BITCOIN PRICE PREDICTION USING LSTM

A Project Report

Submitted to the Faculty of Engineering of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA, KAKINADA

In partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

P. SRI TIRUMALA (18481A05G3)

M. YASWANTH REDDY (18481A05D4)

P. HEMANTH VENKATESH (18481A05G3)

M. SARAH CHAITANYA (18481A05E3)

Under the guidance of

Mrs. D. RAGAVAMSI, M.Tech

Assistant Professor of CSE Department

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESHADRIRAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

SESHADRIRAO KNOWLEDGE VILLAGE

GUDLAVALLERU – 521356

ANDHRA PRADESH

2021-2022

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
SESHADRI RAO KNOWLEDGE VILLAGE, GUDLAVALLERU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "BITCOIN PRICE PREDICTION USING LSTM" is a bonafide record of work carried out by P. SRI TIRUMALA (18481A05G3), M. YASWANTH REDDY (18481A05D4), P. HEMANTH VENKATESH (18481A05H0), M. SARAH CHAITANYA (18481A05E3) under the guidance and super vision in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Jawaharlal Nehru Technological University Kakinada, Kakinada during the academic year 2021-22.

Project Guide
(Mrs. D RagaVamsi)

Head of the Department
(Dr. M. BABU RAO)

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragements crown all the efforts with success.

We would like to express our deep sense of gratitude and sincere thanks to **Mrs, D. Raga Vamsi,** Assistant Professor, Department of Computer Science and Engineering for her constant guidance, supervision and motivation in completing the project work.

We feel elated to express our floral gratitude and sincere thanks to **Dr. M. Babu Rao,** Head of the Department, Computer Science and Engineering for his encouragements all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the project work.

We would like to take this opportunity to thank our beloved principal Dr. G.V.S.N.R.V Prasad for providing a great support for us in completing our project and giving us the opportunity for doing project.

Our Special thanks to the faculty of our department and programmers of our computer lab. Finally, we thank our family members, non-teaching staff and our friends, who had directly or indirectly helped and supported us in completing our project in time.

Team members

P. Sri Tirumala (18481A05G3)

M.Yaswanth Reddy (18481A05D4)

P. Hemanth Venkatesh (18481A05H0)

M.Sarah Chaitanya (18481A05E3)

CONTENTS

ABSTRACT

Bitcoin is a very volatile cryptocurrency that is becoming increasingly popular. It is a cryptographic protocol-based digital currency. It is an unregulated kind of electronic cash that can be used for online transactions or as a medium of exchange between any two parties and is purely electronic cash in a peer-to-peer format. Bitcoin is a particularly volatile currency since it has no central regulatory authority and is governed by the whole population. Its price is influenced by socially generated ideas. Hence Bitcoin's total market value is affected in some way, either directly or indirectly by Twitter attitude about Bitcoin. Cryptocurrency investors frequently consider which cryptocurrency to invest in. Cryptocurrency price prediction is critical for this purpose. The LSTM model can estimate the value of pricing data because it is time series data. For the best results, socially generated opinions (tweets data) are combined with bitcoin price data.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CC	_	Cryptocurrency
BTC	_	Bitcoin
AR	_	Auto Regressive
MA	–	Moving Averages
ARIMA	_	Auto Regressive Integrated Moving Averages
SVM	_	Support Vector Machine
NLP	_	Natural Language Processing
TF – IDF	_	Term Frequency - Inverse Document Frequency
RNN	_	Recurrent Neural Networks
LSTM	_	Long Short Term Memory
STM	_	Short Term Memory
LTM	_	Long Term Memory
MSE	_	Mean Squared Error
RMSE	_	Root Mean Square Error

CHAPTER 1
INTRODUCTION

1.1 Introduction

Cryptocurrency (CC) can be defined as any medium of exchange, apart from real world money, that can be used in many financial transactions whether they are virtual or real transactions. Cryptocurrency is a digital representation of valuable and intangible assets that may be used in a wide range of applications and networks, including as social networking websites, social media games, virtual worlds, and peer-to-peer networks. There are 16,598 cryptocurrencies in circulation as of January 2022.

The total trading volume of all cryptocurrencies is currently $75 billion per 24 hours. With the "dead" cryptos removed, there are roughly 9,631 active cryptocurrencies, which is a significant quantity. Bitcoin is one of the them which is is a decentralized digital currency created in January 2009. It follows the ideas set out in a white paper by the mysterious and pseudonymous Satoshi Nakamoto.12 The identity of the person or persons who created the technology is still a mystery. Bitcoin offers the promise of lower transaction fees than traditional online payment mechanisms do, and unlike government-issued currencies, it is operated by a decentralized authority. Cryptocurrency uses cryptography to keep it secure. There are no physical bitcoins, only balances kept on a public ledger that everyone has transparent access to (although each record is encrypted). All Bitcoin transactions are verified by a massive amount of computing power via a process known as "mining." Bitcoin is not issued or backed by any banks or governments, nor is an individual bitcoin valuable as a commodity. Despite it not being legal tender in most parts of the world, Bitcoin is very popular and has triggered the launch of hundreds of other cryptocurrencies, collectively referred to as altcoins. Bitcoin is commonly abbreviated as BTC when traded.

With so many cryptocurrencies accessible, one may wonder which one to invest in. Having an overview and understanding of bitcoin future values is really beneficial in this case. Bitcoin's price is forecasted using a variety of time series forecasting models. Time series forecasting methods can be divided into two categories named parametric methods like AR, MA, ARMA, ARIMA models and non-parametric models such as non parametric regression , neural network prediction, support vector machine (SVM) and the combination of these algorithms. Parametric methods won't achieve good performance when time series data shows irregular variations. Both bitcoin price data and tweet data about bitcoin come under time series data. Using the mood of users' tweets about bitcoin to predict the price can greatly improve the model's accuracy. The LSTM model is being used to forecast the bitcoin price in this case.

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning (DL). Unlike standard feedforward neural networks, LSTM has feedback connections. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction. LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time. By treating the hidden layer as a memory unit, The LSTM network can deal with short and long-term correlation within time series, making it capable of learning order dependence in sequence prediction tasks. In the conventional feed-forward neural networks, all test cases are considered to be independent. That is when fitting the model for a particular day, there is no consideration for the Bitcoin prices on the previous days.

This dependency on time is achieved via Recurrent Neural Networks.In LSTMs, information is transmitted using a mechanism known as cell states.

1.2 Objectives of the Project:

Crypto become popular since last five years that it became accepted by so many financial organizations. Although machine learning has been successful in predicting stock market prices through a host of different time series models, its application in predicting cryptocurrency prices has been quite restrictive. The reason behind this is obvious as prices of cryptocurrencies depend on a lot of factors like technological progress, internal competition, pressure on the markets to deliver, economic problems, security issues, political factor etc.

Their high volatility leads to the great potential of high profit if intelligent inventing strategies are taken. Unfortunately, due to their lack of indexes, cryptocurrencies are relatively unpredictable compared to traditional financial predictions like stock market prediction. A time series is a data set that tracks a sample over time. In particular, a time series allows one to see what factors influence certain variables from period to period. Time series analysis can be useful to see how a given asset, security, or economic variable changes over time. It can also be used to examine how the changes associated with the chosen data point compare to shifts in other variables over the same time period.

Forecasting methods using time series are used in both fundamental and technical analysis.This project aims to predict the price of one of the cryptocurrency “Bitcoin” since it was the best-known cryptocurrency and has been up nearly 70 percent since the start of 2021, driving the entire crypto market to a combined $2 trillion in value.

1.3 Problem Statement:

In this project for bitcoin price prediction, we are considering the tweets sentiment towards BTC as a feature so that the accuracy will increase. This mainly aims to

i. Get historical cryptocurrency data

ii. Prepare data for training and testing.
iii. Get Tweets about BTC,Bitcoin
iv. Preprocess tweets data
v. Predict the price of crptocurrency using LSTM neural network.
vi. Visualize the prediction results.

CHAPTER 2
LITERATURE REVIEW

Bitcoin:

The idea that "A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution." lead to the invention of BTC. We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

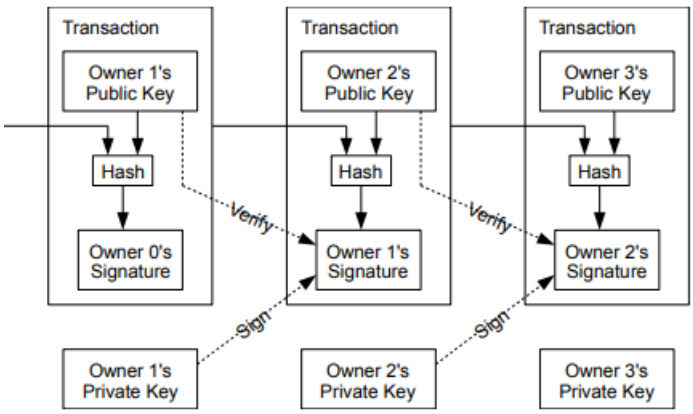


Figure 2.1: Bitcoin Transactions

The problem of course is the payee can't verify that one of the owners did not double-spend

the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. The problem with this solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank.

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced, and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

The solution proposed for this begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

Bitcoin transactions works in below way:

1) New transactions are broadcast to all nodes.

2) Each node collects new transactions into a block.

3) Each node works on finding a difficult proof-of-work for its block.

4) When a node finds a proof-of-work, it broadcasts the block to all nodes.

5) Nodes accept the block only if all transactions in it are valid and not already spent.

6) Nodes express their acceptance of the block by working on creating the next block in the

chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In

that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.

LSTM:

Data of a set of variables collected over time is called time series data. Then What constitutes a time series? A simple additive dissected version will look like this, without digging into too much notation:

$$xt = mt + st + et$$

where:

mt is the trend

st is the seasonality

et is the error or random white noise

Time series data is used either for forecasting the future data or analysing the historical data. Sequence prediction problems have been around for a long time. They are considered as one of the hardest problems to solve. These include a wide range of problems; from predicting sales to finding patterns in stock markets' data.

To put it simply, the components of a time series model are defined as:

Trend : The climbing or dropping data in a series are referred to as a trend.

Seasonality : It is defined as the series' repeated short-term cycles.

Arbitrary white noise - a series of random variations

With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long short

Term Memory networks, a.k.a LSTMs have been observed as the most effective solution among the others like AR,MA,ARMA, ARIMA and RNN models to name a few.

RNN:

A recurrent neural network (RNN) is a special type of an artificial neural network adapted to work for time series data or data that involves sequences. Ordinary feed forward neural networks are only meant for data points, which are independent of each other. 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. Here X represents the data at time particular timestamps.

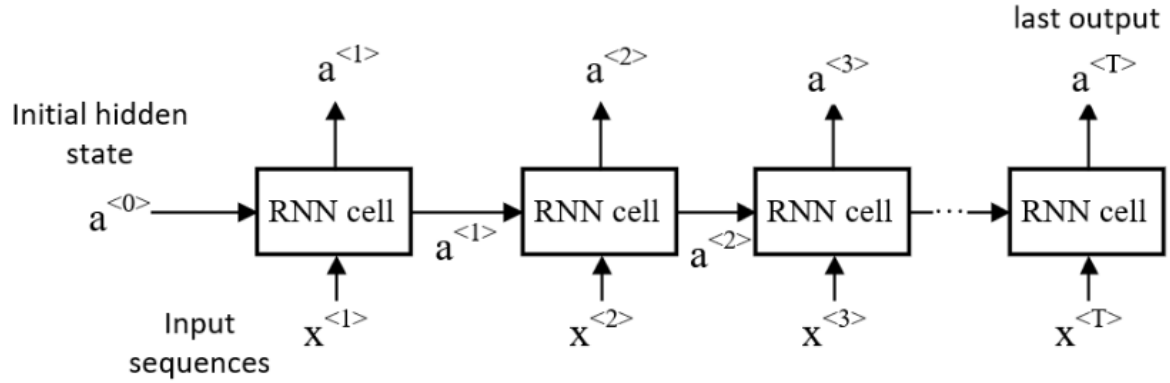


Figure 2.2: Architecture of RNN

The input layer ‘x’ takes in the input to the neural network and processes it and passes it onto the middle layer.

The middle layer ‘h’ can consist of multiple hidden layers, each with its own activation functions and weights and biases. If you have a neural network where the various parameters of different hidden layers are not affected by the previous layer, ie: the neural network does not have memory, then you can use a recurrent neural network.

The Recurrent Neural Network will standardize the different activation functions and weights and biases so that each hidden layer has the same parameters. Then, instead of creating multiple hidden layers, it will create one and loop over it as many times as required. In a normal feed-forward neural

network, the decisions are based on the current input. It doesn't memorize the past data, and there's no future scope. Feed-forward neural networks are used in general regression and classification problems.

In case of a forward propagation in RNNs, the inputs enter and move forward at each time step. In case of a backward propagation in this case, we are figuratively going back in time to change the weights, hence we call it the Back propagation through time(BPTT).

Gradient descent (GD) is an iterative first-order optimisation algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning(DL) to minimise a cost/loss function (e.g. in a linear regression).Gradient descent algorithm does not work for all functions. There are two specific requirements. A function has to be:

- differentiable
- convex

If a function is differentiable it has a derivative for each point in its domain — not all functions meet these criteria. Next requirement — function has to be convex. For a univariate function, this means that the line segment connecting two function's points lays on or above its curve (it does not cross it). If it does it means that it has a local minimum which is not a global one.

DRAWBACKS OF RNN:

Vanishing Gradients:

- Backpropagation finds the derivatives of the network by moving layer by layer from the final layer to the initial one. By the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers.

- When n hidden layers use an activation like the sigmoid function, n small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers.

Exploding gradients:

- Exploding gradients are a problem when large error gradients accumulate and result in very large updates to neural network model weights during training.

Sentiment Analysis:

Sentiment analysis is a vital topic in the field of NLP. It has easily become one of the hottest topics in the field because of its relevance and the number of business problems it is solving and has been able to answer. All of the tweets that have been retrieved have been preprocessed in order to retrieve the polarity and subjectivity. Each tweet (string data) is processed by the textblob function, which determines the polarity and subjectivity of each tweet. Create a new 'Sentiment' column with the resulting polarity and subjectivity in appropriate columns. On a given day, the polarity of all tweets is summed up, and a new column named polarity is formed and added to bitcoin price history data, resulting in the final dataset.

The results of sentiment analysis are divided into two groups: polarity and subjectivity. Polarity can have a value of -1 to +1. In this case, 0 denotes a neutral statement, -1 denotes a severely negative statement, and +1 denotes a highly positive one. Subjectivity, on the other hand, has a value ranging from 0 to 1. Here, 0 denotes a statement that is extremely objective, while +1 denotes a statement that is highly subjective.

Sentiment polarity is 0.5, and subjectivity is 0.6 in the above-mentioned result.

Because the polarity is 0.5, the statement is definitely positive, and the subjective score is 0.6, the statement is subjective.

One can use many techniques like Naïve Bayes classification, NLP in which Bag of words, Tokenization, Lemmatization are involved. For knowing the emotion of user one can build word clouds which contain the most popular words.

Essentially, sentiment analysis or sentiment classification fall into the broad category of text classification tasks where you are supplied with a phrase, or a list of phrases and your classifier is supposed to tell if the sentiment behind that is positive, negative or neutral. Sometimes, the third attribute is not taken to keep it a binary classification problem. Now, from a strict machine learning point of view, this task is nothing but a supervised learning task. You will supply a bunch of phrases (with the labels of their respective sentiments) to the machine learning model, and you will test the model on un labelled phrases.

The necessary steps include (but aren't limited to) the following:

i. Tokenizing sentences to break text down into sentences, words, or other units
ii. Removing stop words like "if," "but," "or," and so on
iii. Normalizing words by condensing all forms of a word into a single form
iv. Vectorizing text by turning the text into a numerical representation for consumption by your classifier

Bag of Words:

A bag-of-words representation of a document does not only contain specific words but all the unique words in a document and their frequencies of occurrences. A bag is a mathematical set here, so by the definition of a set, the bag does not contain any duplicate words.

Several approaches have been developed for converting text to numbers. Bag of Words, N-grams, and Word2Vec model are some of them.

In a simple bag of words, every word is given equal importance. The idea behind TF-IDF is that the words that occur more frequently in one document

and less frequently in other documents should be given more importance as they are more useful for classification.

TF-IDF :

It is a product of two terms: TF and IDF.Term Frequency is equal to the number of times a word occurs in a specific document. It is calculated as:

TF = (Frequency of a word in the document)/(Total words in the document)

Inverse Document Frequency for a specific word is equal to the total number of documents, divided by the number of documents that contain that specific word. The log of the whole term is calculated to reduce the impact of the division. It is calculated as:

IDF = Log((Total number of docs)/(Number of docs containing the word))

Corpus:

The words that you found out in the bag-of-words will now construct the feature set of your document. So, consider you a collection of many movie reviews (documents), and you have created bag-of-words representations for each one of them and preserved their labels (i.e., sentiments - +ve or -ve in this case). Your training set should look like:

document	w1	w2	w3	w4	. . .	wn	sentiment
d1	2	1	3	1		1	positive
d2	1	5	5	5		1	negative
d3	3	8	6	8		2	positive
d4	2	5	1	5		3	positive
d5	3	0	3	0		0	negative
d6	0	0	0	0		0	negative
d7	2	0	0	0		0	positive
d8	9	2	9	2		2	negative

Figure 2.4: Corpus of words

Tokenization is the process of breaking down chunks of text into smaller pieces. spaCy comes with a default processing pipeline that begins with tokenization,

making this process a snap. In spaCy python library one can do either sentence tokenization or word tokenization:

i. Word tokenization breaks text down into individual words.
ii. Sentence tokenization breaks text down into individual sentences.

Stop words are words that may be important in human communication but are of little value for machines. spaCy comes with a default list of stop words that you can customize.

Normalization is a little more complex than tokenization. It entails condensing all forms of a word into a single representation of that word. For instance, “watched,” “watching,” and “watches” can all be normalized into “watch.” There are two major normalization methods:

i. Stemming
ii. Lemmatization

With stemming, a word is cut off at its stem, the smallest unit of that word from which you can create the descendant words. You just saw an example of this above with “watch.” Stemming simply truncates the string using common endings, so it will miss the relationship between “feel” and “felt,” for example.

Lemmatization seeks to address this issue. This process uses a data structure that relates all forms of a word back to its simplest form, or lemma. Because lemmatization is generally more powerful than stemming.

Vectorization is a process that transforms a token into a vector, or a numeric array that, in the context of NLP, is unique to and represents various features of a token. Vectors are used under the hood to find word similarities, classify text, and perform other NLP operations. This particular representation is a dense array, one in which there are defined values for every space in the array. This is in opposition to earlier methods that used sparse arrays, in which most spaces are empty.

The next step is classification of tweets. Many methods and packages are available for this task.

CHAPTER 3
PROPOSED METHOD

3.1 Methodology:

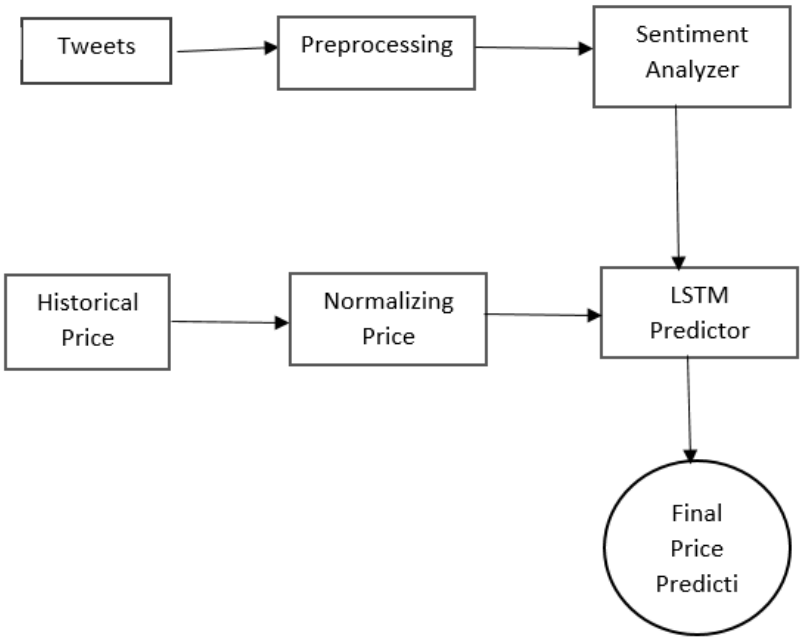


Figure 3.1: Flowchart of project

The project involves collecting tweets from Twitter API, Bitcoin prices from several REST APIs. As BTC is not controlled by any central organization like bank or government and is dependent on socially constructed opinions, considering twitter data about BTC is crucial.

Twitter is a social networking site on which a wide range of information is shared across world like business news, any crisis to name a few.

Above Flowchart indicated that we are going to fetch tweets data first and it undergoes pre processing then on sentiment analysis , corresponding polarity is retrieved which is saved as a vector. Here as the twitter data is unlabelled data, we are not going for supervised learning models such as SVM,Naïve Bayes models and Neural Networks with NLP .On the other hand, BTC historic price is fetched from APIs like Coinbase Pro API etc. Bitcoin prices are

normalized for the benefit of maximizing the accuracy by easing the training process. This also created a feature vector which then merged with the previous vector.

Then this final feature vector is used for training the LSTM network. Due to maintaining a cell state, LSTM can avoid long term dependency problems which arises in RNN.

Advanced RNN(LSTM):

The functioning of LSTM can be visualized by understanding the functioning of three actions. For example to understand something one should store important information, forget unnecessary information. Now all these broken pieces of information cannot be served on directly. So, after a certain time interval, you need to summarize this information and output the relevant things only.

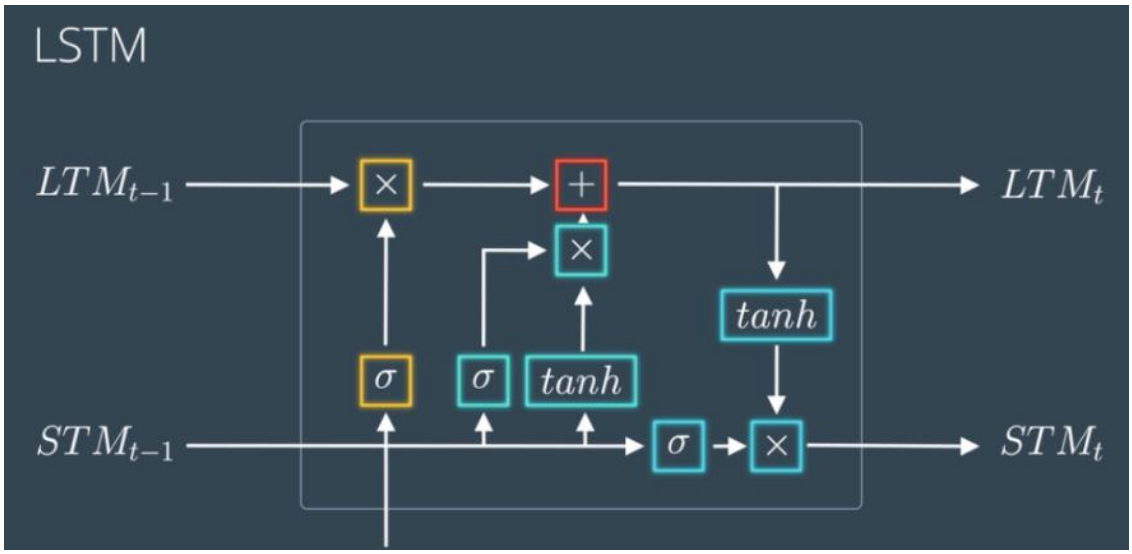


Figure 3.2: Architecture of LSTM

i. Learn Gate: Takes Event (Et) and Previous Short Term Memory (STMt-1) as input and keeps only relevant information for prediction.
   a. Previous Short Term Memory STMt-1 and Current Event vector Et are joined together [STMt-1, Et] and multiplied with the weight matrix Wn having some bias which is then passed to tanh (

hyperbolic Tangent) function to introduce non-linearity to it, and finally creates a matrix Nt.

b. For ignoring insignificant information we calculate one Ignore Factor it, for which we join Short Term Memory STMt-1 and Current Event vector Et and multiply with weight matrix Wi and passed through Sigmoid activation function with some bias.
c. Learn Matrix Nt and Ignore Factor it is multiplied together to produce learn gate result.

ii. The Forget Gate: Takes Previous Long Term Memory (LTMt-1) as input and decides on which information should be kept and which to forget.
    a. Previous Short Term Memory STMt-1 and Current Event vector Et are joined together [STMt-1, Et] and multiplied with the weight matrix Wf and passed through the Sigmoid activation function with some bias to form Forget Factor ft.
    b. Forget Factor ft is then multiplied with the Previous Long Term Memory (LTMt-1) to produce forget gate output.

iii. The Remember Gate: Combine Previous Short Term Memory (STMt-1) and Current Event (Et) to produce output.

Training LSTMs removes the problem of Vanishing Gradient (weights become too small that under-fits the model), but it still faces the issue of Exploding Gradient (weights become too large that over-fits the model). Training of LSTMs can be easily done using Python frameworks like Tensorflow, Pytorch, Theano, etc. and the catch is the same as RNN, we would need GPU for training deeper LSTM Networks.

Since LSTMs take care of the long term dependencies its widely used in tasks like Language Generation, Voice Recognition, Image OCR Models, etc. Also, this technique is getting noticed in Object Detection also.

3.2 Implementation:

Because parametric models for forecasting time series data, such as AR, MA, and ARIMA, require input data to be stationary, we prefer neural networks and use LSTM (Long Short Term Memory), an advanced RNN that can manage long-term dependencies. Unlike RNNs, which have vanishing and exploding gradients, LSTMs keep a cell state that acts as long-term memory and feature three gates that allow the network to pick which information to keep and which to discard. We are using our own dataset, which was created by combining bitcoin price data with tweet sentiment scores over a one-year period. The Long Short Term Memory (LSTM) model is trained using this data.

Bitcoin Price data is collected using CoinBasePro API, which is fetched for a time range in csv format and stored. Coinbase Pro's fully featured trading API offers an efficient way for developers to build digital asset trading applications and to offer related functionality. Coinbase provides a simple and powerful REST API to integrate bitcoin, bitcoin cash, litecoin and ethereum payments into your business or application. This API reference provides information on available endpoints and how to interact with it.

	low	high	open	close	volume
0	33352.54	37000.00	34043.91	36859.26	45744.10320
1	36200.00	40425.00	36859.26	39505.56	50346.30569
2	36565.08	41986.37	39510.55	40665.15	48522.48490
3	38800.00	41406.94	40642.15	40257.43	27152.97103
4	34444.00	41452.12	40257.43	38171.57	43736.57032
...	...	...	...	...	...
355	50480.00	52100.00	50804.33	50717.77	11596.91468
356	47300.23	50720.35	50720.35	47543.09	23116.75584
357	46094.02	48149.58	47542.20	46471.24	20326.32142
358	45938.44	47926.15	46471.24	47122.08	27413.65365
359	45650.00	48574.70	47122.09	46211.24	19010.24135

Figure 3.3: BTC price dataset

Twitter data is collected using snscrape (social network scraping library) which is also fetched between a time period. snscrape is a scraper for social networking services (SNS). It scrapes things like user profiles, hashtags, or searches and returns the discovered items, e.g. the relevant posts. Released on July 8, 2020, snscrape is a scraping tool for social networking services (SNS). It scrapes things like users, user profiles, hashtags, searches, threads, list posts and returns the discovered items without using Twitter's API.

Interestingly, snscrape is not just for scraping tweets but also across various other social networking sites like Facebook, Instagram, Reddit, VKontakte, and Weibo (Sina Weibo).

The following services are currently supported:

i. Facebook: user profiles, groups, and communities (aka visitor posts)
ii. Instagram: user profiles, hashtags, and locations
iii. Mastodon: user profiles and toots (single or thread)
iv. Reddit: users, subreddits, and searches (via Pushshift)
v. Telegram: channels
vi. Twitter: users, user profiles, hashtags, searches, tweets (single or surrounding thread), list posts, and trends
vii. VKontakte: user profiles
viii. Weibo (Sina Weibo): user profiles

To get intended data, the tweets were scraped using 'bitcoin' keyword

3.2.2 Normalization:

Because unscaled input variables could produce a lethargic or inaccurate learning process, all of the Bitcoin value data is normalized to ensure that the quantities are within the range [0,1] . To help the LSTM model to converge faster it is important to scale the data. It is possible that large values in the inputs slow down the learning. We are going to use StandardScaler from sklearn library to scale the data. The scaler is fit on the training set and it is used to transform the unseen trade data on validation and test set. If we would fit the

scalar on all data, the model would overfit and it would achieve good results on this data, but performance would suffer on the real-world data.

```
scaler_train = MinMaxScaler(feature_range=(0, 1))
scaled_train = scaler_train.fit_transform(df_train.reshape(-1,1))
scaled_train = scaled_train.reshape((300,6))
scaler_test = MinMaxScaler(feature_range=(0, 1))
scaled_test = scaler_test.fit_transform(df_test.reshape(-1,1))
scaled_test = scaled_test.reshape((60,6))
```

3.3 Data Preparation:

3.3.1 Pre-processing Tweets:

All of the tweets that have been retrieved have been preprocessed in order to retrieve the polarity and subjectivity. Each tweet (string data) is processed by the textblob function, which determines the polarity and subjectivity of each tweet. Create a new 'Sentiment' column with the resulting polarity and subjectivity in appropriate columns. On a given day, the polarity of all tweets is summed up, and a new column named polarity is formed and added to bitcoin price history data, resulting in the final dataset.

The probabilities of positive and negative statements are illustrated in table below.

• text1=TextBlob("Bitcoin price is increasing")

• print(format(text1.sentiment))

• Sentiment(polarity=0.5, subjectivity=0.6)

The results are divided into two groups: polarity and subjectivity. Polarity can have a value of -1 to +1. In this case, 0 denotes a neutral statement, -1 denotes a severely negative statement, and +1 denotes a highly positive one. Subjectivity, on the other hand, has a value ranging from 0 to 1. Here, 0 denotes a statement that is extremely objective, while +1 denotes a statement that is highly subjective.

Sentiment polarity is 0.5, and subjectivity is 0.6 in the above-mentioned result.

Because the polarity is 0.5, the statement is definitely positive, and the subjective score is 0.6, the statement is subjective.

Datetime	pol
2021-01-01	14.581429
2021-01-02	33.728441
2021-01-03	22.087978
2021-01-04	9.599033
2021-01-05	17.906313

Figure 3.4: Tweets polarity table

On merging both the datasets, we got a new feature vector is that is ready for training on LSTM. Pandas has full-featured, high performance in-memory join operations idiomatically very similar to relational databases like SQL.

Pandas provides a single function, merge, as the entry point for all standard database join operations between DataFrame objects. It provides a huge range of methods and functions to manipulate data, including merging DataFrames. Merging DataFrames allows you to both create a new DataFrame without modifying the original data source or alter the original data source. The how argument to merge specifies how to determine which keys are to be included in the resulting table. If a key combination does not appear in either the left or the right tables, the values in the joined table will be NA. Joining will be performed on index. Join operation honors the object on which it is called. So, a.join(b) is not equal to b.join(a).

Here, we have used the following parameters −

- left − A DataFrame object.
- right − Another DataFrame object.

- on − Columns (names) to join on. Must be found in both the left and right DataFrame objects.
- left_on − Columns from the left DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.
- right_on − Columns from the right DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.
- left_index − If True, use the index (row labels) from the left DataFrame as its join key(s). In case of a DataFrame with a MultiIndex (hierarchical), the number of levels must match the number of join keys from the right DataFrame.
- right_index − Same usage as left_index for the right DataFrame.
- how − One of 'left', 'right', 'outer', 'inner'. Defaults to inner. Each method has been described below.
- sort − Sort the result DataFrame by the join keys in lexicographical order. Defaults to True, setting to False will improve the performance substantially in many cases.

	low	high	open	close	volume	pol
0	33352.54	37000.00	34043.91	36859.26	45744.10320	39.697416
1	36200.00	40425.00	36859.26	39505.56	50346.30569	42.358659
2	36565.08	41986.37	39510.55	40665.15	48522.48490	16.940548
3	38800.00	41406.94	40642.15	40257.43	27152.97103	12.754143
4	34444.00	41452.12	40257.43	38171.57	43736.57032	8.077794
...	...	...	...	...	...	...
355	50480.00	52100.00	50804.33	50717.77	11596.91468	18.403056
356	47300.23	50720.35	50720.35	47543.09	23116.75584	19.171833
357	46094.02	48149.58	47542.20	46471.24	20326.32142	16.132080
358	45938.44	47926.15	46471.24	47122.08	27413.65365	11.595228
359	45650.00	48574.70	47122.09	46211.24	19010.24135	8.413339

Figure 3.3: Merged dataset(Bitcoin price data&Polarity data)

The time series data must be divided into samples, each with its own input and output elements. After scaling we need to transform the data into a format that is appropriate for modeling with LSTM. We transform the long sequence of data into many shorter sequences (7-time bars per sequence) that are shifted by a single time bar.For instance, take the following sequence:

11, 12, 13, 14, 15, 16, 17, 18, 19, 20

Then the samples for training the model will look like:

Input	Output
11, 12, 13, 14, 15	16
12, 13, 14, 15, 16	17
13, 14, 15, 16, 17	18

To create the dataset, we used 7 as the lookback period(sequence leangth). Let's focus on the 1st sequence. The model takes the feature of the time bar at index 0 and it tries to predict the target of the time bar at index 1. Then it takes the feature of the time bar at index 1 and it tries to predict the target of the time bar at index 2, etc. The feature of 2nd sequence is shifted by a 1-time bar from the feature of 1st sequence, the feature of the 3rd sequence is shifted by a 1-time bar from 2nd sequence, etc. With this procedure, we get many shorter sequences that are shifted by a single time bar.

```
import numpy as np
def dataset_generator_lstm(dataset, look_back):
        # A "lookback period" defines the window-size of how many
         # previous timesteps are used in order to predict
        # the subsequent timestep.
        dataX, dataY = [], []

        for i in range(len(dataset) - look_back):
         window_size_x = dataset[i:(i + look_back)]#, 0]
        dataX.append(window_size_x)
```

dataY.append(dataset[i + look_back, 3]) # this is the label or actual y-value

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

Note that in classification or regression tasks, we usually have a set of features and a target that we are trying to predict. In this example with LSTM, the feature and the target are from the same sequence, the only difference is that the target is shifted by a 1-time bar.

regressor = Sequential()

regressor.add(LSTM(units = 128, activation = 'relu',return_sequences=True, input_shape = (trainX.shape[1], trainX.shape[2])))

regressor.add(Dropout(0.2))

 # Adding a second LSTM layer and some Dropout regularisation

regressor.add(LSTM(units = 64, input_shape = (trainX.shape[1], trainX.shape[2])))

regressor.add(LSTM(units = 64, return_sequences = True, input_shape = (trainX.shape[1], trainX.shape[2])))

regressor.add(Dropout(0.2))

regressor.add(Dense(units = 1))

regressor.add(Dense(units = 1))

regressor.summary()

3.3.2 Train-Test Split:

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and

situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

i. Train Dataset: Used to fit the machine learning model.
ii. Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

3.3.2.1 When to use train-test split:

The idea of "sufficiently large" is specific to each predictive modeling problem. It means that there is enough data to split the dataset into train and test datasets and each of the train and test datasets are suitable representations of the problem domain. This requires that the original dataset is also a suitable representation of the problem domain.

A suitable representation of the problem domain means that there are enough records to cover all common cases and most uncommon cases in the domain. This might mean combinations of input variables observed in practice. It might require thousands, hundreds of thousands, or millions of examples.

Conversely, the train-test procedure is not appropriate when the dataset available is small. The reason is that when the dataset is split into train and test sets, there will not be enough data in the training dataset for the model to learn an effective mapping of inputs to outputs. There will also not be enough data in

the test set to effectively evaluate the model performance. The estimated performance could be overly optimistic (good) or overly pessimistic (bad).

If you have insufficient data, then a suitable alternate model evaluation procedure would be the k-fold cross-validation procedure.

In addition to dataset size, another reason to use the train-test split evaluation procedure is computational efficiency.

Some models are very costly to train, and in that case, repeated evaluation used in other procedures is intractable. An example might be deep neural network models. In this case, the train-test procedure is commonly used. The procedure has one main configuration parameter, which is the size of the train and test sets. This is most commonly expressed as a percentage between 0 and 1 for either the train or test datasets. For example, a training set with the size of 0.67 (67 percent) means that the remainder percentage 0.33 (33 percent) is assigned to the test set.

3.3.2.2 Model Evaluation:

Evaluation Matrices: These are tied to ML tasks. There are different matrices for supervised algorithms (classification and regression) and unsupervised algorithms. For example, the performance of classification of the binary class is measured using Accuracy, AUROC, Log-loss, and KS.

Evaluation Mechanism: Model selection refers to the process of selecting the right model that fits the data. This is done using test evaluation matrices. The results from the test data are passed back to the hyper-parameter tuner to get the most optimal hyperparameters.

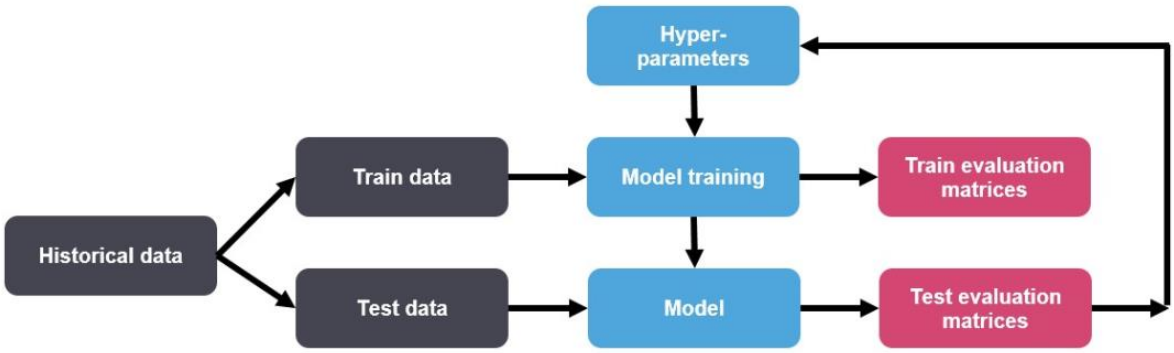


Figure 3.6 Model Evaluation mechanism

Hyper-parameters control the over-fitting and under-fitting of the model. Optimal hyper-parameters often differ for different datasets. To get the best hyper-parameters the following steps are followed:

i. For each proposed hyper-parameter setting the model is evaluated
ii. The hyper-parameters that give the best model are selected.

Grid search picks out a grid of hyper-parameter values and evaluates all of them. Guesswork is necessary to specify the min and max values for each hyper-parameter. Random search randomly values a random sample of points on the grid. It is more efficient than grid search. Smart hyper-parameter tuning picks a few hyper-parameter settings, evaluates the validation matrices, adjusts the hyper-parameters, and re-evaluates the validation matrices. Examples of smart hyper-parameter are Spearmint (hyper-parameter optimization using Gaussian processes) and Hyperopt (hyper-parameter optimization using Tree-based estimators).

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Results:

We used ‘relu’ as the activation function and ‘adam’ as the optimization algorithm for training the LSTM network. The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. An activation function is basically just a simple function that transforms its inputs into outputs that have a certain range. There are various types of activation functions that perform this task in a different manner, For example, the sigmoid activation function takes input and maps the resulting values in between 0 to 1.

One of the reasons that this function is added into an artificial neural network in order to help the network learn complex patterns in the data. These functions introduce nonlinear real-world properties to artificial neural networks. Basically, in a simple neural network, x is defined as inputs, w weights, and we pass f (x) that is the value passed to the output of the network. This will then be the final output or the input of another layer.

If the activation function is not applied, the output signal becomes a simple linear function. A neural network without activation function will act as a linear regression with limited learning power. But we also want our neural network to learn non-linear states as we give it complex real-world information such as image, video, text, and sound.

Relu is defined as :

$$f(x)=\max(0,x)$$

ReLU function is its derivative both are monotonic. The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. Thus it gives an output that has a range from 0 to infinity. As we have seen above, the ReLU function is simple and it consists of no heavy computation as there is no complicated math. The model can, therefore, take

less time to train or run. One more important property that we consider the advantage of using ReLU activation function is sparsity.

Since ReLU gives output zero for all negative inputs, it's likely for any given unit to not activate at all which causes the network to be sparse. Now let us see how ReLu activation function is better than previously famous activation functions such as sigmoid and tanh. he activations functions that were used mostly before ReLU such as sigmoid or tanh activation function saturated. This means that large values snap to 1.0 and small values snap to -1 or 0 for tanh and sigmoid respectively. Further, the functions are only really sensitive to changes around their mid-point of their input, such as 0.5 for sigmoid and 0.0 for tanh. This caused them to have a problem called vanishing gradient problem. Let us briefly see what vanishing gradient problem is.

Coming "Adam" optimizer, it is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

On training with LSTM the results are as follows:

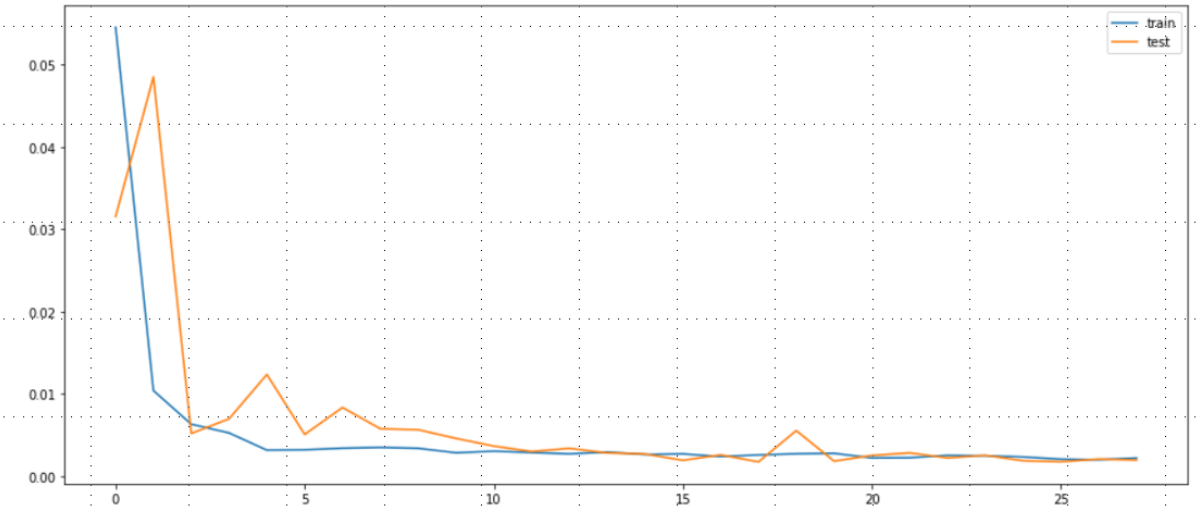


Figure 4.1: Validation loss and training loss

The training loss is a metric used to assess how a deep learning model fits the training data. That is to say, it assesses the error of the model on the training set.

Note that, the training set is a portion of a dataset used to initially train the model. Computationally, the training loss is calculated by taking the sum of errors for each example in the training set.

It is also important to note that the training loss is measured after each batch. This is usually visualized by plotting a curve of the training loss. One of the most widely used metrics combinations is training loss + validation loss over time. The training loss indicates how well the model is fitting the training data, while the validation loss indicates how well the model fits new data. On the contrary, 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 validation loss is similar to the training loss and is calculated from a sum of the errors for each example in the validation set. Additionally, the validation loss is measured after each epoch. This informs us as to whether the model needs further tuning or adjustments or not. To do this, we usually plot a learning curve for the validation loss.

In most deep learning projects, the training and validation loss is usually visualized together on a graph. The purpose of this is to diagnose the model's performance and identify which aspects need tuning. To explain this section, we'll use three different scenarios.

At times, the validation loss is greater than the training loss. This may indicate that the model is underfitting. Underfitting occurs when the model is unable to accurately model the training data, and hence generates large errors. When the model performs well on training data but poorly on the new data in the validation set. At a point, the validation loss decreases but starts to increase again.

A notable reason for this occurrence is that the model may be too complex for the data or that, the model was trained for a long period. In this case, training can be halted when the loss is low and stable, this is usually

known as early stopping. Early stopping is one of the many approaches used to prevent overfitting.

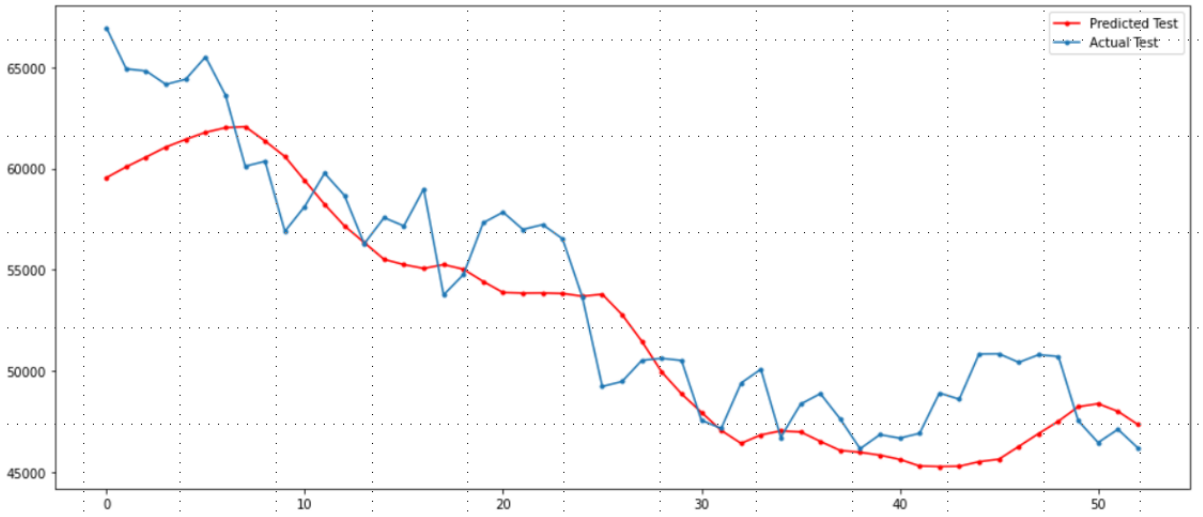


Figure 4.2: Prediction values vs Actual values

We have taken 60 as the testing set size and 305 as the training set size and used RMSE as the error calculation function.

4.1.1 Hyper-parameter Tuning:

Hyper-parameter optimization or tuning is the problem of choosing a set of optimal hyper-parameters for a learning algorithm. A hyper-parameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

Hyper-parameter types:

i. K in K-NN
ii. Regularization constant, kernel type, and constants in SVMs
iii. Number of layers, number of units per layer, regularization in neural network

Mean square error (**MSE**): It is the average squared loss per example over the whole dataset. To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

$$MSE=1/N\sum(x,y)\in D(y-prediction(x))2$$

Upon all the hyper parameter tuning such as loopback size and dropout value, we got Train RMSE as 3489.402 and Test RMSE as 2855.296.

Figure 4.3: Predicted Train vs Actual Train

Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called empirical risk minimization. When a model is trained on a network, the model can't fit the data exactly and it gives some deviation.This is called training loss. The training loss indicates how well the model is fitting the training data, while the validation loss indicates how well the model fits new data.

```
]:  1 plt.figure(figsize=(16,7))
    2
    3 plt.plot(predicted_btc_test_concatenated, 'r', marker='.', label='Predicted Test')
    4
    5 plt.plot(test_actual, marker='.', label='Actual Test')
    6
    7 plt.legend()
    8
    9 plt.show()
```

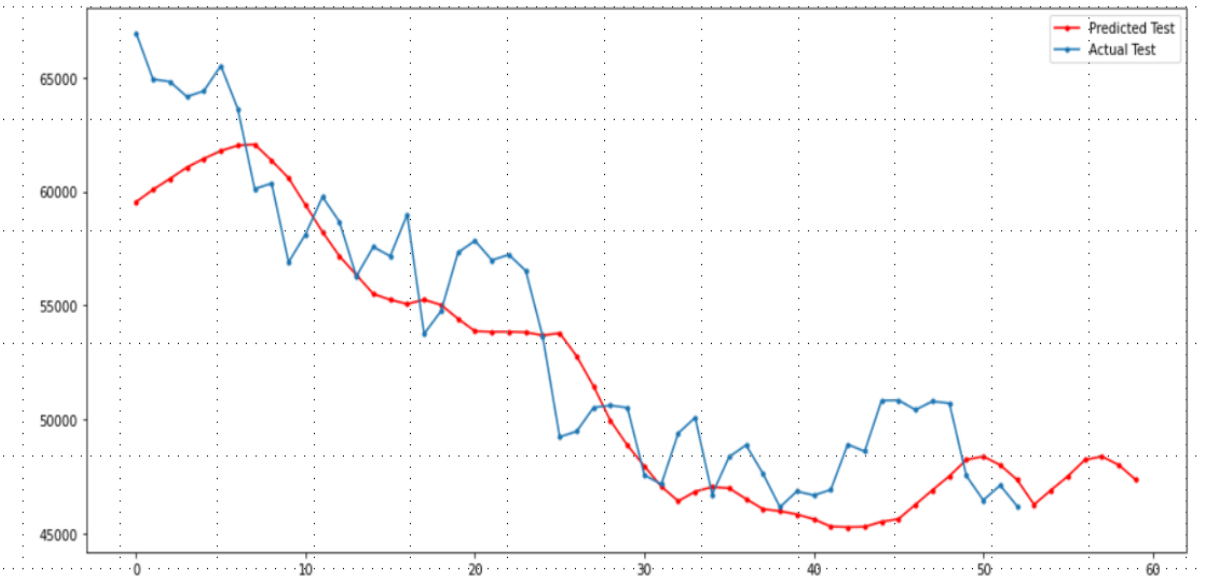


Figure 4.4: Prediction of next 7 days bitcoin price

Using the trained model, we predicted the price of bitcoin for next 7 days. Using this model one can get an idea of the bitcoin price and accordingly one can invest in the cryptocurreny.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

5.1 Conclusion:

Cryptocurrency prices are often said to be erratic and unpredictable to some extent. However, popular social media platforms like Twitter had been proven to be a good bellwether for Bitcoin prices. Several questions have been raised such as to what extent a myriad of voices can influence Bitcoin prices? Do the prices fluctuate according to influencers' tweets or is the market efficient enough to eliminate the effects of the tweets? Consequently, the first objective of this paper is to investigate the effects of tweets on bitcoin prices by using machine learning models to compute the sentiment scores. The influence of social media platform like Twitter had long been perceived as a bellwether of Bitcoin Prices. This paper investigated if the tweets can be modeled and used with NLP to compute the sentiment scores in order to predict the Bitcoin price signal. Through the experiments conducted, the LSTM model indicates some degree of predictive advantage compared to RNN. Using LSTM model over the historical data of 1 year from 2021 January to 2021 December, we got best results compared to RNN. The error values have decreased from 8788.126 to 2855.296 . we conclude that for data with random depth ,LSTM is suitable fit. Further considering the public opinion as a feature by including polarity column decreased the error value. This polarity value is evaluated by applying Textblob function on each tweet which is a simple analysis compared to NLP.

As the bitcoin price time series data is complex data and involves depending on many features , taking 7 days as frequency period(loop back period) and using LSTM resulted better predictions.

5.2 Future Scope:

Further the project could also be improved by sending users notifications when the bitcoin price reaches a certain level. This threshold (limit price) is initially set by the user i:e integrating with the mobile phone to get the notification on time.

It basically works in this way,

i. First the model continuously gets the tweets data from twitter and Bitcoin price data from APIs in real time
ii. And user sets a certain threshold that is used to notify him if the predicted price crosses that threshold.
iii. Then the fetched real time data is trained against the LSTM model to predict the price.
iv. Based on that price ,a real time notification is sent to the user.

Bibliography

1. Bitcoin: A Peer-to-Peer Electronic Cash System
2. The Growth of Cryptocurrency in India: Its Challenges & Potential Impacts on Legislation
3. LSTM network: a deep learning approach for short-term traffic forecast
4. https://bookdown.org/JakeEsprabens/431-Time-Series
5. https://bookdown.org/JakeEsprabens/431-Time-Series/modelling-time-series.html
6. https://towardsdatascience.com/lstm-for-time-series-prediction-de8aeb26f2ca
7. https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/
8. https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Program Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions., component, or software to meet the desired needs.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental

contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

PSO1 : Design, develop, test and maintain reliable software systems and intelligent systems.

PSO2 : Design and develop web sites, web apps and mobile apps.

PROJECT PROFORMA

	Application	Product	Research	Review
Classification of Project	√			

Note: Tick Appropriate category

Project Outcomes	
Course Outcome (CO1)	Identify and analyze the problem statement using prior technical knowledge in the domain of interest.
Course Outcome (CO2)	Design and develop engineering solutions to complex problems by employing systematic approach.
Course Outcome (CO3)	Examine ethical, environmental, legal and security issues during project implementation.
Course Outcome (CO4)	Prepare and present technical reports by utilizing different visualization tools and evaluation metrics.

Mapping Table

CS1537 : MAIN PROJECT														
Course Outcomes	Program Outcomes and Program Specific Outcome													
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12		PSO 1	PSO 2
CO1	**3**	3	1					2	2	2				1	1
CO2	**3**	3	3	3	3			2	2	2		1		3	3
CO3	**2**	2	3	2	2	3	3	3	2	2	2			3	
CO4	**2**		1		3				3	3	2	2		2	2

Note: **Map each project outcomes with POs and PSOs with either 1 or 2 or 3 based on level of mapping as follows:**

1-Slightly (Low) mapped 2-Moderately (Medium) mapped 3-Substantially (High) mapped

Stochastic Modeling & Applications ISSN: 0972-3641
Vol. 26 No. 3 (Special Issue 2022, Special Issue 2022 Part - 2) UGC CARE APPROVED JOURNAL

Received : 28th October Revised : 30th November 2021 Accepted: 15th December 2021

BITCOIN PRICE PREDICTION USING LSTM

Mrs.D. RagaVamsi[1], Parise Sri Tirumala[2], MandapatiYaswanth Reddy[3], PetetiHemanthVenkatesh[4], Motupalli Sarah Chaitanya[5]

1 M. Tech Assistant Professor, CSE Department & SR GEC, Gudlavalleru, Krishna
2Student, CSE Department &SR GEC, Gudlavalleru, Krishna
3Student, CSE Department &SR GEC, Gudlavalleru, Krishna
4Student, CSE Department &SR GEC, Gudlavalleru, Krishna
5Student, CSE Department &SR GEC, Gudlavalleru, Krishna

Abstract:

Bitcoin is a very volatile cryptocurrency that is becoming increasingly popular. It is a cryptographic protocol-based digital currency. It is an unregulated kind of electronic cash that can be used for online transactions or as a medium of exchange between any two parties and is purely electronic cash in a peer-to-peer format [1].Bitcoin is a particularly volatile currency since it has no central regulatory authority and is governed by the whole population. Its price is influenced by socially generated ideas. Hence Bitcoin's total market value is affected in some way, either directly or indirectly by Twitter attitude about Bitcoin. Cryptocurrency investors frequently consider which cryptocurrency to invest in.Cryptocurrency price prediction is critical for this purpose. The LSTM model can estimate the value of pricing data because it is time series data. For the best results, socially generated opinions (tweets data) are combined with bitcoin price data.

Introduction:

Cryptocurrency (CC)[2] can be defined as any medium of exchange, apart from real world money, that can be used in many financial transactions whether they are virtual or real transactions. Cryptocurrency is a digital representation of valuable and intangible assets that may be used in a wide range of applications and networks, including as social networking websites, social media games, virtual worlds, and peer-to-peer networks.There are 16,598 cryptocurrencies in circulation as of January 2022. The total trading volume of all cryptocurrencies is currently $75 billion per 24 hours. With the "dead" cryptos removed, there are roughly 9,631 active cryptocurrencies, which is a significant quantity. With so many cryptocurrencies accessible, one may wonder which one to invest in. Having an overview and understanding of bitcoin future values is really beneficial in this case. Bitcoin's price is forecasted using a variety of time series forecasting models.

Time series forecasting methods can be divided into two categories named parametric methods like AR[9],MA,ARMA,ARIMA models and non-parametric models such as nonparametric regression , neural network prediction, support vector machine (SVM) and the combination of these algorithms[3]. Parametric methods won’t achieve good performance when time series data shows irregular variations[3].

Both bitcoin price data and tweet data about bitcoin come under time series data. Using the mood of users' tweets about bitcoin to predict the price can greatly improve the model's accuracy. The LSTM model is being used to forecast the bitcoin price in this case.

LSTM (Long Short Term Memory) model:LSTM network is a special kind of RNN. By treating the hidden layer as a memory unit,The LSTM network can deal with short and long-term correlation within time series, making it capable of learning order dependence in sequence prediction tasks. In LSTMs, information is transmitted using a mechanism known as cell states. In this way, LSTMs can consciously recall or forget things. There are three types ofreliance on information at a given cell state.

These dependencies can be used to solve any problem in the following way:

1. The cell's former condition (i.e. the information that was present in the memory after the previous time step)
2. The previously unknown state (i.e. this is the same as the output of the previous cell)
3. The input for the current time step (i.e. the new information that is being fed in at that moment)

Sentiment analysis:

Sentiment analysis is the appraisal of a speaker's, writer's, or other subject's opinion on a given topic. Twitter is being utilised to collect a big and diversified data set that represents current popular sentiment on bitcoin. To assess public attitudes, the gathered tweets are analysed using a lexicon-based technique. We calculate the polarity and subjectivity metrics for the gathered tweets in this study, which aid in understanding user opinion on bitcoin price. For example,FarhaNausheen and SayyadaHajera Begum [3] employed sentiment analysis to forecast what would happen in US presidential election results 2016, in which Donald Trump, Hillary Clinton and Bernie Sanders were among the top election candidates. The opinion of the public for a candidate will impact the potential leader of the country.

Literature Survey (related work):

A univariate time series is defined as a collection of measurements of the same variable through time. The measurements are usually taken at regular intervals. One of the distinguishing features of a time series is that it is a collection of observations in which the order is important. Because there is reliance, ordering is crucial, and changing the order could change the meaning of the data.

Various models, such as AR, MA, ARIMA, and Neural Network-based models, can be used to analyse time series data.

1.AR (Auto-Regressive) model:

An autoregressive (AR) model predicts future behaviour based on historical data. It is useful for anticipating whether there is an association between the values in a series data and the values that come before and after them. The procedure entails a linear regression of the current series' data against one or more previous values from the same series.

Thevalue of the outcome variable (Y) at time t is closely attributed to the predictor variable in an AR model just as it is in "ordinary" linear regression (X). The main difference between simple linear regression and AR models is that Y is dependent on X and prior Y values. It has the following representation:

$$Yt = \beta_1 * y_{-1} + \beta_2 * y_{-2} + \beta_3 * y_{-3} + ... + \beta * y_{-}$$

Where Yt = forecasting variable at time period t

β_1 , β_2 ,.. are bias terms

2. MA(Moving Averages) model:

A moving average model(MA)[11] employs past forecast mistakes in a regression-like model, unlike the AR model, which uses past values of the predicted variable in a regression. The current value is linearly dependent on the current and previous error terms, according to this formula.

$$r_t = c + \theta_1 \epsilon_{t-1} + \epsilon_t$$

where r_t = forecasting variable at time t

c = constant factor

θ_1 =numeric coefficient for the value associated with the 1st lag.

ϵ_t = residual at time t

ARIMA(autoregressive integrated moving average) model:

ARIMA is a type of regression analysis that shows how strong a dependent variable is in relation to other variables that change. It's the result of combining AR and MA models.

The data must be stationary to apply all of the following models (AR,MA). The qualities of a stationary time series are independent of the time at which it is viewed. Time series containing patterns or seasonality, on the other hand, are not stationary. When there is evidence of non-stationarity in the data, ARIMA models are used. In time series analysis, non-stationary data is always processed into stationary data by differencing the data.

The ARIMA model is abbreviated as ARIMA (p, d, q), and the parameters p, d, and q are as follows:

- p: the lag order of the autoregressive model AR, or the number of temporal lags (p)
- d: the degree of differencing, or the number of times the data has been subtracted from a previous value.
- q: what is the order of the MA model of moving averages? (q)

Neural network based RNN model:

The most up-to-date sequential data algorithm is recurrent neural networks (RNN). It is the first algorithm with an internal memory that remembers its input, making it ideal for machine learning problems involving sequential data. This is why they're the algorithm of choice for series data, speech, text, financial records, audio, video, climate, and a variety of other sequential data types. The formula for the current state in RNN is

$$h_t = f(h_{t-1}, x_t)$$

$$h_t = tanh(W_{hh}\, h_{t-1} + W_{xh}\, X_{t-1})$$

where $h_t$is the output at time 't'

$W_{hh}$is the weightmatrix(kernel)associated with h_{t-1}

X_{t-1} is the input given at time t-1

Exploding gradients and vanishing gradients are two important challenges that RNNs have had to overcome. Long-term dependencies are impossible for RNNs to handle.

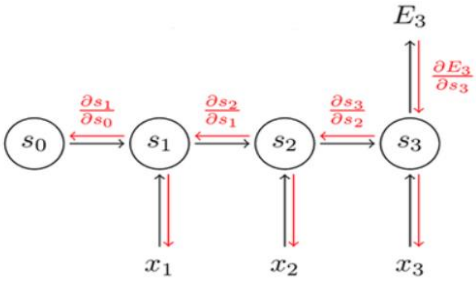


Figure 1. Backpropagation in RNN

Backpropagation in RNN is determined using the chain rule, which involves multiplying the derivatives of all prior state activation functions. As a result, if anyintermediate derivative is close to zero, the updated weight is zero, resulting in a vanishing gradient problem.

Dataset:

Time series data:

Data of a set of variables collected over time is called time series data. Then What constitutes a time series?

A simple additive dissected version will look like this, without digging into too much notation:

$$x_t = m_t + s_t + e_t$$

where:

m_t is the trend

s_t is the seasonality

$e_t$is the error or random white noise

To put it simply, the components of a time series model are defined as[12]:

The climbing or dropping data in a series are referred to as a trend.

Seasonality is defined as the series' repeated short-term cycles.

arbitrary white noise - a series of random variations

Python modules are used to retrieve historical bitcoin price data through the Coinbase Pro API. The data is downloaded as csv files and the date column is preprocessed. On the other hand, data from tweets on the bitcoin topic (keyword) is retrieved using Python's scraping packages, such as snscrape. It appears to be the columns below.

Date (Object)	Low (float)	High (float)	Open (float)	Close (float)	Volume (int)

Sentiment analysis:

All of the tweets that have been retrieved have been preprocessed in order to retrieve the polarity and subjectivity. Each tweet (string data) is processed by the textblob function, which determines the polarity and subjectivity of each tweet. Create a new 'Sentiment' column with the resultingpolarity and subjectivity in appropriate columns. On a given day, the polarity of all tweets is summed up, and a new column named polarity is formed and added to bitcoin price history data, resulting in the final dataset.

The probabilities of positive and negative statements are illustrated in table below.

- text1=TextBlob("Bitcoin price is increasing")
- print(format(text1.sentiment))
- Sentiment(polarity=0.5, subjectivity=0.6

The results are divided into two groups: polarity and subjectivity. Polarity can have a value of -1 to +1. In this case, 0 denotes a neutral statement, -1 denotes a severely negative statement, and +1 denotes a highly positive one. Subjectivity, on the other hand, has a value ranging from 0 to 1. Here, 0 denotes a statement that is extremely objective, while +1 denotes a statement that is highly subjective.

Sentiment polarity is 0.5, and subjectivity is 0.6 in the above-mentioned result.

Because the polarity is 0.5, the statement is definitely positive, and the subjective score is 0.6, the statement is subjective.

Date (Object)	Total polarity per day (int)

Proposed method:

Because parametric models for forecasting time series data, such as AR, MA, and ARIMA, require input data to be stationary, we prefer neural networks and use LSTM (Long Short Term Memory), an advanced RNN that can manage long-term dependencies. Unlike RNNs, which have vanishing and exploding gradients, LSTMs keep a cell state that acts as long-term memory and feature three gates that allow the network to pick which information to keep and which to discard.

We are using our own dataset, which was created by combining bitcoin price data with tweet sentiment scores over a one-year period. The Long Short Term Memory (LSTM) model is trained using this data.

Normalizing the data:

Because unscaled input variables could produce a lethargic or inaccurate learning process, all of the data is normalized to ensure that the quantities are within the range [0,1].

Dataset Generation:

The time series data must be divided into samples, each with its own input and output elements. To create the dataset, a lookback is chosen first.

For instance, take the following sequence:

11, 12, 13, 14, 15, 16, 17, 18, 19, 20

Then the samples for training the model will look like:

Input	Output
11, 12, 13, 14, 15	16
12, 13, 14, 15, 16	17
13, 14, 15, 16, 17	18

To create the dataset, we used 7 as the lookback period.

Training with LSTM model:

Recurrent backpropagation requires a long time to learn to store information over long time periods, owing to insufficient, fading error backflow [4]. This was addressed by introducing long short-term memory, a revolutionary, efficient gradient-based technique (LSTM). Error signals travelling backwards in time with traditional "Back-Propagation Through Time" (BPTT) or "Real-Time Recurrent Learning" (RTRL) tend to either (1) blow up or (2) vanish.

LSTMs deal with both Long Term Memory (LTM) and Short Term Memory (STM), and the concept of gates is used to make the calculations simple and effective.

Forget Gate: When LTM enters the forget gate, it discards information that is no longer helpful.

Learn Gate: The event (current input) and the STM are merged such that the relevant information from the STM may be applied to the current input.

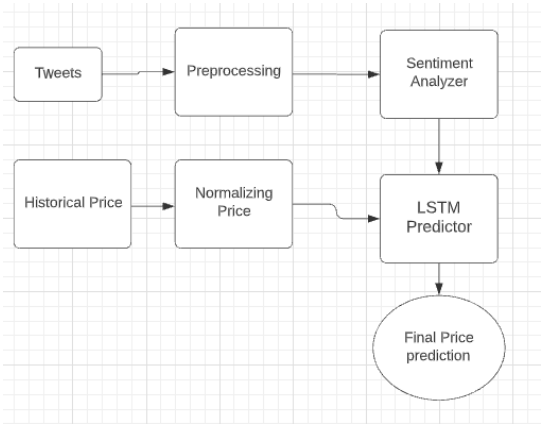
Remember Gate: LTM information that we don't want to forget, as well as STM and Event information, are integrated in Remember Gate, which functions as an updated LTM.

Use Gate: This gate predicts the output of the current event using LTM, STM, and Event, and acts as an updated STM.

The complete dataset is separated into training and testing datasets after it isgenerated with a lookback time. The input layer neurons in the LSTM model are of the train X shape.

We used Relu as the activation function as it’s derivative doesn’t cause vanishing or exploding gradient problem.

Block Diagram:



Experimental Results:

The below figure shows the training loss and validation loss. When the loss becomes constant model training is stopped with early stopping.

Validation loss is 0.0019

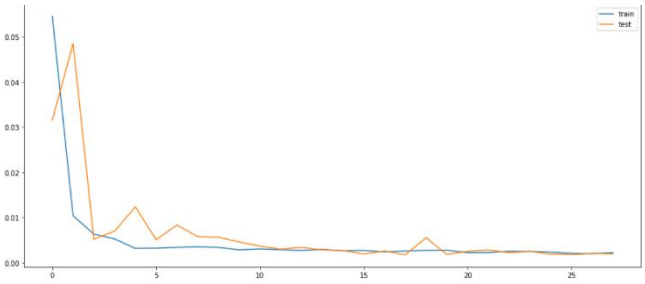


Figure 2.validation loss vs training loss

The testing data prediction plot is included below

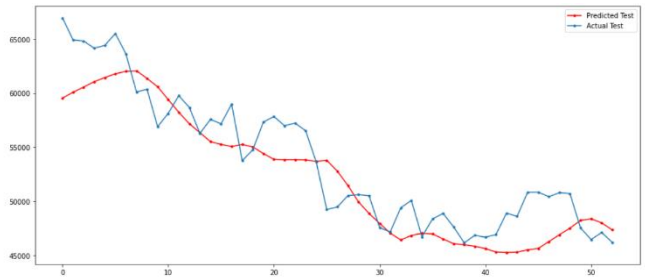


Figure 3. predicted test data vs actual test data

Below is the prediction over Training data

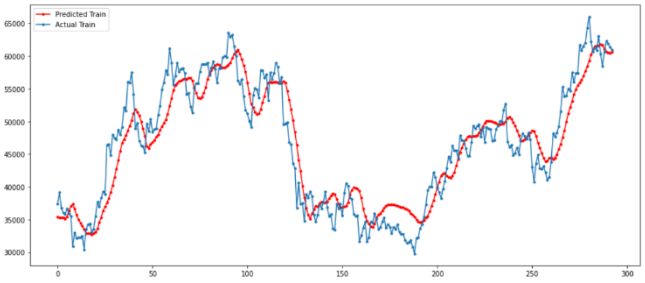


Figure 4. predicted train data vs actual train data

RMSE :

The root of the square mean of the square of all errors (RMSE) [6] is the root mean squared error. The use of RMSE value is fairly prevalent, and it is regarded as a good all-around error statistic.

The error function utilised here is the root mean square error (RMSE)[6]. The RMSE over test data is 2855.296, down from 8788.126 when the lookback period is adjusted. The RMSE on train data is now 3489.402, down from 6079.801 previously.

Conclusion:

Using LSTM model over the historical data of 1 year from 2021 January to 2021 December, we got best results compared to RNN. The error values have decreased from 8788.126 to 2855.296 . we conclude that

for data with random depth, LSTM is suitable fit.

Further considering the public opinion as a feature by including polarity column decreased the error value.

The concept could also be improved by sending users notifications when the bitcoin price reaches a certain level. This threshold (limit price) is initially set by the user.

References:

[1] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", 2008

[2] ShailakJani, "The Growth of Cryptocurrency in India: Its Challenges & Potential Impacts on Legislation", 2018.

[3] Zheng Zhao, Weihai Chen, Xingming Wu, Peter C. Y. Chen, Jingmeng Liu, "LSTM network: a deep learning approach for short-term traffic forecast" 2017

[4] FarhaNausheen, SayyadaHajera Begum "Sentiment analysis to predict election results using Python" ,2018

[5] Sepp Hochreiter "Long Short-term Memory",2016

[6] Simon P. Neill, M. Reza Hashemi, "Root-Mean-Squared Error:Ocean Modelling for Resource Characterization" 2018

[7] Praveen Gujjar J ,Prasanna Kumar H R "Sentiment Analysis:Textblob For Decision Making", 2021

[8] Poongodi M, Vijayakumar V, Naveen Chilamkurti , "Bitcoin Price Prediction using ARIMA Mode", 2018

[9] Xi Chen, Hongzhi, Yanjie Wei, Jianzhong Li, Hong Gao, "Autoregressive-Model-Based Methods for Online Time Series Prediction with Missing Values: an Experimental Evaluation", 2019

[10] Zoran Ivanovski, Ace Milenkovski, "Time Series Forecasting Using a Moving Average Model for Extrapolation of Number of Tourist", 2018

[11] Sepp Hochreiter, JurgenSchmidhuber "Long Short-Term Memory", 1997

[12] George Edward Pelham Box, Gwilym M. Jenkins, "Time Series Analysis: Forecasting and Control", 1994