

Time-Series Forecasting of Ethereum Price Using Long Short-Term Memory (LSTM) Networks

by

Mohammad Samin-Al-Wasee

18101578

Promee Shankar Kundu

18301295

Israt Mahzabeen

18101676

Tasnim Tamim

17301026

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering

Brac University

May 2022

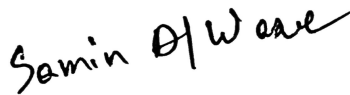
© 2022. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

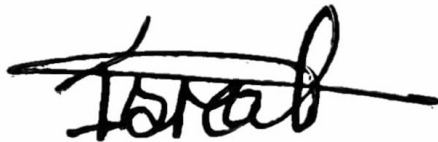
Student's Full Name & Signature:



Mohammad Samin-Al-Wasee
18101578



Promee Shankar Kundu
18301295



Israt Mahzabeen
18101676



Tasnim Tamim
17301026

Approval

The thesis titled “Time-Series Forecasting of Ethereum Price Using Long Short-Term Memory (LSTM) Networks” submitted by

1. Mohammad Samin-Al-Wasee (18101578)
2. Promee Shankar Kundu (18301295)
3. Israt Mahzabeen (18101676)
4. Tasnim Tamim (17301026)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 19, 2022.

Examining Committee:

Supervisor:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

In recent times, ether (ETH) has become one of the most popular cryptocurrencies that is gaining significant interest from crypto investors and developers across the globe. The increased interest in this cryptocurrency is due to the fact that transactions on the Ethereum platform are far more secure, as it combines smart contracts to streamline commerce and trade between both anonymous and recognized parties. Besides, many decentralized financial and nonfinancial apps (DeFi and DApps) are built mainly based on the ether cryptocurrency itself. As a result, the price of this cryptocurrency is also rising gradually. On the other hand, the price of ether sometimes decreases as well due to some unwanted circumstances like political conflicts, wars, natural disasters, and so on. Thus, the ether cryptocurrency market has become very unpredictable and can cause an uncertain situation for market investors. For this purpose, having a specialized prediction method for the ether price based on machine learning and deep learning technologies is crucial. This research aims to find an accurate price prediction model for the ether cryptocurrency based on the long short-term memory (LSTM) network, which is a special variant of the recurrent neural network (RNN). In the proposed model, ether price data was taken in time-series format and fitted into multiple basic and hybrid variants of the LSTM network, and the future prices were predicted based on both univariate and multivariate time-series analysis. Furthermore, a comparative analysis was conducted among the models and also some popular existing forecasting techniques like autoregressive integrated moving average (ARIMA) as the baseline forecast to determine which one can provide the best possible accuracy so that investors may understand the behaviour of the ether market and make proper decisions on their investment.

Keywords: Cryptocurrency, Deep Learning, Ether, Ethereum, Forecasting, Long Short-term Memory, Multivariate, Price Prediction, Recurrent Neural Network, Time-series, Univariate.

Dedication

To our loving parents who motivated us to achieve excellence in every aspect.

Acknowledgement

Firstly, we are grateful to the almighty Allah for giving us the opportunity to complete our thesis without any kind of significant interruption.

Secondly, we express our heartiest appraisals for our honourable supervisor Dr. Md. Golam Rabiul Alam, who always guided us to the right direction throughout the whole work and also our honourable lecturer, Mr. Md. Tanzim Reza for enriching us by providing with valuable knowledge and advice from time to time.

Lastly, we want to thank our beloved parents for always supporting us in our well and woes.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Dedication	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Tables	ix
Nomenclature	xi
1 Introduction	1
1.1 Background	1
1.2 Motivation	1
1.3 Problem Statement	2
1.4 Research Objective	2
1.5 Thesis Outline	3
2 Literature Review	4
2.1 Ethereum Network and Ether	4
2.2 Ethereum Price Prediction	4
2.3 Time Series Analysis	5
2.4 Autoregressive Integrated Moving Average (ARIMA)	6
2.5 Recurrent Neural Networks (RNN)	6
2.6 Long Short-Term Memory (LSTM)	8
2.7 Related Works	9
3 Methodology	13
3.1 Workflow	13
3.2 Hybrid Model Architecture	14
3.3 Data Collection, Preparation, Visualization	15
3.3.1 Acquiring Ethereum Historical Price Data	16

3.3.2	Preparing Collected Data	16
3.3.3	Data Visualization	18
3.4	Baseline Forecasting Using ARIMA	20
3.4.1	Making The Data Stationary	21
3.4.2	Train-Test Split	22
3.4.3	Setting Up The Parameters	22
3.4.4	Final ARIMA Model	24
3.5	Building LSTM Networks	24
3.5.1	Univariate Time-Series Analysis	24
3.5.2	Multivariate Time-Series Analysis	31
3.6	Evaluation Metrics	33
4	Result, Analysis & Comparison	34
4.1	Baseline (ARIMA) Results	34
4.2	LSTM Results	36
4.2.1	Results of Univariate Analysis	36
4.2.2	Results of Multivariate Analysis	39
4.3	Accuracy Comparison	41
4.4	Discussion & Analysis	41
5	Conclusion & Future Work	43
5.1	Conclusion	43
5.2	Future Work	43
	Bibliography	47

List of Figures

3.1	Workflow of proposed methodology	13
3.2	A vanilla LSTM unit[16]	14
3.3	Bidirectional LSTM layer[19]	14
3.4	Hybrid LSTM architecture	15
3.5	Ether closing prices	19
3.6	Ether price trend	19
3.7	Ether price seasonality	20
3.8	First order difference of closing prices	21
3.9	First order difference trend	21
3.10	Training & testing data	22
3.11	PACF of first order difference	23
3.12	ACF of first order difference	23
3.13	Train, validation & test split for LSTM	25
3.14	Distribution plot	26
3.15	LSTM input tensor[20]	27
3.16	Vanilla LSTM architecture	29
3.17	Stacked LSTM architecture	30
3.18	Bidirectional LSTM architecture	31
3.19	Correlation among the features	32
3.20	Stacked-bidirectional LSTM architecture	33
4.1	Short-term prediction using ARIMA	34
4.2	Long-term prediction using ARIMA	35
4.3	Training & validation losses of vanilla LSTM	36
4.4	Training & validation losses of stacked LSTM	37
4.5	Training & validation losses of bidirectional LSTM	37
4.6	Prediction using vanilla LSTM	38
4.7	Prediction using stacked LSTM	38
4.8	Prediction using bidirectional LSTM	39
4.9	Training & validation losses of hybrid model	40
4.10	Prediction using hybrid LSTM model	40
4.11	Accuracy comparison among the models	41

List of Tables

3.1	Data info before time series	17
3.2	Data info after time series	17
3.3	Dataset before preparation	18
3.4	Dataset after preparation	18
3.5	ADF test of original data	20
3.6	ADF test after first order difference	22
3.7	ARIMA model summary	24
3.8	Kolomogorov-Smirnov test results	26
3.9	Shapiro-Wilk test results	26
3.10	Data properties before and after standardization	27
4.1	Performance metrics of ARIMA(short-term)	35
4.2	Performance metrics of ARIMA(long-term)	36
4.3	Performance metrics of basic LSTM networks	39
4.4	Performance metrics comparison of stacked and hybrid LSTM	41

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ACF Autocorrelation Function

ADF Augmented Dickey-Fuller

AR Auto-Regressive

ARIMA Auto-Regressive Integrated Moving Average

DApp Decentralized App

DeFi Decentralized Finance

DType Data Type

EOA Externally Owned Account

GRU Gated Recurrent Unit

LR Linear Regression

LSTM Long Short-Term Memory

MA Moving Average

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

MSE Mean Squared Error

PACF Partial Autocorrelation Function

ReLU Rectified Linear Unit

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

SARIMA Seasonal Auto-Regressive Integrated Moving Average

SARIMAX Seasonal Auto-Regressive Integrated Moving Average with eXogenous factors

SVM Support Vector Machine

SVR Support Vector Regression

TCN Temporal Convolutional Network

Chapter 1

Introduction

1.1 Background

Ether stands for “Ethereum Money” and is the digital currency used by Ethereum platform apps. It is a functional currency used on the Ethereum platform and works as the fuel for operating the network to make transactions easier. Although internet money is a new idea, it is protected by well-established encryption and a security layer that safeguard the digital currency and its transactions. As a result, ether’s popularity is increasing rapidly. Ether is a decentralized and worldwide cryptocurrency which means no corporation or banking institution has the authority to produce additional ether or modify the rules of usage. Even though ether is considered a secured investment sector, its price often goes through rises and falls. One of the major reasons for the rise of ether price can be due to handling multiple decentralized finance (DeFi) projects. Next, updating the Ethereum network also affects the price hike of ether since it is making transactions faster than before. Also, investors are getting more interested in investing in the ether market, which leads to its price rise. Also, due to the COVID-19 pandemic and conflict among the economically dominant countries of the world, the price of ether experienced a fall in recent times like many other crypto markets. Ethereum is a decentralized platform which facilitates peer-to-peer connections with smart contracts where ether is used as a digital currency to operate the network fruitfully.

1.2 Motivation

Ether can be a very good investment for investors as its value keeps rising from time to time. Researchers predict that ether can play a significant role in the marketplace as it is one of the largest selling cryptocurrencies. Moreover, ether is a blockchain-based technology, so it is more secure than a traditional financial medium. As a result, ether has recently been the subject of discussion, garnering it the acceptability that it presently enjoys. In order to invest in a marketplace, it is important to understand the behaviour of that market. Generally, the price of any product rises and falls during a time interval which is referred to as time-series data that has multiple characteristics like trend, seasonality, noise and so on which may cause fluctuations and disturbance in the data. Analyzing the time series data helps to address these issues so that the future of a marketplace can be predicted. There are multiple algorithms used for time series data prediction but in recent times recurrent

neural networks, specifically algorithms like LSTM that can overcome issues like vanishing gradient, are getting popularity among the researchers.

With the unpredictability in the crypto market and the huge popularity of numerous altcoins, we decided to dive more into this sector, notably ether. In this work, we used time-series analysis approaches to explore several models and apply them to anticipate ether prices, as well as using the error metrics to assess the effectiveness of the implemented models.

1.3 Problem Statement

Ethereum and blockchain are two cutting-edge and rapidly expanding technologies at the moment. Ether can be used to pay for different services as a form of cryptocurrency. Besides, this also helps in the development process of the Ethereum platform as a reliable payment gateway. Being the second-largest cryptocurrency, ether paves a better way to move towards that technologically sound world. Since mining and transactions are safe and dependable, the revolutionary technology has provided new frontiers and possibilities for several ether miners, who are already speculating on whether this might be a new tangible property as well as economic security. Investors are also getting interested in investing in the ether market due to its huge number of possibilities in the near future. However, due to the volatile nature of the ether market, many people find it difficult to keep pace with the market movement. There are various reasons that are driving researchers across the globe to gain an interest in the ether market and predict its future. For instance, making an investment in this sector safe for the investors, determining the future of the ether market along with other cryptocurrencies, getting an idea about how rapidly digital money can replace traditional currency, and so on. Moreover, using time-series analysis is necessary in case of making these predictions to deal with the characteristics that affect the values. Consequently, approaches based on machine learning and deep learning can be used to make the prediction process much more accurate than determining it by making intuitive decisions. This research aimed to ensure secure investment for ether market investors by analyzing time-series data and predicting future prices through a statistical model like ARIMA and some basic and hybrid variants of RNN-based models like LSTM and making a comparative analysis in order to determine the best-performing model.

1.4 Research Objective

This research focused on predicting the price of Ether forecasting time series data. Ether investment becomes more prominent, more customers are expected to opt with the more accessible choice. Consequently, numerous people who might normally finance in much more traditional ways are thinking of switching to cryptocurrencies. Despite the fact that the price is so unpredictable, virtual currencies offer a distinct financial instrument that is being acknowledged as a legitimate unit of account. As a result, the goal of this thesis lied on examining several models, utilizing them to forecast the Ether price using time series data, and evaluating overall efficiency on a particular dataset in assisting market participants in identifying the risk associated

with Ether funding.

To sum up the goal of this research can be noted as the following:

- To gain a thorough understanding of ether market price and its fluctuations
- To have an idea about ARIMA and its efficiency in both short-term and long-term forecasting
- To have a thorough understanding of RNN and LSTM networks
- To process time-series data using basic LSTM networks and understand the features and variables
- To combine multiple basic LSTM layers and build a hybrid variant
- To compare and understand the performance and efficiency of ARIMA with basic and hybrid LSTM networks in ether price prediction
- To predict the future of ether utility by analyzing its price.

1.5 Thesis Outline

The research report is organized as follows-

- **Chapter 1** contains the background analysis, motivation behind the research, problem statement and research objectives.
- **Chapter 2** is comprised of necessary theoretical concepts including previous works related to this research.
- **Chapter 3** contains the workflow, architecture and methodology for our research.
- **Chapter 4** includes the result analysis and performance comparison of the implemented techniques.
- **Chapter 5** discusses some future goals and concludes the report.

Chapter 2

Literature Review

2.1 Ethereum Network and Ether

Ethereum is one of the most famous blockchain-based platforms because of its cryptocurrency, which is ether. Furthermore, the Ethereum network is considered safe due to its decentralized nature and security protocol that allows the price hike of ether. Moreover, Ethereum has a large community worldwide because it is totally open source. Additionally, it facilitates a range of applications including DeFi and dApps that can be built using the Ethereum blockchain network. An account is necessary to make transactions through the Ethereum network. Generally, a private key and a public key identify each account. The sender signs the transactions using EOA's private key and a hash value is returned, which can be used to monitor all blockchain transactions after confirmation. EOAs are through which users submit financial transactions. On the other hand, internal transactions do not have a signature feature. Subsequently, the transaction (code) transmits information from an EOA. Ethereum transactions are similar to basic blockchain transactions. The change in the state of the blockchain starts with a transaction sent by EOA. Indeed, this could be a direct payment of ether towards another debit or contract trigger. Consequently, the miner finds out the nonce value while numerous transactions are sent from the same account [9]. According to [10], the currency unit of Ethereum is known as ether, also identified as "ETH". Ether can be used to purchase many valuable stocks and commodities. Developers may design a reliable, auto-financial agreement (smart contract) that would transfer ether in the future by using the Ethereum network. This approach might theoretically allow for long-term financial contracts, providing contract participants with a motivation to maintain and use ether as a measure of wealth [8].

2.2 Ethereum Price Prediction

According to R. Bohme, N. Christin, B. Edelman, and T. Moore, the crypto market has been a significant component of the worldwide financial system. However, cryptocurrency values fluctuate dramatically, influencing trading behaviour. There are also shallow market difficulties, competitor hazards, purchase threats, potential

losses, confidentiality threats, and policy and institutional threats regarding market uncertainties [5]. Due to these fluctuations happening in the market, it becomes essential for the ether market investors to understand the behaviour pattern of the market in depth. As we know, Ethereum is built on highly secured technology like blockchain and smart contracts, hence it is considered a secured financial platform. Financial investment is a crucial topic for discussion. Soon, when technology takes over almost every sector across the globe along with the financial one, people will surely search for a secured online platform where their money will be safe. In fact, the procedure has already begun. Investors are gradually getting interested in investing in various cryptocurrencies. Ether is also included in the race as it is the second-largest cryptocurrency. So, if the investors can get an idea about how the price of ether actually behaves under which circumstances, it will be beneficial for them while investing. This is one of the significant reasons why ether price prediction is essential.

2.3 Time Series Analysis

The process of evaluating and interpreting the features of any sequence of data points collected over a defined period of time is referred to as time-series data analysis. According to E. Parzen, a time series is a collection of observations that are organized in sequential order [1]. Another journal [3] states that a time series is a collection of measurements of an observable variable taken at regular intervals in the past. Time series can be examined for a variety of reasons, including forecasting the future based on past information. Time-series data analysis plays a vital role in almost every aspect of science and engineering, understanding the phenomenon behind the measures or just a concise explanation of the series' key elements. As mentioned in [2], time-series analysis can be of two types: univariate and multivariate. A univariate time-series data uses only one variable observed over time. On the other hand, multivariate time-series data comprises more than one variable data.

Since data points do not always fulfill the conditions for fitting into any model, the first step after collecting time-series data is to prepare the dataset for analysis and forecasting. Trend and seasonality are two prominent characteristics of time-series data that refers to the gradual upward or downward shift in a series' level, or the tendency for the values to increase or decrease with repetitiveness and predictivity respectively. These features of time-series data are used and analyzed to bring precision while forecasting. Furthermore, stationarity in time-series data means that the properties of data like mean, variance, and autocorrelation structure stay constant over time. If not then the dataset is considered non-stationary[6]. Usually, in case of time-series forecasting, removing trend, seasonality, and non-stationarity from the dataset is an essential preprocessing step. However, researchers suggest it is not a necessity to check data stationarity while using an LSTM model. According to[15], the LSTM method is more applicable than other existing algorithms because it can learn the non-linear and non-stationary nature of a time series, reducing prediction error. After processing and analyzing the collected data, the above-mentioned machine learning models are used to interpret them and make certain

assumptions based on the characteristics of the dataset.

2.4 Autoregressive Integrated Moving Average (ARIMA)

ARIMA model is one of the most extensively used time series prediction methods introduced by Box and Jenkin in 1970. ARIMA models belong within the statistical model group which is resilient and efficient in financial time-series forecasting, particularly short-term prediction, as stated by A. A. Adebisi, A. O. Adewumi, and C. K. Ayo in [4]. The model is a linear regression model for tracking linear trends in stationary time-series data, with future values derived from linear functions observed in the past [11].

According to [22], The ARIMA(p, d, q) model essentially applies d-order differential processing to non-stationary historical data $Y(t)$ to create a new stable history sequence $X(t)$, fits the $X(t)$ to the ARIMA(p, q) model, and then restores the original d-time differential restoration to obtain the predicted data of $Y(t)$. The generic formulation of ARIMA(p, q) is shown in the following formula.

$$Y(t) = \phi(0) + \phi(1)Y(t-1) + \dots + \phi(p)Y(t-p) + \epsilon(t) - \theta(1)\epsilon(t-1) - \dots - \theta(q)\epsilon(t-q)$$

$Y(t)$ is the actual value and $\epsilon(t)$ is the random error at t, $\phi(i)$ and $\theta(j)$ are the coefficients, p and q are integers that are often referred to as autoregressive and moving average, respectively.

The autoregressive part of the equation is the first half, with p denoting the autoregressive order and $\phi(p)$ denoting the autoregressive coefficient. The second half is the moving average section where q is the moving average order, and $\theta(q)$ is the moving average coefficient. Furthermore, $Y(t)$ is the random error, and $Y(t)$ is the correlation sequence of the consumed stock data. Model-identification, parameter estimation, and diagnostic checking are the three phases of this procedure.

To illustrate, the ARIMA(1, 1, 1) model can be represented like the following.

$$Y(t) = \phi(0) + \phi(1)Y(t-1) + \epsilon(t) - \theta(1)\epsilon(t-1)$$

2.5 Recurrent Neural Networks (RNN)

According to Z. C. Lipton in[7], RNNs are a strong superset of feedforward neural networks, with the addition of recurrent edges that span consecutive time steps,

allowing the model to have a sense of time. Additionally, in [15] Preeti, R. Bala, and R. P. Singh argued that RNN is a type of neural network that computes current output using the current input and output received from past input. However, they also noted that standard RNNs are unable to handle the problem of exploding and vanishing gradients. Thus, long short-term memory (LSTM) is used.

A. Biswal in one of his papers says that the working mechanism of an RNN is almost similar to the traditional feedforward networks. In general, it contains an input layer, a middle layer, and an output layer along with an additional hidden state. To begin with, the hidden state is the information coming from the previous steps' inputs. Next, the input layer takes input and passes that to the middle layer after processing. Lastly, the middle layer is made of multiple hidden layers having their own activation functions, weights, and biases. The RNN standardizes the activation functions, weights, and biases so that each hidden layer gets the same characteristics. The hidden state has a loop to pass the previous information forward for future processing. The looping mechanism allows the information to travel from one step to another. Instead of constructing several hidden layers, it generates just one and loops over it as many times as needed. As mentioned in [13] by R. Madan, and P. S. Mangipudi, the fast self-learning and self-adapting ability of neural networks while given a series of input points with pre-defined outputs allows them to model and predict complicated nonlinear patterns. RNNs create cycles between the neurons which retain data and transfer feedback from one neuron to the next. This mechanism creates an internal memory that aids in the learning of data that is sequential in nature. Because it incorporates loops, an RNN allows information to be carried while input is being read. This distinguishes them from other neural networks.

The working mechanism is shown mathematically in [7] like the following: At time t , nodes receiving input along recurrent edges receive input activation from the current example " $x(t)$ " and also from hidden nodes " $h(t-1)$ " in the network's previous state. The output " $\hat{y}(t)$ " is calculated given the hidden state " $h(t)$ " at that time step. Thus, input " $x(t-1)$ " at a time " $t-1$ " can influence the output " $\hat{y}(t)$ " at a time " t " by way of these recurrent connections. We can show in two equations, that all calculations necessary for computation at each time step on the forward pass in a simple recurrent neural network:

$$h(t) = \sigma(W(hx) * x + W(hh) * h(t - 1) + bh)$$

$$\hat{y}(t) = softmax(W(yh) * h(t) + by)$$

Here " $W(hx)$ " is the matrix of weights between the input and hidden layers and " $W(hh)$ " is the matrix of recurrent weights between the hidden layers at adjacent time steps. The " bh " and " by " vectors are biases that allow each node to learn an offset.

2.6 Long Short-Term Memory (LSTM)

LSTM is a machine learning algorithm that solves the specified issue of exploding and vanishing gradients [17]. As per this article, the LSTM model is made of one or more memory modules that work as its basic units. Additionally, each module comprises memory units that control the information flow of the system. Moreover, in the LSTM structure, there is a function for memorizing time-series data.

Unlike RNN, LSTM has the ability to memorize time-series data because these cell units contain three individual logic gates based on a sigmoid neural network layer. The logic gates are known as input gates, output gates, and forget gates that are used to selectively pass or process the data. Firstly, the input gate summarizes the cell unit status value, filtered value, and added value in order to create a new value. Secondly, forget gate generates an output ranging from 0 to 1, referring to a value that can be ignored or reserved respectively inside the system. Additionally, a storage gate is used in the system that contains a sigmoid layer and a tanh layer. This gate is for selecting new data to store in the cell. Here, the sigmoid layer selects the value that requires adjustment and the tanh layer's task is to generate vectors of new candidate values followed by adding them to the cell unit state [17].

In [15] it is stated that The LSTM network design is a sequential model with two key components: states and gates. The hidden state is the value of the previously hidden layer, whereas the input state is a linear combination of current input data and the hidden state. An optimizer function is used by each unit of the LSTM cell network, which consists of the three gates, to update the weights associated with the network's units. At the forget gate, "f(t)" is computed using the following equation in order to find which information from the previous state to be kept for further computation:

$$f(t) = \sigma(W(fx) * x(t) + W(fs) * s(t - 1) + b(f))$$

where, " σ " is the sigmoid activation function.

Next, the input gate is used to find an intermediate parameter i(t) and C(t) using the given equations in order to determine if the internal state values serve as a memory cell or not.

$$i(t) = \sigma(W(ix) * x(t) + W(is) * s(t - 1) + b(i))$$

$$c(t) = \tanh(W(cx) * x(t) + W(cs) * s(t - 1) + b(c))$$

Finally, the information to be kept is derived by merging the outputs of the input and forget gates:

$$C(t) = f(t) * C(t - 1) + i(t) * c(t)$$

To compute new information to be stored in cell state, the sigmoid and tanh layers are used. The output layer then generates the output using the o(t) equation, which is then utilized to predict the final output, s(t).

$$o(t) = \sigma(W(ox) * x(t) + W(os) * s(t - 1) + b(o))$$

$$\tilde{s}(t) = o(t) * \tanh(C(t))$$

Lastly, the functions of “W” and “b” denotes the corresponding weights and biases used at different layers, and $\tilde{s}(t)$ denotes the output of LSTM network at time signal t.

There are multiple variants of the LSTM networks such as vanilla, stacked, bidirectional, and so on. To begin with, a vanilla LSTM contains a single unit of LSTM in the hidden layer and one single output layer. As discussed in [7], with the inclusion of the forget gate and peephole connections, the vanilla LSTM is understood as the original LSTM block. Furthermore, the authors of the paper said that while eight different LSTM variations have been identified, the vanilla architecture performs well in a variety of applications. Alternatively, an LSTM model with multiple LSTM layers is known as a stacked LSTM architecture. A. Graves first proposed the stacked LSTM, also known as deep LSTM, in [7], and it was used to solve voice recognition challenges. According to [23], the stacked LSTM model uses numerous LSTM layers that are layered before forwarding to a dropout layer and output layer at the final output, similar to the framework that underpins the RNN model. The first LSTM layer of a stacked LSTM creates sequence vectors that are utilized as input to the subsequent LSTM layer. On the other hand, a bidirectional LSTM is a process of constructing a neural network that can store a sequence of information in both forward and backward directions. As mentioned by I. Sunny, S. Maswood, and A.G. Alharbi in [19], it is a modified augmentation of the LSTM model. Bidirectional LSTM improves the execution of the model for sequence classification types of problems. The main working principle of the bidirectional LSTM model is to incorporate two LSTMs in the training process of the sequence of inputs rather than using only one. This type of architecture has been chosen because of its capacity to handle a wide range of real-world situations by leveraging information in both directions. To summarize, bidirectional LSTM adds an additional LSTM layer that reverses the information flow direction. The outputs from both LSTM layers are then combined in a variety of methods, including average, sum, multiplication, or concatenation by default. Another work on the bidirectional mechanism by J. Shah, R. Jain, V. Jolly, and A. Godbole suggests that the sigmoid layer determines what information needs to be conserved and eliminated from the bidirectional LSTM cell. Similar to the LSTM, if the output is “0”, it discards the information, whereas if the output is “1”, the sigmoid function keeps it [26].

2.7 Related Works

Several important works on price prediction, time-series analysis, and obstacles in the current virtual currency pricing market had been gathered and studied. Following that, an attempt to identify current issues, flaws, or limits in prediction approaches was initiated in order to develop a hybrid solution based on the basic LSTM variants. Several researchers approached different solutions to solve the limits and build accuracy.

One such initiative by G. L. Joshila, P. Asha, D. U. Nandini, and G. Kalaiarasi aimed to enhance current cryptocurrency assessments by estimating the cost of a bitcoin while taking into consideration a variety of factors. The elements that impact the price of bitcoin, as well as daily changes in bitcoin financial markets, were discovered after comprehensive research. This project's data was made up of a range of features gleaned via live tracking over the preceding few years. In this work, the SVM approach was employed because it provided a significantly higher precision than earlier methods. The research forecasted price changes in bitcoin for investors so that they might simply invest in it, as well as for novices towards the marketplace as well as the company. Additionally, the suggested model successfully forecasts the price of bitcoin at a specified date [24].

Another work suggested using SVM and LR to forecast daily bitcoin cloning prices. Statistical metrics like MSE, MAE, RMSE, and Pearson correlation were used to assess the effectiveness of the produced model. It was found that the suggested SVM model outperforms the LR model in this research. Evaluating multiple possible models, SVM using linear and polynomial kernel functions, yielded the prediction model with the least error. Bitcoin price prediction was done using filters with varied weight coefficients for different window lengths. In order to build a model with excellent performance, the 10-fold cross-validation approach was utilized in the training phase of the method [12].

In another work, a method by A. Polities, K. Doka, and N. Koziris had been introduced for systematically identifying the most relevant data attributes for a given cryptocurrency, concerning the development of a collection of deep learning models for time series prediction, including LSTM, GRU, TCN, and model clusters. The goal of reasearchers was accurate cryptocurrency price prediction, but the task was extremely difficult and time-consuming because of their high vulnerability and sharp variances than physical cash. As a result, traditional statistical approaches failed to satisfy the complexities of cryptocurrency movements, leading academics to turn to sophisticated data mining algorithms. Consequently, The proposed approach which was applied to the bitcoin application, accurately estimated the precise price of bitcoin as well as its direction, with a precision of close to 84.2 percent [25].

Another piece of literature looked at the ARIMA models in time series data prediction. The ARIMA model had a significant potential for short-term prediction and could compete favorably with existing stock price prediction strategies, according to the findings of the authors. Consequently, the ARIMA model was used in this research to give a detailed procedure of price forecast accuracy. ARIMA models might compete pretty well with developing forecasting approaches in brief prediction based on the findings obtained. The experimental results produced with the best ARIMA model revealing ARIMA models' ability to accurately anticipate stock values on a short-term basis. This might help investors make better-investing choices [4].

Predicting the potential value of digital currencies by acquiring their previous price is a prominent research issue. S. Dong proposed an SVR approach derived from data segment modeling to predict digital currency prices. An ensemble-SVR approach that relied on the SVR method was presented to enhance the accuracy and

feasibility of forecasting the price of digital currencies. The ensemble-SVR method outperformed SVR and other common existing algorithms in forecasting virtual currency prices, according to the simulation results of this research. The problem of significant divergence of the SVR algorithm in forecasting the price volatility trend of crypto money is tackled by commencing with the features of crypto money. The approach was not using all past data to model evenly but rather analyzes the information in sections and models each fragment using the SVR algorithm. When estimating the price, the approach selected the most comparable data item from all infrastructure-based and applies the SVR model to it to predict the cryptocurrency price change legislation. The modeling results suggested that the strategy presented in this study is clearly useful and improves predictive performance greatly [28].

The forecast of the development of bitcoin was accomplished in [27] using a network model which was done before using Deep Boltzmann Machines for evaluating the data for 2019 with various time intervals aiding to determine the cryptocurrency bitcoin's trend and studying the quantity dynamics of its market capitalization. The trained result was validated to currently assessed using MAE and MSE metrics giving an RMSE of 25.87 and an MAE of 14.83, minute-minute fared best. The use of autoencoder prediction models in Bernoulli Restricted Boltzmann Machine models for strong investment cryptocurrencies like Ethereum and XRP might be investigated.

The study mentioned in [21] looked at how well popular sentiment on Twitter can be used to estimate bitcoin profits. The Twitter sentiment was discovered to have predictive value for bitcoin's results using a sentiment analyzer. The findings of this research demonstrated the existence of a link between them. When the authors made forecasts depending on the cryptocurrency tweet mood and the history of bitcoin price, they achieved 62.48 percent right. Using a neural network architecture, they discovered a partial link between the price volatility of bitcoin and the variation of emotion classes. There was a substantial association between both the bitcoin percentage change and Twitter, according to the data.

In [18], an online model was built that was used to estimate the price of prominent cryptocurrencies such as bitcoin, ethereum, and ripple in Turkish Lira. The price estimation of these three cryptocurrencies was carried out on the web using dynamic data using the applicable model over a certain period of time. Price estimation was done using artificial intelligence approaches such as adaptive neural fuzzy inference systems, ANNs, polynomial curve fitting, and LSTM. The goal of this research was to give monthly forecasts to people or organizations interested in cryptocurrencies, as well as to assess the viability of an exemplary model of artificial intelligence.

In addition, various journals showed the use of multiple machine learning approaches used to analyze time series data. For instance, in the article [14] S. S. Ratakonda, and S. Sasi demonstrated an architecture to analyze seasonal trends on a set of multivariate time series data of the stock market price using regression analysis and random first technique. G. Bontempi, S. B. Taieb and Y. L. Borgne in [3], provided an overview of machine learning techniques in time series forecasting by focusing on three aspects: formalizing one-step forecasting problems as supervised learning

tasks, discussing local learning techniques as an effective tool for dealing with temporal data, and the role of the forecasting strategy when moving from one-step to multiple-step forecasting. In [20] R. P. Masini, M. C. Medeiros, and E. F. Mendes examined current advances in ML approaches for forecasting economic and financial time series. In their survey, they primarily focused on supervised learning techniques, in which the system learns a function that mapped an input or explanatory variable to the output or dependent variable using data organized as input-output pairs. G. Napoles, G. V. Houdt, and C. Mosquera discussed multiple implementations of the LSTM algorithm on various time series data including financial datasets. According to their research, LSTM networks can be beneficial in financial time series data prediction since they can overcome the complex features of it like non-linearity, non-stationarity, and so on [7].

Chapter 3

Methodology

3.1 Workflow

The workflow for the research is shown in the following diagram-

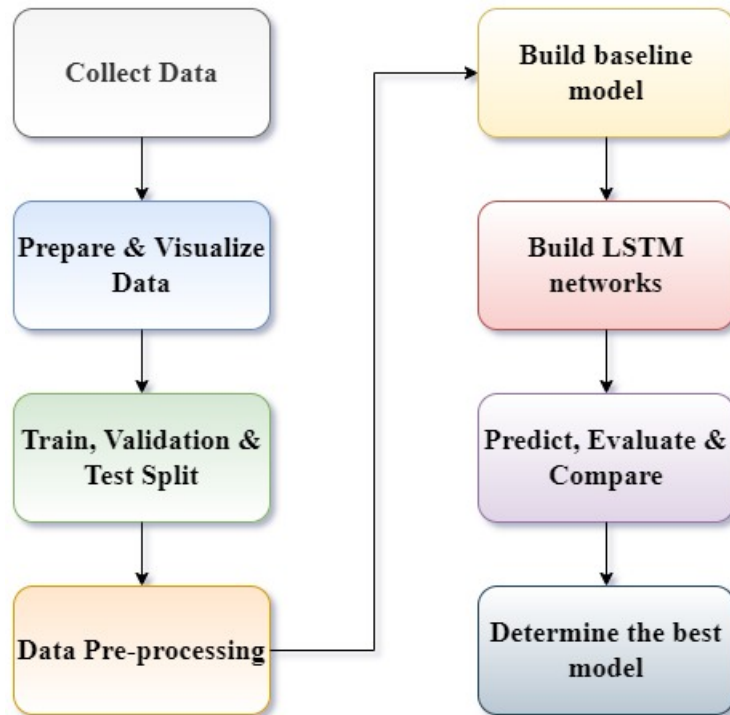


Figure 3.1: Workflow of proposed methodology

According to the workflow, firstly, ether cryptocurrency historical price data was collected. Then, the data was prepared for further usage and visualized to understand the data properties properly. After that, the data was split into the train, validation and test datasets each of which was then pre-processed using suitable pre-processing techniques. Following that, a baseline model was built so that the baseline performance could be compared to the later performance of the LSTM networks to understand whether the LSTM networks really performed well or not. For this research, ARIMA was used for baseline forecasting. Also, three basic LSTM

networks- vanilla, stacked and bidirectional were used for univariate forecasting and a fusional variant- stacked bidirectional was used for multivariate forecasting. Finally, after evaluation and comparison, the best model was determined for ether price forecasting.

3.2 Hybrid Model Architecture

For this research, along with three basic LSTM models, a hybrid LSTM network was also tested on the ether price dataset. the basic LSTM networks were vanilla, stacked and bidirectional LSTM. The following figures (3.2, 3.3) show the internal architecture of vanilla and bidirectional LSTM networks.

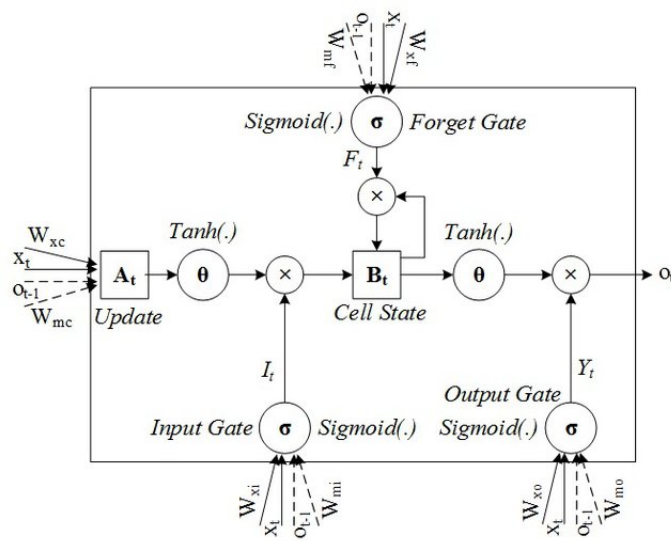


Figure 3.2: A vanilla LSTM unit[16]

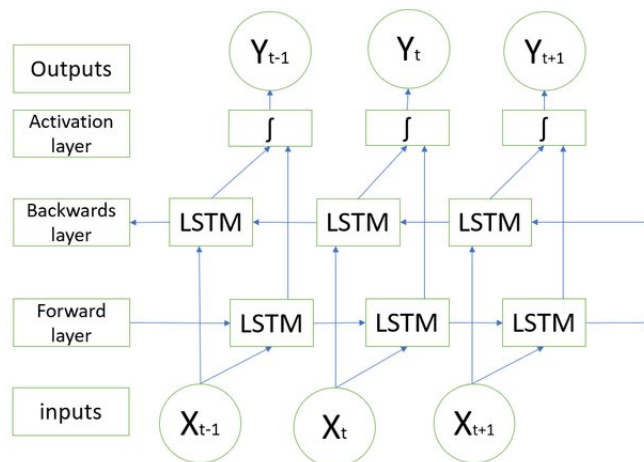


Figure 3.3: Bidirectional LSTM layer[19]

The hybrid LSTM network combined both the bidirectional and vanilla LSTM layers and stacked the latter on the former. The bidirectional layer, as a result, could

be useful to learn price patterns in both forward and backward ways and combine the extracted patterns for better prediction which were then passed to the next unidirectional vanilla LSTM layer. Both the layers were also accompanied by dropout layers so that random outputs from the layers could be made obsolete to prevent the network from being too focused on the training data only, resulting in an increased ability to generalize the learning pattern for all the future prices. The hybrid LSTM box diagram is given in figure 3.4.

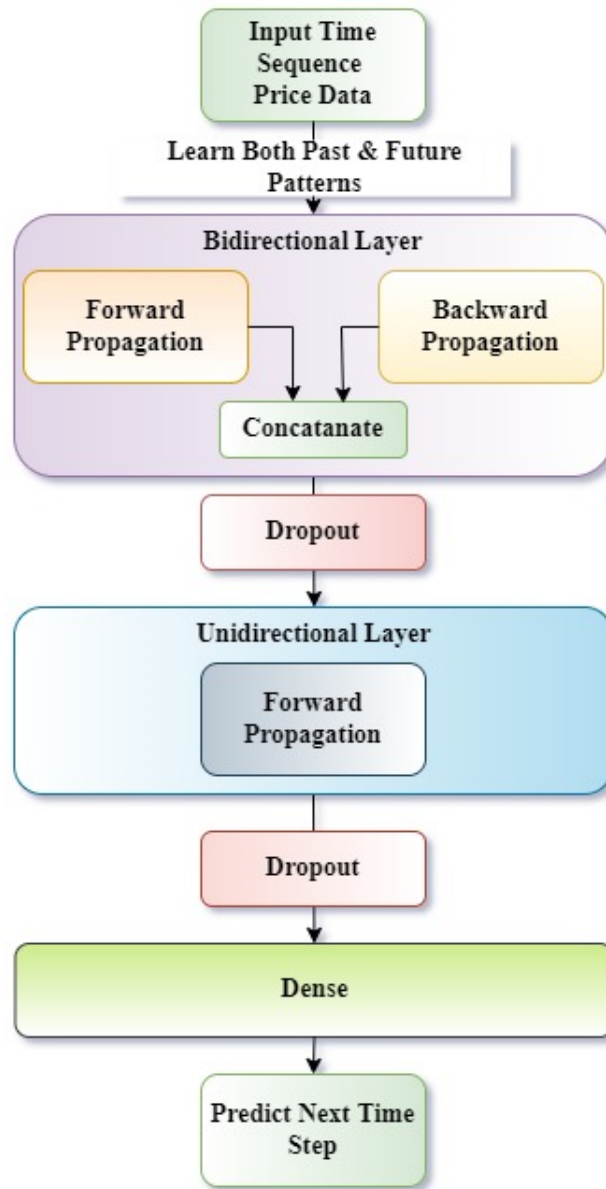


Figure 3.4: Hybrid LSTM architecture

3.3 Data Collection, Preparation, Visualization

Data collection has to be a crucial step of any research work, especially the ones based on machine learning, deep learning, and comparative data analysis. Therefore,

this research was no different. Appropriate data had to be collected and prepared for further usage and also needed to be visualized to understand the underlying patterns and characteristics of the data.

3.3.1 Acquiring Ethereum Historical Price Data

For the proposed research, a dataset that included the opening, closing, high and low price of ether cryptocurrency from March 10, 2016, to May 16, 2022, was collected. In addition, it included the volume and change in percentage of the cryptocurrency on a day-to-day basis. The historical dataset was retrieved from Investing.com which is one of the top global financial websites in the world and offers real-time price data of various cryptocurrencies and stock markets.

3.3.2 Preparing Collected Data

The subsequent step to data collection is data preparation which is used for data curation and data transformation to get a properly prepared dataset from raw data that can be further moved for pre-processing and analysis. Generally, the process of data preparation incorporates various data cleaning, data integration, data transformation, and data reduction techniques. Fortunately, the collected data, in this case, was already in an excellent condition still there was a certain level of ambiguity that needed to be addressed using some data preparation techniques.

Data Types Correction

Initially, the dataset was not in time-series format and all the data points were in string format. Thus, it was converted into time-series data. The string values were converted into their corresponding numerical float values by performing necessary calculations so that they could be easily used for further mathematical procedures. For the numerical conversion purpose, the data was processed through a function so that if there was any value having the letter “K”, that would be multiplied by 1000 and a float value would be returned. The same procedure was followed for the entries with letters “B” and “M” which were then multiplied by 1000000000 and 1000000 respectively. On the other hand, values with “%” were divided by 100. In this way, a dataset with all numerical float values which could be used later for future procedures was obtained. Furthermore, some of the columns were renamed to make the dataset more clarified and understandable.

Time-Series Conversion

Initially, the dataset was not time-series data needed for the time-series analysis. So, it had to be converted to discrete time-series data with daily intervals. For that purpose, all the strings inside the column “Date” were converted to the python data type “datetime64” and the column was made the index of the dataset to get the

time-series dataset. The whole dataset was then sorted by date in ascending order. Additionally, the columns were rearranged in sequence for clarification.

From the following table 3.1, it can be seen that before conversion, the dataset was indexed as “RangeIndex” which is an index of type “int64” representing entries from 0 to 2251, in total 2252 data entries with no null values.

RangeIndex	2252 entries, 0 to 2251	
Column	Non-Null Count	DType
Date	2252	object
Price	2252	object
Open	2252	object
High	2252	object
Low	2252	object
Vol.	2252	object
Change %	2252	object

Table 3.1: Data info before time series

After performing the conversion process, the dataset was transformed into a time-series one where the data points were indexed according to their dates in ascending order which is represented as “DateTimeIndex” in table 3.2.

DateTimeIndex	2252 entries, 2016-03-10 to 2022-05-09	
Column	Non-Null Count	DType
Open	2252	float64
High	2252	float64
Low	2252	float64
Close	2252	float64
Volume	2252	float64
Change	2252	float64

Table 3.2: Data info after time series

Addressing Missing Values

In the dataset, there were several sets of values which were missing (Null) such as volumes from August 03, 2016, to August 06, 2016. To address this, linear interpolation was used to fill those missing values by fitting a straight line along with the existing values and taking fitting values from that line for the missing ones.

Prepared Dataset

Finally, in the curated dataset, the opening price and closing price referred to the daily starting and ending prices respectively. The column “High” referred to the

highest price of the day, in contrast to the column “Low”, the lowest price of that particular day. Additionally, the column “Change” meant the difference between the closing price of two consecutive days and the column “Volume” contained the total amount of ether traded per day. All these preparation steps were done to make the dataset cleaner and easier to understand so that further analysis may become simpler.

Previously before preparation, the dataset had the following sort of values and data types shown in table 3.3:

Date	Price	Open	High	Low	Vol.	Change %
Mar 6, 2022	2,549.40	2,665.42	2,673.19	2,542.19	881.09M	-4.35%
Mar 5, 2022	2,665.42	2,622.15	2,684.50	2,592.07	709.87M	1.65%
Mar 4, 2022	2,622.15	2,834.78	2,835.94	2,575.63	1.50B	-7.50%
Mar 3, 2022	2,834.91	2,947.03	2,967.90	2,889.87	521.36K	-3.81%
Mar 2, 2022	2,947.14	2,975.80	3,041.84	2,914.70	740.37K	-0.96%

Table 3.3: Dataset before preparation

After applying the above mentioned procedure the data became like the Table 3.4:

Date	Open	High	Low	Close	Volume	Change
2022-03-02	2975.8	3041.84	2914.7	2947.14	740370.0	-0.0096
2022-03-03	2947.03	2967.9	2889.87	2834.91	521360.0	-0.0381
2022-03-04	2834.78	2835.94	2575.63	2622.15	1500000000.0	-0.0075”
2022-03-05	2622.15	2684.50	2592.07	2665.42	709870000.0	0.0165
2022-03-06	2665.42	2673.19	2542.19	2549.4	881090000.0	-0.0435

Table 3.4: Dataset after preparation

3.3.3 Data Visualization

After preparation, the dataset needed to be visualized to understand the underlying patterns and characteristics. The following graph (3.5) is a visual representation of the prepared closing price.

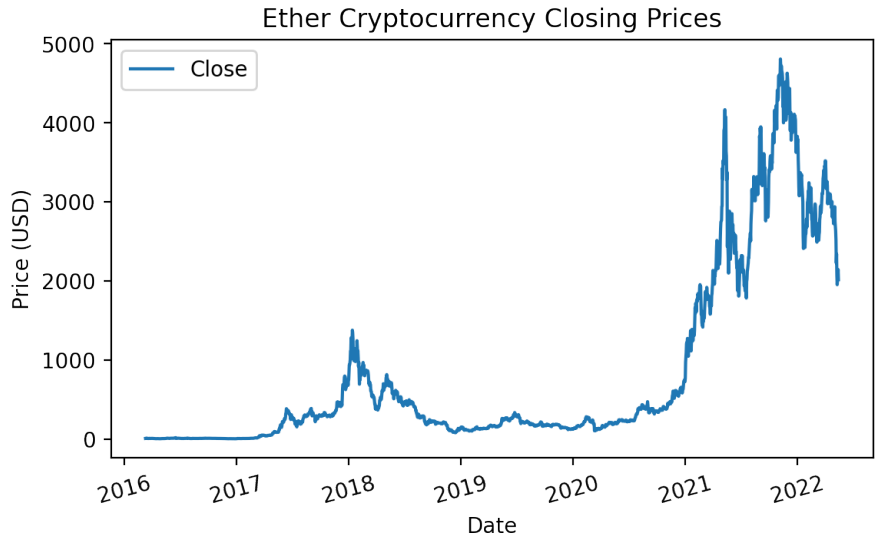


Figure 3.5: Ether closing prices

After plotting the collected dataset, the dataset was decomposed into its components—trend, seasonality, and noise/residue to understand whether the data was stationary or not. To visually understand, the different components were plotted. From the trend and seasonal plots, it was visible that the ether prices had an overall upward trend and yearly seasonality from 2016 to 2022. The presence of a trend among the data points and the observed seasonality was an obvious visual indication that the time-series data might be a non-stationary one. Following are the plots of the trend and seasonality in figure 3.6 and 3.7.

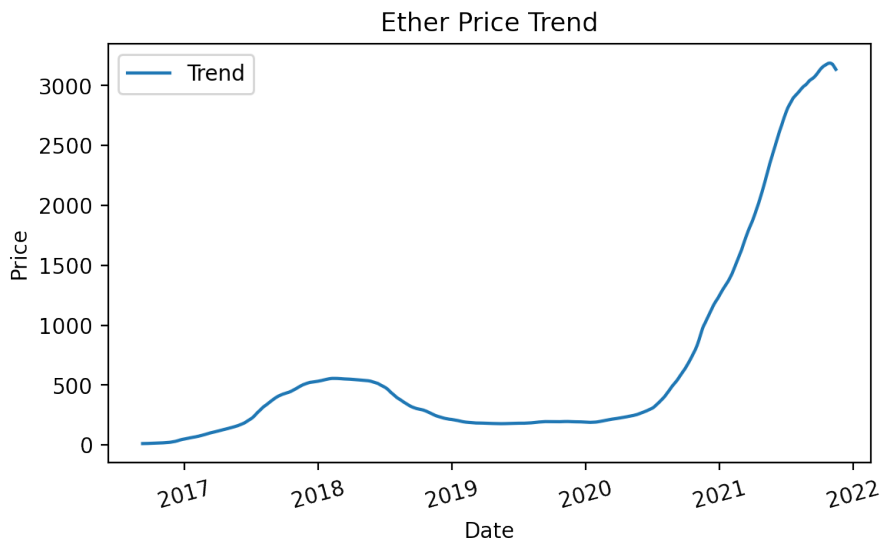


Figure 3.6: Ether price trend

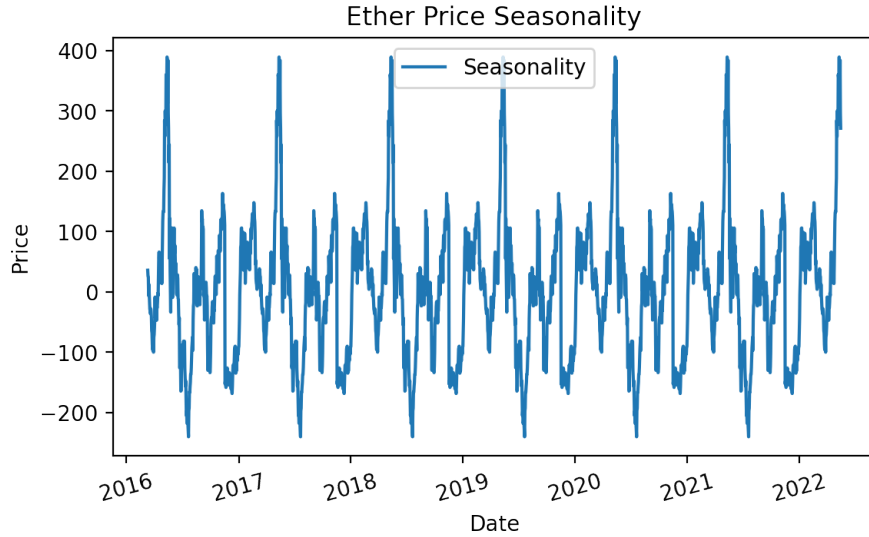


Figure 3.7: Ether price seasonality

However, to be absolutely sure about the non-stationarity, the ADF test was conducted on the dataset which returned the following results in table 3.5.

ADF Test Results	
Test Statistic	-1.18
P-value	0.68
Lags	17.0
Observations	2241.0
Critical Value (1%)	-3.43
Critical Value (5%)	-2.86
Critical Value (10%)	-2.58
Result	Not Stationary

Table 3.5: ADF test of original data

From the results in table 3.5, the p-value is 0.68 which is greater than 0.05 and also the test statistic is greater than the critical value which means the null hypothesis could not be rejected and the data was not stationary.

Due to the abovementioned results, for the baseline ARIMA model, the data needed to be transformed into stationary. However, LSTM networks can, fortunately, work with non-stationary data due to their dynamic and complex architecture. So, the dataset was not needed to be made stationary to use in LSTM networks.

Finally, the successful completion of all the above steps provided a prepared dataset that would be used to perform the tasks described in the upcoming sections.

3.4 Baseline Forecasting Using ARIMA

Before moving on to the LSTM networks, a baseline forecasting method was needed so that the performance of the LSTM networks could be evaluated against the

baseline performance. For this purpose, ARIMA was chosen as it is also a very popular time-series forecasting technique.

3.4.1 Making The Data Stationary

Previously performed ADF test showed that the data was non-stationary. But, to use the data for the ARIMA model, it needed to be made stationary first. To make the data stationary, the first order differencing was performed. After the first order differencing, the following data showed in figure 3.8 was obtained.

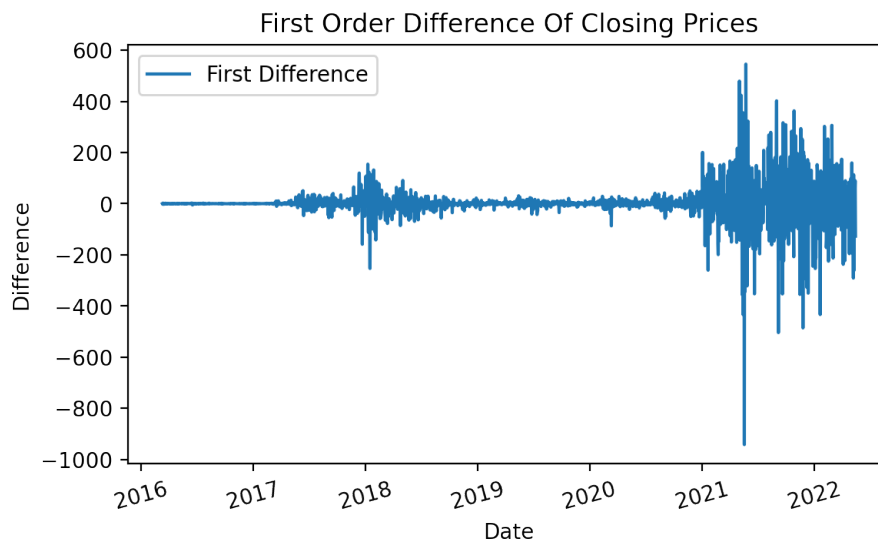


Figure 3.8: First order difference of closing prices

Also, after differencing, the trend was not as clear as the previous data. This could be an indication that the data became somewhat stationary after the first order differencing. The following plot 3.9 shows the trend after the differencing.

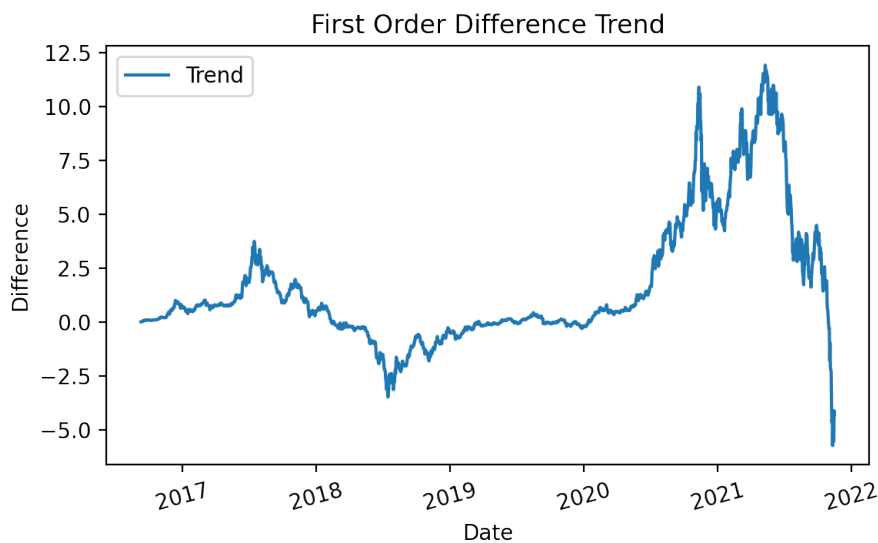


Figure 3.9: First order difference trend

Also, to be sure about the stationarity, the ADF test was performed once again and the test returned the following results.

ADF Test Results	
Test Statistic	-7.85
P-value	5.62e-12
Lags	2.7e+1
Observations	2.23e+3
Critical Value (1%)	-3.43
Critical Value (5%)	-2.86
Critical Value (10%)	-2.57
Result	Stationary

Table 3.6: ADF test after first order difference

This time, the p-value was much lesser than 0.05 and also the test statistic was lower than the critical values. As a result, the null hypothesis could be rejected and the data could be considered stationary.

3.4.2 Train-Test Split

After making the data stationary, the whole dataset was split into the training and testing portions. The split ratio was 80: 20. Following are the training and testing data plots.

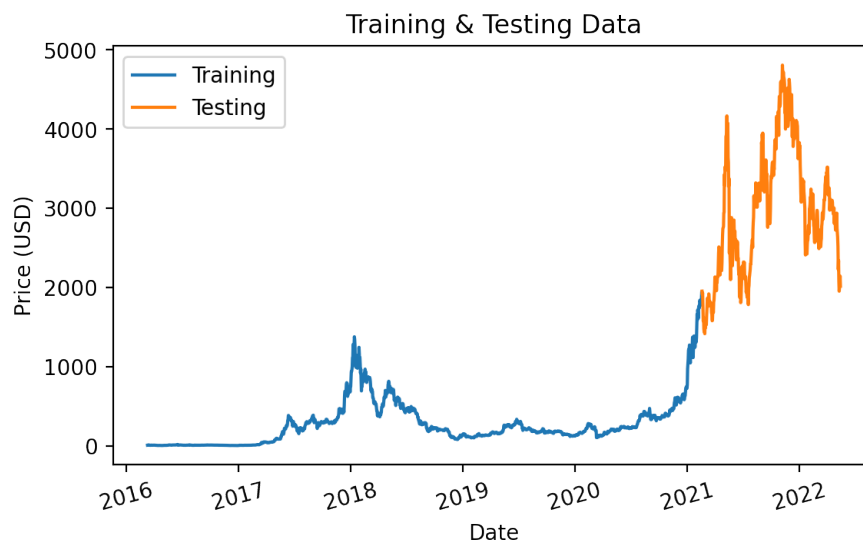


Figure 3.10: Training & testing data

3.4.3 Setting Up The Parameters

After the train-test split of the data, the three main parameters- p, q, and d, for the ARIMA model needed to be determined. As only one differencing was needed to

make the data stationary, the initial value of d could be set to 1. Then, to determine the values of p and q , the PACF and ACF plotting was needed. Below is the PACF plot of the ether first order differenced ether price data.

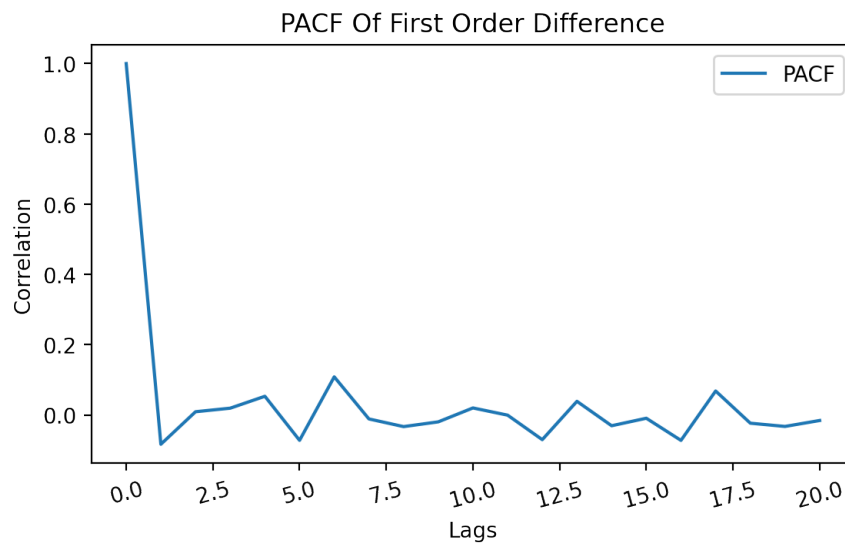


Figure 3.11: PACF of first order difference

From the PACF plot, it was seen that the correlation cut down to zero when the number of lags was 1. So, the initial value for the number of autoregressive terms, p was set to 1.

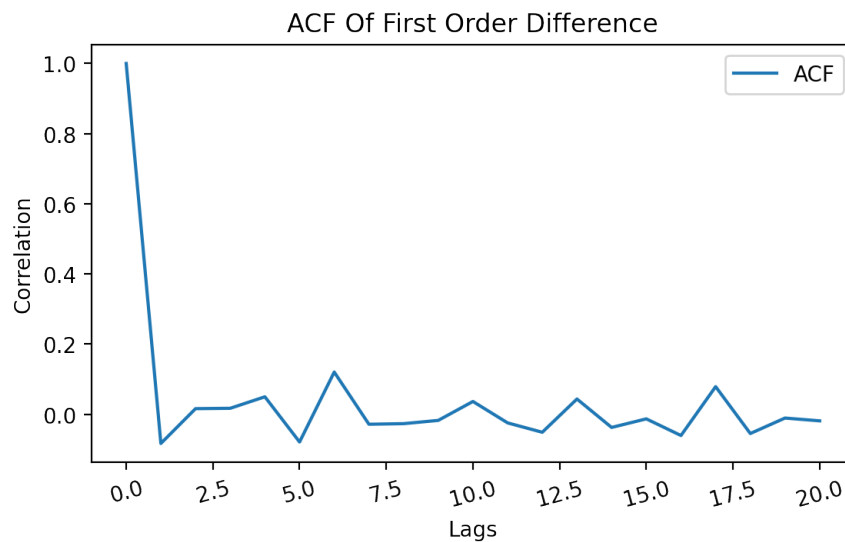


Figure 3.12: ACF of first order difference

Similarly, the ACF plot showed that the correlation cut down to zero for the first time when the number of lags was 1 as well which means the number of lags for the moving average errors, q could be set to 1.

3.4.4 Final ARIMA Model

After setting the values of p, d, and q to 1, 1, 1 respectively, the initial ARIMA model could be built with the order, (1, 1, 1). However, a more appropriate model of order (4, 1, 2) was built afterward by tweaking the parameter on a trial and error basis. The final ARIMA (4, 1, 2) model summary is given in table 3.7 (only p-values).

Term	$P > z $
ar.L1	0.001
ar.L2	0.000
ar.L3	0.034
ar.L4	0.000
ma.L1	0.001
ma.L2	0.000
sigma2	0.000

Table 3.7: ARIMA model summary

Here, it was clear that the p-values for all the terms were very significant as they were all less than 0.05 which meant the corresponding variable X would also be significant. The final ARIMA model was then fitted to the training dataset and was used to forecast future values for a certain number of days in both the short-term and long-term.

3.5 Building LSTM Networks

After baseline forecasting using the ARIMA model was completed, it was time to build various LSTM networks to predict the prices. In this research, both the univariate and multivariate forecasting were performed. For the univariate forecasting, three basic LSTM networks, vanilla, stacked, and bidirectional were used whereas the multivariate forecasting was done using a hybrid stacked bidirectional LSTM network.

3.5.1 Univariate Time-Series Analysis

Initially, univariate time-series analysis was done on the ether price dataset. For this analysis, only a particular feature of the dataset was selected to work on and the future values were predicted by looking at the previous values. In this case, closing prices were analyzed for a particular period and values for the upcoming days were predicted. So, to fit the curated dataset into the LSTM networks, the very first step was to drop all other columns except for the one that would be used for the time-series analysis.

Train, Validation & Test Split

The dataset was split into three sections- training, validation, and testing sets. In this case, 70 percent of the whole data was used as training data, 10 percent was used for validation and the remaining data were used as testing data. So, the final ratio of the train-validation-test split was 70: 10: 20. Figure 3.13 is what the data looked like after the split.

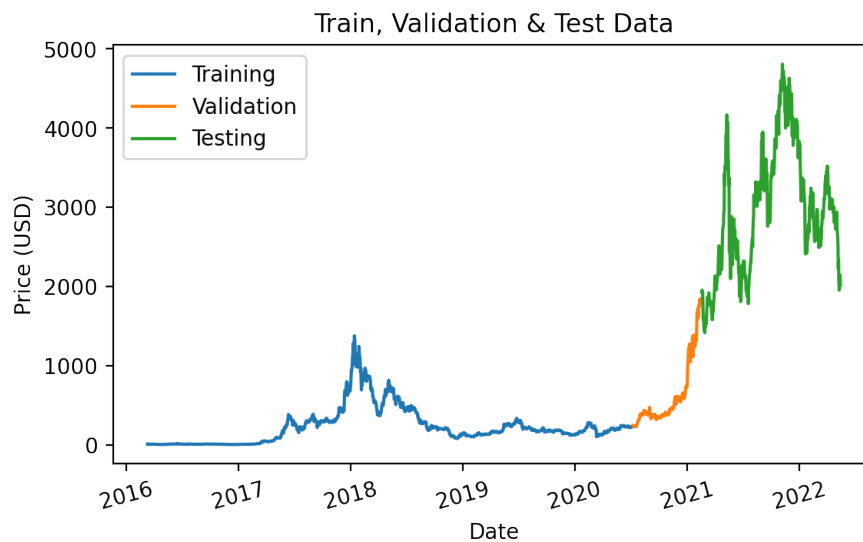


Figure 3.13: Train, validation & test split for LSTM

Normality Testing

After the split, two normality tests were performed on the dataset using the Kolmogorov-Smirnov test and Shapiro-Wilk test to determine if the sample data was collected from a range of data with a normally distributed pattern. While representing graphically, a normally distributed dataset forms a bell-shaped curve which our dataset did not and that was visualized in the graph shown in figure 3.14.

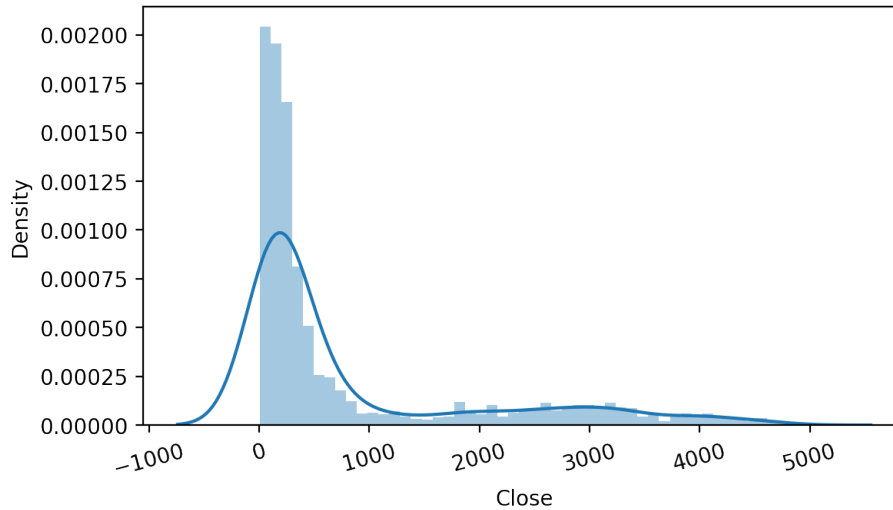


Figure 3.14: Distribution plot

The results of both the normality tests are given in the below tables 3.8 and 3.9.

Kolomogorov-Smirnov Test Results	
Test Statistic	0.99
P-value	0.0
Gaussian	No

Table 3.8: Kolomogorov-Smirnov test results

Shapiro-Wilk Test Results	
Test Statistic	0.68
P-value	0.0
Gaussian	No

Table 3.9: Shapiro-Wilk test results

For the dataset, both the tests provided a p-value of 0.0 which is less than 0.05 so, a conclusion could be reached that the dataset was not normally distributed.

Feature Scaling

According to the normality tests performed, the data did not follow a normal distribution. Although the popular practice in such conditions is min-max scaling, due to the unspecified range of the prices, for this research all the closing prices were standardized instead of min-max scaling. Table 3.10 shows the properties of the training dataset before and after scaling.

Properties	Before	After
Count	1581.00	1581.00
Mean	229.28	-1.35e-17
Standard Deviation	230.20	1.00

Table 3.10: Data properties before and after standardization

As the mean and the standard deviation of the data after scaling was almost 0 and 1 respectively, it would be easier for the deep neural networks to process the data more effectively.

Input Data Preparation

For the next step, the training and testing datasets had to be converted into `train_x`, `train_y`, and `test_x`, `test_y` pairs. For this purpose, a lookback window of 7 days was fixed which was used to determine how many prior prices were going to be inspected for predicting the next price. So, for `train_x` and `test_x`, two arrays were generated in such a way that each sample contained an array consisting of lookback values equal to the lookback period. On the other hand, `train_y` and `test_y` arrays were filled with the immediate next data points to already included lookback values present in `train_x` and `test_x` respectively. After that, the arrays (`train_x` and `test_x`) were reshaped to fit into the LSTM model along with its required parameters as the LSTM network requires 3D Tensor as input, and the initial `train_x` and `test_x` arrays did not meet this requirement. To fix this, both the arrays were reshaped to 3D Tensors where the three dimensions were batch size (total number of samples), lookback period (number of days in time the model will go back for prediction), and number of features which for the univariate time-series analysis, remains one.

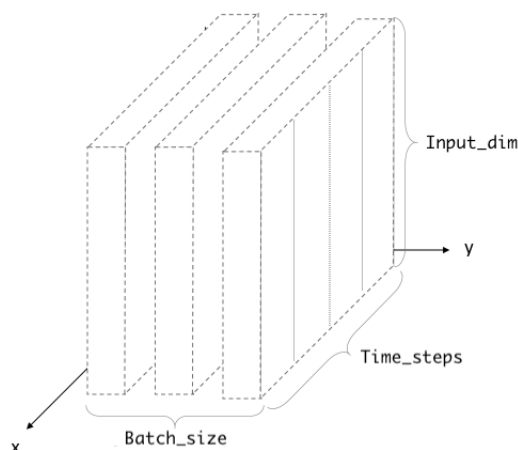


Figure 3.15: LSTM input tensor[20]

Setting The Hyperparameters

To build the basic LSTM networks, several hyperparameters needed to be set and tuned for optimal performance. For this research, all the hyperparameters had similar values for every LSTM network to achieve fairness in the results.

- **Units:** The number of units or neurons in a single LSTM hidden layer was set to 128 for all the networks.
- **Activation Function:** ReLu activation function was used for each model for this research purpose because of its popularity in time-series analysis and robustness against the vanishing gradient problem.
- **Dropout:** Each hidden LSTM layer was accompanied by a dropout layer so that model could have an improved generalization and robustness against overfitting. The dropout ratio was fixed at 0.25 for all the layers.
- **Batch Size:** For all the LSTM networks, the batch size was set to 128.
- **Epochs:** All the models were trained at first for 100 epochs and the number of epochs would be increased if the training and validation losses did not converge.
- **Optimizer:** Optimizer functions mainly determines the learning rate and decay rate. For all the models in this research, the optimizer “Adam” was used.
- **Loss Function:** MSE was used to calculate both the training and validation losses.

Vanilla LSTM Network

At the very beginning, a vanilla LSTM network was implemented with only one single hidden layer for input accompanied by a dropout and one dense layer for output. The hidden layers had 128 units or neurons, the dropout ratio was set to 0.25 and the dense layer helped to achieve a one-dimensional output. While training the model, previously set batch size, optimizer, and loss function were used and the model was trained for 100 epochs.

Figure 3.16 shows the model architecture after implementing the vanilla LSTM model. The output shape of the input and the dense layer was a 2D array that can be defined as (batch_size, unit_size). Since batch size was not defined specifically it was represented as “None” and the unit size illustrated the number of output units. Similarly, for the dense layer. In addition, the number of parameters for LSTM summed to the total number of parameters, 66689. The number of trainable parameters denoted many values could be adjusted according to their gradient. In contrast, the number of non-trainable parameters was 0 means none of the data points was such that could not be optimized during the training procedure.

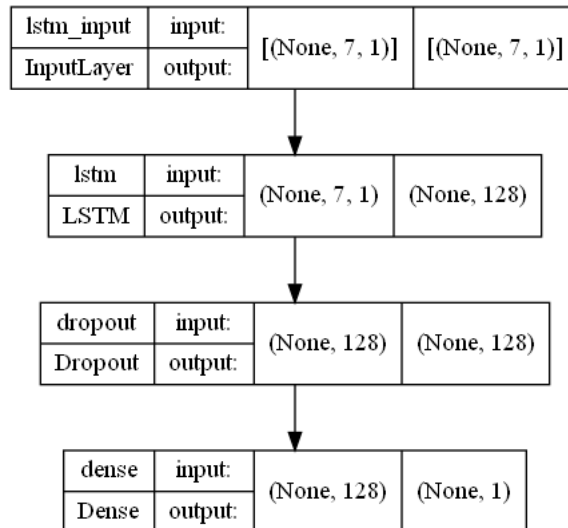


Figure 3.16: Vanilla LSTM architecture

Stacked LSTM Network

A stacked LSTM network was defined to further improve the performance of the previously defined vanilla model. Now, the model had five layers among which the first two and the last one were the same as the vanilla LSTM, being a hidden layer, dropout layer, and a dense layer respectively. The two additional layers, one hidden layer, and one dropout were stacked over the first hidden layer and the accompanied dropout layer and was initialized with 128 units and 0.25 dropout ratio, respectively. Also, an LSTM hidden layer, by default generates an output having two dimensions. But, as the additional hidden layer needed inputs in the format of a 3D Tensor, an additional parameter was activated in the first hidden layer which would then return all the output sequences from the first hidden layer instead of only the last one. The rest of the parameters for the LSTM network were kept identical to the previous vanilla model. The newly defined stacked LSTM model was then ready to fit the training dataset using 100 epochs and predict the test values. While compiling the model, adam and MSE were implemented as the optimization and loss function as usual.

The figure (3.17) shows the model architecture of the stacked LSTM network. In this model, two LSTM hidden layers were implemented where the first layer had a 3D output shape, (batch_size, lookback window, unit_size) and the other two layers' output shapes were the same as the previous. The other elements of the table denote the same things as vanilla LSTM.

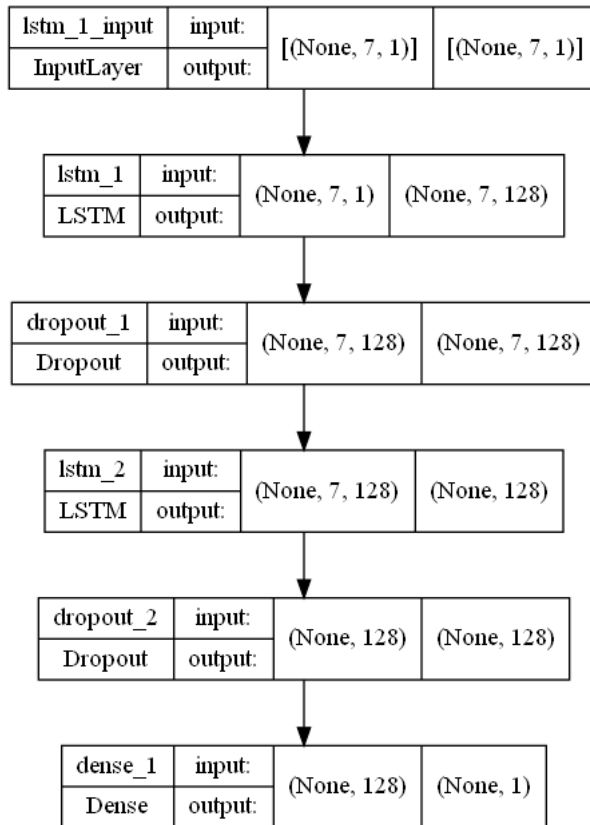


Figure 3.17: Stacked LSTM architecture

Bidirectional LSTM Model

Lastly, a bidirectional LSTM network was also applied to check if a better result could be achieved. The main difference of a bidirectional LSTM in contrast to the other LSTM networks is that it reads the data both ways. Similar to the previously fitted models, batch size was predefined as 128 along with the same optimizer, loss function, and number of epochs. For the bidirectional network, the first layer, unlike the vanilla LSTM, was a bidirectional layer and all the other layers such as the dropout and the dense layer were defined in exactly the same way.

For the Bidirectional LSTM network, the output shape of the only hidden layer was twice the number of the vanilla LSTM as the layer itself was bidirectional. The model architecture is shown in figure 3.18.

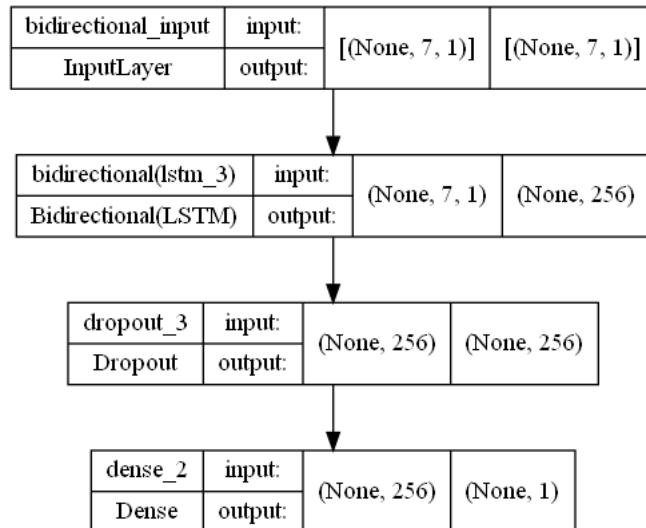


Figure 3.18: Bidirectional LSTM architecture

3.5.2 Multivariate Time-Series Analysis

After univariate time-series analysis, multivariate analysis was also performed on the ether time-series dataset. The distinction between univariate and multivariate analysis is that univariate analysis is based on a single characteristic, whereas multivariate analysis is based on multiple features. For the multivariate price forecasting, a stacked bidirectional LSTM network, which is a hybrid of previously mentioned basic LSTM networks, stacked and bidirectional, was used.

Feature Selection

The very first step of the multivariate time-series analysis of the ether historical dataset was feature selection. To select the main features finding the correlation of the variable “Close” with all other variables in the dataset was necessary as it helped to determine the feature dependencies of the dataset. A higher correlation value indicated that the data features were strongly correlated to each other and thus could be used for multivariate analysis as features. From figure 3.19, it was observed that the “Open”, “Close”, “High” and “Low” features had high correlation values (almost 1) since their type was similar. So, it could be determined that the closing price of any particular day could be predicted by observing the opening, closing, highest, and lowest prices from a specific previous time window. On the other hand, the variables “Volume” and “Change” showed a very low correlation with the other variables so these were not appropriate for predicting the closing price. So, analyzing the correlation of the variables, “Open”, “Close”, “High” and “Low” were selected as features for predicting the closing price.

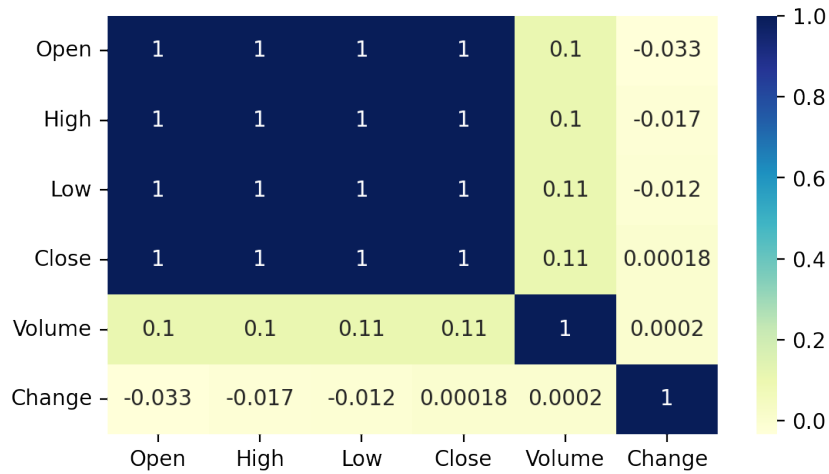


Figure 3.19: Correlation among the features

Data Pre-processing

After feature selection, the dataset was split into training, validation, and testing datasets in the same 70:10:20 ratio as the univariate analysis. Then again standardization was performed to keep the values within a common scale. After that, `train_x`, `train_y` and `test_x`, `test_y` arrays were generated using the same method as the univariate analysis. `train_x` and `test_x`, the 3D input Tensors had the same first two dimensions as the univariate input Tensors, the only exception being the third dimension as the number of features was increased to four which meant for each day of the observation, there were four price values for four features.

Stacked Bidirectional LSTM Network

The stacked bidirectional LSTM network is a combination of the basic bidirectional and stacked LSTM networks which were previously defined in the univariate analysis. Just like the stacked LSTM, two hidden layers were added to the model but unlike the stacked network, the first layer was bidirectional as it was in the bidirectional LSTM. Additionally, dropout layers were implemented after each hidden layer to make the model learn to generalize data and reduce overfitting by adding noise. The dropout ratio was kept exactly the same as in the previous model to ensure fairness during comparison. Additionally, all other hyperparameters like neurons inside hidden layers, batch size, activation function, number of epochs, optimizer and loss function were the exact same as the ones used in all previous basic networks.

Figure 3.20 shows the resultant hybrid model architecture.

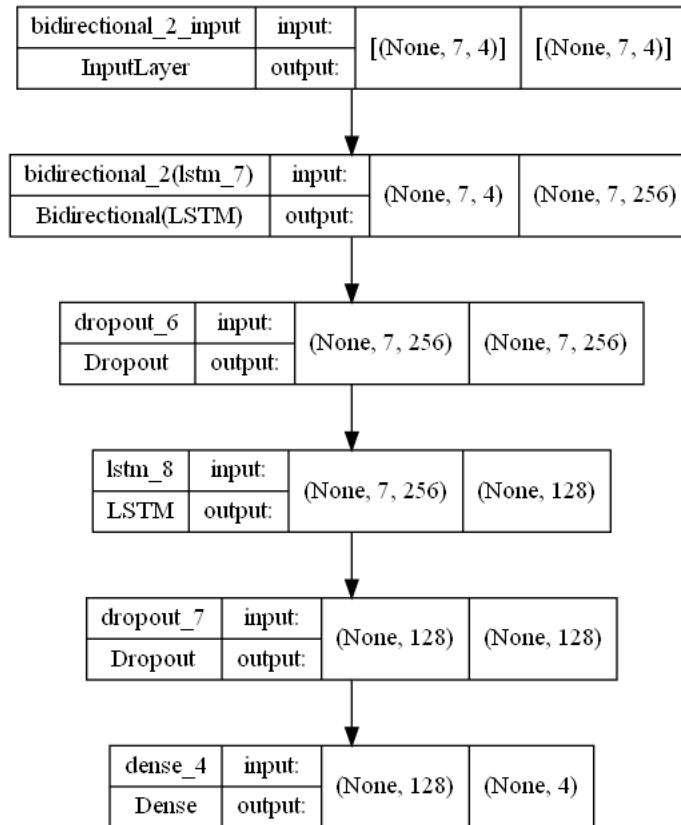


Figure 3.20: Stacked-bidirectional LSTM architecture

3.6 Evaluation Metrics

After building and training all the above-mentioned models, the goal was to make predictions and evaluate the performances for further comparison and analysis. To evaluate the performance the used metrics were mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE) and the R^2 value. Based on these metrics all the models were compared and the best model was determined.

Chapter 4

Result, Analysis & Comparison

4.1 Baseline (ARIMA) Results

The ARIMA model, which was set as the baseline model for this research, could be used to predict a single day's price after being trained with the whole training dataset. The predicted price could then be added to the existing training dataset and another day's price could be predicted after re-training the model with the new training dataset. In such a way, one week's future prices were obtained which are shown in figure 4.1 along with the original prices at that time.

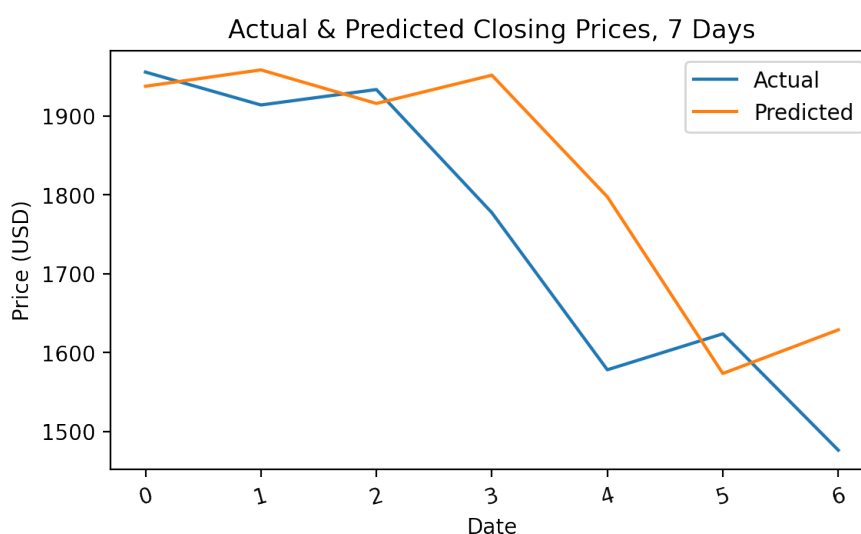


Figure 4.1: Short-term prediction using ARIMA

From the visual perspective, the predictions were not too bad and the general price trend seemed to be somewhat followed by the prediction curve. However, to understand the actual performance, the error metrics would be more useful than only visualization. Table 4.1 shows the performance metrics calculated from the week's actual and predicted daily prices.

Metric	Value
MSE	15275.09
RMSE	123.59
MAE	96.67
MAPE	5.89%
R^2	0.52

Table 4.1: Performance metrics of ARIMA(short-term)

The MSE and RMSE values here could be ignored and considered insignificant as the variance among the prices was very large due to a larger price range. Among the rest, the MAE and MAPE metrics showed promising results, especially the latter only had an error of 5.89%. Although the R^2 value was 0.52 which indicated a weak correlation between the actual and predicted prices.

On the other hand, a trained ARIMA model, while predicting prices for a long period, in this case, 90 days, showed a very different result.

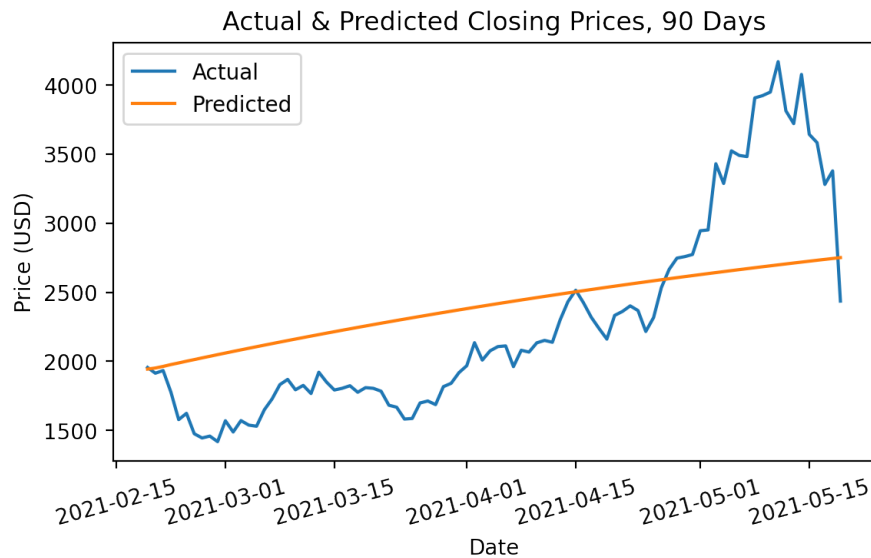


Figure 4.2: Long-term prediction using ARIMA

The results in the plots above which showed the 90-days prediction of the ether price although could follow the overall trend throughout the entire period, could not follow the fluctuations, which in the actual prices curve, were denoted as the upward and downward spikes. The calculated performance metrics are given below for the long-term forecasting using ARIMA.

Metric	Value
MSE	315033.84
RMSE	561.28
MAE	471.28
MAPE	21.16%
R^2	0.41

Table 4.2: Performance metrics of ARIMA(long-term)

The MAPE value for the long-term forecasting was dramatically increased in the long-term to 21.16% and also the value of R^2 further decreased to 0.41 indicating the weakening of the correlation between the actual and predicted values.

4.2 LSTM Results

4.2.1 Results of Univariate Analysis

For the univariate analysis, all of the three basic networks, vanilla, stacked and bidirectional could see their training and validation losses converge to almost zero after training for 100 epochs. So, no additional epochs were necessary. The training and validation losses for all three models are given in figures 4.3, 4.4 and 4.5.

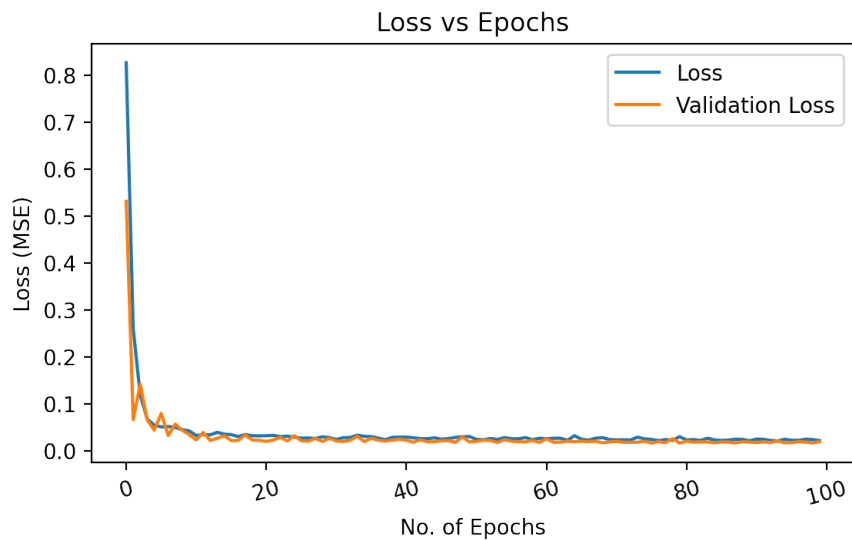


Figure 4.3: Training & validation losses of vanilla LSTM

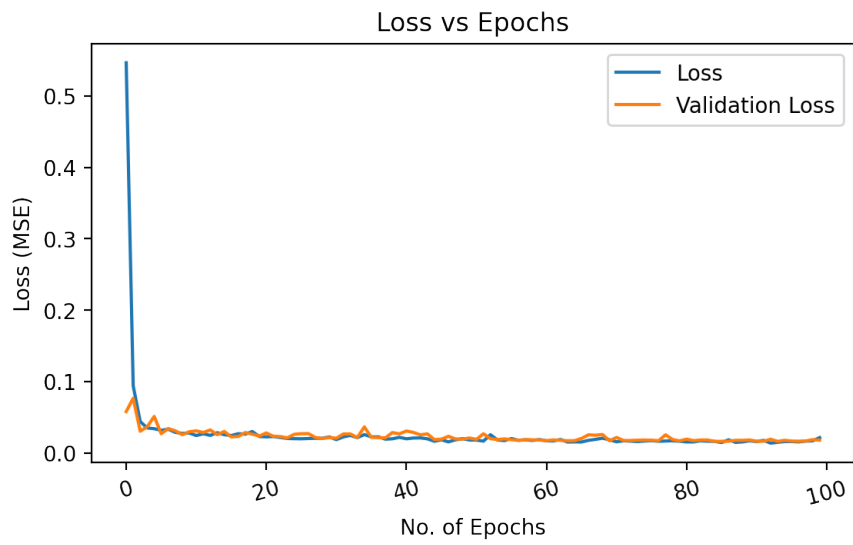


Figure 4.4: Training & validation losses of stacked LSTM

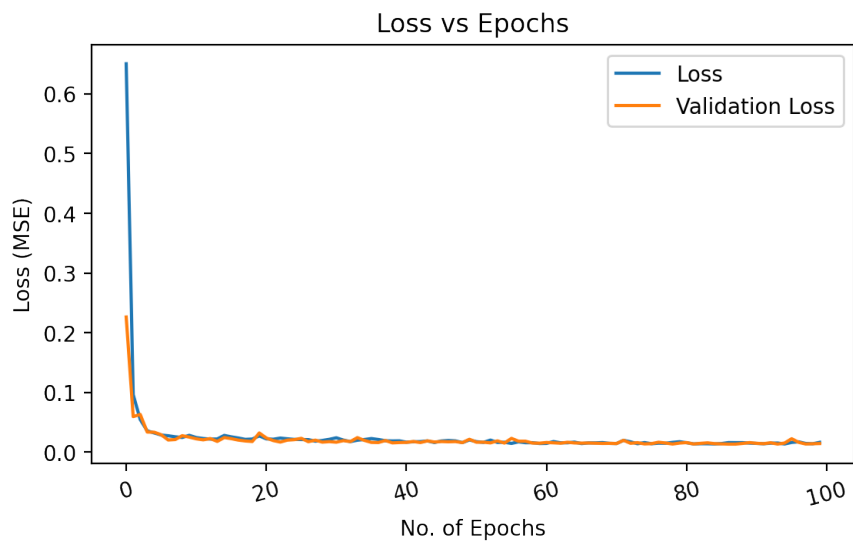


Figure 4.5: Training & validation losses of bidirectional LSTM

After the training was complete, closing prices for the next 90 days were predicted using all three models. The comparative plots of the actual and predicted prices for the vanilla, stacked and bidirectional LSTM networks are shown in the following figures.

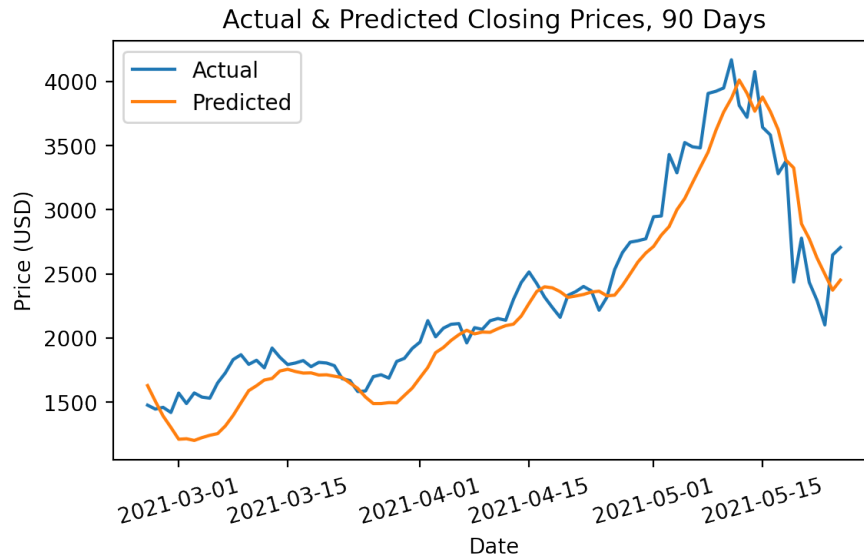


Figure 4.6: Prediction using vanilla LSTM

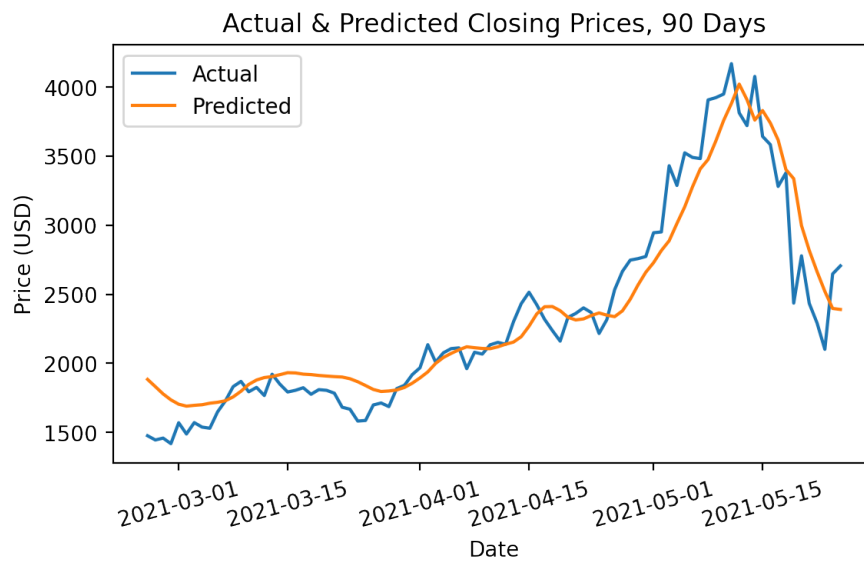


Figure 4.7: Prediction using stacked LSTM

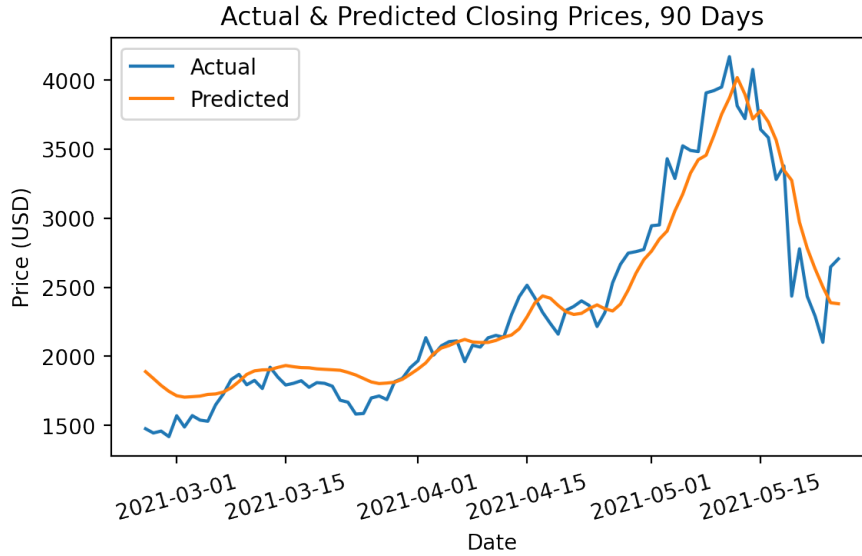


Figure 4.8: Prediction using bidirectional LSTM

Visually, the comparative plots of the actual and predicted prices for all the models looked good and promising. However, the actual performance comparison among the three basic models: vanilla, stacked, and bidirectional LSTM would be possible after looking at their performance metrics shown in Table 4.3.

Metric	Vanilla	Stacked	Bidirectional
MSE	61261.30	50391.92	57661.40
RMSE	247.51	224.48	240.13
MAE	200.06	172.83	196.00
MAPE	8.95%	7.66%	8.93%
R^2	0.88	0.90	0.89

Table 4.3: Performance metrics of basic LSTM networks

From the table, it was clear that the metrics indicated the performances of three models to be pretty much identical. However, the stacked LSTM network still stood out among the three with the lowest error values in all the calculated metrics. So, this model could be used as the bar to beat for the final hybrid LSTM network.

4.2.2 Results of Multivariate Analysis

For the multivariate analysis, a hybrid stacked bidirectional LSTM network was defined. In the training phase, just like the basic LSTM networks, the hybrid LSTM network was able to converge the training and the validation losses quickly towards zero within 100 epochs. The final training and validation losses plot is given in figure 4.9.

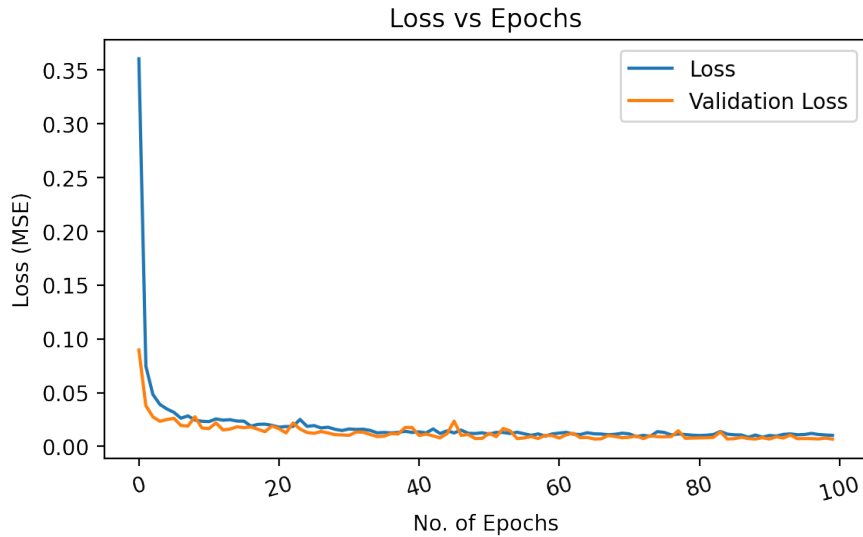


Figure 4.9: Training & validation losses of hybrid model

After the training was complete following 90 days prices were predicted by the hybrid stacked bidirectional LSTM network.

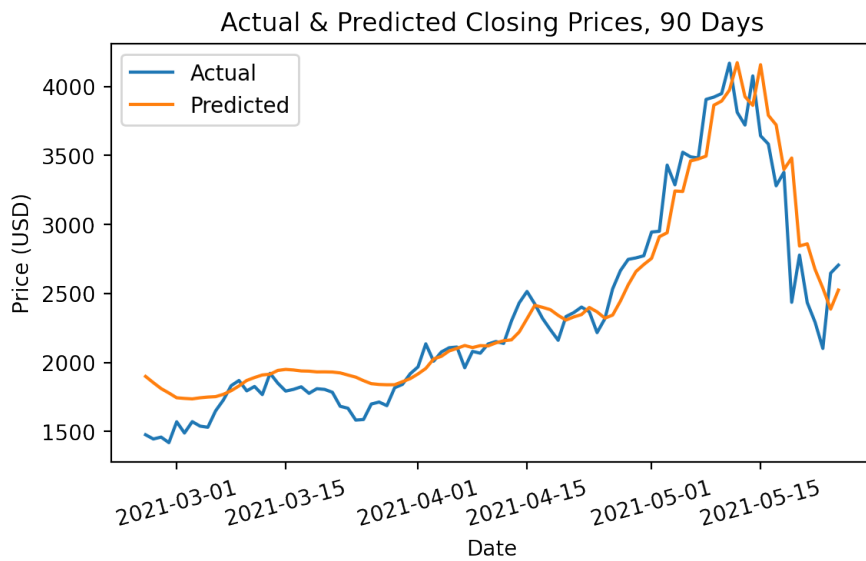


Figure 4.10: Prediction using hybrid LSTM model

From the visualization, there were clearly some improvements detected over the previous predictions from the prior LSTM networks. The prediction followed the original trend of the actual data more precisely and also sudden fluctuations in the price were more accurately detected by the stacked bidirectional LSTM. The improvements could be better explained by comparing the error metrics of the hybrid algorithm with the univariate model having the best performance previously, stacked LSTM.

Metric	Stacked	Stacked Bidirectional
MSE	50391.92	47882.60
RMSE	224.48	218.82
MAE	172.83	154.24
MAPE	7.66%	6.81%
R^2	0.90	0.92

Table 4.4: Performance metrics comparison of stacked and hybrid LSTM

The above-calculated error metrics clearly indicated that the hybrid stacked bidirectional network had an edge in performance over the basic stacked LSTM network in all the metrics. Especially, the three metrics, MAE, MAPE and R^2 , all values were improved over the stacked LSTM. The MAE was decreased to 154.24, and MAPE was decreased to 6.81%. Additionally, the R^2 value was increased and now 92% of the actual variance in the actual prices could be explained by the predicted data.

4.3 Accuracy Comparison

After training and predicting future ether closing prices using all the models, the model accuracies were calculated using the most significant error metric, MAPE value. Figure 4.11 is the accuracy comparison bar chart of all the models.

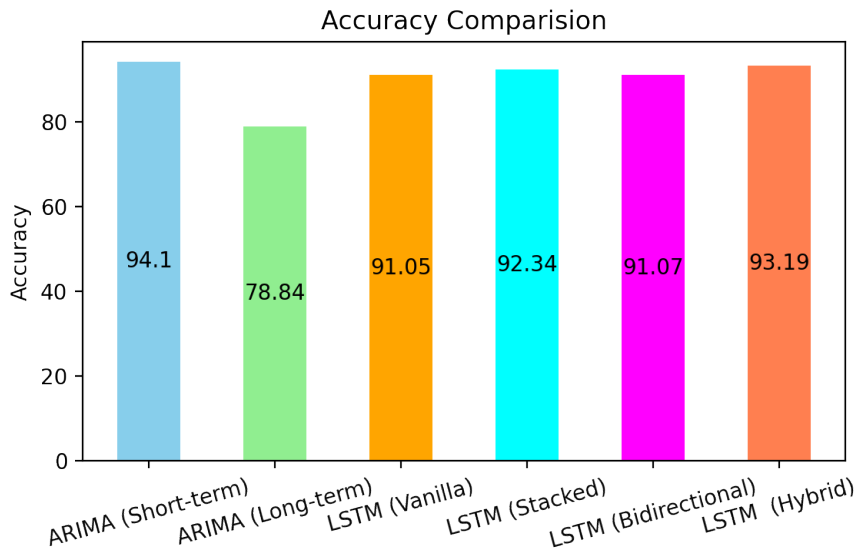


Figure 4.11: Accuracy comparison among the models

4.4 Discussion & Analysis

After acquiring all the results from all the models, looking at all the error metrics and accuracy comparisons, it can be clearly said that all the LSTM networks outperformed the baseline ARIMA model. Although the highest accuracy was achieved

by the ARIMA model in short-term prediction, which is 94.1%, the time and computational power needed for generating and retraining the whole model with an increasing dataset each time just to predict the price of a single day would be huge which makes this approach impractical for the long run. On the other hand, the long-term prediction accuracy for ARIMA, in this research, could climb up to only 78.84%, which compared to all the other models, was very poor. The reason behind low accuracy might be the inability of the ARIMA model to comprehend the seasonal and other irregular patterns which are very crucial to estimating the change and fluctuations of any time-series data. In contrast, LSTM networks are able to learn the complex pattern in the data dynamically. Additionally, the ARIMA model needed the data to be stationary for which the data needed to be differenced. But, differencing data to make it stationary can sometimes be confusing and ambiguous as there persist the problems like under-differencing and over-differencing. Also, the main three parameters for ARIMA, p , d and q are not that straightforward to be determined. Additionally, if the data is seasonal then the ARIMA model needs four additional parameters to determine which can be a challenge. The LSTM networks, on the other hand, can process any kind of data, stationary or not which could be seen from the performance of all the LSTM networks as all of them maintained accuracy of at least 91% in long-term prediction. Finally, the ARIMA model predicts the price using some predetermined parameters which makes it more susceptible to predicting the price in a fixed pattern whereas the LSTM networks can learn and remember previous important patterns by eliminating the less important data from its memory at the same time, making it capable of dynamically adjusting itself to various circumstances.

Also, among all the LSTM networks, the hybrid stacked-bidirectional LSTM network had the best accuracy of 93.19%. The other three basic networks were very good as well, as their accuracy was almost as high as the hybrid one. However, the marginal gap between the performances could be proven very crucial if, in near future, ether cryptocurrency sees a significant change in its price pattern or the crypto market becomes unstable. In that case, the basic networks due to their simple architecture could miss out on important patterns and information whereas the hybrid LSTM networks with complex architecture would be able to learn such irregular patterns more easily.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

This study sought to assure safe investment for ether market investors by evaluating time-series data using statistical models like ARIMA and RNN-based models like LSTM, as well as its modified hybrids, and to perform a comparative analysis to select the best one. The purpose of this research was to incorporate the predictive power of deep learning algorithms to develop a quality prototype capable of detecting future price movements along with predicting actual prices, allowing consumers to get benefited by understanding the market dynamics. Primarily ARIMA model was implemented and it was found that it worked better on short-term prediction only. Later, different variants of LSTM were applied where it marginally worked better when the hybrid stacked bidirectional model was fitted into the dataset. After performing the above mentioned applications, the research found that LSTM surpassed ARIMA in long-term time-series forecasting. Finally, the created methodologies were compared to relevant parameters for further performance evaluation.

5.2 Future Work

According to the proposed model of this research, an LSTM-based solution for predicting ether price was introduced but there is more scope to work on this model in order to look for better performance in the future. To start with, the dataset used for the research includes only daily data on ether price whereas hourly or minutely data could also be used but it was not possible due to the insufficiency of such data. In addition, if an increase in the volume of the ether dataset causes an unwanted and rapid fluctuation among the data points, it will become necessary to evaluate how well the proposed model can detect the random change and in that case, there might be some scopes for further hybridization which in this research was explored in a very limited manner. Next, it was yet to be explored how tuning hyperparameters like activation function, number of hidden units, and batch size affect the performance of the proposed LSTM model. Also, the proposed model could be applied to other cryptocurrency datasets to evaluate its overall performance. Moreover, it was found that the proposed LSTM model outperforms ARIMA but other extended variants of it like SARIMA and SARIMAX are yet to be applied to the collected dataset for checking whether they can produce a better result in handling seasonality and some more exogenous factors affecting the price. Besides, other advanced algorithms

like Prophet, N-Beats, and Temporal Fusion Transformer can be implemented with some modifications on a bigger dataset to make a comparative analysis of their performances. Lastly, more fusion of the LSTM networks can be introduced to make a more efficient model for future studies. All these activities can be beneficial for expanding the solution of the current study to help other market investors boost their investment and make rapid development in this sector.

Bibliography

- [1] E. Parzen, “An approach to time series analysis,” *The Annals of Mathematical Statistics*, vol. 32, no. 4, pp. 951–989, 1961.
- [2] A. Chuang, *Time series analysis: Univariate and multivariate methods*, 1991.
- [3] G. Bontempi, S. Ben Taieb, and Y.-A. Le Borgne, “Machine learning strategies for time series forecasting,” in Jan. 2013, vol. 138, ISBN: 978-3-642-36317-7. DOI: 10.1007/978-3-642-36318-4_3.
- [4] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, “Stock price prediction using the arima model,” in *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, 2014, pp. 106–112. DOI: 10.1109/UKSim.2014.67.
- [5] R. Böhme, N. Christin, B. Edelman, and T. Moore, “Bitcoin: Economics, technology, and governance,” *Journal of Economic Perspectives*, vol. 29, no. 2, pp. 213–38, May 2015. DOI: 10.1257/jep.29.2.213. [Online]. Available: <https://www.aeaweb.org/articles?id=10.1257/jep.29.2.213>.
- [6] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [7] Z. Lipton, “A critical review of recurrent neural networks for sequence learning,” May 2015.
- [8] C. Dannen, *Introducing Ethereum and solidity*. Springer, 2017, vol. 1.
- [9] S. Rouhani and R. Deters, “Performance analysis of ethereum transactions in private blockchain,” in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2017, pp. 70–74. DOI: 10.1109/ICSESS.2017.8342866.
- [10] A. M. Antonopoulos and G. Wood, *Mastering ethereum: Implementing digital contracts*, 2018.
- [11] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.
- [12] S. Karasu, A. Altan, Z. Saraç, and R. Hacıoğlu, “Prediction of bitcoin prices with machine learning methods using time series data,” in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, 2018, pp. 1–4. DOI: 10.1109/SIU.2018.8404760.
- [13] R. Madan and P. S. Mangipudi, “Predicting computer network traffic: A time series forecasting approach using dwt, arima and rnn,” in *2018 Eleventh International Conference on Contemporary Computing (IC3)*, 2018, pp. 1–5. DOI: 10.1109/IC3.2018.8530608.

- [14] S. S. Ratakonda and S. Sasi, “Seasonal trend analysis on multi-variate time series data,” in *2018 International Conference on Data Science and Engineering (ICDSE)*, 2018, pp. 1–6. DOI: 10.1109/ICDSE.2018.8527804.
- [15] Preeti, R. Bala, and R. P. Singh, “Financial and non-stationary time series forecasting using lstm recurrent neural network for short and long horizon,” in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2019, pp. 1–7. DOI: 10.1109/ICCCNT45670.2019.8944624.
- [16] S. R. Venna, A. Tavanaei, R. N. Gottumukkala, V. V. Raghavan, A. S. Maida, and S. Nichols, “A novel data-driven model for real-time influenza forecasting,” *IEEE Access*, vol. 7, pp. 7691–7701, 2019. DOI: 10.1109/ACCESS.2018.2888585.
- [17] Y. Wang, S. Zhu, and C. Li, “Research on multistep time series prediction based on lstm,” in *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, 2019, pp. 1155–1159. DOI: 10.1109/EITCE47263.2019.9095044.
- [18] F. Atlan, I. Peñe, and M. Ş. Çeşmeli, “Online price forecasting model using artificial intelligence for cryptocurrencies as bitcoin, ethereum and ripple,” in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, 2020, pp. 1–4. DOI: 10.1109/SIU49456.2020.9302082.
- [19] M. A. Istiake Sunny, M. M. S. Maswood, and A. G. Alharbi, “Deep learning-based stock price prediction using lstm and bi-directional lstm model,” in *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, pp. 87–92. DOI: 10.1109/NILES50944.2020.9257950.
- [20] R. Masini, M. Medeiros, and E. Mendes, *Machine learning advances for time series forecasting*, Dec. 2020.
- [21] O. Sattarov, H. S. Jeon, R. Oh, and J. D. Lee, “Forecasting bitcoin price fluctuation by twitter sentiment analysis,” in *2020 International Conference on Information Science and Communications Technologies (ICISCT)*, 2020, pp. 1–4. DOI: 10.1109/ICISCT50599.2020.9351527.
- [22] Y. Wang and Y. Guo, “Forecasting method of stock market volatility in time series data based on mixed model of arima and xgboost,” *China Communications*, vol. 17, no. 3, pp. 205–221, 2020. DOI: 10.23919/JCC.2020.03.017.
- [23] C.-C. Wei, “Development of stacked long short-term memory neural networks with numerical solutions for wind velocity predictions,” *Advances in Meteorology*, vol. 2020, 2020.
- [24] G. L. Joshila, A. P, D. U. Nandini, and G. Kalaiarasi, “Price prediction of bitcoin,” in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, 2021, pp. 113–116. DOI: 10.1109/ICOEI51242.2021.9452976.
- [25] A. Politis, K. Doka, and N. Koziris, “Ether price prediction using advanced deep learning models,” in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021, pp. 1–3. DOI: 10.1109/ICBC51069.2021.9461061.

- [26] J. Shah, R. Jain, V. Jolly, and A. Godbole, “Stock market prediction using bi-directional lstm,” in *2021 International Conference on Communication information and Computing Technology (ICCICT)*, 2021, pp. 1–5. DOI: 10.1109/ICCICT50803.2021.9510147.
- [27] S. Sridhar and S. Sanagavarapu, “Analysis and prediction of bitcoin price using bernoulli rbm-based deep belief networks,” in *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2021, pp. 1–6. DOI: 10.1109/INISTA52262.2021.9548422.
- [28] S. Dong, “Virtual currency price prediction based on segmented integrated learning,” in *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)*, 2022, pp. 549–552. DOI: 10.1109/ICPECA53709.2022.9719070.