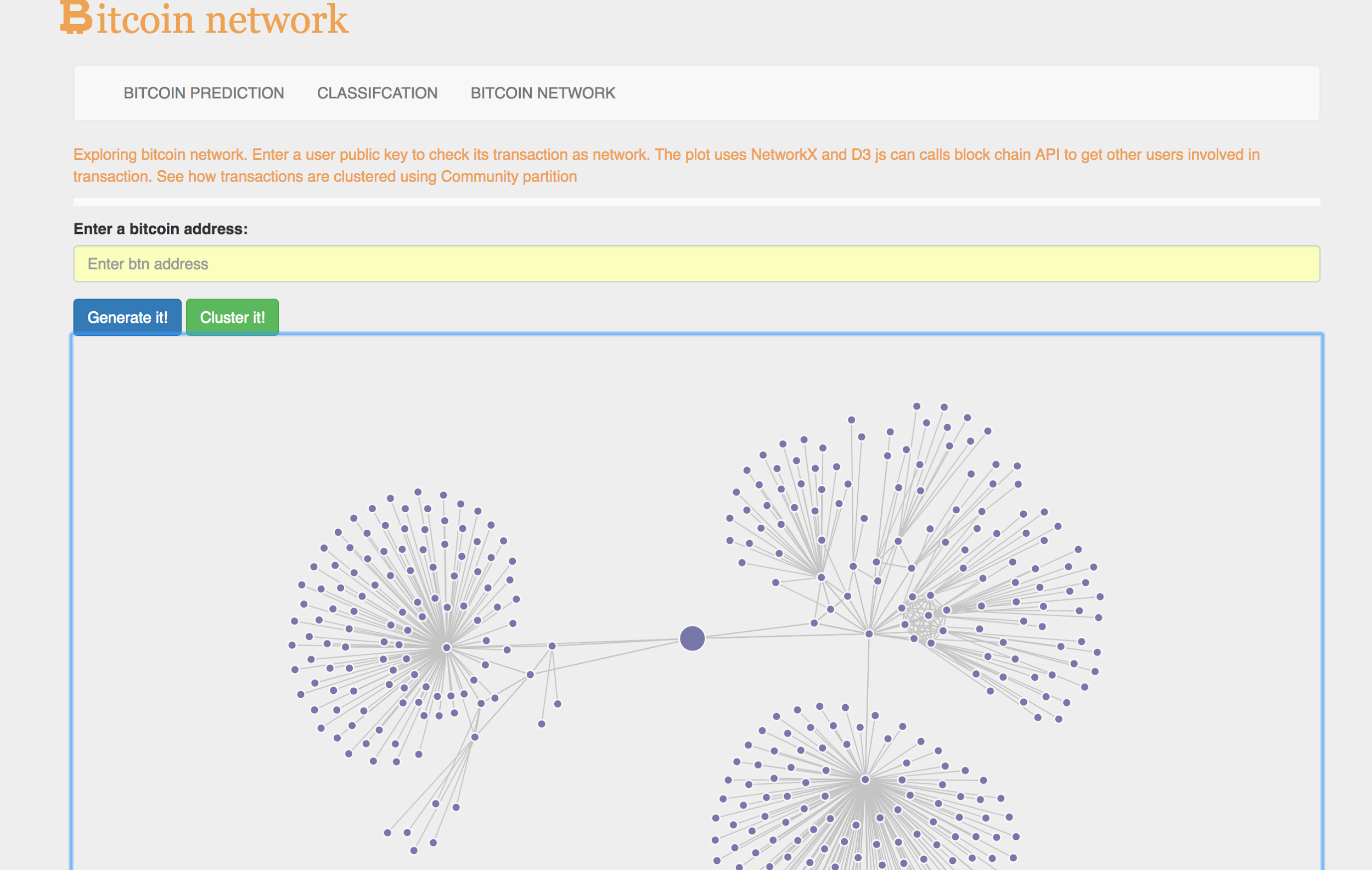
## 

## **7. Examining the Allinvain robbery**

Here we can see the transactions where someone allgedly stole 25,000 Bitcoin from allinvain. <https://bitcointalk.org/index.php?topic=16457.0>

First, we examine the transaction network that follows from the heist transaction:





CASE STUDY:

IDENTIFYING ILLICIT BITCOIN USERS IN THE NETWORK: (ANOMALY DETECTION)

**Goal**: Analysing any financial network for thieves and illegal activities is essential. These activities are anomalies that exists in a network in the form of fraudulent/suspicious transactions. Our goal is to perform anomaly detection to detect suspicious users and transactions in the Bitcoin Transaction network to prevent further illegal actions. This analysis could be used in detecting fraud in any kind of financial network.

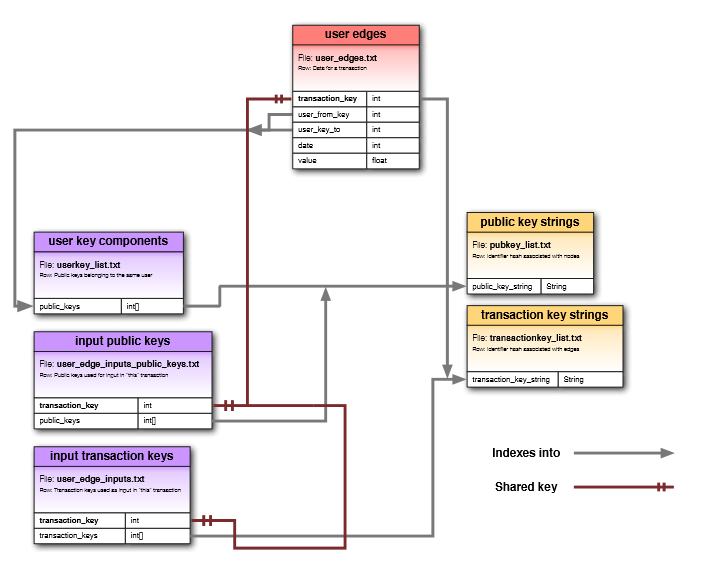
**Approach used:** Unsupervised learning using k-means clustering to detect anomalous users and their transactions

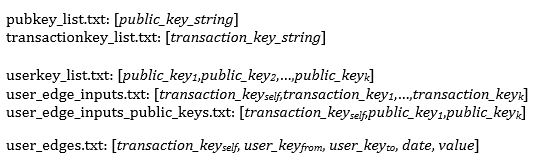
**Data Collection and Parsing:**

**Data Collection:**

Bitcoin transaction data set is obtained from the University of Illinois Urbana- Champaign. The data set contains all Bitcoin transactions beginning from the network's creation until April 7th, 2013. For each transaction there can be multiple senders and receiver addresses. These multiple addresses could belong to the same or different user’s.

The data set is quite large: there are 6, 336, 769 users with 37, 450, 461 transactions. The following are the entities and their relationship with each other.

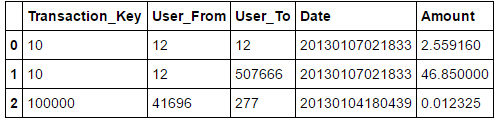




**Data Parsing:** We parsed the data to obtain relationship between user who send the bitcoins and user who received the bitcoins via a transaction. Since the user can have multiple public addresses with which it can make the transaction, it was essential to find a unique private key for every user to make the correct mapping between the transactions. We wrote a Hadoop Map-Reduce program to get represent the users in the transaction with their unique private key.

Note: The Map Reduce job was run on AWS-EMR. Since the data was huge, it required more than 17 hours for the Map-Reduce job to give the desired data.

**Feature Extraction:** The data was fetched from Map-Reduce job in the following format:



The following network properties were computed from them:

• **In-degree:** Number of transactions received by a given user.

• **Out-degree:** Number of transactions sent by a given user.

• **Unique in-degree:** Number of unique users a given user has received transactions from.

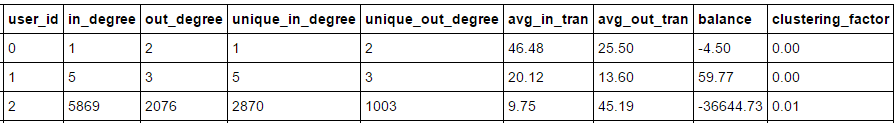
• **Unique out-degree**: Number of unique users a given user has sent transactions to.

• **Average in-transaction:** Average number of bitcoins received per incoming transaction.

• **Average out-transaction:** Average number of bitcoins sent per outgoing transaction.

• **Balance**: Net number of bitcoins retained by user.

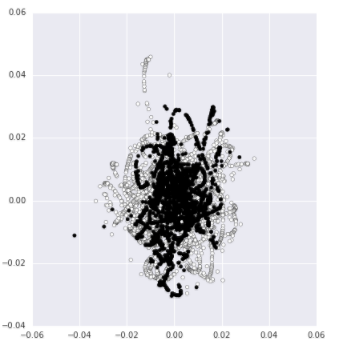
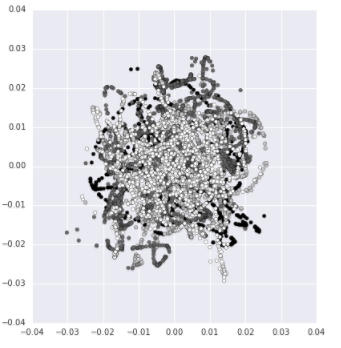
• **Clustering coefficient**: measure of connectivity amongst neighbors of a given user.



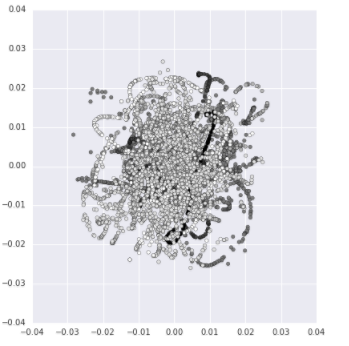
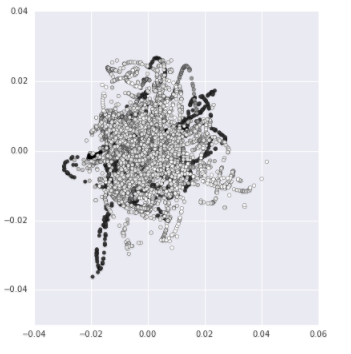
**Anomaly detection technique:**

**K-means clustering:**

K means clustering on its own is not sufficient for anomaly detection, however it helps us identify the plots outside the clusters indicating a potential suspicious behaviour. After careful consideration, the most suitable clusters are 2 for our dataset.

Cluster k=2 Cluster k=4

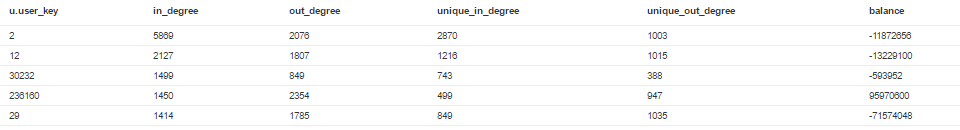
Cluster K=6 Cluster k=8

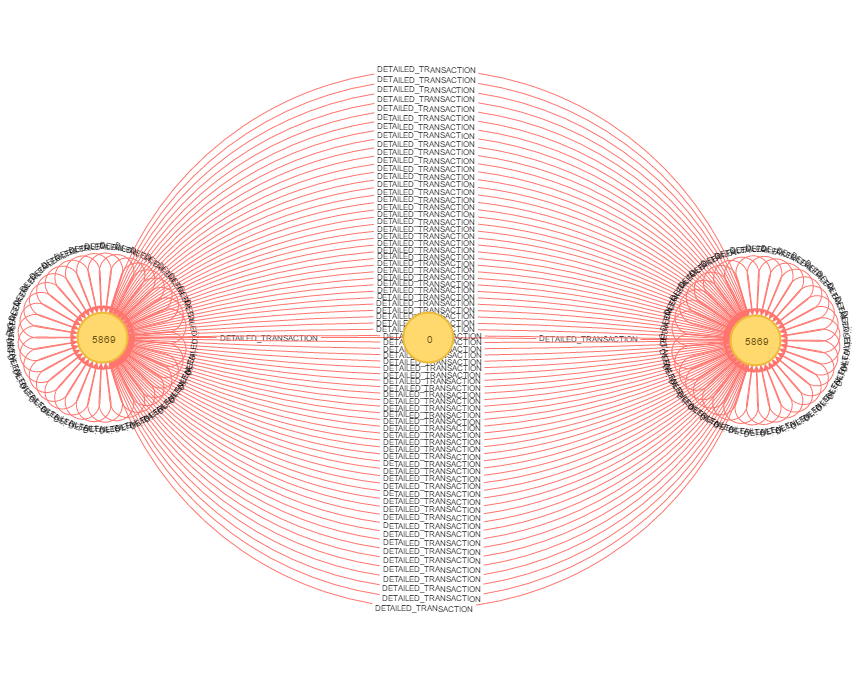
**Analysing the potential anomalies**:

Once the potential anomalies were detected using k-means algorithm, we explored them in an interactive graph database in Neo4j which helped us gain a few insights regarding the user’s and their transactions.

**Observation 1:**

We obtained the user’s who have the highest number of incoming transactions. Following are the top 5 records:

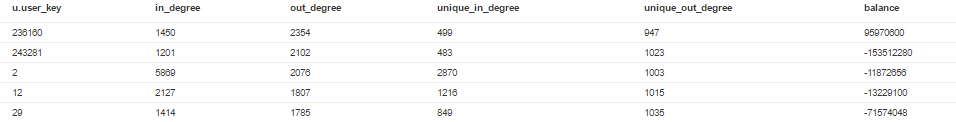


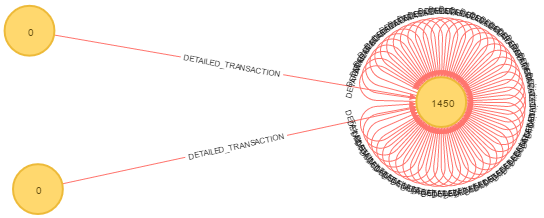
Looking at the statistics for user 2, the following conclusions can be drawn:

1. Majority of the incoming transactions are done by the user to itself. The user is making transactions from one of its own public key to itself.
2. The neighbours of the user are highly correlated and are seen making a lot of transactions to either themselves or with each other

**Observation 2:**

We obtained the user’s who have the highest number of outgoing transactions. Following are the top 5 records:





Looking at the statistics for user 236160 the following conclusions can be drawn:

1. Majority of the outgoing transactions made by the user are to itself
2. The net number of bitcoins retained by the user are 95970600 which is a very large value

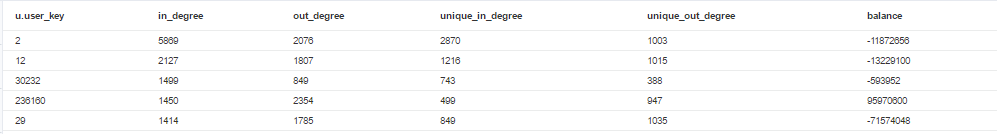
**Observation 3:**

Effect of clustering coefficient vs in\_degree

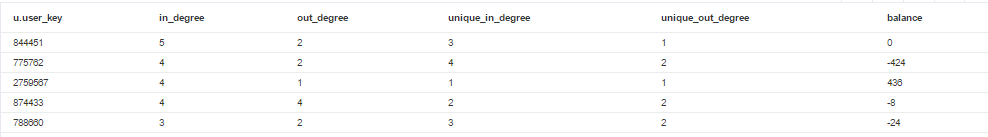
The local clustering coefficient of a [vertex](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) (node) in a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) quantifies how close its [neighbours](https://en.wikipedia.org/wiki/Neighbourhood_(graph_theory)) are to being a [clique](https://en.wikipedia.org/wiki/Clique_(graph_theory)) (complete graph). The higher the clustering coefficients, the lesser would be the number of transactions coming to the user. This relationship is depicted in our graph dataset.

While the clustering in a network can be measured in a number of ways, one common way to do it is to check for triangles, i.e., to check that when two edges share a node, then in a network with high clustering, it is likely that a third edge exists such that the three edges form a triangle.

For clustering coefficient 0:



For clustering coefficient 1:



# graph (3).png

# Thus, the nodes with clustering coefficients 1 tend to form cliques as depicted in the above graph. A strong clique like this could be a potential group of people who are performing multiple transactions to either hide a particular transaction or to affect the price of bitcoin.

User\_Key 2 : Public Address: 16UGfrnx4ew9rfmm7K5i1LbwSg4yEdCw2r is a common user for the above cases so it could be one of the potential anomalous user.

# 

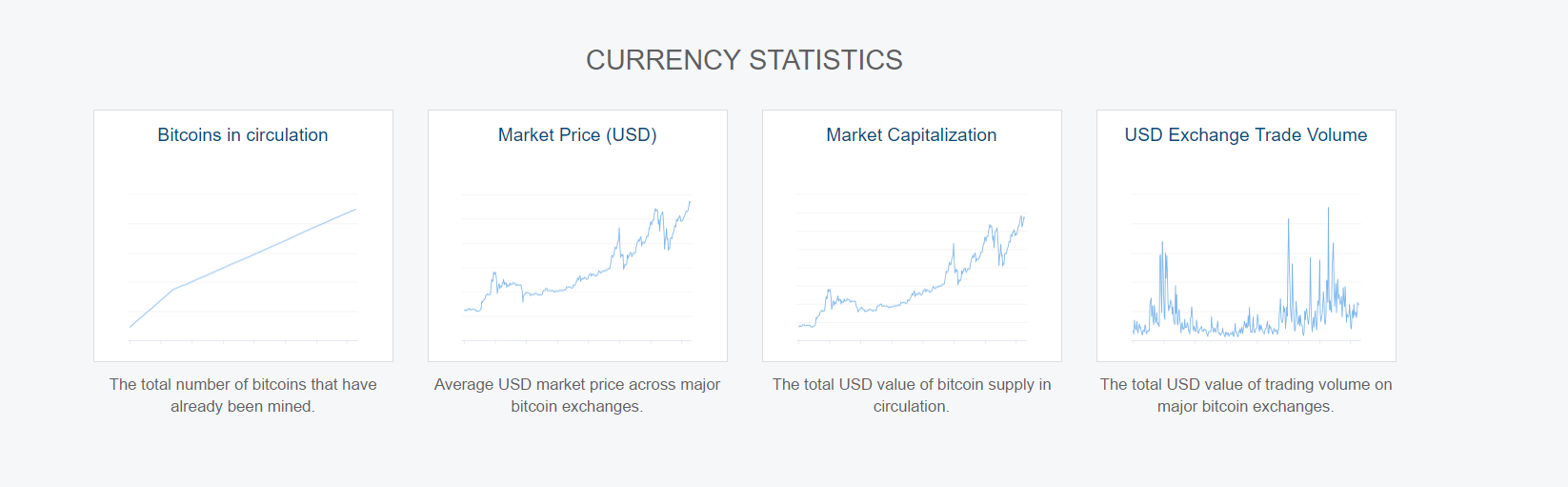
# 

# CLASSIFICATION : Will the price go up or down?

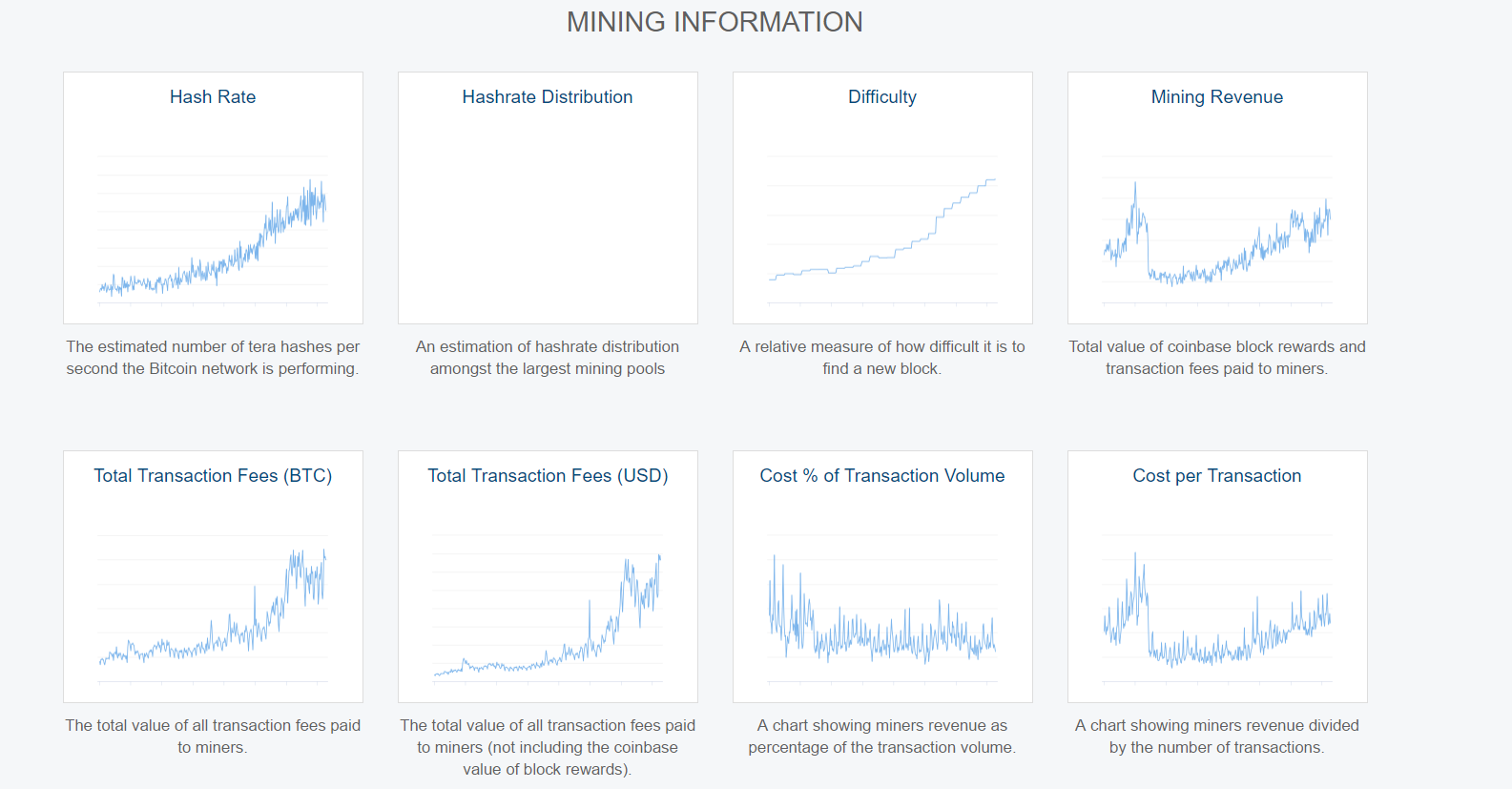
**Goal:** To identify from numerous features if the price of the bitcoin will go up or down

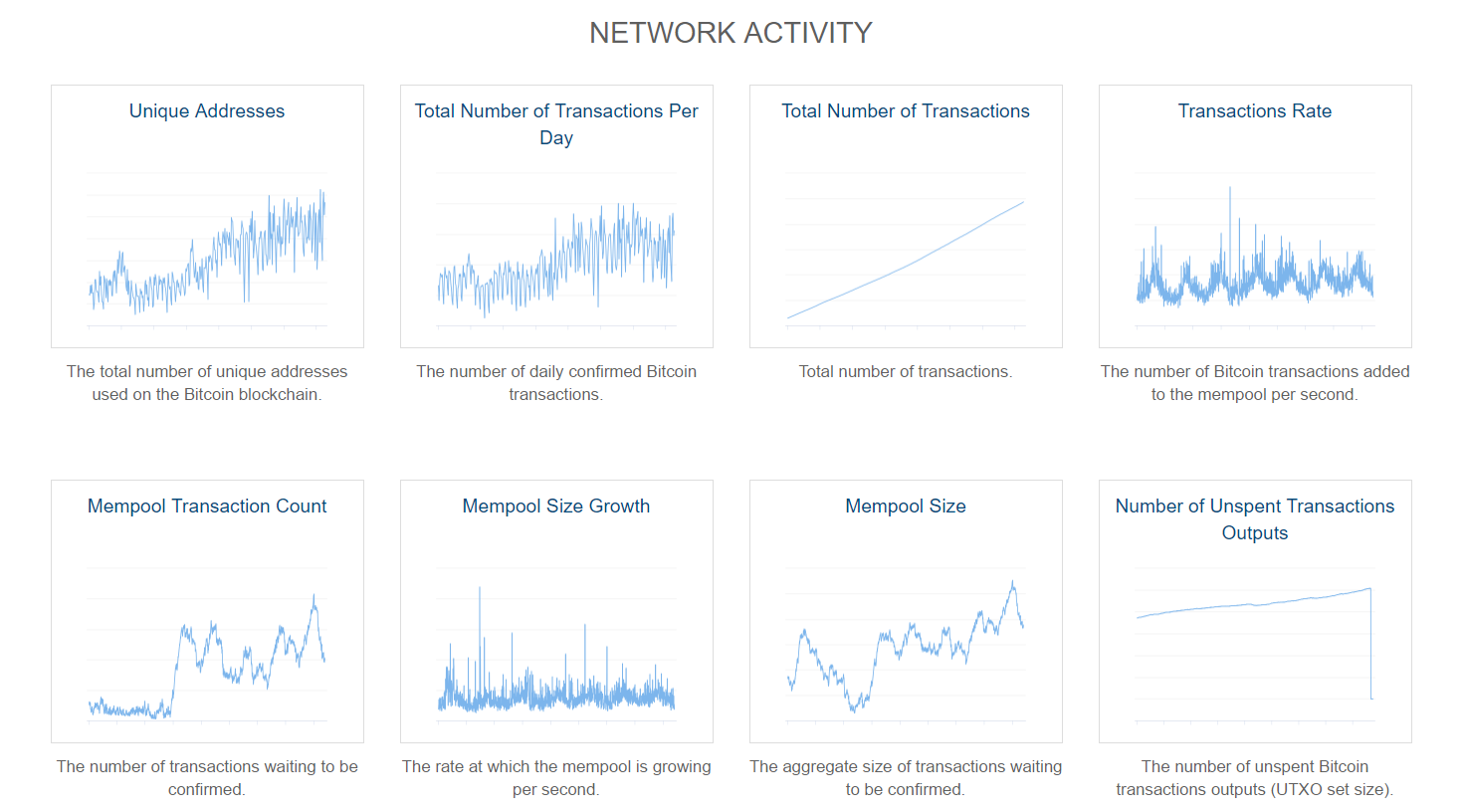
**Data:** The Blockchain API has about 26 features available from 2009 till present. These features are present separately and have been programmatically downloaded and merged on timestamp.

**Features:**









**Feature selection:**

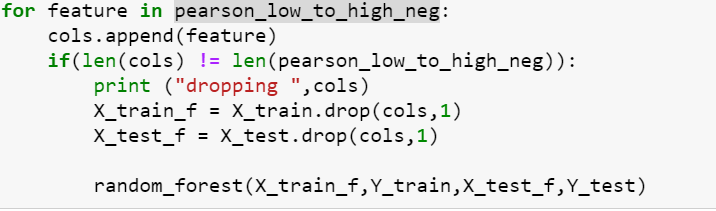
1. Manual brute force shuffle and backward elimination

2. RFE

3. Pearson Correlation

4. Select Percentile were used to select features for maximum accuracy.

Iterating through Pearson or Select Percentile scored features (low to high)



Dropping ['transaction-fees'] gave the best accuracy : 0.651315789474

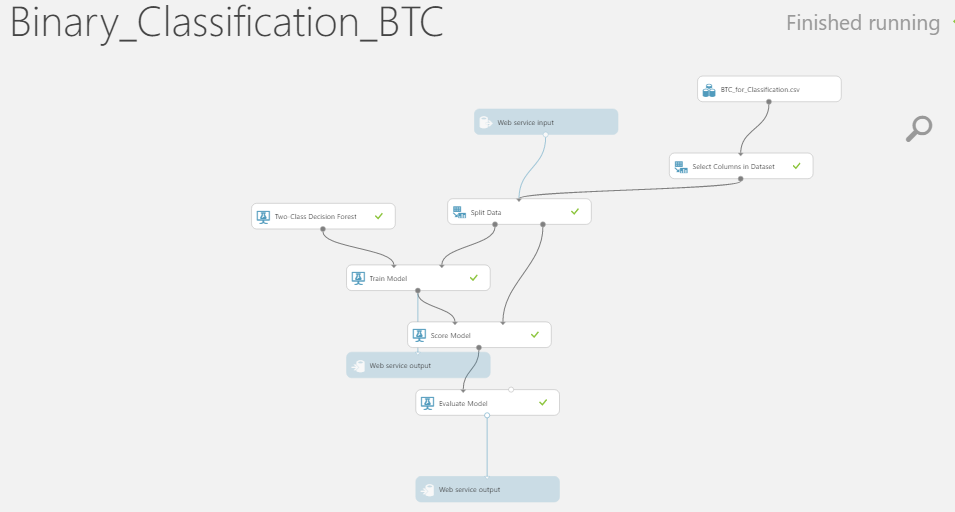
**Summary of finds on Random Forest and Neural Networks:**

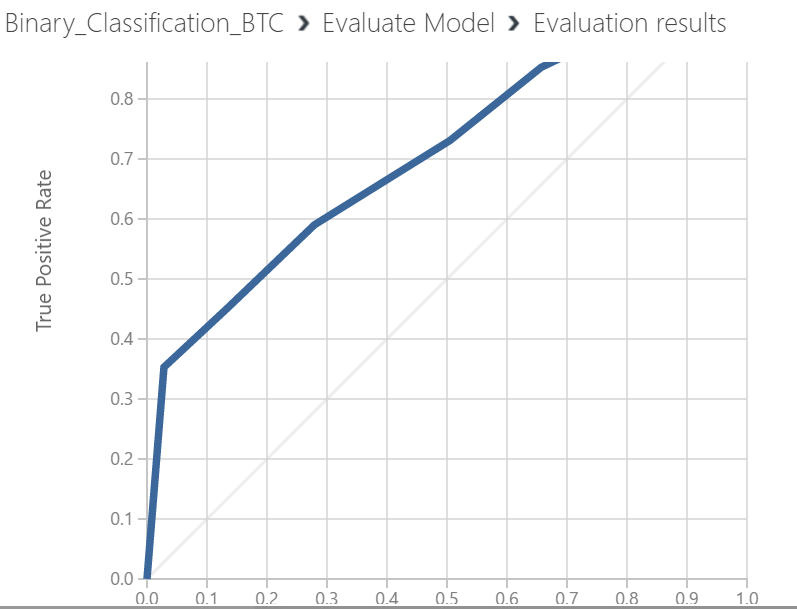
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Technique** | **Features** | **ROC** | **Accuracy** | **Confusion Matrix** |
| RFE, (Random Forest) | **Selected**  - Difficulty  - Est-txn-vol  - Hashrate  - Market-cap  - Miners-revenue  - N-transactions  - N-txn-per-block  - N-unique-addr - Total-bitcoins  - Txn-fees |  | 0.6052631579 | [83 103]  [77 193] |
| RFE,  (Neural Networks) | (Same as above) |  | 0.4030701755 | [133 53]  [230 40] |
| **Pearson**  **Correlation**  **&**  **Select Percentile,**  **(Random Forest)** | **Dropped**  **Transaction-fees** |  | **0.6513157895** | **[ 98 88]**  **[ 71 199]** |
| Pearson  Correlation  &  Select Percentile,  (Neural Networks) | (Same as above) |  | 0.6030701755 | [ 54 132] [ 49 221] |

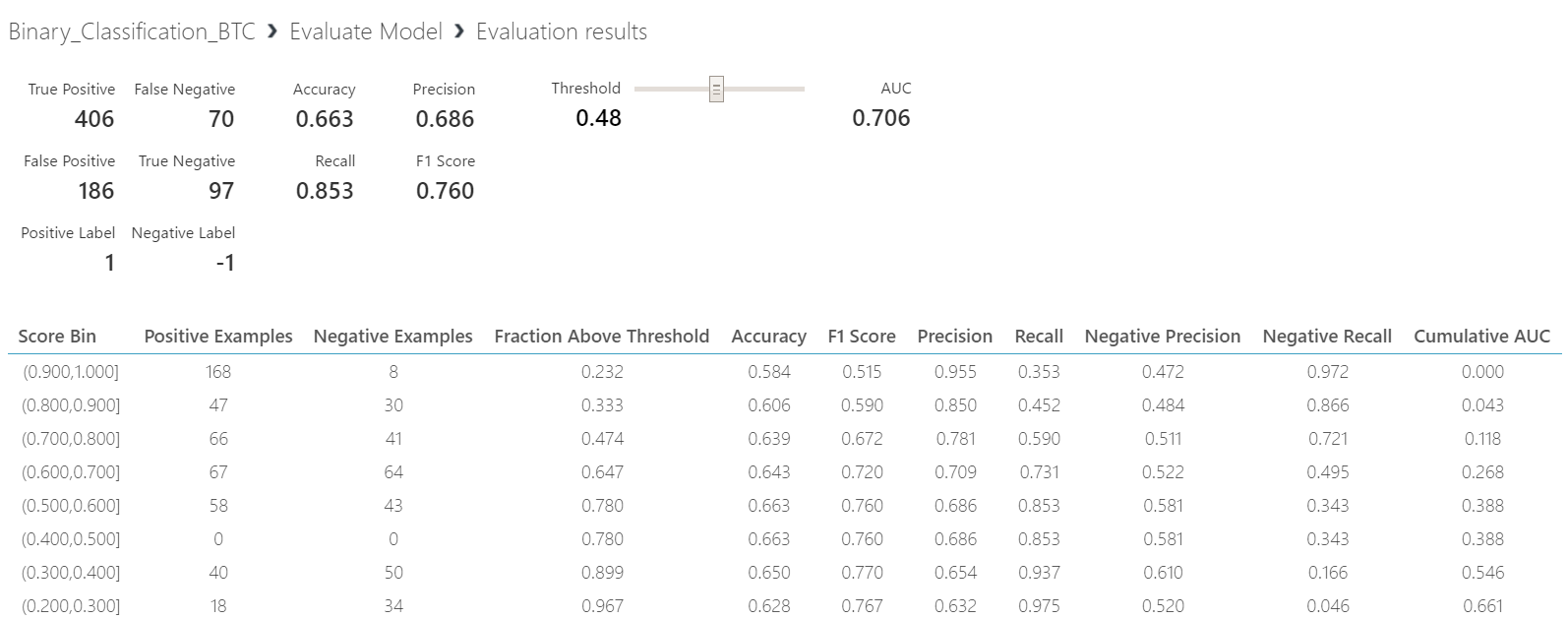
We went ahead with the following 15 features as suggested by both Pearson and Select percentile.

* hash rate
* difficulty
* transaction to trade ratio
* n transactions
* market cap
* n transactions per block
* avg block size
* estimated transaction volume
* n unique addresses
* n orphaned blocks
* miners revenue
* median confirmation time
* cost per transaction
* total bitcoins

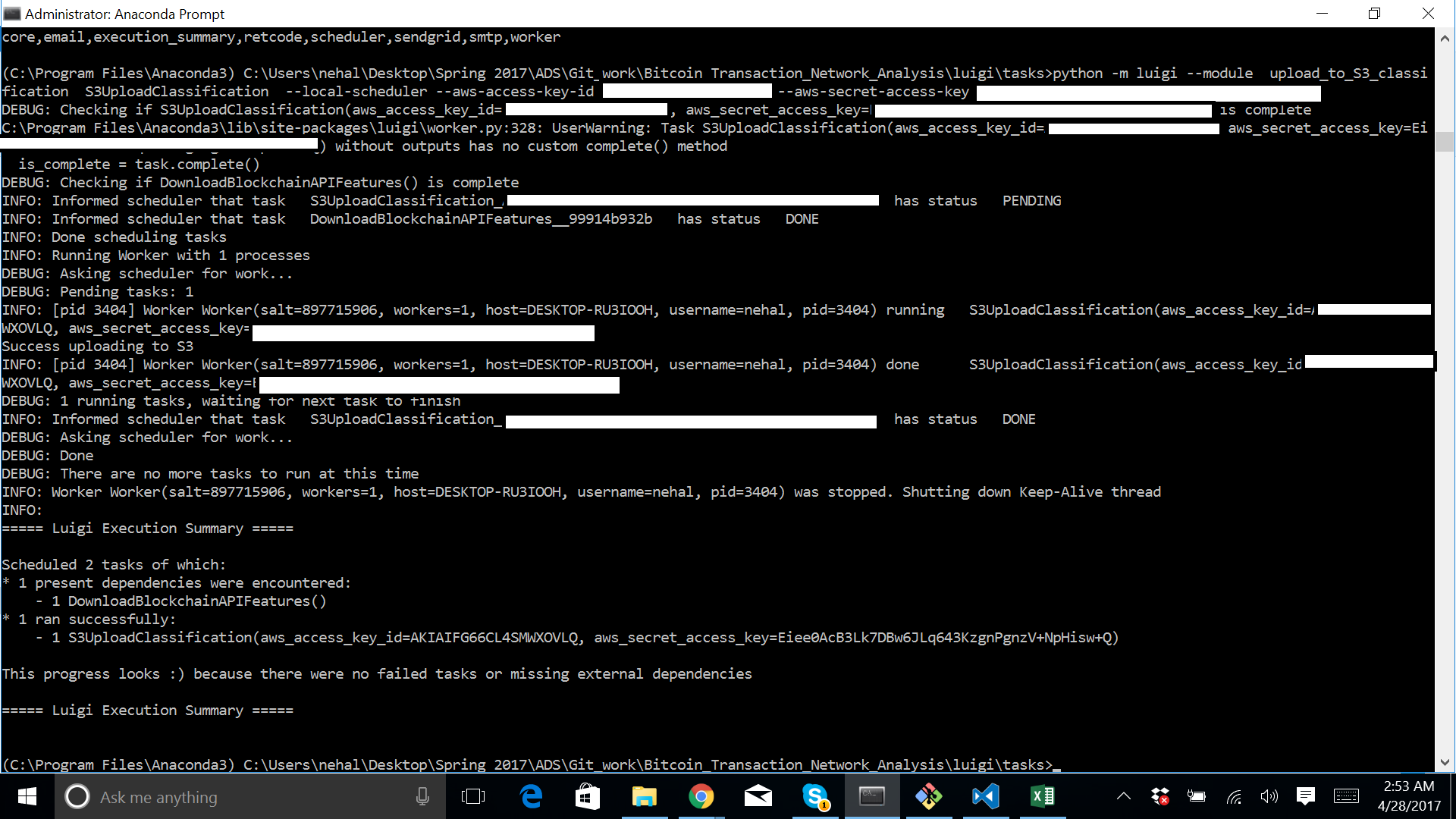
**Machine Learning with Azure:**

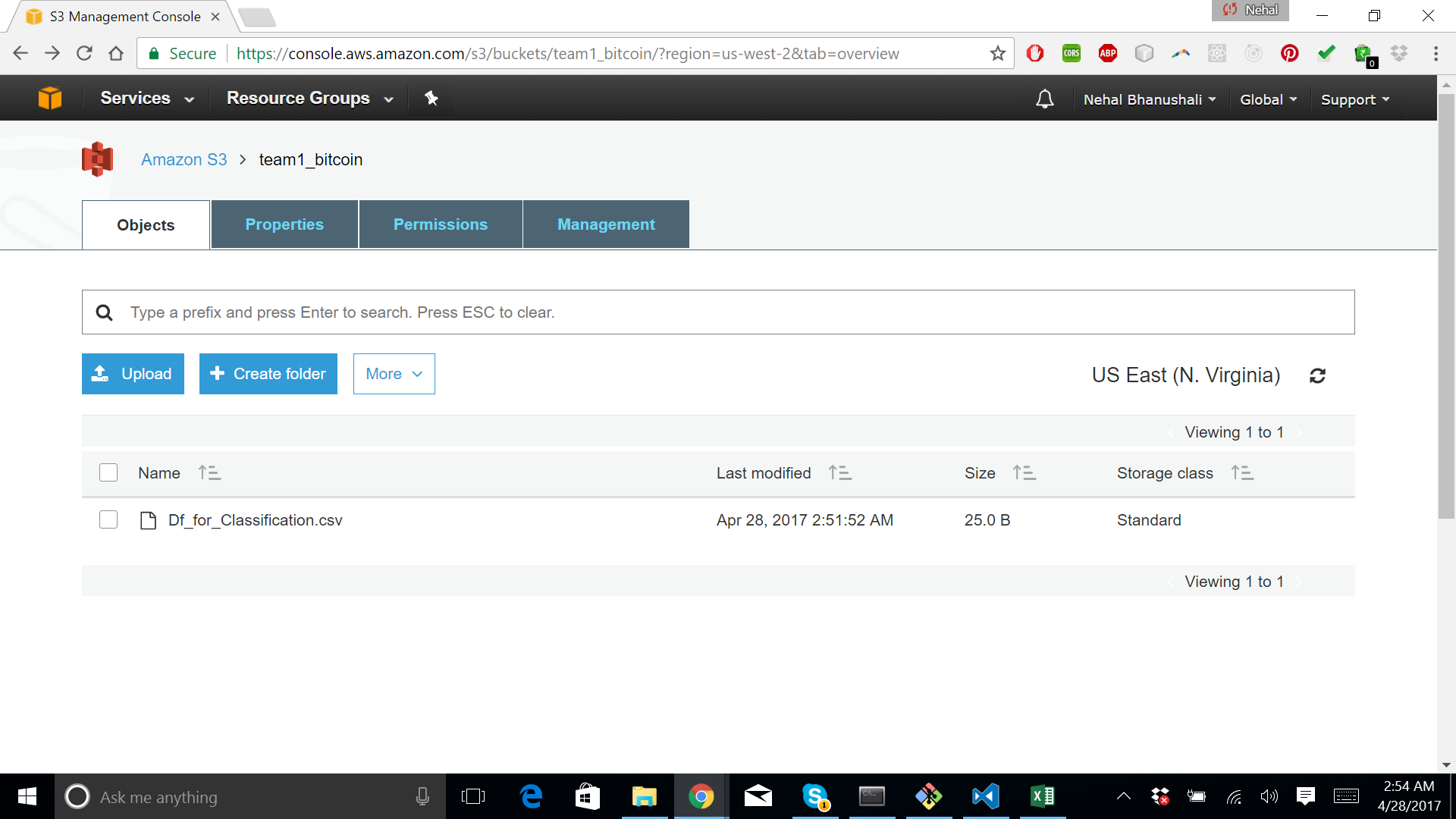






**Luigi Workflow :**





# REGRESSION MODELS TO MAKE HIGH FREQUENCY BITCOIN PRICE PREDICTIONS

# Blockchain Data and Bitfinex exchange Data

The Blockchain dataset consists of 16 features coming from classification.But the models constructed on these features were not enough to make high frequency bitcoin transaction. So data which is series of one second snapshots of open buy and sell orders on the Bitfinex exchange, combined with a record of executed transactions.

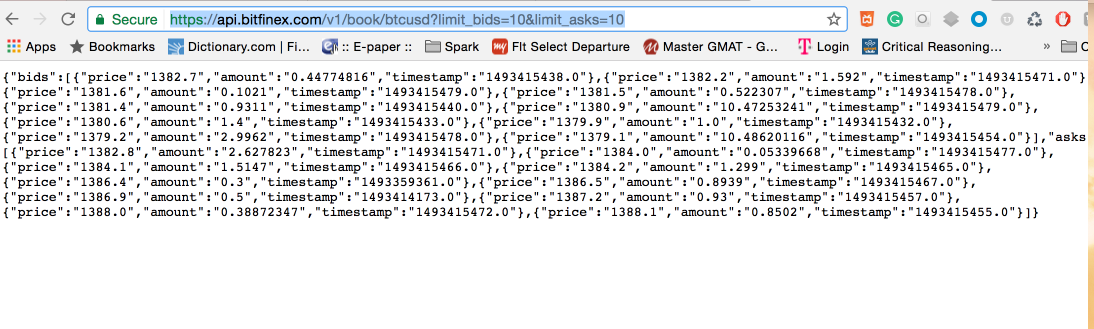
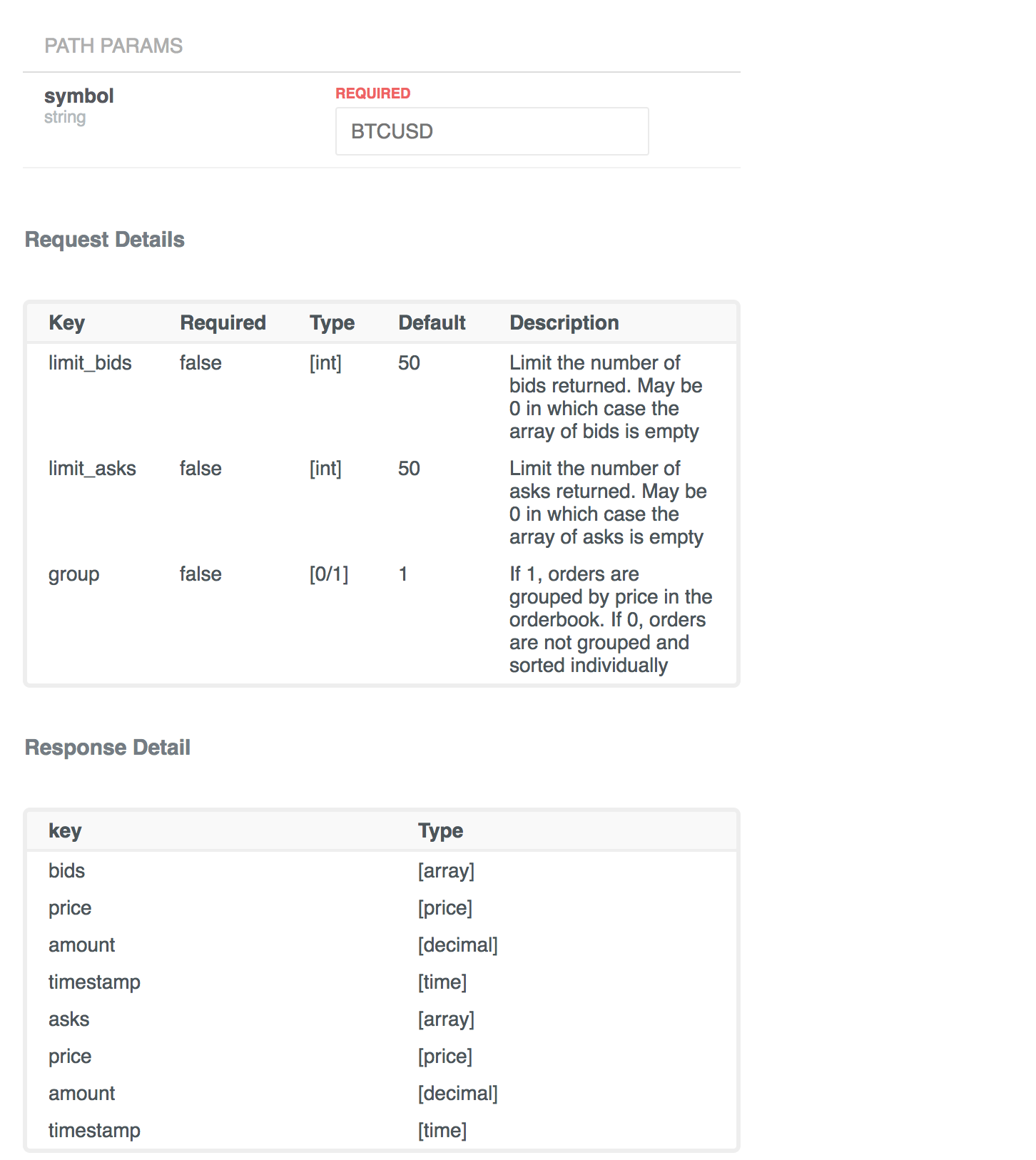
**Data collected and stored in MongoClient**

REST API’s used

ORDERBOOK

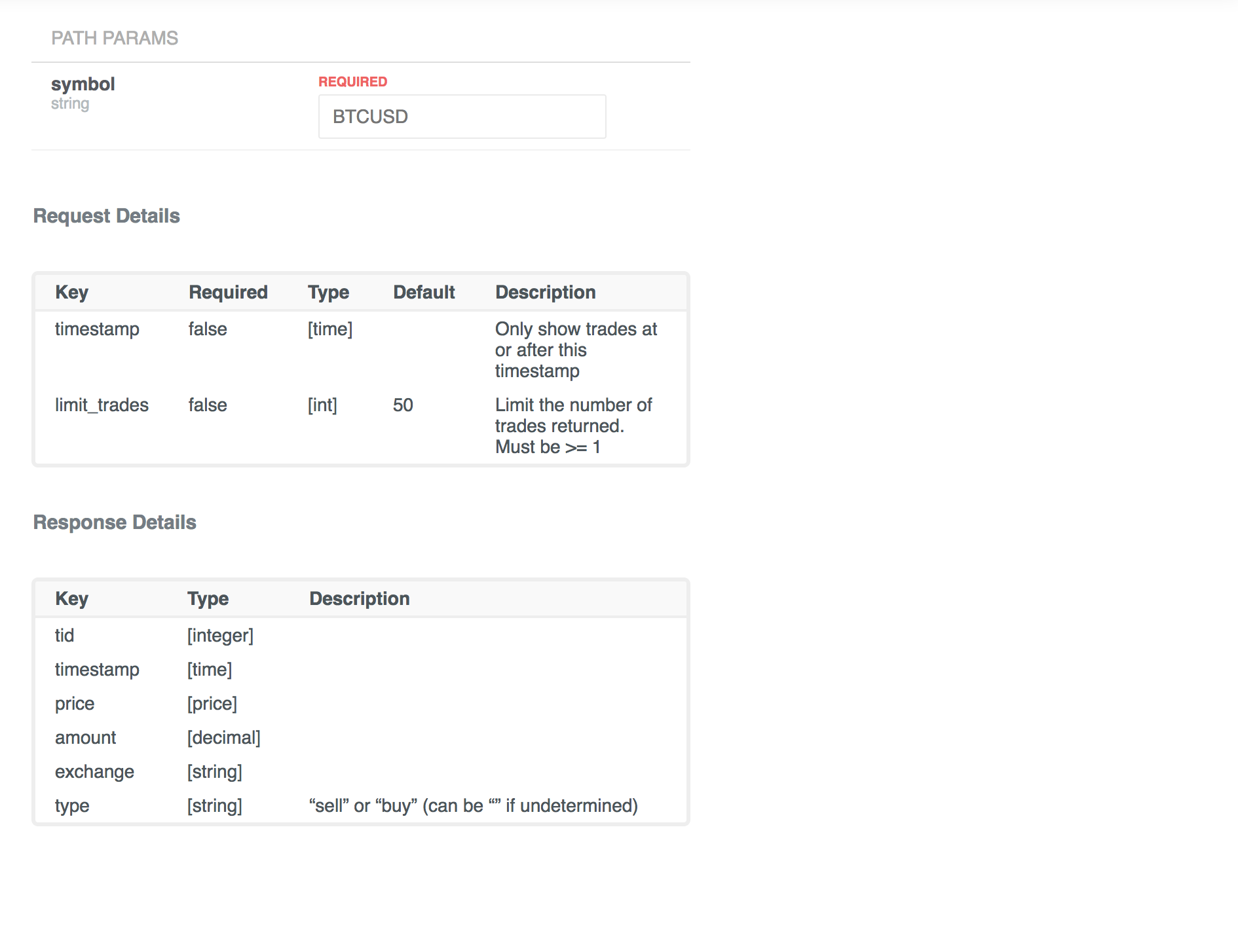
Get the full order book.

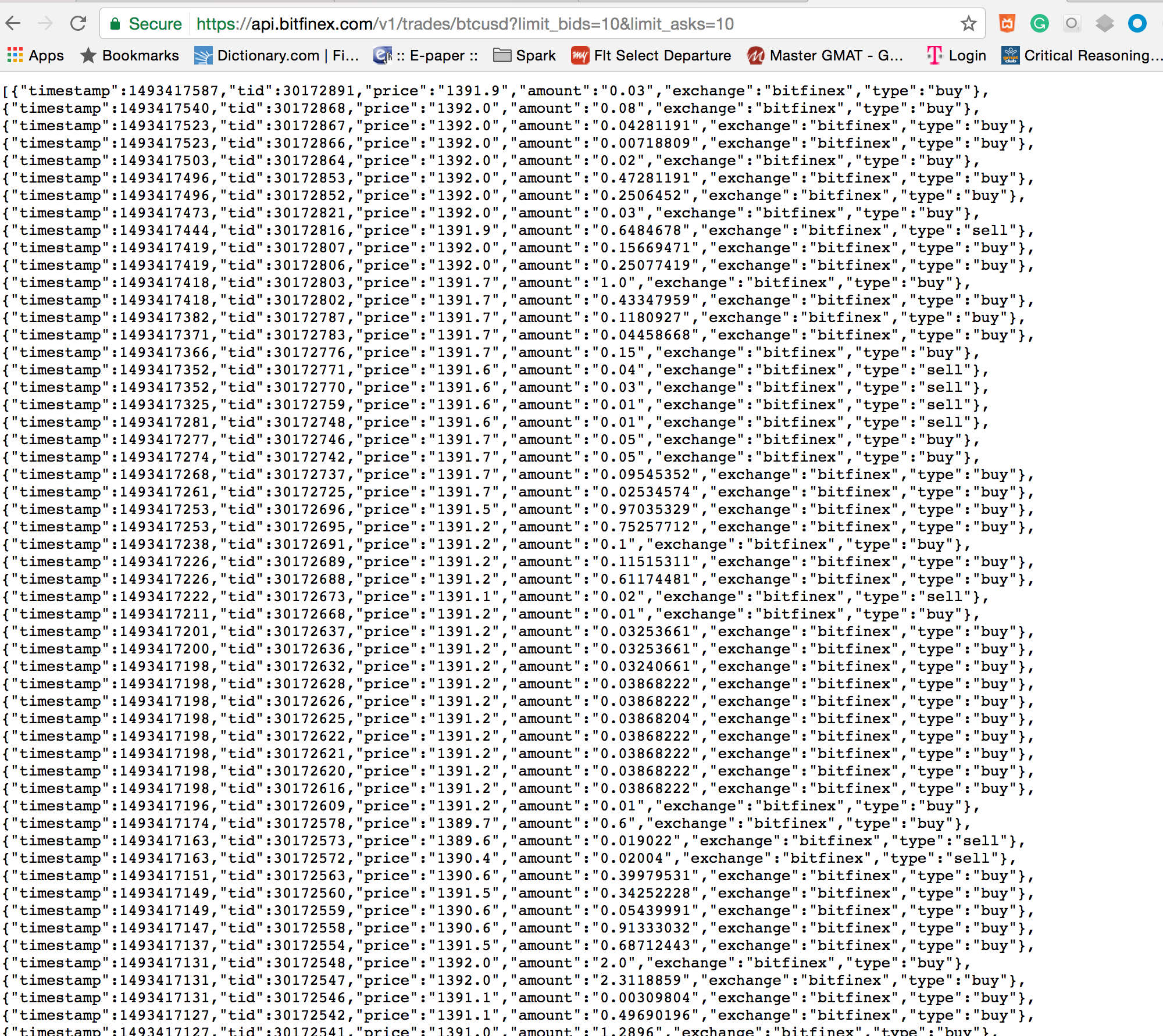
https://api.bitfinex.com/v1/book/btcusd?limit\_bids=10&limit\_asks=10



Trades

Get a list of the most recent trades for the given symbol.





Storing and Processing data (cleaning and deriving features)

## **Target**

The target for prediction is the midpoint price in the future. The midpoint price is the average of the best bid price and the best ask price.

## **Features**

#### **Width :** This is the difference between the best bid price and best ask price.

#### **Power Imbalance :** This is a measure of imbalance between buy and sell orders. For each order, a weight is calculated as the inverse distance to the current midpoint price, raised to a power. Total weighted sell order volume is then subtracted from total weighted buy order volume. Powers of 2, 4, and 8 are used to create three separate features.

#### **Power Adjusted Price:** This is similar to Power Imbalance, but the weighted distance to the current midpoint price (not inverted) is used for a weighted average of prices. The percent change from the current midpoint price to the weighted average is then calculated. Powers of 2, 4, and 8 are used to create three separate features.

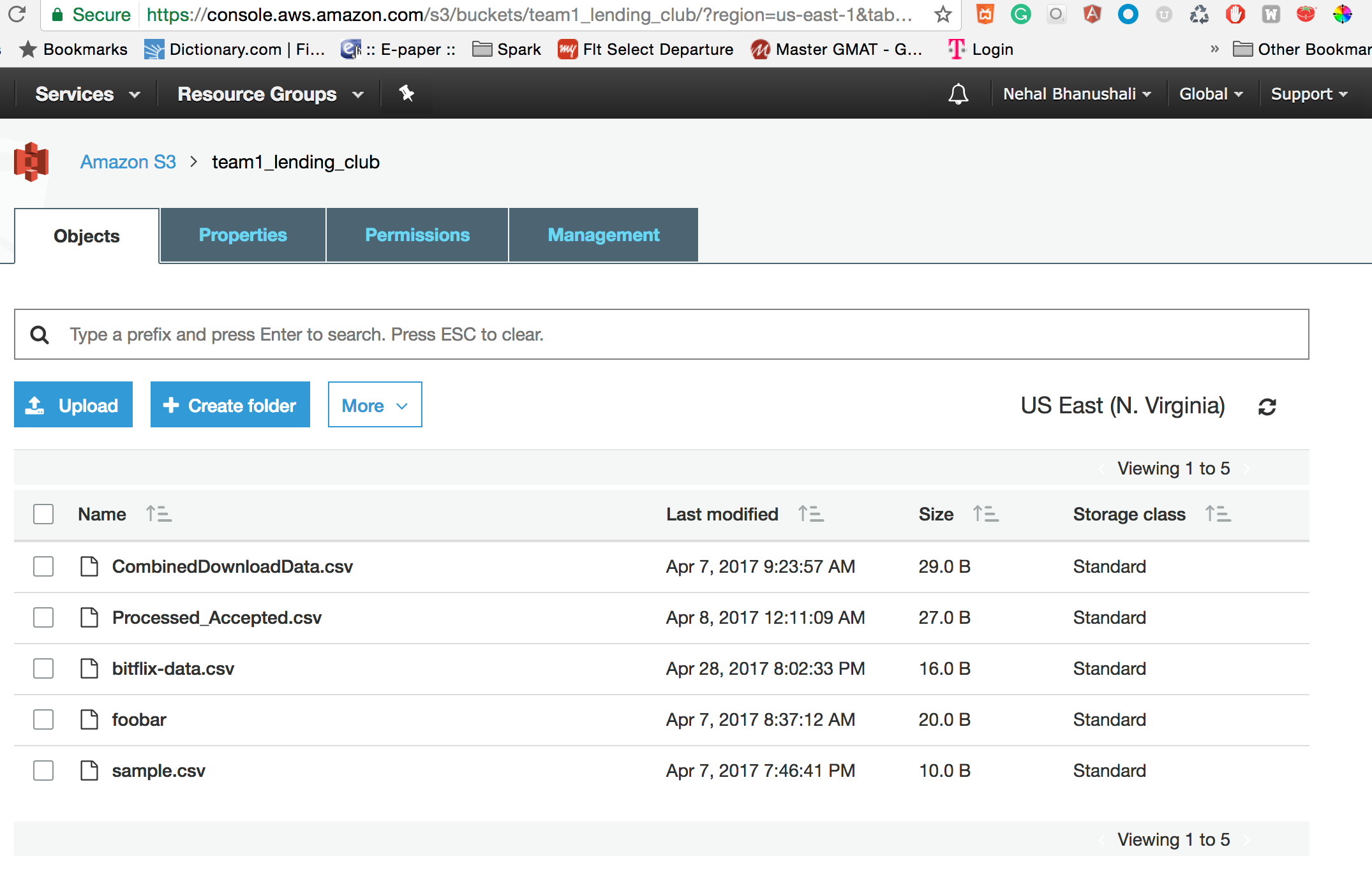
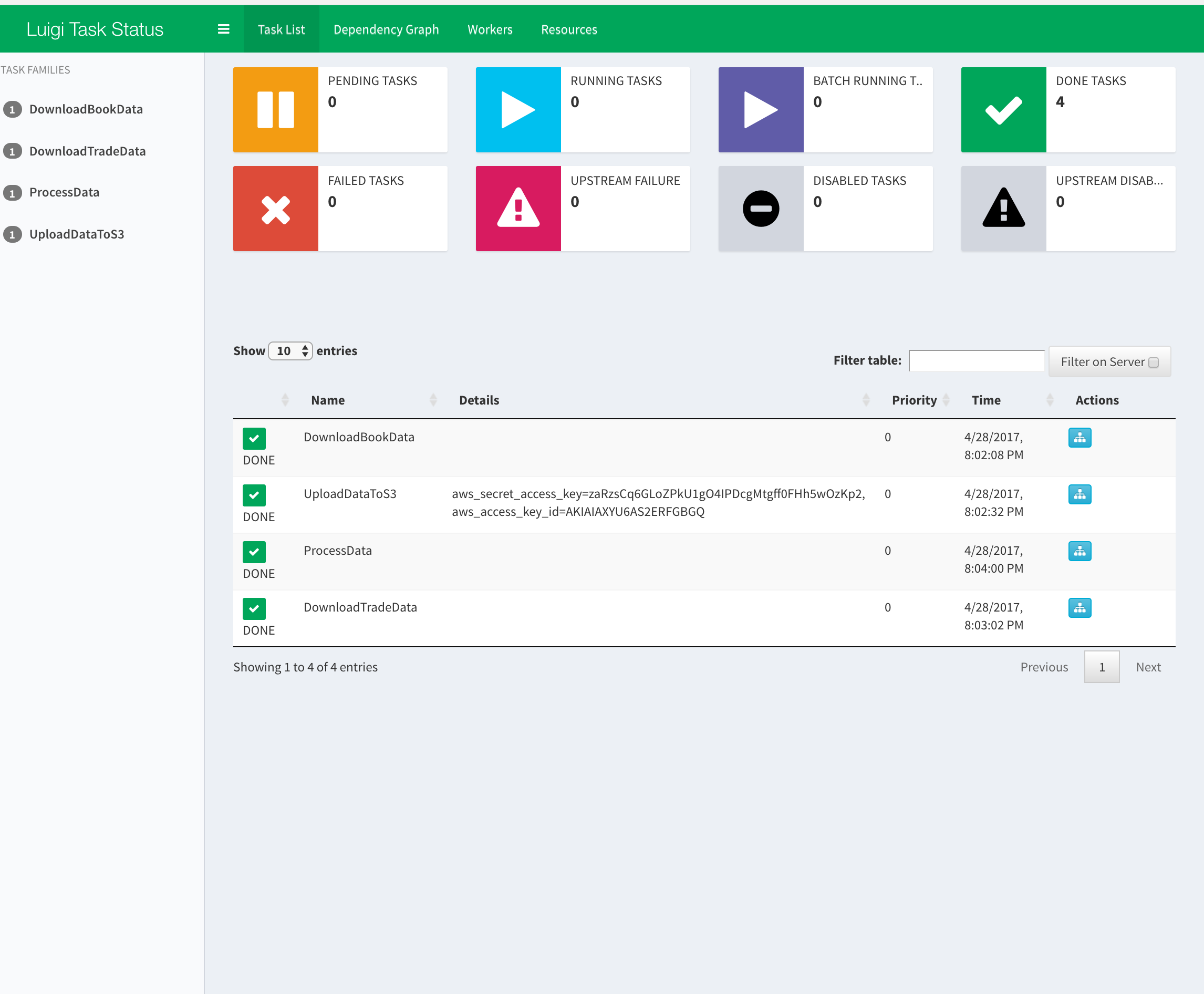
#### **Trade Count :** This is the number of trades in the previous X seconds. Offsets of 30, 60, 120, and 180 are used to create four separate features.

#### **Trade Average :** This is the percent change from the current midpoint price to the average of trade prices in the previous X seconds. Offsets of 30, 60, 120, and 180 are used to create four separate features.

#### **Aggressor :** This is measure of whether buyers or sellers were more aggressive in the previous X seconds. A buy aggressor is calculated as a trade where the buy order was more recent than the sell order. A sell aggressor is the reverse. The total volume created by sell aggressors is subtracted from the total volume created by buy aggressors. Offsets of 30, 60, 120, and 180 are used to create four separate features.

#### **Trend :** This is the linear trend in trade prices over the previous X seconds. Offsets of 30, 60, 120, and 180 are used to create four separate features.

**Downloading Data and processing to derive new features and saving to CSV file and uploading to S3 is part of luigi pipeline.**

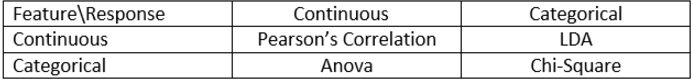


Feature engineering and selection is done to extract important features. Following four techniques are used for feature selection:

* Pearson correlation
* Sklearn’ s Select Percentile
* Randomized Lasso
* Recursive Feature Elimination

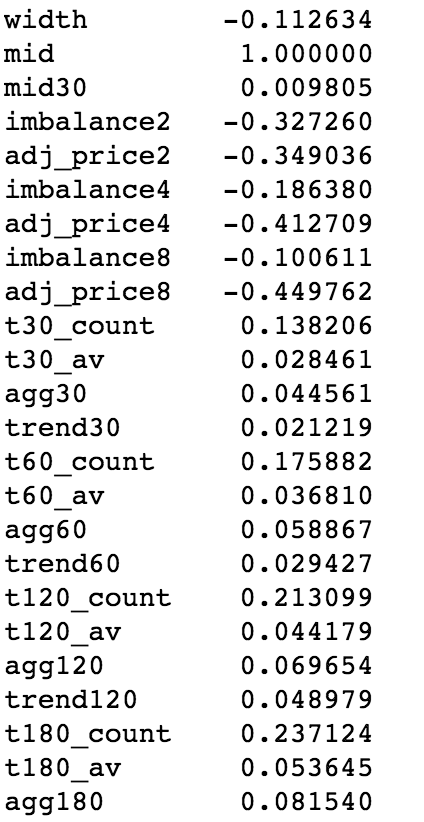
1. **FILTER METHODS:**





Since we are predicting a continuous label from a set of continuous features, we must compute Pearson Correlation between features.

The Pearson Correlation between int\_rate and all other features is shown below:



As we can see the features are not highly correlated, thus this is a poor technique for choosing the features.

Choosing the top most highly correlated features and running Linear Regression algorithm gives us the following scores:

**Parameters**:

['acc\_open\_past\_24mths','num\_tl\_op\_past\_12m','percent\_bc\_gt\_75','term','revol\_util','total\_rec\_int','meanfico','int\_rate']

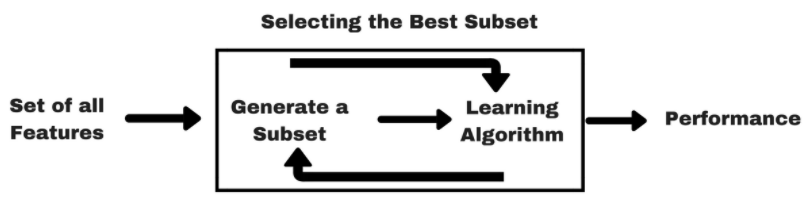
**Accuracy:**

Training score is **0.729775947499**

Testing score is **0.730534846323**

The model did not train with great accuracy as the features were chosen on the basis of Pearson correlation.

2. **WRAPPER METHOD:**



2.1 **Select Percentile:**

Using Sklearn’s Select Percentile feature selection technique we can choose the top 20% percentile of features which contribute the maximum in predicting the interest rate.

The top features obtained from this wrapper based technique and the accuracy of

the Linear Regression model is as follows:

**Features:**

['\_id', 'adj\_price8', 'agg180', 'imbalance2', 'adj\_price4', 'agg120', 'agg60', 'adj\_price2', 'imbalance4', 'width', 'trend180', 'agg30', 'trend120', 'imbalance8', 'trend60', 'trend30', 't180\_av', 't30\_av', 'mid30', 't30\_count', 't60\_av', 't120\_av', 't60\_count', 't180\_count', 't120\_count']

**Accuracy:**

Training score is 0.9169976353  
Testing score is 0.916618072069

2.2 **RFE**

Using Sklearn’s RFE model, the following features were chosen and the Linear

Regression model was run for it:

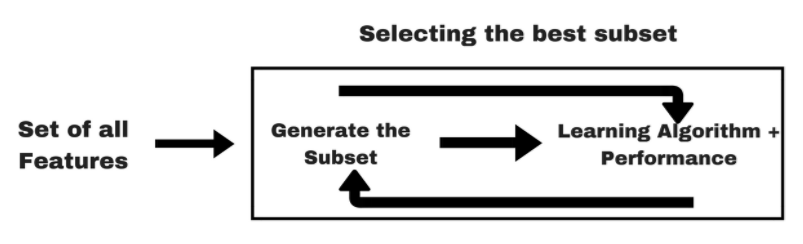
**Features:**

Features sorted by their rank:  
[(1, 'width'), (2, 't30\_count'), (3, 't180\_count'), (4, 't120\_count'), (5, 'imbalance2'), (6, 't60\_count'), (7, 'imbalance4'), (8, 'adj\_price4'), (9, 'adj\_price2'), (10, 'agg180'), (11, 'adj\_price8'), (12, '\_id'), (13, 'agg60'), (14, 'trend180'), (15, 'imbalance8'), (16, 'trend60'), (17, 'agg30'), (18, 'mid30'), (19, 'agg120'), (20, 't60\_av'), (21, 't30\_av'), (22, 'trend120'), (23, 't120\_av'), (24, 'trend30'), (25, 't180\_av')]

**Accuracy:**

Training score is 0.916478380321  
Testing score is 0.916891270894

3. **EMBEDDED METHOD**:



Implemented the L1 Regularization technique Lasso Regression for selecting the features and tested them on a Linear Regression model. The following is the accuracy of the model.

**Features:**

Features sorted by their score:  
[(1.0, 'width'), (1.0, 'trend60'), (1.0, 'trend30'), (1.0, 'trend180'), (1.0, 'trend120'), (1.0, 't60\_count'), (1.0, 't60\_av'), (1.0, 't30\_count'), (1.0, 't30\_av'), (1.0, 't180\_count'), (1.0, 't180\_av'), (1.0, 't120\_count'), (1.0, 't120\_av'), (1.0, 'mid30'), (1.0, 'imbalance8'), (1.0, 'imbalance4'), (1.0, 'imbalance2'), (1.0, 'agg60'), (1.0, 'agg30'), (1.0, 'agg180'), (1.0, 'agg120'), (1.0, 'adj\_price8'), (1.0, 'adj\_price4'), (1.0, 'adj\_price2'), (1.0, '\_id')]

Since all these features have a score of 1.0 thus this is not an accurate measure of

feature selection for our case study.

**Conclusion for Feature Selection:**

Comparing the results of the above methods, Select Percentile is the technique which gives the maximum accuracy and thus the final features for our model are:

\*\*\*Features sorted by score: ['\_id', 'adj\_price8', 'agg180', 'imbalance2', 'adj\_price4', 'agg120', 'agg60', 'adj\_price2', 'imbalance4', 'width', 'trend180', 'agg30', 'trend120', 'imbalance8', 'trend60', 'trend30', 't180\_av', 't30\_av', 'mid30', 't30\_count', 't60\_av', 't120\_av', 't60\_count', 't180\_count', 't120\_count']

REGRESSION MODELS:

**NORMALIZING DATA:**

The categorical data is transformed into numerical data using **Label Encoding**

**MODELS:**

On each of the clustered dataset the following four regression models are run. The following is the performance of each of them. The performance of all the models is evaluated on the basis of R2 score.

The models are build on the entire dataset. The following is accuracy of the model.

**LINEAR REGRESSION:**

Training score is 0.733215791162  
Testing score is 0.717465020426

**NEURAL NETWORK:** (MLP with 50 hidden layers, which optimizes the squared-loss using LBFGS)

Training score is 0.977019787223

Testing score is 0.972865582276

**RANDOM FOREST:**

Accuracy of the model is 0.983368180291

**KNN: (k=11)**

The most optimal accuracy is fetched using 11 nearest neighbours

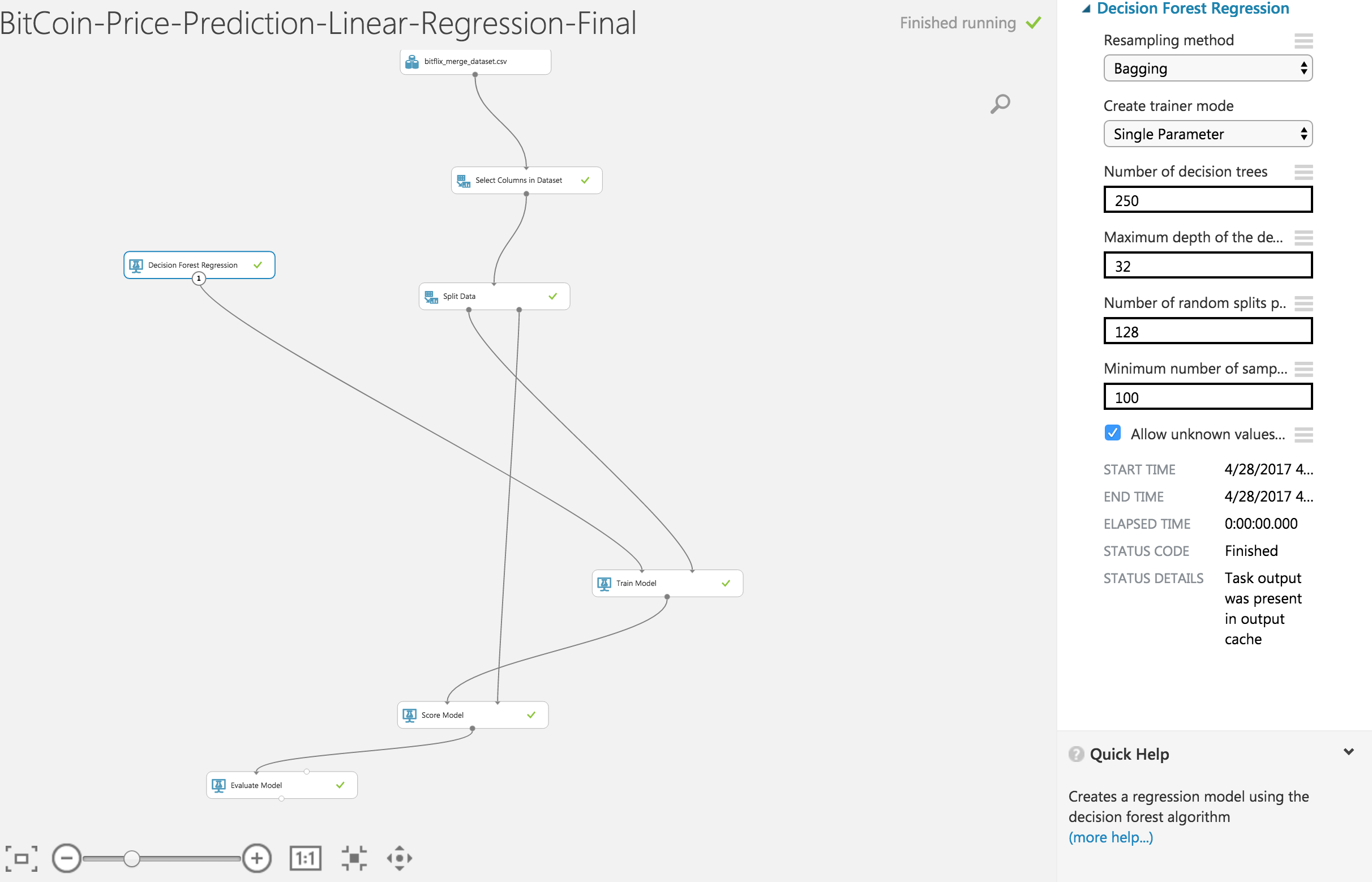
Accuracy of the model is .899980670914

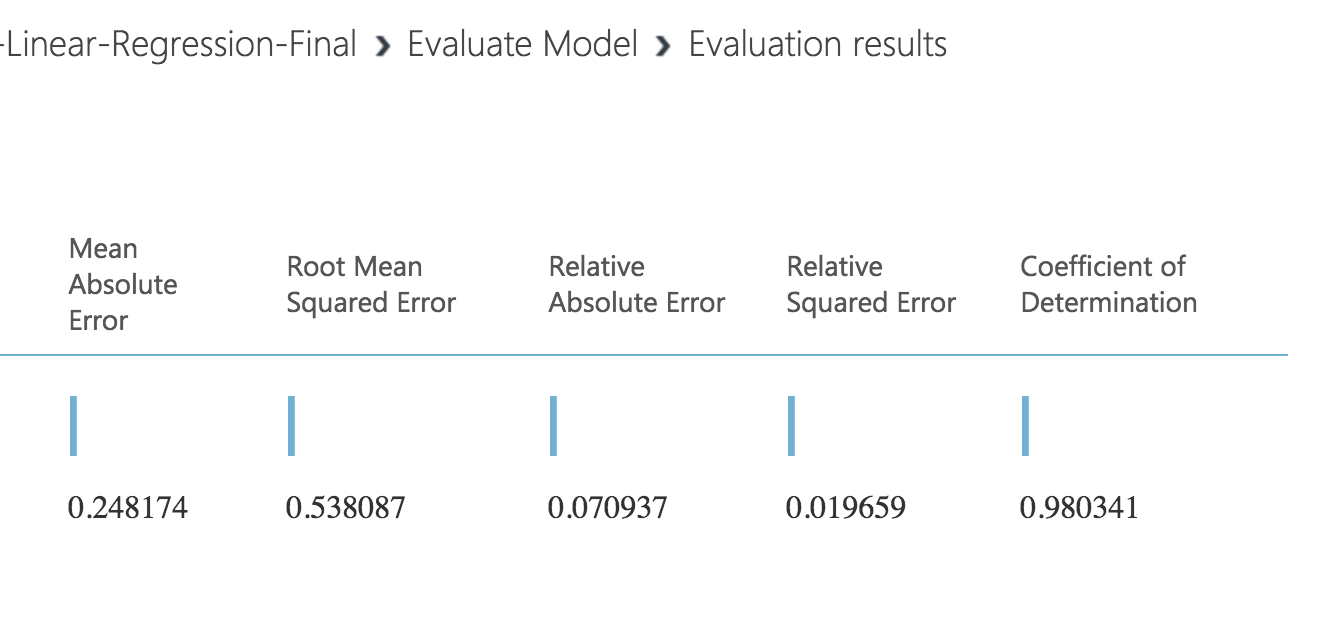
|  |  |
| --- | --- |
| ALGORITHM | TEST ACCURACY |
| LINEAR REGRESSION | 0.717465020426 |
| NEURAL NETWORK | 0.972865582276 |
| RANDOM FOREST | 0.983368180291 |
| KNN | .899980670914 |

# 

# AZURE ML IMPLEMENTATION:

Applied Random Forest on the whole dataset in Azure ML





DOCKERIZING THE PIPELINE

|  |
| --- |
| docker pull jainpranj/bitcoin |

|  |
| --- |
| docker **run** -ti bitcoin --module download\_blockchain\_api\_features DownloadBlockchainAPIFeatures --workers 4 --local-scheduler |

# 

# 

# RESULTS

The pipeline works as follows:

1. The borrower feeds his/her data into the system which calls the Random Forest Classification trained model and returns a result

“ -1 (negative change in price) /1 (positive change in price) ”

1. For bitcoin price prediction Random Forest gave least RMSE value and highest accuracy 98%. The overall model can be improved by collecting more data over a period of minimum 4 weeks. (to have wider range of bitcoin price)

# 

# REFERENCES

[1] Automated Bitcoin Trading via Machine Learning Algorithms IsaacMadan , Shaurya Saluja, Aojia Zhao

[2] [Benedikt Koehler](https://conferences.oreilly.com/strata/strata-ca-2016/public/schedule/speaker/135210)-[Working on the blockchain gang: Crunching and visualizing bitcoin data](https://conferences.oreilly.com/strata/strata-ca-2016/public/schedule/detail/47072)

[3] [Benedikt Koehler](https://conferences.oreilly.com/strata/strata-ca-2016/public/schedule/speaker/135210) (DataLion)-[Working on the blockchain gang: Crunching and visualizing bitcoin data](https://conferences.oreilly.com/strata/strata-ca-2016/public/schedule/detail/47072)

[4] [Danno Ferrin's](http://numisight.com/)- Numsight tool for visualizing bitcoin-Devavrat Shah and Kang Zhang

[5] [Bayesian Regression and Bitcoin](https://pdfs.semanticscholar.org/db15/1836543d8a70db1dabef3dee43637a7cd29f.pdf)

[6] [BitPredict](https://angel.co/christopher-bynum)- Christopher Bynum

TECHNOLOGIES

D3 js

Pickle

Luigi

Docker

Mongo Db

Flask

Azure Machine Learning

Skicit learn

Hadoop Mapreduce

AWS

DEMO LINK

Catch a demo of our presentation on youtube :

https://www.youtube.com/watch?v=mY9vZJzuLMw

WORK DONE

**Nehal Bhanushali:**

1. Web Scraping, Downloading Data from API for classification
2. Feature Selection And Engineering for Classification
3. Generating classification models for predicting sign of price change
4. Training classification models in Azure Machine Learning studio
5. Luigi pipeline-Classification and upload to S3
6. Simplifying the graph database with MapReduce

**Pranjal Jain:**

1. Downloading Data for **Bitniflix REST API** every second and Storing in **Mongo DB**
2. **Deriving New Features** and Data Cleaning
3. **Feature Selection** using various techniques
4. Applying **Regression** for prediction and generating models
5. **Azure Machine learning** to deploy model as **REST API**
6. **Exploratory Data Analysis** bitcoin using **Networkx** -Plotting interesting patterns
7. **Luigi** pipeline-linear Regression
8. **Dockerization** of application
9. Building **web application with Flask** for prediction
10. Deploying web application to Heroku
11. Building **interactive** web application user transactions using **D3 and Network X**

**Vasanti Mahajan:**

1. Downloading data for Anomaly detection case study
2. Data Cleaning and pre-processing
3. Feature Engineering and feature extraction for user and transactions data
4. Clustering for anomaly detection in the Bitcoin network
5. Exploratory data analysis of the network using NetworkX
6. Exploratory data analysis of the anomaly users and transactions
7. Visualising the graph in Neo4j