A PROJECT REPORT ON

**BITCOIN PRICE PREDICTIONS**

***Submitted by***

**SAURABH KUMAR**

**(UID: 21MCI1231)**

## UNIVERSITY INSTITUTE OF COMPUTING

###### In partial fulfillment for the requirement of the award of the degree

**Master in Computer Application**

**in**

**Artificial Intelligence and Machine Learning**



**Chandigarh University**

**Session: 2021-2023**

**BITCOIN PRICE PREDICTIONS**



**BONAFIDE CERTIFICATE**

Certified that this project report “**BITCOIN PRICE PREDICTIONS**”

is the bonafide work of “**SAURABH KUMAR**” who carried out the project work under my/our supervision.

Head of the Department Supervisor

Mr. Abdullah Dr. Swaroop Kaur

Submitted for the project viva-voce examination held on\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

It is great pleasure for us to undertake this project. We feel highly doing this project entitled – **“Bitcoin Price Predictions”.** It is my pleasure to be indebted to various people who directly or indirectly Contributed in the development of this work and who influenced our thinking, behaviour and acts during the course of study.

I take this opportunity to express my profound gratitude and deep regards to my guide for her exemplary guidance, monitoring and constant encouragement throughout this. The blessing, help and guidance given by her time to time shall

carry me a long way in the journey of life on which I am about to embark. I also take this opportunity to express a deep sense of gratitude to University for cordial support, valuable information and guidance, which helped me in completing this task through various stages.

I express our sincere gratitude to **Prof.(Dr.) Swaroop Kaur** for her support , cooperation , and motivation provided to us during the training for constant inspiration presence and blessings

**Name: SAURABH KUMAR (21MCI1231)**

**ABSTRACT**

After the boom and bust of cryptocurrencies’ prices in recent years, [Bitcoin](https://www.sciencedirect.com/topics/mathematics/bitcoin) has been increasingly regarded as an investment asset. Because of its highly volatile nature, there is a need for good predictions on which to base investment decisions. Although existing studies have leveraged machine learning for more accurate Bitcoin price prediction, few have focused on the feasibility of applying different modeling techniques to samples with different data structures and dimensional features. To predict Bitcoin price at different frequencies using machine learning techniques, we first classify Bitcoin price by daily price and high-frequency price.

A set of high-dimension features including [property](https://www.sciencedirect.com/topics/mathematics/sigma-property) and network, trading and market, attention and gold spot price are used for Bitcoin daily price prediction, while the basic trading features acquired from a cryptocurrency exchange are used for 5-minute interval price prediction. The purpose of this project is to predict Bitcoin prices using various machine learning techniques. Due to its high volatility attribute, accurate price prediction is the need of the hour for sound investment decision-making. At the offset, this study categorizes Bitcoin price by daily and high-frequency price. This work on Bitcoin price prediction recognizes the significance of sample dimensions in machine learning algorithms.

A year ago, the world witnessed the most devastating pandemic in human history which led to extreme economic and social disruption. The COVID-19 has affected almost every sphere of a man’s life and has bought uncertainty. One of the sectors which became highly volatile during this period was the Bitcoin, people lost hundreds of millions during the market crash and subsequent events. This event again led to widening the gap of fear among the people willing to invest their money in the bitcoin.

To resolve this problem and fill the gap of fear we’ve tried to predict the bitcoin action with greater accuracy as compared to the other models proposed previously. The prediction will be based on both technical and fundamental analyses of the stocks. We’ll be implementing this concept with help of a hybrid model of frequency decomposition, deep learning, and sentimental analysis. The datasets for the technical analysis will be acquired from stock exchanges and for fundamental analysis, we’ll be referring to social networking sites and online news portals.

**ABSTRACT (HINDI)**

हाल के वर्षों में क्रिप्टोकरेंसी की कीमतों में उछाल और उछाल के बाद, बिटकॉइन को तेजी से एक निवेश संपत्ति के रूप में माना जाने लगा है। इसकी अत्यधिक अस्थिर प्रकृति के कारण, अच्छी भविष्यवाणियों की आवश्यकता है, जिस पर निवेश निर्णयों को आधार बनाया जा सके। हालांकि मौजूदा अध्ययनों ने अधिक सटीक बिटकॉइन मूल्य पूर्वानुमान के लिए मशीन लर्निंग का लाभ उठाया है, कुछ ने विभिन्न डेटा संरचनाओं और आयामी विशेषताओं वाले नमूनों के लिए विभिन्न मॉडलिंग तकनीकों को लागू करने की व्यवहार्यता पर ध्यान केंद्रित किया है। मशीन लर्निंग तकनीकों का उपयोग करके विभिन्न आवृत्तियों पर बिटकॉइन की कीमत का अनुमान लगाने के लिए, हम पहले बिटकॉइन की कीमत को दैनिक मूल्य और उच्च-आवृत्ति मूल्य के आधार पर वर्गीकृत करते हैं।

बिटकॉइन दैनिक मूल्य भविष्यवाणी के लिए संपत्ति और नेटवर्क, व्यापार और बाजार, ध्यान और सोने की हाजिर कीमत सहित उच्च-आयामी सुविधाओं का एक सेट उपयोग किया जाता है, जबकि क्रिप्टोकुरेंसी एक्सचेंज से प्राप्त बुनियादी व्यापारिक सुविधाओं का उपयोग 5 मिनट के अंतराल मूल्य पूर्वानुमान के लिए किया जाता है। इस परियोजना का उद्देश्य विभिन्न मशीन लर्निंग तकनीकों का उपयोग करके बिटकॉइन की कीमतों का अनुमान लगाना है। इसकी उच्च अस्थिरता विशेषता के कारण, सही निवेश निर्णय लेने के लिए सटीक मूल्य पूर्वानुमान समय की आवश्यकता है। ऑफसेट पर, यह अध्ययन दैनिक और उच्च आवृत्ति मूल्य के आधार पर बिटकॉइन की कीमतों को वर्गीकृत करता है। बिटकॉइन मूल्य पूर्वानुमान पर यह कार्य मशीन लर्निंग एल्गोरिदम में नमूना आयामों के महत्व को पहचानता है।

एक साल पहले, दुनिया ने मानव इतिहास में सबसे विनाशकारी महामारी देखी, जिसके कारण अत्यधिक आर्थिक और सामाजिक व्यवधान उत्पन्न हुआ। COVID-19 ने मनुष्य के जीवन के लगभग हर क्षेत्र को प्रभावित किया है और अनिश्चितता खरीदी है। इस अवधि के दौरान जो क्षेत्र अत्यधिक अस्थिर हो गए, उनमें से एक बिटकॉइन था, बाजार दुर्घटना और उसके बाद की घटनाओं के दौरान लोगों को करोड़ों का नुकसान हुआ। इस घटना ने फिर से बिटकॉइन में अपना पैसा निवेश करने के इच्छुक लोगों के बीच भय की खाई को चौड़ा कर दिया।

इस समस्या को हल करने और डर के अंतर को भरने के लिए हमने पहले प्रस्तावित अन्य मॉडलों की तुलना में बिटकॉइन कार्रवाई को अधिक सटीकता के साथ भविष्यवाणी करने का प्रयास किया है। भविष्यवाणी शेयरों के तकनीकी और मौलिक विश्लेषण दोनों पर आधारित होगी। हम इस अवधारणा को आवृत्ति अपघटन, गहन शिक्षण और भावुक विश्लेषण के एक संकर मॉडल की मदद से लागू करेंगे। तकनीकी विश्लेषण के लिए डेटासेट स्टॉक एक्सचेंजों से प्राप्त किए जाएंगे और मौलिक विश्लेषण के लिए, हम सोशल नेटवर्किंग साइटों और ऑनलाइन समाचार पोर्टलों का जिक्र करेंगे।

**Contents**

[List of Figures ix](#_Toc105519811)

[1. Chapter 1 - Introduction 1](#_Toc105519812)

[1.1 Bitcoin 3](#_Toc105519813)

[1.2 Machine Learning 5](#_Toc105519814)

[1.3 Classification 6](#_Toc105519815)

[1.4 Unsupervised Algorithms 6](#_Toc105519816)

[2. Chapter 2 - Project Specification 8](#_Toc105519817)

[2.1 Purpose 8](#_Toc105519818)

[2.2 Research Variables 8](#_Toc105519819)

[3. Chapter 3 - Approach of The Project 10](#_Toc105519820)

[3.1 What Is Data Mining? 10](#_Toc105519821)

[3.2 What Is Descriptive Approach In Data Mining? 11](#_Toc105519823)

[3.3 What Is Predictive Approach Data Mining? 12](#_Toc105519824)

[3.4 Key Differences Between Descriptive And Predictive Data Mining 13](#_Toc105519825)

[4. Chapter 4 - Problem Statement 15](#_Toc105519828)

[5. Chapter 5 - Requirements: 15](#_Toc105519829)

[5.1 Materials Required: 15](#_Toc105519830)

[5.2 Functional Partitioning of Project: 16](#_Toc105519831)

[6. Chapter 6 - Different type of models that we used in our project: 16](#_Toc105519832)

[6.1 LSTM 19](#_Toc105519833)

[6.2 Dense LSTM 19](#_Toc105519834)

[6.3 RNN 20](#_Toc105519835)

[6.4 ARIMA 21](#_Toc105519836)

[6.5 Kalman Filtering 22](#_Toc105519837)

[6.6 Exponential Smoothing 23](#_Toc105519838)

[6.7 Seasonal Naïve 24](#_Toc105519839)

[6.9 Theta 26](#_Toc105519845)

[7. Chapter 7 - Comparing All Algorithms: 28](#_Toc105519846)

[7.1 Import Dataset: 28](#_Toc105519847)

[7.2 Dataset: 29](#_Toc105519848)

[7.3 LSTM 31](#_Toc105519849)

[7.4 Dense LSTM 32](#_Toc105519850)

[7.5 RNN 33](#_Toc105519851)

[7.6 ARIMA 34](#_Toc105519852)

[7.7 Kalman Filtering 34](#_Toc105519853)

[7.8 Exponential Smoothing 35](#_Toc105519854)

[7.9 Seasonal Naïve 35](#_Toc105519855)

[7.10 Naïve Last 36](#_Toc105519856)

[7.11 Theta 37](#_Toc105519857)

[7.12 Comparing All Algorithms: 37](#_Toc105519858)

[8. Chapter 8 - Conclusion 40](#_Toc105519859)

[9. Chapter 9 - Future Scope 41](#_Toc105519860)

[10. Chapter 10 - References 41](#_Toc105519861)

[Appendix 1: 42](#_Toc105519862)

# List of Figures

|  |  |  |
| --- | --- | --- |
| S. No. | Description | Pg. No. |
| 1 | *Block Diagram for price prediction using different algorithms.* | *19* |
| 2 | *Block Diagram for working of algorithms.* | *19* |
| 3 | *Comparison of all the models used in the work.* | *20* |
| 4 | *Libraries used in the project.* | *28* |
| 5 | *Importing the dataset.* | *28* |
| 6 | *Structure of the dataset.* | *29* |
| 7 | *Filling the missing values in the dataset.* | *29* |
| 8 | *Information about total missing values in the dataset.* | *29* |
| 9 | *Visualization of the dataset.* | *30* |
| 10 | *Algorithm for LSTM (Long Short-Term Memory).* | *31* |
| 11 | *Prediction of bitcoin price with LSTM deep learning algorithm.* | *31* |
| 12 | *Algorithm for Dense LSTM (Dense Long Short-Term Memory).* | *32* |
| 13 | *Prediction of bitcoin price with Dense LSTM deep learning algorithm.* | *32* |
| 14 | *Algorithm for RNN (Recurrent Neural Network).* | *33* |
| 15 | *Prediction of bitcoin price with RNN deep learning algorithm.* | *33* |
| 16 | *sMAPE visualization of prediction using ARIMA deep learning model.* | *34* |
| 17 | *sMAPE visualization of prediction using KALMAN FILTERING.* | *34* |
| 18 | *sMAPE visualization of prediction using EXPONENTIAL SMOOTHING.* | *35* |
| 19 | *sMAPE visualization of prediction using Seasonal Naïve.* | *35* |
| 20 | *sMAPE visualization of prediction using Naïve Last.* | *36* |
| 21 | *sMAPE visualization of prediction using Theta.* | *36* |
| 22 | *Functions defined to predict and plot the bitcoin price in the project.* | *37* |
| 23 | *Comparing the last 6 models by their sMAPE value and the time elapsed by each model to predict the bitcoin price.* | *38* |

# 1. Introduction

Cryptocurrencies are a digital way of money in which all transactions are held electronically. It is a soft currency which doesn’t exist in the form of hard notes physically. Here, we are emphasizing the difference of fiat currency which is decentralized that without any third-party intervention all virtual currency users can get the services. However, getting services of these cryptocurrencies impacts on international relations and trade, due to its high price volatility.

There are several virtual currencies such as bitcoin, ripple, Ethereum, Ethereum classic, lite coin, etc. As of now the total percentage of people that are investing in the Indian bitcoin is barely 4% of the total population. We are lagging far behind in this sector when compared to the U.S and U.K which are having more than 50% and 33% respectively of their population investing directly in the bitcoin. This is due to the fear or mindset of people that they’ll end up losing their savings in the market. So, the objective of this research is to provide a more reliable and accurate system for predicting the pricing activity of the bitcoin. Offering people something that they can trust or a piece of technology that they can use to start their journey of investment.

Depending on the characteristics of financial markets and the aim of valuation, this study proposes a novel basic structure which combines Convolution Neural Network (CNN) and Long–Short–Term Memory Neural Network (LSTMN) to produce a much more precise stock price projection (LSTM). Stock Sequence Array Convolutional L.S.T.M is the name of the new approach (SACLSTM). It generates a series of historical data and following indications, feeds the array into the CNN framework, extracts certain feature vectors via the convolutional layer and the pooling layer, feeds the vector into the L.S.T.M, and uses ten stocks from various stock exchanges as experimental data. By evaluating the prediction performance of the proposed method in this research with that of other approaches, it is clear that the proposed method outperforms them all, the proposed algorithm in this article outperforms them.

Bitcoin is the worlds’ most valuable cryptocurrency and is traded on over 40 exchanges worldwide accepting over 30 different currencies. It has a current market capitalization of 9 billion USD and sees over 250,000 transactions taking place per day. As a currency, Bitcoin offers a novel opportunity for price prediction due its relatively young age and resulting volatility, which is far greater than that of fiat currencies. It is also unique in relation to traditional fiat currencies in terms of its open nature; no complete data exists regarding cash transactions or money in circulation for fiat currencies.

Prediction of mature financial markets such as the stock market has been researched at length Bitcoin presents an interesting parallel to this as it is a time series prediction problem in a market still in its transient stage. Traditional time series prediction methods such as Holt-Winters exponential smoothing models rely on linear assumptions and require data that can be broken down into trend, seasonal and noise to be effective. This type of methodology is more suitable for a task such as forecasting sales where seasonal effects are present. Due to the lack of seasonality in the Bitcoin market and its high volatility, these methods are not very effective for this task. Given the complexity of the task, deep learning makes for an interesting technological solution based on its performance in similar areas.

The recurrent neural network (RNN) and the long short-term memory (LSTM) are favored over the traditional multilayer perceptron (MLP) due to the temporal nature of Bitcoin data.

The aim of this paper is to investigate with what accuracy the price of Bitcoin can be predicted using machine learning and compare parallelization methods executed on multi-core and GPU environments. This paper contributes in then following manner: of approximately 653 papers published on Bitcoin only 7 (at the time of writing) are related to machine learning for prediction. To facilitate a comparison to more traditional approaches in financial forecasting, an ARIMA time series model is also developed for performance comparison purposes with the neural network models.

The independent variable for this study is the closing price of Bitcoin in USD taken from the CoinDesk Bitcoin Price Index. Rather than focusing on one specific exchange, we take the average price from five major Bitcoin exchanges: Bitstamp, Bitfinex, Coinbase, OkCoin and itBit. If we were to implement trades based on the signals it would be beneficial to focus on just one exchange.

To assess the performance of models, we use the root mean squared error (RMSE) of the closing price and further encode the predicted price into categorical variable reflecting: price up, down or no change. This latter step allows for additional performance metrics that would be useful to a trader in the formation of a trading strategy: classification accuracy, specificity, sensitivity and precision.

The dependent variables for this paper come from the CoinDesk website, and Blockchain.info. In addition to the closing price, the opening price, daily high and daily low are also included as well as Blockchain data, i.e., the mining difficulty and hash rate. The features which have been engineered (considered as technical analysis indicators) include two simple moving averages (SMA) and a de-noised closing price.

## 1.1 Bitcoin

In 2008, Satoshi Nakamoto created a digital cryptocurrency and exchange system known as a Bitcoin (Madan et al.; 2015; Ron and Shamir; 2013) and came into existence and recognized for its quality in 2012 and become a popular cryptocurrency (Economist; 2013). It depends on decentralized, shared, associate with the arrangement of an assets and trade organization did by the people from the system. On November 17th, 2017 one Bitcoin (Bitstamp) equals to 7831.02 US Dollar and can be gotten through exchanging for items, services, or by mining, or different monetary forms suggested by Kaminski (2014). In the Bitcoin system, cryptography is utilized to exchange the cash and to control the creation, subsequently, the Blockchain i.e., the common open system is utilized to store the computerized marked messages (Kaminski; 2014). Grocer (2013) suggested that the utilization of Bitcoin has been seen appears to be financially little, in light of dangerous cost and high volatilities. As indicated by Economist (2013), new businesses are supported by financial specialists which are related to Bitcoin and has been known as a unit of account by the German finance ministry and on 18th of November, an American Senate Council informed by senior authorities that Bitcoin has legal uses. But there may have also been many cases of Bitcoin robbery, Workplaces that changes over Bitcoin to various fiscal structures have closed. For instance, In October, FBI, America shut the Silk Street which acts as an online discussion where Bitcoin are traded for illegal products and services (Economist; 2013). Also, Bitcoin cost changes profoundly noted $230 in April 2013 and falls distinctly $70 following three months and expanded in November up to $600 as per Economist (2013).

The author has found the relationship between the Bitcoin price, the tweets and Google trends view from previous research which was done by Matta et al. (2015a) and has been utilized as evidence that can be used as predictors. In addition to these, same methodology was implemented by Matta et al. (2015b) for another analysis in which trading volume prediction are used alternatively to predict the price of the Bitcoin and achieved solid interrelationship between Google trend view and price of the Bitcoin. The dataset utilized for this exploration was taken from one year. Furthermore, to discover same outcomes a few researchers utilized Wavelets for their examination work (Kristoufek; 2015; Delfin-Vidal and Romero-Mel´endez; 2016). Likewise, the author used in this research ten different cryptocurrencies closing price to predict the accuracy and the Bitcoin’s price.

Despite the fact, huge information is accessible identifying with Bitcoin and its system, the creator contends that not all researcher used this data adequately and subsequently, it may not be reflected in the cost. The goal of this paper is to gain benefit of this theory over several data mining methods.

Bitcoin is exchanged on more than 40 trades overall tolerating more than 30 distinct monetary forms and has a present market capitalization of 9 billion dollars and its interest has developed significantly with more than 250,000 exchanges now occurring every day. Study shows that $100,000 euros had been stole from a 36-year-old person when he signed on to the unsecured system to check the Bitcoin price proposed by Nick Whigham (2017). But the situation remains unclear by the police that the Bitcoin was already hacked before the victim logged on the network. Moreover, Mt Gox, Japanese based Bitcoin exchange was hacked in January 2014 which acts as a largest Bitcoin intermediary and regarded as the leading Bitcoin exchange before hackers grabbed 85,000 BTC. Bitcoin and Ethereum largest exchange in South Korea were hacked by hackers in June 2017 as reported by the local news and the customer claimed to have lost of 1.2 billion. According to CNBC (2017), first time Bitcoin hit new record $8000 as per the information received from industry website CoinDesk.

On November 12, the price of the Bitcoin was increased more than 47 percent. Moreover, Bitcoin, world leading cryptocurrency exceeded another record that is $8200 on Monday, 20 November 2017. November has been a greatly unstable month for Bitcoin and recorded cryptocurrency price fell to $5500 by Saxena (2017b). Bitcoin was simply above $1000 when year began and achieved 850 percent overall growth. The market capitalization of digital currency has come to $137 billion. Because of this, Bitcoin become more valuable than significant organizations like McDonalds, Mastercard, British American Tobacco or Siemens. Dong (2017) suggested that Bitcoin great worth depends on the hundreds and thousands of miners optimizing and working in distributed fashion to support its value proposition. The more inflexible and secure block chain and historical transactions directly increase the Bitcoin value in the market. After the 51% attack the total hash rate surpasses the theoretical and practical possibilities.

According to Street (2017), government or other commodity does not support Bitcoin and other cryptocurrencies due to the fact of being purely digital tokens. Victims hardly have any option legally or criminally. Bitcoin transaction cannot be reversed, since criminals can rob the owner easily without being tracked. Unlike in the case of savings or checking account making it the largest limiting factor of Bitcoin. Investopedia (2017) suggested that there are several factors which affect the prices of the cryptocurrencies from trading volumes to media and it has been in the concentration of consideration for media and merchants and the rise of the price of Ethereum is not totally surprising. According to Investopedia (2017), within 24 hours, the price of Ethereum and Bitcoin crashed by 25%. By the end of 2017, the price of the Bitcoin hit target 0f $10,000 and the price of Ethereum hit target of $500 as predicted by a billionaire Michael Novogratz, who holds about 10% of his total assets in cryptocurrencies.

## 1.2 Machine Learning

**1.2.1 Definition:**

Machine Learning is a category of algorithms that allow software applications to predict much better results without being specifically programmed. The basic premise of machine learning is to build algorithms that receive input data and use statistical analysis to predict output data while output data is updated like many input data become valid. The processes involved in machine learning are similar to the processes of data mining and predictive modelling. Both require searching for certain patterns by date, and adjusting program actions accordingly.

Many people are also familiar with machine learning from internet shopping and the advertisements that are shown to them depending on what they are buying. This is because referral engines use machine learning to customize ads that are delivered online in near real time. In addition to personalized marketing, other well-known cases in which machine learning is used are fraud detection, spam filtering, threat detection of countries in the network, maintenance, predictability, and building the flow of news.

**1.2.2 How machine learning works:**

Machine learning algorithms are categorized as both supervised and unsupervised. Supervised algorithms. They require a data researcher, or data analyst, who has the knowledge of machine learning to supply the desired input and output data, in addition to delivering feedback on the accuracy of the predictions; acute during algorithm training. Data researchers determine which variables, or characteristics, should be analyzed by the model and used to develop predictions. Once the training is complete, the algorithm will apply what it has learned to new data. Supervised learning problems can be further grouped into regression and classification problems.

## 1.3 Classification

A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”. Regression: A regression problem is when the output

variable is a real value, such as “dollars” or “weight”. Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively. Some popular examples of supervised machine learning algorithms are: Linear regression for regression problems. Random forest for classification and regression problems, Support vector machines for classification problems.

## 1.4 Unsupervised Algorithms

They do not need training with output data. Instead, they use a method called deep learning to review the date and come to conclusions. Unsupervised and learned algorithms, also known as neural networks, are used for more complex processes than supervised algorithms, which include image recognition, speech-to-text, and natural language generation. These neural networks work by first combining millions of training examples with data and automatically identifying subtle correlations between multiple variables. Once trained, the algorithm can be used by associates to interpret new data. These algorithms become feasible only in the information age, because they require massive amounts of data to train. These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher.

Algorithms are left to their own devises to discover and present the interesting structure in the data. Unsupervised learning problems can be further grouped into clustering and association problems. Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior. Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y. Some popular examples of unsupervised learning algorithms are: k means for clustering problems., Apriori algorithm for association rule learning problems.

# 2. Project Specification

Research Question-

How the direction and accuracy of price of Bitcoin can be predicted by using data mining method?

## 2.1 Purpose

The purpose of this research is to determine how the direction and accuracy of price of Bitcoin can be predicted by using data mining methods. Study shows that research is lacking in the area of Bitcoin. Bitcoin is a problem based on time series prediction. In addition, Bitcoin is amazing unpredictable than various monetary standard which acts as a cryptocurrency lie in its transitory level. As a result, Bitcoin is amazingly unpredictable than other currencies like USD.

Additionally, Bitcoin is viewed as a leading cryptocurrency, ranked as four out of the latest five years as per Malkiel and Fama (1970). Therefore, its expectation offers an opportunity, and this gives inspiration to investigate in this area. As a prove by an investigation of the existing research paper, applying machine learning methods on a Bitcoin dataset can offer outstanding performance improvements in the area. This is investigated by applying different machine algorithms such as Random Forest, Support Vector Machine, Gradient Boosting algorithm, and neural network. The result is shown in terms of accuracy. The structure of this thesis builds on existing literature which is discussed in below section.

Further, predicting the price of the Bitcoin provide an advantage to make profit by buying or selling the asset. This paper is simply focus on the accuracy whether the price of the Bitcoin is going up or down and these facilities are provided by many exchanges. Thus, the net worth which will be gained from this strategy is not only depend on accuracy but also depends on the position size.

## 2.2 Research Variables

In this research, the dataset related to ten cryptocurrencies are collected from Kaggle and the closing price of all the ten cryptocurrencies are taken as an independent variable from the datasets. Rather than focusing on one cryptocurrency, in this research nine other cryptocurrencies are utilized to determine the accuracy and direction of the Bitcoin price. If one cryptocurrency were implemented against Bitcoin, probably the accuracy and price prediction could not seem proper. As the result, nine cryptocurrencies are used for this research which will not only show the trend of each cryptocurrency but also show the correlation among each other and thus give a better result. That is why, the new dataset is created by taking the closing price from ten cryptocurrencies.

# 3. Approach of The Project

## 3.1 What Is Data Mining?

Data is unquestionably valuable. However, analyzing it is not easy. With the exponential expansion of data, a technique to extract relevant information that leads to usable insights is required. This is where Data Mining comes into place. Data Mining acts as the backbone for **Business Intelligence**and **Data Analytics**. **Data Mining** can be defined as the process of analyzing large volumes of data to derive useful insights from it that can help businesses solve problems, seize new opportunities, and mitigate risks. It can be leveraged to answer business questions that were traditionally considered to be too time-consuming to resolve manually It is the process of finding patterns in large volumes of data to translate them into valuable information. Data Mining Tools help you get comprehensive **Business Intelligence**, plan company decisions, and substantially reduce expenses.

Due to the expanding significance of Data Mining in a wide range of industries, new tools, and software improvements are constantly being introduced to the market. As a result, selecting the appropriate Data Mining Tool becomes a challenging and time-consuming procedure. So, before making any hasty judgments, it’s critical to think about the company or research needs. There are two types of Data Mining Techniques, Descriptive and Predictive Data Mining.

By using a range of statistical techniques to analyze data in different ways, businesses can seamlessly identify patterns, relationships, and trends. For example, the world’s most popular streaming platform, Netflix, has approximately 93 million active users per month. The data pipeline of Netflix captures more than 500 billion user events per day. This includes data on various things such as video viewing activities, error logs, performance reports, etc.

The storage of this data requires approximately a storage space of 1.3 Petabytes (**1 Petabyte = 1,000,000 Gigabytes**) per day. The advantages of having such high volumes of data are as follows:

* It allows Netflix to plan its future releases by analyzing the kind of content viewers like.
* It allows Netflix to understand how they can make the user experience on their website and Android/iOS applications better by analyzing user behavior on these services.

Key Benefits of Data Mining

* **Pattern Discovery**: Automatic pattern discovery is a strategic advantage, and this technique helps in modeling and predicting future behavior.
* **Trend Analysis**: Understanding trends keeps you up-to-date with current developments in the industry, and helps reduce costs and timeliness to market.
* **Fraud Detection**: Data Mining techniques help in fraud detection by discovering anomalies in datasets. This is used to detect which insurance claims, credit card purchases, etc., are likely to be fraudulent.
* **Forecasting in Financial Markets**: Data Mining techniques are extensively used to model financial markets and predict likely outcomes.

## 3.2 What Is Descriptive Approach in Data Mining?

**Descriptive Mining**, as the name implies, “describes” the data. You convert the data into a human-readable format once it has been collected. Descriptive Analysis is used to extract information from data and to specify current information about past events. In simple terms, Descriptive research entails identifying interesting patterns or associations among data. Descriptive Mining is commonly used to generate correlation, cross-tabulation, frequency, and other similar results. These methods are dedicated to uncovering patterns and finding regularities in data. The other use of Descriptive Analysis is to find the most interesting subgroups in a large set of data. Descriptive Analytics is concerned with summarizing and converting data into usable information for reporting and monitoring. Furthermore, it allows for a thorough examination of the data so that questions like**“what happened?”** and**“what is happening?”** can be easily answered. There are four different types of Descriptive Data Mining tasks. They are as follows:

* **Clustering Analysis:**It is the process of determining which data sets are similar to one another. For example, to increase conversion rates, clusters of customers with similar buying habits can be grouped together with similar products.
* **Summarization Analysis:**It entails methods for obtaining a concise description of a dataset. For example, summarizing a large number of items related to Christmas season sales provides a general description of the data, which can be extremely useful to sales and marketing managers.
* **Association Rules Analysis:**This method aids in the discovery of interesting relationships between various variables in large databases. The retail industry is the best example. As the holiday season approaches, retail stores stock up on chocolates, with sales increasing before the holiday, which is accomplished through Data Mining.
* **Sequence Discovery Analysis:**It’s all about how to do something in a specific order. For instance, a user may frequently purchase shaving gel before purchasing a razor in a store. It all comes down to the order in which the user purchases the product, and the store owner can then arrange the items accordingly.

## 3.3 What Is Predictive Approach Data Mining?

**Predictive Data Mining** is the Analysis done to predict a future event or other data or trends, as the term ‘Predictive’ means to predict something. Business Analysts can use Predictive Data Mining to make better decisions and add value to the analytics team’s efforts. Predictive Analytics is aided by Predictive Data Mining. Predictive Analytics, as we all know, is the use of data to predict outcomes. An example of this is, any retailer can use algorithm-based tools to look through a customer database and predict future transactions by looking at previous transactions. In other words, previous data may allow the shopkeeper to forecast what will happen in the future, allowing businesspeople to plan accordingly. Its main goal is to predict future outcomes rather than current behavior. It predicts the target value using supervised learning functions.**Classification**, **Time-Series Analysis**, and **Regression** are the methods that fall under this category of Data Mining. Data Modeling is a requirement of Predictive Analysis, and it works by combining a few current variables with unknown future data values for other variables to predict the future.

There are four different types of Predictive Data-Mining tasks. They are as follows:

* **Classification Analysis**: It is used to retrieve critical and pertinent data and metadata. It categorizes information into various groups. Classification Analysis is best demonstrated by email providers. They use algorithms to determine whether or not a message is legitimate.
* **Regression Analysis**: It tries to express the interdependence of variables. Forecasting and prediction are common applications.
* **Time Serious Analysis**: It is a series of well-defined data points taken at regular intervals.
* **Prediction Analysis:**It is related to time series, but the time isn’t restricted.

## 3.4 Key Differences Between Descriptive and Predictive Data Mining

**3.4.1 Definition**

Descriptive Mining is frequently used to provide Correlation, Cross-Tabulation, Frequency, and other types of information. It analyses stored data to determine what happened in the past.

Predictive Data Mining is the Analysis done to predict a future event or multiple data or trends. It explains what might happen in the future as a result of past Data Analysis.

**3.4.2 Type of approach**

Descriptive and Predictive Data Mining: Type of Approach It’s crucial to remember that the amount of data available, the type of data, and the dimensions all play a role in determining which Data Mining approach to use.

Descriptive Data Mining is based on the reactive approach that is it just responds to the situation. When you want the data to respond to events after they happen, you use the reactive approach. Reactive Analysis isn’t possible for obvious reasons. It means that businesses respond to situations after the fact, which means they can’t prevent negative consequences or build on past successes. At best, this approach should be used sparingly.

Predictive Data Mining entails both controlling and responding to a situation, implying that it is based on a proactive approach. As it is used to forecast the types of data you’ll see in the future, prediction is one of the most valuable Data Mining techniques. In many cases, simply recognizing and comprehending historical trends is sufficient to make a reasonable prediction of what will occur in the future.

**3.4.3 Preciseness**

Because information is so important in a business, having accurate and reliable data to base your decisions on is critical. This is how you’ll make the right decisions and outsmart your opponents.

The Descriptive approach is more precise and accurate. It is thought to help identify variables and new hypotheses that can then be investigated further in experimental and inferential studies. It is useful because the margin for error is very small. After all, the trends are extracted directly from the data properties.

Predictive Data Miningproduces outcomes without ensuring accuracy. Predictive Data Mining models have always relied on past patterns to forecast the future. It is based on previous behaviors, events, and trends that you believe will occur; however, accuracy cannot be guaranteed.

**3.4.4 Tasks**

The various types of patterns to be identified in Data Mining activities are perceived by Data Mining functionalities. Data Mining features are used to define the types of patterns that will be discovered during Data Mining activities. Descriptive Mining tasks are used to describe the properties of data in a target data set. Descriptive Data Mining tasks are used to find data describing patterns and to extract new, significant information from a data set. A **Descriptive Data Mining task** could be defined as a retailer attempting to identify products that are purchased together. Predictive Mining tasks infer from current and past data to make predictions. **Predictive Data Mining tasks** create a model from the available data set that can be used to predict unknown or future values in a different data set of interest.

**Requirements**

Data Mining is also useful for summarizing the data in such a way that the result is understandable and meaningful to end-users. This relationship is discovered through the use of linear equations, rules, clusters, graphs, and recurrent patterns in time series, among other methods. Find information in data sets that are stored in **Databases**, **Data Warehouses,** **Online Analytical Processes**, and other repositories. To discover historical data, Descriptive Data Mining employs two techniques **Data Aggregation and Data Mining**. To make the datasets more manageable for analysts, data is first collected and sorted by data aggregation.

# 4. Problem Statement

The objective of this project is to collect data from different researchers and find the results

obtained by them when they use different algorithm for dataset and finally come to a conclusion of finding an effective algorithm which has a better accuracy. Analyzing different machine learning method to get the best accuracy in bitcoin price prediction. Testing random dataset which can be downloaded from Kaggle and checking whether the algorithm performs well on a preferred dataset or on a generalized dataset with different families of price prediction. Using Machine Learning Algorithm and Deep Learning I am trying to get a bitcoin price prediction accuracy. Splitting the dataset in training and testing dataset using sklearn. model\_selection. train\_test\_split and then changing the parameter like train size from 0.1 to 0.9 all while keeping the random state as 0,1,2,3,42.

# 5. Requirements:

These are the software and hardware required for our project in order to execute is properly. The hardware and software listed below are best compatible for the project at the time of development.

## 5.1 Materials Required:

**5.1.1 Computer:**

A base machine with windows 8, 8 GB RAM, 250 GB of hard drive with core i5, installed

**5.1.2 Operating System:**

The main OS to be used is Windows 10

**5.1.3 Environment:**

Google Colab

## 5.2 Functional Partitioning of Project:

Overall Project is partitioned into various phases.

**Phase 1**:

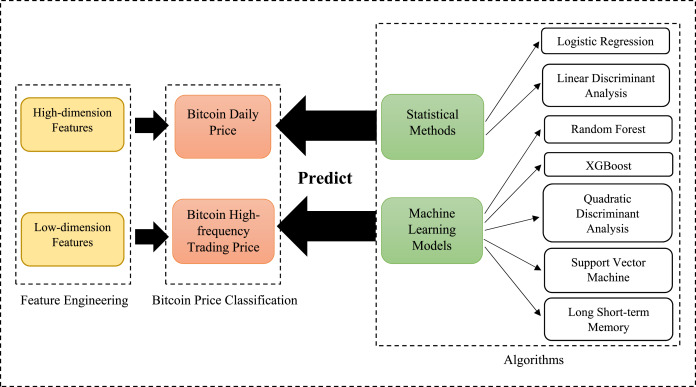
Downloading 1,38,000+ time series dataset of bitcoin price.

**Phase 2:**

Performing price analysis and classification, choosing best out of 9 ML algos and feature classification.

# 6. Different type of models that we used in our project:

Models used in the work are deep learning models which are used for time -series forecasting. All the models basically follow same procedures to predict the price for the bitcoin. First step for all the models includes preprocessing of the acquired data. After that the data is split into two sets of test and train data. Followed by training of the models.



*Fig 1: Block Diagram for price prediction using different algorithms.*

****

*Fig 2: Block Diagram for working of algorithms.*

|  |  |  |  |
| --- | --- | --- | --- |
| Techniques | Advantages | Disadvantages | Parameter |
| Artificial Neural Network | Better performance compared to regression. Lower prediction error | Prediction gets worse with increased noise variation | Stock closing price |
| Theta | Does not lose much accuracy when applied to a sample from outside the training sample | Exaggerate to minor fluctuations in the training data which decrease the predictive ability | Consumer investment,net revenue,net income, price per earnings ratio of a stock |
| ARIMA | Robust and efficient | It is suitable for short term predictions only | Open, high, low, close prices and moving average |
| Time series linear model | Integrate the actual data to the ideal linear model | Traditional and the seasonal trends present in the data | Data and number of months |
| RNN | Previous time points to the input layer contain inputs | It is possible to feed those words in through a much smaller set of input nodes | Input, hidden, and output layers |
| LSTM | Handling long-range dependency. | Get affected by different random weight initialization | Time-series data. |
| Theta | Normalizes all of the input values | Architectures of input images are complicated | An array as the input graph of the CNN |
| Kalman Filter | The capability of providing a reconstructed noise-free time series. | Strong residual noise and spurious artifacts | Nonlinear and non-stationary  data at different time intervals and frequencies. |

*Fig 3: Comparison of all the models used in the work.*

**`**

## ****6.1 LSTM****

Long short-term memory (LSTM) is an [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) used in the fields of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) and [deep learning](https://en.wikipedia.org/wiki/Deep_learning). Unlike standard [feedforward neural networks](https://en.wikipedia.org/wiki/Feedforward_neural_network), LSTM has feedback connections. Such a [recurrent neural network](https://en.wikipedia.org/wiki/Recurrent_neural_network) can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected [handwriting recognition](https://en.wikipedia.org/wiki/Handwriting_recognition), [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), [machine translation](https://en.wikipedia.org/wiki/Machine_translation), robot control, video games, and healthcare. LSTM has become the most cited neural network of the 20th century.

A common LSTM unit is composed of a cell, an input gate, an output gateand a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to [classifying](https://en.wikipedia.org/wiki/Classification_in_machine_learning), [processing](https://en.wikipedia.org/wiki/Computer_data_processing) and [making predictions](https://en.wikipedia.org/wiki/Predict) based on [time series](https://en.wikipedia.org/wiki/Time_series) data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the [vanishing gradient problem](https://en.wikipedia.org/wiki/Vanishing_gradient_problem) that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, [hidden Markov models](https://en.wikipedia.org/wiki/Hidden_Markov_models) and other sequence learning methods in numerous applications.

LSTM networks were designed specifically to overcome the long-term dependency problem faced by recurrent neural networks RNNs (due to the [vanishing gradient problem](https://en.wikipedia.org/wiki/Vanishing_gradient_problem)).

LSTMs have feedback connections which make them different to more traditional feedforward neural networks. This property enables LSTMs to process entire sequences of data (e.g., time series) without treating each point in the sequence independently, but rather, retaining useful information about previous data in the sequence to help with the processing of new data points. As a result, LSTMs are particularly good at processing sequences of data such as text, speech and general time-series.

## ****6.2 Dense LSTM****

LSTM layers mostly in the time series analysis or in the NLP problems, convolutional layers in image processing, etc. A dense layer also referred to as a fully connected layer is a layer that is used in the final stages of the neural network.

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models.

## ****6.3 RNN****

Neural networks are a set or collection of algorithms that try to recognize the underlying relationship in a set of data. This process mimics the functionalities of the human brain and the process of understanding data. These neural networks are designed to work in a very similar fashion to that of the human brain. Neural networks are the core of deep learning algorithms and are used for multiple applications such as speech recognition, facial recognition, bitcoin prediction, social media, handwriting and signature verification, weather forecasting, etc. Neural networks have many interconnected layers that can function with inputs.

RNN is a special type of artificial neural network (ANN) used for time-series or sequential data. Feedforward neural networks are used when data points are independent of each other. In the case of sequential data points, they are dependent on each other. In that case, you need to modify the neural networks to incorporate dependencies between data points. RNNs have the concept of memory, which helps them store states or information of previous inputs to generate the next sequence of output.

It saves the output of a particular layer and feeds this back to the input to predict the output of the layer. As the above image shows, you can convert a normal feedforward neural network to RNN. The nodes in the different layers of the neural network are compressed to form a single layer. In the image below, A, B, and C are the parameters of the network.

Here, x is the input layer, h is the hidden layer, and y is the output layer. A, B, and C are the network parameters that are used to improve the output of the model. At any given time (t), the current input is a combination of input at x(t) and x(t-1). The output is fetched back to the network to improve the output.

This is an important question that needs to be answered to better understand RNNs. Every invention, upgrade, or update offers effective solutions to existing problems. RNNs were created to solve several issues of feedforward neural networks such as:

* Feedforward neural networks not being able to handle sequential data.
* Feedforward neural networks only consider the current input.
* Feedforward neural networks not being able to memorize previous inputs.

The single best solution to these problems is RNNs. They can handle sequential data and accept current input data and previously received inputs. The memory of RNNs can memorize inputs due to their memory.

## 6.4 ARIMA

An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses [time series data](https://www.investopedia.com/terms/t/timeseries.asp) to either better understand the data set or to predict future trends. A statistical model is autoregressive if it predicts future values based on past values. For example, an ARIMA model might seek to predict a stock's future prices based on its past performance or forecast a company's earnings based on past periods. An autoregressive integrated moving average model is a form of [regression analysis](https://www.investopedia.com/terms/r/regression.asp) that gauges the strength of one dependent variable relative to other changing variables. The model's goal is to predict future securities or financial market moves by examining the differences between values in the series instead of through actual values.

An ARIMA model can be understood by outlining each of its components as follows:

* [Autoregression (AR)](https://www.investopedia.com/terms/a/autoregressive.asp): refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
* Integrated (I): represents the differencing of raw observations to allow for the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).
* [Moving average (MA)](https://www.investopedia.com/terms/m/movingaverage.asp):  incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

ARIMA Parameters

Each component in ARIMA functions as a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with p, d, and q, where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as:

* p: the number of lag observations in the model; also known as the lag order.
* d: the number of times that the raw observations are differenced; also known as the degree of differencing.
* q: the size of the moving average window; also known as the order of the moving average.

## 6.5 Kalman Filtering

**If a dynamic system is linear and with Gaussian noise, the optimal estimator of the hidden states is the Kalman Filter.** This online learning algorithm is part of the fundamentals of the machine learning world. Understanding it well is important prior to understanding more complicated topics such as particle filters. In this article I will kick off with an example application of the Kalman filter, then I’ll describe the algorithm itself, I’ll apply it to some simple synthetic data, and finally, I will showcase where the Kalman filter fails.

A Kalman Filter is an algorithm that takes data inputs from multiple sources and [estimates](https://deepai.org/machine-learning-glossary-and-terms/estimator) unknown variables, despite a potentially high level of signal noise. Often used in navigation and control technology, the Kalman Filter has the advantage of being able to predict unknown values more accurately than if individual predictions are made using singular methods of measurement.

Kalman Filters use a two-step process for estimating unknown variables. The algorithm works by first estimating the current state variables, and measures their uncertainties. Then, the algorithm updates the estimates using a weighted average, wherein more weight is attributed to estimates with higher levels of uncertainty. Because the filter takes in information from multiple sources, both current state and predicted state, the filter is able to provide a level of accuracy higher than if estimates were made given only one of the multiple sources.

One of the most common uses for the Kalman Filter is in navigation and positioning technology. Imagine a car with a GPS transmitter is traveling down a mountain road. A Kalman Filter can be applied to take in the GPS data from the car, however GPS devices are not always entirely accurate. So, the Kalman Filter can take in speed and velocity data to adjust the rate of change in the cars position over time. Naturally, given the laws of physics, the level of variable uncertainty is lower when the car is traveling faster, and vice versa. All of this information is used to predict where the car will be, and then the process is repeated with updated information as the car travels down the road. Because the Kalman Filter is recursive, it doesn't need to know the entirety of the cars position and speed data, but rather just the last known position and speed. The underlying model of updating information is similar to that of a [Hidden Markov model](https://deepai.org/machine-learning-glossary-and-terms/hidden-markov-model).

## 6.6 Exponential Smoothing

Exponential smoothing is a time series forecasting method for univariate data that can be extended to support data with a systematic trend or seasonal component. It is a powerful forecasting method that may be used as an alternative to the popular Box-Jenkins ARIMA family of methods. Exponential smoothing is a time series forecasting method for univariate data. Time series methods like the Box-Jenkins ARIMA family of methods develop a model where the prediction is a weighted linear sum of recent past observations or lags. Exponential smoothing forecasting methods are similar in that a prediction is a weighted sum of past observations, but the model explicitly uses an exponentially decreasing weight for past observations. Specifically, past observations are weighted with a geometrically decreasing ratio. Exponential smoothing methods may be considered as peers and an alternative to the popular Box-Jenkins ARIMA class of methods for time series forecasting. Collectively, the methods are sometimes referred to as ETS models, referring to the explicit modeling of Error, Trend and Seasonality. By adjusting parameter values, analysts can change how quickly older observations lose their importance in the calculations. Consequently, analysts can tweak the relative importance of new observations to older observations to meet their subject area’s requirements. In contrast, the [moving average method](https://statisticsbyjim.com/time-series/moving-averages-smoothing/) weights all past observations equally when they fall within the moving average window and it gives observations outside the window zero weight. Like the Box-Jenkins ARIMA methodology, statisticians refer to exponential smoothing as an ETS model because it models error, trend, and seasonality in time series data.

## 6.7 Seasonal Naïve

Naïve forecasting is the technique in which the last period’s sales are used for the next period’s forecast without predictions or adjusting the factors. Forecasts produced using a naïve approach are equal to the final observed value. The naïve forecasting method is the easiest of all methods and it is suitable for finance and sales departments because it ensures that these departments work to improve the company. Other, more sophisticated, forecasting methods include the moving averages method (MA), linear trend forecasting, and exponential smoothing.

The main method we are focusing on in this article is known as naïve because there are no calculations or formulas, only an assertion of the actual sales numbers. In some cases, naïve forecasting can accurately predict situations, while others can be problematic because it considers only the previous period to forecast the next period. Thus, historical sales data is the foremost requirement for naïve forecasting and factors such as seasonality are not considered.

How to calculate a naïve forecast? The naive forecasting method in Microsoft Excel does not generate a final forecast; instead, it provides information about the “system.” The “system” refers to the type of market or industry in which you are forecasting.

As previously mentioned, “calculating” a naïve forecast is a simple technique. Consider the previous month’s sales and use it to forecast the sales for the next period.

## 6.8.1 Naive Models

Naive forecasting models are based exclusively on historical observation of sales or other variables, such as earnings and cash flows. They do not attempt to explain the underlying causal relationships that produce the variable being forecast.

Naive models may be classified into two groups. One group consists of simple projection models. These models require inputs of data from recent observation, but no statistical analysis is performed. The second group is made up of models that while naive, are complex enough to require a computer. Traditional methods such as classical decomposition, moving average, and exponential smoothing models are some examples.

The advantage is that it is inexpensive to develop, store data, and operate. The disadvantage is that it does not consider any possible causal relationships that underly the forecasted variable.

### **Seasonal naïve method**

A similar method is useful for highly seasonal data. In this case, we set each forecast to be equal to the last observed value from the same season (e.g., the same month of the previous year). This looks more complicated than it really is. For example, with monthly data, the forecast for all future February values is equal to the last observed February value. With quarterly data, the forecast of all future Q2 values is equal to the last observed Q2 value (where Q2 means the second quarter). Similar rules apply for other months and quarters, and for other seasonal periods

## 6.8.2 Naïve Last

Some forecasting methods are extremely simple and surprisingly effective. We will use the following four forecasting methods as benchmarks throughout this book. Naive Bayes classifier is an algorithm that uses Bayes' theorem to classify objects. Naive Bayes classifiers assume strong, or naive, independence between attributes of data points. Popular uses of naive Bayes classifiers include spam filters, text analysis and medical diagnosis. These classifiers are widely used for machine learning because they are simple to implement. Naive Bayes is also known as simple Bayes or independence Bayes.

### **Average method**

Here, the forecasts of all future values are equal to the average (or “mean”) of the historical data.

For naïve forecasts, we simply set all forecasts to be the value of the last observation. This method works remarkably well for many economic and financial time series. Because a naïve forecast is optimal when data follow a random walk these are also called **random walk forecasts**.

### **Last naïve method**

A similar method is useful for highly seasonal data. In this case, we set each forecast to be equal to the last observed value from the same season (e.g., the same month of the previous year). This looks more complicated than it really is. For example, with monthly data, the forecast for all future February values is equal to the last observed February value. With quarterly data, the forecast of all future Q2 values is equal to the last observed Q2 value (where Q2 means the second quarter). Similar rules apply for other months and quarters, and for other seasonal periods.

## 6.9 Theta

The Theta model is a times series forecasting model derived from the idea that: “An extrapolative method is practically incapable of capturing efficiently all the available information hidden in a time series”. On the one hand there are models that are too simple to catch all the available information. On the other hand, there are methods with more parameters employed in order to cope with more demanding underlying patterns; unfortunately, while optimizing all these parameters usually these complex methods end up actually over-fitting the actual data. So, this approach aims to help the models capture the data. This is achieved by breaking the data down into several simpler series, each one of which captures part of the information included in the original series.

Thus, in essence, a decomposition approach is employed. As a result of this process simpler models can adapt to these simpler series. For example, instead of trying to adapt Holt Exponential Smoothing to an initial set of data we could alternatively create two series - one that captures the short-term information and one that captures the long or the medium trend. Then we could try fit Naïve, Simple Exponential Smoothing (SES) or even a Linear Regression line (LRL) to each of those two series.

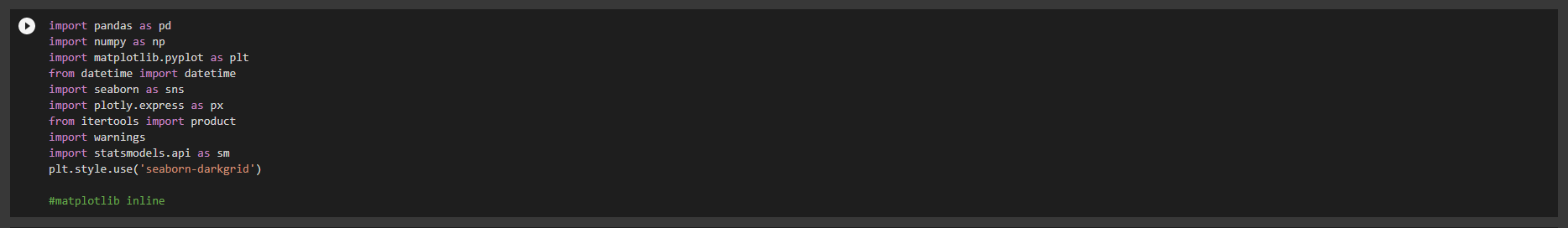
The model is based on the concept of modifying the local curvatures of the time series. This change is obtained from a coefficient, called Θ which is applied directly to the second differences of the time series. Following this procedure, a set of new time series, the so-called Theta-lines are constructed noted as L(Θ). The initial time series is decomposed into two or more Theta-lines L(Θ). Each of the Theta-lines is extrapolated separately and the forecasts are combined either equally weighted or through a weight optimization procedure. Any forecasting method can be used for the extrapolation of L(Θ). In the M3-competition [2] exponential smoothing was used for the extrapolation of L(Θ)

If we consider one of the simplest cases in which the initial time series X={X1,…,Xn} is decomposed into two L(Θ), L(Θ=0) and L(Θ=2), then the algebra can be significantly simplified.

Thus, in practice, the model can be easily implemented in a Microsoft Excel Worksheet via the following steps:

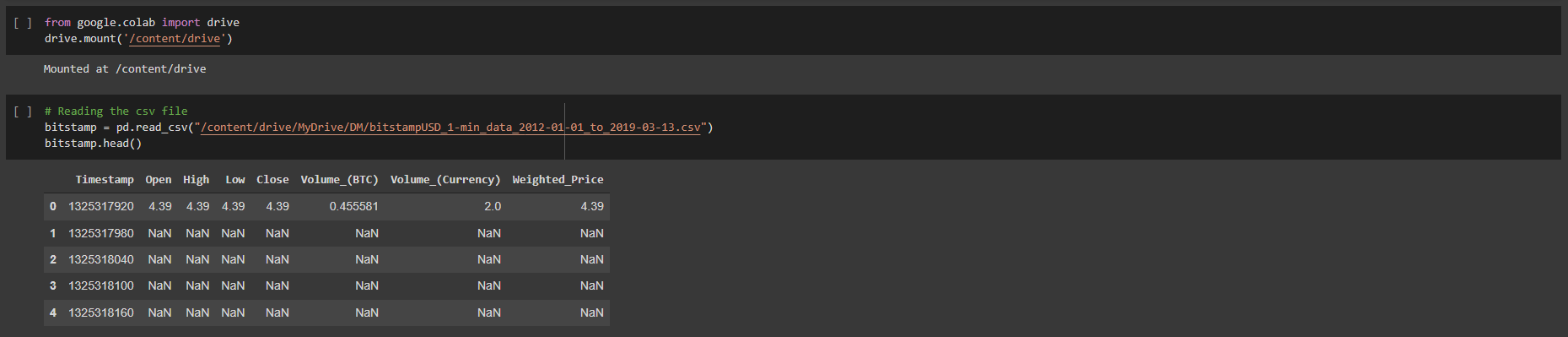
* Step 1: Apply Simple Linear Regression to non-seasonal data and prepare the LRL line and forecasts
* Step 2: Prepare the values for L(Θ=2) with formula [1], that is subtracting the LRL values from the actual data multiplied by two.
* Step 3: Extrapolate L(Θ=2) with either SES (optimized with Microsoft Solver) or with a simpler method, such as a Moving average or even a Naïve forecast [4,5]
* Combine with equal weights the forecasts from SES and LRL.

7. Comparing all algorithms:



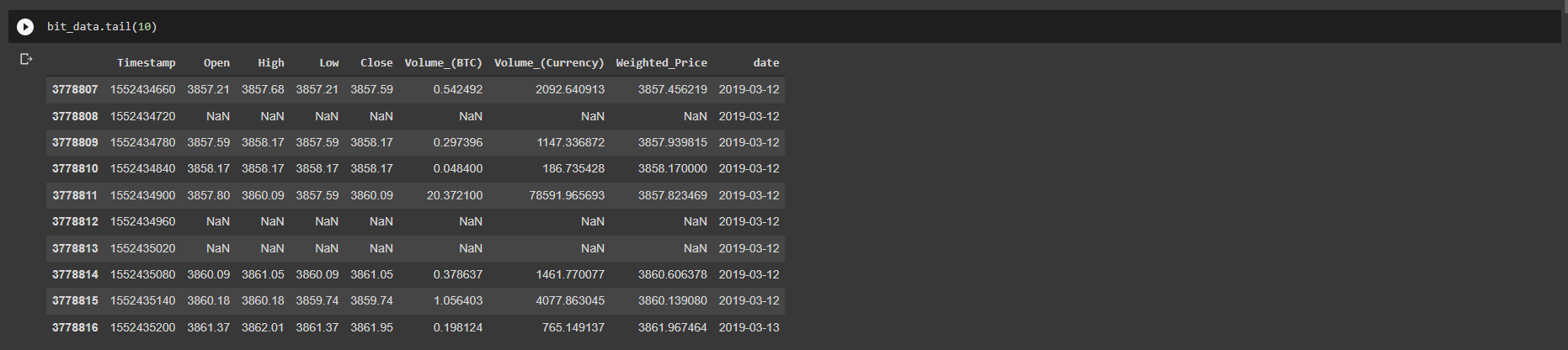
*Fig 4: Libraries used in the project.*

## 7.1 Import Dataset:

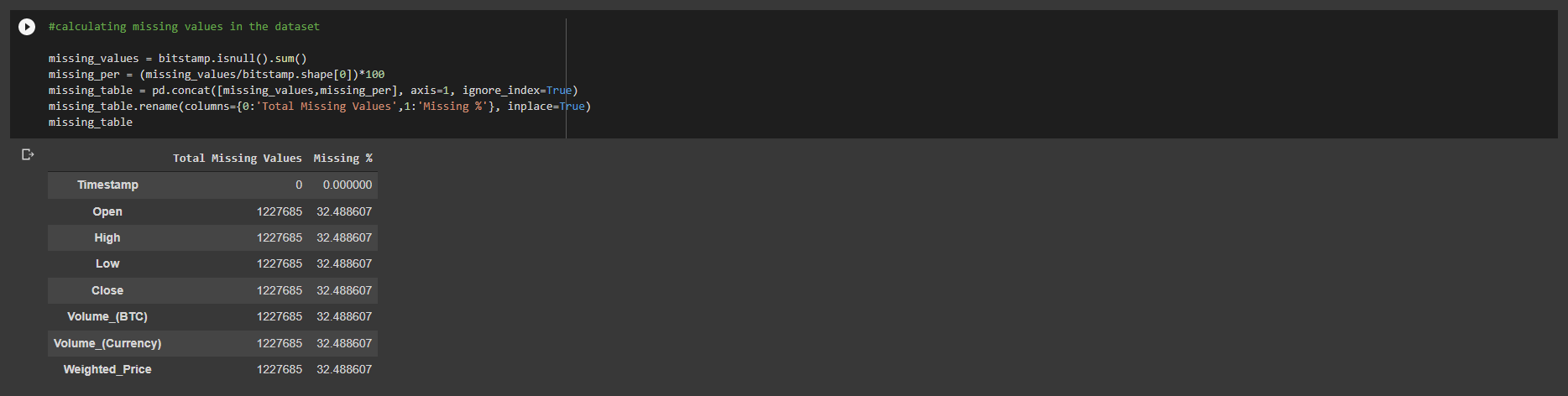


*Fig 5: Importing the dataset.*

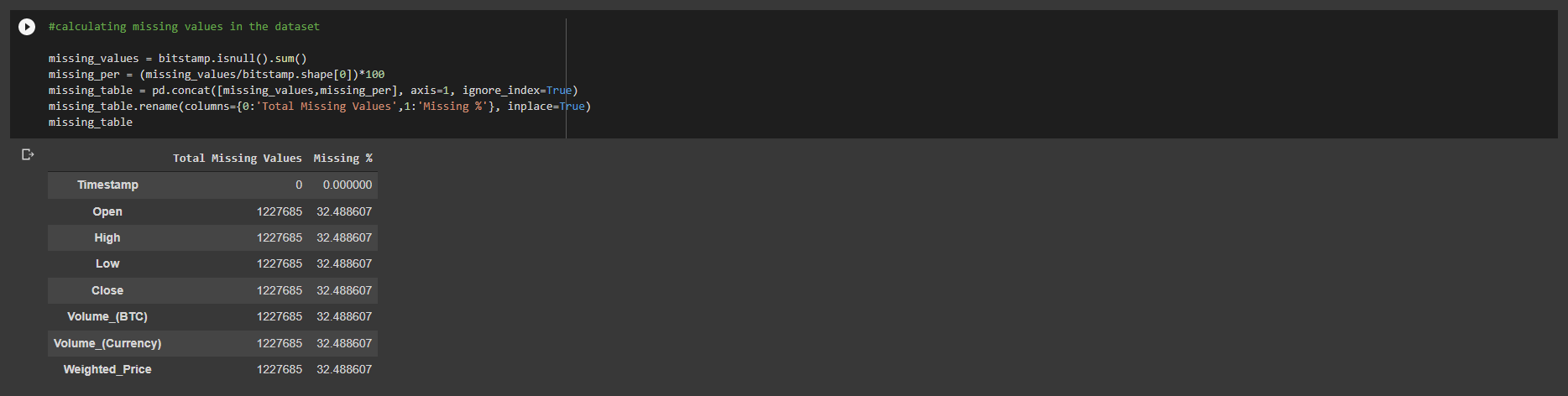
## 7.2 Dataset:



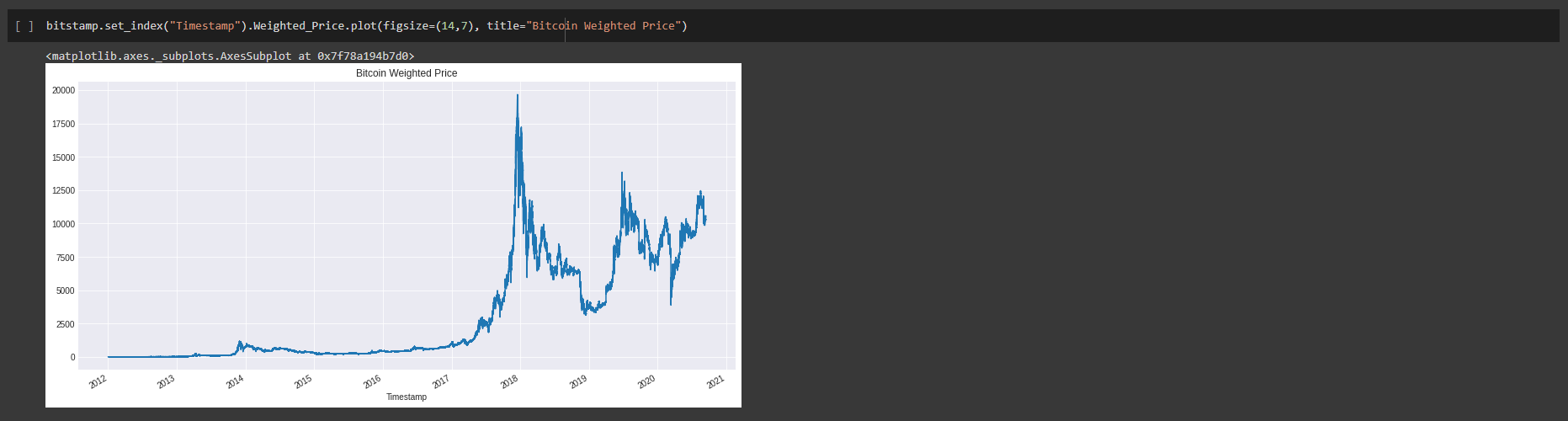
*Fig 6: Structure of the dataset.*



*Fig 7: Filling the missing values in the dataset.*

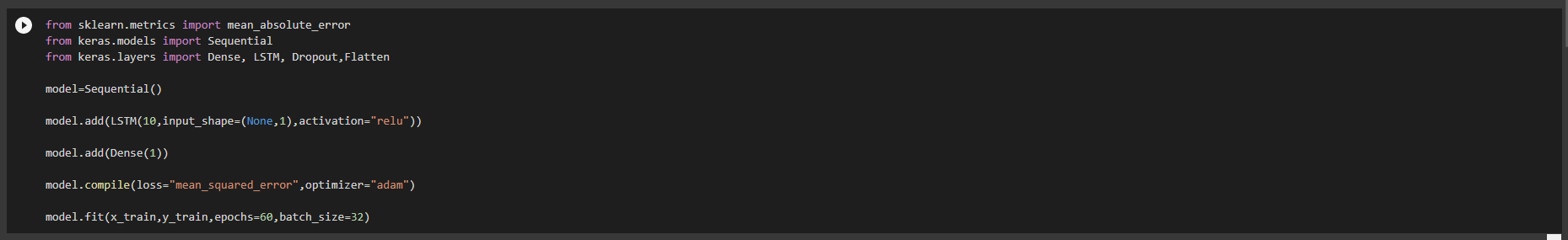


*Fig 8: Information about total missing values in the dataset.*

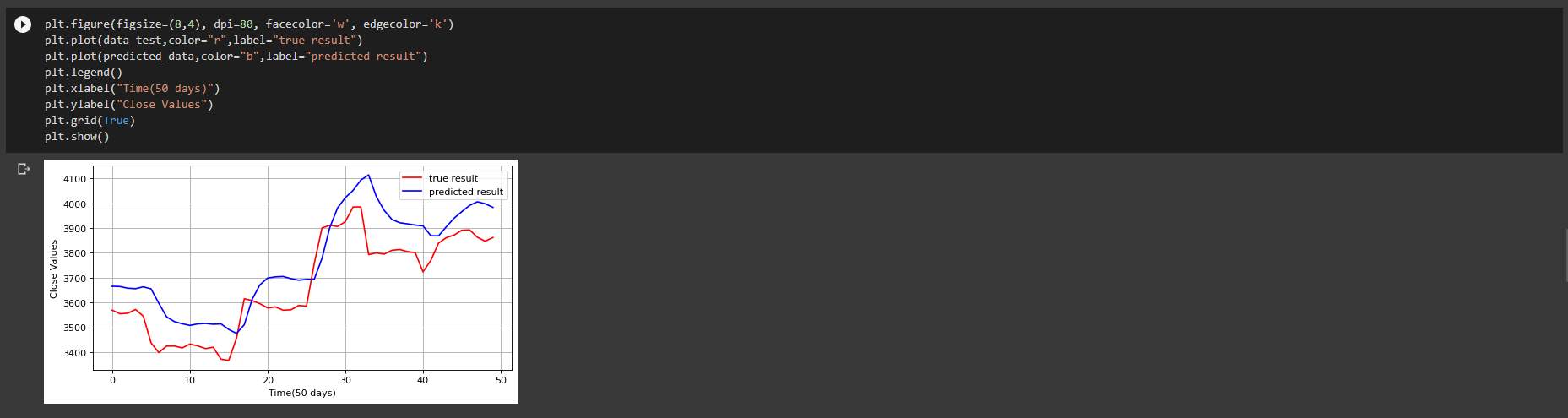


*Fig 9: Visualization of the dataset.*

## 7.3 LSTM

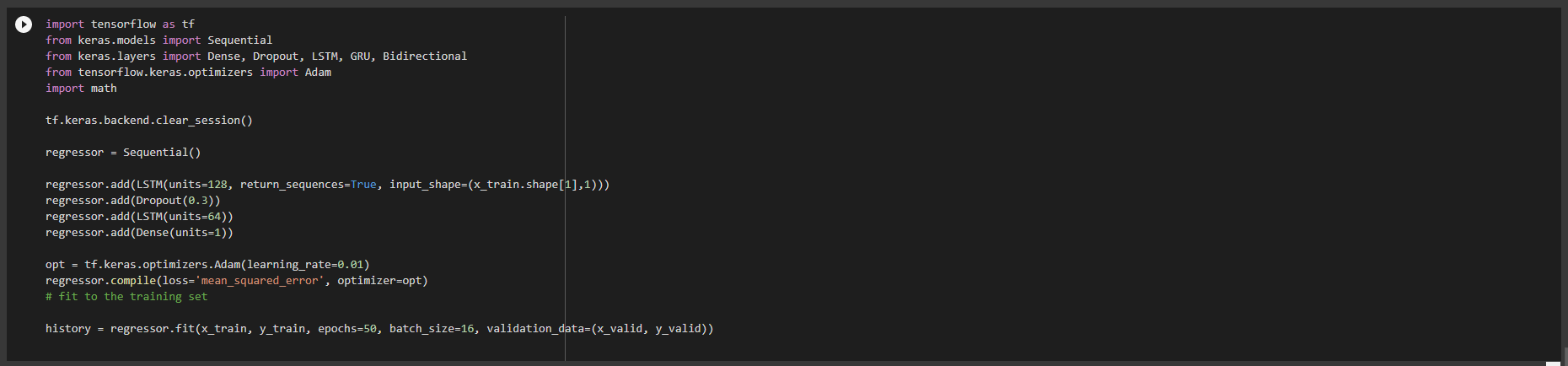


*Fig 10: Algorithm for LSTM (Long Short-Term Memory).*

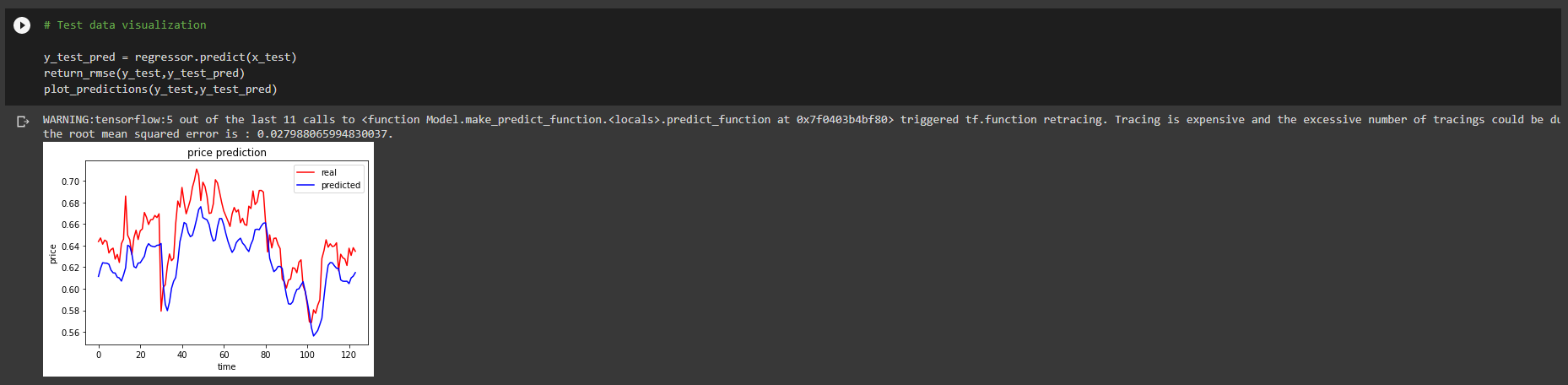


*Fig 11: Prediction of bitcoin price with LSTM deep learning algorithm.*

7.4 Dense LSTM



*Fig 12: Algorithm for Dense LSTM (Dense Long Short-Term Memory).*

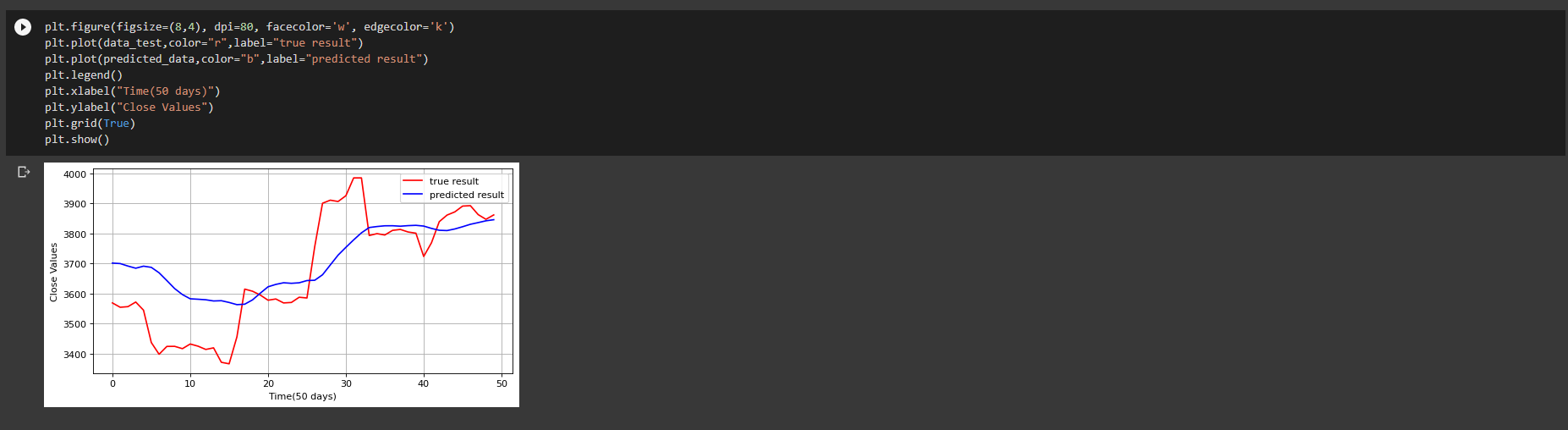


*Fig 13: Prediction of bitcoin price with Dense LSTM deep learning algorithm.*

7.5 RNN

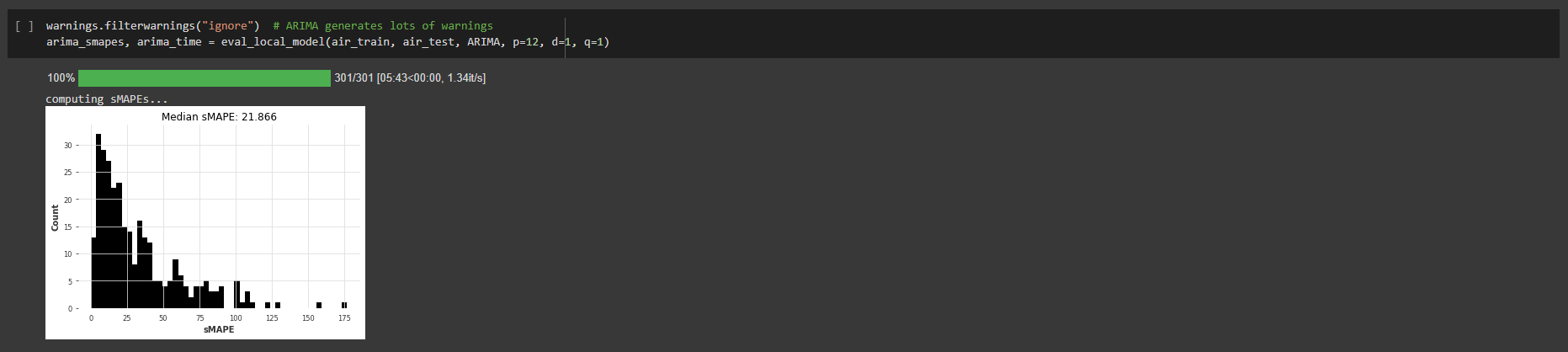


*Fig 14: Algorithm for RNN (Recurrent Neural Network).*



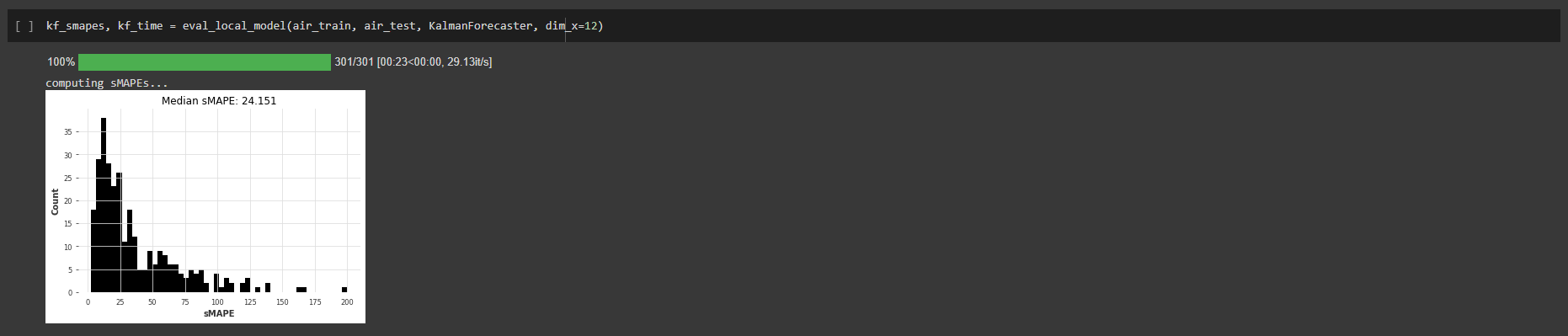
*Fig 15: Prediction of bitcoin price with RNN deep learning algorithm.*

## 7.6 ARIMA



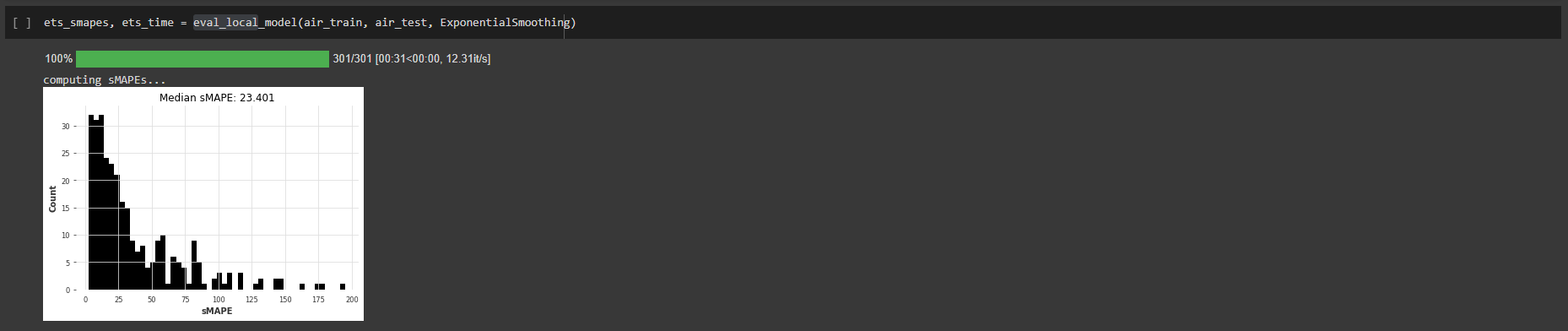
*Fig 16: sMAPE visualization of prediction using ARIMA deep learning model.*

## 7.7 Kalman Filtering



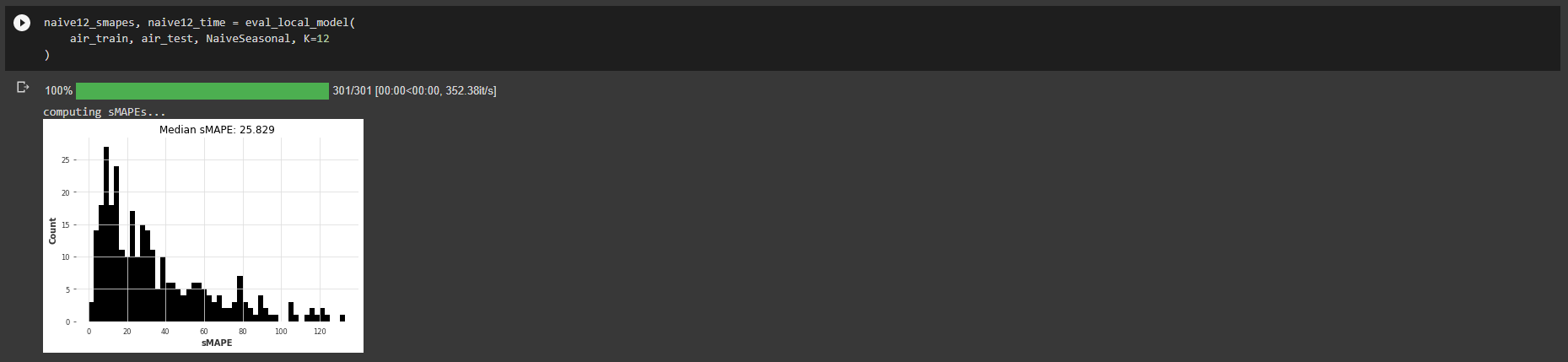
*Fig 17: sMAPE visualization of prediction using KALMAN FILTERING.*

## 7.8 Exponential Smoothing



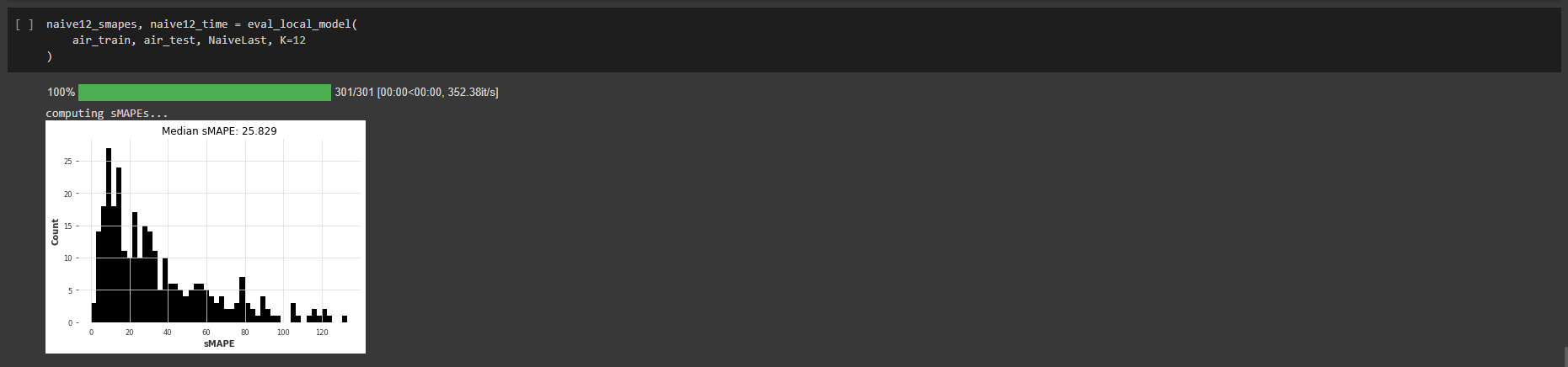
*Fig 18: sMAPE visualization of prediction using EXPONENTIAL SMOOTHING.*

## 7.9 Seasonal Naïve



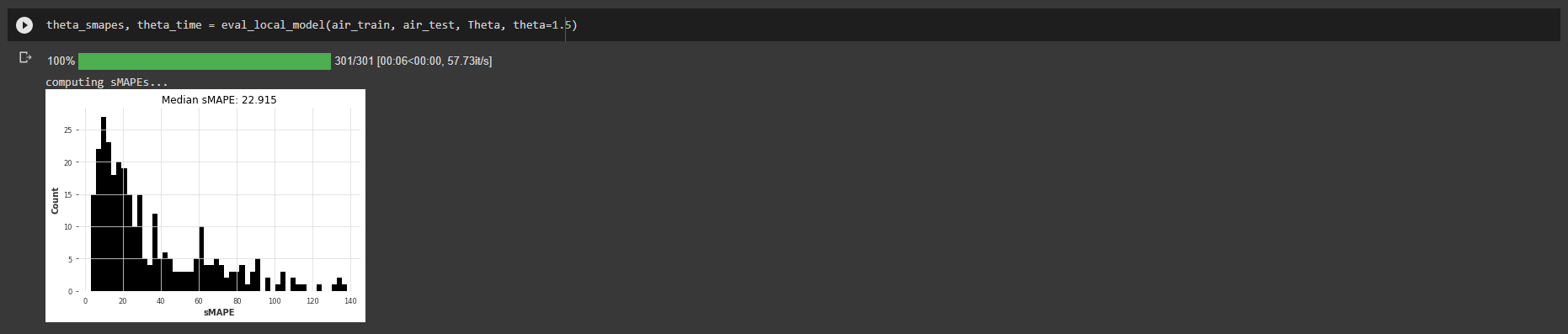
*Fig 19: sMAPE visualization of prediction using Seasonal Naïve.*

## 7.10 Naïve Last



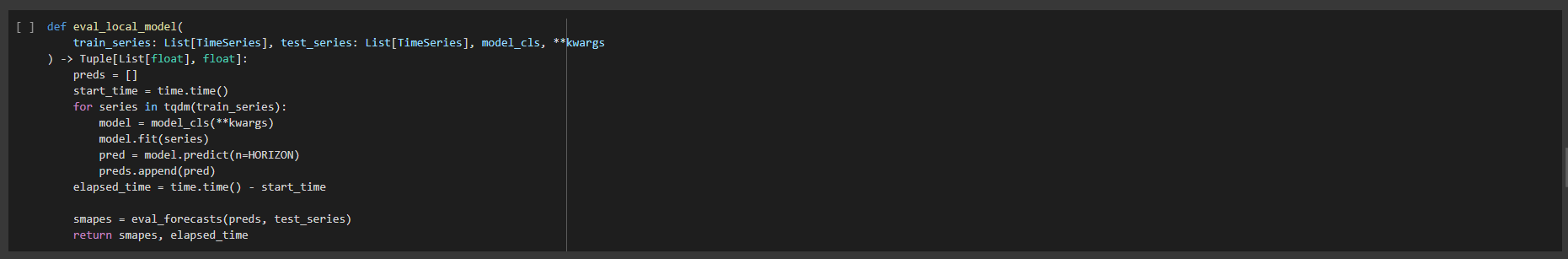
*Fig 20: sMAPE visualization of prediction using Naïve Last.*

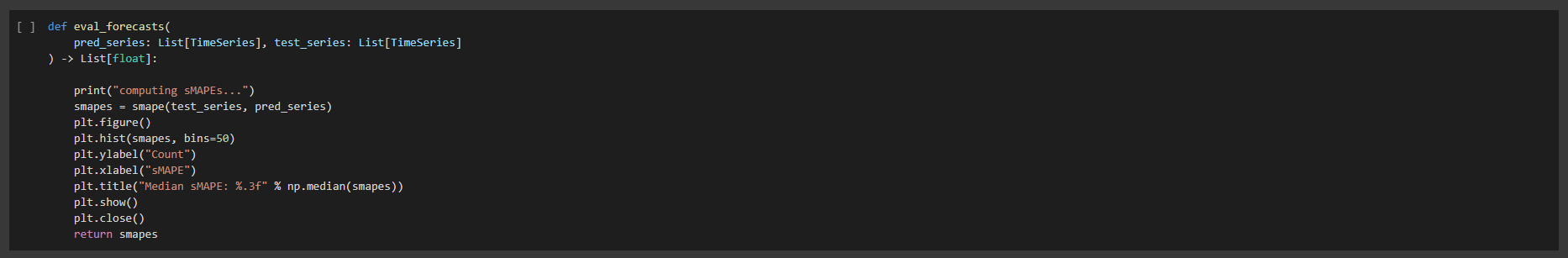
## 7.11 Theta

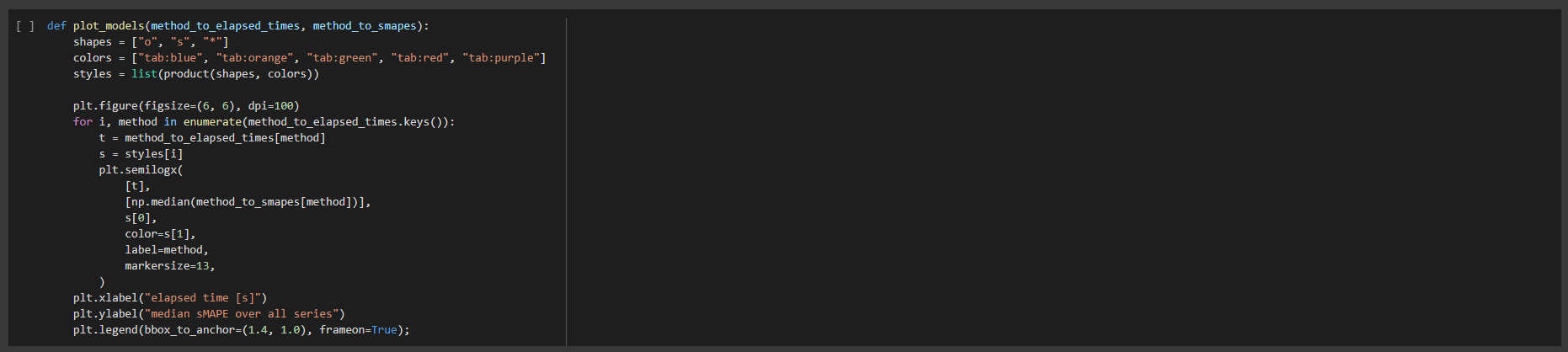


*Fig 21: sMAPE visualization of prediction using Theta.*

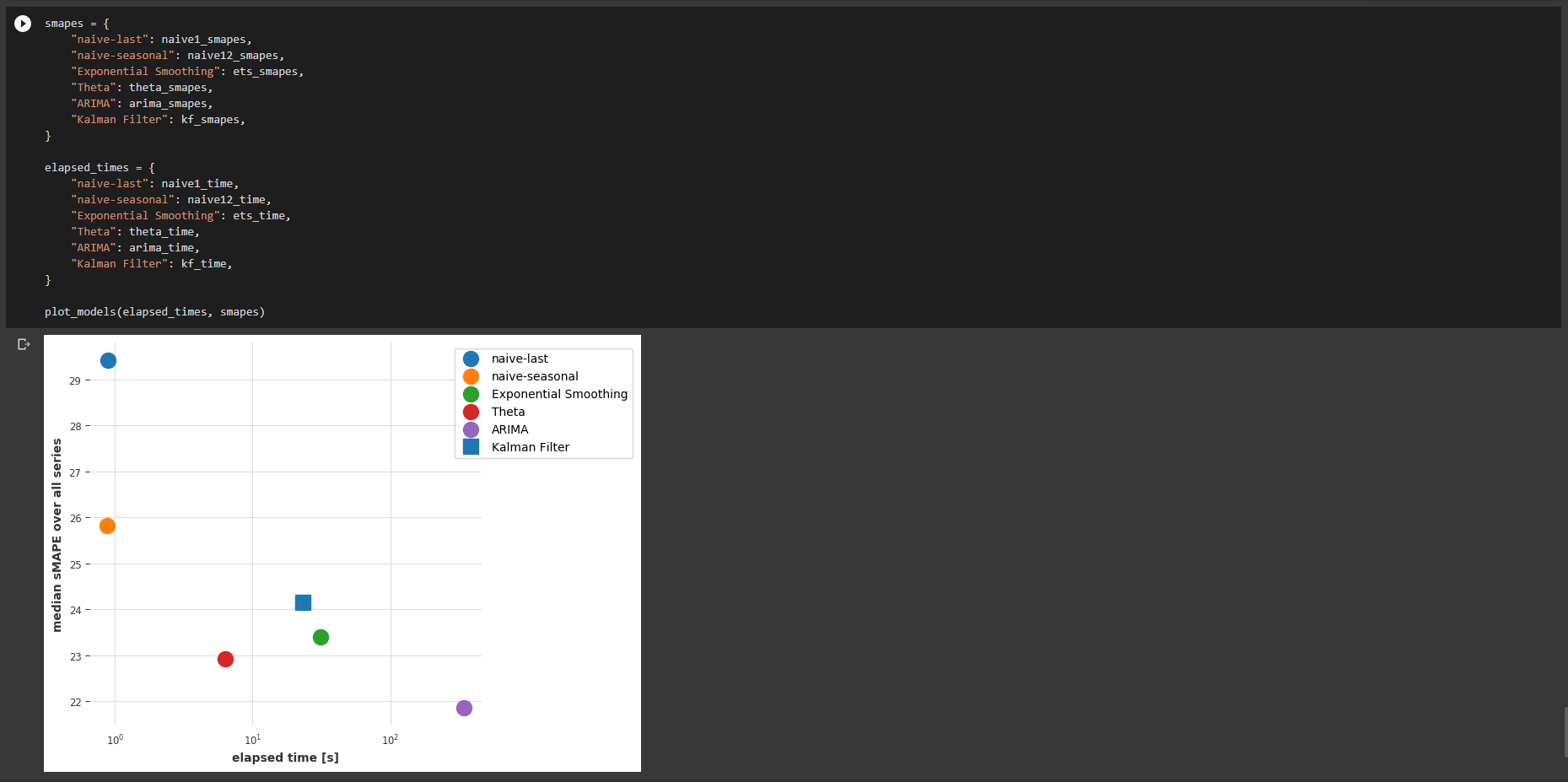
## 7.12 Comparing All Algorithms:

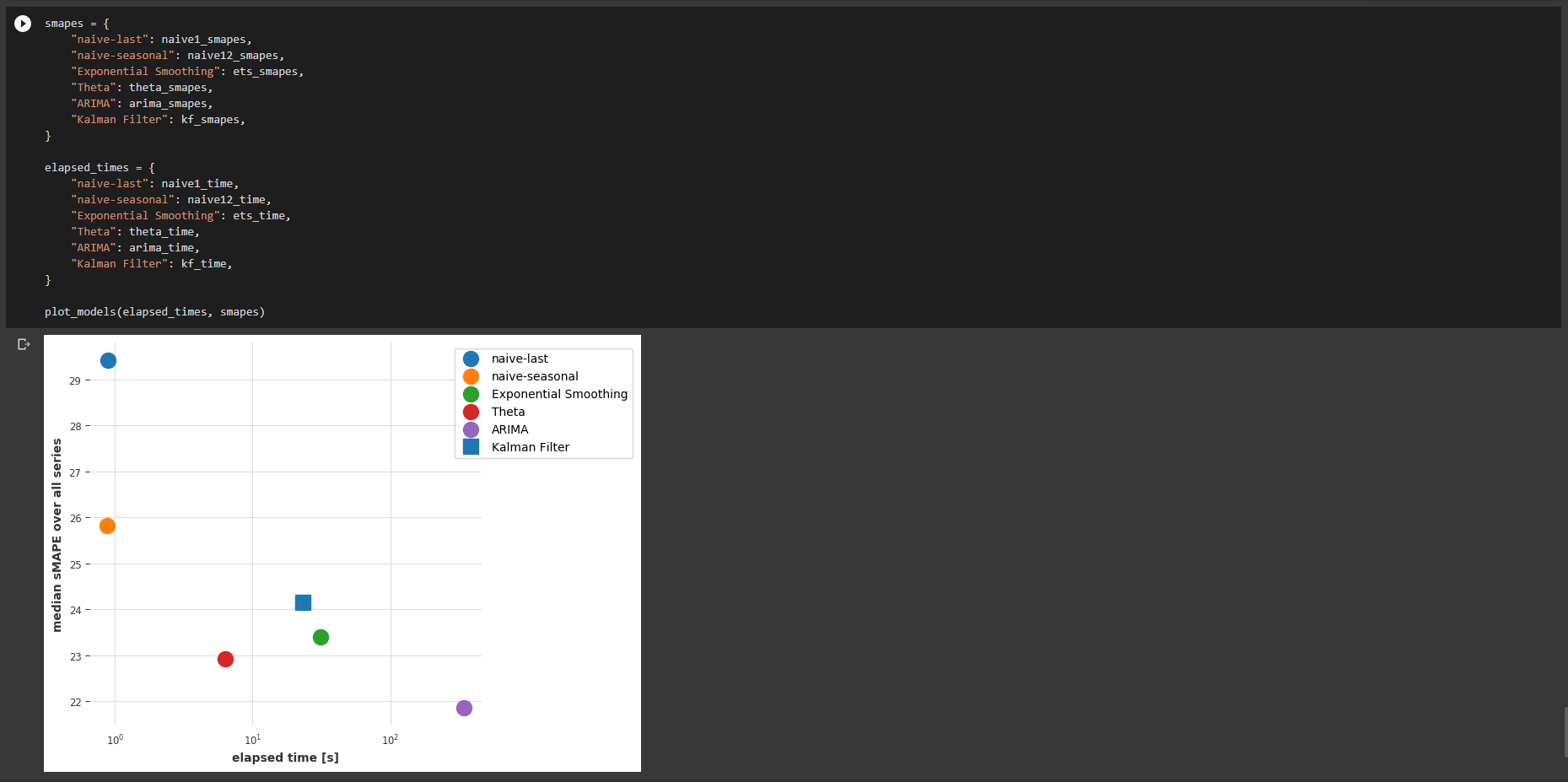






*Fig 22: Functions defined to predict and plot the bitcoin price in the project.*





*Fig 23: Comparing the last 6 models by their sMAPE value and the time elapsed by each model to predict the bitcoin price.*

# 8. Conclusion

All in all, predicting a price-related variable is difficult given the multitude of forces impacting the market. Add to that, the fact that prices are by a large extent dependent on future prospects rather than historic data. However, using deep neural networks has provided us with a better understanding of Bitcoin. Deep learning models such as the RNN and LSTM are evidently effective for Bitcoin prediction with the LSTM more capable for recognizing longer-term dependencies. However, a high variance task of this nature makes it difficult to transpire this into impressive validation results. As a result, it remains a difficult task.

There is a fine line between overfitting a model and preventing it from learning sufficiently. Dropout is a valuable feature to assist in improving this. However, despite using Bayesian optimization to optimize the selection of dropout it still couldn’t guarantee good validation results. Despite the metrics of sensitivity, specificity and precision indicating good performance, the actual performance of the ARIMA forecast based on error was significantly worse than the neural network models.

The LSTM outperformed the RNN marginally, but not significantly. However, the LSTM takes considerably longer to train. The performance benefits gained from the parallelization

of machine learning algorithms on a GPU are evident with a 70.7% performance improvement for training the LSTM model. Looking at the task from purely a classification perspective it may be possible to achieve better results. One limitation of the research is that the model has not been implemented in a practical or real time setting for predicting into the future as opposed to learning what has already happened. In addition, the ability to predict using streaming data should improve the model. Sliding window validation is an approach not implemented here but this may be explored as future work. One problem that will arise is that the data is inherently shrouded in noise.

The work in progress, includes implementing hyperparameter tuning, in order to get a more accurate network architecture. Also, other features can be considered (although from our experiments with Bitcoin, more features have not always led to better results). Microeconomic factors might be included in the model for a better predictive result. Anyway, maybe the data we gathered for Bitcoin, even though it has been collected through the years, might have become interesting, producing historic interpretations only in the last couple of years. Furthermore, a breakthrough evolution in peer-to-peer transactions is ongoing and transforming the landscape of payment services. While it seems, all doubts have not been settled, time might be perfect to act. We think it's difficult to give a mature thought on Bitcoin for the future.

# 9. Future Scope

• This can be made more accurate with adding more data set.

• More algorithms with better performance can add on to accuracy.

• It can host on web for real time analysis of exe files on the cloud.

# 10. References

1. Lu, W., Li, J., Wang, J., & Qin, L. (**2021**). A CNN-BiLSTM-AM method for stock price prediction. Neural Computing and Applications, 33(10), 4741-4753.
2. Wu, J. M. T., Li, Z., Srivastava, G., Tasi, M. H., & Lin, J. C. W. (**2021**). A graph‐based convolutional neural network stock price prediction with leading indicators. Software: Practice and Experience, 51(3), 628-644.
3. Rezaei, Hadi, Hamidreza Faaljou, and Gholamreza Mansourfar. "Stock price prediction using deep learning and frequency decomposition." Expert Systems with Applications 169 (**2021**): 114332.
4. Li, Jiake. "Research on Market Stock Index Prediction Based on Network Security and Deep Learning." Security and Communication Networks 2021 (**2021**).
5. Jing, Nan, Zhao Wu, and Hefei Wang. "A hybrid model integrating deep learning with investor sentiment analysis for stock price prediction." Expert Systems with Applications 178 (**2021**): 115019.
6. Wu, J. M. T., Li, Z., Herencsar, N., Vo, B., & Lin, J. C. W. (**2021**). A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. Multimedia Systems, 1-20.
7. Vijh, M., Chandola, D., Tikkiwal, V. A., & Kumar, A. (**2020**). Stock closing price prediction using machine learning techniques. Procedia Computer Science, 167, 599-606.
8. Yu, Pengfei, and Xuesong Yan. "Stock price prediction based on deep neural networks." Neural Computing and Applications 32.6 (**2020**): 1609-1628.
9. Song, Yoojeong, Jae Won Lee, and Jongwoo Lee. "A study on novel filtering and relationship between input-features and target-vectors in a deep learning model for stock price prediction." Applied Intelligence 49.3 (**2019**): 897-911.
10. Mohan, Saloni, et al. "Stock price prediction using news sentiment analysis." 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService). IEEE, **2019**.

# Appendix 1:

*!pip install darts==0.18.0 &> /dev/null*

*!pip install xarray==0.18.2 &> /dev/null  # required to read pickle files*

*!pip install xlrd==2.0.1 &> /dev/null*

*!pip install pyyaml==5.4.1*

*# Commented out IPython magic to ensure Python compatibility.*

*# %matplotlib inline*

*import warnings*

*warnings.filterwarnings("ignore")*

*import os*

*import time*

*import random*

*import pandas as pd*

*import pickle*

*import numpy as np*

*from tqdm.auto import tqdm*

*from datetime import datetime*

*from itertools import product*

*import torch*

*from torch import nn*

*from typing import List, Tuple, Dict*

*from sklearn.preprocessing import MaxAbsScaler*

*from sklearn.linear\_model import Ridge*

*import matplotlib.pyplot as plt*

*from darts import TimeSeries*

*from darts.utils.losses import SmapeLoss*

*from darts.dataprocessing.transformers import Scaler*

*from darts.metrics import smape*

*from darts.utils.utils import SeasonalityMode, TrendMode, ModelMode*

*from darts.models import \**

*!pip uninstall matplotlib*

*!pip install matplotlib==3.1.3*

*HORIZON = 18*

*def load\_m3() -> Tuple[List[TimeSeries], List[TimeSeries]]:*

*print("building M3 TimeSeries...")*

*# Read DataFrame*

*df\_m3 = pd.read\_excel("m3\_dataset.xls", "M3Month")*

*# Build TimeSeries*

*m3\_series = []*

*for row in tqdm(df\_m3.iterrows()):*

*s = row[1]*

*start\_year = int(s["Starting Year"])*

*start\_month = int(s["Starting Month"])*

*values\_series = s[6:].dropna()*

*if start\_month == 0:*

*continue*

*start\_date = datetime(year=start\_year, month=start\_month, day=1)*

*time\_axis = pd.date\_range(start\_date, periods=len(values\_series), freq="M")*

*series = TimeSeries.from\_times\_and\_values(*

*time\_axis, values\_series.values*

*).astype(np.float32)*

*m3\_series.append(series)*

*print("\nThere are {} monthly series in the M3 dataset".format(len(m3\_series)))*

*# Split train/test*

*print("splitting train/test...")*

*m3\_train = [s[:-HORIZON] for s in m3\_series]*

*m3\_test = [s[-HORIZON:] for s in m3\_series]*

*# Scale so that the largest value is 1*

*print("scaling...")*

*scaler\_m3 = Scaler(scaler=MaxAbsScaler())*

*m3\_train\_scaled: List[TimeSeries] = scaler\_m3.fit\_transform(m3\_train)*

*m3\_test\_scaled: List[TimeSeries] = scaler\_m3.transform(m3\_test)*

*print(*

*"done. There are {} series, with average training length {}".format(*

*len(m3\_train\_scaled), np.mean([len(s) for s in m3\_train\_scaled])*

*)*

*)*

*return m3\_train\_scaled, m3\_test\_scaled*

*def load\_air() -> Tuple[List[TimeSeries], List[TimeSeries]]:*

*# load TimeSeries*

*print("loading air TimeSeries...")*

*with open("passengers.pkl", "rb") as f:*

*all\_air\_series = pickle.load(f)*

*# Split train/test*

*print("splitting train/test...")*

*air\_train = [s[:-HORIZON] for s in all\_air\_series]*

*air\_test = [s[-HORIZON:] for s in all\_air\_series]*

*# Scale so that the largest value is 1*

*print("scaling series...")*

*scaler\_air = Scaler(scaler=MaxAbsScaler())*

*air\_train\_scaled: List[TimeSeries] = scaler\_air.fit\_transform(air\_train)*

*air\_test\_scaled: List[TimeSeries] = scaler\_air.transform(air\_test)*

*print(*

*"done. There are {} series, with average training length {}".format(*

*len(air\_train\_scaled), np.mean([len(s) for s in air\_train\_scaled])*

*)*

*)*

*return air\_train\_scaled, air\_test\_scaled*

*def load\_m4() -> Tuple[List[TimeSeries], List[TimeSeries]]:*

*# load TimeSeries - the splitting and scaling has already been done*

*print("loading M4 TimeSeries...")*

*with open("m4\_monthly\_scaled.pkl", "rb") as f:*

*m4\_series = pickle.load(f)*

*# filter and keep only series that contain at least 48 training points*

*m4\_series = list(filter(lambda t: len(t[0]) >= 48, m4\_series))*

*m4\_train\_scaled, m4\_test\_scaled = zip(\*m4\_series)*

*print(*

*"done. There are {} series, with average training length {}".format(*

*len(m4\_train\_scaled), np.mean([len(s) for s in m4\_train\_scaled])*

*)*

*)*

*return m4\_train\_scaled, m4\_test\_scaled*

*"""Finally, we define a handy function to tell us how good a bunch of forecasted series are:"""*

*def eval\_forecasts(*

*pred\_series: List[TimeSeries], test\_series: List[TimeSeries]*

*) -> List[float]:*

*print("computing sMAPEs...")*

*smapes = smape(test\_series, pred\_series)*

*plt.figure()*

*plt.hist(smapes, bins=50)*

*plt.ylabel("Count")*

*plt.xlabel("sMAPE")*

*plt.title("Median sMAPE: %.3f" % np.median(smapes))*

*plt.show()*

*plt.close()*

*return smapes*

*air\_train, air\_test = load\_air()*

*min([len(s) for s in air\_train])*

*def eval\_local\_model(*

*train\_series: List[TimeSeries], test\_series: List[TimeSeries], model\_cls, \*\*kwargs*

*) -> Tuple[List[float], float]:*

*preds = []*

*start\_time = time.time()*

*for series in tqdm(train\_series):*

*model = model\_cls(\*\*kwargs)*

*model.fit(series)*

*pred = model.predict(n=HORIZON)*

*preds.append(pred)*

*elapsed\_time = time.time() - start\_time*

*smapes = eval\_forecasts(preds, test\_series)*

*return smapes, elapsed\_time*

*!pip install -q yfinance*

*from pandas\_datareader.data import DataReader*

*import yfinance as yf*

*# For time stamps*

*from datetime import datetime*

*from darts import TimeSeries*

*from darts.models import NBEATSModel*

*from darts.dataprocessing.transformers import Scaler, MissingValuesFiller*

*from darts.metrics import mape, r2\_score*

*from darts.datasets import EnergyDataset*

*end = datetime.now()*

*start = datetime(end.year - 1, end.month, end.day)*

*AAPL = yf.download('AAPL', start, end)*

*AAPL.tail()*

*AAPL=AAPL.asfreq('d')*

*AAPL["Close"].plot()*

*plt.title("Apple Closing Price")*

*AAPL\_day\_avg = AAPL.groupby(AAPL.index.astype(str).str.split(" ").str[0]).mean().reset\_index()*

*filler = MissingValuesFiller()*

*scaler = Scaler()*

*series = scaler.fit\_transform(*

*filler.transform(*

*TimeSeries.from\_dataframe(*

*AAPL\_day\_avg, "Date", ["Close"]*

*)*

*)*

*).astype(np.float32)*

*series.plot()*

*plt.title("Apple Closing Price")*

*series.head()*

*train, val = series.split\_after(pd.Timestamp("20220101"))*

*naive1\_smapes, naive1\_time = eval\_local\_model(air\_train, air\_test, NaiveSeasonal, K=1)*

*ets\_smapes, ets\_time = eval\_local\_model(air\_train, air\_test, ExponentialSmoothing)*

*naive12\_smapes, naive12\_time = eval\_local\_model(*

*air\_train, air\_test, NaiveSeasonal, K=12*

*)*

*naive12\_smapes, naive12\_time = eval\_local\_model(*

*air\_train, air\_test, NaiveLast, K=12*

*)*

*theta\_smapes, theta\_time = eval\_local\_model(air\_train, air\_test, Theta, theta=1.5)*

*warnings.filterwarnings("ignore")  # ARIMA generates lots of warnings*

*arima\_smapes, arima\_time = eval\_local\_model(air\_train, air\_test, ARIMA, p=12, d=1, q=1)*

*kf\_smapes, kf\_time = eval\_local\_model(air\_train, air\_test, KalmanForecaster, dim\_x=12)*

*"""### Comparing models*

*def plot\_models(method\_to\_elapsed\_times, method\_to\_smapes):*

*shapes = ["o", "s", "\*"]*

*colors = ["tab:blue", "tab:orange", "tab:green", "tab:red", "tab:purple"]*

*styles = list(product(shapes, colors))*

*plt.figure(figsize=(6, 6), dpi=100)*

*for i, method in enumerate(method\_to\_elapsed\_times.keys()):*

*t = method\_to\_elapsed\_times[method]*

*s = styles[i]*

*plt.semilogx(*

*[t],*

*[np.median(method\_to\_smapes[method])],*

*s[0],*

*color=s[1],*

*label=method,*

*markersize=13,*

*)*

*plt.xlabel("elapsed time [s]")*

*plt.ylabel("median sMAPE over all series")*

*plt.legend(bbox\_to\_anchor=(1.4, 1.0), frameon=True);*

*smapes = {*

*"naive-last": naive1\_smapes,*

*"naive-seasonal": naive12\_smapes,*

*"Exponential Smoothing": ets\_smapes,*

*"Theta": theta\_smapes,*

*"ARIMA": arima\_smapes,*

*"Kalman Filter": kf\_smapes,*

*}*

*elapsed\_times = {*

*"naive-last": naive1\_time,*

*"naive-seasonal": naive12\_time,*

*"Exponential Smoothing": ets\_time,*

*"Theta": theta\_time,*

*"ARIMA": arima\_time,*

*"Kalman Filter": kf\_time,*

*}*

*plot\_models(elapsed\_times, smapes)*

*# For first three algorithms*

*from google.colab import drive*

*drive.mount('/content/drive')*

*import numpy as np # linear algebra*

*import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)*

*import matplotlib.pyplot as plt*

*import warnings*

*warnings.filterwarnings("ignore")*

*bit\_data=pd.read\_csv("/content/drive/MyDrive/DM/bitstampUSD\_1-min\_data\_2012-01-01\_to\_2019-03-13.csv")*

*bit\_data["date"]=pd.to\_datetime(bit\_data["Timestamp"],unit="s").dt.date*

*group=bit\_data.groupby("date")*

*data=group["Close"].mean()*

*bit\_data.shape*

*bit\_data.tail(10)*

*data.shape*

*data.isnull().sum()*

*close\_train=data.iloc[:len(data)-50]*

*close\_test=data.iloc[len(close\_train):]*

*#feature scalling (set values between 0-1)*

*close\_train=np.array(close\_train)*

*close\_train=close\_train.reshape(close\_train.shape[0],1)*

*from sklearn.preprocessing import MinMaxScaler*

*scaler=MinMaxScaler(feature\_range=(0,1))*

*close\_scaled=scaler.fit\_transform(close\_train)*

*timestep=50*

*x\_train=[]*

*y\_train=[]*

*for i in range(timestep,close\_scaled.shape[0]):*

*x\_train.append(close\_scaled[i-timestep:i,0])*

*y\_train.append(close\_scaled[i,0])*

*x\_train,y\_train=np.array(x\_train),np.array(y\_train)*

*x\_train=x\_train.reshape(x\_train.shape[0],x\_train.shape[1],1) #reshaped for RNN*

*print("x\_train shape= ",x\_train.shape)*

*print("y\_train shape= ",y\_train.shape)*

*from keras.models import Sequential*

*from keras.layers import Dense, SimpleRNN, Dropout,Flatten*

*regressor=Sequential()*

*#first RNN layer*

*regressor.add(SimpleRNN(128,activation="relu",return\_sequences=True,input\_shape=(x\_train.shape[1],1)))*

*regressor.add(Dropout(0.25))*

*#second RNN layer*

*regressor.add(SimpleRNN(256,activation="relu",return\_sequences=True))*

*regressor.add(Dropout(0.25))*

*#third RNN layer*

*regressor.add(SimpleRNN(512,activation="relu",return\_sequences=True))*

*regressor.add(Dropout(0.35))*

*#fourth RNN layer*

*regressor.add(SimpleRNN(256,activation="relu",return\_sequences=True))*

*regressor.add(Dropout(0.25))*

*#fifth RNN layer*

*regressor.add(SimpleRNN(128,activation="relu",return\_sequences=True))*

*regressor.add(Dropout(0.25))*

*#convert the matrix to 1-line*

*regressor.add(Flatten())*

*#output layer*

*regressor.add(Dense(1))*

*regressor.compile(optimizer="adam",loss="mean\_squared\_error")*

*regressor.fit(x\_train,y\_train,epochs=20,batch\_size=64)*

*inputs=data[len(data)-len(close\_test)-timestep:]*

*inputs=inputs.values.reshape(-1,1)*

*inputs=scaler.transform(inputs)*

*x\_test=[]*

*for i in range(timestep,inputs.shape[0]):*

*x\_test.append(inputs[i-timestep:i,0])*

*x\_test=np.array(x\_test)*

*x\_test=x\_test.reshape(x\_test.shape[0],x\_test.shape[1],1)*

*predicted\_data=regressor.predict(x\_test)*

*predicted\_data=scaler.inverse\_transform(predicted\_data)*

*data\_test=np.array(close\_test)*

*data\_test=data\_test.reshape(len(data\_test),1)*

*plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')*

*plt.plot(data\_test,color="r",label="true result")*

*plt.plot(predicted\_data,color="b",label="predicted result")*

*plt.legend()*

*plt.xlabel("Time(50 days)")*

*plt.ylabel("Close Values")*

*plt.grid(True)*

*plt.show()*

*from sklearn.metrics import mean\_absolute\_error*

*from keras.models import Sequential*

*from keras.layers import Dense, LSTM, Dropout,Flatten*

*model=Sequential()*

*model.add(LSTM(10,input\_shape=(None,1),activation="relu"))*

*model.add(Dense(1))*

*model.compile(loss="mean\_squared\_error",optimizer="adam")*

*model.fit(x\_train,y\_train,epochs=60,batch\_size=32)*

*inputs=data[len(data)-len(close\_test)-timestep:]*

*inputs=inputs.values.reshape(-1,1)*

*inputs=scaler.transform(inputs)*

*x\_test=[]*

*for i in range(timestep,inputs.shape[0]):*

*x\_test.append(inputs[i-timestep:i,0])*

*x\_test=np.array(x\_test)*

*x\_test=x\_test.reshape(x\_test.shape[0],x\_test.shape[1],1)*

*predicted\_data=model.predict(x\_test)*

*predicted\_data=scaler.inverse\_transform(predicted\_data)*

*data\_test=np.array(close\_test)*

*data\_test=data\_test.reshape(len(data\_test),1)*

*plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')*

*plt.plot(data\_test,color="r",label="true result")*

*plt.plot(predicted\_data,color="b",label="predicted result")*

*plt.legend()*

*plt.xlabel("Time(50 days)")*

*plt.ylabel("Close Values")*

*plt.grid(True)*

*plt.show()*

*data=pd.read\_csv("/content/drive/MyDrive/DM/bitstampUSD\_1-min\_data\_2012-01-01\_to\_2019-03-13.csv")*

*data["Timestamp"] = pd.to\_datetime(data["Timestamp"], infer\_datetime\_format=True, unit="s")*

*data = data.set\_index("Timestamp")*

*# Considering data of last 70 days only*

*freq = 55*

*c = int(60/freq)*

*days = 70*

*data = data.tail(days\*24\*60)*

*#plotfig(data)*

*print(data.shape)*

*#taking interval of 10 minutes*

*data = data[::freq]*

*data.dropna(axis=0,inplace=True)*

*data = data["Weighted\_Price"]*

*data = data.values.reshape(-1,1)*

*print("No. of days: ",data.shape[0]/(24\*c))*

*#scaling*

*scaler\_train = MinMaxScaler(feature\_range=(0, 1))*

*data = scaler\_train.fit\_transform(data)*

*# Train-test split*

*train\_split\_time = 30\*24\*c*

*test\_split\_time = 50\*24\*c*

*train\_data = data[:train\_split\_time]*

*valid\_data = data[train\_split\_time:test\_split\_time]*

*test\_data = data[test\_split\_time:]*

*window\_size = 30*

*batch\_size = 32*

*shuffle\_buffer\_size = 1000*

*print(train\_data.shape)*

*print(valid\_data.shape)*

*print(test\_data.shape)*

*def x\_y\_split(series,lookback):*

*X\_data, Y\_data = [], []*

*for i in range(len(series) - lookback):*

*X\_data.append(series[i:i+lookback])*

*Y\_data.append(series[i+lookback])*

*return np.array(X\_data), np.array(Y\_data)*

*lookback = 15\*24\*c*

*x\_train, y\_train = x\_y\_split(train\_data,lookback)*

*x\_valid, y\_valid = x\_y\_split(valid\_data,lookback)*

*x\_test, y\_test = x\_y\_split(test\_data,lookback)*

*print(x\_train.shape)*

*print(y\_train.shape)*

*print(x\_valid.shape)*

*print(y\_valid.shape)*

*print(x\_test.shape)*

*print(y\_test.shape)*

*import tensorflow as tf*

*from keras.models import Sequential*

*from keras.layers import Dense, Dropout, LSTM, GRU, Bidirectional*

*from tensorflow.keras.optimizers import Adam*

*import math*

*tf.keras.backend.clear\_session()*

*regressor = Sequential()*

*regressor.add(LSTM(units=128, return\_sequences=True, input\_shape=(x\_train.shape[1],1)))*

*regressor.add(Dropout(0.3))*

*regressor.add(LSTM(units=64))*

*regressor.add(Dense(units=1))*

*opt = tf.keras.optimizers.Adam(learning\_rate=0.01)*

*regressor.compile(loss='mean\_squared\_error', optimizer=opt)*

*# fit to the training set*

*history = regressor.fit(x\_train, y\_train, epochs=50, batch\_size=16, validation\_data=(x\_valid, y\_valid))*

*import math*

*from sklearn.metrics import mean\_squared\_error*

*def plot\_predictions(test, predicted):*

*plt.plot(test, color="red", label="real")*

*plt.plot(predicted, color="blue", label="predicted")*

*plt.title("price prediction")*

*plt.xlabel("time")*

*plt.ylabel("price")*

*plt.legend()*

*plt.show()*

*def return\_rmse(test, predicted):*

*rmse = math.sqrt(mean\_squared\_error(test, predicted))*

*print("the root mean squared error is : {}.".format(rmse))*

*# Test data visualization*

*y\_test\_pred = regressor.predict(x\_test)*

*return\_rmse(y\_test,y\_test\_pred)*

*plot\_predictions(y\_test,y\_test\_pred)*