

BITCOIN PRICE PREDICTION

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

Santosh Kumar Singh (38110682)
Vikash Kumar Gupta (38110643)



DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Santosh Kumar Singh (38110682), Vikash Kumar Gupta (38110643) who carried out the project entitled "BITCOIN PRICE PREDICTION" under my supervision from November 2021 to May 2022.

Internal Guide

Dr.R.Sethuraman,B.E.,M.E.,Ph.d.

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on MAY 2022

Internal Examiner

External Examiner

DECLARATION

I, Santosh Kumar Singh (38110682), Vikash Kumar Gupta (38110643) hereby declare that the Project Report entitled "BITCOIN PRICE PREDICTION" done by me under the guidance of Dr.R.Sethuraman,B.E.,M.E.,Ph.d. is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Computer Science Engineering.

DATE: 5-05-2022

Santosh Kumar Singh (38110682)

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to Dr. **T.Sasikala M.E., Ph.D.**, Dean, School of Computing and **Dr.S.Vigneshwari, M.E., Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide : **Dr.R.Sethuraman,B.E.,M.E.,Ph.d.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

The purpose of this study is to find out with what accuracy the direction of the price of Bitcoin can be predicted using machine learning methods. This is basically a time series prediction problem. While much research exists surrounding the use of different machine learning.

Techniques for time series prediction, research in this area relating specifically to Bitcoin is lacking. In addition, Bitcoin as a currency is in a transient stage and as a result is considerably more volatile than other currencies such as the USD. Interestingly, it is the top performing currency four out of the last five years. Thus, its prediction offers great potential and this provides motivation for research in the area. As evidenced by an analysis of the existing literature, running machine learning algorithms on a GPU as opposed to a CPU can offer significant performance improvements. This is explored by benchmarking the training of the RNN and LSTM network using both the GPU and CPU. This provides a solution to the sub research topic.

Finally, in analysing the chosen dependent variables, each variables importance is assessed using a random forest algorithm. In addition, the ability to predict the direction of the price of an asset such as Bitcoin offers the opportunity for profit to be made by trading the asset.

Keywords: Bitcoin Prediction, Time complexity, Machine-learning, Database architecture, RNN, LSTM.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	5
1	INTRODUCTION	7
	1.1 Domain introduction	8
	1.2 Project purpose	9
2	LITERATURE SURVEY	14
3	AIM AND SCOPE OF PRESENT	15
	3.1 Aim	16
	3.2 Scope Of Present Investigation	17
4	METHODS AND ALGORITHMS	33
	4.1 Methods	34
	4.2 Algorithm	35
5	RESULTS AND DISCUSSION	56
6	CONCLUSION	57
	REFERENCES	58

CHAPTER 1

INTRODUCTION

1.1 DOMAIN SPECIFIC INTRODUCTION

Time series prediction is not a new phenomenon. Prediction of most financial markets such as the stock market has been researched at large scale. Bitcoin presents an interesting parallel to this as it is a time series prediction problem in a market still in its beginning stage. As a result, there is high volatility in the market and this provides an opportunity in terms of prediction. In addition, Bitcoin is the leading cryptocurrency in the world with adoption growing consistently over time. Due to the open nature of Bitcoin it also poses another difficulty as opposed to traditional financial markets. It operates on a decentralised, peer-to-peer and trustless system in which all transactions are posted to an open ledger called the Blockchain. This type of transparency is not seen in other financial markets. 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 predicting 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. Tasks such as natural language processing which are also sequential in nature and have shown promising results. This type of task uses data of a sequential nature and as a result is similar to a price prediction task. The recurrent neural network (RNN) and the long short term memory (LSTM) flavour of artificial neural networks are favoured over the traditional multilayer perceptron (MLP) due to the temporal nature of the more advanced algorithms. The aim of this research is to ascertain with what accuracy can the price of Bitcoin be predicted using machine learning.

the research variables. A brief overview of Bitcoin, machine learning and time series analysis concludes section one. Section two examines related work in the area of both Bitcoin price prediction and other financial time series prediction. Literature on using machine learning to predict Bitcoin price is limited.

Out of approximately 653 papers published on Bitcoin only 7 have related to machine learning for prediction. As a result, literature relating to other financial time series prediction using deep learning is also assessed as these tasks can be considered analogous.

1.2 PROBLEM DEFINITION

The popularity of cryptocurrencies has skyrocketed in 2017 due to several consecutive months of super exponential growth of their market capitalization, which peaked at more than \$800 billions in Jan. 2018. Today, there are more than 1,500 actively traded crypto currencies. Between 2.9 and 5.8 millions of private as well as institutional investors are in the different transaction networks, according to a recent survey , and access to the market has become easier over time. Major cryptocurrencies can be bought using fiat currency in a number of online exchanges and then be used in their turn to buy less popular cryptocurrencies. The volume of daily exchanges is currently superior to \$15 billions. Since 2017, over 170 hedge funds specialised in cryptocurrencies have emerged and Bitcoin futures have been launched to address institutional demand for trading and hedging Bitcoin could be effective also in predicting crypto currency prices. However, the application of machine learning algorithms to the cryptocurrency market has been limited so far to the analysis of Bitcoin prices, using random forests, Bayesian neural network , long short-term memory neural network, and other algorithms .The studies were able to anticipate, to different degrees, the price fluctuations of Bitcoin, and revealed that best results were achieved by neural network based algorithms. Deep reinforcement learning was showed to beat the uniform buy and hold strategy in predicting the prices of 12 cryptocurrencies over one-year period.

The Bitcoin's value varies just like any other stock . There are many algorithms used on stock market data for price forecast. However, the parameters affecting Bitcoin are different. Therefore it is necessary to foretelling the value of Bitcoin so that correct investment decisions can be made. The price of Bitcoin does not depend on the business events or intervening government authorities, unlike the stock market. Thus, to forecast the value we feel it is necessary to leverage machine learning technology to predict the price of Bitcoin. So the project aim is to predict the price of bitcoin and help investor's make better investments. This research is concerned with predicting the price of Bitcoin using machine learning. The goal is to ascertain with what accuracy can the direction of Bitcoin price in USD can be predicted.

The price data is sourced from the Bitcoin Price index. The task is achieved with varying degrees of success through the implementation of a Bayesian optimized recurrent neural network (RNN) and Long Short-Term Memory (LSTM) network.

1.3 PROJECT PURPOSE

The purpose of this study is to find out with what accuracy the direction of the price of Bitcoin can be predicted using machine learning methods. This is basically a time series prediction problem. While much research exists surrounding the use of different machine learning techniques for time series prediction, research in this area relating specifically to Bitcoin is lacking. In addition, Bitcoin as a currency is in a transient stage and as a result is considerably more volatile than other currencies such as the USD. Interestingly, it is the top performing currency four out of the last five years¹. Thus, its prediction offers great potential and this provides motivation for research in the area. As evidenced by an analysis of the existing literature, running machine learning algorithms on a GPU as opposed to a CPU can offer significant performance improvements. This is explored by benchmarking the training of the RNN and LSTM network using both the GPU and CPU. This provides a solution to the sub research topic.

Finally, in analysing the chosen dependent variables, each variables importance is assessed using a random forest algorithm. In addition, the ability to predict the direction of the price of an asset such as Bitcoin offers the opportunity for profit to be made by trading the asset. To implement a full trading strategy based on the results of the models is worthy of a dissertation in itself and as a result this paper will focus solely on the accuracy in which direction the price can be predicted. In basic terms, the model would initiate a short position if the price was predicted to go up and a long position if the price was predicted to go down. Several Bitcoin exchanges offer margin trading accounts to facilitate this too. The profitability of this strategy would be based not only on the accuracy of the model, but also on the size of the positions taken. This is outside the scope of this research but could be addressed in future work.

While we will try to build a predictive model for the Bitcoin price value calculator, we are aware in advance that price may differ greatly because of internal and external factors to Bitcoin. By internal factors we are presuming factors inside the Bitcoin security.

By external we are referring to agents which influence indirectly the price of Bitcoin (exchange closures, replacing cryptocurrencies, speculation markets, the fact that as its believed widely over 80% of Bitcoins in circulation is concentrated in a limited number of investors etc.) Anyway, we shall compare our results to other models built for cryptocurrency prediction. Let's not forget that in the first month of 2018 there were models which predicted that Bitcoin would surpass the 100,000.00 USD per Bitcoin till the end of the year, while we are barely reaching the 7,000.00 USD value just 2 months before the end of the year.

1.4 PROJECT FEATURES

The main feature of this system is to propose a general and effective approach to predict the bitcoin price using data mining techniques. The main goal of the proposed system is to analyze and study the hidden patterns and relationships between the data present in the bitcoin dataset. The solution to the bitcoin analysis problem can provide extremely useful information to prevent investors from losing money which is being invested on bitcoin. Most of the existing work solves these problems separately by different models. so dealing with this becomes more important. The analysis and prediction plays an important role in the problem definition.

The constant increase in bitcoin usage has become an extremely serious problem, with the development of technology and hi-tech tools having a significantly greater impact on the bitcoin price. The large amounts of information also poses a challenge to analyze such data and identify similarities or relations between the data. Also there is a challenge of inconsistency that can occur in the data due to incompleteness in the dataset. Therefore, there is an urging need of proper techniques to analyze large volumes of data to get some useful results out of it. So the main aim of this project is to propose a general and effective approach to predict the bitcoin price using data mining techniques.

The main features of the proposed system are:

- More efficient.
- Better bitcoin price monitoring systems.
- Reduces the costs of storage, maintenance and personnel.
- It reduces the time complexity of the system.
- System that has a simpler architecture to understand.
- Processing of large amount of data becomes easier.

1.5 MODULES DESCRIPTION

1.5.1 DATA GATHERING

The first step in this project or in any data mining project is the collection of data to be studied or examined to find the hidden relationships between the data members. The important concern while choosing a dataset is that the data which we are gathering should be relevant to the problem statement and it must be large enough so that the inference derived from the data is useful to extract some important patterns between the data such that they can be used to predict the future events or can be studied for further analysis. The result of the process of gathering and creating a collection of data results into what we call as a Dataset. The dataset contains large volume of data that can be analyzed to get some knowledge from the databases. This is an important step in the process because choosing the inappropriate dataset can lead us to incorrect results.

1.5.2 DATA PREPROCESSING

The primary data collected from the internet resources remains in the raw form of statements, digits and qualitative terms. The raw data contains error, omissions and inconsistencies. It requires corrections after careful scrutinizing the completed questionnaires. The following steps are involved in the processing of primary data. A huge volume of raw data collected through field survey needs to be grouped for similar details of individual responses.

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

Therefore, certain steps are executed to convert the data into a small clean data set. This technique is performed before the execution of Iterative Analysis. The set of steps is known as data preprocessing.

The process comprises:

- Data Cleaning
- Data Integration
- Data Transformation
- Data Reduction

Data Preprocessing is necessary because of the presence of unformatted real world data. Mostly real world data is composed of:

- **Inaccurate data (missing data)** - There are many reasons for missing data such as data is not continuously collected, a mistake in data entry, technical problems with biometrics and much more.
- **The presence of noisy data (erroneous data and outliers)** - The reasons for the existence of noisy data could be a technological problem of gadget that gathers data, a human mistake during data entry and much more.
- **Inconsistent data** - The presence of inconsistencies are due to the reasons such that existence of duplication within data, human data entry, containing mistakes in codes or names, i.e., violation of data constraints and much more.

1.5.3 CLASSIFICATION

This technique is used to divide various data into different classes. This process is also similar to clustering. It segments data records into various segments which are known as classes. Unlike clustering, here we have knowledge of different clusters. Ex: Outlook email, they have an algorithm to categorize an email as legitimate or spam.

CHAPTER 2

LITERATURE SURVEY

2.1 DATA MINING

Literature survey is that the most vital step in code development method. Before developing the tool it's necessary to see the time issue, economy and company strength. Once these things are satisfied, then next steps is to determine which operating system and language can be used for developing the tool Once the programmers begin building the tool the programmers would like heap of external support. This support is obtained from senior programmers, from book or from websites Before building the system the on top of thought area unit taken under consideration for developing the projected system.. We have to analyze the Data mining Outline Survey:

2.1.1 Data Mining Survey

Data mining is a data analysis technique which allows us to study and identify different patterns and relationships between the data. In other words, data mining is a technique which can be employed to extract information from large and extensive datasets and convert the information into a prominent structure so that it can be used further for gaining inference and knowledge on the data so as to prevent the crimes.

Data mining contains techniques for analysis which involve various domains. For instance, some of the domains involved in data mining are Statistics, Machine Learning and Database systems. Data mining is additionally spoken as “Knowledge discovery in databases (KDD)”.

The real task of data mining systems is the semi-automatic or automatic analysis of large volumes of data to extract previously unknown relationships such as groups of data members(clustering analysis),unusual records(outlier or anomaly detection),and dependencies. Normally, this includes database techniques like spatial indices.

These relationships that are discovered can be used as input data or may also be used in depth analysis for example, in machine learning or predictive analysis.

Data mining may identify multiple groups in the data, that can be put to further use for accurate predictions by a decision support system.

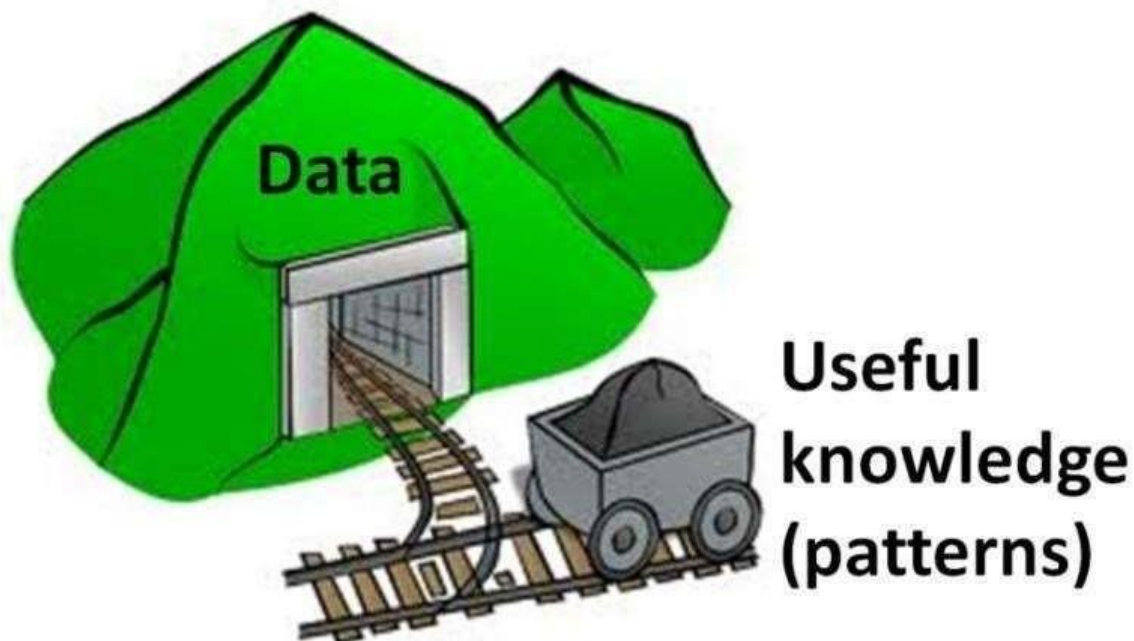


Fig 2.1: Data Mining

2.1.2 Stages in Data Mining

There are 4 major steps in data mining which are described as follows:

1. Data Sources: This stage includes gathering the data or making a dataset on which the analysis or the study has to performed. The datasets can be of many forms for instance, they can be news letters, databases, excel sheets or various other sources like websites, blogs, social media. An appropriate dataset must be chosen in order to perform an efficient study or analysis. The dataset must be chosen which is appropriate and well suited with respect to the problem definition.

2. Data Exploration: This step includes preparing the data properly for analysis and study. This step is mainly focused on cleaning the data and removing the anomalies from the data. As there is a large amount of data there is always a great chance that some of the data might be missing or some data might be wrong. Thus, for efficient analysis we require the data to be maintained properly. So this process includes removing the incorrect data and replacing the data which is missing with either mean or median of the whole data. This step is also generally known as data pre-processing.

3. Data Modeling: In this step the relationships and patterns that were hidden in the data are examined and extracted from the datasets. The data can be modeled based on the technique that is being used. Some of the different techniques that can be used for modeling data are: clustering, classification, association and decision trees.

Deploying Models: Once the relationships and patterns present in the data are discovered we need to put that knowledge to use. These patterns can be used to predict events in the future and also they can be used for further analysis. The discovered patterns can be used as inputs for machine learning and predictive analysis for the datasets.



Fig 2.2: Stages in Data Mining

2.1.3 Techniques in Data Mining:

1. Classification: This technique is used to divide various data into different classes. This process is also similar to clustering. It segments data records into various segments which are known as classes. Unlike clustering, here we have knowledge of different clusters. Ex: Outlook email, they have an algorithm to categorize an email as legitimate or spam.

2. Association: This technique is used to discover hidden patterns in the data and also for identifying interesting relations between the variables in a database. Ex: It is used in retail industry.

3. Prediction: This technique is used only for particular uses. It is used extract relationships between independent and dependent variables in the dataset. Ex: We use this technique to predict profit obtained from sales for the future.

4. Clustering: A cluster is referred to as a group of data objects. The data objects that are similar in properties are kept in the same cluster. In other words we can tell that clustering is a process of discovering groups or clusters.

5. Here we do not have prior knowledge of the clusters. Ex: It can be used in consumer profiling.

6. Sequential Patterns: This is an essential aspect of data mining techniques its main aim is to discover similar patterns in the dataset. Ex: E-commerce websites suggestions are based on what we have bought previously.

7. Decision Trees: This technique is a vital role in data mining because it is easier to understand for the users. The decision tree begins with a root which is a simple question. As they can have multiple answers we get our nodes of the decision tree also the questions in the root node might lead to another set of questions. Thus, the nodes keep adding in the decision tree. At last, we are allowed to make a final decision on it. Apart from these techniques there are certain other techniques which allow us to remove noisy data and also clean the dataset. This helps us to get accurate analysis and prediction results.



Fig 2.3: Data Mining Techniques

2.1.4 Benefits of Data Mining:

Data mining has various uses in various sectors of the society:

- In finance sector, it can be used for modeling risks accurately regarding loans and other facilities.
- In marketing, it can be used for predicting profits and also can be used for creating targeted advertisements for various customers.
- In retail sector, it is used for improving consumer experience and also increasing the amount of profits.
- Tax governing organizations use it to determine frauds in transactions.

2.2 EXISTING SYSTEM

Nowadays Investors and Researchers trying to understand the fluctuation in prices of cryptocurrencies, it is important to have a system that can help to predict the change in prices on daily basis. Like the stock exchange bitcoin price change is quite volatile and can be difficult to get a high accurate prediction. The value of bitcoin or any other cryptocurrency cannot be static and can vary for every second. The fluctuation is completely dependent on the amount being paid for bitcoin by buyers. As bitcoin is used as an investment, the same principle applied in stocks for buying cheap and selling at a high price is applicable for cryptocurrency. The volatile nature of the cryptocurrency makes it much more challenging and interesting for analysts and investors to predict the accurate price.

The prediction and approximation of bitcoin prices is an area where much research has not been done. Since investors are keen to know the direction of cryptocurrency price

i.e. high or low it is vital to have an algorithm that gives the best accuracy in terms of determining the range. A lot of work and research has been done in trying to predict the direction of stock prices and very less in terms of cryptocurrency.

2.2.1 Disadvantages:

- Storing of large amounts of data that contains a lot of information about Bit coin price is posing a challenge for the Researchers and the Investors.
- Sometimes the data is entered manually and humans can make mistakes, so there are chances of incorrect data being entered in the dataset which can lead to inaccurate results while analyzing the data.
- In such a large dataset, there is always a chance of some fields containing missing values, these missing values can make the data noisy and thus we must take appropriate measures to remove inconsistency from the datasets.
- The Investors and the researchers do not have adequate techniques to analyze and study the data to get some inference out of it and use this inference to efficiently predict the price of the bitcoin.

In some researches Omkar S. Deorukhkar¹, Shrutika H. Lokhande², Vanishree R. Nayak³, Amit A. Chougule Uses the simple Feedforward Neural Network for Bit coin price prediction is not so efficient and accurate when it comes to time series or problems which involve dealing with data that is sequential. This is because at fundamental level in a feedforward neural network, the neuron does not consider learning from past data. Hence, Recurrent Neural Networks were designed to overcome this problem. RNNs consider previous time step output along with the current input state. This makes them better than Feedforward neural networks in learning sequential data.

2.3 PROPOSED SYSTEM

The proposed system implements machine algorithm to build the model to predict the price of the bit-coin based on historical dataset available on online database. In the proposed model, The bitcoin price prediction can be done using the LSTM(Long Short Term Memory) is one of the type of the RNN (Recurrent Neural Networks). The tool used for project are anaconda-navigator. The procedure to be followed for the proposed system is given as follows:

- First, collect the data set using the Rest-API to collect the historic of the bit-coin prices from the online database.
- Arrange the data into the data frame according to the problem definition, so as to get analysis correct and produce the results which are efficient to meet goals of the system.
- Then the rows of the dataset which are outdated for analysis/prediction to build a model and in-order to feed the relevant data to the model extra columns are removed and stored into a CSV file.
- Then data-preprocessing is performed to missing values for the attributes, this done to reduce the noise and inconsistency in the data.
- Then we Build the model for the data-set using the LSTM (RNN) algorithm to predict values of bit-coin on daily basis.
- Test the predictions with different layers of the RNN Model.

2.4 SOFTWARE DESCRIPTION

2.4.1 JUPYTER NOTEBOOK

The Jupyter Notebook App is a server-customer application that permits altering and running note pad records by means of an internet browser. The Jupyter Notebook App can be executed on a nearby work area requiring no web access (as portrayed in this report) or can be introduced on a remote server and got to through the web. Notwithstanding showing/altering/running note pad archives, the Jupyter Notebook App has a "Dashboard" (Notebook Dashboard), a "control board" indicating nearby records and permitting to open note pad reports or closing down their portions.

A scratch pad part is a "computational motor" that executes the code contained in a Notebook record. The ipython part, referenced in this guide, executes python code.

Portions for some, different dialects exist (official parts). When you open a Notebook report, the related part is consequently propelled. At the point when the scratch pad is executed (either cell-by-cell or with menu Cell - > Run All), the portion plays out the calculation and produces the outcomes. Contingent upon the sort of calculations, the piece may expend critical CPU and RAM.

Note that the RAM isn't discharged until the part is closed down, the Notebook Dashboard is the part which is indicated first when you dispatch Jupyter Notebook App. The Notebook Dashboard is essentially used to open note pad archives, and to deal with the running portions (picture and shutdown).

The Notebook Dashboard has different highlights like a record director, in particular exploring organizers and renaming/erasing documents.

2.4.2 MATPLOTLIB

People are exceptionally visual animals: we comprehend things better when we see things envisioned. Notwithstanding, the progression to showing investigations, results or bits of knowledge can be a bottleneck: you probably won't realize where to begin or you may have as of now a correct configuration as a top priority, however then inquiries like "Is this the correct method to imagine the bits of knowledge that I need to convey to my group of onlookers?" will have unquestionably gone over your brain.

When you're working with the Python plotting library Matplotlib, the initial step to responding to the above inquiries is by structure up information on themes like: The life structures of a Matplotlib plot: what is a subplot? What are the Axes? What precisely is a figure?

Plot creation, which could bring up issues about what module you precisely need to import (pylab or pyplot?), how you precisely ought to approach instating the figure and the Axes of your plot, how to utilize matplotlib in Jupyter note pads, and so on.

Plotting schedules, from straightforward approaches to plot your information to further developed methods for picturing your information. Essential plot customizations, with an emphasis on plot legends and content, titles, tomahawks marks and plotformat.

Sparing, appearing, ... your plots: demonstrate the plot, spare at least one figures to, for instance, pdf documents, clear the tomahawks, clear the figure or close the plot, and so on.

In conclusion, you'll quickly cover two manners by which you can alter Matplotlib: with templates and the rc settings.

Since all is set for you to begin plotting your information, it's an ideal opportunity to investigate some plotting schedules. You'll regularly go over capacities like `plot()` and `disperse()`, which either draw focuses with lines or markers interfacing them, or draw detached focuses, which are scaled or shaded. In any case, as you have just found in the case of the primary area, you shouldn't neglect to pass the information that you need these capacities to utilize!

These capacities are just the exposed rudiments. You will require some different capacities to ensure your plots look magnificent:

<code>ax.bar()</code>	Vertical rectangles
<code>ax.barh()</code>	Horizontal rectangles
<code>ax.axhline()</code>	Horizontal line across axes
<code>ax.vline()</code>	Vertical line across axes
<code>ax.fill()</code>	Filled polygons
<code>ax.fill_between()</code>	Fill between y-values and 0
<code>ax.stackplot()</code>	Stack plot

2.4.3 NUMPY

NumPy is, much the same as SciPy, Scikit-Learn, Pandas, and so forth one of the bundles that you can't miss when you're learning information science, principally in light of the fact that this library gives you a cluster information structure that holds a few advantages over Python records, for example, being increasingly reduced, quicker access in perusing and composing things, being progressively advantageous and increasingly productive.

NumPy exhibits are somewhat similar to Python records, yet at the same time particularly unique in the meantime. For those of you who are new to the subject, how about we clear up what it precisely is and what it's useful for.

As the name gives away, a NumPy cluster is a focal information structure of the numpy library. The library's name is another way to say "Numeric Python" or "Numerical Python".

At the end of the day, NumPy is a Python library that is the center library for logical registering in Python. It contains an accumulation of apparatuses and strategies that can be utilized to settle on a PC numerical models of issues in Science and Engineering. One of these apparatuses is an elite multidimensional cluster object that is an incredible information structure for effective calculation of exhibits and lattices.

To work with these clusters, there's a tremendous measure of abnormal state scientific capacities work on these grids and exhibits. since you have set up your condition, it's the ideal opportunity for the genuine work. In fact, you have officially gone for some stuff with exhibits in the above DataCamp Light pieces. Be that as it may, you haven't generally gotten any genuine hands- on training with them, since you originally expected to introduce NumPy all alone pc. Since you have done this current, it's a great opportunity to perceive what you have to do so as to run the above code pieces without anyone else.

A few activities have been incorporated underneath with the goal that you would already be able to rehearse how it's done before you begin your own!

To make a numpy exhibit, you can simply utilize the `np.array()` work. You should simply pass a rundown to it, and alternatively, you can likewise indicate the information sort of the information. In the event that you need to find out about the conceivable information types that you can pick, go [here](#) or consider investigating DataCamp's NumPy cheat sheet.

There's no compelling reason to proceed to retain these NumPy information types in case you're another client; But you do need to know and mind what information you're managing. The information types are there when you need more power over how your information is put away in memory and on plate. Particularly in situations where you're working with broad information, it's great that you know to control the capacity type.

Remember that, so as to work with the `np.array()` work, you have to ensure that the numpy library is available in your condition. The NumPy library pursues an import tradition: when you import this library, you need to ensure that you import it as `np`. By doing this, you'll ensure that different Pythonistas comprehend your code all the more effectively.

2.4.4 PANDAS

Pandas is an open-source, BSD-authorized Python library giving elite, simple to-utilize information structures and information examination instruments for the Python programming language.

Python with Pandas is utilized in a wide scope of fields including scholastic and business areas including money, financial matters, Statistics, examination, and so on. In this instructional exercise, we will get familiar with the different highlights of Python Pandas and how to utilize them practically speaking.

This instructional exercise has been set up for the individuals who try to become familiar with the essentials and different elements of Pandas. It will be explicitly valuable for individuals working with information purging and examination. In the wake of finishing this instructional exercise, you will wind up at a moderate dimension of ability from where you can take yourself to more elevated amounts of skill.

You should need have a fundamental comprehension of Computer Programming phrasings. A fundamental comprehension of any of the programming dialects is an or more. Pandas

library utilizes the vast majority of the functionalities of NumPy. It is recommended that you experience our instructional exercise on NumPy before continuing with this instructional exercise.

2.4.5 ANACONDA

Anaconda constrictor is bundle director. Jupyter is an introduction layer. Boa constrictor endeavors to explain the reliance damnation in python—where distinctive tasks have diverse reliance variants—in order to not influence distinctive venture conditions to require diverse adaptations, which may meddle with one another.

Jupyter endeavors to fathom the issue of reproducibility in investigation by empowering an iterative and hands-on way to deal with clarifying and imagining code; by utilizing rich content documentations joined with visual portrayals, in a solitary arrangement.

Boa constrictor is like pyenv, venv and minconda; it's intended to accomplish a python situation that is 100% reproducible on another condition, autonomous of whatever different forms of a task's conditions are accessible. It's somewhat like Docker, however limited to the Python biological system.

Jupyter is an astounding introduction device for expository work; where you can display code in "squares," joins with rich content depictions among squares, and the consideration of organized yield from the squares, and charts created in an all around planned issue by method for another square's code.

Jupyter is extraordinarily great in expository work to guarantee reproducibility in somebody's exploration, so anybody can return numerous months after the fact and outwardly comprehend what somebody attempted to clarify, and see precisely which code drove which representation and end.

Regularly in diagnostic work you will finish up with huge amounts of half-completed note pads clarifying Proof-of-Concept thoughts, of which most won't lead anyplace at first.

A portion of these introductions may months after the fact—or even years after the fact— present an establishment to work from for another issue.

2.4.6 PYTHON

Python could be a translated, object-arranged, abnormal state artificial language with dynamic linguistics. Its abnormal state worked in information structures, joined with dynamic composing and dynamic authoritative, make it appealing for Rapid Application Development, just as for use as a scripting or paste language to interface existing segments together. Python's basic, simple to learn language structure underlines intelligibility and hence decreases the expense of program support. Python underpins modules and bundles, which empowers program seclusion and code reuse. The Python translator and the broad standard library are accessible in source or parallel structure without charge for every single significant stage, and can be openly appropriated. Frequently, software engineers begin to look all starry eyed at Python on account of the expanded efficiency it gives. Since there is no aggregation step, the alter test- troubleshoot cycle is staggeringly quick. Troubleshooting Python programs is simple: a bug or awful information will never cause a division blame. Rather, when the mediator finds a blunder, it raises a special case. At the purpose once the program does not get the special case, the translator prints a stack follow. A source level debugger permits assessment of nearby and worldwide factors, assessment of discretionary articulations, setting breakpoints, venturing through the code a line at any given moment, etc. The debugger is written in Python itself, vouching for Python's contemplative power. Then again, frequently the speediest method to troubleshoot a program is to add a couple of print proclamations to the source: the quick alter test- investigate cycle makes this straightforward methodology successful. Python is an item situated, abnormal state programming language with incorporated unique semantics essentially for web and application improvement. It is surprisingly tempting within the field of fast Application Development since it offers dynamic composing and dynamic proscribing alternatives.

Python is generally basic, so it's anything but difficult to learn since it requires a one of a kind language structure that centers around coherence. Designers can peruse and interpret Python code a lot simpler than different dialects. Thusly, this decreases the expense of program upkeep and improvement since it enables groups to work cooperatively without huge language and experience obstructions.

Moreover, Python underpins the utilization of modules and bundles, which implies that projects can be planned in a secluded style and code can be reused over an assortment of tasks.

When you've built up a module or bundle you need, it very well may be scaled for use in different tasks, and it's anything but difficult to import or fare these modules.

A standout amongst the most encouraging advantages of Python is that both the standard library and the mediator are accessible for nothing out of pocket, in both parallel and source structure. There is no restrictiveness either, as Python and all the important instruments are accessible on every single real stage. In this way, it is a tempting alternative for designers who would prefer not to stress over paying high improvement costs.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

The functions of software systems are defined in functional requirements and the behavior of the system is evaluated when presented with specific inputs or conditions which may include calculations, data manipulation and processing and other specific functionality.

- Our system should be able to read the crime data and preprocess data.
- It should be able to analyze the crime data.
- It should be able to group data based on hidden patterns.
- It should be able to assign a label based on its data groups.
- It should be able to split data into train set and test set.
- It should be able to train model using train set.
- It must validate trained model using test set.
- It should be able to classify the crime data.

3.2 NON-FUNCTIONAL REQUIREMENTS

Nonfunctional needs describe however a system should behave and establish constraints of its practicality. This type of needs is additionally called the system's quality attributes.. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic. They are "developing" properties that emerge from the whole arrangement and hence we can't compose a particular line of code to execute them. Any attributes required by the customer are described by the specification. We must include only those requirements that are appropriate for our project.

Some Non-Functional Requirements are as follows:

- Reliability
- Maintainability
- Performance
- Portability
- Scalability
- Flexibility

3.1.1 ACCESSIBILITY:

Availability is a general term used to depict how much an item, gadget, administration, or condition is open by however many individuals as would be prudent.

In our venture individuals who have enrolled with the cloud can get to the cloud to store and recover their information with the assistance of a mystery key sent to their email ids. UI is straightforward and productive and simple to utilize.

3.2.1 MAINTAINABILITY:

In programming designing, viability is the simplicity with which a product item can be altered so as to:

- Correct absconds
- Meet new necessities

New functionalities can be included in the task based the client necessities just by adding the proper documents to existing venture utilizing ASP.net and C# programming dialects. Since the writing computer programs is extremely straightforward, it is simpler to discover and address the imperfections and to roll out the improvements in the undertaking.

3.2.1 SCALABILITY:

Framework is fit for taking care of increment all out throughput under an expanded burden when assets (commonly equipment) are included.

Framework can work ordinarily under circumstances, for example, low data transfer capacity and substantial number of clients.

3.2.2 PORTABILITY:

Conveyability is one of the key ideas of abnormal state programming. Convenient is the product code base component to have the capacity to reuse the current code as opposed to making new code while moving programming from a domain to another. Venture can be executed under various activity conditions gave it meet its base setups. Just framework records and dependant congregations would need to be designed in such case.

3.3 HARDWARE REQUIREMENTS

Processor	: Any Processor above 500 MHz
RAM	: 4 GB
Hard Disk	: 500 GB
System	: Pentium IV 2.4 GHz

Any system with above or higher configuration is compatible for this project.

3.4 SOFTWARE REQUIREMENTS

- Operating system : Windows 7/8/9/10
- Programming lang : Python
- IDE : Jupyter Notebook
- Tools : Anaconda

CHAPTER 4

DESIGN

4.1 DESIGN GOALS

The goal of this project is to predict the highest and closing price of Bitcoin on a given day based on the Bitcoin data of several preceding quarters. It is technically challenging to predict the accurate price, mainly due to lack of seasonality and highly volatile nature of the cryptocurrency market. This is primarily a statistic prediction drawback. Artificial neural network (ANNs) models of time series is used to perform the prediction task, mainly due to the ability of ANNs to deal with non-linearities in the data such as lack of seasonality. These two models are trained and tested on Bitcoin data starting from 2012 till the first quarter of 2018. In order to make the one day ahead prediction of highest and closing price of Bitcoin, features such as open price, high price, low price, close price and volume of currency (USD) are taken into consideration. To predict the highest and closing price on a day of quarter, both the neural network models are trained with data over the past eight quarters and it is tested over the next quarter. The document explains the info preparation steps followed by the neural network models and their practicality. Quantitative measures like to MSE (mean square error), NMSE (normalized mean square error). The predicted high and closing price using these two neural networks are presented in tabular format. At the end, the report discusses possible improvements that can be made to increase the scope of the experiment. The constant increase in bitcoin usage has become an extremely serious problem, with the development of technology and hi- tech tools having a significantly greater impact on the bitcoin price. The large amounts of information also pose's a challenge to analyze such data and identify similarities or relations between the data. Also there is a challenge of inconsistency that can occur in the data due to incompleteness in the dataset. Therefore, there is an urging need of proper techniques to analyze large volumes of data to get some useful results out of it. So the main aim of this project is to propose a general and effective approach to predict the bitcoin price using data mining techniques.

The main options of the projected system are:

- More efficient.
- Better bitcoin price monitoring systems.
- Reduces the costs of storage, maintenance and personnel.
- It reduces the time complexity of the system.
- System that has a simpler architecture to understand.
- Processing of large amount of data becomes easier.

4.1.1 INPUT/OUTPUT PRIVACY

No sensitive information from the large data sets are taken. The data taken are of use to the society as it helps in solving important problems.

4.1.2 EFFICIENCY

The local computations done by the programmer helps the system that is developed to be more efficient than the rest of the systems. Efficiency is very important when it comes to large systems, as it plays an important role.

4.2 SYSTEM ARCHITECTURE

The architecture of the proposed system has the following components:

- Crime dataset which consists of the crimes that have occurred from day today for 10 years.
- Training data to train the models.
- Testing data to apply the models.
- Data storage- stores data.
- Classification and prediction algorithms.
- Forecast engine.

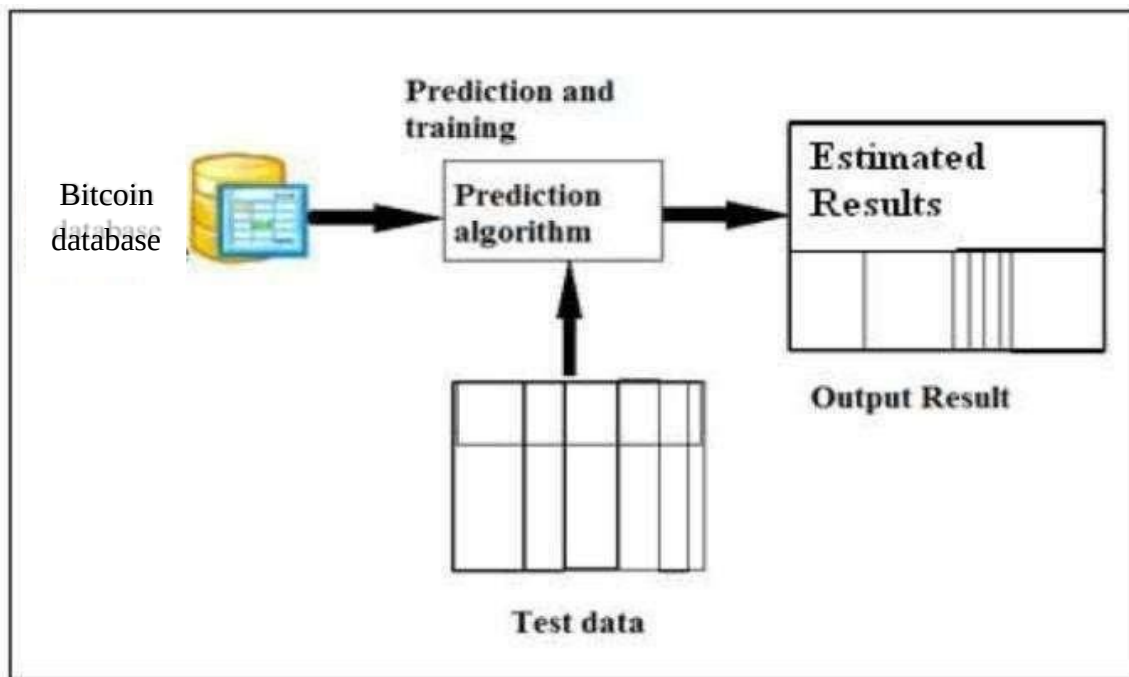


Fig 4.1 System Architecture

4.3 DATA FLOW DIAGRAM

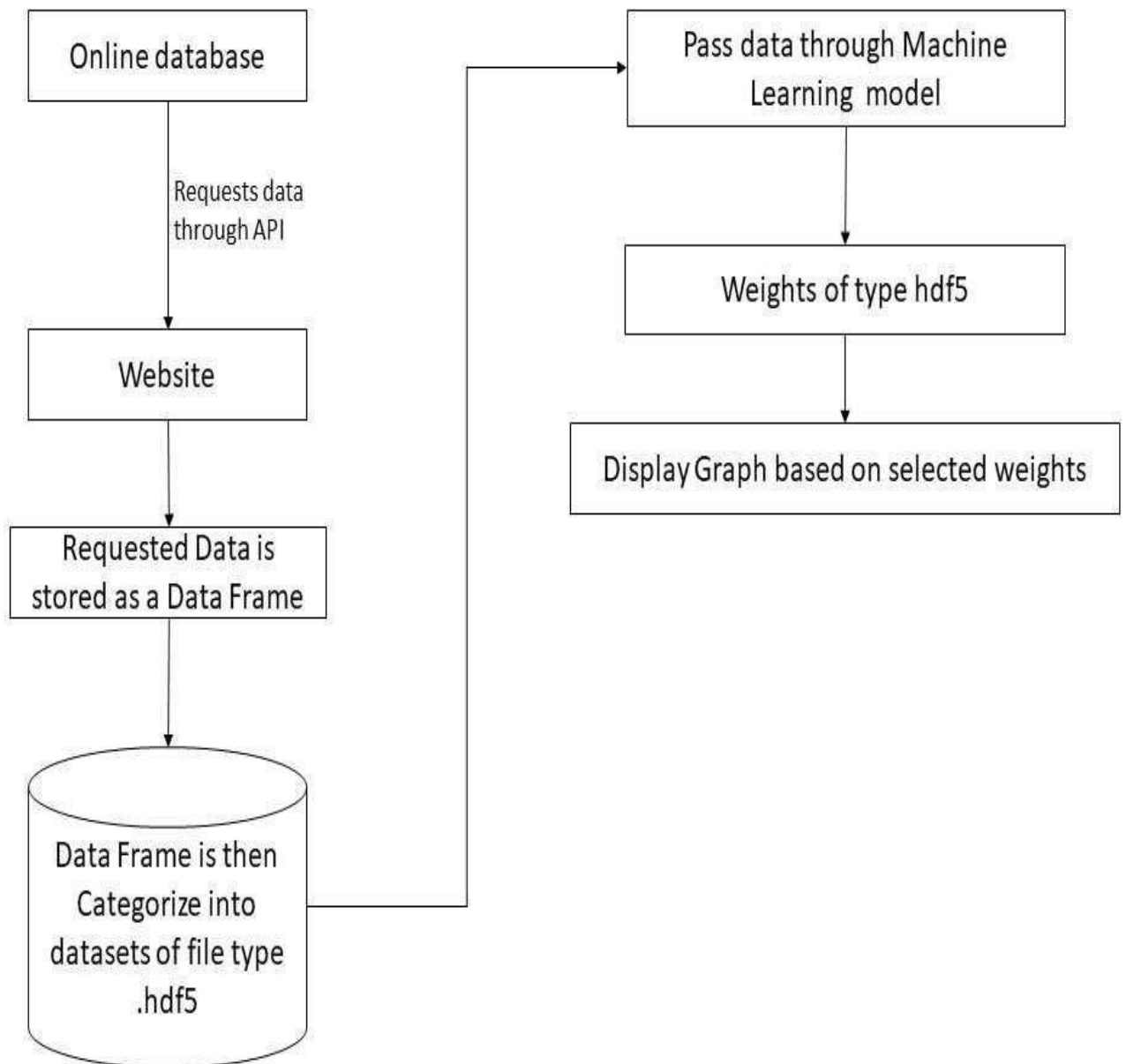


Fig 4.2: Data Flow Diagram

CHAPTER 5

IMPLEMENTATION

5.1 DATASET

Several Bitcoin data sets are available online to download for free. Most of them provide the data related to price of Bitcoin on a minute to minute basis. However, the top goal of the project is to create one-day ahead prediction of highest and shutting worth of Bitcoin. So, we will need data such as highest and closing price of Bitcoin for each day over period of several years. The Quandl API provides the Bitcoin worth knowledge set, ranging from September 2011 – 2018 (present). This API gives access to Bitcoin exchanges and daily Bitcoin values. It permits users to customise the question whereas victimisation the interface to transfer the historical Bitcoin costs. The data is available in three different formats i.e JSON, XML and CSV. Data is downloaded in the

.csv format. Size of data is around 200KB. It has a total of 2381 data records (each record corresponds to a day) consisting of Bitcoin open, high, low, closing price and volume of Bitcoin (USD) starting from Sept 2011 – 2018 (present). However due to inconsistencies in the data from September 2011 to December 2011, this data has been discarded and data records starting from January 2012 – March 2018 are taken into consideration for this project. So, after the data is cleaned, the final data set has a total of 2271 data records. The total data records are divided into three (3) sets, namely: Y12-13 – 2012 and 2013 data, Y14-15 – 2014 and 2015 data, Y16-17 – 2016 and 2017 data. Y12-13 has eight quarters and the neural networks (TDNN, RNN) will be trained on this data and tested on the first quarter of 2014. Similarly, Y14-15 has eight quarters and the neural networks will be trained on this data and tested on the first quarter of 2016. In the same way, Y16-17 has eight quarters and neural networks are trained on this data and tested on the first quarter of 2018.

To predict the highest and closing price of Bitcoin one day ahead, in each of the sub data sets, columns high and close are shifted up by one (1) unit. In the three sub data sets, it should be noted that the testing data is from 1st January to 18th March and it is predicted on 19th March (of years 2014, 2016, 2018) for three sets respectively. The data set has limited features and in the current project almost all these features are considered valuable for the prediction task. To be clear, for predicting the highest and closing price of Bitcoin one step ahead, features such as open, high, low, closing price and volume of Bitcoin (USD) are used.

5.2 DATA PREPROCESSING

The primary knowledge collected from the web sources remains within the raw kind of statements, digits and qualitative terms. The raw data contains error, omissions and inconsistencies. It requires corrections after careful scrutinizing the completed questionnaires. The following steps square measure concerned within the process of primary knowledge. A huge volume of information collected through field survey must be sorted for similar details of individual responses..

Data Preprocessing could be a technique that's accustomed convert the {raw knowledge|data|information} into a clean data set. In alternative words, whenever the info is gathered from totally different sources it's collected in raw format that isn't possible for the analysis.

Therefore, bound steps square measure dead to convert {the knowledge| the info| the information} into a little clean data set. This technique is performed before the execution of reiterative Analysis. The set of steps is understood as knowledge preprocessing.. The process comprises:

- Data Gathering
- Data Cleaning
- Data Normalization

Data Gathering:

Daily data of four channels are considered since 2013. First, the Bitcoin price history, which is extracted from Coin market cap through its open API. Secondly, data from Blockchain is gathered, in particular we choose the average block size, the number of user addresses, number of transactions, and the miners revenue. We found it counter intuitive to have some Blockchain data, given the incessant scaling problem, on the other hand, the number of accounts, by definition is related to the price movements, since an increase in the number of accounts, either means more transactions occurring (presumably for exchanging with different parties and not just transferring Bitcoins to another address), or it is a sign of more users joining the network.

Thirdly, for the sentiment data we obtain the Interest over time for the word 'Bitcoin' using PyTrends library. Lastly, two indices are considered, that of S&P 500 and Dow and Jones. Both are retrieved through Yahoo Finance API.

All in all, these make for 12 features. The Pearson correlation between the attributes is shown in Figure 2. Clearly, some attributes are not too correlated, for example, the financial indices are relevant with each other, but not with any of bitcoinrelated attributes. Also, we see how Google Trends are related to Bitcoin transactions

Data Cleaning:

From exchange data we consider relevant only the Volume, Close, Open, High prices and Market capitalization. For all data sets if NaN values are found to be existent, they are replaced with the mean of the respective attribute. After this, all datasets are merged into one, along the time dimension. Judging from Bitcoin price movements during the period from 2013 until 2014, we considered best to get rid of data points before 2014, hence the data which will be passed to the network lies from 2014 until September 2018.

Data Normalization:

Deciding on the method for normalizing a time series, especially financial ones is never easy. What's more, as a rule of thumb a neural network should load data that take relatively large values, or data that is heterogeneous (referring to time-series that have different scales, like exchange price, with Google Trends). Doing so can trigger large gradient updates that will prevent the network from converging. To make learning easier for the network, data should have the following characteristics

Data Preprocessing is important due to the presence of unformatted universe knowledge. Mostly realworld data is composed of:

- **Inaccurate knowledge (missing knowledge)** - There are several reasons for missing knowledge like data isn't incessantly collected, an error in knowledge entry, technical issues with life science and much more.

- **The presence of clattery knowledge (erroneous knowledge and outliers)** - the explanations for the existence of clattery knowledge can be a technological downside of device that gathers knowledge.
- **human mistake throughout knowledge entry and far additional. Inconsistent knowledge** - The presence of inconsistencies are thanks to the explanations such existence of duplication inside knowledge, human knowledge entry, containing mistakes in codes or names, i.e.,

The column Resolution is dropped because it does not provide any assistance and has no significance in helping to predict the target variable.

5.3 CLASSIFICATION

This technique is used to divide various data into different classes. This process is also similar to clustering. It segments data records into various segments which are known as classes. Unlike clustering, here we have knowledge of different clusters. Ex: Outlook email, they have an algorithm to categorize an email as legitimate or spam.

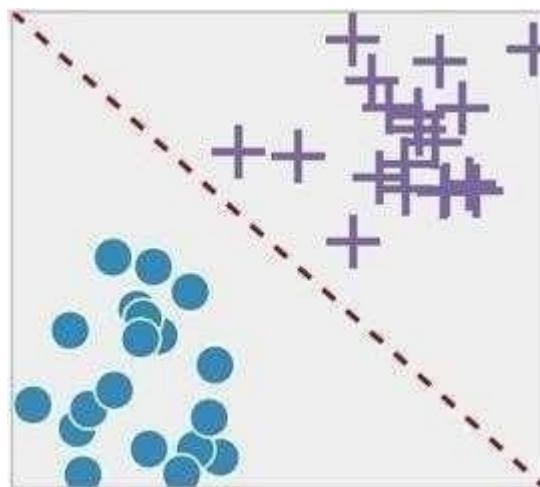


Fig 5.1: Classification

Some of the classification algorithms are:

- Linear Classifiers: Logistic Regression, Naive Bayes Classifier
- Support Vector Machines
- Decision Trees
- Boosted Trees
- Random Forest
- Neural Networks

5.3.1 : TIME SERIES DATA

Normally a time series is a sequence of numbers along time. LSTM for sequence prediction acts as a supervised algorithm unlike its autoencoder version. As such, the overall dataset should be split into inputs and outputs. Moreover, LSTM is great in comparison with classic statistics linear models, since it can easier handle multiple input forecasting problems. In our approach, the LSTM will use previous data to predict 30 days ahead of closing price. First, we have a need to decide on how many previous days one forecast will have access to. This number we refer as the window size. We have opted for 35 days in case of monthly prediction, and 65 days in that of 2 months prediction, therefore the input data set will be a tensor comprising of matrices with dimension 35x12/65x12 respectively, such that we have 12 features, and 35 rows in each window. So the first window will consist of 0 to the 34 row (python is zero indexed), the second from 1 to 35 and so on. Another reason for choosing this window length is that a small window leaves out patterns which may appear in a longer sequence. The output data takes into account not only the window size but also the prediction range which in our case is 30 days. The output dataset starts from row 35 up until the end, and is made of chunks of length 30. The prediction range also determines the output size for the LSTM network.

Split into training and test data:

This step is one of the most important, especially in the case of Bitcoin. We first wanted to predict the year ahead, but this would mean, that data from 1 Jan 2018 until September 2018 would be used for testing, the downside of this, is ofcourse the slight slope in 2017, which would make the neural network learn this pattern by the last input, and the prediction of year 2018 would not be very logical. Thus we go for training data from 2014- 01-01 until 2018- 07-05, this leaves us with approximately 2 months for prediction, while we predict for two months, the data set is split a bit earlier to leave room for 2 months: 2018-06-01. Each training set and test set is composed of input and output features.

Turn data into tensors

LSTM expects that the input is given in the form of a 3 dimensional vector of float values. A key feature of tensors is their shape, which in Python is a tuple of integers representing the dimensions of it along the 3 axis. For instance, in our testing data of Bitcoin, the shape of training inputs is: (1611,35,12), so we have 1611 samples, a window size (timestep) of 35 values, and 12 features.

In overall the idea is simple, in that we separate the data into chunks of 35, and push these small windows of data into a numpy array. Each window is a 35x12 matrix, so all windows will create the tensor. Furthermore, in LSTM the input layer is by design, specified from the input shape argument on the first hidden, the these three dimensions of input shape

5.3.2 : LSTM IMPLEMENTATION

LSTM internals

A chief feature of feed forward Networks, is that they don't retain any memory. So each input is processed independently, with no state being saved between inputs. Given that we are dealing with time series where information from previous Bitcoin price are needed, we should maintain some information to predict the future. An architecture providing this is the Recurrent neural network (RNN) which along with the output has a self-directing loop. So the window we provide as input gets processed in a sequence rather than in a single step. However, when the time step (size of window) is large (which is often the case) the gradient gets too small/large, which leads to the phenomenon known as vanishing/exploding gradient respectively [Chollet2017]. This problem occurs while the optimizer backpropagates, and will make the algorithm run, while the weights almost do not change at all. RNN variations mitigate the problem, namely LSTM and GRU.

The LSTM layer adds some cells that carry information across many timesteps . The cell state is the horizontal line from C_{t-1} to C_t , and its importance lies in holding the long-term or short term memory. The output of LSTM is modulated by the state of these cells. And this is important when it comes to predict based on historic context, rather than only the last input. LSTM networks manage to remember inputs by making use of a loop. These loops are absent in RNN. On the other hand, as more time passes, the less likely it becomes that the next output depends on a very old input, therefore forgetting is necessary. LSTM achieves this by learning when to remember and when to forget, through their forget- gates. We mention them shortly to not consider LSTM just as a black box model [Olah2015].

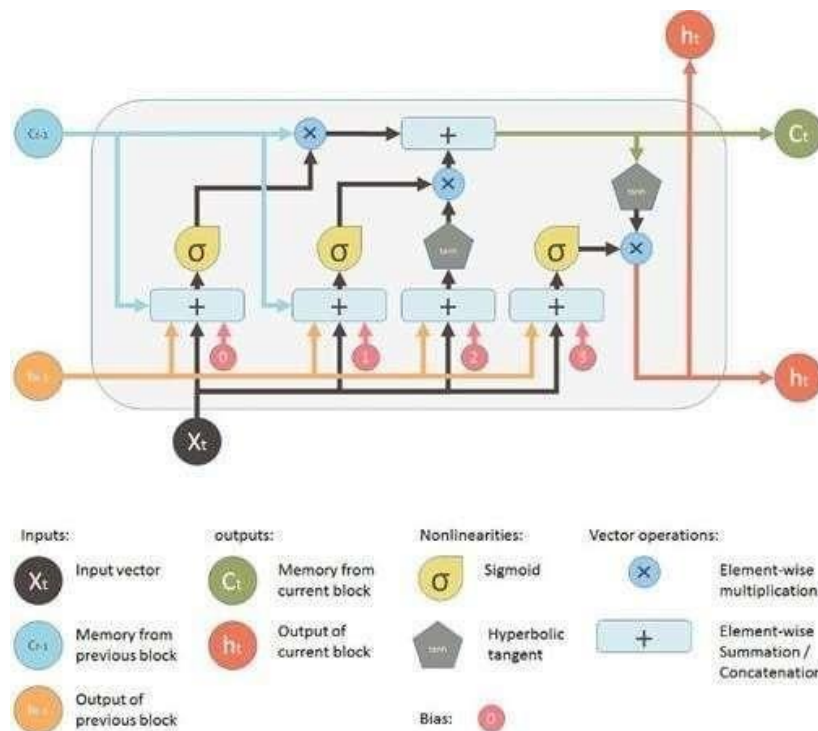


Fig 5.2: LSTM cell

- Forget gate: $f_t = \sigma(W_f S_{t-1} + W_f S_t)$
- Input gate: $i_t = \sigma(W_i S_{t-1} + W_i S_t)$
- Output gate: $o_t = \sigma(W_o S_{t-1} + W_o S_t)$

Architecture of Network:

We used the Sequential API, rather than the functional one. The overall architecture is as follows:

- 1 LSTM Layer: The LSTM layer is the inner one, and all the gates, mentioned at the very beginning are already implemented by Keras, with a default activation of hard- sigmoid [Keras2015]. The LSTM parameters are the number of neurons, and the input shape as discussed above.
- 1 Dropout Layer: Typically this is used before the Dense layer. As for Keras, a dropout can be added after any hidden layer, in our case it is after the LSTM.
- 1 Dense Layer: This is the regular fully connected layer.
- 1 Activation Layer: Because we are solving a regression problem, the last layer should give the linear combination of the activations of the previous layer with the weight vectors. Therefore, this activation is a linear one. Alternatively, it could be passed as a parameter to the previous Dense layer.

5.3.3 : Neural Network Autoregression (NNAR)

Artificial neural network (ANN) methods rely on mathematical models in a similar pattern as 'neurons' in the brain. ANN models help design complex non-linear associations between the dependent variable and its predictors. The simplest ANN models would only have predictors (independent variables or inputs) in the bottom layer and the dependent variable (output) in the top layer, which would be equivalent to a linear regression model. After adding the hidden layer(s) in-between bottom and top layers, the ANN structure becomes non-linear. This type of ANN is called multi-layered feed-forward network, where each layer of neurons (nodes) receive inputs from the previous layer.

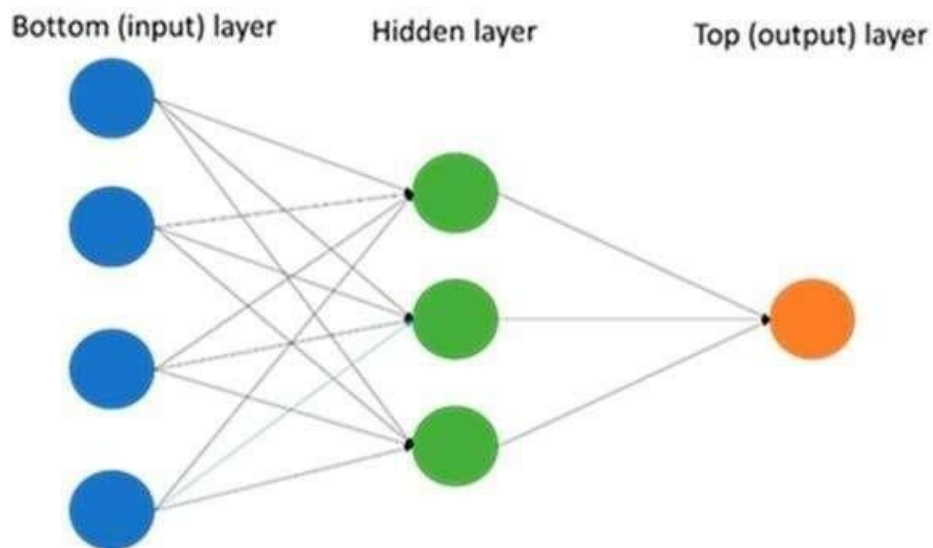


Fig 5.3.3: An ANN model with four inputs and one hidden layer with three neurons.

IMPLEMENTATION

```
import numpy as
np      import
pandas as pd
import matplotlib.pyplot as plt
import math

from sklearn.preprocessing import
MinMaxScaler from sklearn.metrics import
mean_squared_error

from keras.models import
Sequential from keras.layers import
Dense
from keras.layers import LSTM

data =

pd.read_csv("bitcoin.csv")

data.head()

data['rp_key'].value_counts()

df = data.loc[(data['rp_key'] == 'btc_us')]
df.head()

df = df.reset_index(drop=True)

df['datetime'] = pd.to_datetime(df['datetime_id'])
df = df.loc[df['datetime'] > pd.to_datetime('2017-06-28 00:00:00')]

df = df[['datetime', 'last', 'diff_24h', 'diff_per_24h', 'bid', 'ask', 'low', 'high', 'volume']]
df.head()

data1["month"]=data1["Timestamp"].dt.year
data1["year"]=data1["Timestamp"].dt
.month
```

```

data1["day"]=data1["Timestamp"].dt.day
data1.head()

data1["hour"]=data1["Timestamp"].dt.hour
data1["minute"]=data1["Timestamp"].dt

.minute

data1["seconds"]=data1["Timestamp"].dt

.second data1.head()

data1 = data1.rename(columns = {'Volume_(BTC)':'VolumeBTC',
                                'Volume_(Currency)' :
                                'VolumeCurrency', 'Weighted_Price' :
                                'WeightedPrice' })

data1['Open'].plot()
plt.show()

data1["Log_Normalization"]=data1["Open"]/len(data1["Open"])
data1["Log_Normalization"].head()

print('BTC Volume Threshold is : ', threshold)

data1["VolumeLevel"] = ["high" if i > threshold else "low" for i in data1.VolumeBTC]
data1.loc[:,["VolumeLevel","VolumeBTC"]].head(

# adjusting VolumeBTC threshold
threshold = sum(data1.Volume1BTC)/len(data1.VolumeBTC)

```

```

threshold

df = df[['last']]

dataset = df.values

dataset = dataset.astype('float32')

scaler = MinMaxScaler(feature_range=(0,
1)) dataset = scaler.fit_transform(dataset)

train_size = int(len(dataset) * 0.67)

test_size

= len(dataset) - train_size

train, test = dataset[0:train_size, :], dataset[train_size:len(dataset),
:] print(len(train), len(test))

# convert an array of values into a dataset

matrix def create_dataset(dataset,
look_back=1): dataX, dataY = [], []

    for i in range(len(dataset)-look_back-1):

        a = dataset[i:(i+look_back), 0]

        dataX.append(a)

        dataY.append(dataset[i + look_back,
0])

    return np.array(dataX), np.array(dataY)

look_back = 10

trainX, trainY = create_dataset(train,
look_back=look_back) testX, testY = create_dataset(test,
look_back=look_back) trainX

trainY

```

```

# reshape input to be [samples, time steps, features]
trainX = np.reshape(trainX, (trainX.shape[0], 1,
trainX.shape[1])) testX = np.reshape(testX, (testX.shape[0], 1,
testX.shape[1])) model = Sequential()

model.add(LSTM(4, input_shape=(1,
look_back))) model.add(Dense(1))

model.compile(loss='mean_squared_error',
optimizer='adam')

model.fit(trainX, trainY, epochs=100, batch_size=256,
verbose=2) trainPredict = model.predict(trainX)

testPredict = model.predict(testX)

trainPredict =
scaler.inverse_transform(trainPredict) trainY =
scaler.inverse_transform([trainY]) testPredict =
scaler.inverse_transform(testPredict) testY =
scaler.inverse_transform([testY])

trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,
0])) print('Train Score: %.2f RMSE' % (trainScore))

testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,
0])) print('Test Score: %.2f RMSE' % (testScore))

trainPredictPlot =
np.empty_like(dataset)

trainPredictPlot[:, :] = np.nan

```

```

trainPredictPlot[look_back:len(trainpredict) + look_back, :] = trainPredict

testPredictPlot =
np.empty_like(dataset)

testPredictPlot[:, :] = np.nan

testPredictPlot[len(trainPredict) + (look_back * 2) + 1:len(dataset) - 1, :] =

testPredict plt.plot(df['last'], label='Actual')

plt.plot(pd.DataFrame(trainPredictPlot, columns=["close"], index=df.index).close,
label='Training')

plt.plot(pd.DataFrame(testPredictPlot, columns=["close"], index=df.index).close,
label='Testing')

plt.legend(loc='best'

) plt.show()

```

CHAPTER 6

TESTING

The reason for testing is to find blunders. Testing is the way toward endeavoring to find each possible blame or shortcoming in a work item. It gives an approach to check the usefulness of parts, sub gatherings, congregations as well as a completed item it is the way toward practicing programming with the goal of guaranteeing that the Software framework lives up to its necessities and client desires and does not flop in an unsuitable way. There are different kinds of test. Each test type tends to a particular testing prerequisite.

6.1 UNIT TESTING

Unit testing includes the structure of experiments that approve that the inward program rationale is working legitimately, and that program inputs produce substantial yields. All choice branches and inside code stream ought to be approved. It is the trying of individual programming units of the application .it is done after the finishing of an individual unit before combination. This is a basic testing, that depends on data of its development and is obtrusive. Unit tests perform fundamental tests at part level and test a particular business procedure, application, and additionally framework design.

Unit tests guarantee that every extraordinary way of a business procedure performs precisely to the recorded particulars and contains obviously characterized information sources and anticipated outcomes.

6.2 INTEGRATION TESTING

Joining tests are intended to test incorporated programming segments to decide whether they really keep running as one program. Testing is occasion driven and is progressively worried about the fundamental result of screens or fields.

Incorporation tests exhibit that despite the fact that the segments were separately fulfillment, as appeared by effectively unit testing, the mix of parts is right and reliable. Coordination testing is explicitly gone for uncovering the issues that emerge from the blend of segments.

6.3 VALIDATION TESTING

A building approval test (EVT) is performed on first building models, to guarantee that the essential unit performs to plan objectives and particulars. It is imperative in recognizing plan issues, and fathoming them as right off the bat in the structure cycle as could reasonably be expected, is the way to keeping ventures on schedule and inside spending plan. Over and over again, item plan and execution issues are not identified until late in the item improvement cycle — when the item is prepared to be transported. The familiar saying remains constant: It costs a penny to roll out an improvement in building, a dime underway and a dollar after an item is in the field.

Check is a Quality control process that is utilized to assess whether an item, administration, or framework conforms to guidelines, details, or conditions forced toward the beginning of an improvement stage. Check can be being developed, scale-up, or creation. This is regularly an inside procedure.

Approval is a Quality affirmation procedure of setting up proof that gives a high level of confirmation that an item, administration, or framework achieves its planned prerequisites. This regularly includes acknowledgment of qualification for reason with end clients and other item partners.

The testing process overview is as follows:

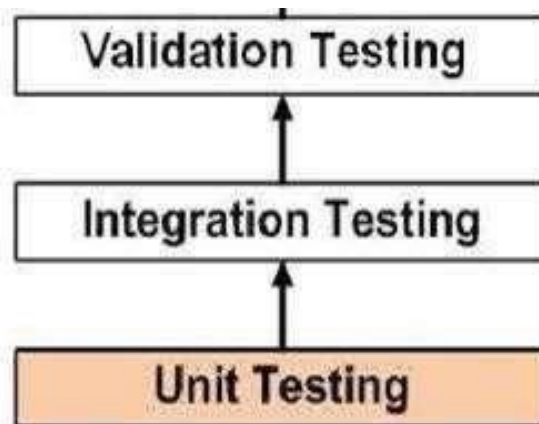


Figure 6.1: The testing process

OUTPUT

	Date	Price
0	03-01-2009	0
1	05-01-2009	0
2	07-01-2009	0
3	09-01-2009	0
4	11-01-2009	0

Figure 7.1:bitcoin dataset

```
Epoch 1/100
341/341 [=====] - 1s 2ms/step - loss: 0.0418
Epoch 2/100
341/341 [=====] - 0s 1ms/step - loss: 0.0139
Epoch 3/100
341/341 [=====] - 0s 1ms/step - loss: 0.0128
Epoch 4/100
341/341 [=====] - 0s 1ms/step - loss: 0.0117
Epoch 5/100
341/341 [=====] - 0s 1ms/step - loss: 0.0104
Epoch 6/100
341/341 [=====] - 0s 1ms/step - loss: 0.0090
Epoch 7/100
341/341 [=====] - 0s 1ms/step - loss: 0.0075
Epoch 8/100
341/341 [=====] - 0s 1ms/step - loss: 0.0059
Epoch 9/100
341/341 [=====] - 0s 1ms/step - loss: 0.0043
Epoch 10/100
```

Figure 7.2: Lstm model

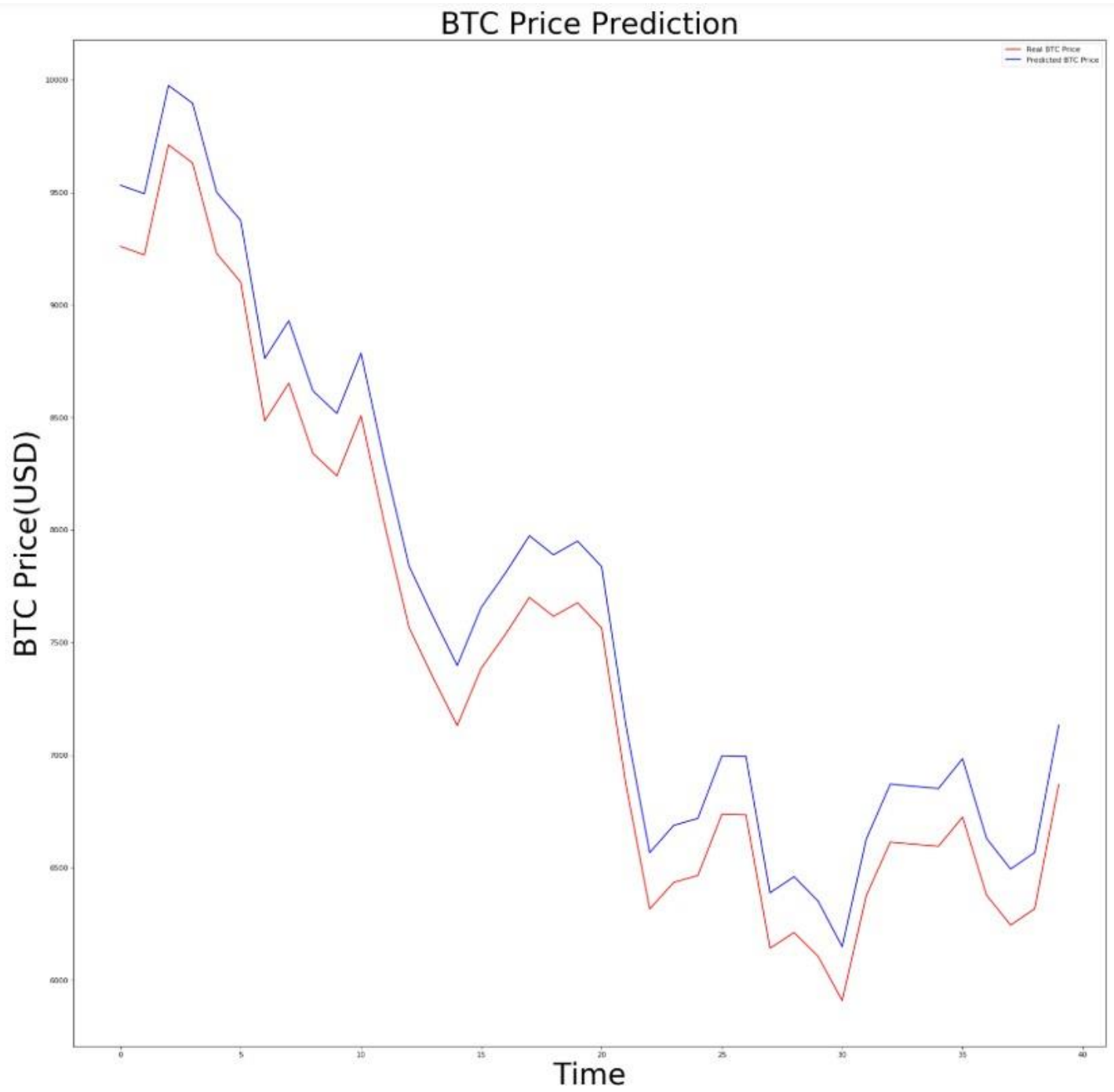


Figure 7.3: prediction graph

```

array([[ 9531.405 ],
       [ 9493.7705],
       [ 9974.662 ],
       [ 9895.743 ],
       [ 9500.857 ],
       [ 9375.266 ],
       [ 8761.879 ],
       [ 8929.134 ],
       [ 8618.275 ],
       [ 8517.4795],
       [ 8784.903 ],
       [ 8292.161 ],
       [ 7839.0737],
       [ 7613.0156],
       [ 7397.477 ],
       [ 7656.0767],
       [ 7807.6772],
       [ 7974.136 ],
       [ 7889.1353],
       [ 7950.296 ],
       [ 7837.2686],
       [ 7138.1553],
       [ 6566.1265],
       [ 6687.301 ],
       [ 6718.3203],
       [ 6996.7705],
       [ 6993.61  ],
       [ 6387.717 ],
       [ 6459.3193],
       [ 6350.475 ],
       [ 6148.662 ],
       [ 6626.5864],
       [ 6870.891 ],
       [ 6860.362 ]],
      dtype=float64)

```

Figure 7.4: predicted price for next 40 days

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

The phase 1 was a learning innovative that we tend as students need to understand loads of things. This was a very nice exposue to learn a lot of new concepts. Phase 1 was more like a learning experience for the domain of data mining.

The main aspect of the major project was seen in this phase, a lot of learning and understanding has gone into doing this.

The phase 2 was a more closer approach to the project where the actual implementation took place. Bitcoin prediction is a very crucial topic to deal with and making a system suitable for it was a challenging role to do.

At the end a fully working is system is made by using the classification techniques of data mining domain.

8.2 FUTURE ENHANCEMENT

- Facilities for maintaining large amounts of data.
- Access of data through Mobile devices.
- Use of a good software.
- More interactive user interface.
- Facilities for Backup creation.

REFERENCES

- <http://www.iosrjournals.org/iosr-jce/papers/Vol19-issue3/Version-1/B1903010617.pdf>
- <https://ieeexplore.ieee.org/document/7395797>
- <https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/>
- <https://www.geeksforgeeks.org/designing-use-cases-for-a-project/>
- <https://dl.acm.org/citation.cfm?id=170072> <https://www.edureka.co/blog/apriori-algorithm/>
- <https://content.iospress.com/articles/intelligent-data-analysis/ida1-1-02>
- <https://link.springer.com/book/10.1007%2F978-3-319-10247-4>
- <https://www.geeksforgeeks.org/apriori-algorithm/>