Cryptocurrency Price Prediction

A Project Report

Submitted to the

University of Madras

in partial fulfillment of requirement for award of the Degree of

Bachelor of Computer Science

by

INDHUMATHI V (Reg. No. 222002441)

SHALINI V (Reg. No. 222002477)

HAJARA MUGASINA M (Reg. No. 222002438)

Under the Guidance of

Mrs. K. Praba M.Sc., M.Phil., B.Ed.,

PRINCE SHRI VENKATESHWARA ARTS AND SCIENCE COLLEGE

[AFFILIATED TO THE UNIVERSITY OF MADRAS]

2022-2023

BONAFIDE CERTIFICATE

This is to certify that this report titled **"Cryptocurrency Price Prediction"** is the bonafide record of the project work by

INDHUMATHI V (Reg. No. 222002441),

SHALINI V (Reg. No. 222002477),

HAJARA MUGASINA M (Reg. No. 222002438)

under our supervision and guidance, towards partial fulfillment of the requirement for award of the Degree of B.Sc Computer Science of Prince Shri Venkateshwara Arts and Science College, Gowrivakkam, Chennai – 600073.

Internal Guide **Head of the Department**

Submitted for the Viva-Voce Examination held on……………………..at Prince Shri Venkateshwara Arts and Science College, Gowrivakkam – 600 073.

Date: **Examiners:**

1.

2.

ACKNOWLEDGEMENT

First of all I would like to thank **Dr. K. VASUDEVAN, M.A., B.Ed.,Ph.D. Founder & Chairman,** and **Er.V.PRASANNA VENKATESH B.E.,M.Sc.,(UK) Vice Chairman** for their scholarly guidance and motivation throughout the project work as well as the curriculum.

I would like to express my gratitude to the Principal **Dr.M.UMA M.Sc.,M.Phil.,Ph.D.,** for her consent and support to the project.

My heartfelt thanks to the Vice Principal, **Dr.J.JANAKI, M.Com., M.Phil, NET., Ph.D.,** for motivating me throughout the course of the work.

I acknowledge my sincere thanks to **Mrs.C.VEERASUSARETHA, MCA, M.Phil.,** Head of the Department, Department of Computer Science for the useful guidance and helpful mind to complete this project.

My sincere and heartfelt thanks goes to my guide **Mrs. K.Praba,M.Sc.,M.Phil.,B.Ed.,** Assistant Professor, Department of Computer Science, for suggesting the problem, offering inspiring guidance and fruitful discussion throughout the course of the work.

My special thanks to all the staff members, Department of Computer Science for their support and encouragement for the successful completion of the project.

I acknowledge my profound thanks to my parents for their encouragement and for social and economical support for completing this project work.

ABSTRACT

Blockchain technology is a structure that stores transactional records,also known as the Block of the public in several databases known as the “chain” in a network connected through peer–to-peer nodes. Typically, this storages is referred to as a “digital ledger”. It’s a decentralized network.Bitcoin is a cryptocurrency which is used Worldwide for digital payment or simply for investment purposes.The record of all the transactions,the timestamp data is stored in a place called Blockchain. Each record in a blockchain is called a block. The aim of the project is understand and identify daily treads in the Bitcoin market. The data set consists of various features relating to the Bitcoin price and payment network.Bitcoin, the first decentralized cryptocurrency, has become popular not only because a growing size of merchants accepts it in transactions,but also because people buy it as an investment.The proposed model uses algorithms like Long short term memory (LSTM) and Gated recurrent units(GRU).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. As bitcoin is growing drastically in recent times, it is very important to predict the price of it more accurately as it is going to be future trend of investment.If prices are predicted more accurately then it will be helpful for people to invest on it.

The GRU model best outperforms the LSTM model, though sometimes it might have a more extreme result.When the price remains constant or change steadily, the predictions are more precise than the fluctuation period.

Keywords: Bitcoin prediction,Time complexity,Machine Learning,Database architecture,

LSTM,GRU.

INDEX

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Bitcoin is a crypto currency which is used worldwide for digital payment or simply for investment purposes. Bitcoin is decentralized i.e. it is not owned by anyone. Transactions made by Bitcoins are easy as they are not tied to any country. Investment can be done through various marketplaces known as “bitcoin exchanges”. These allow people to sell/buy Bitcoins using different currencies. The largest Bitcoin exchange is Mt Gox . Bitcoins are stored in a digital wallet which is basically like a virtual bank account. The record of all the transactions, the timestamp data is stored in a place called Block chain. Each record in a block chain is called a block. Each block contains a pointer to a previous block of data. The data on block chain is encrypted. During transactions the user’s name is not revealed, but only their wallet ID is made public.To predict the future, has always been a goal or dream of humanity; however humans have always been terrible at it. Predicting the price movements for cryptocurrency or bitcoin is a relatively similar to predicting stocks or the price of USD going up or down. However, unlike a company with physical buildings and people working in it, bitcoin is purely digital and therefore very difficult to understand when and why the price would change. With no physical entity and the way bitcoin works, most of the effect on the price is based on how the world feels about it. For this reason, understanding people is one way to help predict bitcoins future. Many researchers have investigated sentiment analysis on different kind of social media to get a better understanding of how people feel. However as mentioned earlier, humans are terrible at predicting, even with information at their hand. For this reason, by using a machine learning algorithm, the algorithm will try to predict the future price based on the information given to it.

There has been significant effort put into predicting bitcoin prices using historical economic data. Bitcoin has become an increasingly important part of economic analysis because it has become the figurehead for the entire cryptocurrency market. While this market of cryptocurrencies is still untested and unverified (as a viable currency) it is undeniably a powerful concept backed by solid technology (block chain). As a result, predicting this market is arguably as important as predicting the traditional stock market.Besides this, the capacity to make unbelievable returns on investment has also been a key point of interest. The tested assumption of this model was the impact of “hype” on the price of bitcoin,especially in the last year of extremely volatile behavior. This hype was measured using sentiment analysis on a dataset of historical news headlines. Using a sentiment analysis library, TextBlob, I was able to generate a "sentiment" score for each headline, and then weight this score by multiplying in an "objectivity" score from the same library. This gave me a feature indicative of "objective positivity" for each headline, which I was then able to average across all headlines for a given day to get a single input feature for every data point (a single day). This was my most critical feature and consumed most of my development time (as described in the discussion section). Because I wanted to even out my feature set and input more to the network, I also pulled in some very basic historical data on bitcoin, specifically: previous day's close, previous day's volume and the label for the previous day (the closing value minus the opening value). I also used the previous day's "objective positivity" score as a feature. In an attempt to boost my results, I also added data from Google trends, by pulling analytics for the search term "Bitcoin,"and using the weighted values for "activity" across the days as another feature. The output of the model is defined as the sign change of bitcoin's price for that day, which I derived for historicals by subtracting the opening price from the closing price.To predict the future, has always been a goal or dream of humanity; however humans have always been terrible at it. Predicting the price movements for cryptocurrency or bitcoin is a relatively similar to predicting stocks or the price of USD going up or down. However, unlike a company with physical buildings and people working in it, bitcoin is purely digital and therefore very difficult to understand when and why the price would Bitcoin Price Prediction change. With no physical entity and the way bitcoin works, most of the effect on the price is based on how the world feels about it. For this reason, understanding people is one way to help predict bitcoins future. Many researchers have investigated sentiment analysis on different kind of social media to get a better understanding of how people feel. However as mentioned earlier, humans are terrible at predicting, even with information at their hand. For this reason, by using a machine learning algorithm, the algorithm will try to predict the future price based on the information given to it.

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 begining 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 decentralized, 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. Section one addresses the project specification which includes the research question, sub research questions, the purpose of the study and 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. We implemented two forecasting methods, including Long Short-term Memory Model (LSTM) and Gated Recurrent Units Model (GRU) with two different emission probabilities. The LSTM model and the GRU model are popular deep learning methods for predicting Time Series data. They are well-known for learning from long-term sequence dependency, though increasing the computing complexity by introducing extra parameters. The reason to use this method is that the price of a cryptocurrency is determined by some underlying stochastic process that is invisible to investors, and Weigend successfully used this model in their previous work on financial time series data . In this study, each model’s performance is evaluated by MAPE and RMSE of the testing set. We will discuss these three methods’ details and results and conclude the best model in the later chapter.

1.1 RELEVANCE OF THE PROJECT

Now that Bitcoin has cemented itself as the defactor figure head for cryptocurrencies, the interest in predicting and modeling its behavior has skyrocketed. Work in this area has seen a dramatic increase in recent years, which I will detail briefly. An earlier model in 2015 by Alex Greaves and Benjamin Au was only able to achieve accuracy of 56% using neural networks to predict bitcoin price fluctuation using block chain specific data. As mentioned earlier, Madan, Saluja and Zhao’s model in 2014 showed much more impressive accuracy of 98.7% in predicting bitcoin price fluctuations on a day by day basis, and competent accuracy (50-55%) when narrowed to a 10 minute widow. Hegazy and Mumford’s paper attempts many different techniques for automated bitcoin trading finding that Boosted Trees provided the best test and training accuracy for their data set. More recently, there has been some significant work using a Long Short Term Memory model to predict bitcoin price fluctuations. Jakob Aungiers was able to develop a model that closely matched the direction of Bitcoin’s price, but was even more volatile than the true fluctuations. Derek Sheehan’s model, also using LSTM models, was able to achieve a greater representation of the true fluctuations, eventually arriving at an average error of0.04 and 0.05 for Bitcoin and Etherium, respectively. While all of the models were able to achieve solid representations of the Bitcoin market, I thought the work done by Madan etal.’s modelling was the most impressive and therefore the best place to focus my efforts. It would seem the “state of the art” (determined by the topics of recent papers) involves these LSTM models, but these are molded specifically for time series data much more granular (by minute/second) than my data (daily).

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 than1, 500 actively traded crypto currencies. Between2.9and 5.8millions 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 overone-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 Gated Recurrent unit (GRU) and Long Short-Term Memory (LSTM) network.

1.3 PROJECT PURPOSE

The purpose of this project is to predict the closing price of Bitcoin on a given day based on the Bitcoin data. This is basically a time series prediction problem. If the bitcoin data is considered to 2022, it makes 52 weeks, 365days, 8760 hours, 525960 minutes. If the total markets which are selling and buying bitcoin are considered we have to multiply these values with 27.000. There is not only Bitcoin in cryptocurrency world; there are also nearly 5.000 other cryptocurrency coins. In this work, data is chosen as daily bitcoin prices. In order to make the one day ahead prediction of closing price of Bitcoin, features such as the opening price, highest price, lowest price, closing price, volume of Bitcoin, volume of currencies, and weighted price are taken into consideration. To predict the closing price on a day, the GRU and LSTM is trained with data over the past historical data and it is tested over the next quarter. Model accuracy is presented as RMSE (root mean square error) and r (Pearson’s correlation coefficient),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.

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 loosing 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 SCOPE OF THE PROJECT

Today bitcoin is a secure transaction system that has a variable impact on capital. They are awarded under a restriction in which customers offers their computer authority to register and listing trades with bitcoins. The purchase and sale of bitcoin in different currencies is carried out in an alternative workplace where “purchase” or “sell” request are placed in the ordered e-book. The purpose of this study is to finfout with what accuracy the direction of the price of Bitcoin ca be predicte using machine learning methods.this is basically time series prediction problem. while much research exists sourrounding the use of different machine learning techniques for time series prediction ,research in the area relating specifically to itcoin is lacting. In addition ,Bitcoin as a currency is a transaction stage and as a result is considerably more volatiles than other currencies such as the USD/INR.

1.6 MODULES DESCRIPTION

- Data Gathering
- Data Preprocessing
- Classification

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.

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.

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

In order to get required knowledge about various concept related to the present application,existing literature were studied. Some of the important conclusion were made through those are mentioned below.

1 "Predicting the price of the Bitcoin Using Machine Learning" -

Sean McNally, Jason Roche, Simon Caton – 2018 IEEE 26th Euromicro International Conference on Parallel, Distributed, Network - Based Processing. The goal of this paper is to ascertain with what accuracy 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 a Long Short-Term Memory (LSTM) network. The LSTM achieves the highest classification accuracy of 52% and a RMSE of 8%. The popular ARIMA model for time series forecasting is implemented as a comparison to the deep learning models. As expected, the non-linear deep learning methods outperform the ARIMA forecast which performs poorly. Finally, both deep learning models are benchmarked on both a GPU and a CPU with the training time on the GPU outperforming the CPU implementation by 67.7%.

2 "A New Forecasting Framework for Bitcoin price with LSTM" -

Wu Chih – Hung, Ma Yu – Feng, Lu Chih – Chiang – 2018 IEEE International Conference on Data Mining Workshops (ICDMW). Long short-term memory (LSTM) networks are a state-of-the-art sequence learning in deep learning for time series forecasting. However, less study applied to financial time series forecasting especially in cryptocurrency prediction. Therefore, we propose a new forecasting framework with LSTM model to forecasting bitcoin daily price with two various LSTM models (conventional LSTM model and LSTM with ARIMA model. The performance of the proposed models are evaluated using daily bitcoin price data during 2018/1/1 to 2018/7/28 in total 208 records. The results confirmed the excellent forecasting accuracy of the proposed model with ARIMA. The test Bitcoin Price Prediction using

Machine Learning Dept. of ISE, CMRIT 2019-20 Page 5 mean squared error (MSE), root mean square error (RMSE), mean absolute percentage error (MAPE), and mean absolute error (MAE) for bitcoin price prediction, respectively. The our proposed LSTM with AR(2) model outperformed than conventional LSTM model. The contribution of this study is providing a new forecasting framework for bitcoin price prediction can overcome and improve the problem of input variables selection in LSTM without strict assumptions of data assumption. The results revealed its possible applicability in various cryptocurrencies prediction, industry instances such as medical data or financial time-series data.

3 "A Study of Opinion Mining and Data Mining Techniques to Analyze the Cryptocurrency Market"-

Akhilesh P. Patil, T.S. Akarsh, A. Parkavi – 2018 3rd International Conference on Computational Systems and Information Technology of Sustainable Solutions (CSITSS). The value of various Cryptocurrencies such as Bitcoin, Litecoin, Ethereum are always elusive. Hence, it would be a great value addition to investors if a model is able to predict what would be the nature of the crypto market for the next day. Through this paper, a time-series model using Long Short-Term Memory Networks is built to determine the value of cryptocurrency in the future. As a study, three cryptocurrencies - Bitcoin, Litecoin and Ethereum has been taken into consideration. A comparison of the results by using opinion mining to interpret the mood of the market on the current day for different currencies has been done. The sentiment scores got from natural language processing of textual data are used as features to the model used for predictions. The time-series charts are plotted using Plotly - python library for graphing plots. The Mean Absolute Error calculated between the actual and predicted values is used as the uncertainty quantification method. These uncertainty quantification methods are compared to analyze the present-day scenario of the market using opinion mining.

CHAPTER 3

SYSTEM STUDY

3.1 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.

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.

3.2 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) and GRU(Gated Recurrent units),accuracy compared to show best algorithm for prediction. 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.

Advantages:

- Implement more than one machine learning to predict the value
- Accuracy are compared to show best algorithm for prediction.

3.3 FEASIBILITY STUDY

All system are feasible when provided with unlimited resources and infinite time. But unfortunately this condition does not prevail in the practical world. So it is both necessary and product to evaluate the feasibility of the system at the earliest possible time. Months of years effort, thousands of rupees and untold professional embarrassment can be averted if an ill-conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into research and development of the system is limited. The expenditures just be justified. Thus, the developed System as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized product had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility , that is the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use a system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the user solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able make some constructive criticism ,which is welcomed,as he is the final user of the system.

CHAPTER 4

SYSTEM REQUIREMENTS SPECIFICATIONS

A System Requirement Specification (SRS) is basically an organization's understanding of a customer or potential client's system requirements and dependencies at a particular point prior to any actual design or development work. The information gathered during the analysis is translated into a document that defines a set of requirements. It gives a brief description of the services that the system should provide and also the constraints under which the system should operate. Generally,SRS is a document that completely describes what the proposed software should do without describing how the software will do it. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an ecommerce website and so on) must provide, as well as states any required constraints by which the system must abide. SRS also functions as a blueprint for completing a project with as little cost growth as possible.SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications,statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system clearly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and can come from any number of sources.

4.1 FUNCTIONAL REQUIREMENT

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements: -

Following are the functional requirements on the system:

1. The entire control model set must be translated to C output Code.

2. Inputs must be models designed using CLAW design components along with standard design components,

3. Multiple design models must be processed and the result must be combined to obtain a single output file.

4.2 NON FUNCTIONAL REQUIREMENT

Non functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Nonfunctional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:
-➢ Product Requirements

➢ Organizational Requirements

➢ User Requirements

➢ Basic Operational Requirements

4.2.1 PRODUCT REQUIREMENTS

Platform Independency: Standalone executables for embedded systems can be created so the algorithm developed using available products could be downloaded on the actual hardware and executed without any dependency to the development and modeling platform.

Correctness: It followed a well-defined set of procedures and rules to compute and also rigorous testing is performed to confirm the correctness of the data.

Ease of Use: Model Coder provides an interface which allows the user to interact in an easy manner.

Modularity: The complete product is broken up into many modules and well- defined interfaces are developed to explore the benefit of flexibility of the product.

Robustness: This software is being developed in such a way that the overall performance is optimized and the user can expect the results within a limited time with utmost relevance and correctness Nonfunctional requirements are also called the qualities of a system. These qualities can be divided into execution quality & evolution quality. Execution qualities are security & usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

4.2.2 ORGANIZATIONAL REQUIREMENTS

Process Standards: The standards defined by DRDO are used to develop the application which is the standard used by the developers inside the defense organization.Design Methods: Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

4.2.3 USER REQUIREMENTS

The coder must request the name of the model file to be processed.

In case of multiple files, the coder must ask the names of the files sequentially.

The output file must be a C code translated from the model.

Only a single output file must be created even if multiple input files are provided.

4.2.4 BASIC OPERATIONAL REQUIREMENTS

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points: -

Mission profile or scenario: It describes the procedures used to accomplish mission objectives. It also finds out the effectiveness or efficiency of the system.

Performance and related parameters: It points out the critical system parameters to accomplish the mission

Utilization environments: It gives a brief outline of system usage. Finds out appropriate environments for effective system operation.

Operational life cycle: It defines the system lifetime.

4.2.5 SYSTEM CONFIGURATION

H/W System Configuration:

Processor - Pentium –IV

RAM - 4GB RAM

Hard Disk - 20 GB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

Monitor – SVGA

S/W System Configuration:

Operating System: windows 7 or higher

Coding Language: Python

IDE: Juypter Notebook

Tool: Anaconda

4.3 TECHNICAL REQUIREMENT SPECIFICATION

4.3.1 SOFTWARE REQUIREMENTS

JUPYTER NOTEBOOK

Jupyter Notebook is an Integrated Development Environment (IDE) for programming in the Python language. It is an open source, cross-platform IDE available on Anaconda. It has a huge volume of inbuilt libraries that can be used. Few of them are:

- NumPy
- SciPy
- Matplotlib
- Pandas
- IPython
- SymPy
- Cython

Jupyter Notebook was developed by Pierre Raybaut in 2009 and post 2012 Jupyter Notebook has been maintained and continuously improved by a team of scientific Python developers and a dedicated community group for the same.

Jupyter Notebook can be extensible with first-party and third-party plugins and has the provision for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments. Examples of this are:

- Pyflakes
- Pylint
- Rope

For GUI,Jupyter Notebook uses Qt. It is designed to use either of the PyQt or PySide Python bindings. A thin abstraction layer developed by the Jupyter Notebook project and later adopted by multiple other packages and provides the freedom to use either backend is called QtPy. It is a python framework used to develop simple frontend GUIs for python projects and programs developed in Python.

As a cross-platform IDE it is available through:

• Anaconda,

• On Windows,

• On macOS through MacPorts.

MATPLOTLIB

Matplotlib is an amazing visualization library in python for 2D plots of arrays.Matplotlib is a multi-platform data visualization library built on Numpy arrays and desidned to work with the broader Scipy stack.It was introduced by John Hunter in the year 2002.One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.Matplotlib consists of several plots like line, bar, scatter, histogram etc. Installation : Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages.

NUMPY

Numpy is a general-purpose array-processing package. It provides a high – performance multidi-mensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with python. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

PANDAS

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, Exploring, and manipulating data. The name “Pandas” has a reference to both “Panel data”, and “python Data Analysis” and was created by Wes McKinney in 2008. Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.

The dataset which is obtained will be in csv format. i.e they will be comma- separated values. This dataset has to be fed into Pandas to perform predictive analysis. Pandas is used along with python programming language and is thus a perfect fit for this project. It is an open source software, made free to use by the public.

SKLEARN

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**. French research scientist David Cournapeau's scikits.learn is a Google Summer of Code venture where the scikit-learn project first began. Its name refers to the idea that it's a modification to SciPy called "SciKit" (SciPy Toolkit), which was independently created and published. Later, other programmers rewrote the core codebase.

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.

PYTHON

Python is a high-level programming language used primarily for general purpose programming. It was created in the year 1991 by Guido van Rossum. It has a designphilosophy that emphasizes mainly on aspects such as code readability and notably uses significant whitespace. It is a very efficient programming language and is widely used across the globe for both small and large scale programming.Python possesses automated memory management. It provides multiple programming paradigms some of which are object-oriented, imperative, functional and procedural. It also has a large and comprehensive standard library.

Advantages of Python which will help us in this project are:

• Python is platform independent and can thus run on any Operating System.

• The syntax of Python is simple and easy to remember.

• It reduces the size of the code and also the complexity is simplified when compared to other programming languages.

• It has an interpreter rather than a compiler, thus speeding up the prototyping process.

• It supports procedural programming, object-oriented programming as well as functional programming, thus being very efficient.

Some of the notable applications of Python are:

• Server-side programming in Web development

• It is used in software development where complex problems need to be resolved

• Used in statistics

• Used for System scripting

• Python supports database connectivity and can also read from, write to and append files.

• It can be used in handling big data and also to resolve complex mathematical problems.

• It is also used for prototyping software in their development process.

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.

4.3.2 Hardware Requirements

A processor above 500 MHz is recommended for the project.

A minimum of 1GB RAM is required to run this project. 4GB RAM or higher is ideal for this project .

Minimum 30GB free hard disk space is required . However , 100GB and above free disk space is optimal to run this system .

An input device (keyboard) to enter the data.

An output device (monitor) to see the output.

CHAPTER 5

REQUIREMENT ANALYSIS

5.1 Block Diagram for proposed workflow

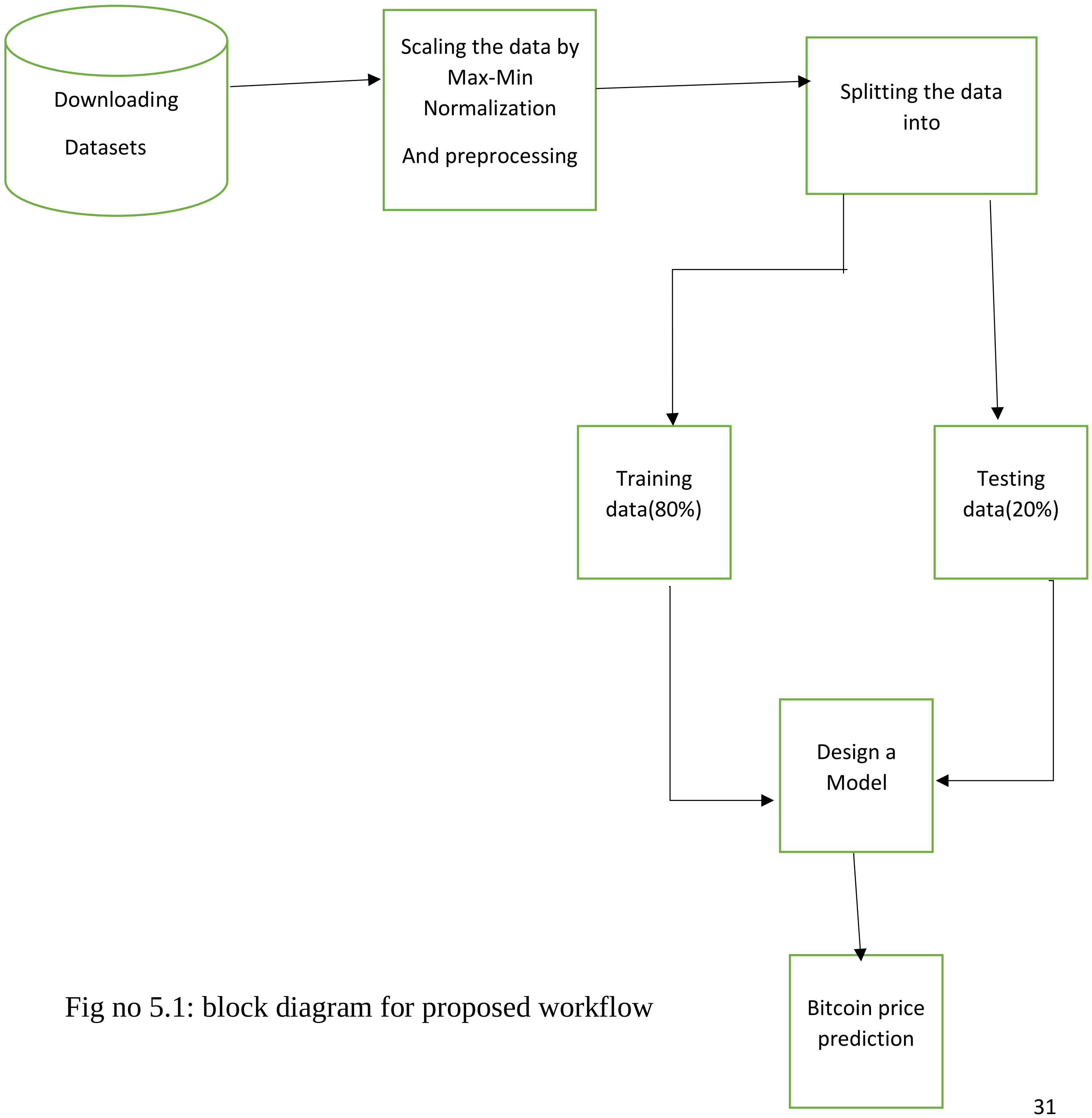
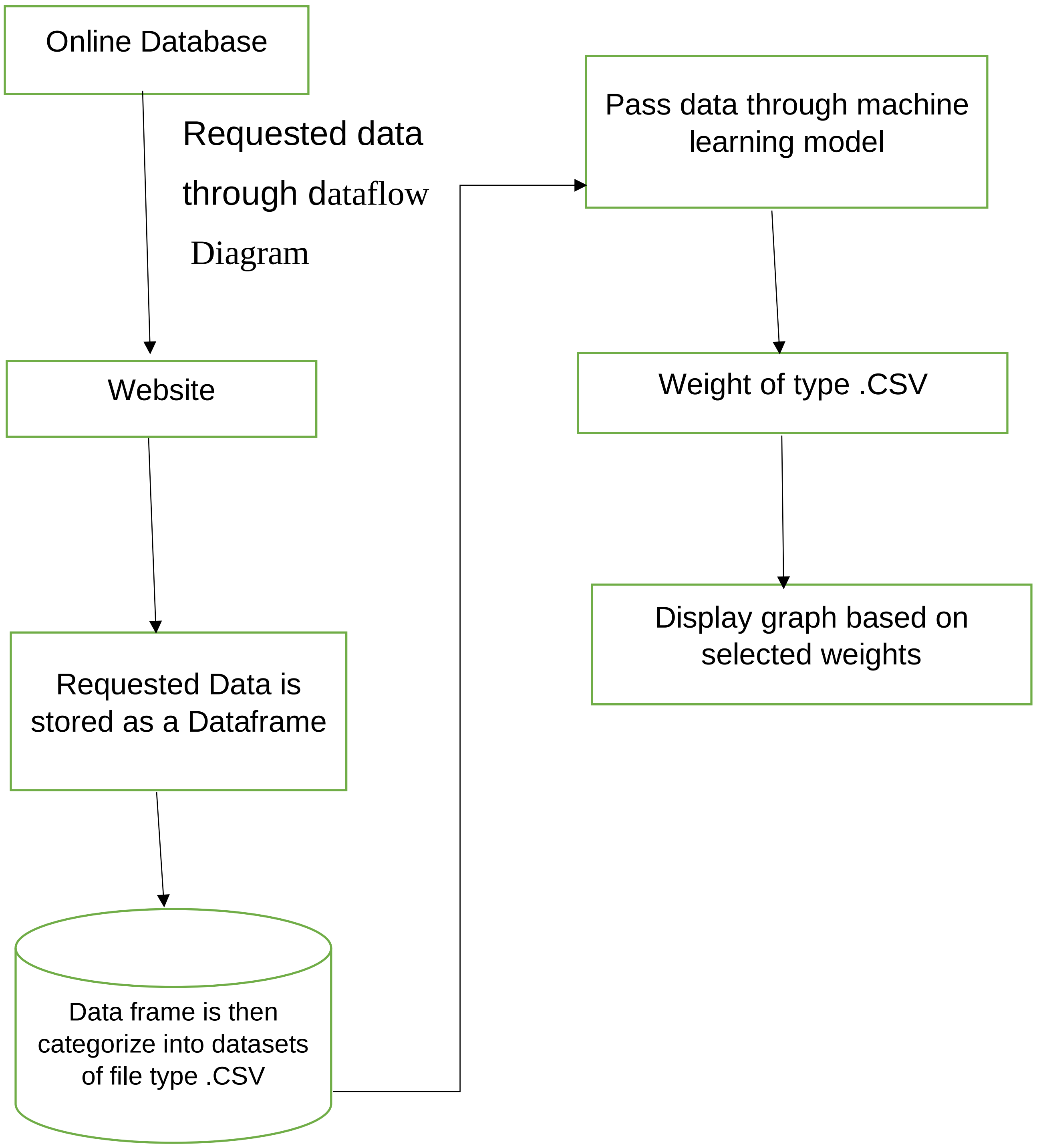


Fig no 5.1: block diagram for proposed workflow

CHAPTER 6

SYSTEM ANALYSIS

6.1 DATA FLOW DIAGRAM


Online Database
Requested data
through dataflow
Diagram
Website
Requested Data is stored as a Dataframe
Data frame is then categorize into datasets of file type .CSV
Pass data through machine learning model
Weight of type .CSV
Display graph based on selected weights

6.2 ARCHITECTURE OF THE PROPOSED MODEL

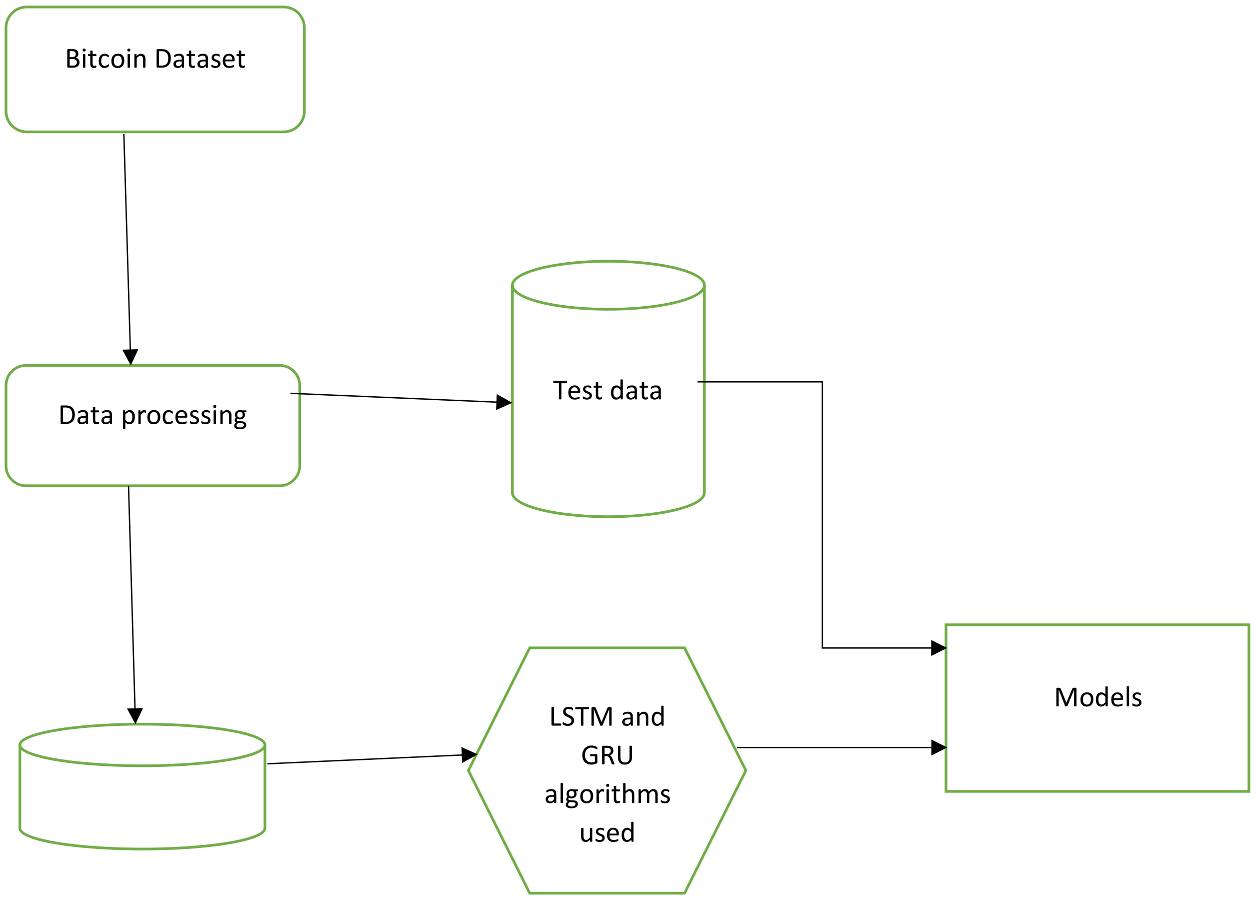


Fig no :6.2 Architecture of the proposed model

CHAPTER 7

SYSTEM DESIGN

7.1 SYSTEM ARCHITECTURE

The architecture of the proposed system has the following components:

- Dataset which consists of the prices that have occurred from day to day for 1 year.
- Training data to train the models.
- Testing data to apply the models.
- Data storage- stores data.
- Classification and prediction algorithms.
- Forecast engine

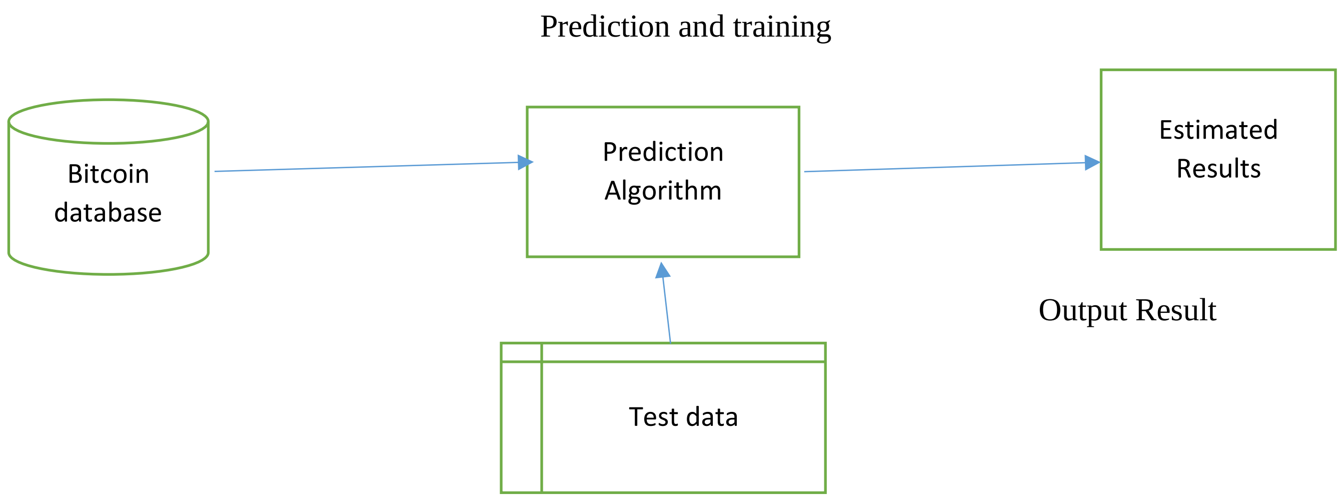


Fig no: 7.1 system Architecture

7.2 ALGORITHM USED

7.2.1 LSTM

Long short-term memory is an artificial recurrent neural network architecture used in the field of deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections. It can process not only single data points, but also entire sequences of data.Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning.LSTMs are often referred to as fancy RNNs. Vanilla RNNs do not have a cell state. They only have hidden states and those hidden states serve as the memory for RNNs. Meanwhile, LSTM has both cell states and a hidden states.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. This problem occurs while the optimizer back propagates, 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 time steps . The cell state is the horizontal line from Ct−1 to Ct, 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

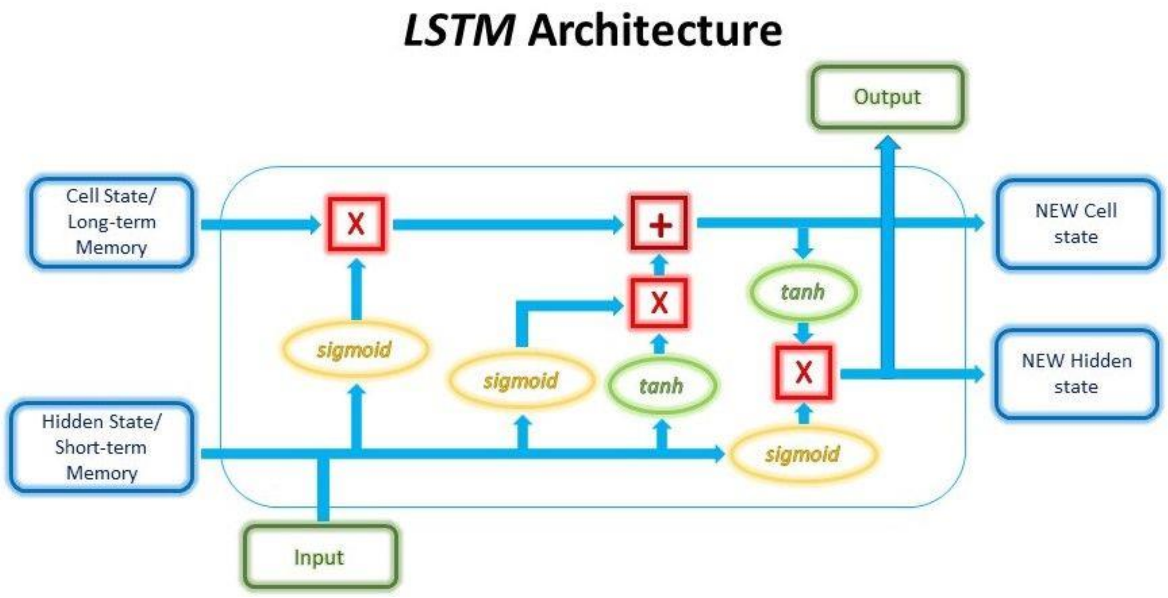


Fig no: 7.2.1 LSTM cell

• Forget gate: ft = σ(WfSt−1 + WfSt)

• Input gate: it = σ(WiSt−1 + WiSt)

• Output gate: ot = σ(WoSt−1 + WoSt)

7.2.2 GRU

Gated Recurrent Units (GRUs) are a type of RNN that were introduced by in 2014 as an improvement over the traditional LSTM networks. Like LSTMs, GRUs are designed to be able to process input sequences of arbitrary length and maintain a state that encodes information about the past. However, unlike LSTMs, which use multiple gates and an internal memory cell to control the flow of information, GRUs use a single update gate to decide which information to retain and a reset gate to decide which information to discard. This makes GRUs simpler and easier to train than LSTMs, while still being able to achieve similar performance on many tasks.

In a study by, GRUs were shown to outperform LSTMs on the task of language modeling on the Penn Treebank dataset. In a comparison of NLP models by, GRUs were found to be competitive with LSTMs and CNNs on several benchmarks. One advantage of GRUs is that they are able to capture long-range dependencies in sequential data more effectively than simple RNNs. This is because the update and reset gates in a GRU allow it to selectively retain or forget information from the past, depending on the current input and the state of the network. This makes GRUs particularly well-suited for tasks that require the ability to remember and use information from long sequences, such as language translation.

the hidden state at time *t*, ht$h$, is updated based on the input at time *t*, xt, and the previous hidden state, ht−1h−1, using the following equations :

$$ut=\sigma(Wu[ht-1,xt])=([h-1,])$$

(6)

$$rt=\sigma(Wr[ht-1,xt])=([h-1,])$$

(7)

$$ht=(1-ut)*ht-1+ut*\tanh(W[rt*ht-1,ut])h=(1-)*h-1+*\tanh([*h-1,])$$

(8)

where ut and rt is update and reset gate,respectively.

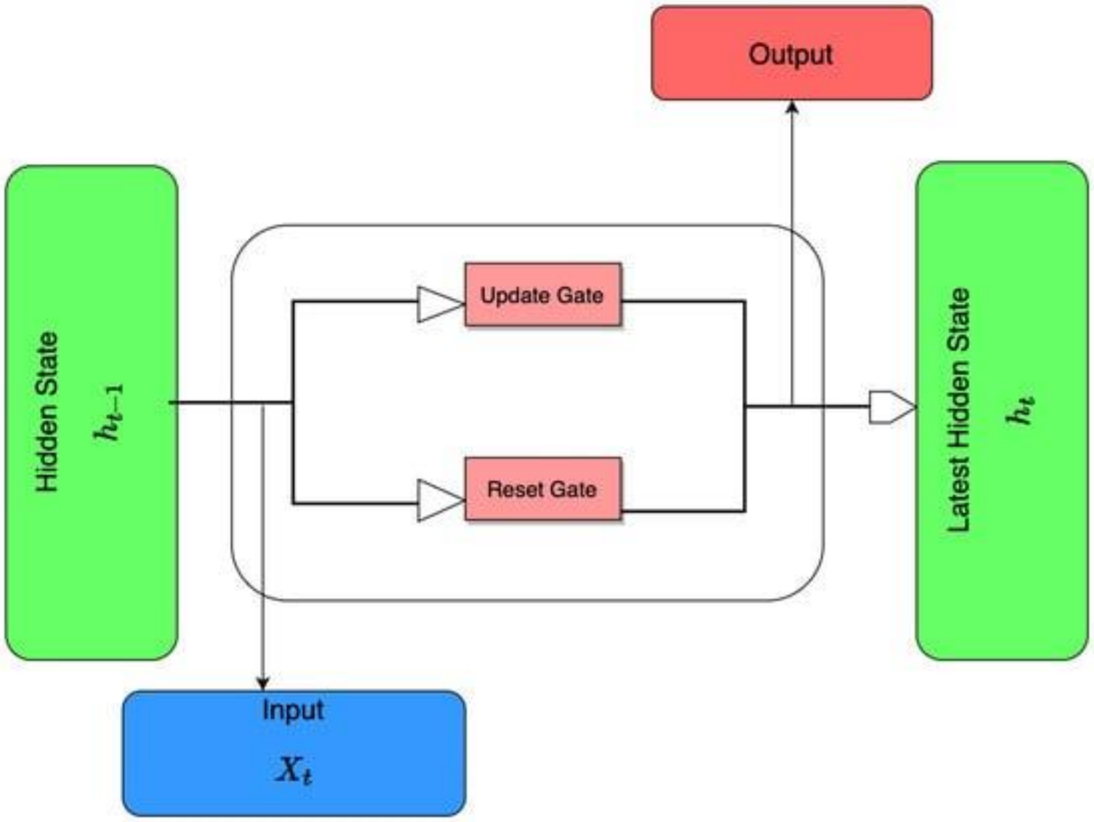


Fig no: 7.2.2 The diagram of a GRU cell.

Gated recurrent units are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al. The GRU is like a long short-term memory with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate.How do GRUs work? As mentioned above, GRUs are improved version of standard recurrent neural network. But what makes them so special and effective? To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.

CHAPTER 8

IMPLEMENTATION

8.1 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 January to decemeber (2022). 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 Jan 1 – Dec 31 (2022). 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. 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.

Open – Price (USD) of the first trade at the current time step

High - Highest Price (USD) of trades at the current time step

Low – Lowest price (USD) of trades at the current time step

Close – Price (USD) of the last trade before the next time step

Volume_BTC – Trade volume in Bitcoins

Volume_Currency – Trade volume in USD

Weighted_Price – Weighted Bitcoin price(USD)

Timestamp – Ten digits integer that represents time

8.2 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

Data preparation is the process of gathering, combining, organizing, and structuring data,which can then be used for data visualization, analytics, and data mining.It is critical to provide accurate data for the problemwe want to solve. Preparing data sets is an important step in machine learning.As previously stated, data preparation has an impact on prediction accuracy.
The dataset used for this research consists of daily price values collected from the website.The overall data collection period is from January 1 2022 to December 31.In this dataset , there are seven attributes such as opening price, high price, low price and closing prices, and also the market cap of publicly traded outstanding shares.

DATA CLEANSING

Data cleansing can be done by simply examining the associated Volume, Close, unlock, higher prices, and market capitalization from exchange data. If the NaN values are found to be correct in any data set , they are replaced with a description of the appropriate attribute. Following that, all datasets are merged into one, based on the magnitude of the time.

When we examine the bitcoin price fluctuations over the period 2022, we have seen it is better to remove data points prior, which is why the details that will be transferred to neural network are dormant.

8.3 DATA NORMALIZATION

Decide how to get used to timeline, particularly in finance, is not an easy task. Aside from that, the neural network must load data from a large number of different time series scales. This can result in significant gradient updates that prevent the network from changing. Data should have the following characteristics for easy reading on the network:

- Use small values-Most values should be in the 0-1 range.
- Be homogeneous, which means that all features should have values in the same range.

So we used Min-max normalization, which is one of the most common ways to normalize data. For every feature, the minimum value of the feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1.

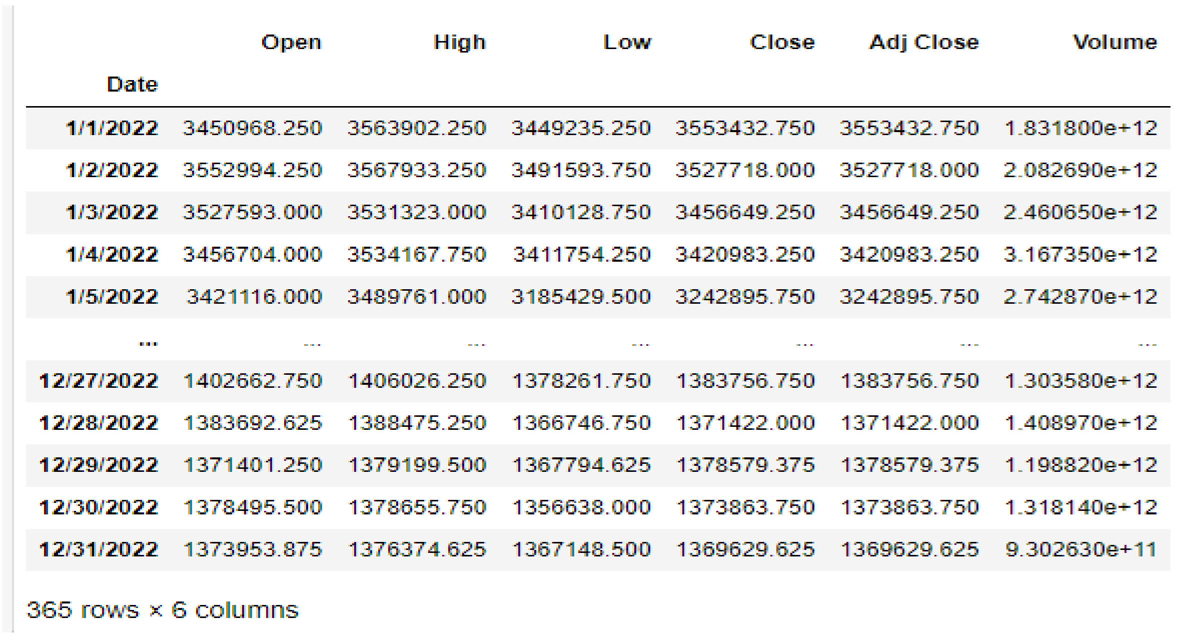
DATA TRAINING AND SPLITTING

The main goal is to test the ablility of the algorithm to predict the values. calculate the MAE error and RMSE for both GRU and LSTM algorithms. So I split the time series into training and validation sets with ratios of 80% and 20% respectively. We divide the data into training and data in statistics and machine learning. The model is intended fit the training data to make prediction. When we do this, there are chances of two things happening. One overfitting the model and the othe underfitting the model. Overfitting means that our model is too well trained, and the predictions are too close and that it does not fit closely with the model when the model is underfitting. The Scikit Library is used here to split the data.

8.4 CODING

LSTM CODE

```
#Import Libraries
import math
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
 #Reading data set and setting index as Date
df = pd.read_csv("c:\\Users\\Admin\\Documents\\Bitcoinprices.csv")
df = df.set_index('Date')
df
```



```
                   Open         High          Low        Close    Adj Close        Volume
      Date
  1/1/2022  3450968.250  3563902.250  3449235.250  3553432.750  3553432.750  1.831800e+12
  1/2/2022  3552994.250  3567933.250  3491593.750  3527718.000  3527718.000  2.082690e+12
  1/3/2022  3527593.000  3531323.000  3410128.750  3456649.250  3456649.250  2.460650e+12
  1/4/2022  3456704.000  3534167.750  3411754.250  3420983.250  3420983.250  3.167350e+12
  1/5/2022  3421116.000  3489761.000  3185429.500  3242895.750  3242895.750  2.742870e+12
       ...          ...          ...          ...          ...          ...           ...
12/27/2022  1402662.750  1406026.250  1378261.750  1383756.750  1383756.750  1.303580e+12
12/28/2022  1383692.625  1388475.250  1366746.750  1371422.000  1371422.000  1.408970e+12
12/29/2022  1371401.250  1379199.500  1367794.625  1378579.375  1378579.375  1.198820e+12
12/30/2022  1378495.500  1378655.750  1356638.000  1373863.750  1373863.750  1.318140e+12
12/31/2022  1373953.875  1376374.625  1367148.500  1369629.625  1369629.625  9.302630e+11

365 rows × 6 columns
```

```
#Droping null values

df=df.dropna()

#Showing the graph High,Low,Open,Close in our dataset. x=Date y=INR

fig=go.Figure()fig.add_trace(go.Scatter(x= df.index, y=
df.High,mode='lines',name='High',marker_color = '#2CA02C',visible = "legendonly"))

fig.add_trace(go.Scatter(x = df.index, y = df.Low,mode='lines',name='Low',marker_color =
'#D62728',visible = "legendonly"))

fig.add_trace(go.Scatter(x = df.index, y = df.Open,mode='lines',name='Open',marker_color =
'#FF7F0E',visible = "legendonly"))

fig.add_trace(go.Scatter(x = df.index, y = df.Close,mode='lines',name='Close',marker_color =
'#1F77B4'))

fig.update_layout(title='Closing price history',titlefont_size = 28,

xaxis = dict(title='Date',titlefont_size=16,tickfont_size=14),height = 800,

yaxis=dict(title='Price in INR (₹)',titlefont_size=16,tickfont_size=14),

legend=dict(y=0,x=1.0,bgcolor='rgba(255, 255, 255, 0)',bordercolor='rgba(255, 255, 255, 0)'))

fig.show()
```

Preprocessing

```
data = df.filter(['Close'])

dataset = data.values

training_data_len = math.ceil(len(dataset) * .8)


training_data_len


out[]: 292
```

```
#MinMaxScaler Formula
#X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
#X_scaled = X_std * (max - min) + min

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

```
out[]: array([[9.83420744e-01],
       [9.72246853e-01],
       [9.41365182e-01],
       [9.25867151e-01],
       [8.48482371e-01],
       [8.35044024e-01],
       [7.79817786e-01],
       [7.85575014e-01],
       [7.91682910e-01],
       [7.85146131e-01],
```

```
train_data = scaled_data[0:training_data_len, :]
x_train = []
y_train = []
for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
x_train.shape
```

```
out[]:(232, 60, 1)
```

LSTM

```
model = Sequential()
model.add(LSTM(50, return_sequences = True, input_shape = (x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences = False))
model.add(Dense(25))
model.add(Dense(1))
model.compile(optimizer = 'adam', loss = 'mean_squared_error') #dog=input output=cat
model.fit(x_train, y_train, batch_size = 1, epochs = 1)

test_data = scaled_data[training_data_len - 60: , :]
x_test = []
y_test = dataset[training_data_len:, :]

for i in range (60, len(test_data)):
    x_test.append(test_data[i - 60:i, 0])

x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

rsme = np.sqrt(np.mean(predictions - y_test) ** 2)
rsme

out[]:63565.981164383564
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
mape = np.mean(np.abs( y_test - predictions))
print('Mean absolute percentage error:' , mae , '%')


out[]:Mean absolute percentage error: 74400.875 %
```

```
fig = go.Figure()
fig.add_trace(go.Scatter(x = train.index, y = train.Close,mode='lines',name='Close',marker_color
= '#1F77B4'))
fig.add_trace(go.Scatter(x = valid.index, y = valid.Close,mode='lines',name='Val',marker_color =
'#FF7F0E'))
fig.add_trace(go.Scatter(x = valid.index, y = valid.Predictions,mode='lines',name='Predictions',m
arker_color = '#2CA02C'))

fig.update_layout(title='Model',titlefont_size = 28,hovermode = 'x',
                  xaxis = dict(title='Date',titlefont_size=16,tickfont_size=14),height = 800,
                  yaxis=dict(title='Close price in INR (₹)',titlefont_size=16,tickfont_size=14),
                  legend=dict(y=0,x=1.0,bgcolor='rgba(255, 255, 255, 0)',bordercolor='rgba(255, 255, 2
55, 0)'))
fig.show()
```

GRU CODE

```
#Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout,GRU
from keras import optimizers

seed = 1234
np.random.seed(seed)
plt.style.use('ggplot')
```

```
#Read Dataset
dataraw = pd.read_csv("C:\\Users\\Admin\\Documents\\Bitcoinprices.csv",index_col='Date', par
se_dates=['Date'])
dataraw

dataraw.describe()

#Feature Selection
# use feature 'Date' & 'Close'

dataset = pd.DataFrame(dataraw['Close'])
print(' Count row of data: ',len(dataset))

fig = plt.figure(figsize=(14, 6))
plt.plot(dataset)
plt.xlabel('Date')
plt.ylabel('Bitcoin Price')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Bitcoin Price')
plt.show()

#Preprocessing Data

#Normalization Min-Max

#Min-Max Normalization
dataset_norm = dataset.copy()
dataset[['Close']]
scaler = MinMaxScaler()
dataset_norm['Close'] = scaler.fit_transform(dataset[['Close']])
dataset_norm
```

out[]:

Date	Close
2022-01-01	0.983421
2022-01-02	0.972247
2022-01-03	0.941365
2022-01-04	0.925867
2022-01-05	0.848482
...	...
2022-12-27	0.040626
2022-12-28	0.035266
2022-12-29	0.038377
2022-12-30	0.036328
2022-12-31	0.034488

365 rows × 1 columns

```
#Graph Data Normalized

fig = plt.figure(figsize=(10, 4))

plt.plot(dataset_norm)

plt.xlabel('Date')

plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))

plt.title('Data Normalized')

plt.show()
```

```
#Data Partition

# Partition data into data train, val & test

totaldata = dataset.values

totaldatatrain = int(len(totaldata)*0.7)

totaldataval = int(len(totaldata)*0.1)

totaldatatest = int(len(totaldata)*0.2)

# Store data into each partition

training_set = dataset_norm[0:totaldatatrain]

val_set=dataset_norm[totaldatatrain:totaldatatrain+totaldataval]

test_set = dataset_norm[totaldatatrain+totaldataval:]

#Training Data Graph

# graph of data training

fig = plt.figure(figsize=(10, 4))

plt.plot(training_set)

plt.xlabel('Date')

plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))

plt.title('Data Training')

plt.show()

#Training Data Graph

# graph of data training

fig = plt.figure(figsize=(10, 4))

plt.plot(training_set)
plt.xlabel('Date')

plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))

plt.title('Data Training')

plt.show()

#Data Test Graph

# graph of data test

fig = plt.figure(figsize=(10, 4))
plt.plot(test_set)
plt.xlabel('Date')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m"))
plt.title('Data Test')
plt.show()
test_set

#Sliding Windows
# Initiaton value of lag
lag = 2
# sliding windows function
def create_sliding_windows(data,len_data,lag):
    x=[]
    y=[]
    for i in range(lag,len_data):
        x.append(data[i-lag:i,0])
        y.append(data[i,0])
    return np.array(x),np.array(y)
# Formating data into array for create sliding windows
array_training_set = np.array(training_set)
array_val_set = np.array(val_set)
```

```
array_test_set = np.array(test_set)
# Create sliding windows into training data
x_train, y_train = create_sliding_windows(array_training_set,len(array_training_set), lag)
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
# Create sliding windows into validation data
x_val,y_val = create_sliding_windows(array_val_set,len(array_val_set),lag)
x_val = np.reshape(x_val, (x_val.shape[0],x_val.shape[1],1))
# Create sliding windows into test data
x_test,y_test = create_sliding_windows(array_test_set,len(array_test_set),lag)
x_test = np.reshape(x_test, (x_test.shape[0],x_test.shape[1],1))
#Model GRU

# Hyperparameters

learning_rate = 0.0001

hidden_unit = 64

batch_size=256

epoch = 100

# Architecture Gated Recurrent Unit

regressorGRU = Sequential()

# First GRU layer with dropout

regressorGRU.add(GRU(units=hidden_unit, return_sequences=True,
input_shape=(x_train.shape[1],1), activation = 'tanh'))

regressorGRU.add(Dropout(0.2))

# Second GRU layer with dropout

regressorGRU.add(GRU(units=hidden_unit, return_sequences=True, activation = 'tanh'))

regressorGRU.add(Dropout(0.2))
```

# Third GRU layer with dropout

regressorGRU.add(GRU(units=hidden_unit, return_sequences=False, activation = 'tanh'))

regressorGRU.add(Dropout(0.2))

# Output layer

regressorGRU.add(Dense(units=1))

# Compiling the Gated Recurrent Unit

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

# Fitting ke data training dan data validation

pred = regressorGRU.fit(x_train, y_train, validation_data=(x_val,y_val), batch_size=batch_size, epochs=epoch)

out[]:

```
Epoch 1/100
1/1 [==============================] - 26s 26s/step - loss: 0.3506 - val_l
oss: 0.0109
Epoch 2/100
1/1 [==============================] - 0s 264ms/step - loss: 0.3116 - val_
loss: 0.0066
Epoch 3/100
1/1 [==============================] - 0s 71ms/step - loss: 0.2772 - val_l
oss: 0.0034
Epoch 4/100
1/1 [==============================] - 0s 120ms/step - loss: 0.2444 - val_
loss: 0.0011
Epoch 5/100
1/1 [==============================] - 0s 167ms/step - loss: 0.2066 - val_
loss: 2.1523e-04
Epoch 6/100
1/1 [==============================] - 0s 126ms/step - loss: 0.1758 - val_
loss: 8.8988e-04
Epoch 7/100
1/1 [==============================] - 0s 161ms/step - loss: 0.1437 - val_
loss: 0.0036
Epoch 8/100
1/1 [==============================] - 0s 211ms/step - loss: 0.1126 - val_
loss: 0.0087
Epoch 9/100
1/1 [==============================] - 0s 163ms/step - loss: 0.0842 - val_
loss: 0.0169
Epoch 10/100
```

```
#Graph Training loss & Validation Loss
# Graph model loss (train loss & val loss)

fig = plt.figure(figsize=(10, 4))

plt.plot(pred.history['loss'], label='train loss')

plt.plot(pred.history['val_loss'], label='val loss')

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(loc='upper right')

plt.show()

#Training Loss & Validation Loss Table

# Tabel value of training loss & validation loss

learningrate_parameter = learning_rate

train_loss=pred.history['loss'][-1]

validation_loss=pred.history['val_loss'][-1]

learningrate_parameter=pd.DataFrame(data=[[learningrate_parameter, train_loss,
validation_loss]],

                                columns=['Learning Rate', 'Training Loss', 'Validation Loss'])

learningrate_parameter.set_index('Learning Rate')
```

```
out[]:

                   Training Loss    Validation Loss
```

Learning Rate

0.00010	.0064230	.000374

```
#Implementation Model into Data Test

# Implementation model into data test

y_pred_test = regressorGRU.predict(x_test)

# Invert normalization min-max

y_pred_invert_norm = scaler.inverse_transform(y_pred_test)

#Comparison Data Test with Prediction Results

# Comparison data test with data prediction

datacompare = pd.DataFrame()

datatest=np.array(dataset['Close'][totaldatatrain+totaldataval+lag:])

datapred= y_pred_invert_norm

datacompare['Data Test'] = datatest

datacompare['Prediction Results'] = datapred

datacompare
```

		prediction results
	Data test	
0	1582514.750	1613915.500
1	1585463.250	1608917.750
2	1615091.125	1612523.875
3	1601073.500	1623637.625
4	1656814.125	1637426.375
...	...	...
67	1383756.750	1439628.000
68	1371422.000	1439988.750
69	1378579.375	1424688.375
70	1373863.750	1419421.750
71	1369629.625	1422304.000

72 rows × 2 columns

```
#Prediction Results Evaluation

# Calculatre value of Root Mean Square Error

def rmse(datatest, datapred):

    return np.round(np.sqrt(np.mean((datapred - datatest) ** 2)), 4)

print('Result Root Mean Square Error Prediction Model :',rmse(datatest, datapred))

def mape(datatest, datapred):

    return np.round(np.mean(np.abs((datatest - datapred) / datatest) * 100), 4)

    print('Result Mean Absolute Percentage Error Prediction Model : ', mape(datatest, datapred),
'%')
```

```
Result Root Mean Square Error Prediction Model : 187508.1888
Result Mean Absolute Percentage Error Prediction Model :  9.7851 %
```

```
# Create graph data test and prediction result

plt.figure(num=None, figsize=(10, 4), dpi=80,facecolor='w', edgecolor='k')

plt.title('Graph Comparison Data Actual and Data Prediction')

plt.plot(datacompare['Data Test'], color='red',label='Data Test')

plt.plot(datacompare['Prediction Results'], color='blue',label='Prediction Results')

plt.xlabel('Day')

plt.ylabel('Price')

plt.legend()

plt.show()
```

8.5 RESULT

Comparison of RMSE and MAPE

Algorithms	RMSE	MAPE
GRU	187508.1888	9.785%
LSTM	63565.981164	97278.246575%

In this study, two types of deep learning techniques LSTM and GRU were used predict the prices of Bitcoin as measured by their market capitalization. The performance of the models was evaluated using two scores RMSE (Root Mean Square Error) and MAPE (Mean Absolute Percentage Error). The result of the study showed that GRU model provided the most accurate predictions than the LSTM. The conclusion of the study is that deep learning algorithms are effective in predicting cryptocurrency prices. In future studies, the effect of tweets and sentiments on cryptocurrency prices will be explored using machine learning techniques.

CHAPTER 9

SCREEN SHOTS

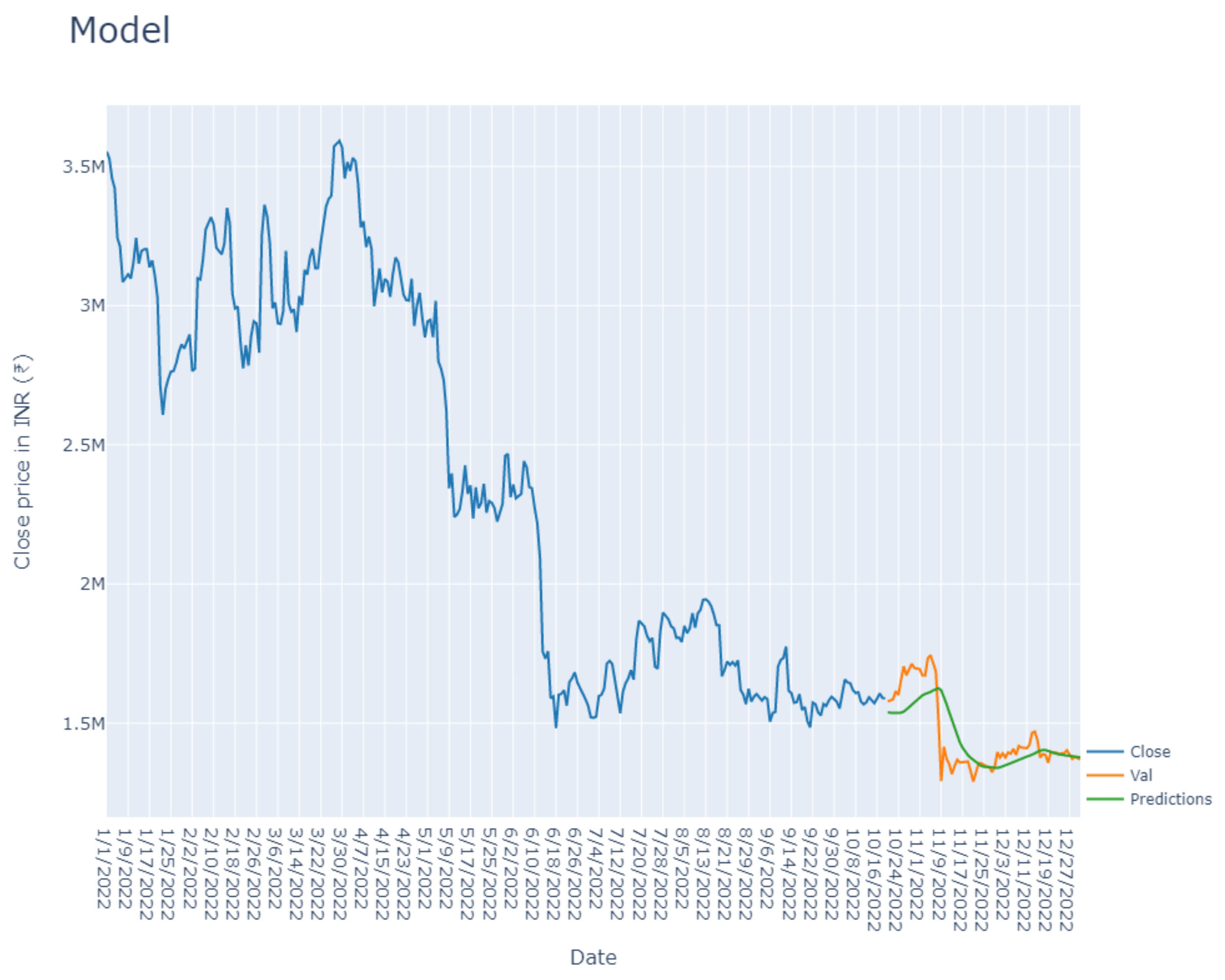


Fig no:9.1 LSTM Model

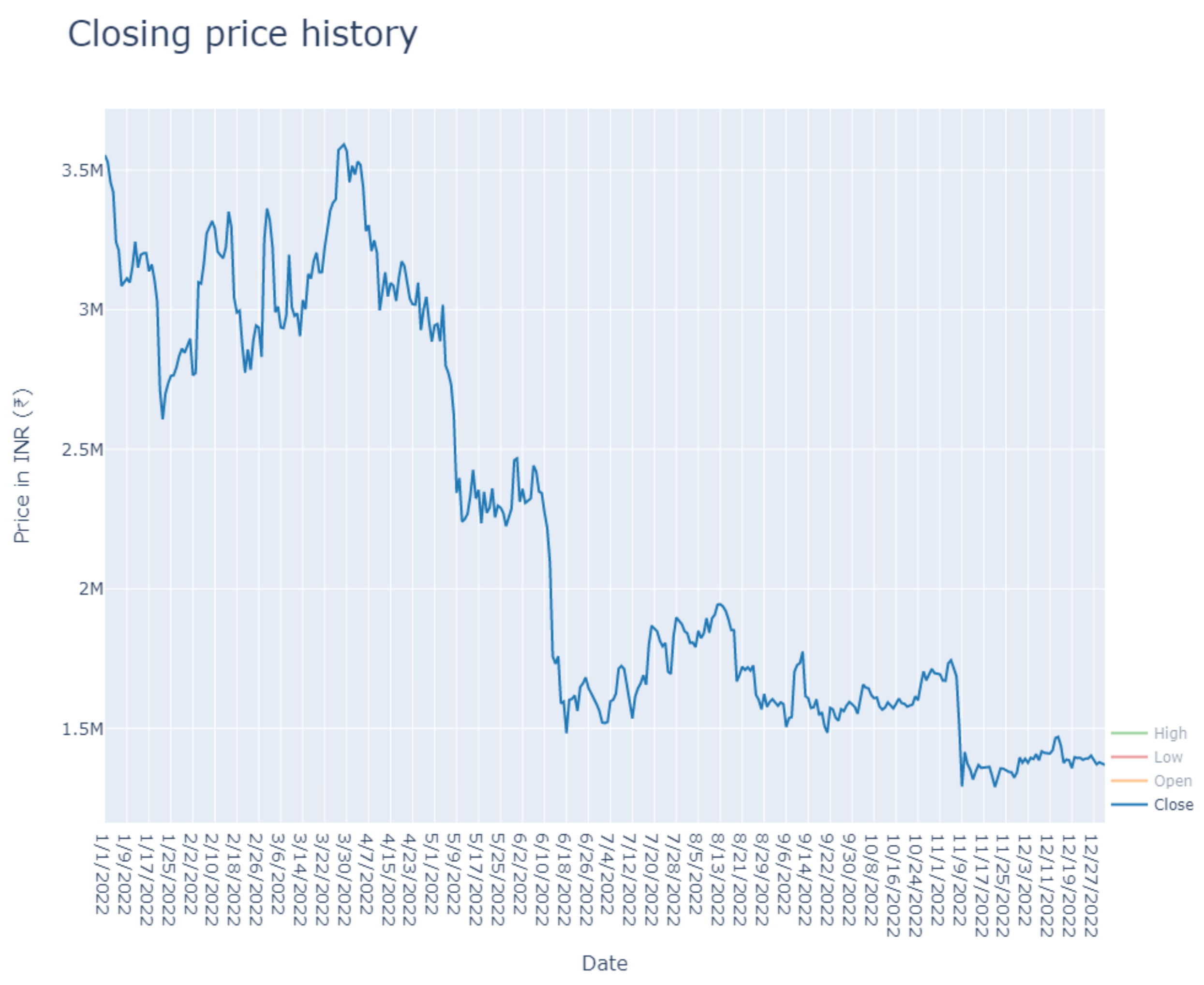


Fig no:9.2 LSTM closing price history

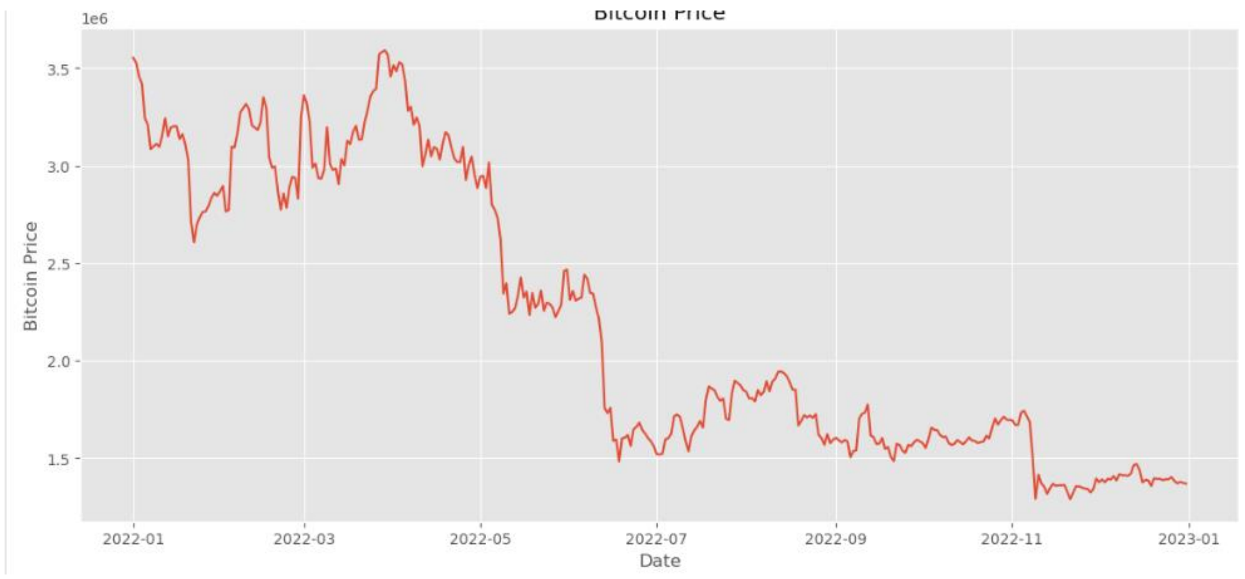


Fig no: 9.3 graph of GRU bitcoin price

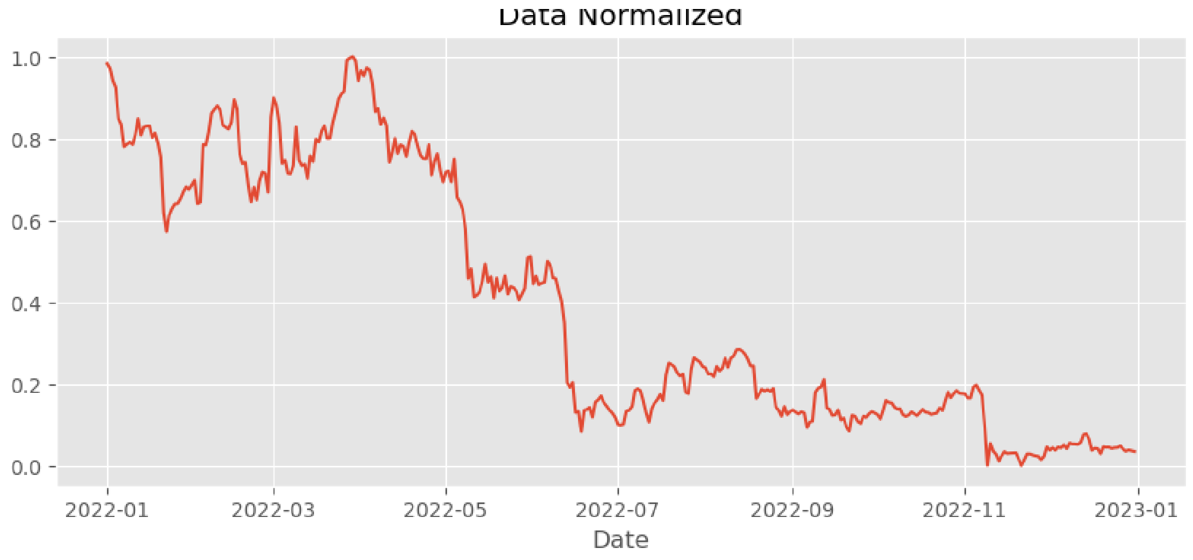


Fig no: 9.4 graph of GRU Data Normalized

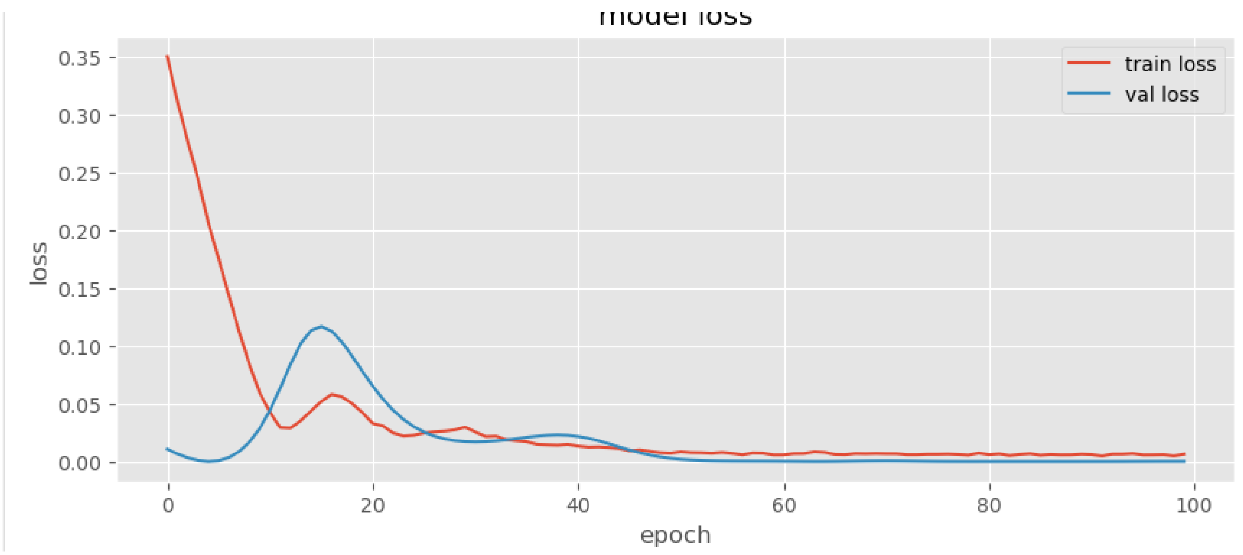


Fig no: 9.5 graph of GRU model loss

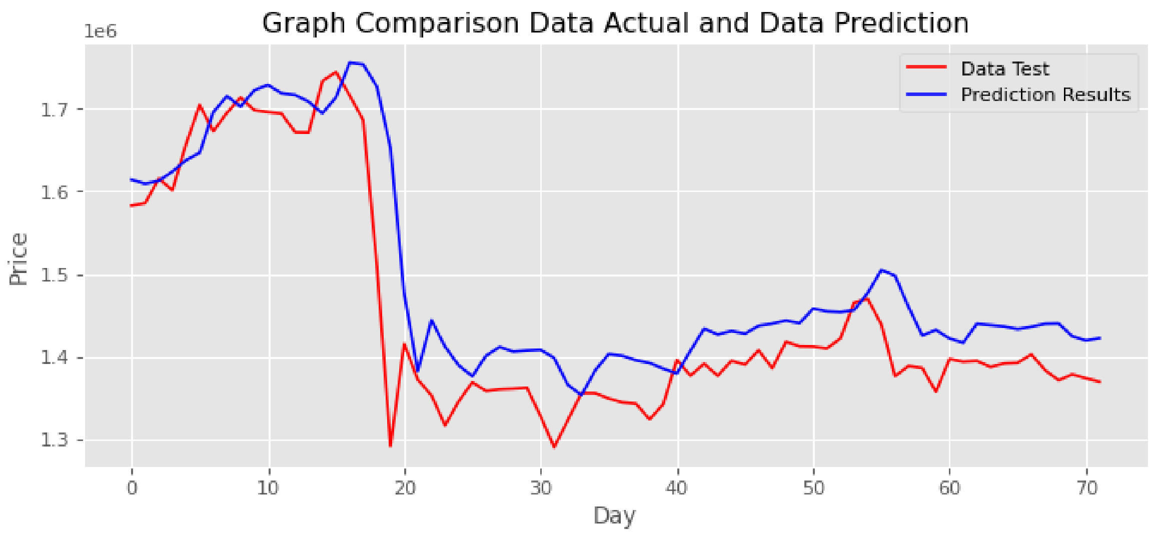


Fig no: 9.6 graph comparison of data actual and data prediction

CHAPTER 10

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.

10.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

10.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

10.3 SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s).System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

10.4 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.

CHAPTER 11

CONCLUSION AND FUTURE ENHANCEMENTS

11.1 CONCLUSION

Since Bitcoin and the Blockchain technology were introduced in 2008, it has taken a predominant place in the cryptocurrency field. There are millions of users around the world, especially in the United States. Predicting the price of cryptocurrencies has been a popular topic, from which we can take advantage of the arbitrage for an investment. Two methods were implemented: Long Short-term Memory Model and Gated Recurrent Units Model,with two different emission probabilities. We conclude the Gated recurrent units has perform best for smaller sequence dataset but LSTM also perform very well it is used for large dataset and predict more accurately, evaluated by the RMSE function and the MAPE function. . Additionally, there is an association between price movement and model accuracy. Therefore, we only include a few features in the dataset at the current step and assume it provides sufficient information, but in reality, Bitcoin might be more complex and correlated with other markets. As future work, we can 33 search for an alternative algorithm or package that is more efficient, and include additional information in the current dataset, such as the market index, to improve the performance. As bitcoin is growing drastically in recent times, it is very important to predict the price of it more accurately as it is going to be future trend of investment. If prices are predicted more accurately then it will be helpful for people to invest on it.

11.2 FUTURE ENHANCEMENTS

As bitcoin is growing drastically in recent years, there are few other type of cryptocurrencies like Ethereum, Litecoin, Monero, Zcash etc. So the same framework and normalization methods can be used for these different types of currencies for predicting the prices too. Another important future enhancement could be predicting the prices on a large time spectrum, i.e., for a longer period of time in the future. This would help the people and the companies to take an informed decision on when to and when not to invest.

CHAPTER 12

REFERENCES

[1] D. Shah and K. Zhang, "Bayesian regression and Bitcoin," in 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2015,pp. 409-415.

[2] Huisu Jang and Jaewook Lee, "An Empirical Study on Modelling and Prediction of Bitcoin Prices with Bayesian Neural Networks based on Blockchain Information," in IEEE Early Access Articles, 2017, vol. 99, pp. 1-1.

[3] F. Andrade de Oliveira, L. Enrique ZÃ¡rate and M. de Azevedo Reis; C. Neri Nobre, "The use of artificial neural networks in the analysis and prediction of stock prices," in IEEE International Conference on Systems, Man, and Cybernetics, 2011, pp. 2151-2155.

[4] M. Daniela and A. BUTOI, "Data mining on Romanian stock market using neural networks for price prediction". informatica Economica, 17,2013.

[5] Madan, S. Saluja, and A. Zhao, "Automated Bitcoin Trading via Machine Learning Algorithms."

Available:http://ai2-s2-pdfs.s3.amazonaws.com/e065/3631b4a476abf5276a264f6bbff40b132061.pdf.

[6] Hegazy, K. and Mumford, S. (2016), "Comparative Automated Bitcoin Trading Strategies." Available at: http://cs229.stanford.edu/proj2016/report/MumfordHegazy-Comparitive Automated BitcoinTrading Strategies-report.pdf

[7] J. Aungiers, "Multidimensional LSTM Networks to Predict Bitcoin Price," Jakob Aungiers, Available: http://www.jakob-aungiers.com/articles/a/Multidimensional-LSTM-Networks-to-Predict-Bitcoin-Price.

[8] D. Sheehan, “Predicting Cryptocurrency Prices With Deep Learning,” dashee87.github.io, 19-Nov-2017.

Available: https://dashee87.github.io/deep%20learning/python/predicting-cryptocurrency-priceswith-deep-learning/.

[9] A. Greaves and B. Au, “Using the Bitcoin Transaction Graph to Predict the Price of Bitcoin,” stanford.edu, 08-Dec-2015. [Online]. Available: http://snap.stanford.edu/class/cs224w-2015/projects_2015/Using_the_Bitcoin_Transaction_Graph_to_Predict_the_Price_of_Bitcoin.pdf.