Algorithmic Trading Model Development for BTC/USDT Crypto Market

ABSTRACT

A critical area of machine learning is Time Series forecasting, as various forecasting problems contain a time component. A series of observations taken chronologically in time is known as a Time Series. In this research, however, we aim to compare three different machine learning models in making a time series forecast. We are going to use Bitcoin's price data-set as our time series data set and make predictions accordingly.The model is also evaluated through performance measures Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) and found that the proposed ARIMA model outperformed the benchmark model. The study underscores ARIMA's efficacy in Bitcoin price forecasting, outperforming both LSTM and XGBoost in terms of accuracy.

KEYWORDS: Time series forecasting, ARIMA, Bitcion's price prediction, LSTM, XGBoost

1. INTRODUCTION

A critical area of machine learning is Time Series forecasting problems involving a time component, time series forecasting is an important subfield of machine learning. It is one of the data science techniques used in supply chain management, business, and production the most. A time series is a collection of observations made sequentially over time. Because it includes a time component, a time-series dataset differs from other datasets. This extra component serves as a constraint as well as a framework providing a different information source.

Time series data takes into account large volumes of high-dimensional, continuously updated data. The dataset's dependencies are preserved by the constant updates in numerical order. As a result, there is a relationship between the past value and the future values, and so forth. When working with data, time series analysis takes intervals within particular time periods into account. A predetermined time interval produces patterns and behaviors in data series that aid in the creation of a stock market forecasting model.

However, the goal of this study is to evaluate three distinct machine learning models in order to create a time series forecast. Our time series data set will be the price dataset for Bitcoin, and we will base our predictions on that. We are going to use the data-set of the BTC/USDT market.

2. MODELS USED

2.1 Long Short-Term Memory (LSTM)

One subset of recurrent neural networks is called a long short-term memory (LSTM) network. They are well-liked for handling sequential data, including time-series data, because they can pick up long-term dependencies. The vanishing gradient problem is handled by LSTMs. A vanishing gradient problem results from the gradient getting too small or large when the time step is significant. This issue arises when the weights essentially stay the same while the optimizer back propagates and executes the algorithm.

An input gate, an output gate (a straightforward multilayer perceptron), and a forget gate are utilized

by the long-short-term memory cell. These gates specify, based on the priority of the data, whether or not it can pass through.The cell state and hidden state are used in gathering data for processing in the next state. The vanishing gradient can, therefore, be protected

- Input Gate: $it = \sigma(Wiht-1 + Wiht)$
- Forget Gate: $ft = \sigma(Wfht-1 + Wfht)$
- Output Gate: $ot = \sigma(Woht-1 + Woht)$
- Intermediate Cell State: $\tilde{C} = tanh(Wcht-1 + Wcht)$
- Cell State (next memory input): $ct = (it * \tilde{C}) + (ft * ct-1)$
- New State: $ht = ot * tanh(ct)$
- Xt Input Vector
- ht Output Vector
- W, U, and f Parameter matrices

2.2 Extreme Gradient Boosting(XGBoost)

XGBoost, or Extreme Gradient Boosting, is a well-known machine learning algorithm that has gained recognition for its remarkable abilities in supervised learning tasks, especially when dealing with structured data. XGBoost sequentially trains several weak models, such as decision trees, using a gradient boosting framework. Iteratively fixing errors results in the formation of a strong ensemble model. L1 and L2 regularization to prevent overfitting, tree pruning to improve model simplicity, and integrated handling of missing data are noteworthy features. It makes use of parallel computing to process different types of data with flexibility and minimal preprocessing, even for large datasets. Because of its speed, accuracy, and scalability, XGBoost's cross-validation support facilitates hyperparameter tuning, resulting in a flexible algorithm that is widely used in a variety of fields.

A weak classifier (WC) can be transformed into a much stronger classifier (SC) using gradient boosting. The desired model, in this case the stock market prediction, dictates how often this process should be carried out. The engineering objective of "boosting" terms is to maximize the performance of boosted tree algorithms from computational resources, even to the brink of their capabilities.

2.3 Autoregressive Integrated Moving Average (ARIMA)

The Auto-Regressive Integrated Moving Average model is used in fitting time series data, to help in getting a better understanding of the data or to make predictions.Using the regression equation, autoregression predicts the value for the subsequent time step based on the observations from earlier time steps.

The Integrated makes the series stationary by using raw observation differencing. The process of changing a non-stationary time series into a stationary one is called differencing. An observation value is subtracted from a prior observation value to complete the process. Until a stable series is achieved, the process is repeated.

The Moving Average divides the simple average for all time frames taken by the total number of time frames in which it is calculated. For a model to be considered pure Auto-Regressive, Xt must only depend on its lags.

$$Xt = \alpha + \beta 1Xt-1 + \beta 2Xt-2 + \cdots + \beta pXt-p + \epsilon 1$$

Where Xt is the lag1 of the series, β is the coefficient of lag1 that the model estimates and α is the intercept term. A pure Moving average is a model, however, is one where the Xt depends on the lagged forecast errors.

$$Xt = \alpha + \epsilon t + \theta 1\epsilon t-1 + \theta 2\epsilon t-2 + \cdots + \theta q\epsilon t-q$$

Where the error terms are the inaccuracies of the autoregressive models of the respective lags. εt and εt-1 are the errors from the equations:

$$Xt = \beta 1Xt-1 + \beta 2Xt-2 + \cdots + \beta 0X0 + \epsilon t$$

$$Xt-1 = \beta 1Xt-2 + \beta 2Xt-3 + \cdots + \beta 0X0 + \epsilon t-1$$

When we combine the Auto-Regressive and Moving Average model, we get the following equation:

$Xt = \alpha + \beta 1Xt-1 + \beta 2Xt-2 + \cdots + \beta pXt-p + \theta 1\epsilon t-1 + \theta 2\epsilon t-2 + \cdots + \theta q\, \epsilon t-\mathrm{q}$

ARIMA (p, d, q) is the standard notation used; the bracketed parameters are substituted with integer values; this shows the specific ARIMA model used.

The following are the definition of the settings:

p: refers to the number of lag observations included in the model, this helps to adjust the line that is being fitted to predict the series

d: refers to the quantity of differencing transformations needed by the time series to get stationary.

q: refers to the size of the moving average window.

3. Setup and Experiments

3.1 Data collection

For this time series forecasting, we use data from Bitcoin transactions . The daily trading exchange data rate in dollars (USD) from 1st January 2018 to 31st January 2022 was used for this research. The data has 5959 rows, this was split into a training set, test set and validation with a ratio of **80:10:10**. This split gives a training set of 4767 rows, 595 rows for the test set and 595 rows for the Validation. The data has the following features; Datetime, Open, High, Low, Close, Volume. The features are explained below:

Features	**Summary**
Datetime	data recorded date and time
Open	Value at which trading opened
High	The day's highest trade value
Low	The day's lowest trade value
Close	Latest trade
Volume	The day's total trade volume

3.2 Dataset Normalization

To make it easy for the network to learn, the data should take small values; mostly between the range of 0 and 1. It should also be homogeneous, meaning all the features should have values at approximately the same range.We, therefore, reduced the data to a scale between 0 and 1. The process of putting the values in the same range is called the MinMax normalization. We import MinMaxScaler from Scikit-learn to do this. The equation for the Min-Max normalization is as follows:

$$z = x - \min(x) / \max(x) - \min(x)$$

Where z represents the normalized value and x represents the observed values in the set. **Min** and **max** are the minima and maxima values in x.

3.3 Training, Testing and Validation

Stock data used in this research work are historical daily stock prices obtained from the dataset of 6-hours bitcoins price prediction . The data is composed of four elements, namely: open price, low price, high price and close price respectively. In this model the **closing price** is chosen to represent the price of the index to be predicted. Closing price is chosen because it reflects all the activities of the index in a trading day.
The obvious strategy of XGBoost in our experiment is as follows: the learning model of XGBoost is built upon the training data, while we tune out every hyperparameters using the validation data, and test data is fed for the final result.

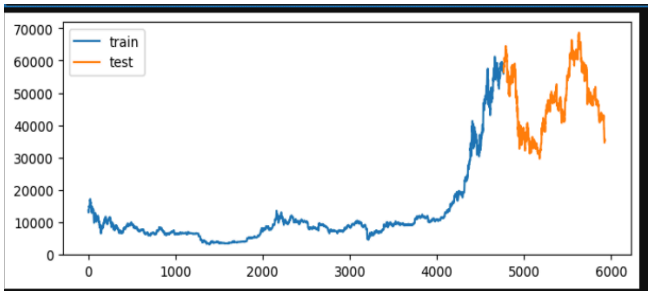

Figure 3.1 XGBoost Dataset division based on closing values

We will use the same strategy as XGBoost for LSTM and tune every parameter using the validation data and test data is fed to the final result.

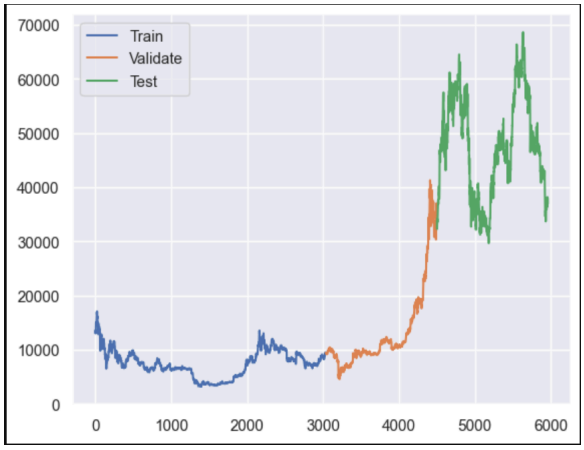

Figure 3.2 LSTM Dataset division based on closing values

3.4 Fitting the models

3.4.1 The ARIMA Model. We start by fitting an ARIMA model to the Bitcoin Closing price dataset and review the residual errors. We fit an ARIMA (5 1, 0) model. This sets the lag value to 5 for autoregression. We already made our dataset stationary, w, however, chose to use a Difference order of 1. We also wanted a moving average model of 0.
After the model has been fit to the training dataset, we proceed to make predictions and compare the results to the test set. We calculate the residual error for each prediction made and even find the final Mean Absolute Percentage Error score (MAPE) and the root-mean-square error score (RMSE).

Here are some of the forecasting values of the ARIMA model:

Predicted=50218.486700, Expected=49902.940000
Predicted=49895.272380, Expected=49954.830000
Predicted=49967.925273, Expected=49915.640000
Predicted=49908.857240, Expected=49688.800000
Predicted=49712.322045, Expected=50198.780000
Predicted=50192.276097, Expected=50299.690000
Predicted=50313.714785, Expected=51756.880000
Predicted=51753.309211, Expected=51759.990000
Predicted=51748.482311, Expected=51400.670000
Predicted=51361.442139, Expected=51672.980000
Predicted=51562.700552, Expected=52663.900000
Predicted=52653.924333, Expected=52665.010000
Predicted=52699.426042, Expected=50941.480000
Predicted=50918.689516, Expected=46606.440000

3.4.2 The LSTM Model. Our LSTM model in Keras has the following parameter:

Parameters	**Values**
Epochs	100
Batch size	128
Dense	1
Input shape	1
Optimizer	Nadam

4. Experimental Results

We used two of the most common metrics to measure the accuracy of the models, the Mean Absolute Percentage Error (MAPE) and the Root Mean Squared Error (RMSE).

The results show that the ARIMA model gave better results than the other machine learning-based regression models. **ARIMA** gives the best results at **1.905% and 847.875** for MAPE and RMSE respectively, Whereas LSTM MAPE and RMSE Values are **2.029%** and **1238.549** respectively.The MAPE and RMSE values of XGBoost are **34.502%** and **5127.097** respectively.

$$RMSE = \text{sqrt}\ [(\Sigma(Pi - Oi)^2) / n]$$

$$MAPE = (1/n) * \Sigma(|actual - forecast| / |actual|) * 100$$

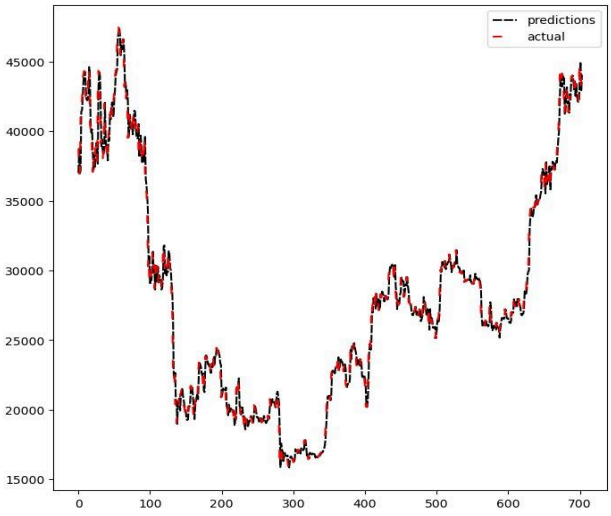

Figure 4.1 Graph of actual and predicted values of ARIMA model

Blue-Predicted values
Orange-Actual value

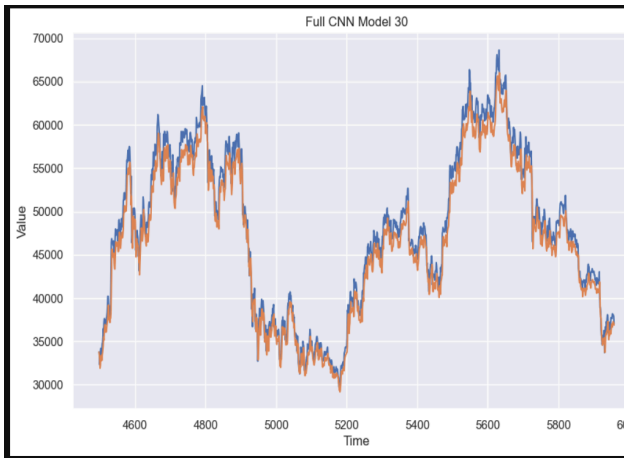

Figure 4.2 Graph of actual and predicted values of LSTM model

Figure 4.3 Graph of actual and predicted values of XGBoost model

4.2 Trading results

The trading strategy involves utilizing a predefined threshold value for decision-making when predicting stock price movements. If the forecasted price surpasses the threshold in a positive direction, signifying an anticipated increase, the strategy will execute a buy action. Conversely, if the prediction doesn't meet the threshold or goes below it, suggesting a potential decrease, the strategy will prompt a sell action, allowing for a profit or loss based on the current stock price. This threshold-based approach aims to capitalize on predicted upward or downward trends to optimize buying or selling decisions in the stock market.Here we have taken the **initial amount of $100,000.**

Net Earning : $163112.093

Strategy return : $63112.093

Max Trade Duration : 72 days

Sharpe Ratio : 1.804

Maximum Drawdown : -66.731 %

Total benchmark return : 113006.219

Buy and hold return : 13006.219

Net Profit Exceeding Benchmark Return of the Model : 44.339 %

Risk to reward ratio : 1.913

Average winning trade : 32839.658

Average losing trade : -17163.668

Largest winning trade : 64146.76286514019

Largest losing trade : -28959.77941006978

Total closed trades : 131

Average holding duration : 3

Sortino ratio : 3.407

5. Conclusion / Insight gained

From the results, it shows that the ARIMA Model gives the **best accuracy and time.** This outcome might also be as a result of several factors. The parameters that are chosen and the total amount of data can also affect the results.

Though, XGBoost is highly efficient, flexible and portable algorithm but ARIMA obtains the superior results over LSTM and XGBoost

Due to the working of the ARIMA model, it was unable to capture the sharp instabilities in the Bitcoin price of the future predictions.With the results obtained ARIMA models can compete reasonably well with emerging forecasting techniques in short-term prediction.

Github Repository

AvaterClasher/crypto: Crypto ARIMA model (github.com)