Predicting Bitcoin Price using Machine Learning Algorithms

Yunus Emre CICEN

DEPEND Erasmus Mundus Joint MSc in Advanced Systems Dependability

Department of Computer Science,

Maynooth University,

Co. Kildare, Ireland.

A dissertation submitted in partial fulfillment

of the requirements for the

Erasmus Mundus MSc Dependable Software Systems

Head of Department: Dr. Joseph Timoney

Supervisor : Dr. Phil Maguire

Date: 22 July 2021

Word Count: 6710

ABSTRACT

In this project, we aim to use Long Short Term Memory (LSTM) model to predict Bitcoin values in the future. We provide price, trend, and rate estimations using 4h data and 1h data. We also exercise the implementation of a system where it gathers price data from various sources, prepares datasets for predictive machine learning models, and share the outputs through a web application and REST API. Our approach yields an accuracy rate of 46.38% and 43.22%.

1. INTRODUCTION

Bitcoin is decentralized digital money that can be sent directly from one user to another using a peer-to-peer bitcoin network. [1] Network nodes utilize cryptography to validate transactions, which are then stored on a blockchain, which is a public distributed ledger.

Predicting the cryptocurrency values are different than traditional currency and stock predictions as there are various factors involved with these digital assets. As Bitcoin is leading the cryptocurrency market, there are several alternative crypto coins that gained much attention from investors. There are over 10.000 different cryptocurrencies listed as of May 2021, with the number growing constantly. [2]

As CoinMarketCap reports, the total market cap for all cryptocurrencies is around $1.5tn, and over 60% comes from Bitcoin and Ethereum solely. This gives an appetite to many interested parties to predict future prices and create financial strategies.

1.1 The Topic Addressed in This Project

The project focused on creating a predictive model for highly volatile cryptocurrencies. We are using Bitcoin price data for the prediction model. However, the same methodology can be applied to different altcoins. In this project, we want to leverage from Long Short Term Memory (LSTM) model [3] to predict bitcoin values in the future. We also examine the possible outcomes of acquiring price data from different mediators where this can be leveraged to develop arbitrage-based model predictions. We give data scraping strategies from several sources, develop an LSTM-based machine learning model, and evaluate the results of the prediction models.

1.2 Motivation

Bitcoin is decentralized and has no authorities controlling it. This makes it vulnerable to manipulative attacks and many unforeseen challenges. With its availability to buy and sell through different exchange sites at any given time, predicting the price is really challenging. Trying to solve this problem, would give us many learnings in time-series analysis, prediction and finance altogether. Testing machine learning's abilities on such challenging topics can be very rewarding as well, both in knowledge and reputation. Lastly, if the aimed results are good, with good trade strategies there can be profitable options to make.

1.3 Problem Statement

Bitcoin fluctuates so often and so randomly. The ability to buy and sell very easily at any given time puts any predictive algorithm under pressure. The decentralized way of bitcoin means there is no controlling mechanism for manipulative attacks from big whales. Bitcoin has a limited supply, there can be a maximum of 21 million of them that can be mined in total. Approximately 18.5 million bitcoins have been mined to date. [4] There are only about three million left to be released into circulation.

Bitcoin also has a high volatility rate and aggressive price changes. From 5 cents in 2008 to 60 thousand dollars in 2020, we can see how flexible prices are. Figure 1 represents the price change and volatility of Bitcoin since 2010. [5]

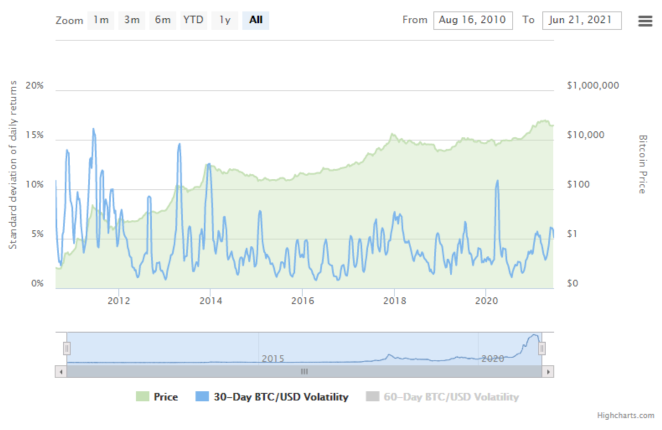


Figure 1: Price Change and Volatility of Bitcoin 2010-2021

1.4 Approach

We aimed to solve this prediction problem with the use of machine learning. We created a Long Short Term Memory (LSTM) model to predict prices and another LSTM model to predict trends and changes in the rate on the prices. We collected price data from various sources and fed this to our model to get predictions. We have used the past 4 hours of data to predict the next 2 minutes and the past 1 hour of data to predict the 30 seconds.

1.5 Metrics Evaluations

To evaluate the correctness of our algorithm, we used Mean Squared Error [6] as a loss function to optimize our predictive model. Furthermore, we used Adam optimizer [7] to optimize the hyper-parameters of our model. When the predictions from the model were generated, we compared these results with the actual price data and the trend change. We calculated the accuracy of our estimations by calculating the difference between our prediction and the real-time data.

2. RELATED WORK

Prediction of the value and trend of financial assets is an interesting subject with a long history. The rise of digital cryptocurrencies, especially after the 2008 economic crisis, has gained more attraction from researchers. There are many studies done to predict the outcome of this financial trend.

2.1 Research Material

- Awoke *et al*, compared LSTM and GRU time-series deep learning models to check their efficacy in forecasting the price of bitcoin. [8]
- Yang & Li *et al* tried to predict price fluctuation with the help of basic features and traditional technical trading indicators, and features generated by a denoising autoencoder. They evaluate these features using an Attentive LSTM network. [9]
- Velankar *et al* attempted to predict the Bitcoin price accurately taking into consideration various parameters that affect the Bitcoin value. [10]
- Begum *et al* tried to predict bitcoin prices with the help of data manipulation in Apache Spark and Python. [11]

- Phaladisailoed *et al* aimed to find a less time-consuming and accurate model for the prediction of Bitcoin prices using various machine learning models like (Multivariate Linear Regression, Theil-Sen Regression, Huber Regression) and deep learning algorithms like (LSTM, GRU). [12]
- Alahmari, applied the Linear, Polynomial, and Radial Basis Function (RBF) kernels to predict the prices of the three major cryptocurrencies. [13]
- Bai et al created an algorithm called C2P2 Collective Cryptocurrency Price Prediction by taking into account prices and predicted prices of other cryptocurrencies. [14]

2.2 Technical Material

We developed our machine learning model using Python [15] language and Pytorch [16] library with the help of various others. PyTorch is an open-source machine learning library based on the Torch library [17], used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR). It is free and open-source software released under the Modified BSD license.
PyTorch is a Python package with two high-level capabilities: A tape-based autograd system to build deep neural networks and Tensor computation (similar to NumPy) with strong GPU acceleration. Tensors are data vectors of Pytorch with rich features. Tensors that can be run on either the CPU or the GPU, greatly speeding up the computation. Hence Pytorch mainly focuses on Python, we can use top-notch python packages like Pandas, Numpy, SKLearn when needed, to enhance our machine learning abilities.

PyTorch provides Tensors that can live either on the CPU or the GPU and accelerates the computation by a huge amount. PyTorch uses a technique called reverse-mode auto-differentiation, which allows you to change the way the network behaves arbitrarily with zero lag or overhead. PyTorch is designed to be intuitive, linear in thought, and easy to use.

PyTorch has minimal framework configuration and has been tested for years to run smoothly on various devices with either small or large neural networks. The memory usage in PyTorch is extremely efficient compared to alternatives. Hence we selected Pytorch as the main library to build our models.

Machine Learning models rely on good optimization techniques to minimize loss of

correctness and reduce the need for iterations to select hyper-parameters. In this project, we used the famous Adam optimizer.
The Adam optimization algorithm is a derivation of a stochastic gradient descent algorithm that has gained popularity for deep learning applications in recent years in the fields of computer vision and natural language processing. To update network weights iteratively based on training data, Adam can be used instead of the traditional stochastic gradient descent approach.

Adam was presented by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 ICLR paper titled “Adam: A Method for Stochastic Optimization“. They described it as a method for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The approach is simple to develop, computationally efficient, requires minimal memory usage, is insensitive to gradient diagonal rescaling, and is ideally suited for situations with huge amounts of data and/or parameters

Adam can also be used to solve problems with non-stationary targets and/or very noisy and/or sparse gradients. The hyper-parameters have straightforward interpretations and require little adjustment in most cases. Therefore it is suitable to use in time-series data such as weather forecasting or price analysis.

We gathered data from different price providers in order to check our model’s accuracy in different data subsets. The sources we used are:

Nomics [18] is an API-first crypto asset data company delivering professional-class market data APIs to institutional crypto investors & exchanges. They offer products and services that allow funds, fintech apps, and exchanges to access clean, and gapless primary sources of trade data.

CoinMarketCap[19] Founded by Brandon Chez in May 2013, is the most widely used price-tracking website for crypto assets. Its goal is to make cryptocurrency more accessible and efficient around the world by providing unbiased, high-quality, and reliable data to retail customers so they can make their own informed decisions. Binance Capital purchased CoinMarketCap in April 2020. Binance is a worldwide blockchain startup that operates the world's largest digital asset exchange in terms of trading volume and users, with the goal of making cryptocurrency more accessible and systemically significant to people all over the world.

Messari [20] founded by Ryan Selkis in 2018, is an open-source library of information that delivers data insights, pricing, and research on crypto-assets through an online database for the crypto business. It aims to be the crypto industry's equivalent of Crunchbase and the US Security and Exchange Commission's EDGAR database, which contains publicly available company registration statements and reports, among other documents.

CoinDesk [21] founded in May 2013, is a media platform for the future generation of investors interested in learning more about how cryptocurrencies and digital assets are influencing the global financial system evolution. Its mission is to use news, data, events, and education to inform, educate, and connect the worldwide investment community.

BitMEX [22] is the world's most advanced peer-to-peer crypto-products trading platform, powered by best-in-class API, and giving knowledge, confidence, and precision to hundreds of thousands of traders, transacting the equivalent of billions of USD every day. Their leveraged contracts make trading and hedging bitcoin risk quick, easy, and liquid. Their XBTUSD perpetual swap, which is the most traded bitcoin product of all time, revolutionized the market.

2.3 The Technologies Used in This Project

While implementing our project, we have used several tools to develop our model, create a back-end service to serve our findings, and provide a user interface through a web application. Tools used in this project demonstrated as in the following table:

Tech	**What is For**
Front-end	
Next.js [23]	To accelerate the development of React applications with built-in routers and data functions
React.js [24]	To create fast single-page applications with built-in functions

Tailwind CSS [25]	To accelerate the development of front-end styles and providing visual effects in a responsive way
ReCharts [26]	To create a graph-based interface for displaying the output of the predictive models
Vercel [27]	To deploy, ship, and host the web application
Backend	
Node.js [28]	To create a web server for REST API endpoints
Express.js [29]	To create routers for web server endpoints
Cors npm module [30]	To enable Cross-Origin Resource Sharing for REST API
Dotenv npm module [31]	To hide credentials such as API keys and configuration management
Mongoose npm module [32]	To connect MongoDB Database
UUID npm module [33]	To create unique identifiers for database records
MongoDB [34]	To store our database in a cloud cluster
Heroku [35]	To deploy our backend service to make it available online
Machine Learning	
Python	To implement our proposed solutions
Pytorch	To accelerate the development of our machine learning approach
NumPy [36]	To read and write the data using NumPy arrays
Pandas [37]	To read and format the CSV files
Scikit-learn [38]	To provide assistance to our ML and calculate scores
Requests [39]	To make a POST request with our findings

3. THE PROBLEM

Bitcoin is a peer-to-peer electronic cash system that allows payments to be transmitted directly from one party to another via the internet without having to go through a banking institution. Digital signatures are a part of the solution, but the major benefits are lost if a trusted third party is still needed to avoid double-spending. Transactions in the Bitcoin system are hashed into a continuing chain of hash-based proof-of-work by the network, establishing a record that cannot be modified without redoing the proof-of-work.

The bitcoin system is made up of a network of computers (also known as "nodes" or "miners") that execute bitcoin's code and store its blockchain. A blockchain can be viewed as a collection of blocks in metaphorical terms. A set of transactions can be found in each block. No one can trick the system because all computers running the blockchain have the same list of blocks and transactions and can watch new blocks being filled with new bitcoin transactions transparently.

These transactions may be seen in real-time by anyone, whether or not they run a bitcoin node. The Bitcoin system is highly secure due to creating an attack on this digital asset requires massive computing power. To carry out a criminal deed, a bad actor would need to control 51% of bitcoin's computational power. As of June 2021, Bitcoin has over 10,000 nodes, and this number is growing, making such an attack highly implausible. [40]

Additionally, regulatory developments had a significant impact on Bitcoin's price as it gained public notice in 2017. Institutional investors' interest has also thrown a long shadow over Bitcoin's price dynamics. In the last 10 years, Bitcoin has shifted its focus away from regular investors and toward institutional investors. This is viewed as a positive development because it adds liquidity to the ecosystem while reducing volatility.

Bitcoin's price is also influenced by industry advancements. Because of Bitcoin's unique origins, which span both technology and finance, these advancements are relevant to both industries. Price hikes have also been accelerated by Bitcoin halving events, in which the total supply of Bitcoin available in the market decreases due to a fall in miner rewards due to an algorithmic adjustment.

In addition, economic instability is a predictor of Bitcoin price swings. The cryptocurrency has established itself as a supranational hedge against local economic volatility and government-controlled fiat currency since its creation. For example, Venezuelans are using the cryptocurrency revolution to combat hyperinflation in the country [41]. Another example is El Salvador became the first country in the world to make bitcoin an official tender [42].

3.1 The Difference Between Stock Prediction Approaches and Bitcoin Predictions

Stocks are backed by real businesses that are expected to make money. They include tangible assets in their valuation, and math may be used to establish whether a stock is valued correctly based on market pricing. [43]

Cryptocurrencies, on the other hand, aren't always backed by companies. They are generally valued depending on how popular they are, while some are also valued based on how useful they are. It's not always straightforward to forecast whether a currency is worth it because it's a more subjective estimate.

Since anyone can make a blockchain token, it's easy enough to start your own ICO. The same cannot be said for stocks
When stocks are created, they must be cleared and audited by government bodies. Before they can get on the market, they must also follow specific laws.

Moreover, stocks are strictly regulated, and most must undergo annual audits in order to continue trading on the stock exchange. Not only do true ICOs and cryptocurrencies have the potential for exit scams, but cryptocurrency exchange scandals mean that you might easily lose your entire position in a short period of time.

These factors affect the trust and price volatility of digital currencies, hence using the same prediction patterns in classical financial algorithms can fail to provide an estimation.

4. THE SOLUTION

4.1 Analytical Work

The data obtained from various sources written to a CSV file. The price, market volume, and the timestamp of the price collected from each source. The data collected every 30 seconds. Then this data is read by another script for the data preparation phase. We converted float timestamp values to int timestamp values in order to get a truncate milliseconds difference. We also converted price and volume factors into 6 digit floating points, which were easier to compute. For price prediction, we sliced our data for different time intervals. In this model, we used the last 4 hours of data in a comparison with the last hour of data. For predicting the trend and the rate of change in price, we created another script. Using the following formula, we computed the rate of change and trend.

$\forall i \in \{1,\ldots,n\}\ f(P(i)) = (P(i+1) - P(i)) / P(i+1)$

$\forall i \in \{1,\ldots,n\}\ f(T(i) = P(i) > 0 \Rightarrow 1$ else 0.

We subtracted each price from the next price and divided it into the value of the next price, where it gave us the rate of change. If the next price is higher than the current price, that would mean the trend is going up, where we demonstrated as 1, if the rate is negative, we denoted the trend as down, 0.

After the data is prepared for the model, we created a data loader function to create our dataset. We split the data into train and test datasets. We split our data as 80% of train data and 20% of test data. The data then piped into our machine learning model. We are using the Long Short-Term Memory model, a branch of Recurrent Neural Network (RNN). [44] Before we dive into the next section, we would like to give more understanding of these models.

Recurrent Neural Network (RNN)

An RNN is a type of artificial neural network which uses sequential data or time-series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing, speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate.

An RNN is a sort of artificial neural network that works with time series or sequential data. These deep learning algorithms are often employed for ordinal or temporal problems like language translation, speech recognition, natural language processing, forecasting problems, and image captioning. They are the backbone of some popular apps like Siri voice search and Google Translate [45].

Recurrent neural networks use training data to learn, just like feedforward and convolutional neural networks (CNNs) [46]. They are distinguished by their "memory," which allows them to alter current input and output by using data from previous inputs. Recurrent neural networks' output is dependent on the prior elements in the sequence, whereas classic deep neural networks presume that inputs and outputs are independent of one another. While future events may be useful in predicting the outcome of a series, unidirectional recurrent neural networks cannot account for them in their predictions.

Recurrent Neural Network remembers the past and its decisions are influenced by what it has learned from the past. While RNNs learn similarly while training, in addition, they remember things learned from prior inputs while generating outputs. It’s part of the network. RNNs can take one or more input vectors and produce one or more output vectors and the outputs are influenced not just by weights applied on inputs like a regular NN, but also by a “hidden” state vector representing the context based on prior input(s)/output(s). So, the same input could produce a different output depending on previous inputs in the series.

The previous states are remembered by the recurrent neural network, which influences its decisions. While RNNs learn in a similar way when they are trained, they also remember what they have learned from previous inputs when they generate outputs. RNNs can take one or more input vectors and produce one or more output vectors. The outputs are impacted not just by weights applied to the inputs, as in a conventional NN, but also by a "hidden" state vector indicating the context based on previous inputs/outputs. As a result, depending on prior inputs in the series, the same input may produce a different output. Figure 2 shows a simple RNN model. [47]

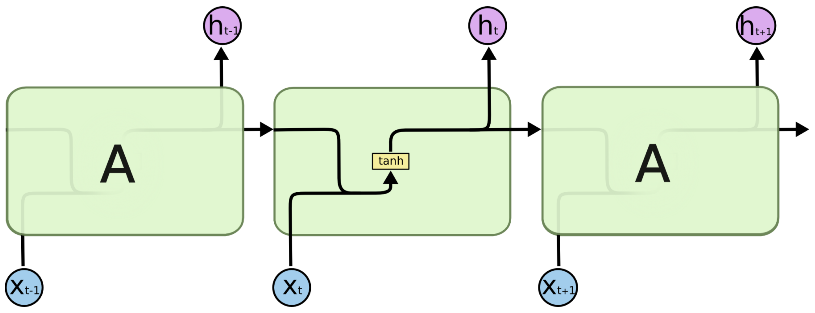


Figure 2: Simple RNN model

Long Short-Term Memory (LSTM)

LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. LSTM has feedback connections, unlike normal feedforward neural networks. It can analyze not only single data points (such as images) but also complete data sequences like audio or video (real-time price data in our case).LSTM can also be used for tough tasks including unsegmented, connected handwriting recognition, speech recognition, and network traffic anomaly detection.

LSTM networks are well-suited to categorizing, processing, and making predictions based on time series data since there can be lags of unknown duration between important events in a time series. LSTMs were created to solve the problem of vanishing gradients that can occur when training traditional RNNs. In many cases, LSTM outperforms RNNs, hidden Markov models, and other sequence learning algorithms due to its relative insensitivity to gap length.

LSTMs aid in the preservation of error that can be propagated backward in time and layers. They allow recurrent nets to learn across multiple time steps by maintaining a more constant error. Because algorithms are usually confronted by situations where reward signals are sparse and delayed, this is one of the major difficulties to machine learning and AI.

A simple LSTM unit consists of a cell, an input gate, an output gate and a forget gate. The three gates control the flow of information in and out of the cell, and the cell remembers values across arbitrary time intervals.

LSTMs keep the data outside the normal flow of the recurrent network in a gated cell. The information stored in the cell can be written or read, much like data in a computer's memory. Through open and close gates, the cell decides what to store and when to allow readings, writes, and erasures. These gates are analog, meaning they developed with element-wise multiplication by sigmoids, which are in the range of 0 to 1. This gives the advantage of being differentiable, which makes it suitable for backpropagation.
The figure 3 represents a typical LSTM model. [47]

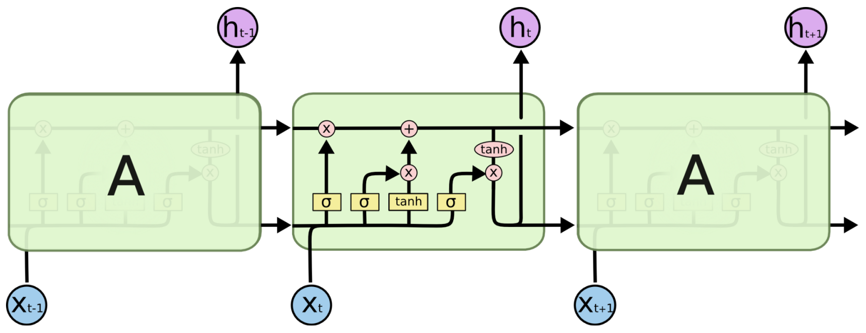


Figure 3: Simple LSTM model

In our project, we used the following hyper-parameters: *input_dim = 2, hidden_dim = 50, num_layers = 2, output_dim = 2.* We also iterated our training loop in 200 epochs. Where our input dimension is price and volume for price prediction; trend and rate of change for the trend prediction. We initially have 50 hidden dimensions to create our network within two layers. And we have the output vector of our estimations of price, volume; trend, and rate.

4.2 Architectural Level Details

Our architecture consists of different layers and steps. Mainly, we have three modules, the machine learning model, the back-end module with REST API provider, and the front-end module to visualize our findings.

The machine learning module is responsible for creating the predictions, the backend module writes these prediction outcomes to our MongoDB database and serves from the REST endpoint, and lastly, our front-end module fetches the prediction outcomes from the server and demonstrates them on our web application. Figure 4 represents the overall approach and the architecture of our proposed solution.

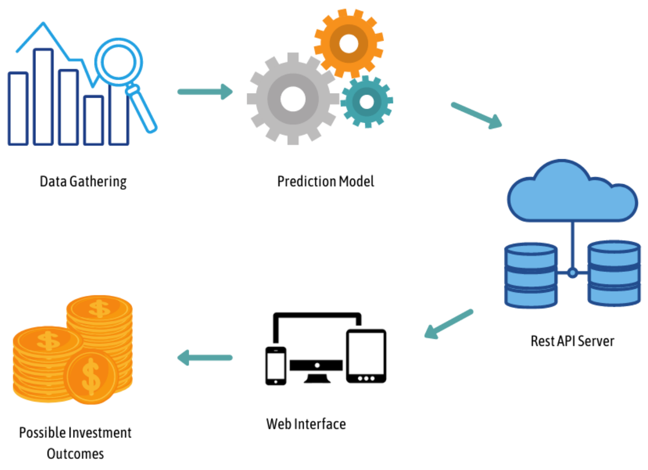


Figure 4: Overall Architecture

4.3 High-Level Details

Initially, we gather data from various sources mentioned earlier. The data collected every 30 seconds and written into a CSV file regarding the source. We collected timestamp, price, and the volume of the Bitcoin data. Then the data preparation module takes these CSV files as input and creates a stream-line for the model. Here we slice the data for our given time interval (4hours, 2 hours) and convert the timestamps to dates. We also calculate the trend and rate of change in the data preparation step to assure our model has the task of predictions only. Separation of concerns rule applied by this division. Our Data Provider module writes the prepared data into another CSV file where it can be fed directly to our model. The LSTM class reads this CSV file. Firstly, we created our train and test datasets. In the next step, we converted the datasets to Tensors and loaded these datasets to our model. The model evaluates the data and creates a prediction while trying to minimize the loss of accuracy. The evaluator module takes the accuracy results, calculates test and train scores, and compares the prediction with the actual data. This process then streamed to our Publisher module, where the results are written to a CSV file and sent to our database by a POST request. Figure 5 shows our class architecture for the first layer, the machine learning module.

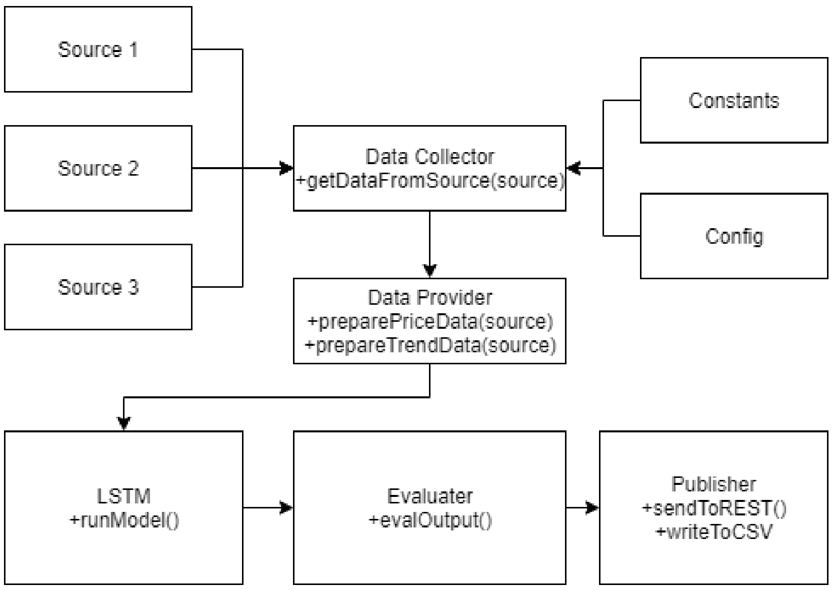


Figure 5: Class Architecture for ML module

Our back-end module receives POST requests from the publisher and GET requests from our frontend module and other interested parties. The router of Express.js converts the request directly through the requested path. The Controller class takes care of the response of the request. If it's a POST request, it creates a view of the related database model and writes this data to the database. If the request is a GET request, then the Controller class retrieves the data from the database, fills the related empty View object, and serves the requester in JSON format. Figure 6 illustrates our back-end approach.

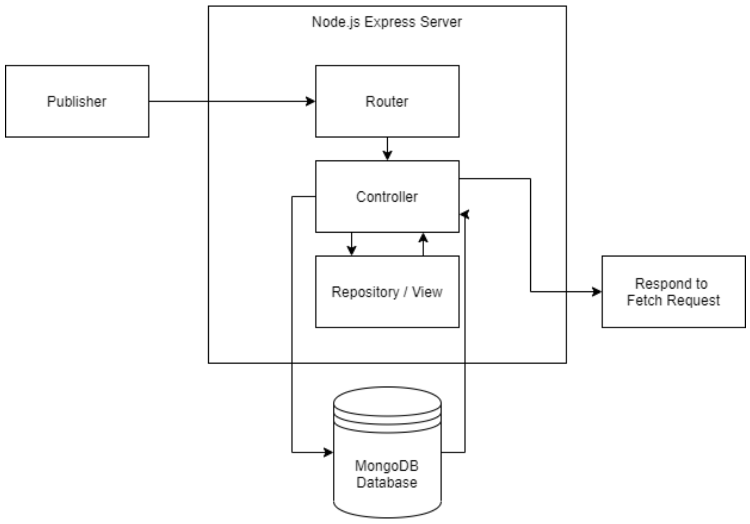


Figure 6: Overall Architecture for Back-end module

Our front-end module leverages the cutting-edge tools that are used in front-end development. We created a Single-Page Application (SPA) [48] to present our findings from the predictive model using Next.js, React.js, and Tailwind CSS. Instead of a web browser fetching complete new pages, a single-page application (SPA) interacts with the user by dynamically rewriting the current web page with fresh data from the web server. The idea is for the website to feel more like a native app thanks to speedier transitions.

A single-page application (SPA) is a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of a web browser loading entire new pages. The goal is faster transitions that make the website feel more like a native app. A page refresh is never required in a SPA; instead, the browser retrieves all necessary HTML, JavaScript, and CSS code with a single page load, or the proper resources are dynamically loaded and appended to the page as needed, usually in reaction to user activities. We preferred this approach as our prediction is continuous and we would like to provide the latest predictions to our users without any delay. We used ReCharts to provide graph functionality and we feed this component with the data from our backend server. Figure 7 illustrates our frontend project structure.

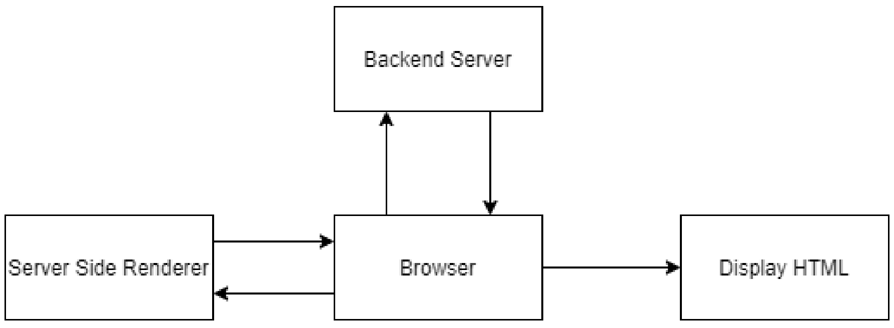
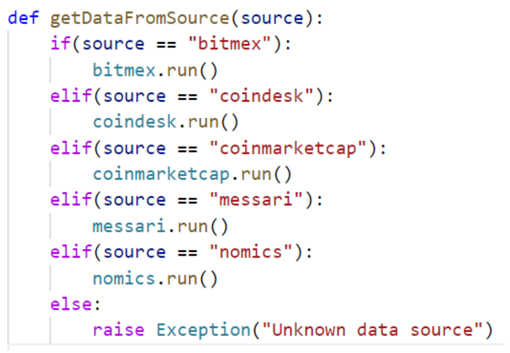


Figure 7: Overall Architecture for Front-end module

4.4 Low-Level Details

We implemented our proposed solution using various languages such as Python and JavaScript. With the help of libraries and tools mentioned in the "Technologies Used" section, we delivered our project as a product that can be accessed worldwide. We would like to bring out the following aspects of our implementation.

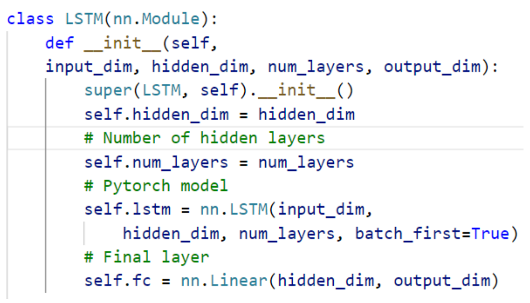
By creating a function named getDataFromSource, we were able to access different data providers. This gave us an edge on changing the model data when one is inaccessible. By using various sources, our prediction became source-agnostic. The code can be represented as in figure 8.



```
def getDataFromSource(source):
    if(source == "bitmex"):
        bitmex.run()
    elif(source == "coindesk"):
        coindesk.run()
    elif(source == "coinmarketcap"):
        coinmarketcap.run()
    elif(source == "messari"):
        messari.run()
    elif(source == "nomics"):
        nomics.run()
    else:
        raise Exception("Unknown data source")
```

Figure 8: Data Provider method

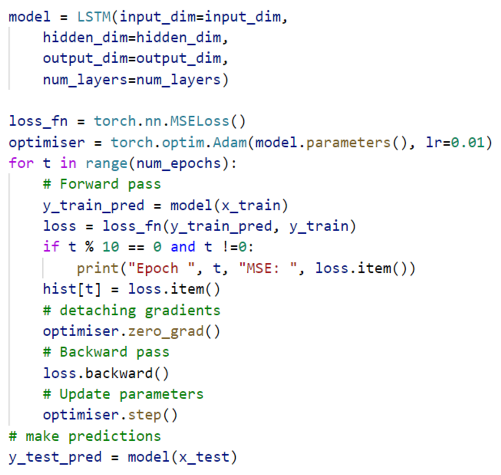
To use Pytorch modules, one must provide their own class to modify the required machine learning approach. Hence, we created a class named LSTM where it derives from Pytorch's Neural Network (NN) module. We initialize our hyperparameters by defining a __init__ function and we simply create our model with the help of Pytorch. As the Final output layer, we chose the Linear Regression technique to provide the output of our LSTM model. The code represents our implementation in figure 9.



```
class LSTM(nn.Module):
    def __init__(self,
    input_dim, hidden_dim, num_layers, output_dim):
        super(LSTM, self).__init__()
        self.hidden_dim = hidden_dim
        # Number of hidden layers
        self.num_layers = num_layers
        # Pytorch model
        self.lstm = nn.LSTM(input_dim,
            hidden_dim, num_layers, batch_first=True)
        # Final layer
        self.fc = nn.Linear(hidden_dim, output_dim)
```

Figure 9: LSTM Class

Later, we initialized our model object with the given hyperparameters; we initialized our loss function and optimizer for the model. Next, we created our training loop to teach our model. We do a forward pass on our data and calculate the loss. After we emptied the gradients from our model we did a backpropagation on our loss function. The optimizer evaluated this process and updated the weight of the cells. Thus giving us the predictive model to work what is needed. Finally, we ran our model and stored the prediction outputs into an array. The code is demonstrated in figure 10.



```
model = LSTM(input_dim=input_dim,
    hidden_dim=hidden_dim,
    output_dim=output_dim,
    num_layers=num_layers)

loss_fn = torch.nn.MSELoss()
optimiser = torch.optim.Adam(model.parameters(), lr=0.01)
for t in range(num_epochs):
    # Forward pass
    y_train_pred = model(x_train)
    loss = loss_fn(y_train_pred, y_train)
    if t % 10 == 0 and t !=0:
        print("Epoch ", t, "MSE: ", loss.item())
    hist[t] = loss.item()
    # detaching gradients
    optimiser.zero_grad()
    # Backward pass
    loss.backward()
    # Update parameters
    optimiser.step()
# make predictions
y_test_pred = model(x_test)
```

Figure 10: Model Training Loop

5. EVALUATION

We used mean squared error (MSE) to evaluate our model's efficiency. The mean squared error (MSE) measures the distance between a regression line and a set of points. It accomplishes this by squaring the distances between the points and the regression line which they have also known as "errors". Squaring is required to receive a positive outcome. It also gives significant discrepancies more weight. Because we are calculating the average of a series of errors, it's named the mean squared error. Lower MSE yields better predictions. The formula as follows: [49]

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

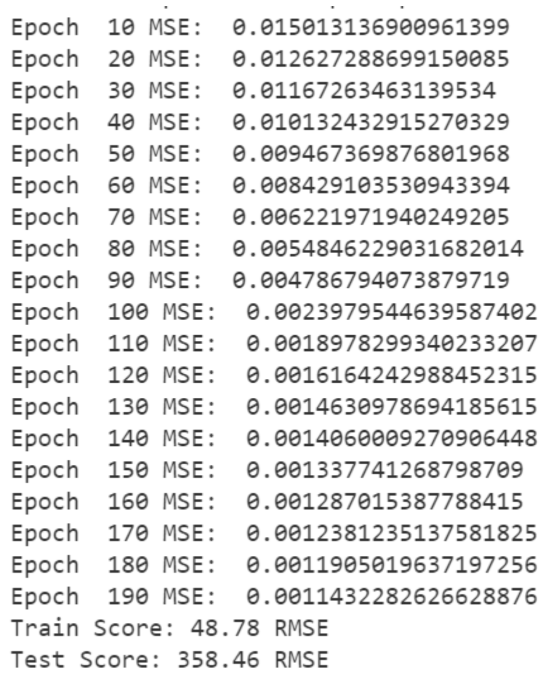
In our case, y values are the prediction values from our model. where they are subtracted from the values of the train data prices. We took the square of this distance and then we calculate the average of these "errors". Our algorithm uses this function to minimize the loss of accuracy.

We also leverage the Root Mean Squared Error (RMSE) when calculating test and train scores of our LSTM model.

Root Mean Square Error (RMSE) is the standard deviation of our prediction errors. In other words, square root of our MSE. These errors calculated above are a measure of how far from the regression line data points are; RMSE is a measure of how to spread out these residuals are. It gives us an idea about the data concentration on our predictive model. Root mean square error is commonly used in forecasting, and regression analysis to verify experimental results. The formula of RMSE as follows [50]

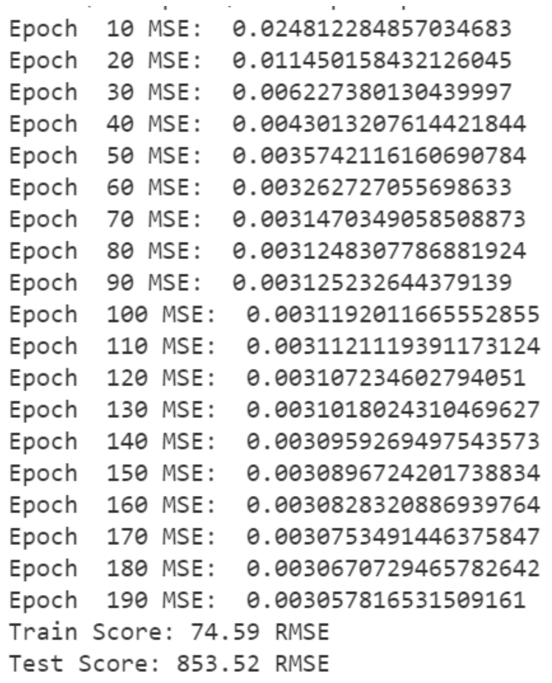
$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y_i} - y_i)^2}{n}}$$

Within the guidance of these functions, our train and test score for predicting the price using the last 4 hour data was 48.78 and 358.46 respectively. The same values when using the last hour data were 74.59 and 853.52. We can clearly see that with less data model errors prone to higher values. The epochs represented in the following figures, figure 11 and figure 12:



```
Epoch  10 MSE:  0.015013136900961399
Epoch  20 MSE:  0.012627288699150085
Epoch  30 MSE:  0.01167263463139534
Epoch  40 MSE:  0.010132432915270329
Epoch  50 MSE:  0.009467369876801968
Epoch  60 MSE:  0.008429103530943394
Epoch  70 MSE:  0.006221971940249205
Epoch  80 MSE:  0.0054846229031682014
Epoch  90 MSE:  0.004786794073879719
Epoch  100 MSE:  0.0023979544639587402
Epoch  110 MSE:  0.0018978299340233207
Epoch  120 MSE:  0.0016164242988452315
Epoch  130 MSE:  0.0014630978694185615
Epoch  140 MSE:  0.0014060009270906448
Epoch  150 MSE:  0.001337741268798709
Epoch  160 MSE:  0.00128701538778841
Epoch  170 MSE:  0.0012381235137581825
Epoch  180 MSE:  0.0011905019637197256
Epoch  190 MSE:  0.0011432282626628876
Train Score: 48.78 RMSE
Test Score: 358.46 RMSE
```

Figure 11: Model Score using 4h data



```
Epoch  10 MSE:  0.024812284857034683
Epoch  20 MSE:  0.011450158432126045
Epoch  30 MSE:  0.006227380130439997
Epoch  40 MSE:  0.0043013207614421844
Epoch  50 MSE:  0.003574211616069078
Epoch  60 MSE:  0.003262727055698633
Epoch  70 MSE:  0.003147034905850887
Epoch  80 MSE:  0.0031248307786881924
Epoch  90 MSE:  0.003125232644379139
Epoch  100 MSE:  0.0031192011665552855
Epoch  110 MSE:  0.0031121119391173124
Epoch  120 MSE:  0.00310723460279405
Epoch  130 MSE:  0.0031018024310469627
Epoch  140 MSE:  0.0030959269497543573
Epoch  150 MSE:  0.0030896724201738834
Epoch  160 MSE:  0.0030828320886939764
Epoch  170 MSE:  0.0030753491446375847
Epoch  180 MSE:  0.0030670729465782642
Epoch  190 MSE:  0.003057816531509161
Train Score: 74.59 RMSE
Test Score: 853.52 RMSE
```

Figure 12: Model Score using 1h data

Our results for price prediction were acceptable within range, however, they didn't lead to a predictive edge. As of date, the maximum and minimum difference between our prediction and the actual price for the last 4 hours of data to predict the next 2 minutes were 2006.814 $ and 0.158 $ respectively.

These values reduced greatly when we used the last hour data to predict the next 30 seconds as the maximum difference was 424.728$ and the minimum was only 1.293$ The following figures, figure 13 and figure 14 represents our prediction and the actual price, where purple is the actual price and the green line indicates the prediction.

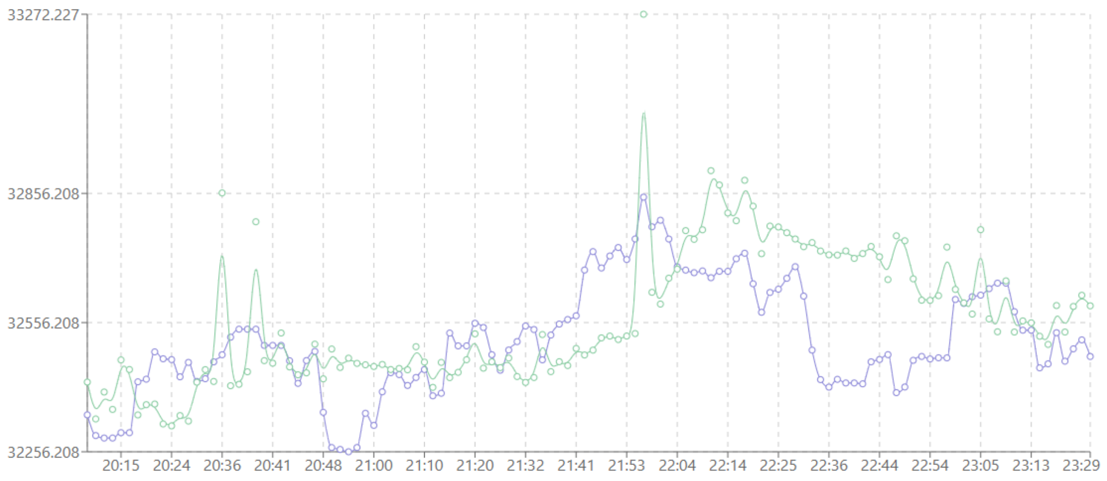


Figure 13: Actual Bitcoin Prices vs Predicted Bitcoin Prices using 1h data

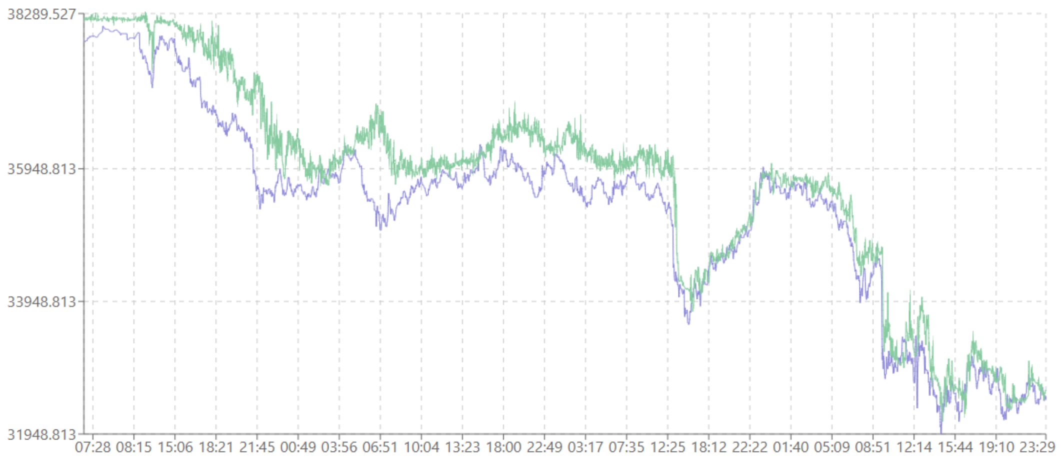


Figure 14: Actual Bitcoin Prices vs Predicted Bitcoin Prices using 4h data

6. CONCLUSION

To sum up, we proposed a prediction approach and an implementation to predict price and the trend of volatile Bitcoin prices, using the Long Short-Term Memory model. We created a REST API to publish our findings which can be accessible from using the following URL: https://bitcoinpriceprediction-mu.herokuapp.com/api/bitcoin. We also created a single-page application to visualize our findings which can be located at the following URL: https://bitcoinpriceprediction.vercel.app. We shared our implementation details using a Github repository to leverage from open-source, which can be found using the following link: https://github.com/yecicen/bitcoinpriceprediction.

With this project, we provided data gathering mechanisms and data preparation processes for our machine learning models. We then used this real-time data to create LSTM models for price and trend prediction.

6.1 Results Discussion

Our predictions of the trend (whether the price will go up or down) using the last 4 hours of data to predict the next two minutes had an accuracy of 46.38% while using last hour data to predict the next 30 seconds gave us an accuracy of 43.22%. Even though our model didn't provide a predictive edge, the price prediction model produced satisfying results overall as the maximum difference between our prediction and the actual price on the last 4hours of data was 2006.314$ and the minimum was 0.158$. The same values for the last hour data to predict the next 30 seconds produced a maximum of 424.728$ and a minimum of 1.293$ difference. Throughout the iterations on machine learning models, the RMSE of the trained model varied from 72.345 to 89.23 and for the test model, it was from 234 to 2345. Our proposed approach using LSTM models had a leaning towards the upper trend where bitcoin prices fall most of the time, meaning the LSTM model doesn't yield a predictive edge.

6.2 Lessons Learned

In this project, we benefited from various libraries and tools. We added many new tools to our toolset and gained insights and know-how about digital currencies and machine learning approaches. Key lessons learned from our implementation and research progress were,

- LSTM models can be used effectively for time-series and forecasting data
- It is better to use a relatively short span of a time period to increase the accuracy of predictions when using RNN based approaches.
- Trying out different optimizers and loss functions can yield a better understanding of the data and provide more accurate estimations.
- Visualizing the process and the outputs improve our ability to make decisions when using predictive approaches.
- Creating a REST API endpoint and actual product with a web application leads to a degree of professionalism, which improves the quality of the process and end result.
- Agile development and continuous integration / continuous delivery speed up the implementation of the backend and frontend requirements; the same methodology could have been applied to creating machine learning models with greater time intervals, as the ML models require time to train them.

6.3 Future Work

We are hoping to provide more functionality in terms of both prediction models and publishing the results of these prediction models. We are hoping to add more data source providers and train parallel models to evaluate each model's findings. Within the timeline of 3 months, we are hoping to try out different approaches on machine learning models, such as developing an Autoregressive integrated moving average (ARIMA) [51] model and another model based on Random Forest Tree [52] algorithms. Our focus on creating better user interfaces throughout various charts and analysis tools, and providing more REST endpoints will continue.

We are also planning to create an arbitrage-focused model, where we will examine the price differences on the same timestamp from different data providers. We are hoping to research the correlation between price changes in vendor A and vendor B, where the price changes on vendor A can yield the same rate of change on vendor B. If so, there can be another machine learning model to predict the frequency of these gaps to exploit them using High-Frequency Algorithmic Trading approaches.

Lastly, we are hoping to provide API documentation where other developers can use our predictive system more easily.

7. REFERENCES

[1] Nakamoto, Satoshi. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. Cryptography Mailing list at https://metzdowd.com.

[2] https://coinmarketcap.com/all/views/all/

[3] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

[4] https://coinmarketcap.com/halving/bitcoin/

[5] https://www.buybitcoinworldwide.com/volatility-index/

[6]https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/

[7] A Method for Stochastic Optimization, Diederik P. Kingma and Jimmy Ba, 2017

[8] Muniye, Temesgen & Rout, Minakhi & Mohanty, Lipika & Satapathy, Suresh. (2020). Bitcoin Price Prediction and Analysis Using Deep Learning Models. 10.1007/978-981-15-5397-4_63.

[9] Yang & Zheng, Zibin & Dai, Hong-Ning. (2020). Enhancing Bitcoin Price Fluctuation Prediction Using Attentive LSTM and Embedding Network. Applied Sciences. 10. 4872. 10.3390/app10144872.

[10] S. Velankar, S. Valecha, and S. Maji, "Bitcoin price prediction using machine learning," 2018 20th International Conference on Advanced Communication Technology (ICACT), 2018, pp. 144-147, doi: 10.23919/ICACT.2018.8323676.

[11] Predicting The Prices Of Bitcoin Using Data Analytics Dr. M. Sharmila Begum1 , G. Jayashree2 , Z. Mahaboob Asfia3 , R. Subhasri4 and L. Vishnu Priya

[12] Phaladisailoed, Thearasak & Numnonda, Thanisa. (2018). Machine Learning Models Comparison for Bitcoin Price Prediction. 506-511. 10.1109/ICITEED.2018.8534911.

[13] Alahmari, Saad. (2020). Predicting the Price of Cryptocurrency using Support Vector Regression Methods. JOURNAL OF MECHANICS OF CONTINUA AND MATHEMATICAL SCIENCES. 15. 10.26782/jmcms.2020.04.00023.

[14] C2P2: A Collective Cryptocurrency Up/Down Price Prediction Engine, Chongyang Bai and Tommy White and Linda Xiao and V. S. Subrahmanian and Ziheng Zhou, 2019, 1906.00564

[15] https://www.python.org/

[16] https://pytorch.org/

[17] http://torch.ch/

[18] https://nomics.com/

[19] https://coinmarketcap.com/

[20] https://messari.io/

[21] https://www.coindesk.com/

[22] https://www.bitmex.com/

[23] https://nextjs.org/

[24] https://reactjs.org/

[25] https://tailwindcss.com/

[26] https://recharts.org/en-US/

[27] https://vercel.com/

[28] https://nodejs.org/en/

[29] https://expressjs.com/

[30] https://www.npmjs.com/package/cors

[31] https://www.npmjs.com/package/dotenv

[32] https://mongoosejs.com/

[33] https://www.npmjs.com/package/uuid

[34] https://www.mongodb.com/

[35] https://www.heroku.com/

[36] https://numpy.org/

[37] https://pandas.pydata.org/

[38] https://scikit-learn.org/

[39] https://docs.python-requests.org/en/master/

[40] https://www.investopedia.com/terms/b/bitcoin.asp

[41]https://www.dw.com/en/venezuelans-try-to-beat-hyperinflation-with-cryptocurrency-revolution/a-57219083

[42]https://www.reuters.com/world/americas/el-salvador-approves-first-law-bitcoin-legal-tender-2021-06-09/

[43] https://vocal.media/theChain/10-major-differences-between-crypto-and-stocks

[44] Sherstinsky, Alex. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena. 404. 132306. 10.1016/j.physd.2019.132306.

[45] Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, Yonghui Wu and Mike Schuster and Zhifeng Chen and Quoc V. Le and Mohammad Norouzi and Wolfgang Macherey and Maxim Krikun and Yuan Cao and Qin Gao and Klaus Macherey and Jeff Klingner and Apurva Shah and Melvin Johnson and Xiaobing Liu and Łukasz Kaiser and Stephan Gouws and Yoshikiyo Kato and Taku Kudo and Hideto Kazawa and Keith Stevens and George Kurian and Nishant Patil and Wei Wang and Cliff Young and Jason Smith and Jason Riesa and Alex Rudnick and Oriol Vinyals and Greg Corrado and Macduff Hughes and Jeffrey Dean, 2016,1609.08144

[46] Ghosh, Anirudha & Sufian, A. & Sultana, Farhana & Chakrabarti, Amlan & De, Debashis. (2020). Fundamental Concepts of Convolutional Neural Network. 10.1007/978-3-030-32644-9_36.

[47] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[48] https://developer.mozilla.org/en-US/docs/Glossary/SPA

[49]https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/

[50] https://statweb.stanford.edu/~susan/courses/s60/split/node60.html

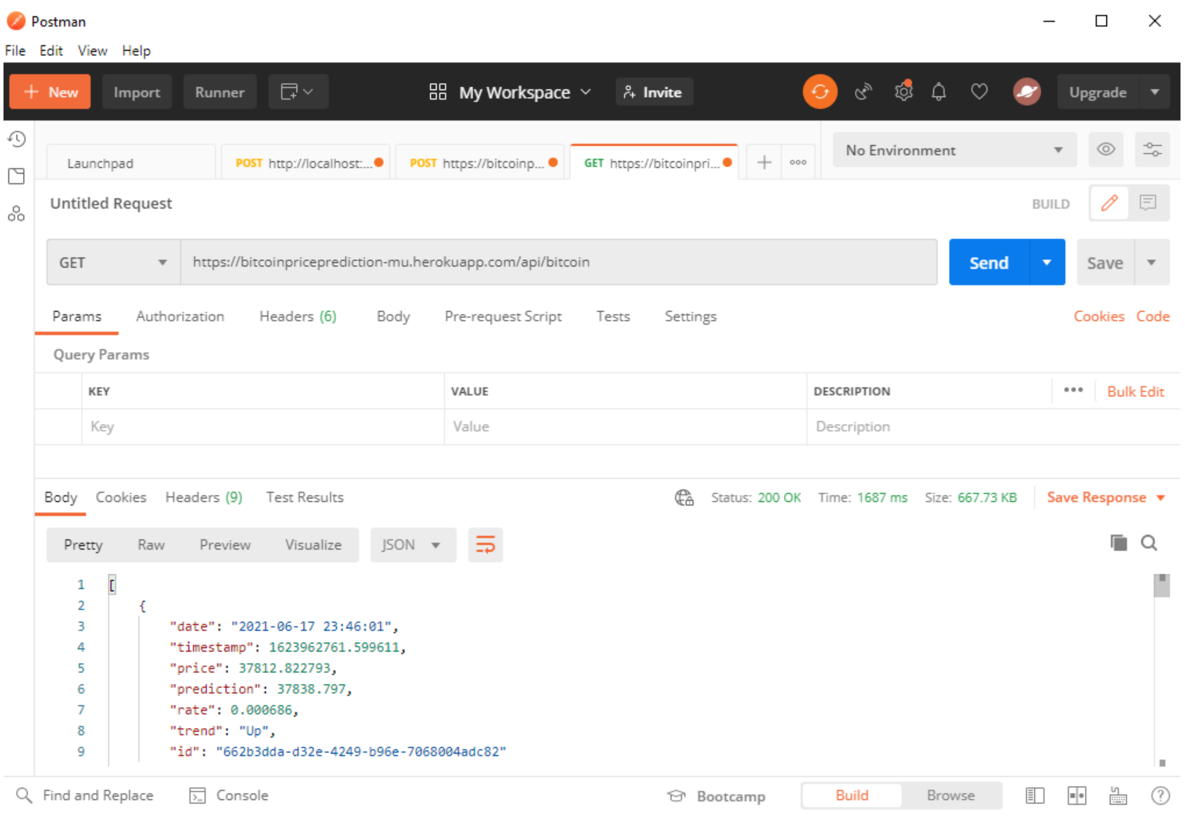
[51]Abdulqader, Qais. (2016). Time Series Forecasting Using Arima Methodology with Application on Census Data in Iraq. Science Journal of University of Zakho. 4. 258-268. 10.25271/2016.4.2.116.

[52] Ali, Jehad & Khan, Rehanullah & Ahmad, Nasir & Maqsood, Imran. (2012). Random Forests and Decision Trees. International Journal of Computer Science Issues(IJCSI). 9.

8. APPENDICES

Please navigate to https://github.com/yecicen/bitcoinpriceprediction for the source code and data.

Appendix A, Screenshot of REST API

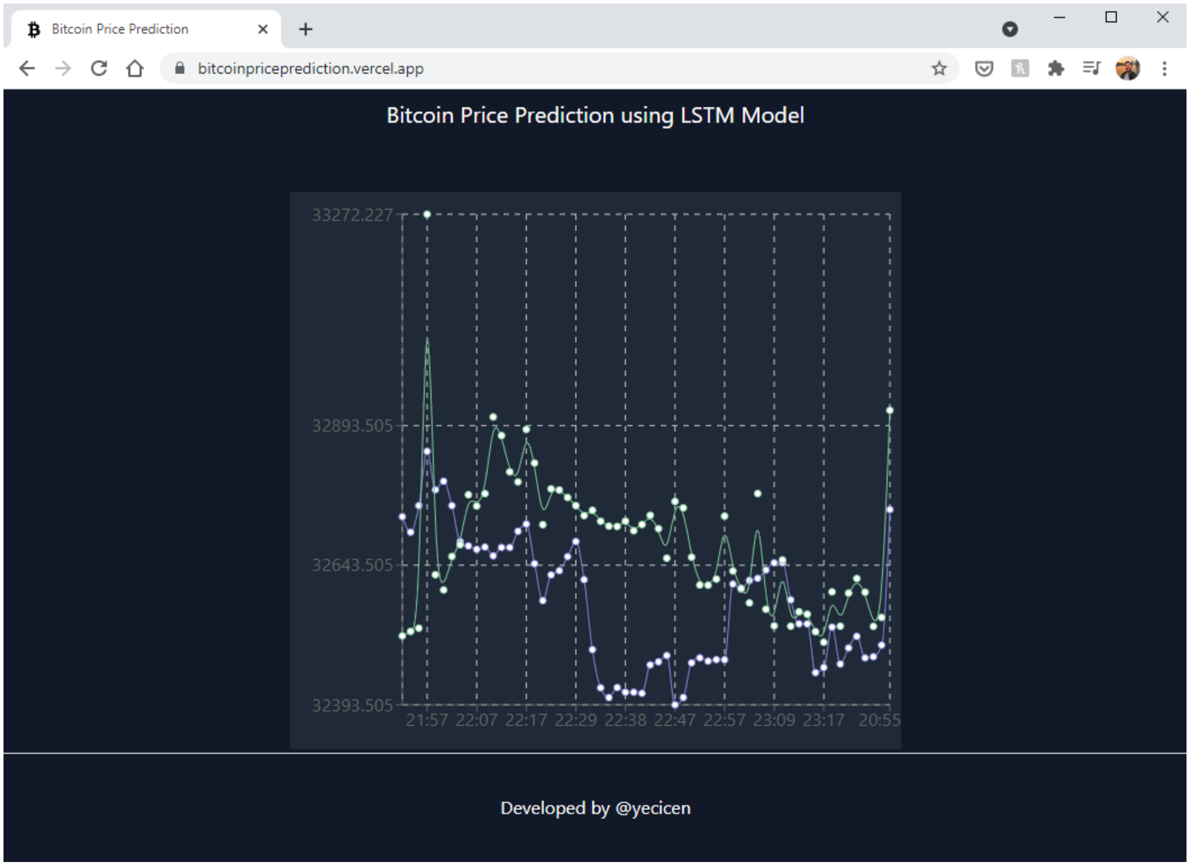


GET https://bitcoinpriceprediction-mu.herokuapp.com/api/bitcoin

Status: 200 OK Time: 1687 ms Size: 667.73 KB

```
[
    {
        "date": "2021-06-17 23:46:01",
        "timestamp": 1623962761.599611,
        "price": 37812.822793,
        "prediction": 37838.797,
        "rate": 0.000686,
        "trend": "Up",
        "id": "662b3dda-d32e-4249-b96e-7068004adc82"
```

Appendix B: Screenshot of our website


Bitcoin Price Prediction
bitcoinpriceprediction.vercel.app
Bitcoin Price Prediction using LSTM Model
33272.227
32893.505
32643.505
32393.505
21:57
22:07
22:17
22:29
22:38
22:47
22:57
23:09
23:17
20:55
Developed by @yecicen