“PERFORMANCE ANALYSIS OF SUPPORT VECTOR MACHINE, REGRESSION TREE AND XGBOOST IN PREDICITING THE BITCOIN VALUE”

PROJECT REPORT

Submitted for CAL in B.Tech Large Scale Data Processing(CSE3025)

By

(Mayank Murali 15BCE1048)
(Debaditya Mitra 16BCE1341)

Slot: F2

Name of faculty: **M. Sivagami**

(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING)

November 1, 2018
FALL SEM 2018-2019

ABSTRACT

Bitcoin is the world's leading cryptocurrency, allowing users to make transactions securely and anonymously over the Internet. In recent years, The Bitcoin the ecosystem has gained the attention of consumers, businesses, investors and speculators alike. While there has been significant research done to analyse the network topology of the Bitcoin network, limited research has been performed to analyse the network's influence on overall Bitcoin price. To investigate the predictive power of blockchain network-based features on the future price of Bitcoin, we apply the primary performance metric for evaluation of the models was to check how close the predicted price was to the actual price the next day. If the predicted price was within 25$ of the actual price, we considered the prediction to be accurate. For this, few analytics models will be applied to the dataset and compared for performance. The analytics models proposed to be applied are SVM for Regression and Regression Tree and XGBoost. These models will be applied on the Bitcoin Dataset.
The goal is to ascertain with what accuracy can the direction of Bit-coin price in USD can be predicted.

INTRODUCTION

Bitcoin is a cryptographic convention and disseminated organize that empowers clients to, store, and exchange computerized money. The plan was first executed by Satoshi Nakamoto in January 2009. In view of the straightforwardness and security of the Bitcoin convention, the Bitcoin biological system has increased noteworthy footing from buyers, organizations, examiners and speculators. Prediction of mature financial markets such as the stock market has been researched at length [3][4].

Lately, Bitcoin has set up itself as the main decentralized, cryptographic money. All the while, the estimation of individual Bitcoin money has soar in esteem, from pennies to more than 1000 USD at its pinnacle, driving numerous to take to Bitcoin as a method for hypothesis. The prevalence of Bitcoin, the computerized idea of the money itself, and in addition the accessibility of high-dimensional system and valuing information make machine learning expectation of the Bitcoin cost especially fascinating.

Traditional time series prediction methods such as Holt-Winters exponential smoothing models rely on linear assumptions and require data that can be broken down into trend, seasonal and noise to be effective [5]. This type of methodology is more suitable for a task such as forecasting sales where seasonal effects are present. Due to the lack of seasonality in the Bitcoin market and its high volatility, these methods are not very effective for this task. Given the complexity of the task, deep learning makes for an interesting technological solution based on its performance in similar areas. Tasks such as natural language processing which are also sequential in nature and have shown promising results [6]. This type of task uses data of a sequential nature and as a result is similar to a price prediction task. The non-linear regression and the regression tree favour of artificial neural networks are favoured over support vector machine due to the temporal nature of the more advanced algorithms [7].

LITERATURE SURVEY

Our work builds on prior research to leverage blockchain network features, as a basis to conduct supervised machine learning prediction on the price of Bitcoin. Ron et. al [1] used the Union-Find algorithm to group accounts belonging to the same individual or entity. Analysis demonstrated that many attempts were made to obfuscate an entities' transaction histories, finding many long chains and oddly shaped clusters in the graph that could not have been produced organically.

Furthermore, their exploration demonstrated that while the net ow of the diagram far surpasses the quantity of Bitcoins available for use, most Bitcoins are in actuality not available for use and have not been moved or utilized since their mining. This finding verifies the theory that critical tax evasion for anonymization reasons for existing is as a result. Shah et al. guaranteed to create an effective Bitcoin value forecast procedure in light of Bayesian relapse with an inactive source demonstrate [2].

This model, initially created by Shah to foresee slanting subjects in Twitter, got critical accomplishment in anticipating Bitcoin cost, asserting a 50-day 89% ROI with a Sharpe proportion of 4.10, utilizing 10-second verifiable cost and Bitcoin restrict arrange book highlights. Madan et al. utilized bitcoin blockchain

arrange highlights, and in addition seconds-level verifiable bitcoin cost in recorded time deltas of 30, 60 and 120 minutes to create highlights for regulated learning. Utilizing arbitrary woods, SVM and binomial calculated relapse classifiers, value deltas 10 minutes later on were anticipated, getting consequences of around 55% accuracy.

DATASET DESCRIPTION

We use the Bitcoin transaction data available on the CS224W website, which contains every Bitcoin transaction made prior to April 7, 2013. All transactions are available on a public ledger, and this data set is a large text file containing a line for every transaction. Each line includes the transaction id, sender, recipient, value (in BTC), and a timestamp. Transactions involving multiple senders and multiple receivers are represented by multiple lines with the same transaction id.

This record accompanies a coordinating document containing one line for each gathering of addresses that seem together in some exchange, henceforth having a place with a similar client or element. We utilized this document to change our informational collection into a disentangled one filed by "client" substance instead of address utilizing the Union Find calculation, in view of the instinct that for any given exchange, just a single element could be the sender of that exchange, regardless of whether numerous sending accounts were utilized. The first informational collection contains about 37 million exchanges between approximately 6 million locations. Once lessened, we got a similar number of exchanges between a little more than 3 million extraordinary clients. We speak to the information in a coordinated chart, where every hub is a client and each edge is an exchange (from sender to beneficiary). Bitcoin mining is spoken to in the information as an exchange from one client to itself and is subsequently showed in the diagram as a self-circle. This portrayal of the information enables us to investigate different properties of the diagrams and furthermore utilize them as highlights in value expectation.

What's more, to perform value forecast we required a total chronicled posting of costs for Bitcoin. We obtained from api.bitcoincharts.com the whole histories of a few Bitcoin trades, where each line in a given history contains the timestamp and the conversion scale in USD. From these exchanges we processed the normal cost of Bitcoin over all exchanges as the authority Bitcoin cost at 15 second interims going back to soon after the main Bitcoin was mined.

All features are computed using only information available an hour prior to the target prediction time, and for node features we used both in and out-degree. The following features were extracted:

- btc_total_bitcoins
- btc_avg_block_size
- btc_hash_rate
- btc_cost_per_transaction_percent
- btc_estimated_transaction_volume_usd
- btc_estimated_transaction_volume
- btc_output_volume
- btc_n_transactions_excluding_chains_longer_than_100
- btc_n_transactions_excluding_popular
- btc_n_transactions_total
- btc_n_transactions
- btc_n_unique_addresses

Many of these features are somewhat correlated with the price of Bitcoin, though of those most are only loosely related.

PROPOSED METHODOLOGY

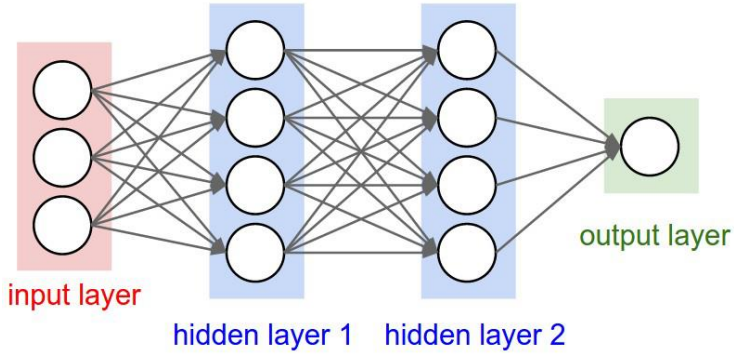
Learning Algorithms :

Support Vector Machine

Support Vector Machine (SVM) is a discriminative classifier that generates a separating hyperplane. Error tolerance budget is included to make separating hyperplane robust in case of inseparable class data. Linear decision boundaries are augmented to more complex boundary shape through kernel implementation (e.g. polynomial, Gaussian and radial kernel). SVM obtains decision boundary by creating a margin which maximizes the functional and geometric margins between classes. SVMs have received attention for the impressive performance in classification. SVM can also be augmented for regression problems as Support Vector Regression (SVR) optimizing response variable distance from the decision boundary. Below we provide formulation of the optimization and constraint model for SVM.

Support vector machine objective and constraints Neural Network Neural Networks are a family of learning methods inspired by biological neural networks by modelling a system of interconnected neurons, which are tuned based on iterative learning. Feedforward neural networks connect a multi-dimensional input into one or more hidden layers of neurons before predicting an output. Dropout is used to prevent overfitting of the model. Hidden layers are modelled by a fine transformation and final layers are modelled by softmax.

In addition, a non-linearity (such as the tanh function) is often used after each layer. Below is a visualization of a Feed-forward neural network with two hidden layers, similar to the one we implemented.



Regression Tree

Decision tree builds regression or classification models in the form of a tree structure. It brakes down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.

We can compute the accuracy individually per class by giving our classification function only samples from the same class and remember and count the number of correct classifications and incorrect classifications then compute *accuracy*:=correct/(correct+incorrect). We repeat this for every class. If we have a classification function that can accurately recognize class A but will output a random guess for the other classes then this results in an accuracy of 1.00 for A and an accuracy of 0.33 for the other classes. This already provides us a much better way to judge the performance of our classification function. An oracle always guessing the same class will produce a per class accuracy of 1.00 for that class, but 0.00 for the other class. If our test is useful all the accuracies per class should be >0.5. Otherwise, our test isn't better than chance. However, accuracy per class does not take into account false positives. Even though our classification function has a 100\% accuracy for class A there will also be false positives for A (such as a B wrongly classified as a A).

XGBoost

XGBoost has become a widely used and really popular tool among Kaggle competitors and Data Scientists in industry, as it has been battle tested for production on large-scale problems. It is a highly flexible and versatile tool that can work through most regression, classification and ranking problems as well as user-built objective functions. As an open-source software, it is easily accessible and it may be used through different platforms and interfaces. The amazing portability and compatibility of the system permits its usage on all three Windows, Linux and

OS X. It also supports training on distributed cloud platforms like AWS, Azure, GCE among others and it is easily connected to large-scale cloud dataflow systems such as Flink and Spark. Although it was built and initially used in the Command Line Interface (CLI) by its creator (Tianqi Chen), it can also be loaded and used in various languages and interfaces such as Python, C++, R, Julia, Scala and Java.

Its name stands for **eXtreme Gradient Boosting**, it was developed by Tianqi Chen and now is part of a wider collection of open-source libraries developed by the Distributed Machine Learning Community (DMLC). XGBoost is a scalable and accurate implementation of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed. Specifically, it was engineered to exploit every bit of memory and hardware resources for tree boosting algorithms.

The implementation of XGBoost offers several advanced features for model tuning, computing environments and algorithm enhancement. It is capable of performing the three main forms of gradient boosting (Gradient Boosting (GB), Stochastic GB and Regularized GB) and it is robust enough to support fine tuning and addition of regularization parameters. According to Tianqi Chen, the latter is what makes it superior and different to other libraries.

RESULTS AND DISCUSSION

We utilized a few relapse models to anticipate the cost of utilizing our enhanced list of capabilities. To set a standard forecast, we picked an innocent approach: we took the normal percent cost increment every hour (_1%) and connected it to the present cost to foresee the cost. The execution is assessed by utilizing non-linear regression model.

In addition to the above regression models, we also attempted several other such as Support Vector Machine, Regression Tree and XGBoost which are able to predict price change better. By the end of the test period, the baseline prediction has accrued noticeably greater total error than our predictions.

In addition to a regression-based approach, we formulated the problem as
a classification task, wherein we predict whether the price increased or decreased based on the error. We trained several different classifiers with the same set of features. In this case, our baseline model was simply choosing the most common output class, which was a price increase. We evaluated our performance by using classification accuracy as in Table1. The average results are summarized as follows:

Machine learning algorithm	Average accuracy obtained
Support Vector Machine	0.83346
Regression Tree	0.97945
XGBoost	0.16723

Table 1: Accuracy results

The results depicts the accuracy we have obtained for the bitcoin dataset as shown in Table 2. The accuracy results for different values of USD are as follows for Support Vector Machine:

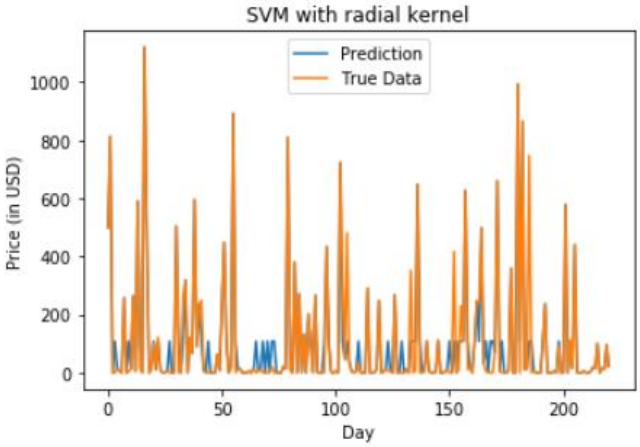
Value of USD as margin	Accuracy value
10$	0.8506787330316742
25$	0.832579185520362
50$	0.832579185520362
100$	0.8190045248868778

Table 2: Accuracy result for Support Vector Machine

The accuracy obtained by running the code for support vector machine is as follows for USD with a margin of 100, 50, 25 and 10 dollars:

```
Accuracy stats of SVM with radial kernel :
Accuracy with a margin of 100$ :  0.8506787330316742
Accuracy with a margin of 50$ :  0.832579185520362
Accuracy with a margin of 25$ :  0.832579185520362
Accuracy with a margin of 10$ :  0.8190045248868778
```

The graph portrays the Day vs. Price (in USD) plot. We can see that the error being little accurate, almost close to 0.8 , hence the blue graph coincides with the orange which indicates true data.




```
Average error :  24.1148417406
```

The following graph portrays Day vs. price (in USD) for SVM errors.

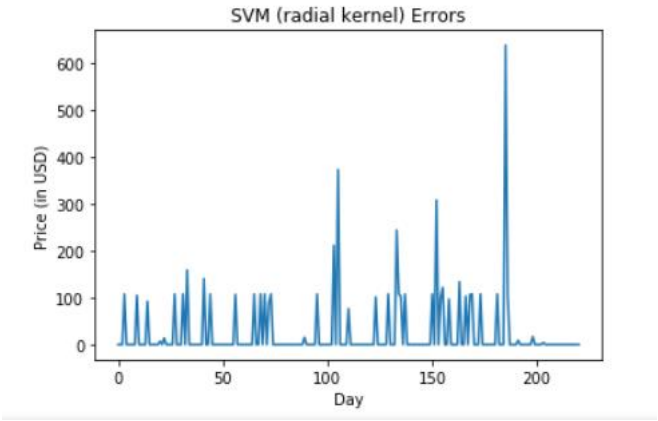


Table 3 shows the accuracy results when Regression Tree is applied.

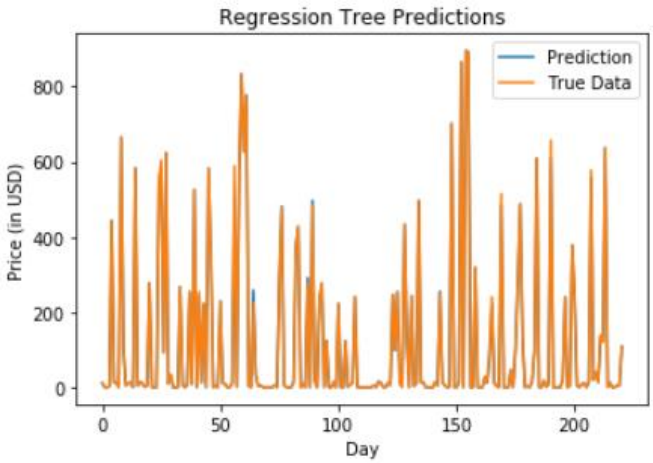
Value of USD as margin	Accuracy value
10$	1.0
25$	0.995475113122172
50$	0.9728506787330317
100$	0.9502262443438914

Table 3: Accuracy result for Regression Tree

The accuracy obtained by running the code for support vector machine is as follows for USD with a margin of 100, 50, 25 and 10 dollars:

```
Accuracy stats of Regression Tree Model :
Accuracy with a margin of 100$ :  1.0
Accuracy with a margin of 50$ :  0.995475113122172
Accuracy with a margin of 25$ :  0.9728506787330317
Accuracy with a margin of 10$ :  0.9502262443438914
```

The graph depicts the Day vs. Price (in USD) plot. We can see that the error being more accurate, almost close to 1, hence the blue graph coincides with the orange which indicates true data.




```
Average error :  1.7104801744
```

The following graph portrays Day vs. price (in USD) for regression tree errors.

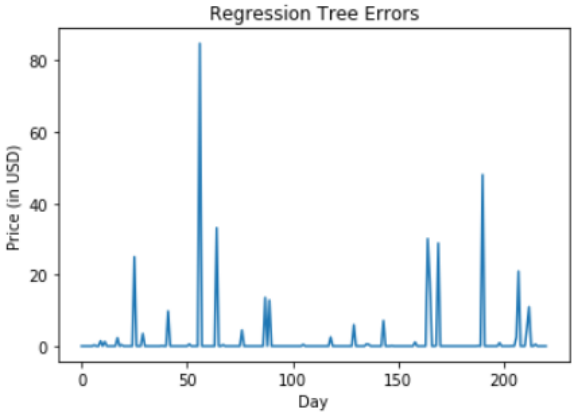
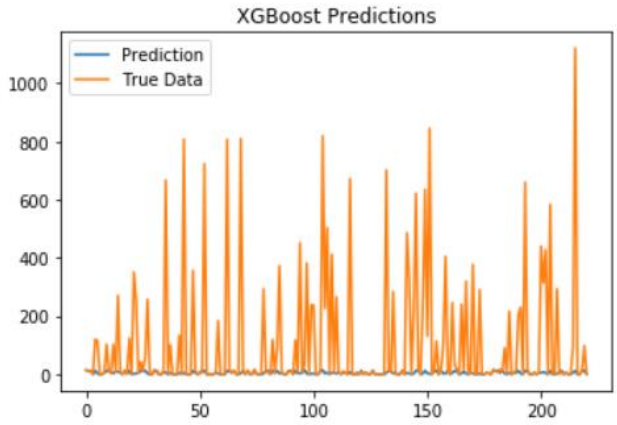


Table 4 shows the accuracy results when XGBoost is applied.

Value of USD as margin	Accuracy value
10$	0.0889466
25$	0.1655998
50$	0.1956647
100$	0.2187999

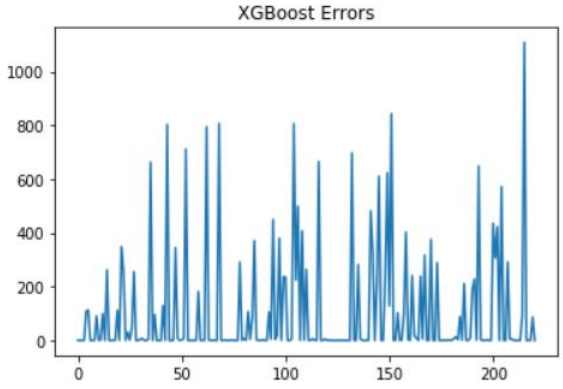
Table 4: Accuracy result for XGBoost

The graph depicts the Day vs. Price (in USD) plot. We can see that the predicted data being least accurate, almost close to 0, so the blue graph doesnt coincides with the orange which indicates true data.




```
('Average error : ', 102.96590350678869)
```

The following graph portrays Day vs. price (in USD) for SVM errors.



CONCLUSION

In analyzing the Bitcoin prediction, we extracted several network-based features, such as inertial agents, now features, and centrality measures. We ran three machine learning algorithms to figure out the accuracy of the models. While our models were able to beat baseline performance, we ultimately found that only limited amounts of predictive information are embedded in these network features themselves. In reaction, this seems reasonable, given that Bitcoin price is technically dictated by exchanges whose behaviors lie largely outside the realm of the actual Bitcoin prediction

The features based on the second by second Bitcoin exchanges are likely the most informative in predicting future price. Also, we have noticed that the best accuracy is provided by Regression Tree model followed by Support Vector Machine and finally XGBoost model. However, when account A sent many more Bitcoin than it received, Bitcoin price tended to increase, and when it received more than it sent, the price tended to decrease. Hence the accuracy of models are subjected to slight changes based on the dataset.

REFERENCES

[1] Dorit Ron, Adi Shamir Quantitative Analysis of the Full Bitcoin Transaction Graph, The Weizmann Institute of Science, 2012.
[2] Dorit Ron, Adi Shamir Quantitative Analysis of the Full Bitcoin Transaction Graph, The Weizmann Institute of Science, 2012.
[3] I. Kaastra and M. Boyd, \Designing a neural network for forecasting financial and economic time series," Neurocomputing, vol. 10, no. 3, pp. 215{236, 1996.
[4] H. White, \Economic prediction using neural networks: The case of ibm daily stock returns," in Neural Networks, 1988., IEEE International Conference on. IEEE, 1988, pp. 451-458.
[5] C. Chatfield and M. Yar, \Holt-winters forecasting: some practical issues," The Statistician, pp. 129{140, 1988.

[6] A. Karpathy, \The unreasonable effectiveness of recurrent neural networks," Andrej Karpathy blog, 2015.
[7] J. L. Elman, \Finding structure in time," Cognitive science, vol. 14, no. 2, pp. 179{ 211, 1990.