

**Student ID: 20835006**

**Name: Kareem Ayad**

**Module Code: CI601**

**2022/2023**

**Applying the LSTM model on Bitcoins Historical Prices**

Table of Contents

Abstract …………………………………………………………………………………………………3

Introduction ……………………………………………………………………………………………..3

Aims and Objectives …………………………………………………………………………………..3

Motivation ………………………………………………………………………………………………..3

Related work …………………………………………………………………………………………….6

Methodology …………………………………………………………………………………………….7

RNN ……………………………………………………………………………………………..8

LSTM ……………………………………………………………………………………………9

Programming Language ……………………………………………………………………..10

Libraries ……………………………………………………………………………………….11

Design ………………………………………………………………………………………………….12

Gathering the data ……………………………………………………………………………12

Preprocessing ………………………………………………………………………………...13

Building The LSTM Model …………………………………………………………………...14

Time Frames ………………………………………………………………………………….15

Train Test Split ………………………………………………………………………………..15

Plotting ………………………………………………………………………………………...15

Results ………………………………………………………………………………………………….16

Hyperparameter ………………………………………………………………………………16

Plots ……………………………………………………………………………………………17

Discussion …………………………………………………………………………………….18

Conclusion ……………………………………………………………………………………………..23

Future Work …………………………………………………………………………………...23

Reflections …………………………………………………………………………………….24

References …………………………………………………………………………………………….25

Abstract

Cryptocurrencies have become increasingly popular and widely adopted globally as the technology behind them has grown rapidly. Bitcoin, the first and most prominent cryptocurrency, was released in 2009 with an initial price of 0.0008 USD and has since reached an all-time high of 69000 USD in November 2021. With no signs of slowing down, the use cases for cryptocurrencies continue to expand.

In the field of finance, artificial intelligence has played a significant role in predicting prices for time series data. This dissertation explores the use of LSTM models to predict future Bitcoin prices and gain insight into its trajectory. The study also investigates how the LSTM model performs when presented with different time frames, and how well it can accurately predict future prices under these varying conditions.

Introduction

Over the past few years, the financial and economic sector has experienced significant changes due to emerging technologies. One of these technologies is cryptocurrencies, which have impacted various sectors by allowing digital or virtual currencies to be traded for goods and services. Cryptocurrencies use a decentralized system that employs cryptography to verify transactions and maintain records, unlike traditional centralized authorities. Bitcoin is the first and most popular cryptocurrency, launched in 2009 at a price of 0.0008 USD per one Bitcoin, and reaching an all-time high of 69000 USD per Bitcoin in November 2021.

The popularity of cryptocurrencies stems from their high security, enabled by cryptography methods that prevent counterfeiting and double-spending. They are also decentralized, distributed across many machines rather than relying on a single server, making them immune to government manipulation. The cryptocurrency market has grown exponentially, with a market cap of 2.7 trillion USD by the end of 2021.

Another emerging technology impacting various sectors is machine learning and artificial intelligence, which enable problem-solving by combining computer science and robust datasets. The global artificial intelligence market has been growing at a rate of approximately 54% year on year, reaching 22.6 billion USD in 2020. Financial institutions have recognized the potential of these technologies to increase revenue growth opportunities and minimize operation expenses by automating intensive processes.

In the cryptocurrency space, numerous individuals have been exploring the potential of machine learning and artificial intelligence to gain profits. While cryptocurrencies are still considered a new and emerging technology, the possibilities it presents have motivated data scientists and analysts to apply different ways to explore its potential.

Aims and Objectives

The goal of this project is to examine the effectiveness of the LSTM (Long Short Term-Memory) model in predicting the future prices of Bitcoin, as well as determining whether the market will rise or fall. Additionally, the project seeks to evaluate how the LSTM model performs when provided with various time frames for training and testing. The key objectives for this project are to obtain Bitcoin's historical price data sets from exchanges, which will be used to train and test the LSTM model, to construct and develop the LSTM model, and to apply the LSTM model to Bitcoin's historical data, which was retrieved from the exchange. Finally, the project will examine how the LSTM model functions when given different time frames for analysis.

Motivation

Investors have been drawn to the cryptocurrency market due to its significant growth in recent years and the potential investment opportunities it presents. Cryptocurrencies are highly volatile, which makes them a popular choice for investors looking to gain profits quickly. This volatility sets them apart from more traditional investment options like stocks and gold. The market leader, Bitcoin, is particularly volatile, as shown in Figure 1, which compares its volatility to that of gold, the S&P 500, and the HS300.

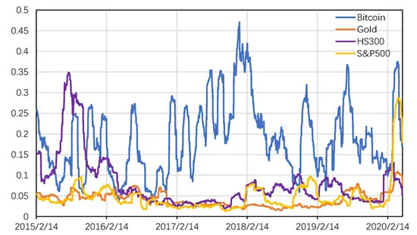


Figure 1: Volatility Between the Top Investment Assets (Li, Zheng and Dai, 2020).

Figure 1 shows that the volatility of Bitcoin is significantly higher than that of other investment options. The significant price fluctuations in Bitcoin are due to multiple factors but can be categorized into two main reasons. Firstly, the cryptocurrency market is relatively new and constantly evolving, with no physical assets backing it. This makes it susceptible to market manipulation and misinformation, which can lead to investors making unwise decisions. The lack of regulation in the cryptocurrency market makes it vulnerable to market manipulation, fake news and other factors that can impact the decisions of investors and cause significant fluctuations in the price of Bitcoin. This, coupled with the fact that the cryptocurrency space is a new and developing market with no physical representation, contributes to the high volatility of Bitcoin. (Li, Zheng and Dai, 2020). Another key reason for the high price fluctuations of Bitcoin is the lack of government regulations. Unlike traditional financial markets that are government-regulated, the cryptocurrency space currently lacks such regulations. Unlike traditional financial markets that are government regulated, the field of cryptocurrencies lacks government regulations. Bitcoin was the first cryptocurrency to be created and it currently holds the position of being the market leader in the cryptocurrency space. Bitcoin's success has inspired many developers to explore the potential of cryptocurrency technology and create new cryptocurrencies with different use cases. Following Bitcoin's success, many other projects have emerged in the cryptocurrency space, with the hope of becoming the next dominant player after Bitcoin. The data depicted in Figure 2 illustrates a steady increase in the number of cryptocurrencies developed from 2013 to November 2021, which spans a period of eight years. Figure 2 illustrates that in 2013, 66 cryptocurrencies were created. The number of cryptocurrencies has steadily increased over the years, reaching over 7500 in November 2021 and showing no indication of slowing down. The cryptocurrency market has been continuously growing and is expected to keep growing in the future. However, the rapid increase in the number of cryptocurrencies in a short period of time has made it difficult for researchers to thoroughly investigate and research each of them. This implies that there is still a great deal of unexplored territory and research that needs to be done in the cryptocurrency space.

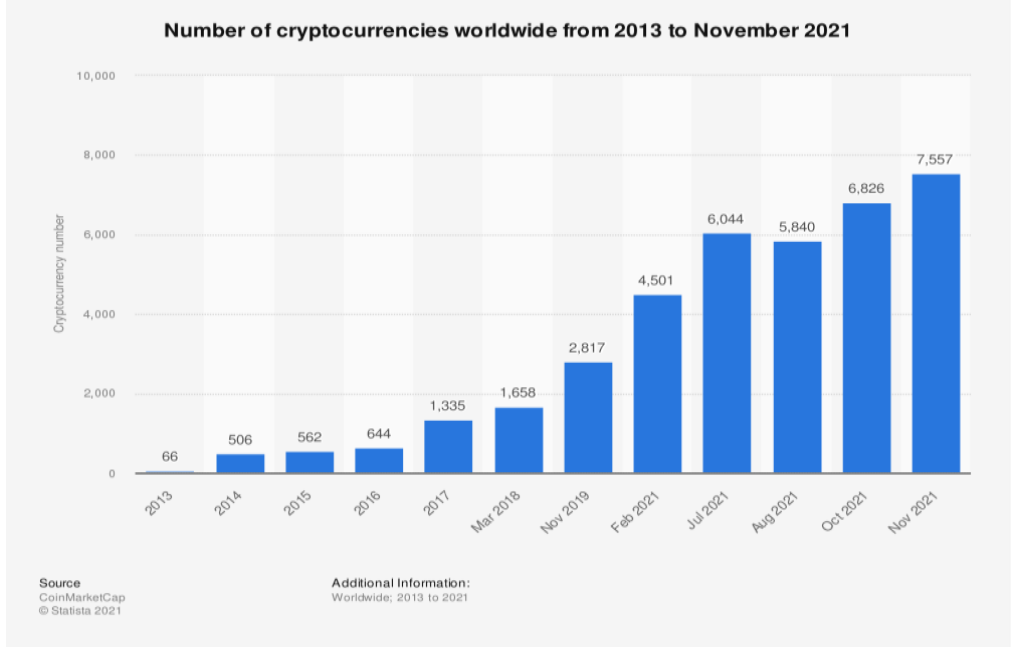


Figure 2: Number of Cryptocurrencies from 2013 to November 2021(Statista, n.d.)

The rapid growth of the cryptocurrency market has caught the attention of major institutions. In October 2021, Bank of America published its first research on Bitcoin and cryptocurrencies, acknowledging that they are too significant to disregard (Nasdaq.com, 2021). Furthermore, the report from Bank of America also expressed the opinion that there may be more potential for cryptocurrencies than skeptics anticipate. Another prominent financial institution, JP Morgan, has predicted that the long-term price of Bitcoin may reach up to 150,000 USD in the future (cnbctv18.com, 2022). These comments indicate the significance of the emerging cryptocurrency market and the potential it might have in the future.

Related work

The growth of the Cryptocurrency space in recent years has drawn the attention of many researchers, who have undertaken investigations and studies in the area. Several papers have been published with similar objectives to this research project. The research paper utilized both Artificial Neural Networks (ANN) and Support Vector Machines (SVM) as analytical tools to analyze the Bitcoin Blockchain, with the goal of predicting Bitcoin's future prices. The research paper utilized Artificial Neural Networks (ANN) and Support Vector Machines (SVM) to analyze the Bitcoin blockchain and attempt to forecast Bitcoin's future prices. The study achieved a 55% accuracy rate in its predictions. The authors also found that using blockchain data alone had limited predictability (Arowolo et al., 2022). Another study published by IEEE titled "Predicting the Price of Bitcoin Using Machine Learning" employed the LSTM network to forecast Bitcoin's future price. This paper was published in early 2018, before the significant price fluctuations of Bitcoin in 2020. To summarize, the study published by IEEE attempted to predict Bitcoin's future price using the LSTM network but concluded that there was not enough previous data to accurately predict prices. The paper was published in early 2018, just before the high price fluctuations of Bitcoin that occurred in 2020 (McNally, Roche and Caton, 2018). A research paper published by IEEE named "A New Forecasting Framework for Bitcoin Price with LSTM" used various LSTM models, including LSTM with AR model and a conventional LSTM model, to predict Bitcoin price (Wu et al., 2018). Another notable research paper by "IEEE" titled "Bitcoin Transaction Forecasting with Deep Network Representation Learning" used transaction graphs to predict Bitcoin's price. This paper is distinct from others as it utilizes transaction graphs rather than Bitcoin price graphs for forecasting (Wei, Zhang and Liu, n.d.). Numerous research papers have been published using diverse machine learning techniques to attempt to forecast the future prices of cryptocurrencies. Many of the research papers in the cryptocurrency space have focused on using Bitcoin's data, since it is the market leader in the industry. Many of the research papers that have been published on using machine learning to predict cryptocurrency prices have focused on Bitcoin, which is the leader in the cryptocurrency space and has the longest historical data as it was the first cryptocurrency to be implemented. This research paper utilizes the LSTM model and applies it to the historical data of Bitcoin. This paper differs from other papers in that it applies the LSTM model to Bitcoin's data but with different time frames. The purpose of this paper is to examine the performance of the LSTM model when applied to different time frames of Bitcoin's historical data. The purpose of exploring different time frames in this research is to identify potential patterns in the Bitcoin data that might not be apparent when analyzing it at a single time frame. By combining these patterns, the goal is to generate more accurate predictions using the LSTM model.

Methodology

It is important to first understand and develop the model that will analyze the data before beginning the project. To work on the data in this project, the LSTM (Long Short-Term Memory) model has been selected. LSTM, which stands for Long Short-Term Memory, is a subtype of Recurrent Neural Network (RNN) that can recognize and learn patterns in sequential data that have a specific order dependence. I have selected the LSTM model for this project as it has shown effectiveness in handling time series data by processing, classifying, and making predictions. The LSTM was developed to overcome the issues that arose when RNNs were used to analyze time-series data.

RNN

An artificial neural network called RNN (Recurrent Neural Network) is designed to handle sequential or time-series data. Recurrent Neural Networks (RNNs) are commonly applied in scenarios involving sequential or time-series data, such as translation, speech recognition, natural language processing (NLP), and image captioning tasks. RNNs learn and improve by utilizing training data. RNNs are distinctive from other types of neural networks because they have the ability to retain memory. RNNs have a unique characteristic in that they have a memory that allows them to take into account previous inputs when determining the current output. This contrasts with traditional neural networks, which treat inputs and outputs as completely independent of each other. One of the primary issues with traditional Neural Networks is that they only consider the current input and ignore any prior inputs. The second problem with traditional neural networks is that they cannot retain or store information from previous inputs. The final issue is that traditional neural networks are not suitable for processing sequential data. As a result, RNNs are preferred because they rely on previous elements in the sequence to process the current input (Simplilearn.com, n.d.).

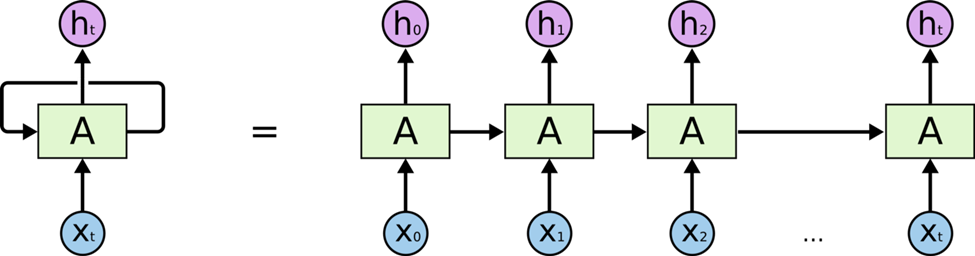


Figure 3: RNN Model (Olah, 2015).

In an RNN, the information is cycled through a loop within the hidden layer, allowing the network to maintain a memory of past inputs and use them to influence the current output. In Figure 3, the input layer (X) receives the input data and processes it before passing it to the middle layer (A). The middle layer (A) is composed of multiple hidden layers, each with its own biases, weights, and activation function. To standardize these parameters across the hidden layers, the RNN will normalize them, ensuring that each layer has the same parameters. The RNN model does not create multiple hidden layers, but instead, it creates one hidden layer and loops over it as many times as needed. This approach is different from the traditional neural network architecture where multiple hidden layers are used (Donges, 2019). The RNN model has different types, such as one-to-one, one-to-many, many-to-one, and many-to-many.

LSTM

The LSTM (Long-Short-Term-Memory) operates similarly to the control flow of RNN, as it processes input data while also propagating information forward. The key distinction between LSTM and RNN lies in the operations taking place inside the LSTM cell. The operations inside the LSTM cell allow for selective retention or forgetting of information. The primary concept of LSTM is based on the cell state and the gates associated with it. The cell state is responsible for carrying essential information throughout the sequence, which is regulated by the gates. The key idea behind LSTM is the cell state and its gates, which serve as the memory of the network and allow information from earlier time steps to affect later time steps, thereby reducing the impact of short-term memory limitations. The gates in LSTM control the flow of information into and out of the cell state as it moves through the network, allowing for removal or addition of information. The gates in the LSTM cell are neural networks that determine what information should be added or removed from the cell state. They learn to differentiate relevant information from irrelevant information during training.

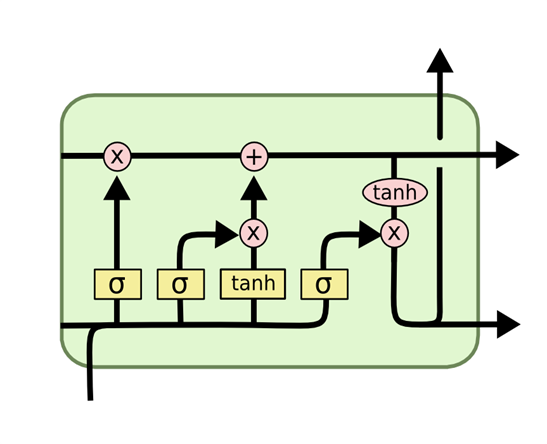


Figure 4 LSTM Model (Olah, 2015)

The gates in the LSTM cell use Sigmoid activation functions, which map values to a range between 0 and 1, unlike other activation functions that map to a range between -1 and 1. This property of sigmoid activation is useful for forgetting or updating information because any value multiplied by 0 becomes 0. The effect of this is that certain values can be removed or retained n (Jason. brownlee.39,

2017). When a value is multiplied by 0, it disappears or is forgotten. On the other hand, when it is multiplied by 1, it stays the same and is retained. During training, the LSTM can learn which values should be forgotten and which ones should be retained. Within the LSTM cell, there are three key gates that regulate information flow. The initial gate is known as the forget gate. The forget gate of the LSTM cell is responsible for determining what information should be retained and what should be discarded. To make this decision, the information from both the current input and the previous hidden states is passed through a sigmoid function. The forget gate of the LSTM cell determines which information to retain and which to discard. It utilizes the sigmoid function to process both the current input and information from previous hidden states, producing output values between 0 and 1. An output closer to 1 signifies the information should be retained, while an output closer to 0 signifies the information should be discarded (Analytics Vidhya, 2021). The second gate in the LSTM cell is called the Input gate, which is responsible for updating the cell state. It takes input the previous hidden state and the current input, and processes it through the sigmoid function. After passing the previous hidden state and the current input through the sigmoid function, the Input gate in the LSTM cell selects the information that will be kept and the information that will be discarded. The output gate in the LSTM cell determines the information that will be output as the next hidden state. At each iteration, the LSTM cell has three gates: forget gate, input gate, and output gate. The forget gate decides what information from previous steps should be kept. The input gate decides what new information should be added to the current state. The output gate determines what the next hidden state will be. These gates work together to control the flow of information in the LSTM cell.

Programming Language

In preparation for the project, I needed to select a programming language to develop and construct the model. I chose Python as the programming language for my project due to its extensive library support. Python has a wide range of libraries available, and all the functions I required for my project were available in these libraries. Python was chosen as the programming language for the project due to its extensive libraries and easy-to-learn syntax. The availability of necessary functions in libraries saves time and makes it easier to focus on problem-solving. Compared to other programming languages, Python is relatively simple to use, minimizing the need for debugging code.

* Vast libraries: Python has a vast collection of open-source libraries for scientific computing and machine learning, such as NumPy, Pandas, TensorFlow, and Keras, that make it easier to implement and train LSTM models.
* Easy-to-learn syntax: Python has a simple and easy-to-learn syntax, which makes it an ideal language for beginners who want to learn machine learning.
* Large community: Python has a large community of developers who contribute to the development of various libraries and frameworks. This means that there are plenty of resources available online for learning and troubleshooting.
* Interpreted language: Python is an interpreted language, which means that the code can be run and tested immediately without the need for compiling. This makes it a highly productive language for data analysis and machine learning.
* Flexibility: Python is a flexible language that can be used for a wide range of applications, including web development, data analysis, and machine learning. This means that once you learn Python, you can apply it to many different projects and industries.

Libraries

I chose to utilize two primary libraries for this project, the first being sklearn. Sklearn is a well-known and widely used machine learning library. The sklearn library was chosen for this project and it is a popular machine learning library that offers a range of effective tools for statistical modeling and machine learning. It was utilized to perform various tasks such as data splitting, scaling, and accuracy testing. Sklearn is a popular machine learning library that offers several benefits, including:

* Easy to use: Sklearn provides a simple and intuitive API that makes it easy to implement machine learning models, including LSTMs.
* Comprehensive documentation: Sklearn has extensive documentation that includes examples and tutorials for different machine learning tasks, making it easy to learn and use.
* Wide range of algorithms: Sklearn offers a wide range of machine learning algorithms, including classification, regression, clustering, and dimensionality reduction.
* Integration with other libraries: Sklearn integrates well with other popular Python libraries such as NumPy, Pandas, and Matplotlib, making it easy to use in data analysis and visualization tasks.
* Active development: Sklearn is actively developed and maintained, ensuring that users have access to the latest updates and bug fixes.

I utilized a second library, TensorFlow, which is a software library specifically designed for machine learning and artificial intelligence. TensorFlow is a widely used library that provides a lot of functionality for building and training neural networks. It has a large number of pre-built functions for creating different types of layers, optimization algorithms, and other building blocks commonly used in deep learning. TensorFlow is also designed to work efficiently with GPUs and other hardware accelerators, which can significantly speed up the training process. Additionally, TensorFlow has a large community of users and developers who have contributed to its development, making it a reliable and well-supported tool for deep learning projects. Additionally, I utilized Keras, a neural network library that operates on top of TensorFlow and provides a simplified high-level interface for constructing deep learning models. Both libraries offer interfaces for constructing and training models, and I utilized them to construct and train the LSTM model for this project.

Design

Gathering Data

To begin the project, the first step was to obtain historical data for Bitcoin. I chose to retrieve the data from cryptodatadownload.com, a website that has been a prominent source for cryptocurrency price data since 2017. To initiate the project, the first step was to find a source to collect the historical data of Bitcoins. I opted to obtain the data from a website named cryptodatadownload.com, which is a prominent website in the cryptocurrency market for gathering historical price data since 2017. According to the website, their data is widely used by academic and educational institutions for conducting research. The website offers two main benefits for collecting data. One advantage of using the website cryptodatadownload.com to gather Bitcoin's historical data is that it provides data for all time frames from any year. Additionally, the website updates its historical data regularly. Another advantage of the website is that it provides data from more than 25 global exchanges, making it convenient to obtain data from any exchange in the world. This enables users to easily find the desired data they need. Using hard coding to get the data myself would have been a difficult task because it would have restricted me to only one exchange for data collection. Moreover, I would not have had the option to choose from a diverse range of exchanges. Additionally, most APIs provide real-time data, whereas for this project, I needed historical data to test the accuracy of the prediction. If I had used hard coding, I would have faced several limitations in obtaining the historical data.

Preprocessing

Upon receiving the data from the exchanges in a CSV file, the initial step is to scale and remove any superfluous details. The CSV file contains five key pieces of information that are classified into columns.

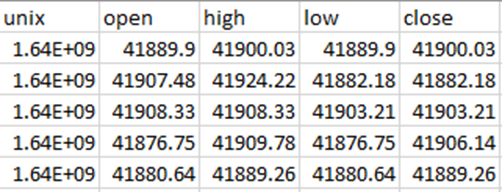


Table 1: Example of Retrieved Data from Exchanges

We can observe from Table 1 that the data includes the Unix time, opening price, closing price, lowest price in the given time frame, and the highest price in the given time frame. The LSTM model will require only the open price and the UNIX time for training and testing purposes. This implies that the columns representing high, low, and close prices are not relevant to the training and testing of the LSTM model and will be excluded from the analysis. Only the open price column and the UNIX time column will be used for this purpose. The subsequent step involves removing the columns containing the high, low, and close prices to retain only the UNIX time and open price data within the designated time frame. The time-series data obtained from exchanges is usually in the Unix time format, which is difficult to read and interpret. Thus, the next step is to convert the Unix time into a readable date and time format. After obtaining the data from the exchange in Unix time, I used the to\_datetime function available in the Pandas library to convert it into readable date and time. This allowed me to have all the required data to proceed with my project. The next step after obtaining the necessary information and removing any unnecessary data is to scale the data. Scaling the data is crucial in machine learning because the data may be too large and vary greatly. Scaling data is crucial in machine learning as it helps to normalize the data, especially when the data is on different scales or ranges. It ensures that all features contribute equally to the model and prevents any one feature from dominating the others. This is particularly important when comparing multiple datasets, as they may not be on the same scale, leading to incorrect or misleading results. To ensure that the data used for machine learning is accurate, it's important to scale it. This is especially important when dealing with multiple sets of data that may not be on the same scale. In this project, the StandardScaler() function from Sklearn library was used to scale the data to be between -1 and 1. After applying the StandardScaler() function from the Sklearn library, the data is now transformed and scaled, making it suitable for machine learning model training and testing. To test if scaling the data differently would result in different outcomes, I created another function for scaling the data in a distinct manner. In this new function, I utilized the MinMaxScaler() function from the Sklearn library to scale the data. This function scales the data between 0 and 1, and if the data contains any negative values, it will scale it between -1 and 1. The key distinction between the two scaling techniques is that StandardScaler() aligns with the standard normal distribution, which results in a mean of 0 and scales the data to have unit variance. On the other hand, MinMaxScaler() scales the data to be between 0 and 1. In contrast, MinMaxScaler() scales the data between 0 and 1, regardless of whether the data contains negative values or not. If there are any negative numbers in the data, MinMaxScaler() will scale them to -1 and 0 if the feature range is set to (-1,1), otherwise, it will scale them to 0 and 1. Using two different scaling methods gives me the flexibility to choose how I want to scale the data and observe whether it will produce different results during the training and testing stages.

Building The LSTM Model

To train and test the data, I needed to construct the LSTM model using the Keras library. Keras is a high-level neural networks API, written in Python and capable of running on top of several lower-level frameworks, including TensorFlow and Theano. It provides easy-to-use functions and interfaces for building, training, and testing deep learning models, including LSTM models. Keras library is used to build the LSTM model effectively by providing all the required functions. This allows us to easily construct the model for training and testing. I chose to utilize two layers of LSTM along with one extra dense layer in my model. I set the number of units in the LSTM layers to be 32, which means that both LSTM layers will have 32 units. I included a dropout layer in both LSTM layers to prevent overfitting. Dropout is a technique that randomly drops out some units during training to reduce the dependency on certain features. A dropout is a regularization technique that helps prevent overfitting in neural networks. It works by randomly dropping out (i.e., setting to zero) a fraction of the neurons in the layer during training. This forces the network to learn more robust representations of the input data and helps prevent it from memorizing the training examples too well, which can lead to poor performance on new data. After incorporating the two LSTM layers, I included a dense layer which connects all the neurons in the network, receiving inputs from the neurons of all the previous layers.

Time Frames

A critical aspect of my project is the creation of the timeframes function, which is responsible for generating different time-framed data required for training and testing the LSTM model. Fetching different time-framed data from exchanges every time I want to train and test the data can be inefficient. To avoid the time-consuming process of fetching data from exchanges and manually adding it to the code, I have developed a function that efficiently transforms the available data to any desired time frame. After obtaining the 1-minute time-framed data from the exchange, the function I have created will allow me to transform the data into any desired time frame. For instance, if I require data in time frames such as 5 minutes or 45 minutes, the function I have developed will efficiently transform the existing 1-minute data into the desired time frames. This capability enables me to test the LSTM model quickly and effectively on different time frames.

Train test Split

To facilitate the training and testing process, I need to partition my data into two sets: training data and testing data. To accomplish this task, I utilized the pre-existing function called train\_test\_split from the Sklearn library. This function proved to be highly convenient, enabling me to split, train, and test the data with just a single line of code. Notably, it offers the flexibility to specify the desired percentage of data for training, with the remaining portion automatically allocated for testing purposes. Before proceeding with the training and testing process, it is essential to scale the data appropriately to ensure optimal performance of the function. In this project, I chose to allocate 80% of the data for training the model and reserve the remaining 20% for testing purposes. Once the data is split and the training phase is completed, the function will proceed to evaluate the model's performance through testing.

Plotting

The final step of the project involves visualizing the results through plotting. For this purpose, I have selected two graphs to present and analyze the outcomes. The first graph displays the training loss throughout the epochs, providing insights into how effectively the model is fitting the training data. The second graph compares the predicted Bitcoin price generated by the model with the actual price. This visual representation allows for a clear assessment of the model's performance. These two graphs provide visual representations that aid in evaluating the performance and effectiveness of the model. They offer insights into the training progress and the model's ability to predict Bitcoin prices accurately, allowing for a better understanding of its performance.

Results

Hyperparameter

In this section, I will be analyzing and discussing the outcomes of my LSTM model. To assess and evaluate the model's performance, I conducted tests using various time frames to observe its behavior and assess the diverse outputs it produces.

To begin the testing phase of the model, it is necessary to adjust the hyperparameters. The adjustable hyperparameters in the model include the batch size, the number of epochs, and the time frames used. These parameters play a crucial role in shaping the model's performance and can be optimized to achieve better results. The batch size refers to the number of training examples used in each iteration during the training process. It is often recommended to choose a batch size that is a power of 2, such as 32, 64, or 128. This is because using power-of-2 batch sizes can improve the efficiency of GPU processing, which can lead to faster training times. The commonly recommended batch sizes for training neural networks are 16, 32, and 64. These values are frequently used in practice due to their effectiveness in balancing computational efficiency and model performance. I opted to use a batch size of 32 for my LSTM model since it is well-suited to the size of my dataset. By choosing a batch size that aligns with the dataset size, I can effectively utilize computational resources and optimize the model's training process. The other adjustable hyperparameter is the number of epochs. An epoch represents the number of times the model's algorithm will iterate through the entire dataset during training. The epoch number is a crucial parameter in the model's training algorithm. The epoch number determines the number of complete iterations the model will go through the entire training dataset. Each epoch can be seen as a loop that iterates through the entire dataset, with each iteration processing a batch of data. The batch size has an impact on various aspects, including the duration of training per epoch, the overall training time, and the model's performance or quality. I have chosen to utilize 10 epochs during the testing phase, as it is a commonly used standard. Additionally, when I convert my data to larger time frames, the data size decreases, thereby requiring fewer epochs to train the data effectively. The final adjustable hyperparameter is the number of time frames used. This refers to the duration or interval of data used for training and testing the LSTM model. A time frame refers to the duration or interval of data used for analysis and prediction. It determines the granularity of the data and can vary depending on the specific requirements of the project. In this project, I have utilized data with a time frame of 1 minute. To provide flexibility in analyzing the data at different time resolutions, I have developed a custom function that allows me to modify the time frame to any desired duration. This function enables me to adjust the data to different time frames based on my specific requirements. By utilizing the custom function, I developed, I can easily transform the existing 1-minute time frame data to other time frames such as 60 minutes or 30 minutes. This flexibility allows me to experiment with different time frames and observe how the LSTM model performs and predicts the data in each case. It provides me with the opportunity to assess the model's behavior across various time resolutions and make informed decisions based on the outcomes.

Plots

To assess the performance of each model run, I employed two plots to visualize the results. The first plot focuses on illustrating the training loss. The training loss metric measures the extent to which the model fits the training data. It provides insight into how effectively the model captures the patterns and trends present in the training dataset. By monitoring the training loss, we can evaluate how well the model is learning from the data and adjusting its parameters to minimize the discrepancy between the predicted and actual values during training. Lower training loss values generally indicate a better fit of the model to the training data. Figure 5 illustrates the training loss of the model as a function of the number of epochs used for training. The plot provides a visual representation of how the training loss changes over the course of training. It allows us to observe the convergence of the model's training loss, indicating how well the model is learning from the data. A decreasing trend in the training loss over epochs indicates that the model is progressively fitting the training data better. Figure 6 illustrates the comparison between the model's predictions and the actual data. It provides a visual representation of how well the model performs in predicting the target variable. By plotting the predicted values alongside the actual values, we can visually assess the accuracy of the model's predictions. Ideally, the predicted values should closely align with the actual values, indicating a strong predictive performance of the model. Discrepancies between the predicted and actual values may indicate areas where the model can be further improved. In Figure 6, the red line represents the model's predicted values, while the blue line represents the actual data. By comparing the two lines, we can visually observe how well the model's predictions align with the actual data points. A close alignment between the red and blue lines indicates a good prediction performance, while significant deviations suggest room for improvement. This plot allows us to assess the accuracy and effectiveness of the model's predictions in a visual manner. The Root Mean Square Error (RMSE) is a commonly used metric to evaluate the accuracy of predictions. It is calculated as the square root of the mean of the squared differences between the predicted values and the actual values. In the context of regression analysis, the residuals represent the differences between the observed data points and the predicted values. The RMSE provides a measure of how much the predicted values deviate, on average, from the actual values. By calculating the RMSE, we can quantify the level of accuracy of the prediction and assess the performance of the model in a numerical manner. The RMSE (Root Mean Square Error) is a metric used to assess the accuracy of predictions. It is calculated as the standard deviation of the residuals, which measures the differences between the predicted values and the actual values. In this project, the RMSE function from the Sklearn library was utilized to calculate the RMSE score. A lower RMSE score indicates better prediction accuracy, with values between 0.2 and 0.5 generally considered good. Evaluating the RMSE score allows for a mathematical assessment of the model's performance and facilitates comparisons with other models or benchmarks.

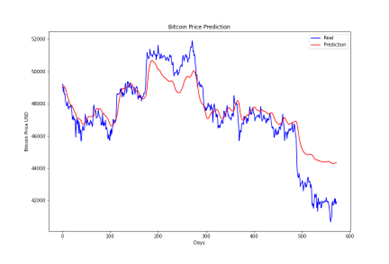
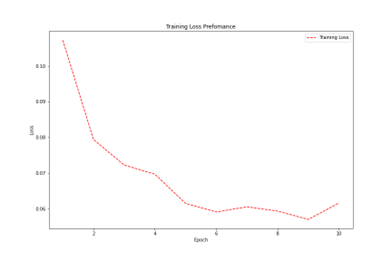


Figure 5: Training Loss Plot Example Figure 6: Real Prediction vs Price Prediction Plot Example.

Discussion

To assess the performance of the model, testing was conducted using various time frames for the data. This involved feeding the model with different time intervals of data to observe how it performed and predicted the outcomes. By testing the model with different time frames, it was possible to evaluate its ability to handle different temporal patterns and capture relevant trends and patterns in the data. This approach provided insights into how the model performed under different conditions and helped assess its generalization and predictive capabilities across varying time scales. The evaluation of the model began with testing it on small time-framed data, specifically 1-minute time frames, which represent the raw and original data. Surprisingly, the model's prediction on this time frame was nearly perfect, exhibiting a high level of accuracy. This can be observed in Figure 7, where the predicted values closely align with the actual data points. The model's ability to accurately capture the patterns and trends in the raw data is a positive indication of its performance. The visual comparison between the predicted values (represented by the red line) and the real data (blue line) indeed shows a remarkable similarity. The prediction closely aligns with the actual data points, indicating a strong performance of the model. Furthermore, the Root Mean Square Error (RMSE) score of 0.353 further supports the model's accuracy. With an RMSE score close to 0, the model's predictions are very close to the actual values, indicating a high level of precision in its forecasting capabilities. Upon investigating the unusually accurate predictions of the model, you discovered that the input data you were using, which consisted of 1-minute time-framed data, had very minimal variations between consecutive data points. This proximity between data points could have led to the model learning the patterns and trends in the data with a high level of precision. The proximity of the numbers in the 1-minute time-framed data resulted in very small price differences between consecutive data points. As a result, the model was able to capture these subtle patterns and make accurate predictions. This was evident in the training loss plot, where the training loss dropped rapidly in the initial epochs and then decreased gradually over subsequent epochs.

However, it's important to note that such accuracy may not be realistic or applicable to real-world scenarios with larger time frames or more volatile price movements.

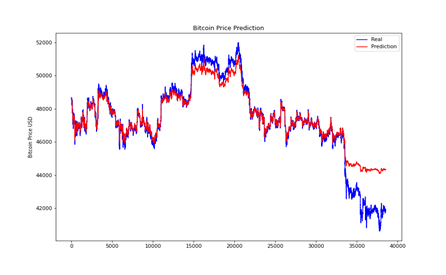
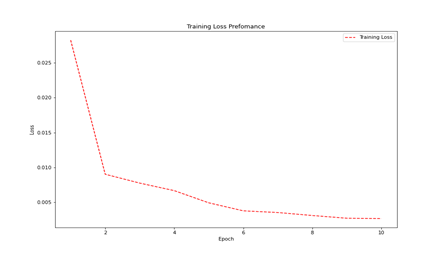
 

Figure 7: 1-min Time-Framed Real Price vs Prediction Plot. Figure 8: 1-Min Time-Frame Training Loss Plot

After obtaining nearly perfect results with the 1-minute time-framed data, I proceeded to test the model on larger time-framed data. I converted the 1-minute data into 15-minute time-framed data and evaluated the model's performance on this new dataset. The results obtained from testing the model on the 15-minute time-framed data were also promising. The predicted price closely followed the real price over time, as depicted in Figure 9. The RMSE value of 0.439 indicated that the model's predictions were accurate, although slightly less precise compared to the 1-minute time-framed data. This outcome can be attributed to the fact that the data in the 15-minute time-framed dataset still exhibited relatively small price differences. As a result, the model was able to capture and predict these subtle variations effectively, leading to accurate predictions. The training loss for the 15-minute time-framed data showed a rapid decrease in the first 3 epochs, indicating that the model quickly adapted to the data patterns. After that, the training loss stabilized, suggesting that the model reached a relatively optimal state and further epochs did not significantly improve its performance. This faster drop in training loss compared to the 1-minute time-framed data suggests that the model was able to learn and capture the patterns in the 15-minute data more efficiently.

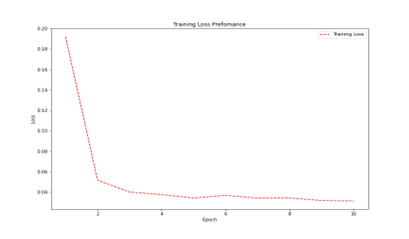


Figure 9: 15-min Time-Framed Real Price vs Prediction Plot. Figure 10: 15-Min Time-Frame Training Loss Plot

The results of testing the model on 60-minute time-framed data were not as favorable as the previous test runs. The prediction line deviated more from the real price line, indicating lower accuracy. The RMSE score increased, indicating higher prediction errors. The significant increase in the time frame made it more challenging for the model to capture the fine-grained patterns and fluctuations present in the data. As a result, the model's prediction performance suffered. The training loss plot showed a gradual decrease in training loss over the epochs, suggesting the model was learning, but the overall predictive ability on the longer time frame was not as satisfactory. This emphasizes the importance of considering the appropriate time frame for training the model based on the characteristics of the data. The significant increase in the time frame introduced a notable difference in the data. This change should have resulted in distinct predictions from the model. The larger time frame encompassed more substantial price variations between consecutive data points, which could have influenced the model's ability to accurately capture and predict the patterns in the data. As a result, the prediction line deviated further from the actual price line, indicating a reduced level of precision. The broader price differences between the data points likely affected the model's capability to capture fine-grained details and short-term fluctuations in the price data. As depicted in figure 11, the model encountered challenges in making accurate predictions for the 60-minute time-framed data. Visually, there was a noticeable difference between the predicted line and the actual price line, indicating a decrease in prediction accuracy compared to the previous runs. The RMSE score of 0.574, while not exceptionally high, suggests that the model's predictions deviated to a moderate extent from the actual prices. Overall, the model's performance in this particular run was not as good as in the previous tests with smaller time frames. I believe the prediction accuracy was compromised in this case due to the larger difference in the data that was fed into the model. The increased magnitude of price differences made it more challenging for the model to make accurate predictions. As shown in figure 12, the training loss exhibited a smoother drop compared to the previous runs. The decrease in training loss was not as significant, indicating that the model struggled to fit the larger time-framed data. The training loss was distributed more evenly across the 10 epochs, suggesting a slower convergence of the model's training.

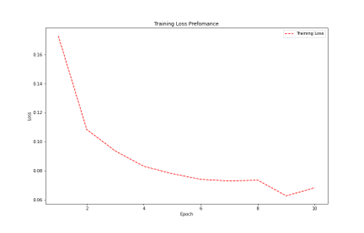
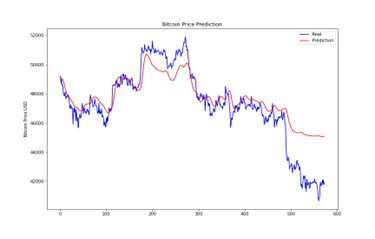


Figure 11: 60-min Time-Framed Real Price vs Prediction Plot. Figure 12: 60-Min Time-Frame Training Loss Plot

In the next run, I transformed the 1-minute time-framed data into 120-minute time-framed data, doubling the time frame compared to the previous run. This led to more significant differences in prices within the dataset. However, the results were not as favorable, as shown in figure 13. The prediction line deviated more from the real data, indicating a lower accuracy. The RMSE score for this run was 0.683, suggesting a higher level of error in the predictions. Visually, it is evident that the model struggled to accurately predict the data in the case of 120-minute time-framed data. The red prediction line deviates noticeably from the real data, indicating a lower level of accuracy. This is further confirmed by the RMSE score of 0.767, which indicates a relatively high level of error in the predictions. The challenge in this case was the significant and noticeable differences in prices within the 120-minute time-framed data. This made it difficult for the model to capture and predict the patterns effectively. As a result, the predictions were less accurate. Regarding the training loss, it exhibited a relatively linear decrease throughout the 10 epochs, as shown in figure 14. This suggests that the model gradually improved its fit to the training data, although it still struggled to accurately predict the test data.

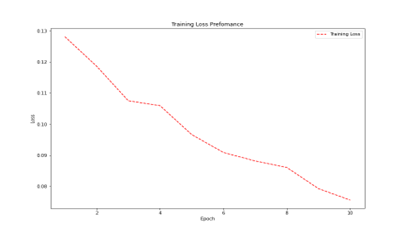
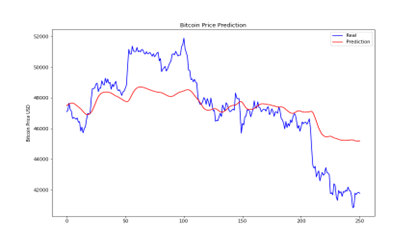


Figure 13: 120-min Time-Framed Real Price vs Prediction Plot. Figure 14: 120-Min Time-Frame Training Loss Plot

After conducting multiple tests as described above, I began to gain insights into the reasons behind the output of the LSTM model. Based on my analysis, I reached the conclusion that when providing the LSTM model with low time-framed data, the differences between the data points are minimal. Consequently, the model produces highly accurate predictions. As the time frames were increased, I observed a gradual decline in the model's performance, resulting in less accurate predictions. This deterioration can be attributed to the larger differences in prices within the higher time-framed data, which posed a challenge for the model in making accurate predictions. To confirm my theory, I conducted a final test using even larger time-framed data. I expected that the model's predictions would be even less accurate due to the increased differences in prices. For the last run, I converted the 1-min timeframe data to 180-min time-framed data. The 180-min time-framed data has a bigger difference in price than the last 4 predictions which means it should produce worse results. After testing the model on different time-framed data, my theory regarding the impact of price differences on prediction accuracy was confirmed. In the last run with 180-min time-framed data, the prediction was indeed poor, as evident in Figure 15. The visual misalignment between the prediction line and the real data indicates the model's inaccuracy. This was further supported by the high RMSE score of 0.886, signifying a significant deviation from the actual prices. Additionally, the training loss in this run dropped more slowly compared to previous runs, as depicted in Figure 16. It took more time and epochs for the training loss to decrease, indicating the model's struggle to fit the data. Overall, this final test validated my theory that as the price difference increases, the LSTM model's predictions become less accurate.

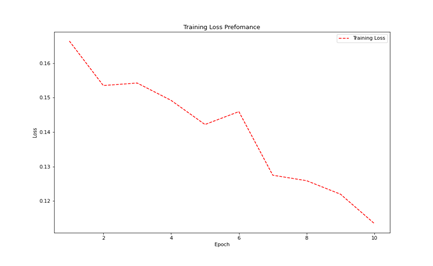
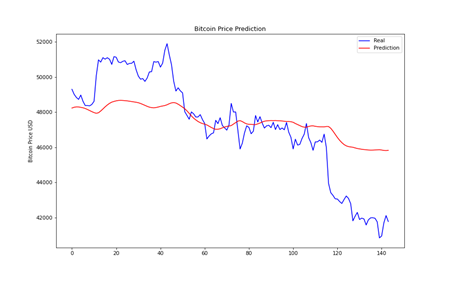


Figure 15: 180-min Time-Framed Real Price vs Prediction Plot Figure 16: 180-Min Time-Frame Training Loss Plot

Conclusion

In summary, I have successfully developed an LSTM model to analyze Bitcoin's historical data. The project aimed to gather Bitcoin prices from exchanges and convert the data into different time frames for testing the LSTM model's performance. Through evaluating the model, I observed that it performed exceptionally well when provided with low time-framed data, which can be attributed to the small price differences and easier predictability. However, when the model was given larger time-framed data with significant price variations, it struggled to make accurate predictions. This confirmed my theory that the model's performance is influenced by the magnitude of price differences.

Future work

If I were to work on this project in the future, there are several improvements and implementations I would consider. Firstly, I would incorporate more statistical tools to analyze the predictions numerically. This would provide a deeper understanding of the LSTM model's behavior and help identify its flaws. By utilizing additional statistical tools, I would be able to adjust the hyperparameters more effectively and improve the accuracy of the predictions, especially in higher time frames. Another aspect I would focus on is obtaining live data from exchanges to predict Bitcoin prices in real-time. This would allow me to assess the model's performance in a real-life scenario and monitor its accuracy throughout the day. By continuously feeding the model live data, I could evaluate its capabilities and assess its effectiveness for real-time predictions. Expanding the scope of the project to include other cryptocurrencies, such as Ethereum, would also be beneficial. Testing the LSTM model on the same time frame as Bitcoin and analyzing any correlation between the two would provide valuable insights into the model's behavior. This comparative analysis would help me refine the hyperparameters and improve the model's performance. Furthermore, I would incorporate measures of volatility into the analysis. High volatility often leads to significant price changes, which can impact the model's performance. Identifying areas in the data with high volatility and examining the model's response in comparison to periods of lower volatility would provide insights into the model's effectiveness. Assessing the error percentage in different volatility regimes would enable me to determine the best conditions for using the model. For instance, if the model performs poorly during high volatility, it might be wise to avoid using it for predicting trending markets. To gain a better understanding of the LSTM model's performance in relation to other artificial intelligence models, I would compare its results against similar models. This comparative analysis would offer insights into the strengths and weaknesses of the LSTM model and provide a broader perspective on its predictive capabilities. Additionally, I would experiment with the architecture of the LSTM model by adding or removing layers. Assessing the impact of these modifications on the model's performance would help me explore alternative configurations and potentially improve the accuracy of the predictions. Overall, these proposed improvements and implementations would enhance my understanding of the LSTM model, refine the hyperparameters, evaluate its performance in real-time scenarios, explore correlations with other cryptocurrencies, account for volatility, compare it with alternative models, and experiment with different architectural configurations. By undertaking these steps, I aim to enhance the accuracy and reliability of the LSTM model for predicting cryptocurrency prices.

Reflections

As a reflection on the project, I consider it to be a success. However, there were some challenges and obstacles that I encountered throughout the process. In the initial stages of the project, I faced a lack of motivation due to the overwhelming tasks ahead of me. To overcome this, I found the weekly meetings with my supervisor to be highly beneficial. These meetings provided guidance, support, and regular feedback, which helped me stay on track and improve my work on a weekly basis. One of the major difficulties I faced during the project was getting the LSTM model to run successfully. It took me some time to build the model and integrate it with the rest of the functions without encountering any issues. Understanding the behavior of the LSTM model was particularly challenging in the beginning. The model produced results that I struggled to comprehend, and it required numerous trial runs before I started to gain a better understanding of its behavior. Adjusting the hyperparameters was another aspect that posed initial challenges. It took considerable time and effort to grasp how different hyperparameters influenced the model's performance and prediction outcomes. One of the main issues I encountered throughout the project was the lengthy runtime of the LSTM model. Each run of the model on small time frames took approximately 20 minutes to train on the data and generate results. This prolonged runtime made it difficult for me to work with the code in a continuous manner. I had to run the model, pause, check the results later, and then rerun the model, leading to significant time delays in understanding the model's behavior. I attempted to address this problem by seeking permission from my supervisor to access the university's GPU, hoping for faster execution. However, it turned out that the runtime was even longer on the university servers, rendering the GPU usage ineffective. As a result, I continued to rely on my laptop for running the model. Despite the challenges, the project progressed smoothly overall, thanks to the guidance and support from my supervisor. Their assistance proved invaluable in navigating the project and overcoming obstacles along the way.

References

1. The article "LSTM | Introduction to LSTM | Long Short Term Memory" from Analytics Vidhya provides an overview and introduction to LSTM models. It can be found at:[Learn About Long Short-Term Memory (LSTM) Algorithms](https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/).
2. Arowolo, M.O., Ayegba, P., Yusuff, S.R., and Misra, S. (2022). "A Prediction Model for Bitcoin Cryptocurrency Prices." This paper discusses a prediction model for Bitcoin prices and may offer insights relevant to the project.
3. The article "JP Morgan predicts long-term Bitcoin price may reach $150,000" on cnbctv18.com explores the long-term price prediction for Bitcoin according to JP Morgan. The article can be accessed at: [Jp Morgan Predicts Long-Term Bitcoin Price May Reach $150,000](https://www.cnbctv18.com/cryptocurrency/jp-morgan-predicts-long-term-bitcoin-price-may-reach-150000-12452972.htm).
4. Donges, N. (2019). "Recurrent neural networks 101: Understanding the basics of RNNs and LSTM." This article provides a basic understanding of recurrent neural networks, including LSTM. It can be found on Built In's website:[A Guide to Recurrent Neural Networks: Understanding RNN and LSTM Networks](https://builtin.com/data-science/recurrent-neural-networks-and-lstm).
5. The blog post "A Gentle Introduction to Long Short-Term Memory Networks by the Experts" by Jason Brownlee provides an introduction to LSTM networks. It is available on the Machine Learning Mastery website:[A Gentle Introduction to Long Short-Term Memory Networks by the Experts - MachineLearningMastery.com](https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/).
6. Investing News Network's article "Bitcoin Price History | INN" provides a historical overview of Bitcoin's price. The article can be accessed at: <https://investingnews.com/daily/tech-investing/blockchain-investing/bitcoin-price-history/>
7. Li, Y., Zheng, Z., and Dai, H.-N. (2020). "Enhancing Bitcoin Price Fluctuation Prediction Using Attentive LSTM and Embedding Network." This research paper discusses the use of LSTM and embedding networks to improve Bitcoin price prediction. It is published in the journal Applied Sciences, volume 10, issue 14.
8. The report "Global $22.6Bn AI in Fintech Market Outlook, 2020-2025" by Markets and Markets provides insights into the AI market in the fintech industry. It includes profiles of various companies operating in this space. The report can be accessed at: : https://www.prnewswire.com/news-releases/global-22-6bn-ai-in-fintech-market-outlook-2020-2025-featuring-profiles-of-ibm-intel-complyadvantagecom-narrative-science-amazon-web-services-ipsoft-and-more-301083938.html [Accessed 24 Apr. 2022].
9. McNally, S., Roche, J., and Caton, S. (2018). "Predicting the Price of Bitcoin Using Machine Learning." This research paper explores the use of machine learning techniques for predicting Bitcoin prices. It was presented at the 2018 Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP).
10. The article "Bank Of America: Bitcoin And Cryptocurrencies Are Too Large To Ignore" on Nasdaq.com discusses Bank of America's perspective on the significance of Bitcoin and cryptocurrencies. The article can be found at: : https://www.nasdaq.com/articles/bank-of-america%3A-bitcoin-and-cryptocurrencies-are-too-large-to-ignore-2021-10-06 [Accessed 24 Apr. 2022].
11. Olah, C. (2015). "Understanding LSTM Networks" is a blog post that provides a comprehensive explanation of LSTM networks. It is available on colah's blog: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.
12. The tutorial "Recurrent Neural Network Tutorial" on Simplilearn.com offers insights into recurrent neural networks, including LSTM. The tutorial can be accessed at: [Understanding LSTM Networks -- colah's blog](https://colah.github.io/posts/2015-08-Understanding-LSTMs/).
13. Statista provides data on the number of cryptocurrencies from 2013 to 2021. The information can be found at: [Understanding LSTM Networks -- colah's blog](https://colah.github.io/posts/2015-08-Understanding-LSTMs/).
14. Statista offers data on the market value of cryptocurrencies from 2013 to 2018. The statistics can be accessed at:[Understanding LSTM Networks -- colah's blog](https://colah.github.io/posts/2015-08-Understanding-LSTMs/).
15. The article "Bitcoin: The First Cryptocurrency Is As Volatile As It Is Valuable" on Time.com discusses the volatility and history of Bitcoin. It is available at: [Understanding LSTM Networks -- colah's blog](https://colah.github.io/posts/2015-08-Understanding-LSTMs/).
16. Wei, W., Zhang, Q., and Liu, L. (n.d.). "Bitcoin Transaction Forecasting with Deep Network Representation Learning" is a research paper that explores Bitcoin transaction forecasting using deep network representation learning. It can be found: https://arxiv.org/pdf/2007.07993.pdf [Accessed 24 Apr. 2022].
17. Wu, C.-H., Lu, C.-C., Ma, Y.-F. and Lu, R.-S. (2018). A New Forecasting Framework for Bitcoin Price with LSTM. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/abstract/document/8637486 [Accessed 10 Dec. 2021].