Prediction of Cryptocurrency prices using Machine learning

Minor project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

Kaushik Deka 191378

UNDER THE SUPERVISION OF

Dr.Himanshu Jindal

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, INDIA

TABLE OF CONTENTS

DECLARATION

I hereby declare that this project has been done by me under the supervision of **Dr.Himanshu Jindal, Assistant Professor(SG),** Jaypee University of Information Technology. I also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:
(Dr.Himanshu Jindal)
Assistant Professor(SG)
Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology

Submitted by:
(Kaushik Deka)
(191378)
Computer Science & Engineering Department
Jaypee University of Information Technology

CERTIFICATE

This is to certify that the work which is being presented in the project report titled "**Predicting Cryptocurrency prices using Machine Learning**" in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by Kaushik deka 191378 and Atishya Jain 191511, during the period from January 2022 to May 2022 under the supervision of Dr.Himanshu Jindal, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Kaushik Deka
(191378)

The above statement made is correct to the best of my knowledge.

(Dr. Himanshu Jindal)
Assistant Professor(SG)
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat,

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible for us to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr.Himanshu Jindal,** Assistant Professor(SG) Department of CSE Jaypee University of Information Technology,Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of "**Prediction of Cryptocurrency prices using Machine learning**" to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr.Himanshu Jindal,** Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Kaushik Deka
(191378)

ABSTRACT

The goal of our project is to predict the cryptocurrency prices using Deep learning. This was done in order to get a better understanding of skills that are needed in the field of Machine learning. The popularity of cryptocurrencies has been massive in recent days and due to the recent hype in cryptocurrencies we found it interesting to predict crypto price. Our goal is to predict the cryptocurrency price for 10 days using different machine learning models and to maintain a low value of mean absolute error and root mean absolute error which shows a relatively accurate prediction of price from actual price of crypto on that date.

We are using CNN (Convolutional Neural Networks) and RNN architecture i.e Recurrent Neural Networks which includes LSTM : Long short term memory , Bi-LSTM : Bidirectional long short term memory and GRU: Gated recurrent unit. A convolutional neural network is used to detect and classify objects in an image .In recent times, CNN has also been used for Time series data. They have some unique features such as pooling layers, full connection layers, etc.

Cryptocurrency price is related to various other factors like closing price and volume among other factors ranging from social dominance to government regulation. Our data is sequential(time-serial) therefore rnn architecture will be good for our prediction . LSTM and Bi-LSTM are popular because they can learn how and when to forget and when not to use gates .Using Bi-LSTM we feed the algorithm with original actual data from the beginning to end and from end to beginning. It usually learns faster than the unidirectional approach although it depends on the task. Traditional neural networks can not do this which seems to be a major shortcoming. Recurrent neural networks address this issue .They are networks with loops in them allowing information to persist . GRU is an advancement of standard recurrent neural networks. GRU uses less training parameters and therefore uses less memory and executes faster than LSTM whereas LSTM is more accurate on a larger dataset.

Chapter 01: INTRODUCTION

1.1 Introduction

Cryptocurrency is a virtual or digital currency used in financial systems. It is secured by cryptography that makes it impossible to be counterfeited or double-spent. Furthermore, it is not issued from a central authority or central banks, and it is decentralized virtual currencies that can be converted via cryptography procedures and this make it distinguishable from traditional currencies. The other feature is that it is created by technology called blockchain, which is an extremely complex concept, and aims at storing data that makes it difficult or impossible to alter, hack, or defraud the system. The most prominent cryptocurrency, Bitcoin, was established in 2009 and for more than two years was the sole Blockchain-based cryptocurrency. Today, however, there are over 5000 cryptocurrencies and 6 million active users in the cryptocurrency industry. Because of its intrinsic nature of mixing encryption technology with monetary units, Bitcoin has recently gotten a lot of attention in the disciplines of economics, cryptography, and computer science.

Cryptocurrency market prediction is a prominent branch of financial research and has been studied extensively. There is mixed evidence regarding the predictability and efficiency of financial markets. Cryptocurrency prices are difficult to forecast due to price volatility and dynamism. Famous cryptocurrencies such as Bitcoin and Ethereum are affected and interacted with by external influences such as the news, social media, and small cryptocurrencies that have a limited market share, which are often not taken into account by investors and traders. Due to the strong relationships between cryptocurrencies, the smaller ones have become a source of shocks that can positively or negatively affect other cryptocurrencies. Around the world there are hundreds of cryptocurrencies that clients use. In this project, we focus on three of the most popular ones. As a result, the project aims to achieve this by using deep learning algorithms, which can discover hidden patterns from data, integrate them, and create far more efficient predictions. We will be trying to present a comprehensive study of the various existing schemes to predict the prices of Bitcoin(BTC), Ethereum(ETH), and Litecoin(LTC) cryptocurrencies by using Deep Learning algorithms such as CNN, LSTM, bi-LSTM, and GRU to accurately predict the prices of cryptocurrencies.

Evaluating the proposed models using evaluation matrices such as RMSE and MAE for Bitcoin, Ethereum, and Litecoin and comparing them. The main idea behind these models is to achieve a reliable prediction model that investors can rely on based on historical cryptocurrency prices. It is to be noted that this is just a project to get some experience in deep learning and ML and the predicted results might differ from actual ones. So this project is not to be used for financial advice.

1.2 Objective

The purpose of this project is to use Deep Learning to forecast Cryptocurrency prices. We will use layered Neural Network models with the application of CNN(Convolutional neural network) and various RNN(Recurrent Neural Networks) mechanisms such as GRU(Gated recurrent units), LSTM(Long Short Term Memory) and Bi-LSTM(Bi-directional Long Short Term Memory). We'll also cover Early Stopping and Dropout, two approaches for avoiding overfitting. Finally, the model will be tested and used to anticipate the price of three cryptocurrencies, listed: 1. Bitcoin (BTC) 2. Ethereum(ETH) 3.Litecoin(LTC). We will implement a multivariate analysis of the various models to predict the price for the next ten days or 10 days in future. We will also try to compare the proposed models using evaluation parameters such as Root mean squared error (RMSE) and Mean Absolute Error(MAE) to decide which model will be best for predicting the prices of each coin.

1.3 Motivation

Traditional currency has been a medium of exchange for goods and services ever since humankind has been in existence. A currency in the most specific sense is money, in the form of paper or coins, usually issued by a government and generally accepted at its face value as a method of payment. As the 21st century began, virtual currency became an area of vast interest in the world of business and trading. The idea for cryptocurrency actually began in the late 1980's, the idea was for a currency that could be sent untraceable and in a manner that did not require centralized entities (i.e. Banks). As this is no longer just a concept but has become an integral reality of today’s financial backbone. The popularity of cryptocurrencies has skyrocketed in the last five years, especially during COVID lock-down. Due to its volatility and rapid growth, it has become a popular asset for investors and traders. We found it interesting to dive into the crypto world and try to predict cryptocurrency price. This is a good chance for us to align our machine learning interest along with the cryptocurrency hype. Since we also have a machine learning course in our current college semester, we have decided to implement our understanding of ML concepts on this particular topic. This project will combine two large topics that we are really keen on learning which are Machine learning and Cryptocurrencies. Moreover, it will help us get hands-on experience on various machine learning and deep learning concepts.

1.4 Language used.

PYTHON 3:

Python 3.0 (a.k.a. "Python 3000" or "Py3k") is a new version of the language that is incompatible with the 2.x line of releases. The language is mostly the same, but many details, especially how built-in objects like dictionaries and strings work, have changed considerably, and a lot of deprecated features have finally been removed.

IDE USED

Project Jupyter

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating notebook documents.

A Jupyter Notebook document is a browser-based REPL containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media. Underneath the interface, a notebook is a JSON document, following a versioned schema, usually ending with the ".ipynb" extension. Jupyter notebooks are built upon a number of popular open-source libraries:

1.5 Technical Requirements(Hardware)

CPU: AMD Ryzen 5 3550H or better/ Intel i5 9th gen or better.

GPU: Nvidia Gtx 1650 or better.

RAM: 16 GB(Recommended)

INTERNET CONNECTION: Required.

STORAGE: SSD of size 256 GB to 512 GB or HDD of size 1TB to 2TB

1.6 Deliverables/Outcomes

1. **ipynb file.**
2. **Project document/report.**
3. **Github Repository.**

Chapter 02:

Feasibility Study, Requirements Analysis and Design

2.1 Feasibility Study

Literature Review

Machine learning (ML) is a sort of artificial intelligence that uses historical data to predict the future. Prior research has demonstrated that ML-based models offer several advantages over traditional forecasting models, including the ability to give a result that is nearly or exactly the same as the real result while also improving the accuracy of the result [6]. Neural networks (NN), support vector machines (SVM), and deep learning are examples of machine learning. The authors of [7] used minute-sampled Bitcoin returns over 3 hour periods. A variety of machine learning methods, including ANN (MLP, GRU, and LSTM), SVM, and ridge regression, were used to predict future values based on past samples, which are compared to the real data and the results were found to be highly accurate. The results reveal that the proposed approach accurately forecasts prices, implying that the technology might be used to forecast prices for a range of cryptocurrencies. To forecast Bitcoin values, the authors of [8] use standard support vector machines and linear regression algorithms. For the building of Bitcoin prediction models, this study uses a time series prediction made up of daily Bitcoin closing values.

The price swings of Bitcoin, Ethereum, and Ripple are investigated in [9]. The researchers used powerful artificial intelligence frameworks, such as a fully linked artificial neural network (ANN) and a long short-term memory (LSTM) recurrent neural network, to discover that ANN relies more on long-term history, whereas LSTM relies more on short-term dynamics, implying that LSTM is more efficient at extracting meaningful information from time series data than ANN. The use of neural networks (NN), support vector machines (SVM), and random forest is investigated in [10]. The findings show that sentiment analysis and machine learning may be used to predict cryptocurrency markets (with Twitter data alone being able to predict specific currencies) and that NN beats the other ML models. The LSTM model is utilized in [10] to anticipate and identify techniques for forecasting Bitcoin on the stock market through Yahoo Finance, which may predict a result of over 12,600 USD in the days after the projection. Researchers have concentrated on increasingly inventive models due to the importance of developing a solid and trustworthy strategy for predicting bitcoin prices. Using a multi-linear regression model and analyzing two major capital market cryptocurrencies, BTC and LTC, the study in [11] focused on social aspects, which are increasingly being used for online transactions around the world. The R2 ratings for LTC were 44 percent and 59 percent, respectively, according to the authors of [11]. In ref. [11], two LSTM models were utilized (a normal LSTM model and an Bi LSTM model). This study used an LSTM model to forecast Bitcoin daily prices and developed a forecasting methodology.

Chowdhury et al. (2020) employed numerous machine learning algorithms to predict nine different cryptocurrencies, including ANN, KNN, gradient boosted trees, and an ensemble model that integrates various ML techniques. The ensemble learning model has the lowest prediction error among the offered models. Derbentsev et al. (2021) also employed a Gradient Boosting Machine (GBM) and RF ensemble model. They estimated the Mean Absolute Percentage Error (MAPE) for Bitcoin, Ethereum, and Ripple, three digital currencies. MAPE values for the ensemble model varied between 0.92-2.61 percent, according to the findings[20].

For predicting BTC's price, researchers compared three different models (ARIMA, LSTM, and GRU) in [12]. According to the experimental results in [12], ARIMA had the best performance, with a MAPE of 2.76 percent and an RMSE of 302.53. To anticipate the price of BTC, the study [12] showed two types of prediction models built using Bayesian optimized RNN and LSTM. The study found that LSTM performed better, with a 52 percent accuracy and an RMSE of 8%.

Since prediction is so important in the investment process, many people rely on it to make money. This project focuses on four models that can predict future cryptocurrency prices using machine learning algorithms and deep learning approaches to achieve accurate prediction models with the goal of assisting investors. The important findings by different authors that are related to our project are displayed below in Table 1.

Table 1: A comparison of previous works.

Author's name and Reference	Cryptocurrency	Method	Result
[22] R.Józefowicz , W.Zaremba , I.Sutskever	BTC , LTC	Multi-linear regression model	R2:44% LTC and 59% BTC
[23]R.Zhao , D.Wang, R.Yan, K.Mao, F.Shen, J.Wang	BTC	Logistic regression	66%
		Linear Discriminant analysis	65.3%
[24H.Sebastião ,P.Godinho	BTC	ARIMA	RMSE:302.53
		LSTM	6003.68
		GRU	381.34
[19] J.H.Mohammad , A.Y.Owda	BTC	LSTM	MAPE:1.1234%
			RMSE:410.399
		GRU	MAPE:0.2454%
			RMSE:174.129
		BiLSTM	MAPE: 5.990%
			RMSE:2927.006
[19] J.H.Mohammad , A.Y.Owda	ETH	GRU	MAPE:0.8267%
			RMSE:26.59
		LSTM	MAPE:1.5498%
			RMSE:59.507
		BiLSTM	MAPE :6.85%
			RMSE:321.061
[19] J.H.Mohammad , A.Y.Owda	LTC	GRU	MAPE:0.2116%
			RMSE:0.825
		LSTM	MAPE:0.8474%
			RMSE:3.069
		BiLSTM	MAPE:2.332%
			RMSE:4.307

Table 1 shows a comparison between the proposed model in various research papers and other models in the literature [22,23,24]. The MAPEs values of the proposed model in [19] for GRU predicting LTC represents the best performance compared to all other models as the predicted results are very close to the actual results. Results obtained from [19] show that the GRU performed better when predicting the price of all types of cryptocurrency than the LSTM and the bi-LSTM models.

The Multi-linear regression model [22] seems to work relatively well although the R2 score seems to be worse than other Deep learning models. This might be due to the fact that Linear regression works well with sequential data but Deep learning models , especially RNN, seems to learn the trend of the dataset more accurately. Logistic regression in [23] shows a more accurate prediction than Multi linear regression. Linear Discriminant analysis also shows a similar R2 score. This is due to the use of the sigmoid function and the fact that logistic regression is one of the closest ML models that functions similar to deep learning models.
ARIMA is an acronym that stands for AutoRegressive Integrated Moving Average. It is a class of models that captures a suite of different standard temporal structures in time series data. In this tutorial, you will discover how to develop an ARIMA model for time series forecasting in Python. ARIMA shows the least Root Mean Squared error out of GRU and LSTM [24]. It is also considered one of the best models to predict time series data. GRU also performed really well compared to ARIMA. Performance measures for [19] were conducted to test the accuracy of different models as shown in Tables 1. The results show that GRU outperformed the other algorithms with a MAPE of 0.2454%, 0.8267%, and 0.2116% for BTC, ETH, and LTC, respectively. The RMSE for the GRU model was found to be 174.129, 26.59, and 0.825 for BTC, ETH, and LTC, respectively. Based on these outcomes, the GRU model for the targeted cryptocurrencies can be considered efficient and reliable. This model is considered the best model. However, bi-LSTM represents less accuracy than GRU and LSTM with substantial differences between the actual and the predicted prices for both BTC and ETH. From the survey we can conclude that, ARIMA model usually performs the best. Deep Learning models, especially RNN, work brilliantly with sequential data. They perform better than traditional ML models.

2.1.1 Problem Definition

The problem statement for this project is the prediction of Cryptocurrency price in USD using various Machine Learning/Deep learning models. We will use three most famous crypto-currencies for prediction. The coins are Bitcoin(BTC), Ethereum(ETH) and Litecoin(LTC). We will be using Deep learning models such as CNN(Convolutional Neural Networks) and RNN(Recurrent Neural Networks) mechanisms (LSTM, Bi-LSTM, GRU) to predict each crypto price in 10 days into the future starting from 25th April 2022 to 5th May 2022. We'll also cover Early Stopping and Dropout, two approaches for avoiding overfitting. Finally, We will compare the proposed models using evaluation parameters such as Root mean squared error (RMSE) and Mean Absolute Error(MAE) to decide which CNN and RNN model will be best for predicting the prices of each coin.

Task: To Predict/Forecast cryptocurrency prices using Deep Learning models and to compare the models' performance.

Data: We will be using datasets of three different Cryptocurrencies (BTC , ETH, LTC) taken from https://finance.yahoo.com/[1] dated from 24.04.2017 to 24.4.2022.

Model: CNN(Convolutional Neural Networks) and RNN(Recurrent Neural Networks) models (LSTM, Bi-LSTM, GRU) will be used to predict future price and their accuracy will be compared to find out the best solution for the problem.

2.1.2 Problem Analysis

The value of cryptocurrency is determined by supply and demand, just like anything else that people want [2]. If demand increases faster than supply, the price goes up. Cryptocurrency gains value when demand rises higher than supply. But it is generally considered harder to predict crypto prices than traditional stocks because of the fact that crypto prices largely depend on various other factors too [3]. These factors can range from a country's economic situation to a billionaire's tweet [2]. Due to this, it becomes extremely difficult to create a prediction model that can take into account such large numbers of factors and forecast the price of a particular cryptocurrency. In this particular project, we are taking into account the two most contributing factors namely, closing price and volume. We are not considering other factors such as social dominance, popularity, financial backing, government regulations [2] etc. This will make our model much less accurate but it is relatively difficult to, first, gather all the data for all the factors, and second, fill all those attributes in our model.

Since our data is sequential or more accurately Time series data, which means it is a collection of observations obtained through repeated measurements over time, RNN and CNN models will be most efficient. Recurrent Neural Network (RNN) is a Deep learning algorithm and it is a type of Artificial Neural Network architecture that is specialized for processing sequential data. RNNs are mostly used in the field of Natural Language Processing (NLP) [4]. RNN maintains internal memory, due to this they are very efficient for machine learning problems that involve sequential data. RNNs are also used in time series predictions as well [4]. Convolutional neural networks have their roots in image processing. Recently, the research community has been showing a growing interest in using CNNs for time-series forecasting problems[5]. Due to the importance of prediction in the investment process that many people depend on to earn revenue, this project focuses on three models that can predict future cryptocurrency prices using machine learning algorithms and artificial intelligence approaches to achieve accurate prediction models with the aim of helping investors.

2.1.3 Solution

To meet the goals of this study, we used historical cryptocurrency prices to train four separate models for three different types of cryptocurrency price prediction. Then, in order to assess the effectiveness of the proposed schemes, we compare the accuracy of our proposed model to that of existing models in five stages:

(1) Gathering historical cryptocurrency data;
(2) Data exploration and visualization;
(3) Training four different types of models;
(4) Testing the models; and
(5) Analyzing and comparing the findings

We propose and analyze four types of algorithms—Convolutional Neural Network(CNN) ,long short-term memory (LSTM), gated recurrent unit (GRU), and bidirectional LSTM (bi-LSTM)—for predicting the price of Bitcoin (BTC), Litecoin (LTC), and Ethereum (ETH) based on historical data. It begins with data gathering, followed by the use of data visualization to explain and study the data's behavior and distribution, as well as the link between the cryptocurrencies. The models are then trained using 80-90 percent of the data collected. The dataset covers the period from 24th April 2017 to 24th April 2022.

While for the training data, we have taken the last 1161 days from the beginning of the testing dataset. The testing dataset covers the period of 10 days from 14 April 2022 to 24th April 2022. The models were then tested once they had been trained. The findings were then retrieved and evaluated, and the best model was chosen based on the daily closing price.

2.2 Requirements

2.2.1 Functional Requirements

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. Multiple designs must be processed and results are analyzed and compared with different models to see which model predicts the most accurate result . Models used:-

- LSTM: Long Short Term Memory . It is a Neural Network Model.
- BiLSTM : Bidirectional Long Short TErm Memory . It is a Neural Network Model .
- GRU : Gated Recurrent Unit . It is a Neural Network Model .
- CNN : Convolutional Neural Network.

Libraries used :

- **pandas:** we have used the pandas library for cleaning , exploring , analyzing and manipulating the dataset .
- **sklearn:** It can be used in supervised and unsupervised dataset for machine learning and is used to predict as well as to determine the accuracy of the model.
- **TensorFlow:** It contains pre-trained tensorflow models . TensorFlow can be used to fine tune the learning models. It is most commonly used for Neural Network Models.
- **keras:** It is a Python library that uses tensorflow in the back end. We used it to detect the trend and make the prediction using its library such as model.predict() .
- **matplotlib:** usd to plot the graphs and import pyplot from it .
- **MinMaxScalar:** It rescale the variables into the range [0,1] without changing the shape of the original distribution .

2.2.2 Non-Functional Requirements

Nonfunctional requirements are those that aren't directly related to the system's ability to perform a given function. They define the criteria that may be utilized to assess a system's performance rather than specific actions. They could have anything to do with emergent system features like Reliability, reaction speed, and store occupancy are all factors to consider. The user generates nonfunctional needs. Budget limits, organizational regulations, and the necessity for compatibility with other systems are all factors that must be considered external factors such as: - software and hardware systems or - external elements such as:

- **Product Requirement:** we can use IDE like JupiterNotebook, Google Collaboratory , VScode .
- **Operating System :** Window 8 or above.
- **Basic Operational Requirement :** Dynamic dataset of cryptocurrency prices like yahoo finance API.

2.3 ER Diagram

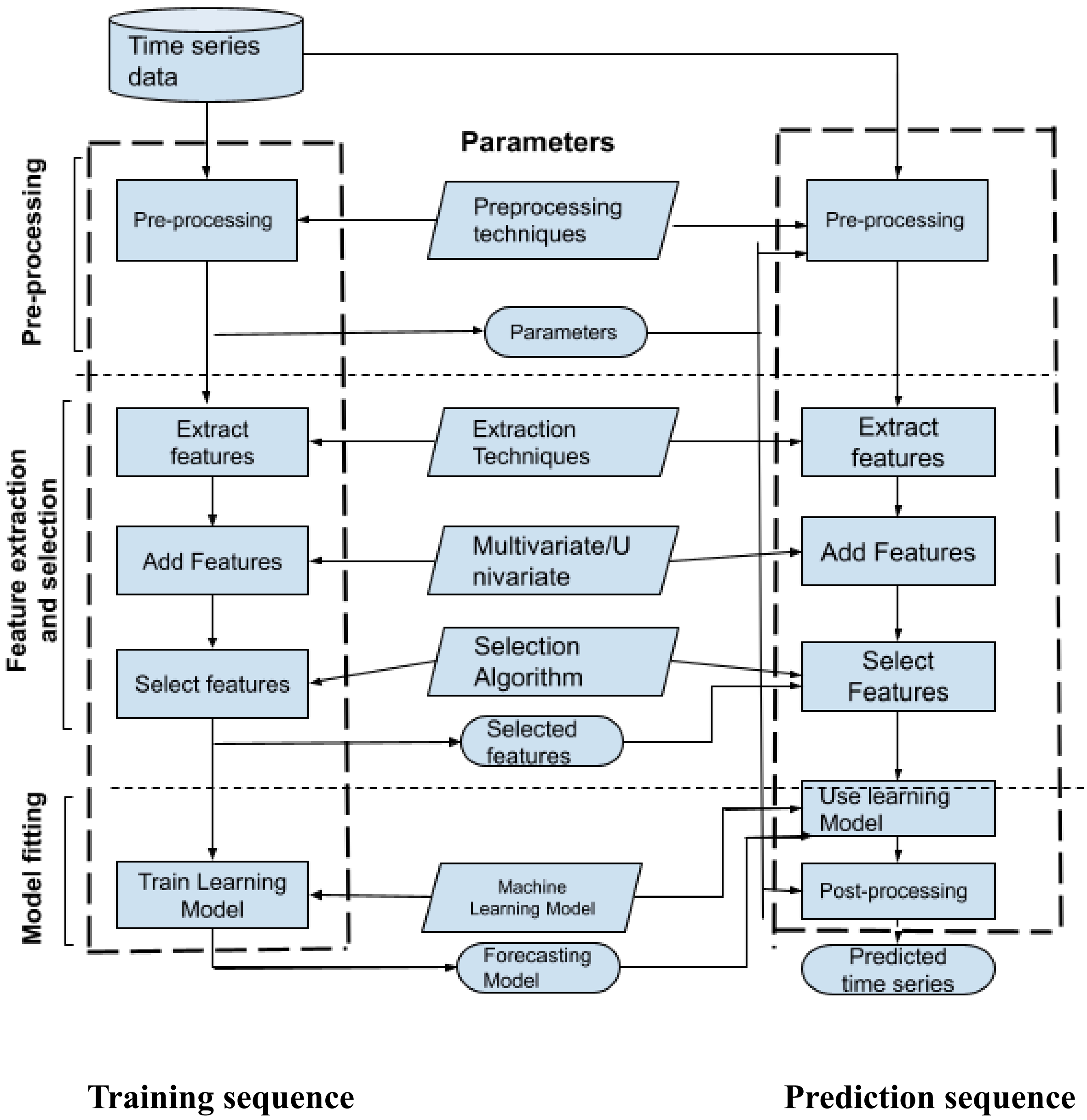

Fig 2.1 ER diagram of the Project Model

Chapter-3(Model Definition and Functions)

3.1 ML models and functions (used for this project)

Machine Learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Deep Learning

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans[18].
A computer model learns to execute categorization tasks directly from images, text, or sound in deep learning. Deep learning models can attain state-of-the-art accuracy, even surpassing human performance in some cases. Models are trained utilizing a huge quantity of labeled data and multilayer neural network topologies[18].

1. CNN

For this project we are using 1 Dimensional CNN.

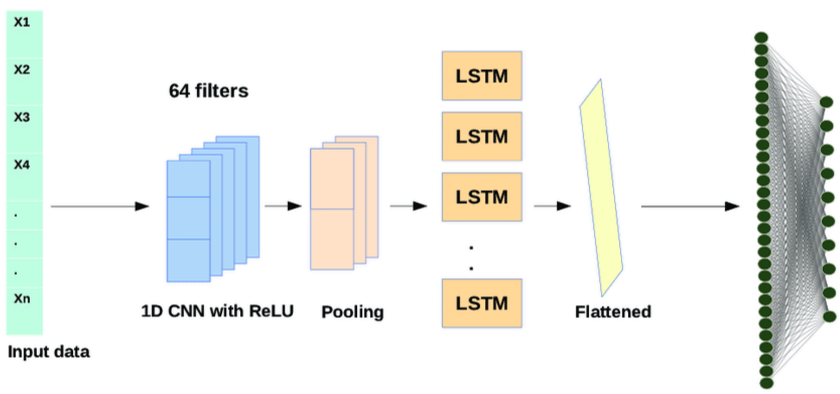

Fig 3.1: CNN-1D and LSTM combination Model

A convolutional neural network is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology. It's also known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image .In recent times, CNN has also been used for Time series data. They have some unique features such as pooling layers, full connection layers, etc. The number of hidden layers in a convolutional neural network is more than that in a traditional neural network, which, to some extent, shows the capability of the neural network. The more the hidden layers are, the higher feature it can extract and recognize from the input[14].
As shown in figure 3.1, In 1D CNN, the kernel moves in 1 direction. Input and output data of 1D CNN is 2 dimensional. Mostly used on Time-Series data [15].
Kernels are filters that are used to extract the features in the data which is necessary to learn the trend in the dataset.

2. RNN

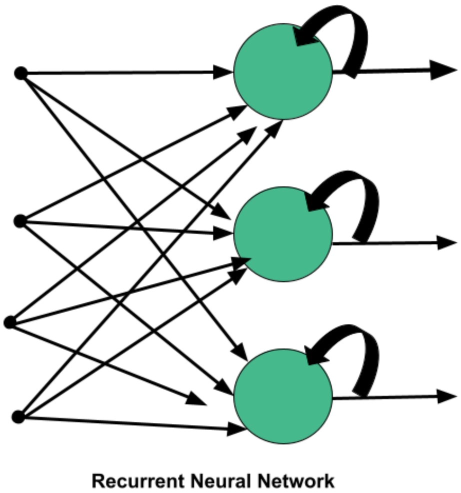

Fig 3.2: RNN-Recurrent Neural Network Model

RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer as shown in Fig 3.2.[16]
The feed-forward neural network had a few flaws, which led to the development of RNN:

1. Cannot handle sequential data
2. Considers only the current input
3. Cannot memorize previous inputs

The RNN is the answer to these problems. An RNN can deal with sequential data by accepting both current and previously received inputs. Because of their internal memory, RNNs can remember past inputs.[16]

RNN Mechanisms:

2.1 Gated Recurrent Units(GRU)

The workflow of **GRU** is the same as **RNN** but the difference is in the operations inside the GRU unit. Let's see the architecture of it.[16]

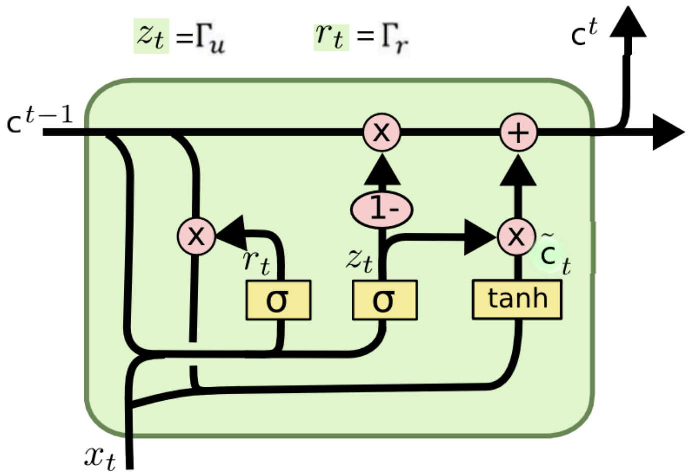

Fig3.3: GRU Model

Inside GRU it has two gates **1) Reset gate 2) Update gate**

Gates are nothing but neural networks, each gate has its own weights and biases(but don't forget that weights and biases for all nodes in one layer are the same)[16].

1. **Reset gate**

The reset gate determines whether or not the preceding cell state is significant. In simple GRU, the reset gate is not always used.

2. **Update gate**

The update gate determines whether or not the candidate state (current activation value) should be updated to the cell state.

3. **Candidate cell**

It's exactly the same as the RNN's hidden state (activation).

4. **The final state**

The update gate determines the final cell state. It may or may not be updated with the current status of candidates. Remove some content from the previous cell state and replace it with fresh content.

2.2 Long Short-Term Memory(LSTM)

LSTMs are pretty much similar to GRU's, they are also intended to solve the vanishing gradient problem.

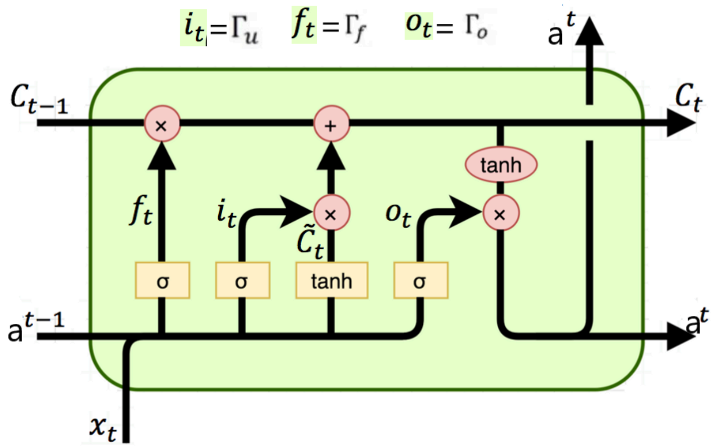

Fig 3.4: LSTM Model

From GRU, we already know about all other operations except forget gate and output gate. Additional to GRU here there are 2 more gates 1)forget gate 2)output gate.[16]

1.Forget Gate

It determines what is remembered and what is forgotten from a previous cell state. In layman's words, it will determine how much data from the previous state should be retained and how much should be deleted.

2. Output Gate

It determines which cell components are output to the hidden state. It will determine what will be the next secret state.

2.3 Bidirectional Long Short-Term Memory(Bi-LSTM)

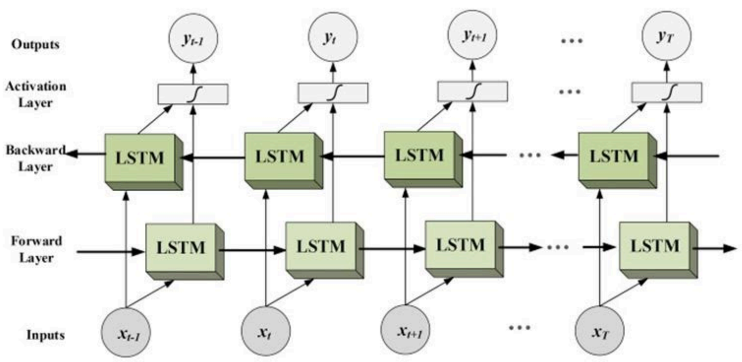

Fig 3.5:BiLSTM Model

Bidirectional long-short term memory (bi-lstm) is the process of allowing any neural network to store sequence information in both backwards (future to past) and forwards (present to future) orientations (past to future).[17]

Our input runs in two directions in a bidirectional LSTM, which distinguishes it from a conventional LSTM. We can make input flow in one way, either backwards or forwards, with a normal LSTM. However, with bi-directional input, we can have the information flow in both directions, preserving both the future and the past.[17]

3.2 Activation Functions used in this project

An Activation Function determines whether or not a neuron is activated. This means that it will use simpler mathematical operations to determine whether the neuron's input to the network is essential or not throughout the prediction phase.

These types of functions were used for this project.

1. Softsign function.

A Softsign Activation Function is a neuron activation function that is based on the mathematical function: f(x)= x/(1+|x|).

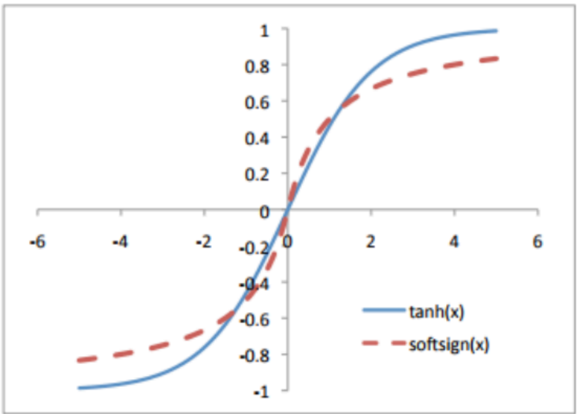

Fig 3.6 : Softsign Function Graph

The Softsign function is an activation function which rescales the values between -1 and 1 by applying a threshold just like a sigmoid function. The advantage, that is, the value of a softsign is zero-centered which helps the next neuron during propagation.

2. Sigmoid Function.

A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve.

$$S(x) = \frac{1}{1 + e^{-x}}$$

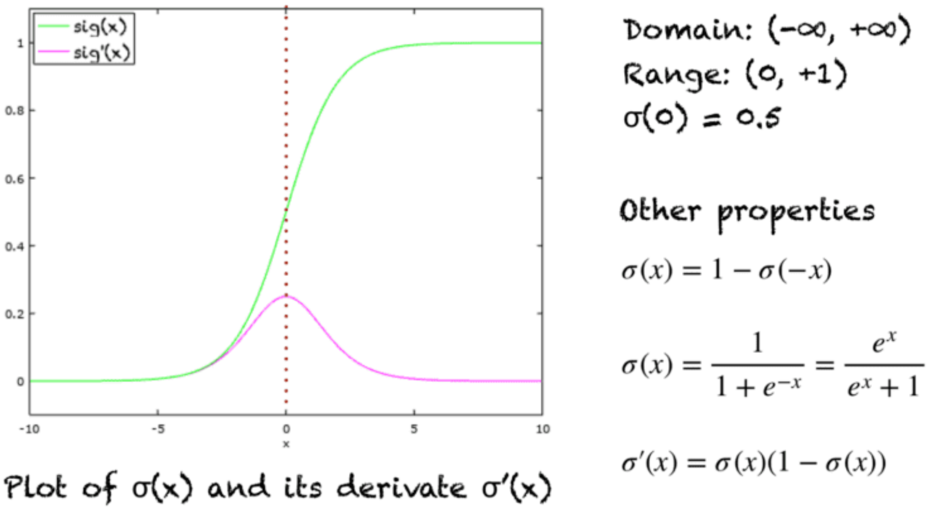

Fig 3.7 : Sigmoid function and its derivative Graph

The graph of sigmoid function is an S-shaped curve as shown by the green line in the graph below. The figure also shows the graph of the derivative in pink color. The expression for the derivative, along with some important properties are shown on the right.

3. Relu

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So it can be written as

f(x)=max(0,x)

Graphically it looks like this (fig 3.8)

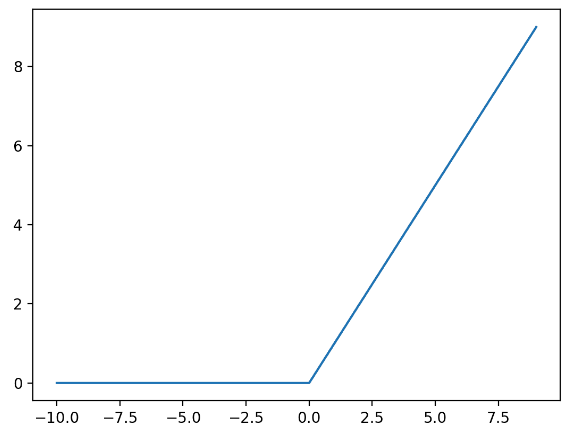

Fig 3.8 : Sigmoid Function Graph

It's surprising that such a simple function (and one composed of two linear pieces) can allow your model to account for non-linearities and interactions so well. But the ReLU function works great in most applications, and it is very widely used as a result.

3.3 OPTIMIZATION FUNCTIONS USED

Adam optimizer

Adaptive Moment Estimation is a technique for optimizing gradient descent algorithms. When working with huge problems with a lot of data or parameters, the method is quite efficient. It is efficient and takes minimal memory. It's essentially a hybrid of the 'gradient descent with momentum' and the 'RMSP' algorithms.

3.4 ERROR MEASUREMENT

Root Mean Squared Error(RMSE)

Mean squared error (MSE) measures the amount of error in statistical models. It assesses the average squared difference between the observed and predicted values. When a model has no error, the MSE equals zero. The equation is given below.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} \left(x_i - \hat{x}_i\right)^2}{N}}$$

Here, Xi is the actual value. Xi<cap> is the predicted value. N is the number of datapoints.

Mean Absolute Error(MAE)

The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures *accuracy* for continuous variables. The equation is given below.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Here, Yi is the actual value. Xi is the predicted value. N is the number of datapoints.

Chapter 04: IMPLEMENTATION

4.1 Date Set Used in the Minor Project

We are using https://finance.yahoo.com/[1] to get the datasets. We have tried to implement the **Yahoo Finance** API directly with the project code which will automatically update the dataset. But instead we have decided to manually download the dataset from finance.yahoo.com due to complications integrating into the program file. The datasets are dated from April 24th 2017 to April 24th 2021(for BTC and LTC), November 9 th 2017 to April 24th 2021(for ETH) and . We are having the datasets for 3 different cryptocurrencies namely **Bitcoin(BTC), Ethereum(ETH) and Litecoin(LTC).**

4.2 Date Set Features

4.2.1 Types of Data Set

BITCOIN

The dataset is dated from April 24th 2017 to April 24th 2021. It contains a total of 1827 entries. Fig 4.1

```
]: dataset = pd.read_csv('BTC-USD.csv')
   dataset
```

]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-04-24	1209.630	1250.940	1209.630	1250.150	1250.150	235806000
1	2017-04-25	1250.450	1267.580	1249.970	1265.490	1265.490	242556000
2	2017-04-26	1265.990	1294.830	1265.930	1281.080	1281.080	329631008
3	2017-04-27	1281.880	1319.700	1281.300	1317.730	1317.730	449196992
4	2017-04-28	1317.740	1331.280	1292.370	1316.480	1316.480	527488992
...	...	...	...	...	...	...	...
1822	2022-04-20	41501.746	42126.301	40961.098	41374.379	41374.379	27819532341
1823	2022-04-21	41371.516	42893.582	40063.828	40527.363	40527.363	35372786395
1824	2022-04-22	40525.863	40777.758	39315.418	39740.320	39740.320	28011716745
1825	2022-04-23	39738.723	39935.859	39352.203	39486.730	39486.730	16138021249
1826	2022-04-24	39503.586	39799.473	39503.586	39755.301	39755.301	16056100864

1827 rows × 7 columns

Fig 4.1: Bitcoin(BTC) dataset.

ETHEREUM

The dataset is dated from November 9th 2017 to April 24th 2021. It contains a total of 1627 entries. Fig 4.2

```
dataset = pd.read_csv('ETH-USD.csv')
dataset
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-11-09	308.644989	329.451996	307.056000	320.884003	320.884003	893249984
1	2017-11-10	320.670990	324.717987	294.541992	299.252991	299.252991	885985984
2	2017-11-11	298.585999	319.453003	298.191986	314.681000	314.681000	842300992
3	2017-11-12	314.690002	319.153015	298.513000	307.907990	307.907990	1613479936
4	2017-11-13	307.024994	328.415009	307.024994	316.716003	316.716003	1041889984
...	...	...	...	...	...	...	...
1623	2022-04-20	3103.935059	3157.885742	3045.288330	3077.745850	3077.745850	15547362265
1624	2022-04-21	3077.829346	3173.451416	2962.410400	2987.480713	2987.480713	20783591093
1625	2022-04-22	2986.938721	3024.854492	2942.358643	2964.835693	2964.835693	16782795477
1626	2022-04-23	2964.802246	2975.322754	2926.740234	2938.114014	2938.114014	9116955609
1627	2022-04-24	2937.347168	2961.882080	2922.128662	2922.732666	2922.732666	9696829579

1628 rows × 7 columns

Fig 4.2: Ethereum(ETH) dataset.

LITECOIN

The dataset is dated from April 24th 2017 to April 24th 2021. It contains a total of 1827 entries. Fig 4.3

```
dataset = pd.read_csv('LTC-USD.csv')
dataset
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-04-24	13.964000	14.965500	13.331300	14.965500	14.965500	60177200
1	2017-04-25	14.978700	15.430200	14.243600	15.212800	15.212800	56668300
2	2017-04-26	15.205900	15.788000	14.555800	14.848100	14.848100	71308800
3	2017-04-27	14.843600	15.003500	13.474500	14.622000	14.622000	76208000
4	2017-04-28	14.593100	14.783800	14.056200	14.341500	14.341500	41067800
...	...	...	...	...	...	...	...
1822	2022-04-20	113.815033	113.921585	111.043251	111.979691	111.979691	669392298
1823	2022-04-21	111.975327	114.220154	106.301636	106.997986	106.997986	747220318
1824	2022-04-22	106.996887	108.344902	105.546104	105.600624	105.600624	668097133
1825	2022-04-23	105.602066	107.133324	105.316483	105.504372	105.504372	502713461
1826	2022-04-24	105.492462	105.930840	104.106628	104.622604	104.622604	487505921

1827 rows × 7 columns

Fig 4.3: Litecoin(LTC) dataset

4.2.2 Number of Attributes, fields, description of the Data set

Attributes:

The datasets contain a total of 6 features. The details for them are as follows:

1. **Close Price** — It is the market close price for currency for that particular day.
2. **High Price** — It is the highest price of currency for the day.
3. **Low Price** — It is the lowest price for currency for that day.
4. **Open Price** — It is the market open price for currency for that day.
5. **Adj Close** —- It is the adjacent market close price for currency for that particular day.
6. **Volume** — The volume of currency that is being traded for that day.

dataset.info() : The **info() method** prints information about the DataFrame.The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values). .info() is used on all three datasets are shown below : (fig 4.4,4.5,4.6)

```
# checking for nulls
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1827 entries, 0 to 1826
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1827 non-null   object
 1   Open       1827 non-null   float64
 2   High       1827 non-null   float64
 3   Low        1827 non-null   float64
 4   Close      1827 non-null   float64
 5   Adj Close  1827 non-null   float64
 6   Volume     1827 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 100.0+ KB
```

Fig 4.4: Checking for NULL values(BTC)

```
# checking for nulls
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1628 entries, 0 to 1627
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1628 non-null   object 
 1   Open       1628 non-null   float64
 2   High       1628 non-null   float64
 3   Low        1628 non-null   float64
 4   Close      1628 non-null   float64
 5   Adj Close  1628 non-null   float64
 6   Volume     1628 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 89.2+ KB
```

Fig 4.5: Checking for NULL values(ETH)

```
# checking for nulls
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1827 entries, 0 to 1826
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1827 non-null   object 
 1   Open       1827 non-null   float64
 2   High       1827 non-null   float64
 3   Low        1827 non-null   float64
 4   Close      1827 non-null   float64
 5   Adj Close  1827 non-null   float64
 6   Volume     1827 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 100.0+ KB
```

Fig 4.6: Checking for NULL values(LTC)

dataset.describe(): The **describe()** method returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains this information for each column: count - The number of not-empty values. mean - The average (mean) value. All three datasets are described below : (fig 4.7,4.8,4.9)

```python
# checking the main parameters
dataset.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	1827.000	1827.000	1827.000	1827.000	1827.000	1827.000
mean	17997.014	18473.591	17473.224	18015.608	18015.608	22728674114.771
std	17661.644	18119.807	17132.462	17659.911	17659.911	20640517689.295
min	1209.630	1250.940	1209.630	1250.150	1250.150	235806000.000
25%	6488.170	6594.295	6396.410	6492.095	6492.095	5561076483.000
50%	9290.960	9450.336	9138.322	9293.521	9293.521	19030650914.000
75%	32643.938	33883.109	31225.665	32742.024	32742.024	33430652172.500
max	67549.734	68789.625	66382.062	67566.828	67566.828	350967941479.000

Fig 4.7: Describing the dataset(BTC)

```python
# checking the main parameters
dataset.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	1628.000000	1628.000000	1628.000000	1628.000000	1628.000000	1.628000e+03
mean	1065.960299	1101.689640	1025.976092	1067.307849	1067.307849	1.251875e+10
std	1254.106917	1293.161477	1209.537850	1254.278923	1254.278923	1.114294e+10
min	84.279694	85.342743	82.829887	84.308296	84.308296	6.217330e+08
25%	197.521683	203.492073	193.221314	197.299900	197.299900	3.296230e+09
50%	391.583801	405.960159	381.292007	393.039139	393.039139	9.745153e+09
75%	1799.694641	1841.851746	1729.146210	1805.556030	1805.556030	1.775240e+10
max	4810.071289	4891.704590	4718.039063	4812.087402	4812.087402	8.448291e+10

Fig 4.8: Describing the dataset(ETH)

```python
# checking the main parameters
dataset.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	1827.000000	1827.000000	1827.000000	1827.000000	1827.000000	1.827000e+03
mean	100.527776	104.599290	96.078930	100.547231	100.547231	2.378725e+09
std	65.666940	69.367026	61.538684	65.588988	65.588988	2.452438e+09
min	13.964000	14.783800	13.331300	14.341500	14.341500	3.869210e+07
25%	49.087545	50.971559	47.493004	49.120861	49.120861	4.767525e+08
50%	75.074013	77.560341	72.763000	75.152588	75.152588	1.818356e+09
75%	140.236534	145.940727	134.633416	140.084297	140.084297	3.261149e+09
max	387.869171	412.960144	345.298828	386.450775	386.450775	1.799426e+10

Fig 4.9: Describing the dataset(LTC)

4.3 Pseudo code of the Project Problem

1. **IMPORT ALL THE REQUIRED LIBRARIES**
    a. Import pandas, numpy, sklearn, min max scalar, keras ,tensorflow etc
    b. Libraries are an important part of Python and data science. They contain the necessary modules, functions and methods.

2. **IMPORT DATASET**
    a. Import all three datasets for each coin
    b. The data sets are obtained from Yahoo Finance.

3. **PREPROCESSING OF DATA**
    a. 1. Putting together all the data and randomizing it.
    b. 2. Cleaning the data to remove unwanted data, missing values, rows, and columns, duplicate values, data type conversion, etc. Restructure the dataset and change the rows and columns or index of rows and columns.
    c. 3. Visualize the data
    d. 4. Scaling the data between 0 and 1.
    e. 5. Splitting the cleaned data into two sets - a training set and a testing set.

4. **CREATION OF MODEL**
    a. We will be using 4 different models.
        - CNN 1d
        - GRU
        - LSTM
        - Bi-LSTM

5. **TRAINING THE MODEL**
    a. Fitting the model on the training set using fit().
    b. The dataset is trained through multiple epochs.

6. **EVALUATING THE MODEL**
    a. Evaluation of the Loss vs Validation loss of each epoch.

b. Evaluation of the testing dataset using Root mean squared Error(RMSE) and Mean absolute error(MAE).

7. POST PROCESSING OF DATA

a. Inverse scaling of the prediction results.
b. It is important to invert the scale for the results to be readable.

8. VISUALIZATION OF RESULTS

a. Visualization of the results of the test set. Predicted vs Actual.
b. Using matplotlib for visualization.

9. PREDICTION

a. prediction of future 10 days starting from 25.04.22 to 05.05.2022
b. Visualization of prediction.

4.6 Screenshots of the various stages of the Project

The First step is to initialize all the necessary variables and parameters for the project like number of past observations(n_past), number of future observations(n_future), activation function, dropouts, layers etc. Fig 4.10 and Fig 4.11

A. RNN (GRU,LSTM,Bi-LSTM)

```
# number of past observations to be considered for the LSTM training and prediction
n_past = 30

# number of future datapoints to predict (if higher than 1, the model switch to Multi-Step)
n_future = 10

# activation function used for the RNN (softsign, relu, sigmoid)
activation = 'softsign'

# dropout for the hidden layers
dropout = 0.2

# number of hidden layers
n_layers = 8

# number of neurons of the hidden layers
n_neurons = 20

# features to be considered for training (if only one is Close, then its Univariate, if more, then it's Multivariate)
features = ['Close', 'Volume']
#features = ['Close']

# number of inputs features (if higher than 1, )
n_features = len(features)

# optimizer (adam, RMSprop)
optimizer='adam'
```

Fig 4.10: Parameter settings for RNN

B. CNN-1D

```
# number of past observations to be considered for the training and prediction
n_past = 30

# number of future datapoints to predict (if higher than 1, the model switch to Multi-Step)
n_future = 10

# activation function used for the RNN (softsign, relu, sigmoid)
activation = 'softsign'

# number of neurons of the hidden layers
n_neurons = 4

# features to be considered for training (if only one is Close, then its Univariate, if more, then it's Multivariate)
features = ['Close', 'Volume']
#features = ['Close']

# number of inputs features (if higher than 1, )
n_features = len(features)

# patience for the early stopping (number of epochs)
patience = 25

# optimizer (adam, RMSprop)
optimizer='adam'
```

Fig 4.11: Parameter settings for CNN-1D

1. IMPORT LIBRARIES

Importing all the required libraries .

In Fig 4.12 we are importing libraries for LSTM Model and following are common in LSTM, GRU , BiLSTM , CNN:

1. **numpy**: is a Python library used for working with arrays.
2. **pandas** : used to analyze data.
3. **datetime** : supplies classes for manipulating date and time.
4. **math**: access to common math functions and constant.
5. **matplotlib** : a cross-platform, data visualization and graphical plotting library.
6. **keras** : developing and evaluating Deep Learning model.
7. **sequential** : linear stack of layer
8. **dense** : regular deeply connected neural network layer.
9. **dropout**: randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.
10. **sklearn** : useful and robust library for machine learning.
11. **MinMaxScaler**: Transform features by scaling each feature to a given range.

LSTM

```
import numpy as np
np.set_printoptions(suppress=True)
import pandas as pd
import datetime
import math
from matplotlib import pyplot as plt
from keras.models import Sequential, load_model
from keras.layers import Dense, LSTM, Dropout
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_score
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

Fig 4.12: Libraries imported for LSTM

For LSTM :-

from keras.layer import LSTM : quite effective in predicting the long sequences of data like sentences and stock prices over a period of time

GRU

```
from keras.models import Sequential, load_model
from keras.layers import Dense, GRU, Dropout
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import MinMaxScaler
```

Fig 4.13: Extra libraries imported for GRU

In Fig 4.13 we are using keras library to import gru ;-

form keras.layer import GRU : GRu implementation in keras .Comprises the reset gate and the update gate instead of the input, output and forget gate of the LSTM.

Bi-LSTM

```
from keras.models import Sequential, load_model
from keras.layers import Dense, LSTM, Dropout, Bidirectional
from keras.callbacks import EarlyStopping
```

Fig 4.14: Extra libraries imported for Bi-LSTM

In Fig 4.14 we are using keras library to import bilstm ;-

form keras.layer import Bidirectional : a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction.

CNN

```
from keras.models import Sequential, load_model
from keras.layers import Dense
from keras.layers.convolutional import Conv1D
from sklearn.preprocessing import MinMaxScaler
```

Fig 4.15: Extra libraries imported for CNN

```
# Building the CNN
from keras.layers.convolutional import MaxPooling1D
from keras.layers import Flatten
```

Fig 4.16: Extra libraries imported for CNN model

In Fig 4.15 and Fig 4.16 we are using keras library to import cnn ;-

- from keras.layer.convolutional import Convo1D : This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs.
- MaxPooling1D: Downsamples the input representation by taking the

maximum value over a spatial window of size pool_size .

- Flatten: fattens the multi-dimensional input tensors into a single dimension,

2. IMPORT DATA SET

In Fig 4.17 , Fig 4.18 , Fig 4.19 Importing all the three datasets.

```
dataset = pd.read_csv('BTC-USD.csv')
dataset
```

```
dataset = pd.read_csv('ETH-USD.csv')
dataset
```

```
dataset = pd.read_csv('LTC-USD.csv')
dataset
```

Fig 4.17 BTC-USD **Fig 4.18 ETH-USD** **Fig 4.19 LTC-USD**

3. DATA VISUALIZATION

Visualization of each dataset.

The dataset for this project are BTC-USD, ETH-USD and LTC-USD data taken from Yahoo Finance for the chosen time interval, comprising BTC Open, High, Low, Close, Adj Close, and Volume. In this situation, we retrieved all accessible data and elected to consider the most recent 1200 observations (n past total) in the script because the behavior in the early years may change significantly from what is seen currently.

BTC

We plotted the 'Close' price and 'Volume' to look for any visual anomaly. Indeed, an outstanding spike in the Volume on the 26th of February 2021 — around 3x its normal value. The relation between the closing price and volume is shown in Fig 4.20.

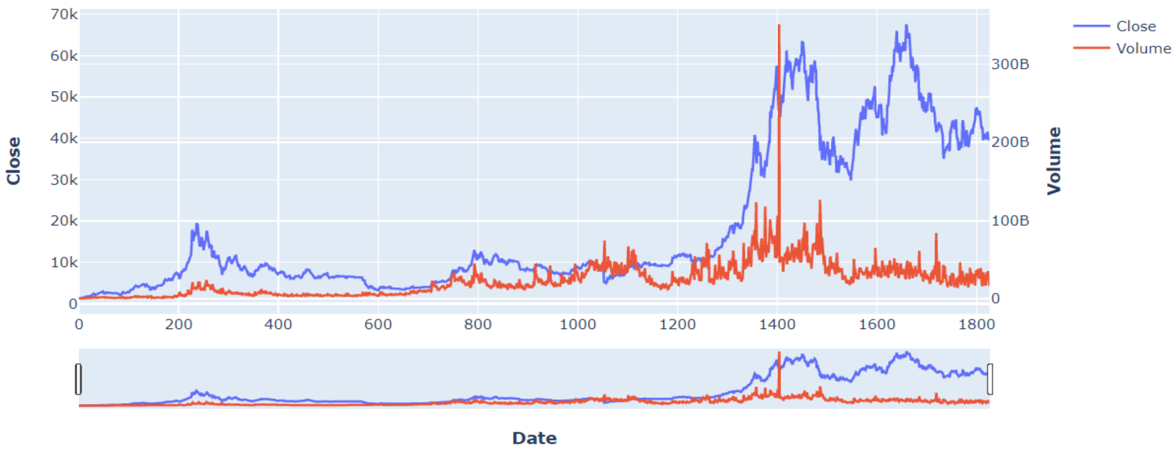

Fig 4.20: Graph correlation between Closing price and Volume (BTC)

In Fig 4.21. We are looking at the data points between March 1 , 2022 and March 30, 2022 of the Bitcoin(BTC) dataset. Bar style is used to show the volume traded each particular date

```
In [10]: # looking at march22 datapoints
         dataset[(dataset['Date']>'2022-03-01') & (dataset['Date']<'2022-04-01')].style.bar(subset=['
```

Out[10]:

	Date	Open	High	Low	Close	Adj Close	Volume
1773	2022-03-02	44357.617188	45077.578125	43432.851563	43924.117188	43924.117188	29183112630
1774	2022-03-03	43925.195313	44021.578125	41914.750000	42451.789063	42451.789063	24967782593
1775	2022-03-04	42458.140625	42479.613281	38805.847656	39137.605469	39137.605469	28516271427
1776	2022-03-05	39148.449219	39566.335938	38777.035156	39400.585938	39400.585938	16975917450
1777	2022-03-06	39404.199219	39640.175781	38211.648438	38419.984375	38419.984375	19745229902
1778	2022-03-07	38429.304688	39430.226563	37260.203125	38062.039063	38062.039063	28546143503
1779	2022-03-08	38059.902344	39304.441406	37957.386719	38737.269531	38737.269531	25776583476
1780	2022-03-09	38742.816406	42465.671875	38706.093750	41982.925781	41982.925781	32284121034
1781	2022-03-10	41974.070313	42004.726563	38832.941406	39437.460938	39437.460938	31078064711
1782	2022-03-11	39439.968750	40081.679688	38347.433594	38794.972656	38794.972656	26364890465
1783	2022-03-12	38794.464844	39308.597656	38772.535156	38904.011719	38904.011719	14616450657
1784	2022-03-13	38884.726563	39209.351563	37728.144531	37849.664063	37849.664063	17300745310
1785	2022-03-14	37846.316406	39742.500000	37680.734375	39666.753906	39666.753906	24322159070
1786	2022-03-15	39664.250000	39794.628906	38310.210938	39338.785156	39338.785156	23934000868
1787	2022-03-16	39335.570313	41465.453125	39022.347656	41143.929688	41143.929688	39616916192
1788	2022-03-17	41140.843750	41287.535156	40662.871094	40951.378906	40951.378906	22009601093

Fig 4.21: March 2022 data points(BTC).

ETH

We plotted the 'Close' price and 'Volume' to look for any visual anomaly. Indeed, similar to bitcoin we can see an outstanding spike in the Volume around data points number 1150 around 3x its normal value. Corresponding to the increase in volume we can also see the massive increase in the price as well around that time.

The relation between the closing price and volume is shown in Fig 4.22

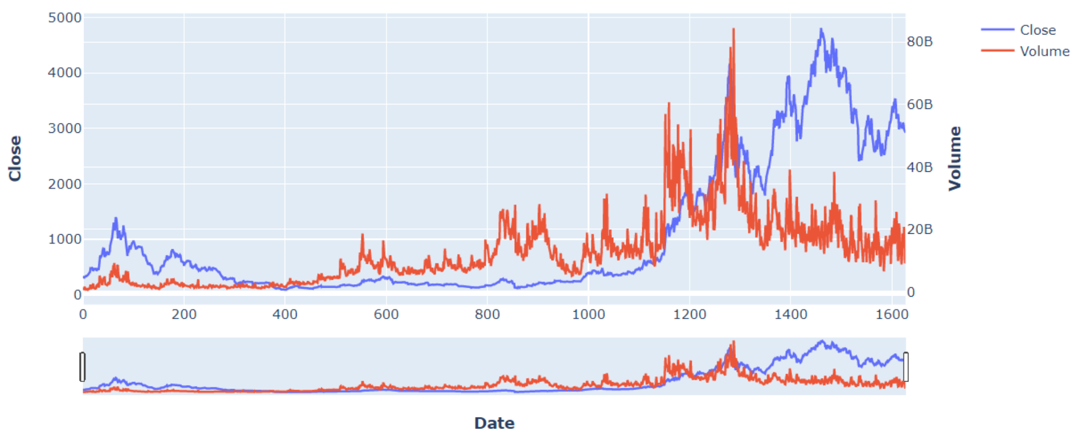

Fig 4.22:Graph correlation between Closing price and Volume (ETH)

In Fig 4.23. We are looking at the data points between March 1 , 2022 and March 30, 2022 of the Etheruem (ETH) dataset. Bar style is used to show the volume traded each particular date

```
# looking at march22 datapoints
dataset[(dataset['Date']>'2022-03-01') & (dataset['Date']<'2022-04-01')].style.bar(sub
```

	Date	Open	High	Low	Close	Adj Close	Volume
1574	2022-03-02	2972.471924	3026.868164	2919.943115	2950.118408	2950.118408	16636517503
1575	2022-03-03	2950.156738	2964.673340	2797.319336	2834.468994	2834.468994	13091199728
1576	2022-03-04	2834.987305	2835.176270	2587.748291	2617.156006	2617.156006	14496939024
1577	2022-03-05	2618.473633	2679.102539	2596.989990	2664.831055	2664.831055	8072368396
1578	2022-03-06	2664.943604	2673.637207	2555.037354	2555.037354	2555.037354	8872976607
1579	2022-03-07	2555.297607	2639.943115	2455.593750	2497.771240	2497.771240	14594098731
1580	2022-03-08	2497.721436	2618.166016	2489.755127	2576.747559	2576.747559	13922922903
1581	2022-03-09	2577.165283	2761.796387	2573.655273	2729.783447	2729.783447	14173665398
1582	2022-03-10	2729.116455	2729.116455	2566.193115	2608.048584	2608.048584	13292477213
1583	2022-03-11	2608.271240	2664.558594	2534.688232	2559.562988	2559.562988	12382419582
1584	2022-03-12	2559.660645	2606.438721	2559.126953	2574.754150	2574.754150	6532996574
1585	2022-03-13	2573.488037	2594.549805	2503.885254	2518.944580	2518.944580	8632000379
1586	2022-03-14	2518.486328	2604.034424	2505.299316	2590.696045	2590.696045	11244398839
1587	2022-03-15	2590.668945	2662.329590	2515.765869	2620.149658	2620.149658	12861105614

Fig 4.23: March 2022 data points(ETH).

LTC

We plotted the 'Close' price and 'Volume' to look for any visual anomaly. Indeed, similar to bitcoin we can see an outstanding spike in the Volume around data points number 200 around 5x its normal value.Corresponding to the increase in volume we can also see the massive increase in the price as well around that time But what's interesting is that the price increased multiple times around datapoint 700 and another massive spike around 1300. The relation between the closing price and volume is shown in Fig 4.24.

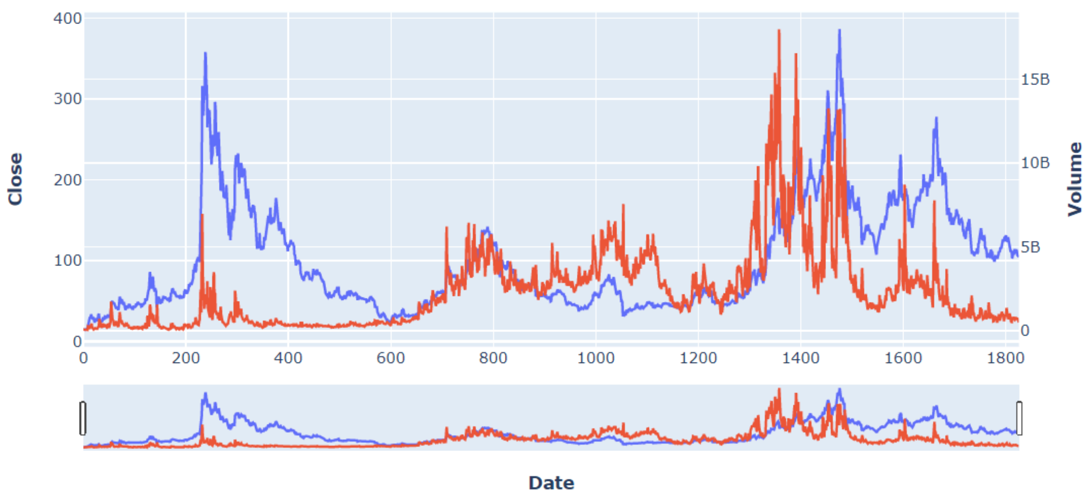

Fig 4.24:Graph correlation between Closing price and Volume (LTC)

In Fig 4.25. We are looking at the data points between March 1 , 2022 and March 30, 2022 of the Bitcoin(BTC) dataset. Bar style is used to show the volume traded each particular date

```
# looking at march22 datapoints
dataset[(dataset['Date']>'2022-03-01') & (dataset['Date']<'2022-04-01')].style.bar(s
```

	Date	Open	High	Low	Close	Adj Close	Volume
1773	2022-03-02	112.540298	114.243759	109.672516	110.351357	110.351357	838169853
1774	2022-03-03	110.356354	112.503716	108.962349	111.352119	111.352119	905649306
1775	2022-03-04	111.412834	112.356232	100.207321	101.391800	101.391800	930473366
1776	2022-03-05	101.408401	105.351410	100.218079	104.977188	104.977188	548783174
1777	2022-03-06	104.980621	105.609085	101.545616	101.593918	101.593918	563304861
1778	2022-03-07	101.595879	103.662041	97.236954	98.869263	98.869263	756482342
1779	2022-03-08	98.861572	102.160332	98.499702	100.617645	100.617645	699527175
1780	2022-03-09	100.621696	109.045860	100.621696	106.942093	106.942093	812074655
1781	2022-03-10	106.934196	106.972275	100.669655	102.655510	102.655510	814685365
1782	2022-03-11	102.649483	106.566574	100.503815	104.965355	104.965355	764305866
1783	2022-03-12	104.964760	107.766373	104.807281	105.832397	105.832397	536251476
1784	2022-03-13	105.812271	107.201889	101.904175	102.114799	102.114799	535011513
1785	2022-03-14	102.111214	105.776672	101.501625	105.673653	105.673653	743010065

Fig 4.25: March 2022 data points(LTC).

4. DATA PREPROCESSING

We looked for null values and discovered four days with no data. As a result, we decided to use Pandas' 'ffill' method to fill in missing data, with nulls being substituted with the prior observation (the last available observation will be propagated). Another option would have been to remove any missing values. Fig 4.26.

```
# use close only and fill NaN with ffil
df = dataset.set_index('Date')[features]#.tail(n_past_total)
df = df.set_index(pd.to_datetime(df.index))
df.fillna(method='ffill',inplace=True)
```

Fig 4.26: Filling of NULL values

We calculated the Pearson correlation between the Close price and the Volume. We decided to include Volume as a secondary feature used to predict the future prices for each crypto currency.. Fig 4.27

```
# looking at the correlation of the main possible variables
dataset[['Close','Volume']].corr()
```

Fig 4.27: Pearson correlation between the Close price and the Volume

Scaling the data between 0,1 to prevent higher dimensional values to affect the results. It is important to do that because the higher valued data points might affect the final results. For example in these datasets, the volume of coins traded each day is much bigger than the closing price of the coin on that day. In this case, we used the minmaxscaler but a standard scaler could also be used. Fig 4.28

```
# train test split
training_set = df.values
print('training_set.shape:\t', training_set.shape)
```

```
training_set.shape:      (1200, 2)
```

```
# scale
sc = MinMaxScaler(feature_range=(0,1))
training_set_scaled = sc.fit_transform(training_set)
print('training_set_scaled.shape: ', training_set_scaled.shape)
training_set_scaled
```

Fig 4.28: Creation of training set and scaling

Training Testing split

The Close Price and Volume for past observations (n past) will be used as the training dataset, and the output will be Close Price forecasts for n future days (n future). The model is Multivariate because we'll be using two variables as input.Secondly, we will re-arrange the training data in the required format for the model. For this, we defined the X_train array that contains the n_past observations that will be used to predict the n_future prices in a way that, if n_past is 30 and n_future is 10: Fig 4.29

```
# creating a data structure with 60 timesteps and 1 output
X_train = []
y_train = []

for i in range(n_past, len(training_set_scaled) - n_future + 1):
    X_train.append(training_set_scaled[i-n_past:i, :])
    y_train.append(training_set_scaled[i:i+n_future, 0])

X_train, y_train = np.array(X_train), np.array(y_train)
X_train.shape, y_train.shape
```

```
((1161, 30, 2), (1161, 10))
```

Fig 4.29: Training testing split

The Fig 4.30 shows that we need to reshape the train set to fit inside the models.

```
# reshaping (needed to fit RNN)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], n_features))
X_train.shape
```

```
(1161, 30, 2)
```

Fig 4.30 Reshaping to fit into model

5. BUILDING MODELS

In this case, the models were developed with Keras which is a Python library that uses TensorFlow in the backend. Keras' API simplifies the implementation of the Neural Network**.** The codes for the creation of each model are given below(Fig 4.31, 4.32, 4.33, 4.34)

CNN

```
# Building the CNN
from keras.layers.convolutional import MaxPooling1D
from keras.layers import Flatten
# Initialising the CNN
regressor = Sequential()

# Input layer
regressor.add(Conv1D(filters=64, kernel_size=2, activation=activation, input_shape=(X_train.shape[1], n_features)))
# Hidden layers

regressor.add(MaxPooling1D(pool_size=2))

regressor.add(Flatten())
# Adding the output layer
regressor.add(Dense(units=n_future))

# Compiling the CNN
regressor.compile(optimizer=optimizer, loss='mse')

# Model summary
regressor.summary()
```

Fig 4.31 CNN Model build

GRU

```
# Initialising the RNN
regressor = Sequential()

# Input layer
regressor.add(GRU(units=n_past, return_sequences=True, activation=activation, input_shape=(X_train.shape[1], n_features)))

# Hidden layers
for _ in range(n_layers):
    regressor.add(Dropout(dropout))
    regressor.add(GRU(units=n_neurons, return_sequences=True, activation=activation))

# Last hidden layer (changing the return_sequences)
regressor.add(Dropout(dropout))
regressor.add(GRU(units=n_neurons, return_sequences=False, activation=activation))

# Adding the output layer
regressor.add(Dense(units=n_future))

# Compiling the RNN
regressor.compile(optimizer=optimizer, loss='mse')

# Model summary
regressor.summary()
```

Fig 4.32: GRU Model build

LSTM

```
# Building the RNN

# Initialising the RNN
regressor = Sequential()

# Input layer
regressor.add(LSTM(units=n_past, return_sequences=True, activation=activation, input_shape=(X_train.shape[1], n_features)))
#regressor.add(LSTM(units=neurons, return_sequences=True, activation=activation, input_shape=(X_train.shape[1], 1)))

# Hidden layers
for _ in range(n_layers):
    regressor.add(Dropout(dropout))
    regressor.add(LSTM(units=n_neurons, return_sequences=True, activation=activation))

# Last hidden layer (changing the return_sequences)
regressor.add(Dropout(dropout))
regressor.add(LSTM(units=n_neurons, return_sequences=False, activation=activation))

# Adding the output layer
regressor.add(Dense(units=n_future))

# Compiling the RNN
regressor.compile(optimizer=optimizer, loss='mse')

# Model summary
regressor.summary()
```

Fig 4.33 LSTM Model build

Bi-LSTM

```
# Initialising the RNN
regressor = Sequential()

# Input layer
regressor.add(Bidirectional(LSTM(units=n_past, return_sequences=True, activation=activation, input_shape=(X_train.shape[1

# Hidden layers
for _ in range(n_layers):
    regressor.add(Dropout(dropout))
    regressor.add(Bidirectional(LSTM(units=n_neurons, return_sequences=True, activation=activation)))

# Last hidden layer (changing the return_sequences)
regressor.add(Dropout(dropout))
regressor.add(Bidirectional(LSTM(units=n_neurons, return_sequences=False, activation=activation)))

# Adding the output layer
regressor.add(Dense(units=n_future))

# Compiling the RNN
regressor.compile(optimizer=optimizer, loss='mse')

#summary
regressor.summary()
```

Fig 4.34 Bi-LSTM Model build

MODEL SUMMARY (LSTM for example)

The summary() method is a generic function used to produce result summaries of the results of various model fitting functions. The model summary of LSTM is shown in Fig 4.35

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 30, 30)            3960
_________________________________________________________________
dropout_3 (Dropout)          (None, 30, 30)            0
_________________________________________________________________
lstm_5 (LSTM)                (None, 30, 4)             560
_________________________________________________________________
dropout_4 (Dropout)          (None, 30, 4)             0
_________________________________________________________________
lstm_6 (LSTM)                (None, 30, 4)             144
_________________________________________________________________
dropout_5 (Dropout)          (None, 30, 4)             0
_________________________________________________________________
lstm_7 (LSTM)                (None, 30, 4)             144
_________________________________________________________________
dropout_6 (Dropout)          (None, 30, 4)             0
_________________________________________________________________
lstm_8 (LSTM)                (None, 30, 4)             144
_________________________________________________________________
dropout_7 (Dropout)          (None, 30, 4)             0
_________________________________________________________________
lstm_9 (LSTM)                (None, 4)                 144
_________________________________________________________________
dense_1 (Dense)              (None, 10)                50
=================================================================
Total params: 5,146
Trainable params: 5,146
Non-trainable params: 0
```

Fig 4.35 Model summary (LSTM)

In order to prevent overfitting, dropout and early stopping were included in all the models.

1. Dropout is like a layer added to use only a portion of the nodes of the next layer and drop-out the others; if dropout is 0.2, 20% of the nodes of the next layer will be ignored.
2. Early stopping halts the training when a monitored metric stops improving even after multiple iterations. In this case, the metric we want to minimize is ‘val_loss’, and a patience of 25 which is the number of epochs with no improvement after which training will be stopped.

6. TRAINING THE MODEL

Despite the fact that we ran multiple tests, the key improvements were found by fine-tuning the following:

1. Early stopping: It was extremely helpful in determining a good number of epochs to run and avoiding overfitting the model by running numerous epochs.

2. Dropout: Using this regularization strategy, the model's performance on test data was greatly improved. The model performed substantially better on taught data but much worse on test data without it.

3. The initial model merely used the Close price as an input, but this made the model

excessively sensitive to its own changes. The model predictions become smoother after adding the Volume as a second feature.

4. Activation function: Softsign was found to be slower but smoother than relu or sigmoid in terms of model convergence.

Fig 4.36 shows the use of the fit function with the parameters added to the method as mentioned above. Fig 4.37 shows the running of the epoch as the model iterates through the dataset.

```
# Fitting the RNN to the Training set
res = regressor.fit(X_train, y_train
                    , batch_size=30
                    , epochs=100
                    , validation_split=0.1
                   )
```

Fig 4.36 Training of Model

```
Epoch 1/100
14/14 [==============================] - 2s 111ms/step - loss: 0.0057 - val_loss: 0.0096
Epoch 2/100
14/14 [==============================] - 1s 106ms/step - loss: 0.0055 - val_loss: 0.0117 ETA:
Epoch 3/100
14/14 [==============================] - 1s 102ms/step - loss: 0.0063 - val_loss: 0.0103
Epoch 4/100
14/14 [==============================] - 1s 104ms/step - loss: 0.0063 - val_loss: 0.0069
Epoch 5/100
14/14 [==============================] - 1s 102ms/step - loss: 0.0058 - val_loss: 0.0063
Epoch 6/100
14/14 [==============================] - 1s 105ms/step - loss: 0.0056 - val_loss: 0.0074
Epoch 7/100
14/14 [==============================] - 1s 106ms/step - loss: 0.0056 - val_loss: 0.0066
Epoch 8/100
14/14 [==============================] - 1s 107ms/step - loss: 0.0056 - val_loss: 0.0079
```

Fig 4.37 Epoch runs

7. VALIDATION

Loss vs Validation loss

Plot the Loss with their corresponding validation sets defined as 0.1 (10%) in the 'model.fit' method. Ideally, we'd like to see both of them converge as the number of epochs grows. Overfitting is evident when both of them depart from one another. Dropout and Early Stopping cause them to converge up to a point in this example. Both will converge without them and then diverge once the model begins to overfit.

Fig 4.38 shows the code written for the visualization of the loss vs val_loss relation of each epoch. matplotlib is used to display the graph.

```python
results = res

history = results.history
plt.figure(figsize=(12,4))
plt.plot(history['val_loss'])
plt.plot(history['loss'])
plt.legend(['val_loss', 'loss'])
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```

Fig 4.38 Loss vs Val_loss graph code

We have shown the visualization of the loss vs validation loss graph. Loss is the metric used to assess the performance on the training set. While validation loss is a metric used to assess the performance of a deep learning model on the validation set. The validation set is a portion of the dataset set aside to validate the performance of the model. The graphs are as follows.

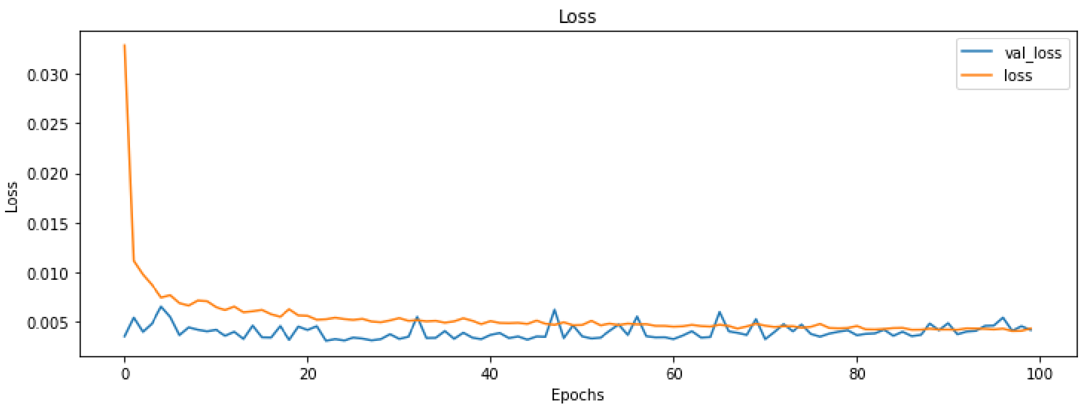

Fig 4.39: CNN-1d BTC Loss vs Val Loss

Fig 4.39 shows that the loss is gradually decreasing but the val loss is fluctuating. This is a result of overfitting which shows that the validation loss of the model is not going down as it is supposed to per epoch. This can be solved by using dropout and early stopping.

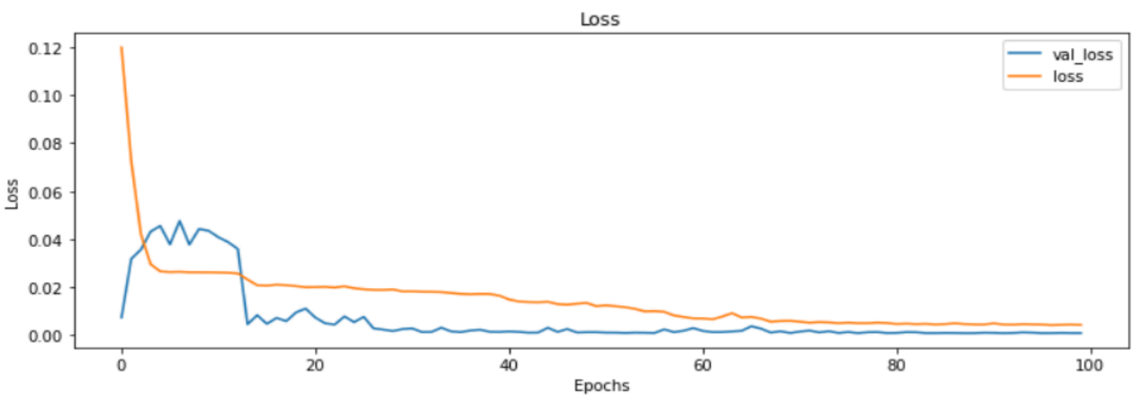

Fig 4.40: GRU-BTC Loss vs Val Loss

Unlike Fig 4.39, The graph in Fig 4.40 shows initial high val_loss value and then sudden decrease. Val_loss values are following the expected pattern as the value is going down every epoch. The model is not overfitting.

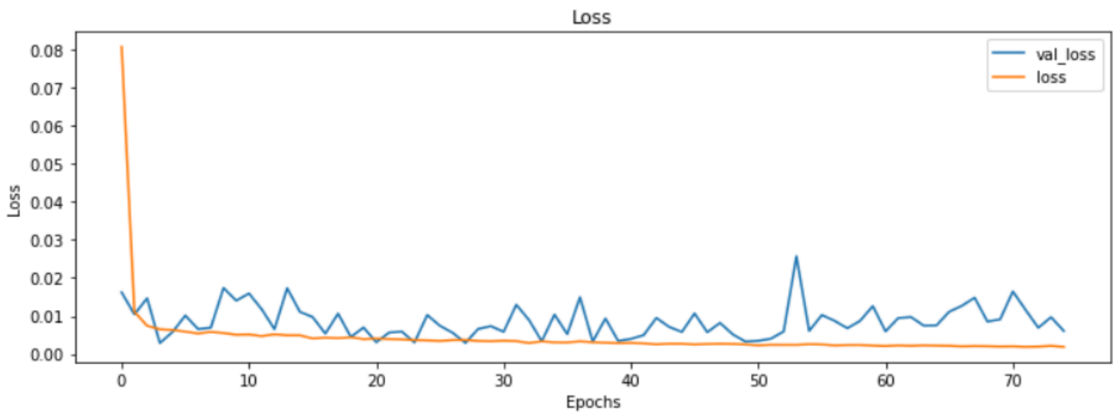

Fig 4.41: BiLSTM-BTC Loss vs Val Loss

Fig 4.41 shows that the loss is gradually decreasing but the val loss is again fluctuating. The fluctuation is more than what is shown in Fig 4.39. This is a result of overfitting which shows that the validation loss of the model is not going down as it is supposed to per epoch.

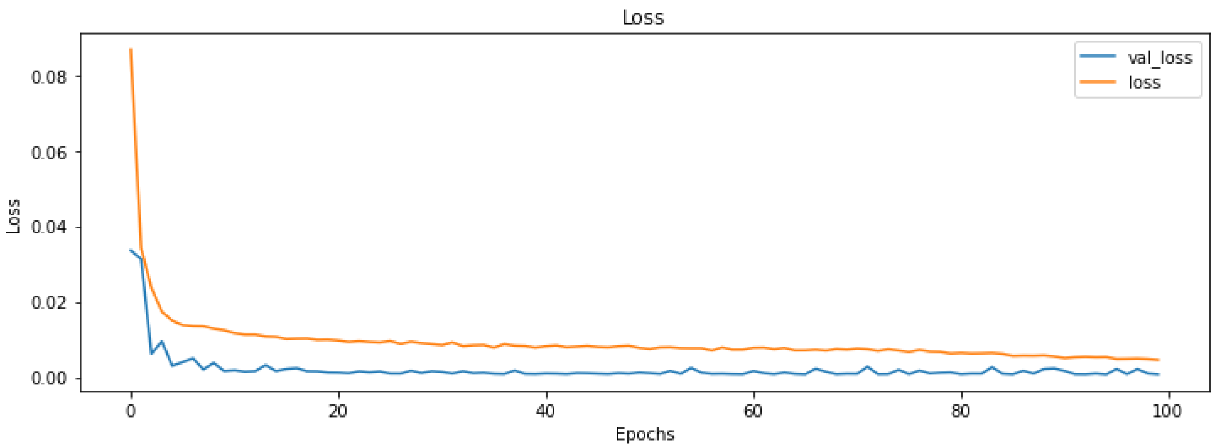

Fig 4.42 : LSTM-BTC Loss vs Val Loss

The graph in Fig 4.42 shows initial high val_loss value and then decreases suddenly. But val_loss values are following the expected pattern as the value is going down every epoch. The model is not overfitting as a result. This is close to what's to be expected from a model.

8. POST-PROCESSING OF DATA

It is important to post-process the data because remember, we have scaled all the data points between 0 and 1. If we do not inverse scale, the results and predictions will be displayed between 0 and 1. So we have created a function to invert the scaling. Since the scalar was trained into 2 features, it needs two features to perform the inverse scale.

For that purpose, this function will create a dummy array and concatenate it to the y_pred/y_true.

That dummy of ones will be dropped after performing the inverse_transform.(Fig 4.43)

```
def dummy_invscaler(y, n_features):
    '''
    Since the scaler was trained into 2 features, it needs two features to perform the inverse scaleer.
    For that purpose, this function will create a dummy array and concatenate it to the y_pred/y_true.
    That dummy of ones will be drop after performing the inverse_transform.
    INPUTS: array 'y', shape (X,)
    '''
    y = np.array(y).reshape(-1,1)
    if n_features>1:
        dummy = np.ones((len(y), n_features-1))
        y = np.concatenate((y, dummy), axis=1)
        y = sc.inverse_transform(y)
        y = y[:,0]
    else:
        y = sc.inverse_transform(y)
    return y
```

Fig 4.43 Inverse scaling

Chapter 05: RESULTS

Hypertuning of the Parameters

Hyperparameter tuning is an important application that affects the machine learning algorithm's performance significantly. With the best hyperparameters, the algorithm's performance may enhance remarkably and the model can conduct more accurate predictions. If the model runs with the best known parameters, the algorithm can learn tasks faster than the non-tuned model (Amirabadi et al., 2020). Therefore, before the final run of the DL algorithm, it is necessary to identify the best values for the parameters. In this study, the number of neurons in each layer, epoch size and batch size are used as the parameters to be tuned. One complete forward and backward run of the entire dataset during the execution of the model is called an epoch. The whole dataset must be divided into groups during the execution and the number of these divided groups are called batch size. Table 2 shows the chosen values of the parameters which demonstrated the best performance in the test set based on the RMSE.

Table 2- Best Parameters for the Models

Crypto currency	Model	Number of Neuron	Epoch	Batch	Total layers
BTC	CNN-1D	8	100	5	2
	LSTM	4	75	5	6
	Bi-LSTM	4	25	30	6
	GRU	5	75	5	4
LTC	CNN-1D	4	100	5	2
	LSTM	20	100	30	10
	Bi-LSTM	20	50	30	10
	GRU	20	100	20	6
ETH	CNN-1D	4	100	5	2
	LSTM	20	100	30	10
	Bi-LSTM	20	50	30	10
	GRU	20	100	20	6

5.1 Discussion on the Results Achieved

We test our model against actual data to see how well it performs once the training is complete and we are satisfied with how the Loss converged. The prediction date for the testing set starts from 15.04.22 to 24.04.22 .We can accomplish so by showing the model predictions and the actual values as follows:

CNN : Convolutional Neural Networks

- **BTC : Testing Result**

The predicted results seem to follow the general trend of the actual results. The predicted price is a bit lower than the actual one apart from the last two days where the predicted price seems to rise and the actual price drops.(Fig 5.1)

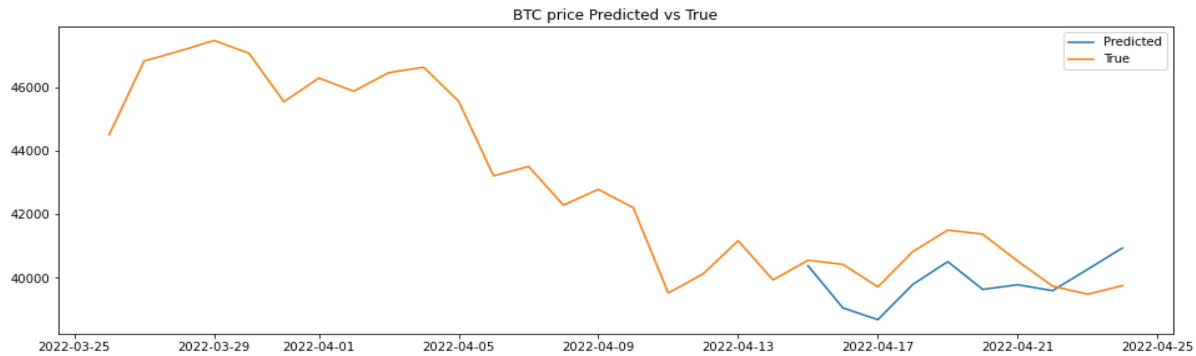

Fig 5.1:BTC Actual and predicted price using CNN

- **LTC:Testing Results**

Again similar to BTC the predicted results seem to follow the general trend of the actual results. The predicted price is a bit lower than the actual one apart from the last two days where the predicted price seems to rise and the actual price drops. (Fig 5.2)

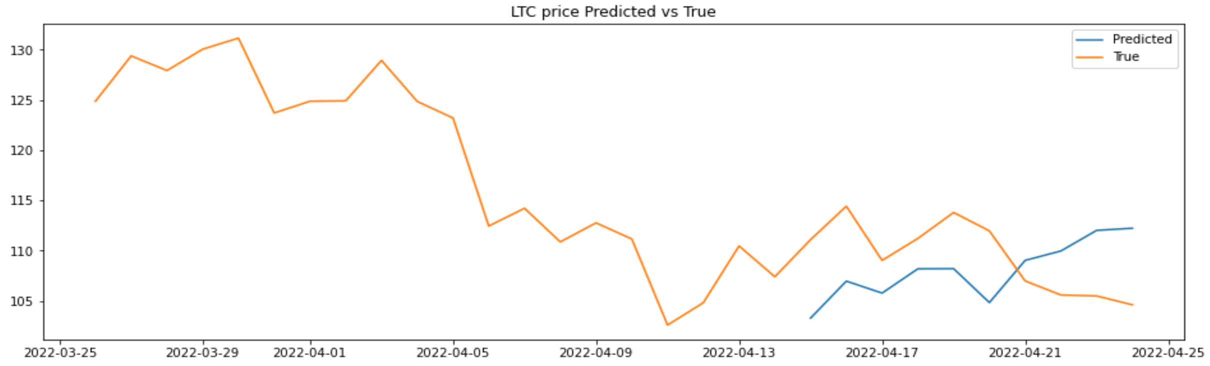

Fig 5.2: LTC Actual and predicted price using CNN

- **ETH:Testing Results**

It seems that all three cases follow the similar pattern .The predicted results seem to follow the general trend of the actual results. The predicted price is a bit lower than the actual one apart from the last two days where the predicted price seems to rise and the actual price drops.(Fig 5.3)

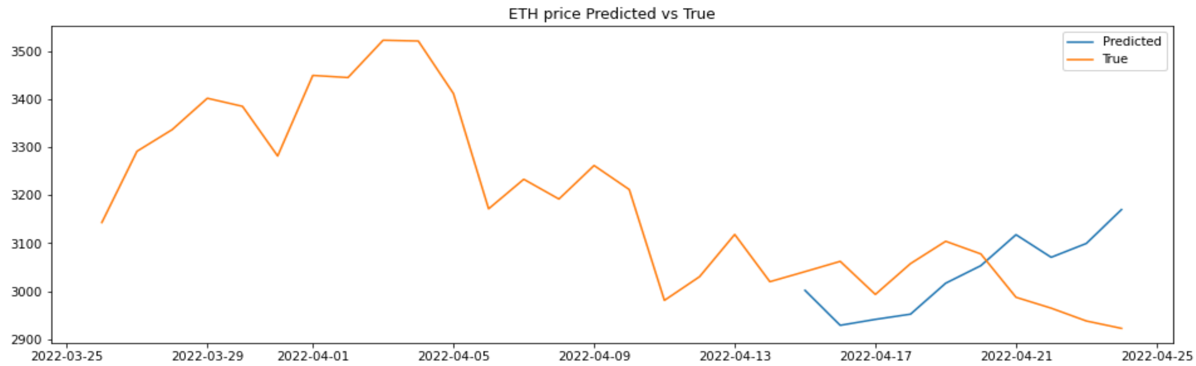

Fig 5.3: ETH Actual and predicted price using CNN

Table 3: CNN Model Results

Model	**Cryptocurrency**	**RMSE**	**MAE**
CNN:Convolutional Neural Network	**BTC**	1030.79	918.63
	LTC	5.85	5.48
	ETH	125.19	108.63

GRU : Gated Recurrent Unit

- **BTC:Testing Results**

Fig 5.4 shows that the predicted price initially increases with the actual price. The actual price declines in the end but the predicted price doesn't.

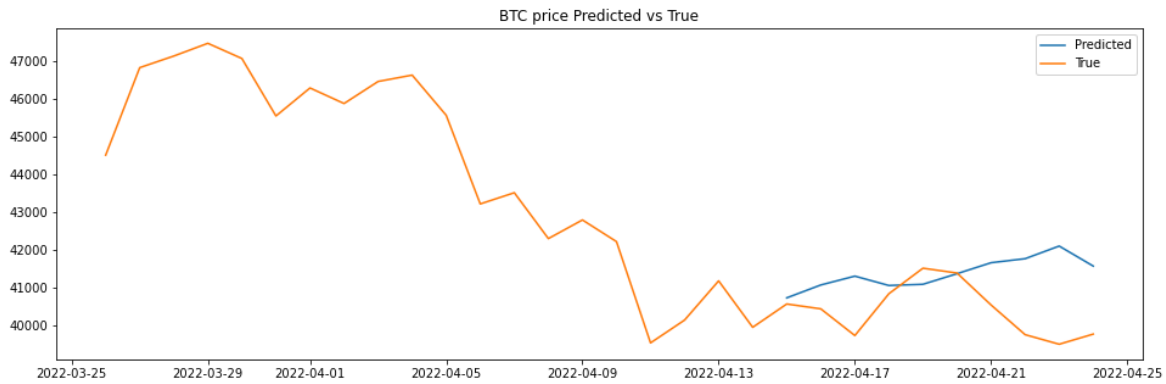

Fig 5.4: BTC Actual and predicted price using GRU

- **LTC:Testing Results**

Fig 5.5 shows that the predicted sort of follows the general trend of the actual price. Again the actual price declines in the end but the predicted price doesn't.

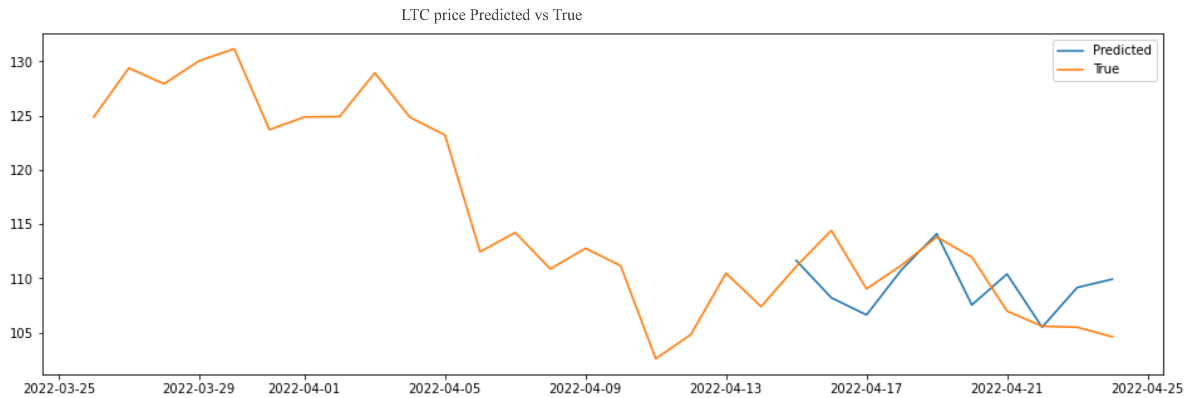

Fig 5.5: LTC Actual and predicted price using GRU

- **ETH: Testing Result**

Fig 5.6 shows that the predicted price initially follows the trend of the actual price but the predicted price is always higher than the actual price.

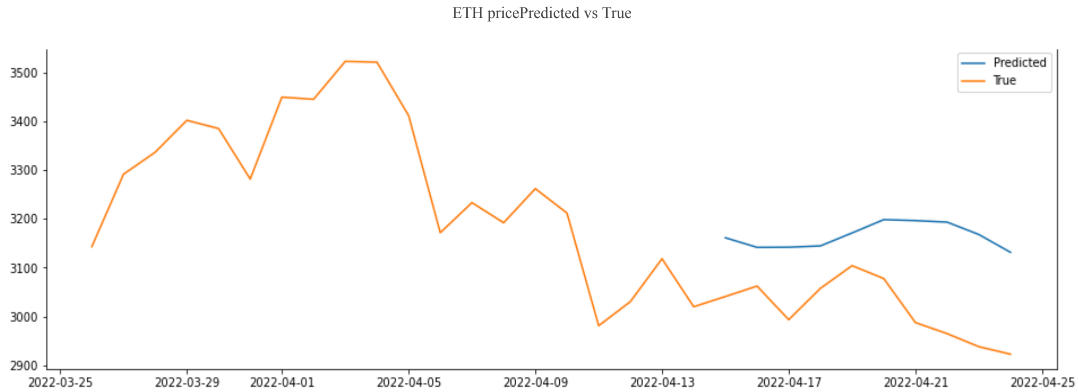

Fig 5.6: ETH Actual and predicted price using GRU

Table 4 :GRU Model Results

Model	Cryptocurrency	RMSE	MAE
GRU : Gated Recurrent Unit	**BTC**	1359.44	1057.52
	LTC	3.43	2.67
	ETH	161.82	149.93

LSTM:Long short term memory

- **BTC:Testing Results**

Fig 5.7 shows that the predicted price is able to follow the initial trend but the actual price initially increases while the prediction price only increases slightly. The actual price slightly increases in the end, the same as the predicted price.

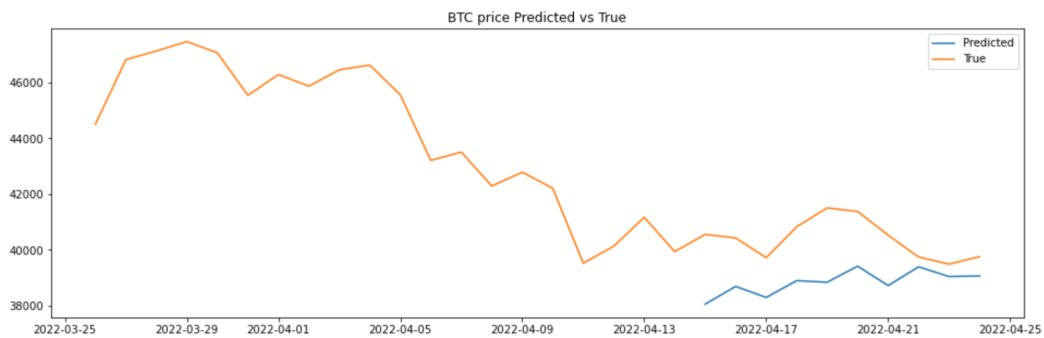

Fig 5.7: BTC Actual and predicted price using LSTM

- **LTC: Testing Results**

Fig 5.8 shows a sudden initial decline in the actual price which the predicted price is not able to follow. The actual price keeps on declining towards the end but the predicted price increases.

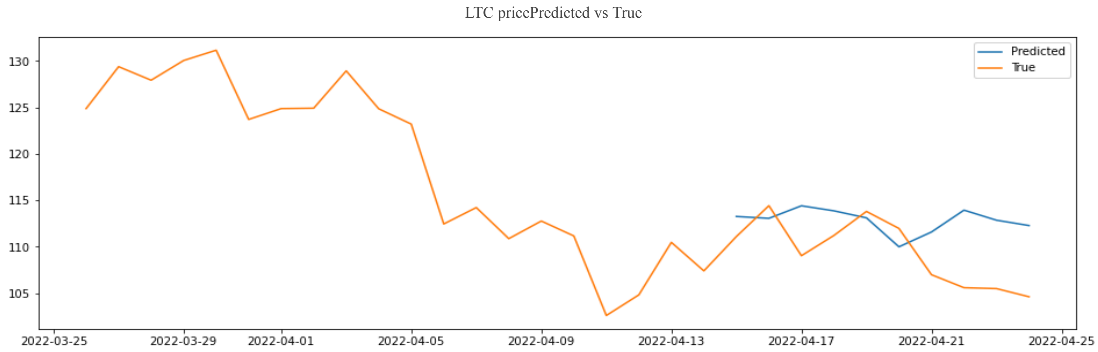

Fig 5.8: LTC Actual and predicted using price LSTM

- **ETH:Testing Results**

Fig 5.9 shows an initial decline in the actual price The actual price increases but the predicted price doesn't. Both declines towards the end with minimal difference in price.

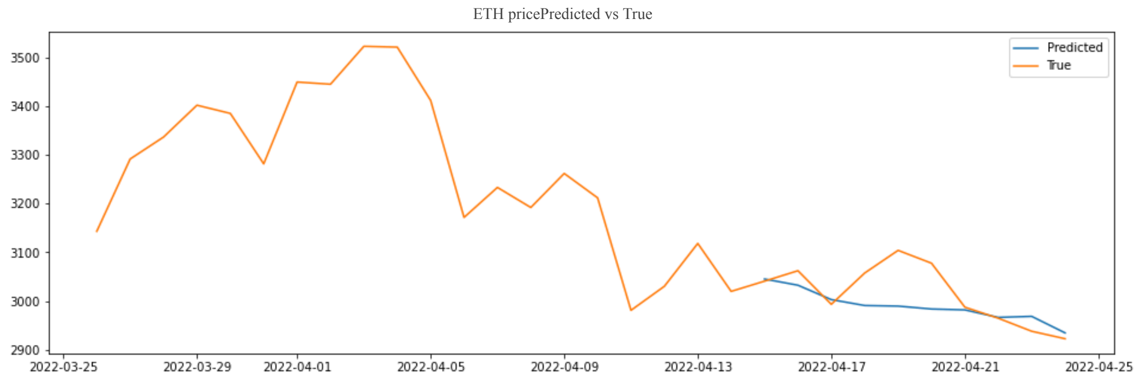

Fig 5.9: ETH Actual and predicted price using LSTM

Table 5 :LSTM model Results

Model	Cryptocurrency	RMSE	MAE
LSTM:Long short term memory	**BTC**	934.46	832.87
	LTC	5.00	4.21
	ETH	53.36	36.86

Bi-LSTM: Bidirectional Long short term memory

- **BTC:Testing Results**

Fig 5.10 shows a sudden initial decline in the actual price which the predicted price is not able to follow. The predicted price takes a dip on 19th April 2022. The actual price keeps on declining towards the end with a slight increase at the end but the predicted price increases continuously.

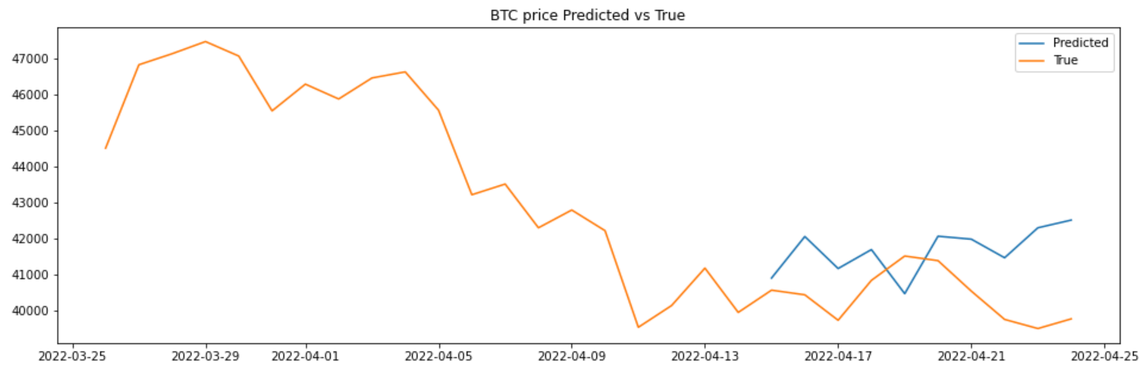

Fig 5.10: BTC Actual and predicted price using BiLSTM

- **LTC:Testing Results**

Fig 5.11 shows a sudden initial incline in the actual price which the predicted price is not able to follow. The actual price increased again on 19th April, but the predicted price did not increase to that extent. The actual price declines in the end but the predicted price doesn't.

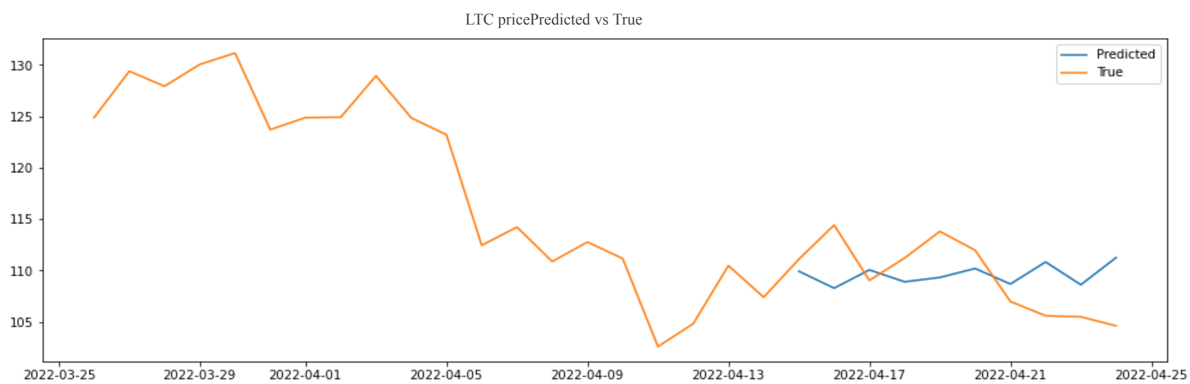

Fig 5.11: LTC Actual and predicted price using BiLSTM

- **ETH:Testing Results**

Fig 5.12 shows an initial decline in the actual price The actual price increases but the predicted price doesn't. Both declines towards the end with minimal difference in price.

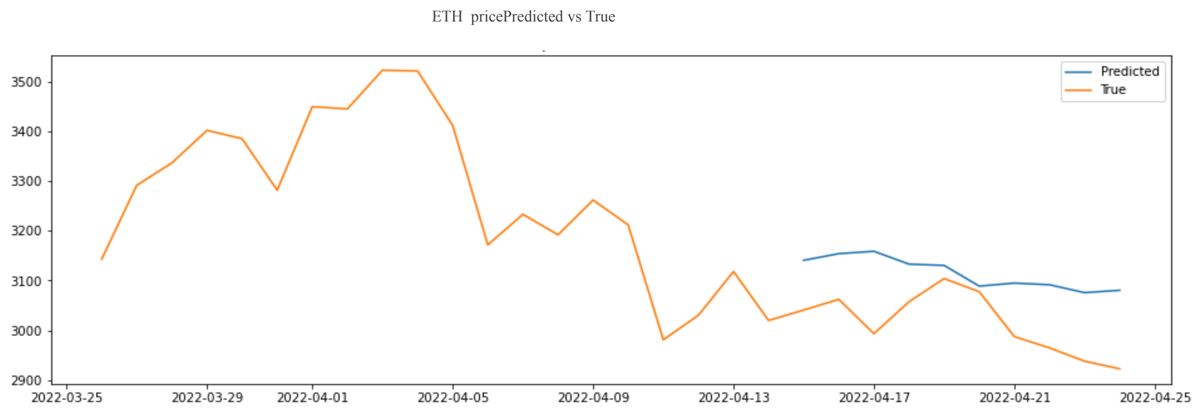

Fig 5.12: ETH Actual and predicted price using BiLSTM

Table 6 : BiLSTM model Results

Model	**Cryptocurrency**	**RMSE**	**MAE**
Bi-LSTM: Bidirectional Long short term memory	**BTC**	1658.72	1468.34
	LTC	3.90	3.35
	ETH	111.30	100.04

The results do not look very promising. It was not able to increase the model performance further by experimenting with all other parameters, CNN and RNN architecture in the period allotted. This is due to BTC Price's high volatility, which is heavily influenced by external variables rather than its own price and volume. Elon Musk's comments, countries dropping approval/restrictions, and businesses adopting BTC are just a few examples (VISA, PayPal, Tesla, etc).

ANALYSIS OF TEST RESULTS

Table 7 : Performance Results of the Proposed Models combined.

Cryptocurrency	Model	RMSE	MAE
BTC	CNN-1D	1030.79	918.63
	LSTM	934.46	832.87
	Bi-LSTM	1658.72	1468.34
	GRU	1359.44	1057.52
LTC	CNN-1D	5.85	5.48
	LSTM	5.00	4.21
	Bi-LSTM	3.90	3.35
	GRU	3.43	2.67
ETH	CNN-1D	125.19	108.63
	LSTM	53.36	36.86
	Bi-LSTM	111.30	100.04
	GRU	161.82	149.93

This section shows the results obtained from Convolutional Neural Network(CNN), long short-term memory (LSTM), gated recurrent unit (GRU), and bidirectional LSTM (bi-LSTM) algorithms using three types of popular cryptocurrency: BTC, ETH, and LTC. For each model, the results are illustrated in the above table(Table 7). The model that gives the lowest RMSE and MAE is considered the best model. To be noted that all these results are in USD.

BITCOIN(BTC)

The LSTM model seems to give the least Mean Absolute Error (MAE) of 823.87 and Root Mean Squared Error (RMSE) of 934.46. Although the CNN-1d model is not far off with MAE of 918.63 and RMSE of 1030.79. It shows more promising results compared to GRU and Bi-LSTM which was unexpected. As a Feedforward Neural Network(FNN) model, CNN performs really well against the two Recurrent Neural Network(RNN) models (GRU and Bi-LSTM). RNN models are expected to handle Time series data better but it's not the case here. The most surprising result was the Bi-LSTM one with MAE and RMSE of 1468.34 and 1658.72 respectively. Usually Bi-LSTM models are expected to perform similar or even better than LSTM in some cases but final error results of the Bi-LSTM in this case was almost 50% more than LSTM. Our results in the case of Bi-LSTM vary significantly compared to the results achieved by the authors of [20]. GRU also performs worse than CNN which was unexpected although GRU uses a more primitive RNN mechanism.

LITECOIN(LTC)

In the case of Litecoin(LTC), the results went more towards the expected side. Here, GRU performed the best, which was a bit unexpected but it was observed before in previous experiments and research that it sometimes performs better than LSTM and Bi-LSTM. Similar results were seen by the authors of [19]. The MAE and RMSE of GRU were 2.67 and 3.43 respectively. The second best model was the Bi-LSTM with MAE of 3.35 and RMSE of 3.90. This is really different from what we have seen previously in the case of Bitcoin(BTC). It performed much better and was close to GRU which showed the best results. The third best was LSTM with MAE of 4.21 and RMSE of 5.00. And the worst was the CNN-1d. The results in the case of Litecoin were pretty much what was expected.

ETHEREUM(ETH)

Here, LSTM again performed the best with the least Mean Absolute Error (MAE) of 36.86 and Root Mean Squared Error (RMSE) of 53.36. Bi-LSTM performed the second best with MAE of 100.04 and RMSE of 111.30. But what is fascinating is that the difference between LSTM and Bi-LSTM in terms of error is huge. LSTM performed extremely well for ethereum. Another result that was unexpected was that CNN with MAE of 108.63 and RMSE of 125.19 performed way better than GRU. GRU was the worst with MAE of 149.93 and RMSE of 161.82. Fig 2.6 shows that the predicted line was very much off the actual price line. Also the predicted prices were much higher than the actual one.

CONCLUSION

In this report, four types of machine learning algorithms are constructed and used for predicting the prices of three types of cryptocurrency—BTC, ETH, and LTC. Performance measures were conducted to test the accuracy of different models as shown in Tables 2.1.

Then, we compared the actual and predicted prices. Performance scores - RMSE and MAE - were calculated for every currency to test the accuracy of the proposed models.

Based on the results.

1. LSTM performed the best for two coins, BTC and ETH. It actually performed extremely well for ETH. CNN actually performed better than expected.
2. The most polarizing result was the BTC prediction by Bi-LSTM. For the other two coins, th Bi-LSTM prediction errors were better.
3. Similarly GRU also performed in the same manner as Bi-LSTM. Its prediction error for Litecoin(LTC) was very low and provided the most accurate prediction.
4. Our predictions differed from what the authors achieved in [19] where the bi-directionals(GRU and LSTM) models performed better than the traditional LSTM and GRU models.

So we can conclude that the LSTM model performed consistently with the best prediction for BTC and ETH. GRU and Bi-LSTM were the most inconsistent for all three datasets. CNN 1-d’s predictions were reasonable providing comparable results to all three RNN models. Different predictors of digital currency will be a topic of research in future studies. More precisely, machine learning techniques will be used to assess how people's tweets and the feelings expressed in those tweets would affect the pricing of cryptocurrencies.

Predicted Results of Models (For next 10 days)

Finally, we predict and plot Bitcoin's price for the next 10 days from 25.04.2022 to 5.05.22

➢ CNN:Convolutional Neural Networks

- **BTC**

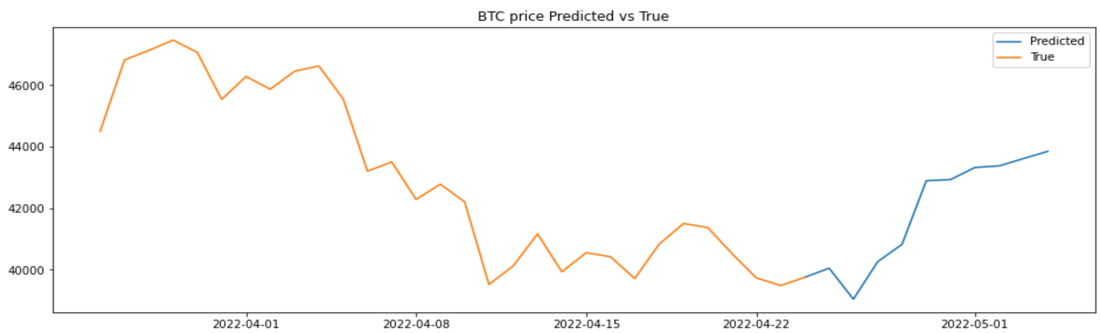

Fig 5.13 : BTC prediction using CNN

- **LTC**

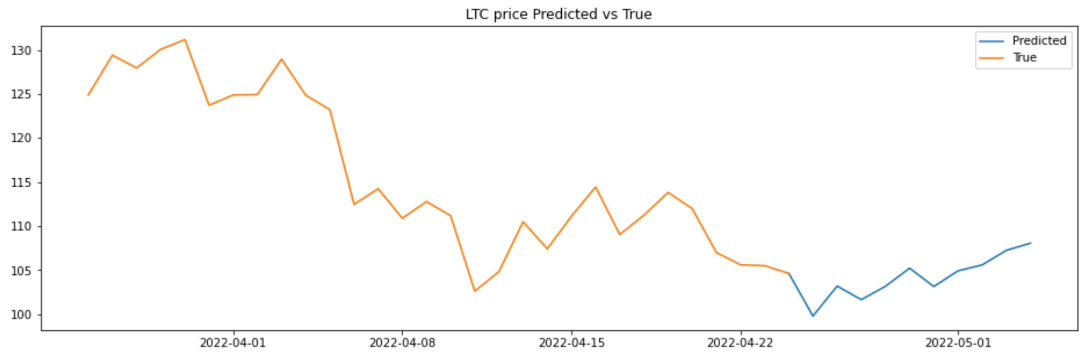

Fig 5.14 : LTC prediction using CNN

- **ETH**

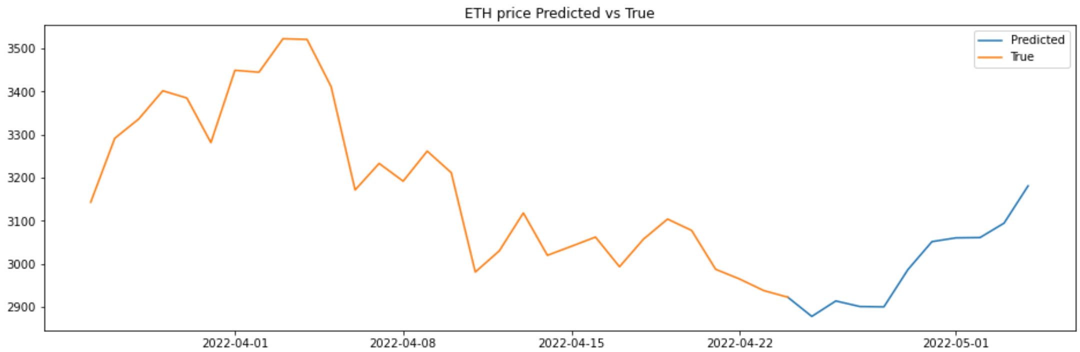

Fig 5.15 : ETH prediction using CNN

Table 8: Closing Price Prediction using CNN

BTC		LTC		ETH	
	Close		Close		Close
2022-04-24	39755.301	2022-04-24	104.623	2022-04-24	2922.733
2022-04-25	40051.015	2022-04-25	99.780	2022-04-25	2878.086
2022-04-26	39044.900	2022-04-26	103.189	2022-04-26	2913.979
2022-04-27	40262.926	2022-04-27	101.642	2022-04-27	2901.059
2022-04-28	40825.850	2022-04-28	103.162	2022-04-28	2900.321
2022-04-29	42893.316	2022-04-29	105.219	2022-04-29	2986.828
2022-04-30	42931.544	2022-04-30	103.132	2022-04-30	3051.710
2022-05-01	43324.909	2022-05-01	104.912	2022-05-01	3060.517
2022-05-02	43376.830	2022-05-02	105.584	2022-05-02	3061.150
2022-05-03	43614.894	2022-05-03	107.238	2022-05-03	3094.698
2022-05-04	43849.684	2022-05-04	108.052	2022-05-04	3181.240

➢ GRU:Gated Recurrent Unit

- **BTC**

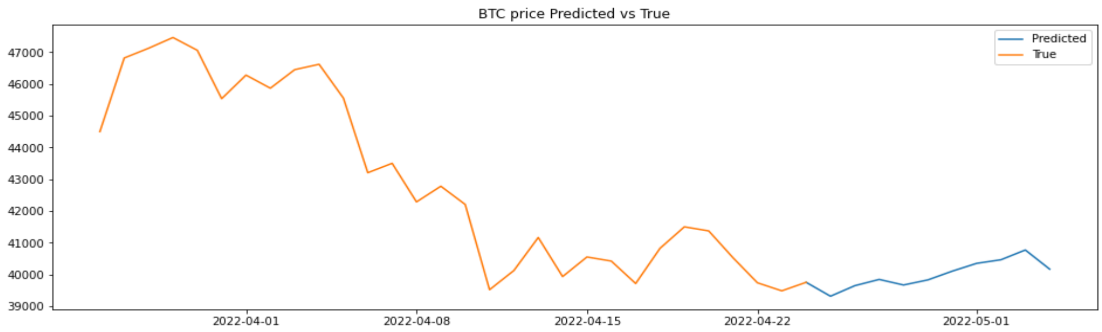

Fig 5.16: BTC Prediction using GRU

- **LTC**

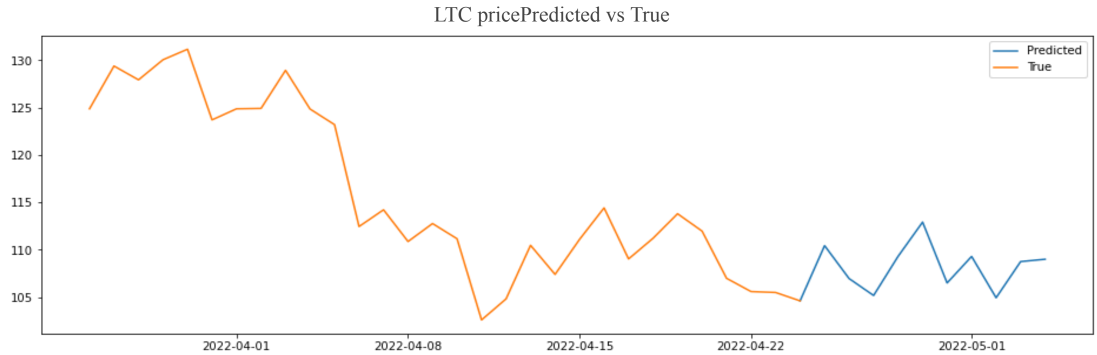

Fig 5.17: LTC Prediction using GRU

- **ETH**

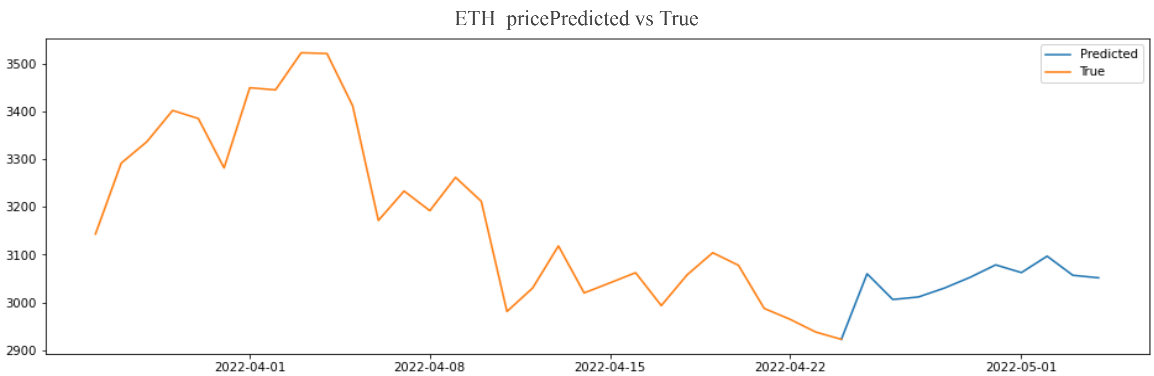

Fig 5.18: BTC Prediction using GRU

Table 9 :Closing Price Prediction using GRU

BTC		**LTC**		**ETH**	
	Close		**Close**		**Close**
2022-04-24	39755.301	**2022-04-24**	104.623	**2022-04-24**	2922.733
2022-04-25	39317.181	**2022-04-25**	110.446	**2022-04-25**	3060.066
2022-04-26	39649.721	**2022-04-26**	106.974	**2022-04-26**	3006.178
2022-04-27	39846.397	**2022-04-27**	105.188	**2022-04-27**	3011.501
2022-04-28	39670.616	**2022-04-28**	109.319	**2022-04-28**	3029.964
2022-04-29	39831.829	**2022-04-29**	112.931	**2022-04-29**	3052.462
2022-04-30	40106.510	**2022-04-30**	106.509	**2022-04-30**	3078.677
2022-05-01	40352.321	**2022-05-01**	109.307	**2022-05-01**	3062.518
2022-05-02	40466.323	**2022-05-02**	104.947	**2022-05-02**	3096.910
2022-05-03	40773.743	**2022-05-03**	108.765	**2022-05-03**	3056.874
2022-05-04	40169.380	**2022-05-04**	109.011	**2022-05-04**	3051.588

➢ LSTM: Long short Term Memory

- **BTC**

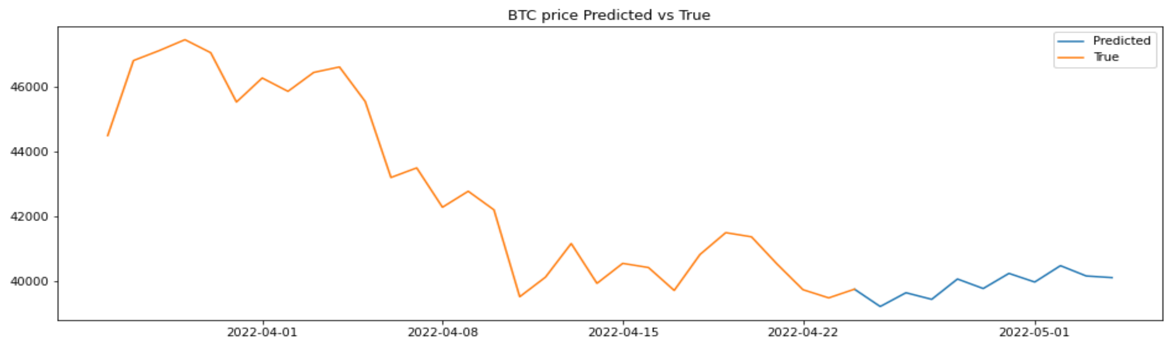

Fig 5.19 : BTC Prediction Using LSTM

- **LTC**

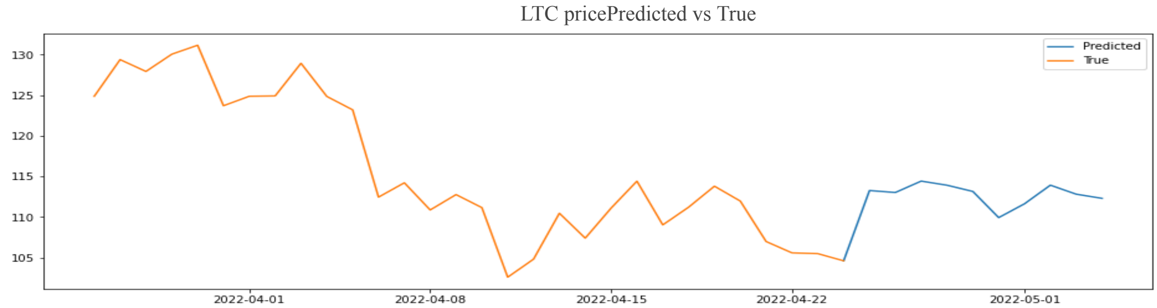

Fig 5.20 : LTC Prediction Using LSTM

- **ETH**

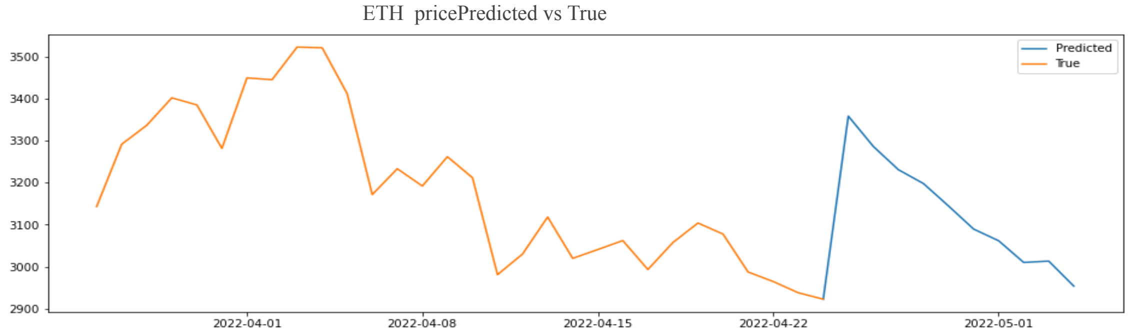

Fig 5.21 : ETH Prediction Using LSTM

Table 10 : Closing Price Prediction using LSTM

BTC		LTC		ETH	
	Close		**Close**		**Close**
2022-04-24	39755.300781	**2022-04-24**	104.622604	**2022-04-24**	2922.732666
2022-04-25	39219.969663	**2022-04-25**	113.276940	**2022-04-25**	3358.487495
2022-04-26	39645.938736	**2022-04-26**	113.032912	**2022-04-26**	3286.233335
2022-04-27	39441.651505	**2022-04-27**	114.437519	**2022-04-27**	3230.924853
2022-04-28	40070.117863	**2022-04-28**	113.930898	**2022-04-28**	3197.955046
2022-04-29	39775.916406	**2022-04-29**	113.163878	**2022-04-29**	3144.587565
2022-04-30	40243.773293	**2022-04-30**	109.941179	**2022-04-30**	3089.684931
2022-05-01	39976.313946	**2022-05-01**	111.647488	**2022-05-01**	3061.839590
2022-05-02	40481.935648	**2022-05-02**	113.938918	**2022-05-02**	3010.211966
2022-05-03	40163.707599	**2022-05-03**	112.823347	**2022-05-03**	3013.400009
2022-05-04	40113.535545	**2022-05-04**	112.315335	**2022-05-04**	2953.825546

➢ BILSTM: Bidirectional Long short Term Memory

- **BTC**

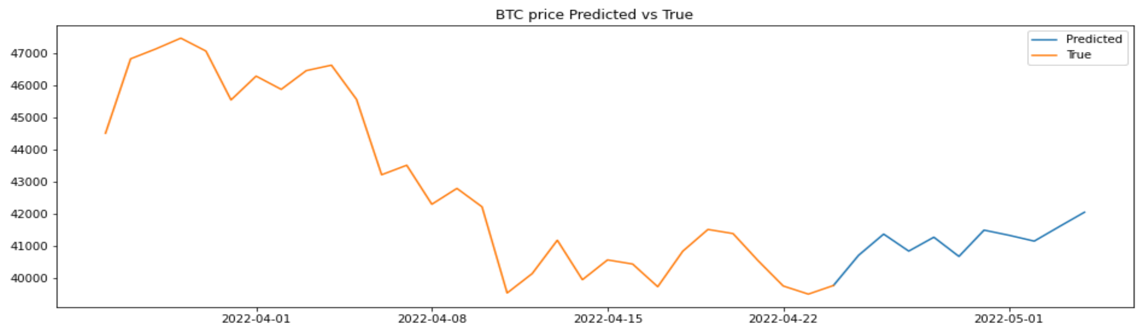

Fig 5.22 :BTC Prediction Using Bi-LSTM

- **LTC**

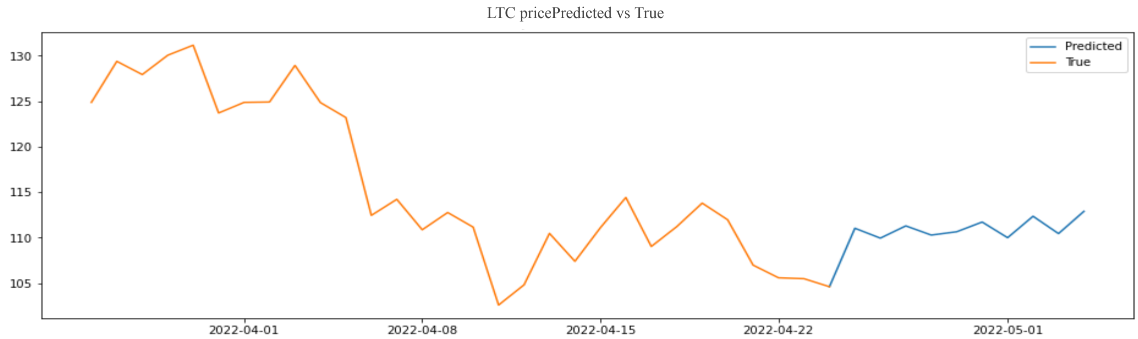

Fig 5.23 : LTC Prediction Using Bi-LSTM

- **ETH**

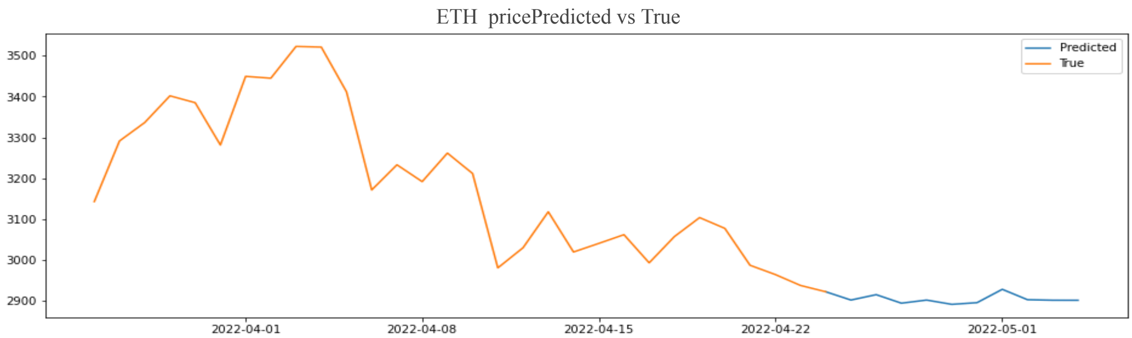

Fig 5.24 : ETH Prediction Using Bi-LSTM

Table 11 : Closing Price PredictionUsing Bi-LSTM

BTC		LTC		ETH	
	Close		Close		Close
2022-04-24	39755.301	**2022-04-24**	104.623	**2022-04-24**	2922.733
2022-04-25	40695.666	**2022-04-25**	111.049	**2022-04-25**	2902.393
2022-04-26	41358.279	**2022-04-26**	109.960	**2022-04-26**	2915.587
2022-04-27	40827.086	**2022-04-27**	111.310	**2022-04-27**	2894.739
2022-04-28	41260.540	**2022-04-28**	110.299	**2022-04-28**	2902.358
2022-04-29	40661.351	**2022-04-29**	110.662	**2022-04-29**	2891.899
2022-04-30	41481.678	**2022-04-30**	111.734	**2022-04-30**	2896.007
2022-05-01	41320.143	**2022-05-01**	110.007	**2022-05-01**	2928.661
2022-05-02	41138.796	**2022-05-02**	112.366	**2022-05-02**	2903.237
2022-05-03	41592.364	**2022-05-03**	110.462	**2022-05-03**	2902.001
2022-05-04	42039.369	**2022-05-04**	112.909	**2022-05-04**	2901.896

5.2 Application of the Minor Project

Bitcoin as a cryptocurrency has risen to prominence in the international financial landscape since its inception, coinciding with the global financial crisis of 2008 and the resulting lack of trust in the financial system. It has garnered widespread media attention, as well as the attention of regulators, government institutions, institutional and individual investors, academia, and the general public[21].

The success of bitcoin, as measured by its quick market capitalization increase and price appreciation, led to the emergence of a large number of other cryptocurrencies (altcoins) that, for the most part, differ from bitcoin in only a few criteria (e.g., block time, currency supply, and issuance scheme). By now, the cryptocurrency market has grown to be one of the world's largest unregulated markets, with more than 5.7 thousand cryptocurrencies, 23 thousand online exchanges, and a total market capitalization of more than 270 billion USD (data obtained from the CoinMarketCap site—https://coinmarketcap.com/)[21].

Because of the volatility of the crypto currencies as mentioned above, they are one of the most popular means for financial traders to earn profits. They can buy coins when there is dip in prices and sell them when the price rises. This change in price can happen within days and even hours because of the volatility of the crypto coins. Cryptocurrencies provide a faster method of earning profits because of its volatility than stocks. Stocks are much less volatile. But it has to be kept in mind that, it is very much possible to lose money as it is possible to earn profits. Due to this fact, it is very important for traders to forecast the prices of these coins. ML and Deep Learning is a good means to be able to get a good prediction beforehand.

Our project also serves a research purpose of studies of different Machine Learning models and their performance towards predicting time series data. They will help in a better understanding of these models and to get a picture of which model is the most efficient and reliable. Cryptocurrencies have been around for more than a decade but their popularity and usability are rising at a rapid pace and will grow exponentially in the future. They will soon and in some way have already become an integral part of our society and economy. Since prediction is so important in the investment process, many people rely on it to make money.

5.3 Limitation of the Minor Project

The limitations of the project are listed below:

1. The main limitation is that almost all results do not look very promising. None of our four models were able to completely follow the trend in the testing dataset. The main reason for this is the volatility of the price of Crypto currencies. We have taken into account only 2 features namely Closing price and Volume. But the price of crypto currency depends on many other factors as well. Elon Musk's comments, countries dropping approval/restrictions, and businesses adopting BTC are just a few examples (VISA, PayPal, Tesla, etc). Even the recent Russian invasion of Ukraine and the ban of crypto in certain countries have greatly affected the price of the coins.
2. Although machine learning has been successful in predicting stock market prices using a variety of time series models, it has been limited in its use to predicting cryptocurrency prices. The reason for this is obvious: as we have mentioned before, cryptocurrency values are influenced by a variety of factors such as technological advancements, internal competitiveness, market pressure to produce, economic troubles, security concerns, political factors, and so on. Because of their tremendous volatility, they have a huge profit potential if smart inventing tactics are used. Unfortunately, cryptocurrencies are less predictable than typical financial predictions such as stock market projections due to their lack of indexes.
3. Even though we used dropout,some of our models have shown a certain degree of overfitting which have affected the final results.
4. Some of the future predicted prices show a huge rise in price on a single day which is fairly unlikely to occur but is possible. The LSTM model seems to produce such predictions more often.
5. Although previous works from authors and results mentioned in [19] shows that Deep learning models provide much more accurate results than traditional Machine learning models, the results were still a bit far off. Because the problem we're trying to solve is a random walk process or something very close to one, any prediction effort may be of poor quality, or the problem is simply too complicated for even advanced deep learning algorithms to identify any pattern that would lead to a reliable forecast. A solution to this is to take more recent data points from the data set because they are more likely to be similar to the short future prices. When a time series prediction problem follows a random walk process or it is so complicated that most models face it as a random process, then the more efficient method to face it, is

the employment of present values as the prediction values for the next state.

However, as previously stated, the DL models did not attain a notable performance score in our experiments, as their score was nearly identical to the ML models achieved before by other authors and researchers. As a result, we conclude that these advanced DL models are unable to accurately predict cryptocurrency prices because the datasets used, in the specific form in which we "fed" them to our prediction models, are likely to follow a random walk process, and thus do not contain enough information to make accurate and reliable future predictions. So it is to be noted that this project was done for educational purposes and is not to be used for financial advice.

5.4 Future Work

1. We can include more features to predict the cryptocurrency price as we have used only the closing price and volume to predict the price. As we have mentioned in previous sections, cryptocurrency prices depend on a lot more factors than just closing price and volume such as sociale hype, the trader's sentiments , twitter data, government regularization, technical development, billionaires investments,altcoin data etc. The more of these features we include, the more accurate our predictions will be. The main hurdle will be to collect these datasets. Due to time constraint and lack of resources we were only able to include Volume and closing price as the main variables.
2. We can also include other models to compare the results, such as, ARIMA, SARIMA, Facebook Prophet, among others. We can also combine various models such as LSTM and CNN, LSTM and GRU and various such ML models.
3. Like the authors of [20] we can use bi-GRU models to predict prices and also improve our Bi-LSTM models by increasing the number of layers and neurons.
4. In future research, we'll look into other factors that could influence cryptocurrency prices, with a particular focus on the impact that social media in general, and tweets in particular, can have on cryptocurrency prices and trading volume, by analyzing tweets using natural language processing (NLP) techniques and sentiment analysis.
5. Increase the number of datapoints by reducing the time intervals. For this project, we used daily prices, but we could try with hourly prices for example.
6. We can also create a crypto trading bot that can sell and buy automatically depending on the market. It will have an integrated model that will help with the prediction and depending on the rise and fall of prices, it can invest or sell. The bot will be connected to a few major online trading platforms such as Binance or WazirX where the transactions of coins can occur.

5.5 References

1. Yahoo finance (Yahoo Finance - Stock Market Live, Quotes, Business & Finance News.[Online]. Available : yahoofinanceapi.com.[Accessed April 12 2022].
2. A.Levi, “What Makes Cryptocurrency Go Up or Down?”. [Online].Available: fool.com . [Accessed April 15, 2022].
3. J.Royal, “Cryptocurrency vs. stocks: What’s the better choice for you?”. [Online]. Available : Bankrate.com.[Accessed April 16 2022].
4. S.Jayawardhana, “Sequence Models & Recurrent Neural Networks (RNNs) Understanding Deep Recurrent Neural Networks (RNNs),” towardsdatascience.com . [Accessed April 18 2022].
5. A.K.Pandey, “Hands-On Stock Price Time Series Forecasting using Deep Convolutional Networks,” August 19, 2021. Available : analyticsvidhya.com. [Accessed April 18 2022].
6. N.A.Hitam , A.R.Ismail, “Comparative Performance of Machine Learning Algorithms for Cryptocurrency Forecasting.” *Indones. J. Electr. Eng. Comput. Sci.* pp.1121–1128,2018.
7. V.Derbentsev , V.Babenko , K.Khrustalev , H.Obruch , S.Khrustalova, “ Comparative Performance of Machine Learning Ensemble Algorithms for Forecasting Cryptocurrency Prices.” *Int. J. Eng. Trans.* vol 34,pp.140–148.A Basics 2021.
8. S.Karasu , A.Altan , Z.Sarac , R.Hacioglu, “Prediction of Bitcoin prices with machine learning methods using time series data”, presented at 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2–5 May 2018.
9. W.Yiying , Z.Yeze, “ Cryptocurrency Price Analysis with Artificial Intelligence,” In Proceedings of the 5th International Conference on Information Management (ICIM), Cambridge, UK; pp. 97–101,24–27 March 2019 [Accessed April 20 2022].
10. F.Valencia , A.G.Espinosa , B.V.Aguirre, “Price Movement Prediction of Cryptocurrencies Using Sentiment Analysis and Machine Learning.”21, 589. Entropy 2019.[Accessed April 21 2022
11. A.Jain , S.Tripathi , H.D.Dwivedi , P.Saxena, “Forecasting Price of Cryptocurrencies Using Tweets Sentiment Analysis” In Proceedings of the 11th International Conference on Contemporary Computing (IC3), Noida, India ,2–4 August 2018, pp. 1–7.
12. P.T.Yamak , L.Yujian , P.K.Gadosey, “ A Comparison between ARIMA, LSTM, and

GRU for Time Series Forecasting," In proceeding of the 2nd International Conference on Algorithms, Computing and Artificial Intelligence, Sanya, China, 20–22 December 2019,pp. 49–55.

13. E.C.Nisa , Y.D.Kuan, "Comparative Assessment to Predict and Forecast Water-Cooled Chiller Power Consumption Using Machine Learning and Deep Learning Algorithms" vol 13 ,14 Dec.2020.[Accessed April 21 2022].
14. S.Chen and h.Hongxiang , "Stock Prediction Using Convolutional Neural Network",2018. [Accessed April 21 2022].
15. S.Verma " Understanding 1D and 3D Convolutional Neural Network",[Online] Available : towardsdatascience.com. [Accessed April 22 2022].
16. H.Pedamallu,"RNN vs GRU vs LSTM" Nov 14, 2020.[Online].Available : medium.com. [Accessed April 22 2022].
17. Y.Verma *Complete Guide to Bidirectional LSTM* july 17 2021.[Accessed April 22 2022].
18. C.Moler ,"What is Deep Learning?".[Online].Available: mathworks.com . [Accessed April 22 2022].
19. J.H.Mohammad , A.Y.Owda "A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms,".[Accessed April 23 2022].
20. S.O.Birim , "An analysis for cryptocurrency price prediction using LSTM, GRU, and the Bi-LSTM implications,".[Accessed April 23 2022].
21. S.Yang, " Network Behavior Anomaly Analysis Based on Bidirectional LSTM," proceedings of the IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 15–17 March 2019.
22. R.Jozefowicz , W.Zaremba , I.Sutskever "An empirical exploration of Recurrent Network architectures" presented at 32nd International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015; Vol.3, pp. 2332–2340.
23. R.Zhao , D.Wang, R.Yan, K.Mao, F.Shen, J.Wang, "Machine Health Monitoring Using Local Feature-Based Gated Recurrent Unit Networks." I*EEE Trans. Ind. Electron.* , vol.65, pp.1539–1548,2018.[Accessed April 24 2022].
24. H.Sebastião ,P.Godinho , "Forecasting and trading cryptocurrencies with machine learning under changing market conditions" ,6 Jan, 2021.
25. K.Hsiang, S.Huang "Predicting Cryptocurrency Price With Tensorflow and Keras" , Jan 1, 2018.