A PYSPARK SOLUTION FOR BITCOIN PRICE PREDICTION

Chi Wang
School of Computer Science
University of Nottingham
Nottingham, UK
psxcw4@nottingham.ac.uk

Luer Lyu
School of Computer Science
University of Nottingham
Nottingham, UK
scyll1@nottingham.ac.uk

Joel Ligma
School of Computer Science
University of Nottingham
Nottingham, UK
psxjl16@nottingham.ac.uk

Junfeng Wang
School of Computer Science
University of Nottingham
Nottingham, UK
alyjw19@nottingham.ac.uk

***Abstract*—As highly volatile cryptocurrencies become increasingly accepted among financial institutions and investors, the need for forecasting solutions is evident to make informed investment decisions. The aim of this paper is to develop a big data solution to predict the bitcoin closing price, applying techniques other than deep learning. To achieve this, we have designed four methods to impute large numbers of missing values and designed three local models to predict the bitcoin closing price. We compare their performances against Spark's global linear regression and random forest model.**

The results have shown that forward fill is not only the fastest but also a competitive imputation technique when it comes to the quality of the dataset to obtain forecasting results. Moreover, our experiments reveal that more data and an increase in granularity of the data improve predictive accuracy. Namely, 10 years of 1-minute interval data produce the best results. Unlike other works, the results of this paper did not support that blockchain information and technical indicators improve model performance. It was also found that tree-based models are not suitable for bitcoin price prediction as a regression task when working with raw closing price values. Their inability to capture price values of magnitudes not seen in training, causes them to exhibit poor performance. Finally, while our self-designed local ARIMA model exhibits poor predictive performance with an RMSE of 868.22, the results indicate that our local linear regression and vector ARIMA models are suitable for the big data context. Both models exhibit efficient training times and RMSEs of 66.67 and 65.43, respectively.

***Index Terms*—Bitcoin Price Prediction, PySpark, ARIMA, VAR, Linear Regression**

I. Introduction

Bitcoin, the best-known cryptocurrency, was created by Satoshi Nakamoto [1] in 2009 as a decentralized digital currency with open transaction records. This decentralized ledger of transaction records is made possible through blockchain technology. This technology enables people to verify transactions without a central clearing authority since it works as a peer-to-peer network. Thus, transactions are recorded and made public upon peer verification. Bitcoin can be seen as a means of exchange, similarly to fiat currencies, to purchase products and services [2]. In recent years, cryptocurrencies such as bitcoin gained a huge rise in popularity with businesses and individual investors. It is currently the most popular cryptocurrency, reaching a peak of daily number of bitcoin transactions of around 400,000 in January 2021. While for many, Bitcoin may be a store of value, others trade it in a similar way to stocks. However, a big difference is that the crypto market is not limited to conventional stock exchange opening times. The crypto market runs around the clock, every day of the year. As cryptocurrencies experienced a significant surge in price and trading volumes in recent years, the ability to accurately predict their prices has become increasingly sought after by businesses and individual investors [3]. Therefore, it goes without saying that the need for forecasting solutions to make informed investment decisions is evident.

The aim of this paper is to investigate a big data solution using PySpark to predict the future bitcoin closing price. To achieve accurate predictions, a suitable dataset is needed. Therefore, data of different time periods and granularities will be explored. However, with an increase in granularity a new problem arises. Missing values may occur in very granular data as there may be timestamps without any completed transactions. Additionally, unforeseen technical issues can cause errors in recording data for a specific timestamp. Imputing these missing values is a major challenge to overcome to enhance predictive performance and create a reliable forecast.

Few have done work in a distributed way, incorporating more data for better performance. With the amount of usable data increasing daily, the contribution of our research will focus on a scalable solution for bitcoin price prediction. Concretely, we will design four imputation methods in PySpark to tackle the problem of missing data. In addition, a local linear regression, autoregressive integrative moving average (ARIMA) and Vector Autoregression (VAR) model were designed, capable of efficient training due to harnessing the power of distributed, parallel processing. These contributions can be especially beneficial in future when the input data grows beyond the capabilities of a single machine.

As a disclaimer, this paper will not thematize the use of deep learning techniques for modelling nor sentiment analysis, google trends and volume of tweets on twitter as input features. These have been extensively explored in other works.

This paper is structured as follows. First, a review of related works will be presented in which different models, input features and big data solutions are described that have been applied to the same or similar problems in the past. Second,

our proposed methodology is explained which covers data collection and data pre-processing steps. Next, the experimental setup of this research is introduced. This section describes our selected performance metrics, datasets, validation procedures and definition of baselines. Subsequently, the findings of our research are presented in the Results and Discussion section. Lastly, in the conclusion section covers a reflection of our work and future steps for this research.

II. Related Works

A. Statistical & Machine Learning Models

Relevant literature shows that researchers have applied traditional statistical forecasting models such as ARIMA and PROPHET as well as machine learning models such as Linear Regression (LR), Random Forest (RF), Long Short Term Memory Recurrent Neural Network (LSTM-RNN) and others with varying degrees of success [4], [5], [6], [7], [8]. In summary, the results suggest that there are two perspectives used to build forecasting models: statistical and machine learning.

Among statistical approaches, ARIMA is supported by numerous researches to have the best performance on the task of predicting bitcoin price [9], [10]. Most of the existing work using ARIMA on price prediction problems are using statsmodel library from python, and very few have implemented it in a distributed way as a big data solution. Kyungjoo et al. [11], Merh et al. Reference [12] further pointed out that when predicting the stock market, which is a similar problem domain as predicting bitcoin price, ARIMA can outperform sophisticated deep learning neural networks, especially for short-term predictions. This paper will propose implementations of ARIMA with single or multiple feature inputs in PySpark.

However, with additional data, for example Twitter hashtag information, machine learning methods like RNN will be able to outperform ARIMA by a lot [9]. In this project, we seek to find a solution that is suitable for distributed implementation in PySpark, Although, deep artificial neural networks like RNN also have achieved promising results in time series problems, its training process will require the massive integration for data across rows following the timeline, which is not efficient to implement in a distributed way across nodes. Purbarani and Jatmiko [13] obtained the lowest MSE value of 0.0004 using LR implemented in a spark cluster environment with bitcoin price and multiple currency information. Görür's [14] RF model implemented in a distributed way yielded the lowest RMSE of 576 on predicting bitcoin price on a daily basis. Thus, we will compare the performance of machine learning and statistical forecasting approaches suitable both for a big data implementation and solving the bitcoin price prediction problem: LR and ARIMA.

B. Features

Besides the choice of model, researchers have also investigated the influence of input features for Bitcoin price prediction. Researchers have concluded that the factors that affect the Bitcoin closing price depend on whether the goal is short-term or long-term forecasting. For the purpose of this paper, the features affecting short-term forecasting will be further discussed.

Directly related information appear to be among the strongest predictors for the future closing price. These consist of the Bitcoin stock market data, the demand (transaction volume), the underlying blockchain technology data as well as technical indicators [15], [16], [17].

A combination of Bitcoin stock market data and blockchain variables such as hash rate, difficulty, miner's revenue, total number of unique Bitcoins and more have been successfully applied in past research [18], [16]. In addition, the impact of user and network activity on the price prediction has been investigated. The results suggest that those have a high impact on Bitcoin price [19].

Regarding technical indicators, the influence of moving average (average values of certain time period), Relative Strength Index (indicator based on recent closing price to evaluate the strength of the market), Rate of Change, Moving Average Convergence Divergence (a momentum indicator showing the relationship between two different moving averages), and other technical indicators was examined. Researchers found that these indicators can have a strong effect on the forecasting performance [2].

Past research also thematized macroeconomic factors. For instance, Jang and Lee [18] have experimented using macroeconomic factors such as the US stock market indices NASDAQ, S&P 500 and Dow Jones Industrial Average and others as well as gold and crude oil prices as input features to improve model performance. The use of exchange rates of major fiat currencies (EUR, JPY, GBP, etc.) to enhance the forecasting accuracy has also been explored [16]. However, it was suggested that due to Bitcoin's unique characteristics, macroeconomic factors only have a weak relationship with its closing price in the short-term [20]. Generally, the price development of cryptocurrencies appears to be relatively isolated from market shocks and separated from popular financial assets, making them comparably difficult to predict [21].

The literature further reveals that alternative approaches such as Google Trends, volume of tweets on Twitter and social media sentiment analysis to price prediction have also been researched [16]. The findings suggest that there is a positive correlation between those and the Bitcoin price. Hence, they are acknowledged as potentially suitable inputs for price forecasting [7], [22]. However, as aforementioned, this will not be further thematized in this paper.

III. Proposed Methodologies

In this section, we describe our data collection procedure, how we plan to deal with the large number of missing values, how we converted the original time-series problem into a regression task and present our self-designed local linear regression, ARIMA and vector ARIMA models.

It should be noted that the local model and global model in this article are both built on the spark distributed computing

framework. This paper defines that each node independently runs a complete algorithm as a local model, and all nodes jointly run a prediction algorithm as a global model.

A. Data collection

Based on previous works, a preliminary set of suitable candidate features has been identified. These are Bitcoin stock data, Bitcoin blockchain information as well as feature-engineered technical indicators.

The OHLC (open, high, low, close) prices, volume of bitcoin, volume of the corresponding currency in transaction and the weighted price at a one-minute interval from Kaggle Bitcoin Historical Data[1]. The time interval of the dataset is between December 31, 2011, 8:30 (a.m.) to December 31, 2020, 00:00 (p.m.).

The Blockchain data of the same time span was crawled from the official blockchain website[2]. Features collected of blockchain are hash rate, mining difficulty, confirmed transactions per day, estimated transaction value, total transaction fees, miners' revenue (USD).

The feature-engineered technical indicators for this project are the seven simple moving average with different time periods. These are 5-days, 7-days, 10-days, 20-days, and 50-days to cover short-term trends in the price as well as 100-days and 200-days to cover long-term trends.

$$MA(n) = \frac{\sum_i^n y_i}{n} \tag{1}$$

Where n is the number of observations during the required time period, y_i is the value of each observation.

Additionally, the rate of change in price has been computed.

B. Data Imputation

For the full dataset of with one-minute interval, there are 1,243,608 values missing. To restore the continuity of the data for training the model, different imputation techniques deigned in a distributed way are experimented.

There are four possible ways of filling the values of missing data adopted in this project: interpolation, forward imputation, backward imputation and mean imputation. Interpolation is done by taking the last available data and the next available data and retrieving the linear formula between the values and the row numbers. This formula will be used to calculate the null data.

Window function in PySpark is employed to gather information between rows. As shown in Algorithm 1, the previous available data and the next available data information will be added to the rows where missing data is spotted. Different partitions of the RDDs will be passes to each node to complete this process.

Algorithm 1 Interpolation

Input: column c having missing values
Output: column c with interpolated data

1: **for all** row r_i in c **do**
2: **if** $(r_i \neq null)$ **then**
3: $RowCurrent_i \leftarrow i$
4: $ValueCurrent_i \leftarrow r_i$
5: **end if**
6: **end for**
7: **for** i = 1 to $RowNumberInThisNode$ **do**
8: **if** $(r_i = null)$ **then**
9: $RowPrevious_i \leftarrow RowCurrent_{i-1}$
10: $ValuePrevious_i \leftarrow ValueCurrent_{i-1}$
11: **end if**
12: **end for**
13: **for** i = $RowNumberInThisNode$ to 1 **do**
14: **if** $(r_i = null)$ **then**
15: $RowNext_i \leftarrow RowCurrent_{i+1}$
16: $ValueNext_i \leftarrow ValueCurrent_{i+1}$
17: **end if**
18: **end for**
19: **for all** row r_i in c **do**
20: **if** $(r_i = null)$ **then**
21: $r_i \leftarrow (ValueNext_i - ValuePrevious_i) * (RowNext_i - RowCurrent_i)/(RowNext_i - RowPrevious_i)$
22: **end if**
23: **end for**
24: **return** c

Similar concepts are used to design the function to impute missing data forwards and backwards shown in Algorithm 2 and 3 respectively. A window following the time order is applied to retrieve the last available and the previous available data.

Algorithm 2 Forward Fill Imputation

Input: column c having missing values
Output: column c with interpolated data

1: **for all** row r_i in c **do**
2: **if** $(r_i = null)$ **then**
3: $r_i \leftarrow r_{i-1}$
4: **end if**
5: **end for**
6: **return** c

Algorithm 3 Backward Fill Imputation

Input: column c having missing values
Output: column c with interpolated data

1: **for all** row r_i in c **do**
2: **if** $(r_i = null)$ **then**
3: $r_i \leftarrow r_{i+1}$
4: **end if**
5: **end for**
6: **return** c

Mean imputation combines the information that can be retrieved from backward filling and forward filling. The mean

[1]https://www.kaggle.com/mczielinski/bitcoin-historical-data
[2]https://www.blockchain.com/

is calculated for each row that contains a missing value as shown in Algorithm 4.

Algorithm 4 Mean Imputation

Input: column c having missing values
Output: column c with interpolated data

for i = 1 to $RowNumberInThisNode$ **do**
  if $(r_i = null)$ **then**
    $ValuePrevious_i \leftarrow ValueCurrent_{i-1}$
  end if
end for
for i = $RowNumberInThisNode$ to 1 **do**
  if $(r_i = null)$ **then**
    $ValueNext_i \leftarrow ValueCurrent_{i+1}$
  end if
end for
for all row r_i in c **do**
  if $(r_i = null)$ **then**
    $r_i \leftarrow (ValuePrevious_i + ValueNext_i)/2$
  end if
end for
return c

C. Converting the Time Series to a Regression Task

To be able to also use regression models for this task, the time series problem is translated into a regression problem, by shifting the next close price $Price_{t+1}$ to the previous row t as the label to predict, combining the PySpark window and lag function. As a result, regression models can now be applied

D. Proposed Models

The aim of our work is to implement linear regression, ARIMA and Vector ARIMA as local models for bitcoin price prediction in a big data context. Based on our research of related works, these types of models show promising performances in similar tasks in a non-big data context. With our implementation, we expect to increase training efficiency, achieve timely predictions, and ensure scalability for large datasets.

The implementation of our customised Spark local model structure for regression and time-series forecasting is as follows. In stage 1, the data is split into training and test data without shuffling. This is done to ensure that the test split contains the most recent data. In stage 2, both training and test data are split into several partitions, once again without shuffling. Then, in stage 3, a data transformation function is applied on each partition to transform Spark RDD to Pandas DataFrame. After that, the respective models are fit on each partition in stage 4. Then, all the trained models are collected to the Spark driver and broadcast to the Spark workers in stage 5. In stage 6, the models are then utilised to make predictions on each partition of the test data. A weighted summation method is applied to the predictions of each model to make sure the models trained on the more recent data have the largest influence on the result. At last, the predictions on each partition of test data are aggregated to the Spark driver to form a whole prediction of the test data. All three models follow the same process. An illustration can be seen in Figure 1 below.

ARIMA can be divided into three processes, Integrated difference (I), Auto Regression (AR) and Moving Average (MA), as shown in the formulas (5), (6), (7).

$$I : \breve{y}_t = y_t - y_{t-1} \tag{2}$$

$$AR : y_t = \mu + \sum_{i}^{p} \gamma_i y_{t-i} + \epsilon_t \tag{3}$$

$$MA : y_t = \mu + \sum_{i}^{p} \theta_i \epsilon_{t-i} + \epsilon_t \tag{4}$$

Where y_t is the price at time t. γ_i and θ_i are obtained coefficients.

IV. Experimental set-up

In this section, the experimental set-up of our research is introduced.

A. Details of the Experiments Performed

First, to assess which of the four imputation techniques will be selected for our final solution the efficiency and effectiveness will be compared. Second, the influence of different time periods and time granularities on model performance will be explored to select the optimal time-period and granularity. Subsequently, additional input features, identified in the literature review, will be added to examine their influence on the predictions. Lastly, our self-designed local linear regression, ARIMA and vector ARIMA models are applied on the final dataset with the aim of investigating whether they show competitive performance and efficiency. An overview is depicted in Figure 2 below.

B. Performance Metrics

As aforementioned, this project regards price prediction as time series and regression problem, and uses the metrics shown in equations (5), (6), (7) to compare the performances of the models:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i}^{n}\left(\widehat{y_i} - y_i\right)^2} \tag{5}$$

$$R^2 = 1 - \frac{\sum_i^n \left(\widehat{y_i} - y_i\right)^2}{\sum_i^n \left(\overline{y_i} - y_i\right)^2} \tag{6}$$

$$R^2_{adjusted} = 1 - \frac{\left(1 - R^2\right)(n-1)}{n-p-1} \tag{7}$$

, where n is the number of samples, y_i is the predicted value, y_i is the ground truth and p is the number of features. Root Mean Squared Error (RMSE) is the most used evaluation metric for regression models, which calculates the mean square root of difference of the predicted values and the expected values. In contrast to the other metrics such as Mean Squared

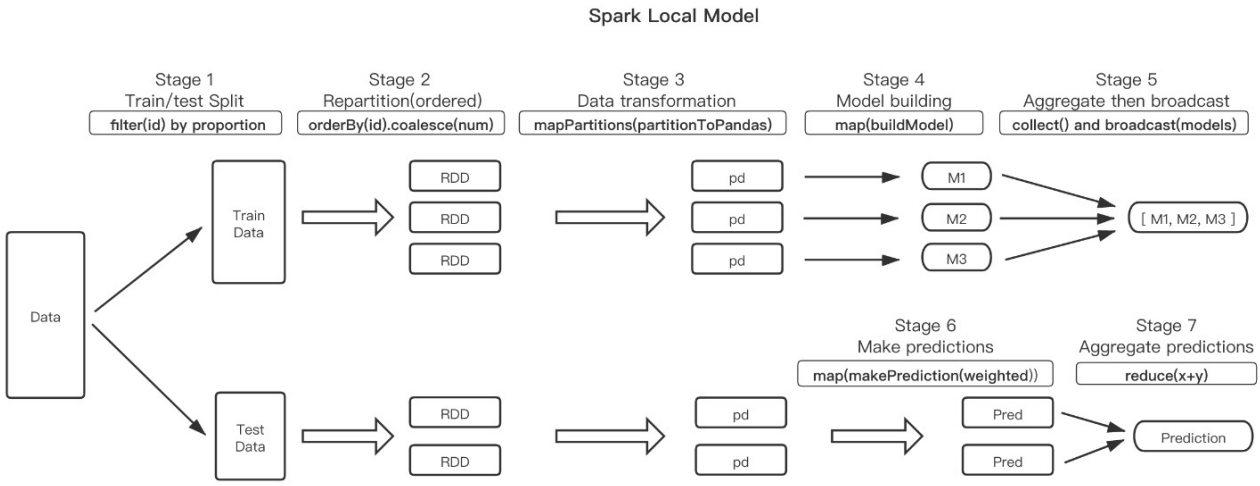

Fig. 1. General Design for Local Models

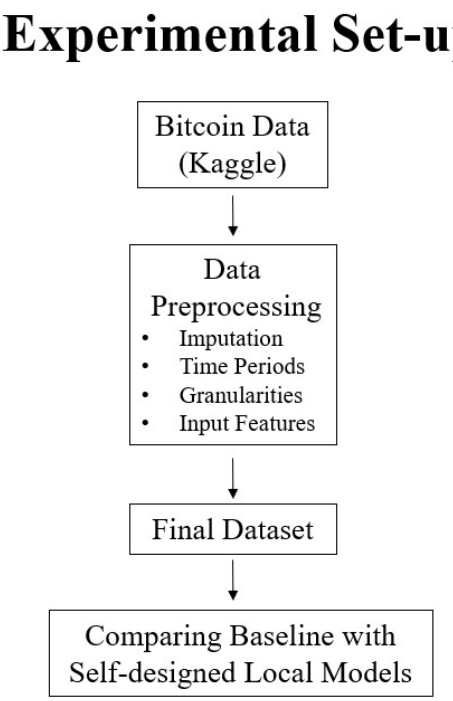

Fig. 2. Experimental Set-up Overview

Error, the RMSE can be more intuitively compared against the real values, as the units are on the same scale. Generally, for the RMSE, a low value is desired. To further illustrate the accuracy, the adjusted R-Squared ($R^2_{adjusted}$) metric is proposed. R^2 shows the proportion of variance in the output feature that can be explained by the model using all the input features. An issue of the R^2 metric is that it tends to increase with the total number of input features. To counter this the adjusted R-Squared can be used. The adjusted R^2 value accounts for the total number of predictors in the model and is therefore a more reliable metric. The adjusted R-Squared ranges from 0 to 1 and the closer the output is to 1, the better the model. Lastly, we consider execution or training time in seconds to evaluate the efficiency of our solutions.

C. Datasets Considered

First, this paper examines three datasets with different time periods. The four and two-year periods consist of the most recent four and two years.

- Bitcoin Data (Kaggle), 10 years, 1-minute intervals
- Bitcoin Data (Kaggle), 4 years, 1-minute intervals
- Bitcoin Data (Kaggle), 2 years, 1-minute intervals

Second, two datasets with different granularities are considered. For the daily data from Kaggle, only 8 years were available. These are:

- Bitcoin Data (Kaggle), 10 years, 1-minute intervals
- Bitcoin Data (Kaggle), 4 years, 1-day intervals

Lastly, one dataset with additional input features is studied.

- Bitcoin Data (Kaggle), 10 years, 1-minute intervals, including blockchain and technical indicator features

Experiments were initially conducted using the cloud platform Databricks. However, the service stopped working for us which forced us to use a subset to conduct final experiments. This subset contains 1-month, 1-minute intervals of the latest bitcoin data.

- Bitcoin Data (Kaggle), 1 month, 1-minute intervals

D. Validation Procedures

All techniques and models are evaluated using a training and test split with a 70-30 ratio. Additionally, our self-designed local models as well as our baseline model are evaluated using cross validation to obtain a more robust estimate of the generalization error for final reporting.

1) Training and Test Split:

In time-series tasks, the training and test split is conducted slightly different than in conventional regression problems. Concretely, the data must not be shuffled because the sequence of the data must be preserved. Otherwise, information about the future is leaked into the training process and the model performance becomes unreliable. This means that the training data will always be information dating back longer in the past than the test data. For our local models, we manually implemented the training and test split in a way which keeps the structure of the sequence intact in each node as PySpark automatically shuffles the data.

2) Cross-Validation:

In order to obtain a robust estimate of the generalization error, we implemented Blocked Time Series (BTS) Cross Validation, which is a cross validation method specifically used for time series problem. To prevent the leakage of information in the previous folds, BTS separates every split of the dataset into independent segments [23]. Within each segment, the training and testing set is sampled as shown in Figure 3. The advantage of using this approach is that the fairness of the evaluation of the model is ensured.

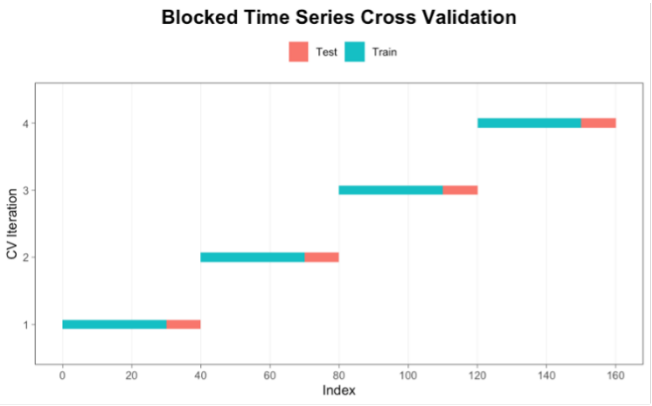

Fig. 3. Blocked Time Series Cross Validation

E. Definition of Baseline

The baseline model selected for all our experiments is Spark's built-in, global linear regression model. It is used to compare our self-designed local models against and put the performance into context.

V. Results and Discussion

In the following section, the results of our research will be discussed, and the best performing solutions identified.

A. Results of Identifying the Optimal Dataset

1) Imputation methods: In this section, the results in terms of efficiency and effectiveness of the four imputation techniques are presented.

To compare the efficiency, the different imputation methods are applied to the Bitcoin Data (Kaggle), 10 years, 1-minute intervals dataset, containing approximately 1.2 million missing values. The results are shown in Table I. With only 13 seconds runtime, it is concluded that forward fill takes the least time to run. In comparison, the second fastest method is Mean fill which takes 28.05 seconds. It is assumed that it is easier for Spark to fetch data following the order while reading data backwards is more time-consuming. The dataset we used is organized in a chronological order thus forward fill can take advantage of that and result in a shorter operation time. However, all other three methods require reading data in a reverse order across column one or more time as shown in Algorithm 1, 3 and 4.

TABLE I
DIFFERENT IMPUTATION METHODS AND TIME COMPARISON

Method	Time (in seconds)
Interpolation (INT)	32.84
Forward Fill (FF)	13.05
Backward Fill (BF)	29.71
Mean Fill (MF)	28.05

All following RMSE values are reported based on a 30% test split, for each of the datasets used. The two models used to assess the influence of the imputation techniques, time periods, granularities and input features on the prediction error are Spark's built-in, global linear regression and random forest models.

With that in mind and considering the large number of missing data, we experimented how different imputation method can affect the final prediction error. The results are as follows.

TABLE II
PERFORMANCE COMPARISION OF IMPUTATION TECHNIQUES ON THE 10 YEARS, 1-MINUTE DATA ACCORDING TO RMSE

Model	INT	FF	BF	MF
Linear Regression	23.087	23.096	23.096	23.088
Random Forest	9540.1	9415.8	8175.4	9275.3

From Table I and II, it can be concluded that forward fill (FF) is the best solution as it is the fastest and the model performances using forward filled data are quite competitive. Another interesting fact is that Spark's built-in random forest model seems to perform significantly worse on this task than the linear regression model.

2) Investigating Different Time Periods and Granularity: Next, the results of our experiments on different time periods and granularities are discussed. As seen in Table III., the longer the time-period used as input for our models, the better the model performance. Once again, Spark's built-in, global linear regression outperforms its random forest model for each time-period, with 10 years, 1-minute interval data showing the lowest RMSE (23.087).

The results of our experiments on different time granularities show that the 10 years, 1-minute interval dataset produced better results than the 10 years, daily interval dataset as seen in Table IV. Again, the random forest model does exhibit poor results on this task.

TABLE III
INITIAL EXPERIEMENTS WITH TIME PERIOD CONSIDERED

Model	2 Years RMSE	4 Years RMSE	10 Years RMSE
LR	44.270	34.435	23.087
RF	20925.8	14034.6	9540.1

TABLE IV
INVESTIGATING DIFFERENT DATA GRANULARITIES ON THE 10 YEAR BITCOIN DATA ACCORDING TO RMSE

Model	1-min data	1-day data
LR	23.087	865.724
RF	9540.1	7120.7

Incidentally, on the performance of random forest, our hypothesis is that tree-based models are not suitable for the way we set up the problem of bitcoin price prediction. This is because during training, a regression tree splits the data based on information theory until a leaf node is reached based on certain stopping criteria. The leaf node will be assigned a label according to the mean of all training examples in the node. This label will then serve as the prediction for each test example that reaches a specific leaf node. As the closing prices in our training data range from 178 USD to around 20,000 USD and the test data mostly contains values that exceed 20,000 USD, this demystifies the poor performance results. Because of the way tree-based models arrive at a prediction, they cannot capture the trend of values they have not seen during the training phase. This fact is depicted in Figure 4, which shows the random forest predictions against the ground truth for the latest 6 months. Hence tree-based models will not be considered further.

Fig. 4. Random Forest Model Predictions on the Test Data

In comparison, linear regression seems to be quite suitable for this task. It successfully finds the relationship between the features and the predicting price and yield lower RMSE values than random forest.

In summary, based on all results presented thus far, we conclude that not only more data, but also more granularity increases model performance for bitcoin price prediction. It could be that the aggregation of data in the case of daily intervals causes some information loss which is why more granularity increases performance. As a result, the optimal time-period and granularity are 10-years of historical data in 1-minute intervals and a big data solution appears to be reasonable to handle the size of the dataset, especially as the size of the data grows daily.

3) Adding Blockchain Data and Technical Indicators: As a final step to arrive at a suitable dataset for bitcoin price prediction we investigated the influence of additional input features. As seen in Table V, the results show that adding these additional input features does not improve predictive performance. This does not confirm the findings from the literature review. It appears that the way our experiment is set up, these additional features are not beneficial. In fact, the outcome is slightly worse with an RMSE of 23.496. Perhaps those features ended up adding noise to the data.

TABLE V
PERFORMANCE COMPARISON OF LR ON THE ORIGINAL DATASET AND THE DATASET WITH ADDITIONAL FEATURES

Dataset	RMSE	Adj R-Squared
Original	23.087	0.999
Original including Blockchain and Technical Indicators	23.496	0.999

As a result, the original dataset containing 10-years of historical bitcoin stock data in 1-minute interval will be used moving forward.

B. Comparing the Performance of our Local Linear Regression, ARIMA and Vector ARIMA with the Baseline

Now that the optimal time-period, granularity, and input features are identified, this dataset will be used to assess the performance of our self-designed local linear regression, ARIMA and vector ARIMA models. Our models are compared to Apache Spark's global linear regression model which we set as our baseline model earlier. When comparing the model results, we evaluate their suitability for the big data context through training time and prediction error. As aforementioned, it is important to note that the following results are obtained on a subset of the 10-year, 1-minute interval data, containing only the most recent month of data. The reason for this is that unfortunately Databricks was not available to us for further modelling.

Looking at Table VI., the two linear regression models performed quite similarly with RMSEs of approximately 59. Both local ARIMA models perform worse. The vector ARIMA model shows an RMSE of around 99 while the regular ARIMA model stands out due to very poor performance with an RMSE of roughly 854. It appears that using more than one feature for prediction improves predictive performance. In terms of predictive accuracy, Spark's global linear regression exhibits slightly better performance than our self-designed linear regression model and is therefore the best performing model by a very slight margin. As global models make use of all of the data for training, these results are expected. Nonetheless, our local linear regression implementation shows very competitive results.

Next, if we consider the time component, our local linear regression and vector ARIMA implementations are faster than

TABLE VI
PERFORMANCE COMPARISON ON OUR MODELS WITH THE BASELINE

Model	RMSE	Time (in seconds)
Global LR	59.1138	0.0693
Local LR	59.1420	0.0036
Local Vector ARIMA	99.3003	0.0095
Local ARIMA	854.7430	0.49

TABLE VII
BLOCKED TIME SERIES CROSS VALIDATION MODEL COMPARISON

Model	RMSE
Global LR	65.3442
Local LR	66.6728
Local Vector ARIMA	65.4367
Local ARIMA	868.2242

Spark's global linear regression model. The local model only needs to train on each partition's data, and there is no further data communication between each node during the model training process compared to the global model. This the reason why the local model is faster than the global model. Keep in mind that the difference shown seems small, but these are the results on only one month of data with a one-minute interval.

The nature of the algorithm of ARIMA might account for the low efficiency of ARIMA model. The dataset used in this article is obviously not a stationary series, so integrated difference operations shown in equation (2) on every pair of data points are required. During the process of AR and MA as shown in equation (3), (4), it is also necessary to perform thorough calculations on each pair of historical data. These two processes are more computationally expensive than linear regression calculations and memory call requirements, which increase the running time. Furthermore, the efficiency of the implementation may also be dependent on the packages used to implement the local model. Our regular local ARIMA model uses the Python version of R's popular auto.arima implementation which could be an explanation for the comparably slower training time. The implementation might not yet be optimal. In contrast, for our local vector ARIMA model, we relied on an implementation of the statsmodels package.

Finally, to obtain a more reliable estimate on the generalization error, we conducted a block time series cross validation as described in the experimental set up section. The results are depicted in Table VII. Although its performance experienced a slight dip, the results confirm that Spark's global implementation of linear regression remains the best performing model with a cross-validated RMSE of around 65.3. Surprisingly, our local vector ARIMA model exhibits more competitive performance now with a cross-validated RMSE of 65.4, making it the second-best performing model overall and the best performing model out of our proposed solutions. This reason for this finding could be that the training and test split ratio from earlier, was unfavorable for our vector ARIMA but the cross-validated results show a more robust picture of its performance. This is followed by our local linear regression model exhibits a cross-validated RMSE of 66.7. Lastly, the poor performance of our regular ARIMA model was further confirmed with a cross-validated RMSE of 868.2.

As a result, we infer that our self-designed local linear regression and vector ARIMA models can compete with Spark's global implementation in terms of performance, while at the same time offering an efficient and scalable solution for the big data context. Due to the nature of the local implementation design, our solution can be scaled for even larger datasets. This would simply require additional partitions, extending what is depicted in Figure 1.

VI. CONCLUSION

In this paper, we have developed a big data approach for bitcoin price prediction using PySpark. We have designed four methods to impute large numbers of missing values and designed three local models to predict the bitcoin closing price. The results have shown that forward fill is not only the fastest but also a competitive imputation technique when it comes to the quality of the dataset to obtain forecasting results. Moreover, our findings reveal that more data and an increase in granularity of the data improve predictive accuracy. Unlike other works, the results of this paper did not support that blockchain information and technical indicators improve model performance. It was also found that tree-based models are not suitable for bitcoin price prediction as a regression task when working with raw closing price values. Their inability to capture price values of magnitudes not seen in training, causes them to exhibit poor performance. Finally, while our self-designed local ARIMA model exhibits poor predictive performance, the results indicate that our local linear regression and vector ARIMA models are suitable for the big data context. The reason for this is their training efficiency and effectiveness for the bitcoin price prediction problem. Due to the nature of the local implementation design, our solution can be applied to even larger datasets. This would simply require additional partitions.

As future work, we consider the evaluation of larger datasets. This is especially relevant because we unfortunately encountered issues with the Databricks cloud service during our experiments. As a response, we had to resort to using a small subset of our original data to report final results. We also think that global ARIMA and vector ARIMA implementations would be a great next step for future works. Additionally, we consider the development of an appropriate method to combine our approach with data streaming and incremental learning. Using these techniques would allow us to avoid retraining the entire model whenever new information is available and therefore, we can achieve predictions in a more timely fashion.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," p. 9.

[2] E. Akyildirim, A. Goncu, and A. Sensoy, "Prediction of cryptocurrency returns using machine learning," Ann. Oper. Res., vol. 297, no. 1, pp. 3–36, Feb. 2021, doi: 10.1007/s10479-020-03575-y.

[3] J. H. F. Flores, P. M. Engel, and R. C. Pinto, "Autocorrelation and partial autocorrelation functions to improve neural networks models on univariate time series forecasting," in The 2012 International Joint Conference on Neural Networks (IJCNN), Jun. 2012, pp. 1–8. doi: 10.1109/IJCNN.2012.6252470.

[4] C.-H. Wu, C.-C. Lu, Y.-F. Ma, and R.-S. Lu, "A New Forecasting Framework for Bitcoin Price with LSTM," in 2018 IEEE International Conference on Data Mining Workshops (ICDMW), Nov. 2018, pp. 168–175. doi: 10.1109/ICDMW.2018.00032.

[5] I. Yenidoğan, A. Çayir, O. Kozan, T. Dağ, and Ç. Arslan, "Bitcoin Forecasting Using ARIMA and PROPHET," in 2018 3rd International Conference on Computer Science and Engineering (UBMK), Sep. 2018, pp. 621–624. doi: 10.1109/UBMK.2018.8566476.

[6] W. Waheeb, H. Shah, M. Jabreel, and D. Puig, "Bitcoin Price Forecasting: A Comparative Study Between Statistical and Machine Learning Methods," in 2020 2nd International Conference on Computer and Information Sciences (ICCIS), Oct. 2020, pp. 1–5. doi: 10.1109/ICCIS49240.2020.9257664.

[7] S. McNally, J. Roche, and S. Caton, "Predicting the Price of Bitcoin Using Machine Learning," in 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), Mar. 2018, pp. 339–343. doi: 10.1109/PDP2018.2018.00060.

[8] H. Kavitha, U. K. Sinha, and S. S. Jain, "Performance Evaluation of Machine Learning Algorithms for Bitcoin Price Prediction," in 2020 Fourth International Conference on Inventive Systems and Control (ICISC), Jan. 2020, pp. 110–114. doi: 10.1109/ICISC47916.2020.9171147.

[9] A. Adebiyi, A. Adewumi, and C. Ayo, "Stock price prediction using the ARIMA model," presented at the Proceedings - UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, UKSim 2014, Mar. 2014. doi: 10.1109/UKSim.2014.67.

[10] D. U. Sutiksno, A. S. Ahmar, N. Kurniasih, E. Susanto, and A. Leiwakabessy, "Forecasting Historical Data of Bitcoin using ARIMA and α-Sutte Indicator," J. Phys., p. 5.

[11] "NEURAL NETWORK MODEL VS. SARIMA MODEL IN FORECASTING KOREAN STOCK PRICE INDEX (KOSPI)," Issues Inf. Syst., 2007, doi: 10.48009/2_iis_2007_372-378.

[12] N. Merh, V. Saxena, and K. Pardasani, "A comparison between Hybrid Approaches of ANN and ARIMA for Indian Stock Trend Forecasting," Bus. Intell. J., vol. 3, pp. 23–44, Jul. 2010.

[13] S. C. Purbarani and W. Jatmiko, "Performance Comparison of Bitcoin Prediction in Big Data Environment," in 2018 International Workshop on Big Data and Information Security (IWBIS), May 2018, pp. 99–106. doi: 10.1109/IWBIS.2018.8471691.

[14] Y. Görür, BITCOIN PRICE DETECTION WITH PYSPARK USING RANDOM FOREST. 2018. doi: 10.13140/RG.2.2.26508.16008.

[15] P. Ciaian, M. Rajcaniova, and d'Artis Kancs, "The economics of BitCoin price formation," Appl. Econ., vol. 48, no. 19, pp. 1799–1815, Apr. 2016, doi: 10.1080/00036846.2015.1109038.

[16] Z. Chen, C. Li, and W. Sun, "Bitcoin price prediction using machine learning: An approach to sample dimension engineering," J. Comput. Appl. Math., vol. 365, p. 112395, Feb. 2020, doi: 10.1016/j.cam.2019.112395.

[17] W. Chen, H. Xu, L. Jia, and Y. Gao, "Machine learning model for Bitcoin exchange rate prediction using economic and technology determinants," Int. J. Forecast., vol. 37, no. 1, pp. 28–43, Jan. 2021, doi: 10.1016/j.ijforecast.2020.02.008.

[18] H. Jang and J. Lee, "An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information," IEEE Access, vol. 6, pp. 5427–5437, 2018, doi: 10.1109/ACCESS.2017.2779181.

[19] M. Saad and A. Mohaisen, "Towards characterizing blockchain-based cryptocurrencies for highly-accurate predictions," in IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Apr. 2018, pp. 704–709. doi: 10.1109/INFCOMW.2018.8406859.

[20] T. Zeng, M. Yang, and Y. Shen, "Fancy Bitcoin and conventional financial assets: Measuring market integration based on connectedness networks," Econ. Model., vol. 90, pp. 209–220, Aug. 2020, doi: 10.1016/j.econmod.2020.05.003.

[21] G. Giudici, A. Milne, and D. Vinogradov, "Cryptocurrencies: market analysis and perspectives," J. Ind. Bus. Econ., vol. 47, no. 1, pp. 1–18, Mar. 2020, doi: 10.1007/s40812-019-00138-6.

[22] M. Matta, I. Lunesu, and M. Marchesi, "Bitcoin Spread Prediction Using Social And Web Search Media," p. 10.

[23] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," Inf. Sci., vol. 191, pp. 192–213, May 2012, doi: 10.1016/j.ins.2011.12.028.