Spring 5-14-2020

Understanding Impact of Twitter Feed on Bitcoin Price and Trading Patterns

Ashrit Deebadi
San Jose State University

Understanding Impact of Twitter Feed on Bitcoin Price and Trading Patterns

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Ashrit Deebadi

May 2020

The Designated Project Committee Approves the Project Titled

Understanding Impact of Twitter Feed on Bitcoin Price and Trading Patterns

by

Ashrit Deebadi

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2020

Dr. Thomas Austin	Department of Computer Science
Dr. Mark Stamp	Department of Computer Science
Dr. Robert Chun	Department of Computer Science

ABSTRACT

Understanding Impact of Twitter Feed on Bitcoin Price and Trading Patterns

by Ashrit Deebadi

''Cryptocurrency trading was one of the most exciting jobs of 2017''. ''Bitcoin'',''Blockchain'', ''Bitcoin Trading'' were the most searched words in Google during 2017. High return on investment has attracted many people towards this crypto market. Existing research has shown that the trading price is completely based on speculation, and its trading volume is highly impacted by news media. This paper discusses the existing work to evaluate the sentiment and price of the cryptocurrency, the issues with the current trading models. It builds possible solutions to understand better the semantic orientation of text by comparing different machine learning techniques and predicts Bitcoin trading price based on Twitter feed sentiment and additional Bitcoin metrics. We observe that the statistical machine learning model was able to better predict the sentiment of Twitter tweet feed compared to the advanced BERT model. Using Twitter feed sentiment and additional Bitcoin metrics, we were able to improve the prediction of bitcoin price compared to only using bitcoin's previous day closing pricing.

ACKNOWLEDGMENTS

I take this opportunity to sincerely thank my thesis advisor, Dr. Thomas Austin, for his constant assistance, valuable supervision, and encouragement. His work ethic and constant endeavor to achieve perfection have been a great source of inspiration.

I sincerely thank Dr. Robert Chun, and Dr. Mark Stamp for agreeing to be on my defense committee, giving valuable recommendations to my project without which this project would not have been successful.

I would also like to thank my family and friends for their continuous support and motivation throughout my graduation.

TABLE OF CONTENTS

CHAPTER

LIST OF FIGURES

CHAPTER 1

Introduction

In 2018, cryptocurrencies market value rose from 19 Billion USD to 741 Billion USD with Bitcoin (BTC) alone, having a 59.87 percent market cap [2]. Such an exponential increase in market volume had attracted everyone around the world to look into cryptocurrencies as a good asset to invest and make money. As a result, there have been large swings in their price and trading volume. Bitcoin alone increased 2000 percent from 833 USD on Jan 14, 2017, to 19535 USD on Dec 16, 2017, then falling to 8000 USD in Feb 2018. Blockchain, the technology behind cryptocurrencies, is suggested to be the beginning of Web 3.0, and the value of these cryptocurrencies will continue to grow in the future as the blockchain field grows.

The cryptocurrencies are bought and traded on trading platforms similarity to traditional stock market platforms. The traditional stock markets operate only for a fixed number of hours, and the company's stocks are exclusively listed on government-regulated stock exchanges like NASDAQ, Bombay Stock Exchange, Hong Kong Stock Exchange. Unlike these traditional markets, the crypto markets are not yet regulated by any country, and anyone can buy and sell them across the world at any time and hence makes their price more volatile. Government announcements, regulations, alliances, and promotions from famous personalities have played a key role in deciding the price of cryptocurrencies. If a statement is made in the USA, the news propagates to other countries like India, China via the internet, Social media websites, etc.

The paper aims to understand the sentiment of the market through twitter and analyze its effect on the trading volume and price of BTC. Since ''BTC – Fiat'' currencies combination is the most widely used exchange pair and transactional data of Bitcoin is highest, we choose BITCOIN as the cryptocurrency to conduct research. The selected articles and proceedings of this survey include research in the field of

sentiment-based prediction models, bitcoin price formations, and Natural Language Processing. Once we are able to see any relation between twitter sentiment and bitcoin price, we try to use the sentiment to predict the price of Bitcoin.

CHAPTER 2

2.1 Machine Learning Fundamentals

Machine Learning Algorithms are broadly classified into two categories: Supervised and Un-Supervised Learning, as shown in Figure 1.

Supervised Learning: It is the process of having input variables (X) and output variable (Y) and then use mathematical models to find how these input variables (X) are mapped to output (Y). Here we build a model using lots of training data (existing labeled data) and then use it to predict output (y) for new incoming data.

Unsupervised Learning: It is the process of having only input data (X) and no corresponding output, then use mathematical models to find some underlying structure in this data and group them. Here we build a model using existing un-labeled data to group them by finding some similarities between them.

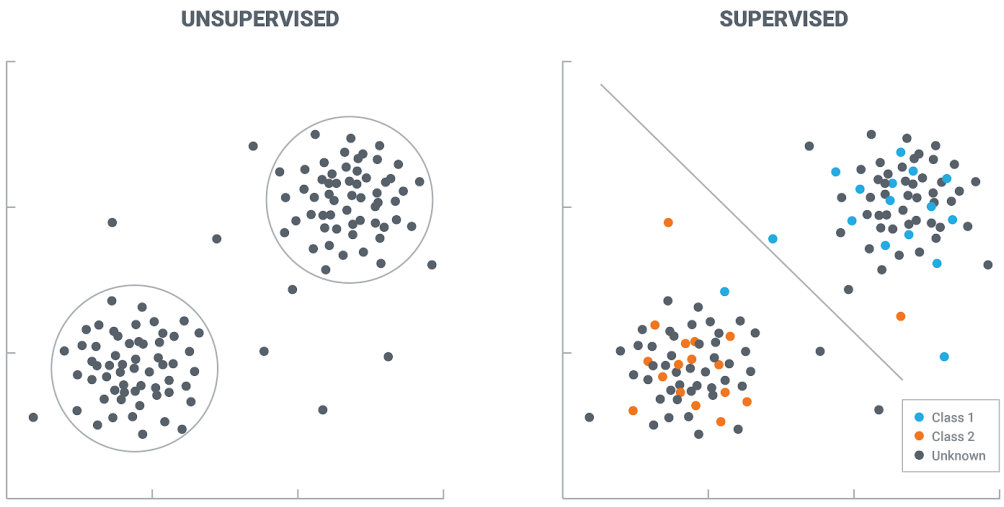

Figure 1: Supervised and Unsupervised Learning
Source : [3]

Since we are using only supervised learning for our project, we will discuss the two types of supervised learning algorithms- classification and regression.

Classification is where the output variable is a category. The output is always a binary number. E.g., For the problem of classifying the sentiment of an article, the output is always either positive, negative, or neutral. In every case, the output will belong to any one of the categories- positive, negative, or neutral.

Regression is where the output variable is a real value. The output is always in terms of an actual number like price or units. For example, the problem of predicting the price of a stock, here the output is the price in USD, which is always an actual value like 200, 3000, 10000 USD rather than categorical values.

2.2 Textual Analytics Background

Semantic orientation (SO) is a measure of subjectivity and opinion in the text. It helps in capturing an evaluative factor and strength of a subject topic or idea. The evaluative factor is whether the opinion is positive or negative, and the strength is the degree to which the word or sentence, in the opinion, is positive or negative. When we use semantic orientation in the analysis of public opinion, like online articles, social media posts, it helps in measuring the popularity and success of that particular topic.

Sentiment analysis refers to the general method of extracting subjectivity and polarity from the text. It is essential to understand the semantic orientation of the text, i.e., the polarity and strength of words, to find the sentiment of a given text. Hence, it has drawn a lot of attention, and there have been several machine learning approaches to understand how semantic orientation improves sentiment analysis.

2.2.1 Bag of Words Approach

Bag-of-Words (BOW) is a representation of the text that describes the occurrence of words within a document. It consists of two main components - a list of all words in the document, and the number of times each word has occurred. The idea behind this approach is, documents are similar if they have similar content. By using this,

we can infer information between texts following the below steps :

1. Taking all the training data and pre-processing it by ignoring cases, punctuation and fixing misspelled words, stemming the word.

2. Building a vocabulary dictionary by taking unique words and keeping the count of how many times each word has occurred. The vocabulary can be built by taking N-Grams, where N is a number 1,2 or 3. 1-Gram means taking each word, 2-Grams is making a pair of words, etc.

We can further fine-tune this approach by using Term Frequency - Inverse Document Frequency (TF-IDF) where:

1. Term Frequency is the score if the word in the current document.

2. Inverse Document Frequency is the score of how rare the word is ac-cross the document.

By doing this, we can get details of what the important words are, which help us understand the SO problem, the limitations of this approach are as follows:

1. If the test data has a word which is not present in the vocabulary dictionary, the model cannot address that word.

2. It has sparsity where some words have very few values, and some are large.

3. It cannot consider the context of the text as it is just considering word count.

2.2.2 Word2Vec Approach

Word2Vec, published in 2013, brought a significant boost to the field of Natural Language Processing. With the intuition, a word can have different meanings when used in a different context; word2vec tries to capture the different contexts of words based on its position in the sentence. E.g., the word ‘‘Apple’’ can represent fruit or the company, and when we say, ‘‘I love eating an apple’’, we refer to the fruit, and when we say ‘‘I love the new Apple iPhone’’, we refer to the company. So, the context

in which we use the word makes it very important in understanding its meaning.

A vector represents each word in the data, and each word has two vectors, center vector (v_c) and context vector (u_o). The center vector is accessed when the word is the center word at position wt, and we obtain the context vector when the word is in the surrounding window (n).

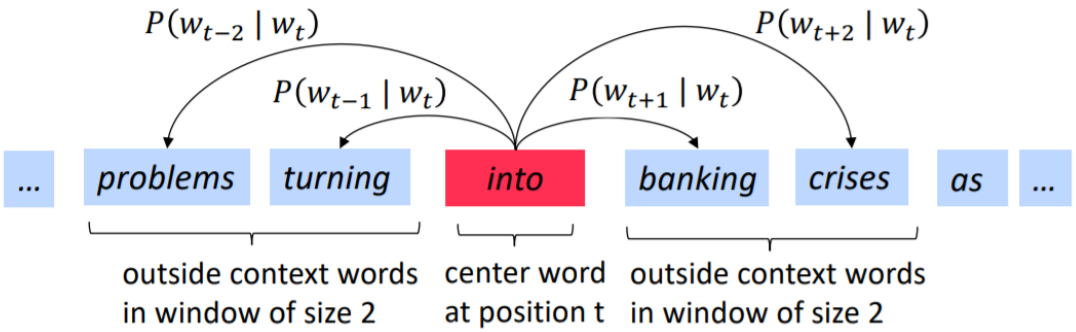

Figure 2: Word2Vec model
Source : [3]

As shown in Figure 2, at each position, ‘‘t’’ in the text, we calculate the probability of context word given center word using the similarity of word vectors for center having a fixed window size. We keep updating these vector values to maximize the probability of context word given center word.

$$\log p(o|c) = \log \frac{\exp(u_o{}^T v_c)}{\sum_{w=1}^{V} \exp(u_w{}^T v_c)} \qquad J(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

(a) (b)

Figure 3: (a) P(w_{t+1}/w_t) simplest first formula,and (b) Objective function of Word2Vec

Source : [4]

Figure 3(a), represents Figure 2's P(w_{t+1}/w_t) in a generalized form where u is the probability distribution when it is the center word, and v is the probability distribution when it is the context word as shown above. As discussed earlier, the aim is to maximize these vectors probabilities subject to minimize the objective function J(θ) Figure 3 (b)using gradient descent.

There are two variations of the Word2Vec model, Continuous Bag of Words (CBOW) and Skip Gram. In CBOW, the model is used to predict the next word from the current window of surrounding context words, and in Skip Gram, the model uses the current word to predict the surrounding window of context words.

Limitations:

1. The sub-linear relationships are not explicitly defined, and there is little theoretical support behind such characteristic [4].

2. As the size of vocabulary increases, using softmax function would be challenging to train the model despite approximation algorithms like negative sampling (NEG) and hierarchical softmax (HS) to tackle this issue, other complications still occur [4].

2.2.3 Recurrent Neural Network

Recurrent Neural Networks (RNN) is a unique architecture of the feedforward Neural Network with memory. RNN uses the information learned in previous states and propagates it forward into the network. Using this architecture is helpful in many applications like Speech Recognition, Predictive Typing, and Machine Translation where the sequence of data is essential.

In general feedforward Neural Networks, the inputs of the vector's are considered to be independent, and the information that it learns at X_0 passes forward to the next state, say X_1, as shown in Figure 4. In RNN, the sequential data generate the hidden state and adds along with the output of network-dependent with the hidden state, as

shown in Figure 5. But as we proceed to move forward deeper into the network to X_5, X_{10}, the X_0 information gets lost. This architecture doesn't perform well in cases where the sequence of data is essential.

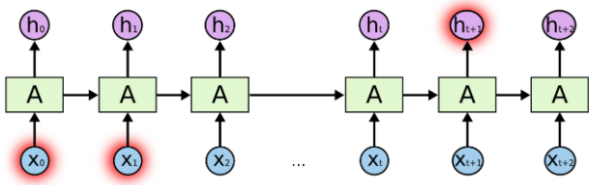

Figure 4: RNN Forward Transmission
Source : [4]

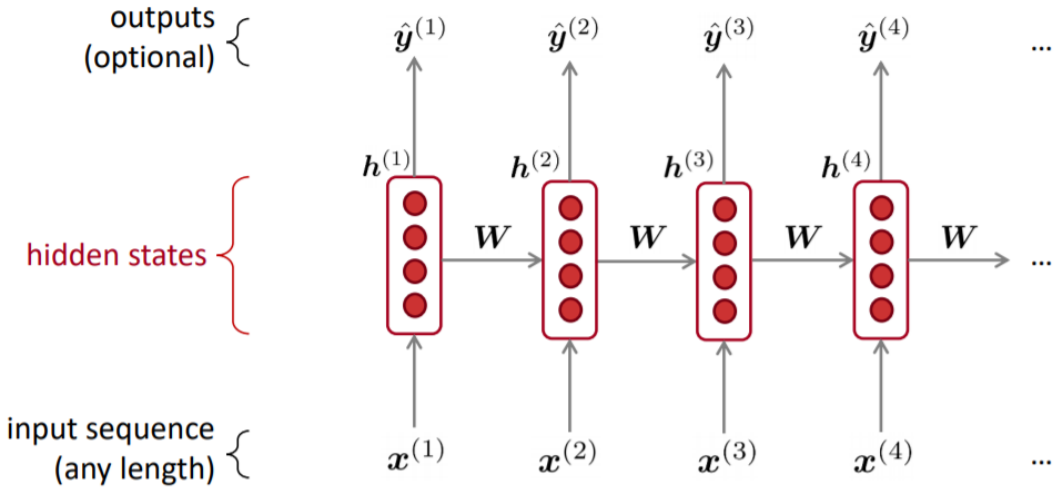

Figure 5: RNN Architecture
Source : [4]

The weights in all the hidden layers (h) are updated using backpropagation. So, if we change the intensity of weights at each state, then we can influence the rate at which propagation moves forward, i.e., the increasing weight moves the data forward and decreasing the weight stops from moving forward. But as the network

start's becoming complex, even this creates a problem called vanishing gradient and exploding gradient where the vanishing gradient is when the weight is too small, and the exploding gradient is where weight is too large.

2.2.3.1 Stacked Auto Encoders + Long Short-Term Memory

Long Short-Term Memory (LSTM) solves the vanishing gradient and exploding gradient problem by explicitly introducing a memory unit called the cell into the network. Figure 6 is the diagram of an LSTM building block where each block is a cell.

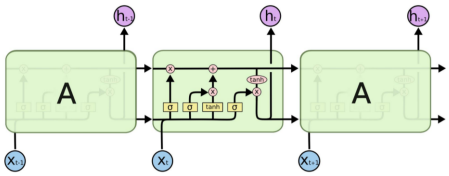

Figure 6: LSTM Cell
Source : [5]

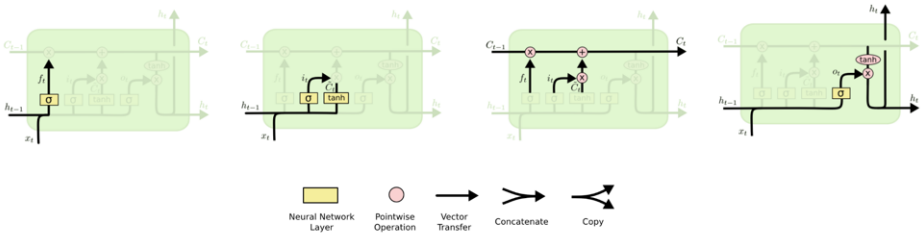

Figure 7: LSTM Architecture
Source : [5]

The Figure 7 ordered from left to right explains the internal working of an LSTM cell. The first step in LSTM is to decide which information goes forward from the previous cell. Using the sigmoid function with inputs as h_{t-1} and x_t, we filter the input and output (f_t) information from the previous cell is taken forward. This layer is called the forget gate layer.

The second step is to decide which information from the current cell is kept, using another sigmoid and tanh layer; the outputs from these functions are taken using pointwise operations and updated to the current cell. Next, we update the old cell state, C_{t_1}, into the new cell state C_t. In the final step using the sigmoid layer, tanh layer, we decide which data goes out from current cell state C_t.

By controlling forget gates, errors will be backpropagated through any number of layers. This architecture enables the network to learn tasks that depend on events that occurred many steps back [5].

Though LSTM solves the memory issue of RNNs, when sentences are too long, LSTM still doesn't perform well. This is because, as the distance from the current cell keeps increasing, the probability of keeping the word's context decreases exponentially in cells that are far away [6]. E.g., in the sentence, ''Bitcoin is insanely amazing, I will go online and buy some of it immediately''. When the model is reading the final word ''it'', we are not sure what it's referring to and what influence it has on overall sentence meaning.

Another problem with LSTM is that it's hard to parallelize the work for processing sentences since you have to process word by word.

2.2.4 Transformer

As discussed in the earlier section, LSTM can be used to deal with sequence-depended data where it either propagates the information forward or forgets it; Seq2Seq models can be used to deal with similar sequence- depended data for translation and understanding the context. Every Seq2Seq model has an Encoder and a Decoder where the Encoder uses the input data to transform into a higher-dimensional space, and Decoder takes this higher dimensional text and translates into the output sequence.

The Transformer is similar to the Seq2Seq model, which has 6 six encoders and 6 decoders, as shown in Figure 8. All the Encoders have the same architecture and similar internal workflow. Likewise, all the Decoders have the same architecture and similar internal workflow. An Encoder consists of two layers: Self-attention and a Feedforward Neural Network.

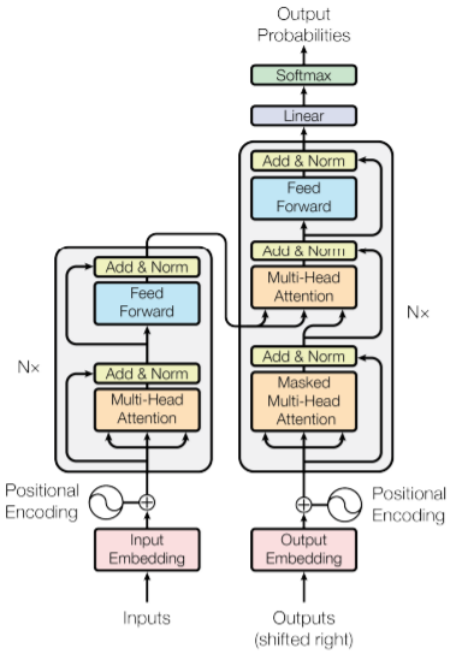

Figure 8: Transformer Architecture
Source : [5]

Attention [7], which proposes a new way of translating the word by paying attention to a part of a subset of the information. In the, E.g., ‘‘Bitcoin is insanely amazing, I will go online and buy some of it immediately’’ when the model is reading the final word ‘‘it’’, attention helps in associating ‘‘it’’ with ‘‘Bitcoin’’.

Unlike RNN, where we encode the entire sentence in a hidden state, here we encode each word in a corresponding hidden state, which passes to the decoding stage. We do this because every word in the sentence can have relevant information that can be transferred later in the network instead of dropping them in earlier states, which

was the main issue in LSTM. So for the decoding to be precise, using Attention, it takes every word of the input into account.

Attention works as shown in Figure 9 [7], the first step takes an input sentence e,g, ‘‘Thinking Machines,’’ embeds each word of a sentence using embedding technique like Word2Vec. The second step is to generate a Query vector W-q, a Key Vector W-k, and a Value vector W-v for each of this embedded vector. The third step is to calculate the score by taking the dot product of the query vector and key vector, which is the score of each word at each position. The fourth step is to normalize the score using the softmax layer, and the fifth step is to calculate each value vector with the softmax score so that only words with importance dominates in the vector. Finally, we sum up the weighted value vectors, which gives output Z of a single attention layer. Since Transformers uses 6 Attention heads, we combine outputs of 6 different self-attention layers with a weight matrix to provide a final output of this layer.

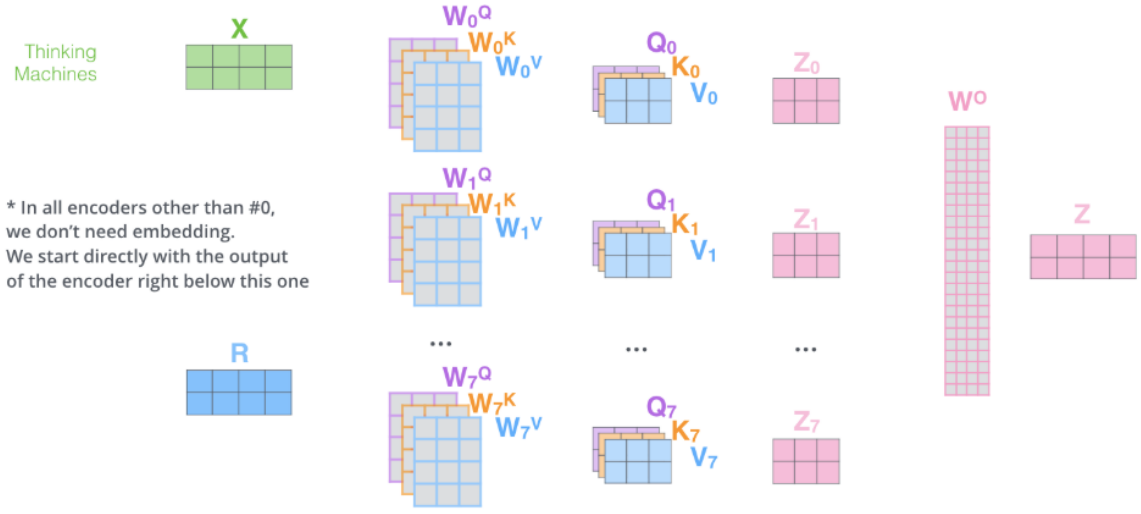

Figure 9: Attention Workflow
Source : [8]

The model maintains the order of the words in the input sequence by adding a vector to each input embedding. The embedded vectors follow a pattern that the

model learns, which helps in determining the position of each word, or the distance between different words in the sequence [7] as shown in Figure 13.

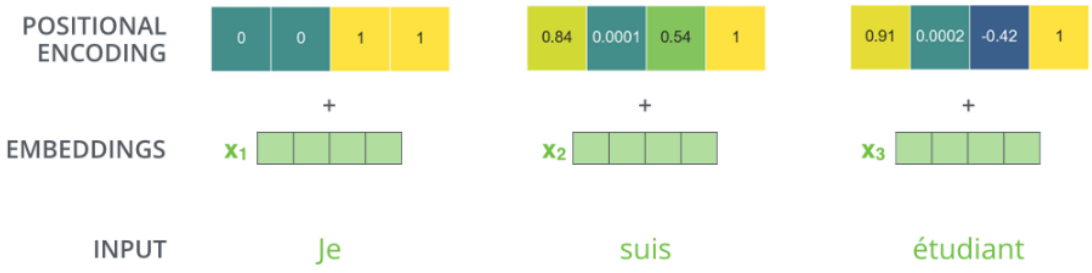

Figure 10: Positional encoding with an embedding size of 4
Source : [8]

Summarizing the Encoder section, the inputs are first embedded, then positional vectors are added. Next, it passes through the self-attention layer whose output is layer normalized, and this normalized output goes trough a Feedforward network whose output is again layer- normalized.

The Decoder works in the same way, where the only difference is that self -attention layer is only allowed to see the previous positions of the output sequencing masking the future positions before sending it to softmax calculation step in self -attention layer. The output of the final Decoder is a float vector, which passes through the final softmax layer, which gives the log-probability of each word in the final vector.

2.2.5 Bidirectional Encoder Receiver from Transformers

Bidirectional Encoder Receiver from Transformers, i.e., BERT, is a new language representation model that gives the state of the art results [9] for 11 Natural Language Processing tasks. It is based on Transformer architecture and uses only Encoders for word embedding as shown in Figure 11. As the name states, it's Bi-Directional, unlike most embedding techniques, e.g., Word2Vec, which are unidirectional, either right to

left or left to right, BERT considers both left and right directions making it better understand the context while embedding the word.

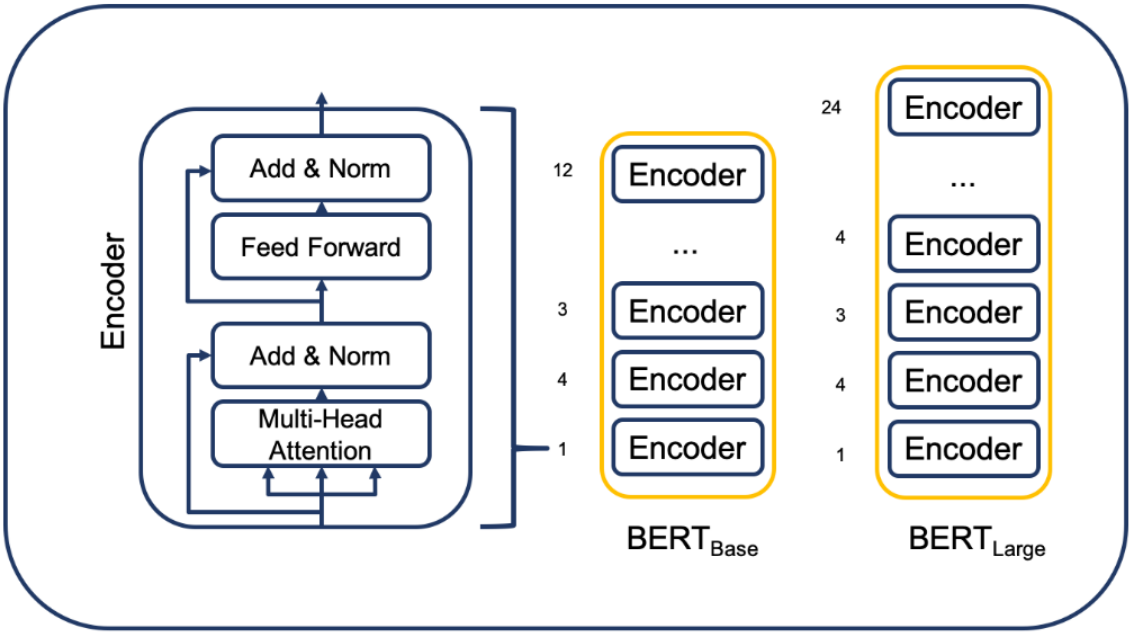

Figure 11: BERT Architecture
Source : [10]

Using Transformer and its Attention technique, our example ‘‘Bitcoin is insanely amazing, I will go online and buy some of it immediately’’ was able to associate the embedded word ‘‘it’’ with ‘‘bitcoin’’. Here the input to Transformer was an embedded vector, transforming the sentence into vector form using embedding technique like Word2Vec. As we known the limitations of Word2Vec from earlier discussion, only a fixed context window is considered while embedding current center word, which does not word well for long and complex sentences. Overcoming this problem, BERT uses Attention and Transformer's encoder Technique improving word embedding.

BERT currently has two different models: BERT base and BERT large the differences, as shown in Figure 13, where the BERT base has the same of existing models like Open AI GPT and BERT large was developed to outperform the base

model [9].

	BERT Base	BERT Large
Layers	24	112
Hidden Size	768	1024
Heads	12	16
Parameters	110M	340M

Figure 12: BERT base and BERT large parameters
Source : [10]

BERT is developed in two steps: Pre-Training and Fine-Tuning, where Pre-Training is done on a huge unlabeled English Wikipedia and BooksCorpus dataset. The pre-training tasks are Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). BERT is trained using tokens which are generated using WordPiece tokenizer, an inbuilt library which consists of most commonly used 30,000 English words and letter of all alphabets. While reading the word, if it's not present in the WordPiece vocabulary, it is broken into smaller words until it finds that word. If the words are still not found, then it further breaks the word into single letters, the tokenizer remembers the order of the words using two number signs "##".

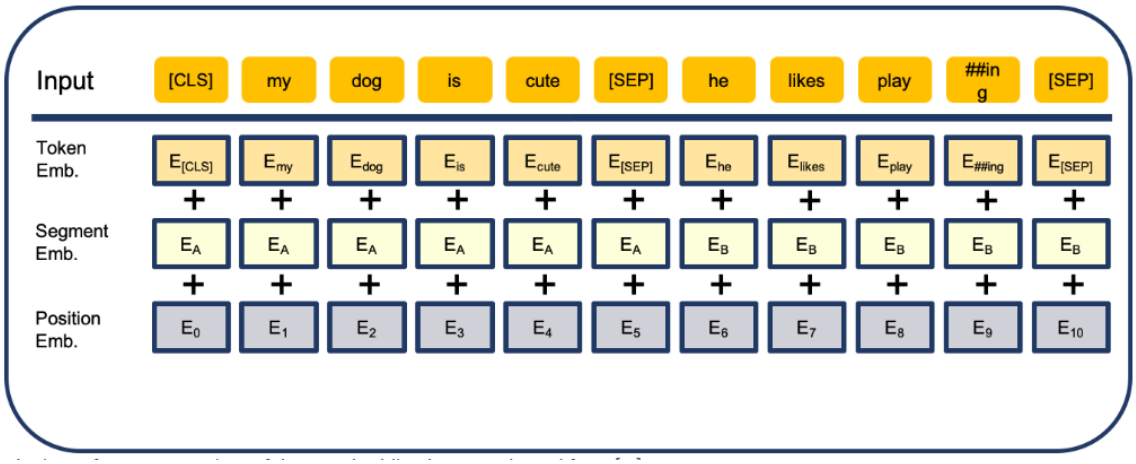

Figure 13: Embedding using WordPiece tokenizer
Source : [9]

Three special tokens are added in each text sequence: a [CLS] token, a [SEP] token, and positional embeddings. [CLS] token is added at the starting of each sentence, [SEP] token at the end of the sentence, and word positional embedding to remember the position of each word in the input sequence as shown in Figure 13. The output generated using the above tokens is fed as the input to the BERT.

BERT is pre-trained using two different tasks: Masked Language Modeling and Next Sentence Prediction. In Masked Language Modeling in every input sequence, some words are masked, and the model should predict the masked words, as shown in Figure 14 , here 15 percent of the words are selected randomly, and out of these, a [MASK] token replaces 80 percent words. Out of the remaining 20 percent, a different word replaces 10 percent of the words, and the rest 10 percent are left unchanged.

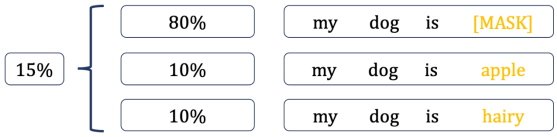

Figure 14: Masked Language Modeling in BERT
Source : [9]

BERT uses Next Sentence Prediction to infer relationships between two sentences. As shown in Figure 15, the model takes two sentences A and B, where it has to predict if sentence A is followed by sentence B. The goal of training the BERT model using Masked LM and Next Sentence Prediction is to minimize the combined loss function.

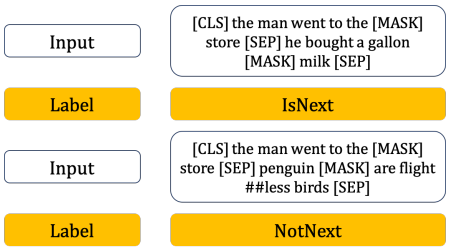

Figure 15: Masked Language Modeling in BERT
Source : [9]

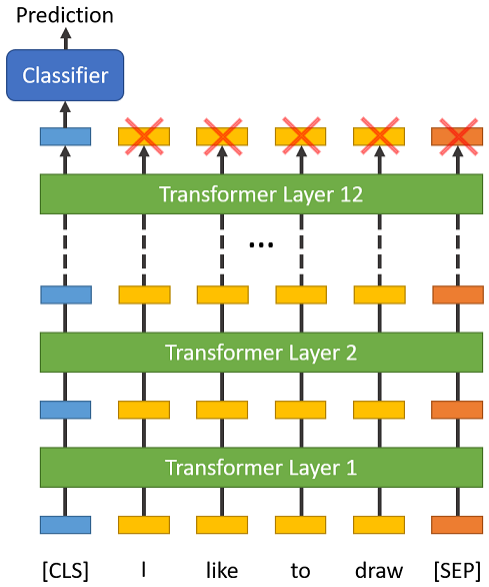

Figure 16: Fine tuning Next Sentence Prediction Layer in BERT
Source : [9]

The second step in BERT modeling is Fine-tuning. BERT can be fine-tuned for any specific task by adding an additional layer to the base model. For the sentiment analysis task, which is a classification problem, a classification layer is added on top of Next Sentence Prediction Transformer output, as shown in Figure 16. Fine-tuning training retains most hyper-parameters of the original BERT model tuning it to our specific task.

2.3 Predictive Analytics Background

Predictive Analytics is the technique to predict the future using historical data. Generally, we take historical data, and mathematical techniques are applied to understand the trends, based on which the value is estimated in the future [11]. With the rise of big data, as huge volumes of data are being stored, applying statistics and machine learning to this data for understanding the trends and patterns is becoming an emerging topic. Using this technique and applying it to predict stock price has caught many researcher's attention.

Stock Market is an industry where millions of transactions are processed and stored every day [12]. The price on which a single stock trades is always uncertain; it can be somewhat understood by looking at its historical trading prices and other data points. These kinds of problems are framed as Time series problems. With the practical application of ML and statistical techniques, we can utilize this historical data and other points to estimate somewhat the future price, which is better than random guessing.

For example, if the company Apple's stock is being traded current at 100 USD, if we want to estimate its price after a month, it would be a very tough task. E.g., analyzing the historical trading prices of Apple, we noticed the price on an average fluctuates 4 percent monthly. If the market is a bull market - Trend is increasing, we can say there is a high chance that the price of apple would be 104 next month, but if it was a bear market - Trend is decreasing, the price would be 96 USD. Therefore, using historical data, we were able to capture trends, and then using applied mathematics, we noticed a 4 percent fluctuation. This estimation is generally better random guessing.

A Time-series problem usually has four components- Level, Trend, Seasonality, Noise. Out of these, trend - Increasing/decreasing pattern and seasonality, repetition, if any are the important features that models capture. Many research papers were published in the last few decades to capture these components effectively. These techniques can be broadly classified into statistical and machine learning models and based on volume, type of data respective techniques are used.

2.3.1 Linear Regression

Linear regression is used to find the relationship between a dependent variable (y), and an independent variable (x). For a simple linear regression, we will have only one independent variable x, but for multilinear regressor, there can be many

independent variables of x. E.g., For Bitcoin price prediction, we will consider only date and its price as the dependent variable, and our aim is to predict the price for a given day. A simple Linear equation is shown below.

$$y = a + bx$$

Here y is the predicted value, a is the y-intercept, b is the slope of the line, and x is the coefficient variable. The aim is to find the best-fitting line, which minimizes the sum of squared errors (SSE) between the actual value of a Bitcoin price (y) and our predicted stock price. The drawback with this approach is, it doesn't capture trends, seasonality, and is not a good approach for time series data.

2.3.2 Auto Regression Integrated Moving Average

Auto Regression Integrated Moving Average (ARIMA) is an advanced statistical technique that integrates regression and moving average to capture trends. Auto Regression(AR) uses previous values from past time intervals to estimate the next value:

AR(p) is the method where p stands for periods to consider in the past to predict the next outcome. AR(1) considers only $t-1$ value. AR (2) considers $t-1$, $t-2$ values while predicting the value at time t.

Formula:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t,$$

Moving Average (MA) residual errors from past time stamps forecasts to predict the next outcome.

Formula:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q},$$

Combining AR and MA gives ARIMA, which helps in better analyzing the past components.

Formula:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t,$$

ARIMA model can be represented as ARIMA (p,d,q), where p is the order of AR, d is the degree of the first difference, and q is the order of MA.

2.3.3 Prophet

Prophet, an open-source time-series model, was developed by Facebook and released in 2017 [13]. The prophet is a model for forecasting time series data; it uses an additive model where we capture non-linear trends controlling yearly, weekly, daily seasonality, and holiday effects. It gives good results for time series data, which has strong seasonal effects and several seasons of historical data. It works well in handling missing data, changes in the trend, and outliers well.

The model takes a window size N, which can be referred to as number of days back in time, as shown below. The model can change individual θ values. The model gives us the option of handling seasonality and special events.

$$F(y_{n+1}) = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... + \theta_n x_n$$

For time-series models, there can exist seasonality that happens due to particular human behavior. For example, a stock market operates only 5 days a week, which can produce a pattern that repeats every week. Vacations and breaks are effects that might repeat every year, handling these event smoothens the forecasting curve. The model manages the seasonal events by using Fourier series transformations.

For handling certain events which caused the minor spike, say there is a product launch on Day 24. While considering that day in the window N, the models check if

that day is flagged or not, improving overall impact on the surrounding days of Day 24. Internally the flagged days are handled by adding regularization to the θ of that day, smoothing the model's predictive curve.

2.4 Relevant Classification Algorithm

Classifying data is a common task in machine learning for given data points, each belonging to one of two classes, and the goal is to determine which class a new data point belongs to.

2.4.1 Support Vector Machines

Support Vector Machines, proposed by Vapnik [14], is a widely used classification model where it maps input vectors into a high-dimensional feature space by constructing a separating hyperplane in the higher dimensional space. While predicting, the class of the unlabeled data point is determined by plotting it and checking which side of the hyperplane does this point belongs to.

Support Vector Regression (SVR) is a type of Support vector machine that supports linear and non-linear regression. As shown in Figure 18, the model aims to fit as many data points as we can between the lines without violating the margin of error. The margin of error is given by epsilon.

$$f(X) -> y$$

The SVR takes the input X and gives an output y, internally using the training data X, y the model learns the co-relation of how y varies with X.

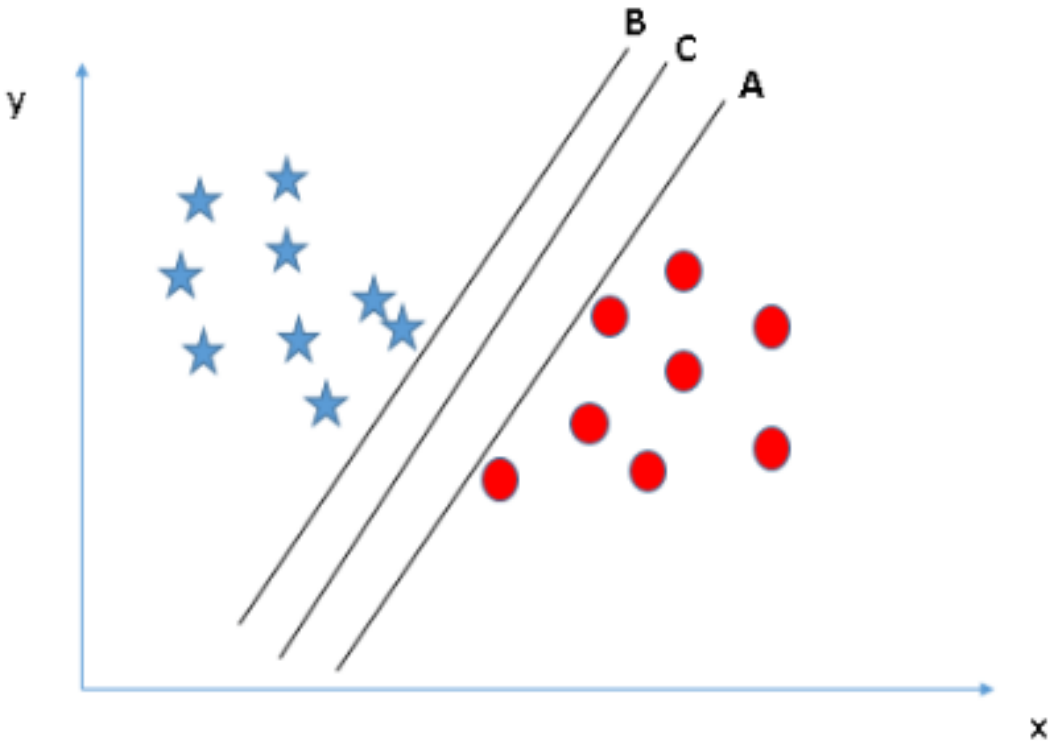

Figure 17: data points represented in a 2D space by SVM
Source : [15]

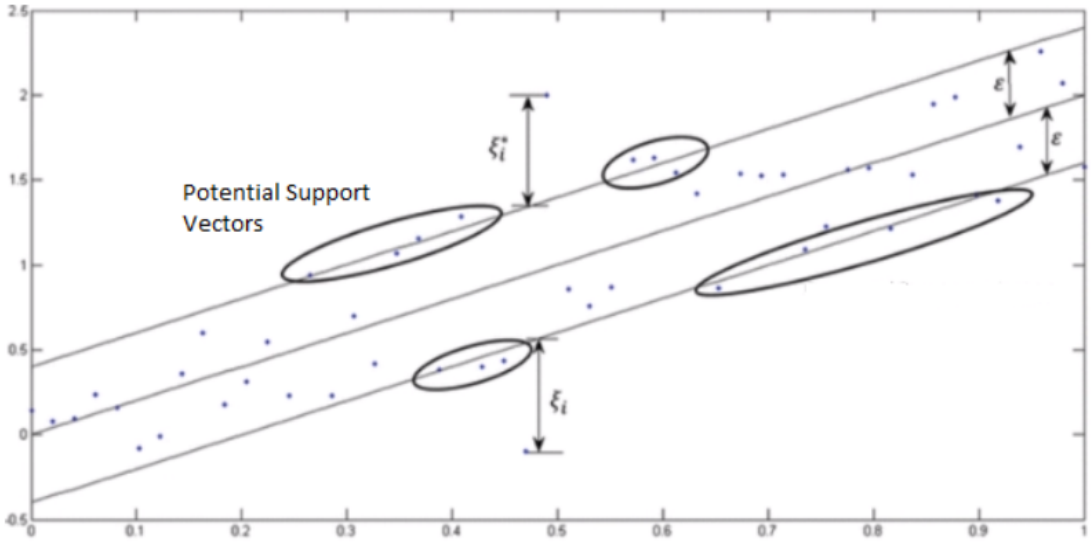

Figure 18: data points represented in a 2D space by SVR
Source : [16]

CHAPTER 3

Proposed Model

3.1 R2 score

For Regression, we would like to know what is the variance between the actual number and predicted number; in such a case, R^2 is the metric that is used. R-squared is the statistical measure that tells us how close the data points are to the fitted regression line.

$$R^2 = ExplainedVariation/TotalVariation$$

The best possible score is 1.0, and it can be negative. As shown in Figure 19, the model score for the left model is low compared to the right model, this is because the data points for the left model are scattered a lot, which makes it hard to fit into our model. Whereas data points in the right model are not scattered around, and the model was able to fit better, giving it a better R^2 score.

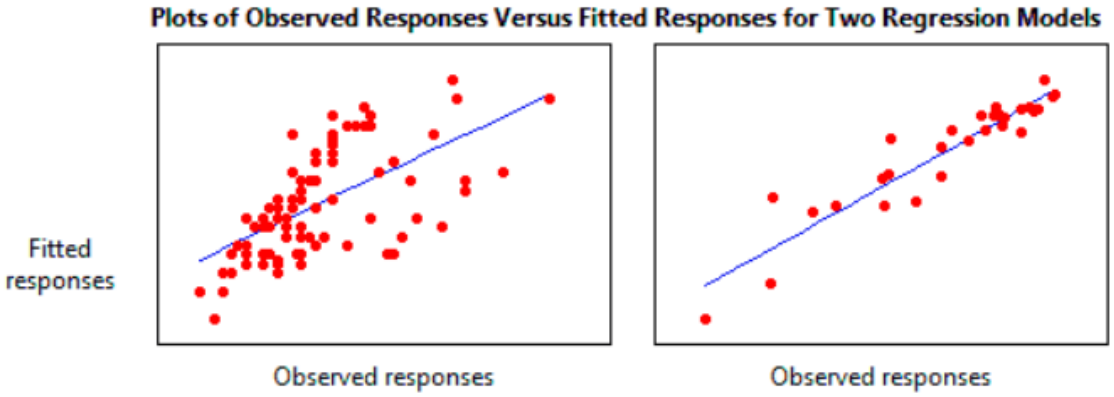

Figure 19: Regression Model for two scattered points
Source : [17]

3.2 Predictive Model

We use the Prophet model as discussed in the earlier section, ref performs well to capture daily, weekly trends, and performs well for daily predictions; hence we

choose this model to set up a base model to predict the bitcoin price. We are using the sliding window technique, as shown in Figure 20.

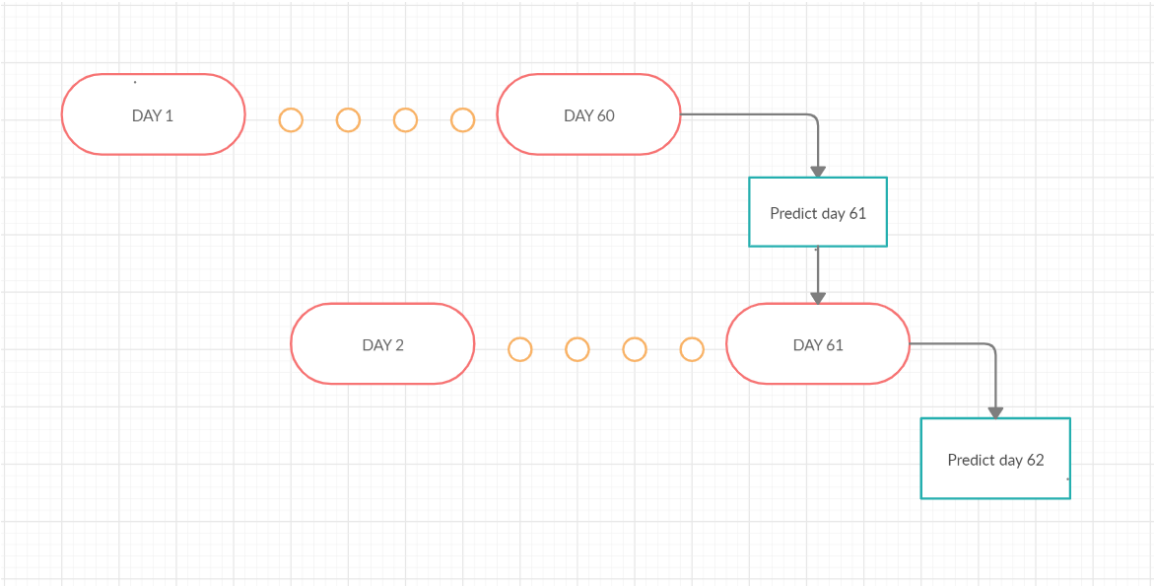

Figure 20: Prophet Model

We take a sliding window with range n, then the model trains on BTC close price from t_0 to t_n and predicts the value on t_{n+1}, then we retrain the model from t_1 to t_{n+1} and predicts t_{n+2} and so-on. For example, n = 60 where n is the number of days, we predict the bitcoin price of Day 61. Then we retrain the model from Day 2 to Day 61, maintaining n = 60 window size to predict the bitcoin price of Day 62. We keep retraining the model again so that the model predicts better based on changing trends. We use the R2 score on this test set as our evaluation metric. Once the base model is running, the aim to see if we can improve the R2 score of the test set by adding a daily sentiment and additional features.

3.3 Sentiment Analysis Model

As we are using twitter feed for aggregating Bitcoin opinion for a day, the tweets are posted throughout the day, which needs to be collected and processed each day. As shown in Figure 21, using Twitter API, we first extract Bitcoin-related tweets

from the API using ‘‘#BTC’’, ‘‘#BITCOIN’’ hashtags, with the English language as the filter. Once the tweets are collected, they are pre-processed using pre-processing techniques, which will pass through a classification algorithm for classifying. If the individual tweet is positive 1 or negative 0, as shown in Figure 22. Once all the tweets are classified, we need to aggregate that day's overall sentiment, i.e., if it was a positive day for Bitcoin stating people were supporting bitcoin or negative day stating people were not supporting on that day.

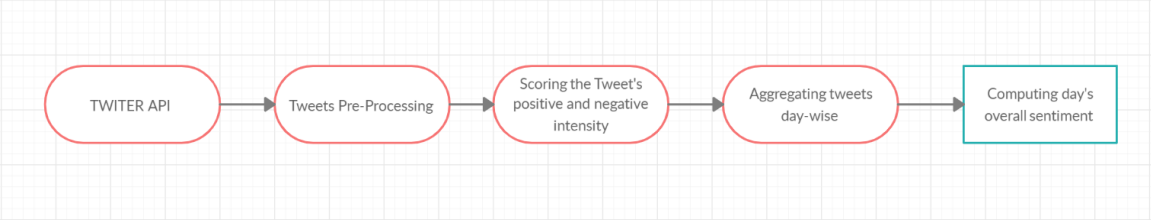

Figure 21: Sentiment Analysis Pipeline for Twitter Tweets

TEXT	SENTIMENT
awww that bummer you shoulda got david carr of...	0
is upset that he can not update his facebook b...	0
dived many times for the ball managed to save ...	0
my whole body feels itchy and like its on fire	0
no it not behaving at all mad why am here beca...	0
just woke up having no school is the best feel...	1
thewdb com very cool to hear old walt intervie...	1
are you ready for your mojo makeover ask me fo...	1
happy th birthday to my boo of alll time tupac...	1
happy charitytuesday	1

Figure 22: Classification Algorithm Result

3.4 Final Merged Model

The final proposed model is as shown in the Figure 23, where the left side of the figure is the workflow of the predictive model as explained in the earlier section, the center workflow is the sentimental analysis workflow as explained in the earlier section. The SVM model has three inputs - next day price prediction from the predictive model, current-day aggregated twitter sentiment, and additional features, which will be explained in the dataset preparation section. These inputs, when passing through the SVM model, give us the adjusted bitcoin price, which is expected to be better than our predictive analysis base model.

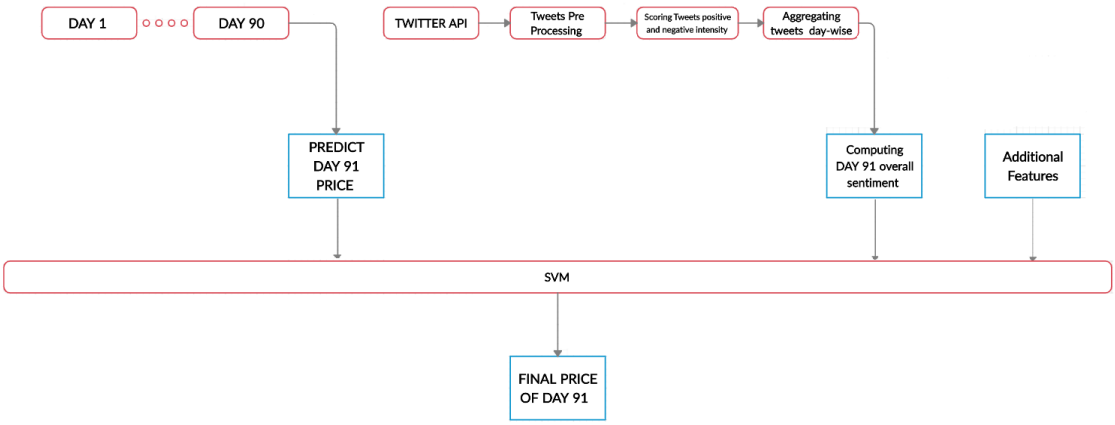

Figure 23: Final Proposed pipeline

3.5 Python Libraries

We use a couple of python libraries throughout our experiment, and this section will be covering an overview of some common python libraries -SentimentAnalyzer, GridsearchCV.

1. SentimentAnalyzer: A SentimentAnalyzer is an NLTK function implemented to facilitate Sentiment Analysis tasks. Using NLTK features and classifiers, with the help of the package we can extract the intensity of positivity, negativity from a given

sentence.

2. GridsearchCV: Tuning hyperparameters for a model is very important in improving its performance. In the sci-kit-learn package, there is an inbuilt function GridsearchCV that makes it easy to fine-tune the parameters performing an exhaustive search. For example, for Support Vector Classifier, C, kernel, and gamma are some hyperparameters which can be fine-tuned using GridsearchCV. The search takes an estimator (regressor or classifier such as sklearn.svm.SVC()), a parameter space, a method for searching or sampling candidates, a cross-validation scheme, and a score function.

CHAPTER 4

Dataset Preparation

A general tweet on Twitter can have handle names of other users, URLs to a website, text, and emojis expressing their opinion about a certain topic. When we make an API call and download a tweet, any of the above details if present comes with the tweet. Most of these details don't bring much information to understand the user's emotion while writing the tweet, and we can consider it as noise. We need to remove this noise as much as we can to get good results in our experiments. Hence, we do data cleaning on the collected tweets to reduce noise from them. It is done by removing punctuation, removing stop words, keeping only digits, and English letters.

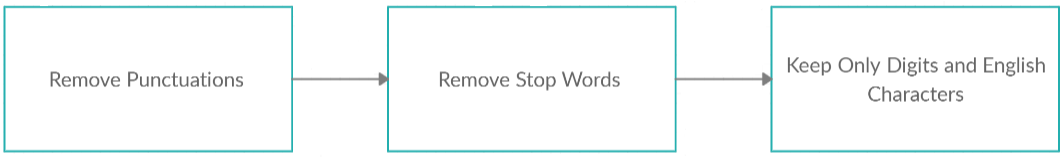

Figure 24: Data Cleaning Workflow

As shown above, in Figure 24 if the tweet has any punctuation's like ‘‘!’’ ,‘‘?’’, ‘‘rt’’, ‘‘via’’, ‘‘and’’ are removed. Next, if the tweet has any stop words like ‘‘a’’, ‘‘an’’, ‘‘the’’, ‘‘there’’ are removed. Later, we check if a character is a valid digit or an English letter and only valid ones are retained. By the end of this process, we get a cleaned tweet from which noise is reduced.

4.1 Dataset 1

We download the dataset from Kaggle [18], which is prepared by Stanford University. It consists of 1.6 million tweets equally distributed into positive and negative tweets. Each tweet consists of six fields- sentiment of the tweet, tweet ID, tweet date, query, userID, and text as shown in Figure 25 :

Sentiment	ID	TweetDate	query	userID	text
0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

Figure 25: Preview of the Tweets downloaded from [1]

Out of these six features, we use only sentiment and text for our experiments. Using the data preparation technique, we drop all other features and keep only text and sentiment, as shown in Figure 26. Sentiment feature has two binary values 0 for negative, and 1 for positive, the text feature has tweet written by the user.

Text	Sentiment
awww that bummer you shoulda got david carr of...	0
is upset that he can not update his facebook b...	0
dived many times for the ball managed to save ...	0
my whole body feels itchy and like its on fire	0
no it not behaving at all mad why am here beca...	0

Figure 26: Pre-processed data containing user tweet and polarity

A standard tweet can have a maximum of 140 characters limit in it, but as we see below, in our dataset, some tweets have over 200 characters, which means some illegal characters need to be removed. From Figure 27, we can see the result of pre-processing removed anomalies, illegal characters, and reduced the size of the tweets to standard 140 characters.

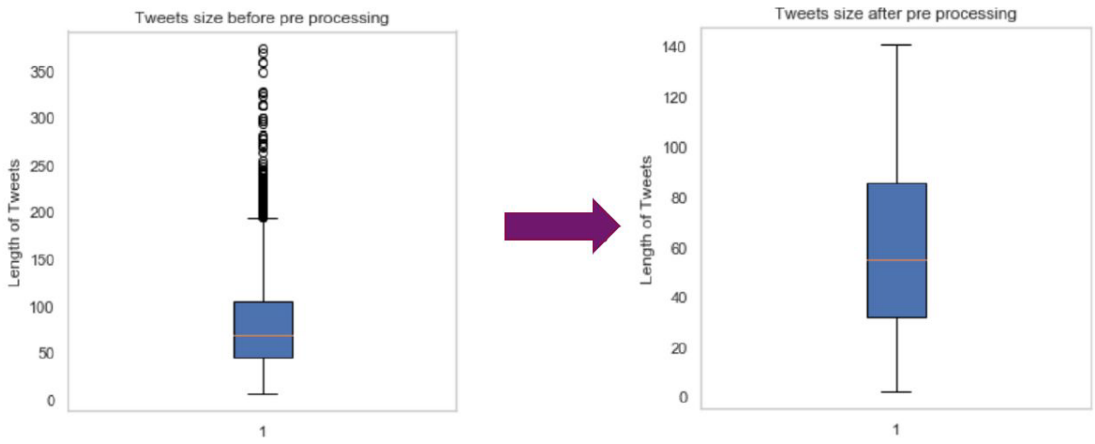

Figure 27: Change in tweet size before and after preprocessing

4.2 Dataset 2

We collect this data from Jan 01, 2017 to Feb 28, 2018 available at [18]. The data set consists of 8 fields, as shown in Figure 28:

1. Timestamp: The date on which features are collected. It follows the yyyy-mm-dd convention.

2. Tweet Volume: Total number of tweets tweeted worldwide on that day related to bitcoin. We collect this metric from bitinfocharts [19], and the values are on a scale of USD thousands, i.e., 14 is equal to 14000 USD.

3. Average Transaction Fee: The average transaction fee for all bitcoin transactions done during that day. We collect this metric from bitinfocharts [19], and the values are on a scale of BTC, i.e., 0.35 is equal to 0.35 BTC.

4. Average Transaction Value: The average Transaction value for all the bitcoin transactions done during that day. We collect this metric from bitinfocharts [19], and the values are on a scale of USD thousands, i.e., 3.5 is equal to 3500 USD.

5. BTC close price: The closing price of Bitcoin for that day. We collect this metric from Yahoo Finance [20], and the values are on a scale of USD thousands, i.e.,

3.5 is equal to 3500 USD.

6. Volume: The Total Bitcoin volume circulated for that day. We collect this metric from Yahoo Finance [20], and the values are on a scale of USD thousands, i.e., 3.5 is equal to 3500 USD.

7. Bullish: No. of Bullish tweets for that day. We collect this metric using augmento API [1].

8. Bearish: No. of Bearish tweets for that day. We collect this metric using augmento API [1].

Unnamed: 0	TWEET VOLUME(in K)	AVG Transaction Fee	AVG Transaction Value(in K)	Close	Volume	Bullish	Bearish
2017-01-01	21.0	0.350	5.80	998.325	147775008	41	10
2017-01-02	26.4	0.392	5.50	1021.750	222184992	27	8
2017-01-03	27.2	0.396	6.50	1043.840	185168000	14	14
2017-01-04	31.6	0.633	8.10	1154.730	344945984	73	22
2017-01-05	33.4	0.410	10.80	1013.380	510199008	70	41
...	...	...	...	...	...	...	...
2018-02-24	56.6	6.200	48.09	9813.070	6917929984	195	129
2018-02-25	63.3	2.700	40.20	9664.730	5706939904	144	128
2018-02-26	66.5	2.400	44.90	10366.700	7287690240	269	120
2018-02-27	64.8	2.400	45.60	10725.600	6966179840	247	122
2018-02-28	68.8	2.800	77.40	10397.900	6936189952	247	109

Figure 28: Data set 2

4.3 Dataset 3

Usually, we can extract bitcoin tweets from twitter by using Twitter API, but the timeline for analysis is from 10-01-2017 to 03-01-2018. Twitter has restrictions over mining historical tweets. Hence, we use ‘‘Get Old Tweets Programmatically’’ [21] an API to get old tweets.

As shown in Figure 29, we use 3 AWS EC2 instances to extract tweets of different months parallelly from 10-01-2017 to 03-01-2018, collecting 5000 tweets every day by running automated scripts that call the API every 5 minutes and downloads the tweets. We initially store the tweets on the local EC2 file system, then we use scp calls and download them to our local desktop and merge all the files.

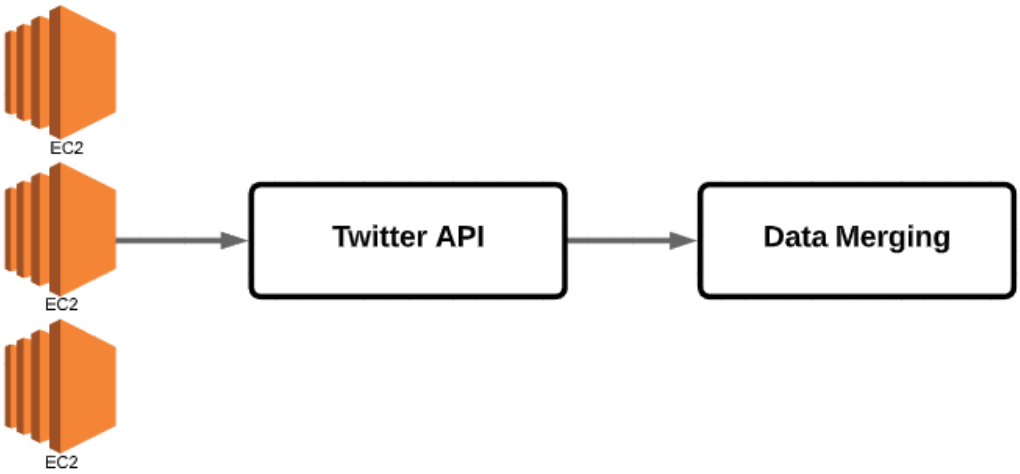

Figure 29: Data set 3 workflow

A total of 467361 are collected; we preprocess these tweets by removing tags, emoji’s, URL, and following Figure 29. Each tweet consists of twitter handle, timestamp, tweet. Using the handle, if duplicates are present, we remove them. We keep the date and cleaned tweet, as shown in Figure 30, for our experiments.

Date	Text
2017-10-01	beyond bitcoin power struggle trust based tech...
2017-10-01	bitcoinagile power struggle trust based techno...
2017-10-01	power struggle trust based technology bitcoin fyi
2017-10-01	free bitcoins faucet earn free btc minutes bit...
2017-10-01	extremely bias whole bitcoin novelty theme pla...

Figure 30: Data set 3

CHAPTER 5

Experiments

5.1 Hardware and Software Requirements

5.1.1 Hardware Requirements

We use 4 different machines for different tasks during the experiments, they are :

1. Machine 1: Processor Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz Desktop with 6 Core(s), 12 Logical Processor(s),Ram : 16Gb DDR4

2. Machine 2: Google Collab TPU enabled instance

3. Machine 3: AWS p2.xlarge instance with 1GPU, 4 vCPU, 61Mem (GiB), 12 GPU Memory (GiB), and High Network Performance

4. Machine 4: AWS t2.micro instance with 1 vCPU, 1Mem (GiB), EBS-Only Storage, and Low to Moderate Network Performance

5.1.2 Software Requirements

We use python3 as the primary programming language for the majority of the experiments. Where we use Python Packages like pandas, numpy, matplotlib, numba, sklearn, bs4, pyspark for data manipulation, and modeling. We use the Jupyter notebook to conduct all our experiments. Natural Language manipulations are done using python NLP libraries like nltk, re. For predictive analytics, we use Facebook's prophet fbprophet. BERT experiments are performed using Pytorch, where we use pytorchpretrainedbert, torch libraries. Other essential libraries are pickle, os for storing values and reading files Tweets are gathering using GetOldTweets-python [1], augmento [21].

5.2 Sentiment Analysis

We use Machine 1 and dataset 1 for sentiment analysis experiments; we split this data into Train and Test data in the ratio of 80-20 and perform 3 fold cross validation. We use Logistic Regression with TF-IDF, Logistic regression with count vectorizer, N-gram, and BERT, as discussed in earlier sections to compare their performance. We use Machine 2 for experiments related to BERT; with a configuration of BERT cased, text length 120, and run three epochs. The following results were obtained, as shown in Figure 31, Figure 32, and Figure 33.

Model	Average Validation Accuracy	Test Accuracy	ROC – AUC	Runtime (mins)
Logistic Regression + TF-IDF	87.4	85.5	79.2	37
Logistic Regression + TF-IDF+ Count Vectorization	83.6	79.4	83.8	19
N- Gram	86.8	81.2	84.2	42

Figure 31: Results for Statistical Models

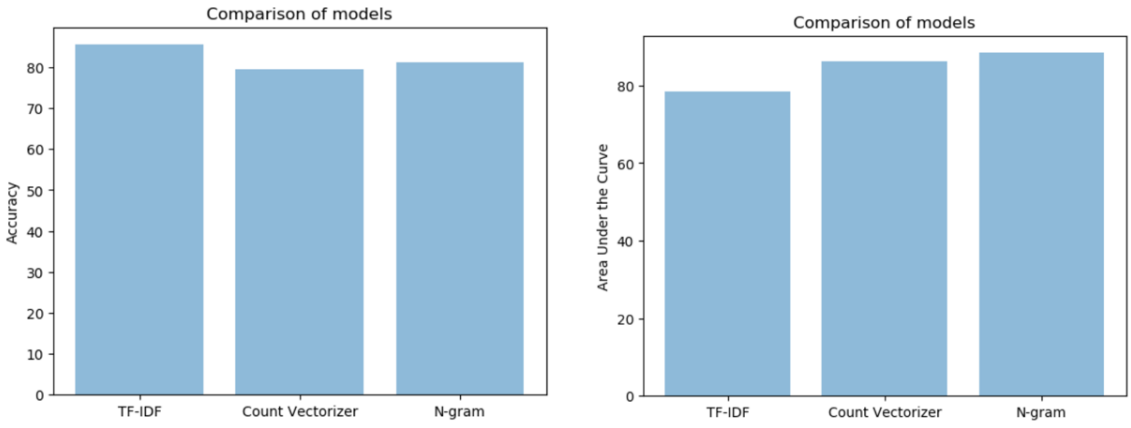

Figure 32: Model Accuracy and ROC for Statistical Models

From Figure 31 experiments, we see that although being a statistical machine

Epoch	Loss	Accuracy	Runtime
1	0.63	70.8	17h
3	0.03	71.2	51h

Figure 33: Model Accuracy and runtime for BERT

learning model, Logistic Regression with TF-IDF gave the best test accuracy of 85.5 percent with a runtime of 37 mins for classifying the tweets. In comparison, BERT, which is one of the best classification algorithms, takes significantly long training time (51 hrs) and provides an accuracy of just 71 percent, as seen in Figure 33. The Machine used to train BERT was substantially more powerful in terms of CPU computation and memory than the one used for Figure 31 experiments.

The reason for such poor performance is BERT is pre-trained using Wikipedia articles, which has well-written articles, i.e., which follows grammar rules. In comparison, the tweets are usually written using locally spoken slang like ‘‘wanna’’, ‘‘gonna’’, ‘‘YOLO’’, which generally doesn’t follow strong grammar rules making the classification problem harder.

Now dataset 3 is taken, using Machine 1 sentiment features are extracted from them using VADER and spacy NLP. Each pre-processed tweet passes through VADER and spacy functions, and their rate of positively, negativing are derived from them. At the end of this process, the dataset obtained is as seen in Figure 34.

We aggregate all tweets during a day for this data and plot them to understand how they change over the duration. As we see in Figure 35, negative sentiment dominates positive sentiment in this prepared dataset. Now we take Bullish and Bearish features from dataset 2 and see how they change during the same period. In Figure 36, we see there are a few days when Bearish sentiment is high and a few days when Bullish sentiment is high.

Date	Text	Pos_sent	Neg_sent
2017-10-01	beyond bitcoin power struggle trust based tech...	0.311	0.217
2017-10-01	bitcoinagile power struggle trust based techno...	0.411	0.185
2017-10-01	power struggle trust based technology bitcoin fyi	0.447	0.202
2017-10-01	free bitcoins faucet earn free btc minutes bit...	0.452	0.000
2017-10-01	extremely bias whole bitcoin novelty theme pla...	0.299	0.137

Figure 34: Dataset 2 feature extraction results

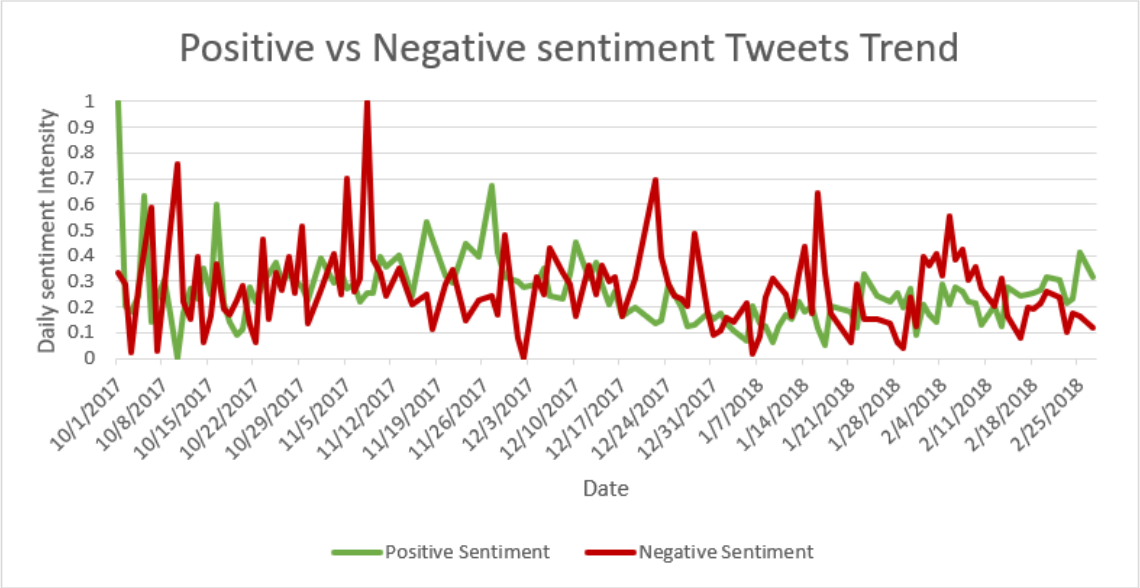

Figure 35: Dataset 2 Tweet's daily positive and negative trend

When we are comparing these two charts, in Figure 35, we don't observe any significant spikes or changes between two consecutive days, whereas there are many spikes and changes in Figure 36. Hence, Figure 36 reflects better emotions of twitter user's sentiment during this period.

Our collected dataset 3 didn't perform well because there was still noise present in them despite cleaning. This noise is instead the content of the tweet where only

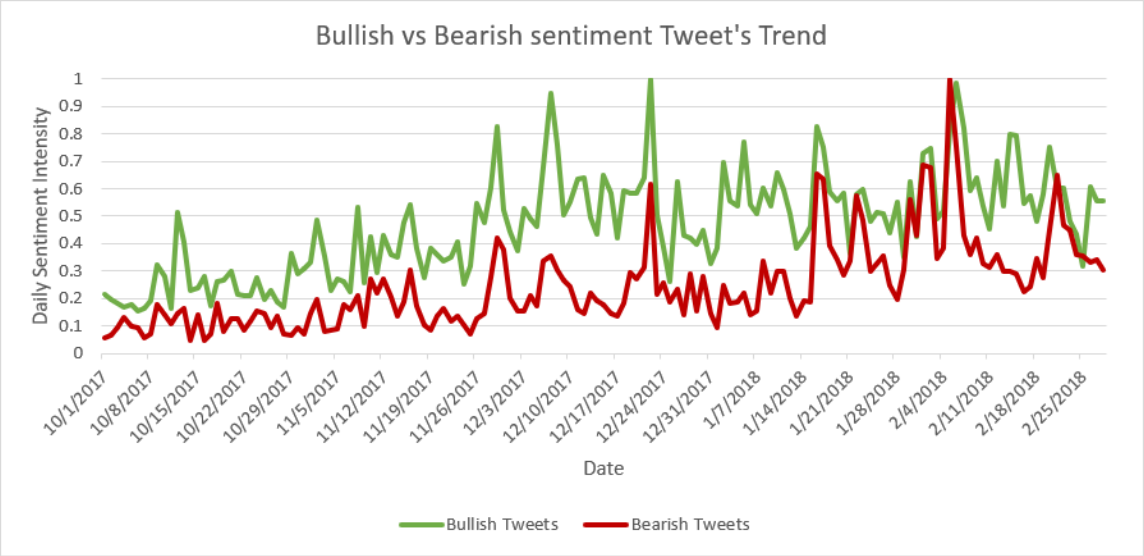

Figure 36: Data source [1] Tweet's daily positive and negative trend

general statements were made rather than specifically talking about price movement. Whereas the dataset 2 tweets were explicitly talking about price moment, making them a better metric. Hence we will dataset 2 features in the further analysis of price prediction.

From Figure 37, we can say that there seems to be some relation between sentiment, bitcoin closing price, and tweet volume. We try to understand these relations further before using this for final experiments. We combine the bullish and bearish sentiment of a day and generate a sentiment score between -4 to +4 for that day. A maximum negative score is -4, and the maximum positive score is +4 for a day. The following shows the trend between twitter tweet volume, daily bitcoin closing price, and overall daily sentiment.

From Figure 38, we can see a relation between bitcoin's price, sentiment, and tweet volume for a day. From Figure 39 we can see that

1. When tweet volume increased, and the sentiment trend was bullish, the bitcoin price increased.

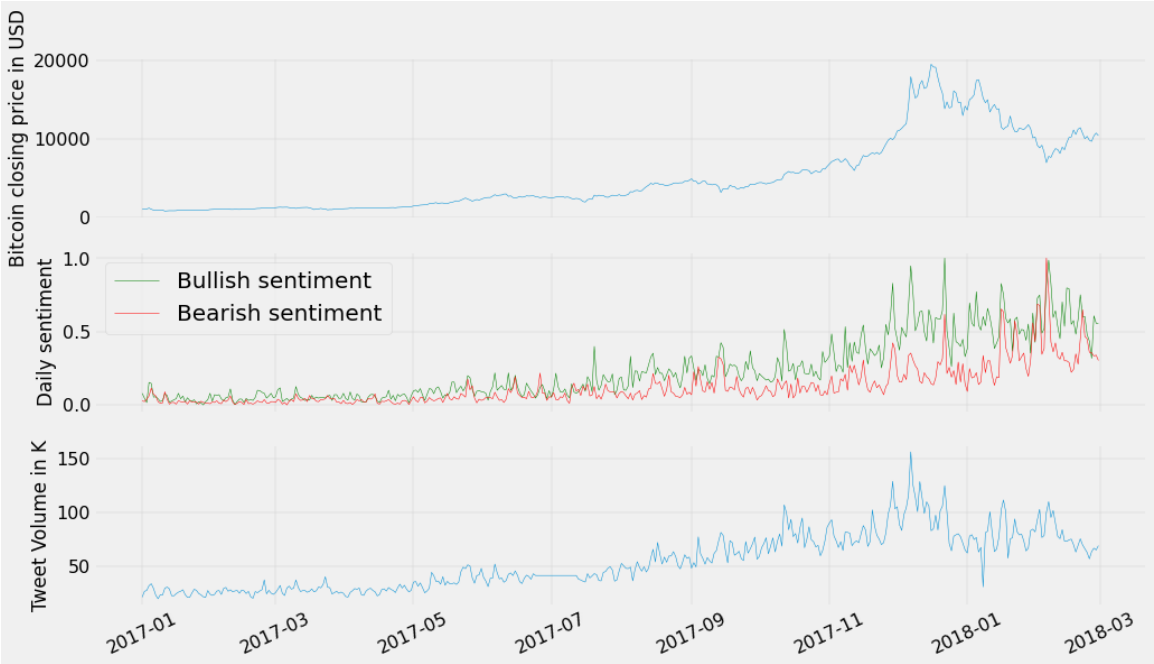

Figure 37: daily twitter tweet volume vs bitcoin closing price vs daily sentiment

2. When sentiment trend was bearish, and tweet volume was low, the bitcoin price decreased.

From these observations, we can say there seems to be some influence of these features on bitcoin's closing price. Hence, we move on to the bitcoin price prediction to build our proposed model.

.

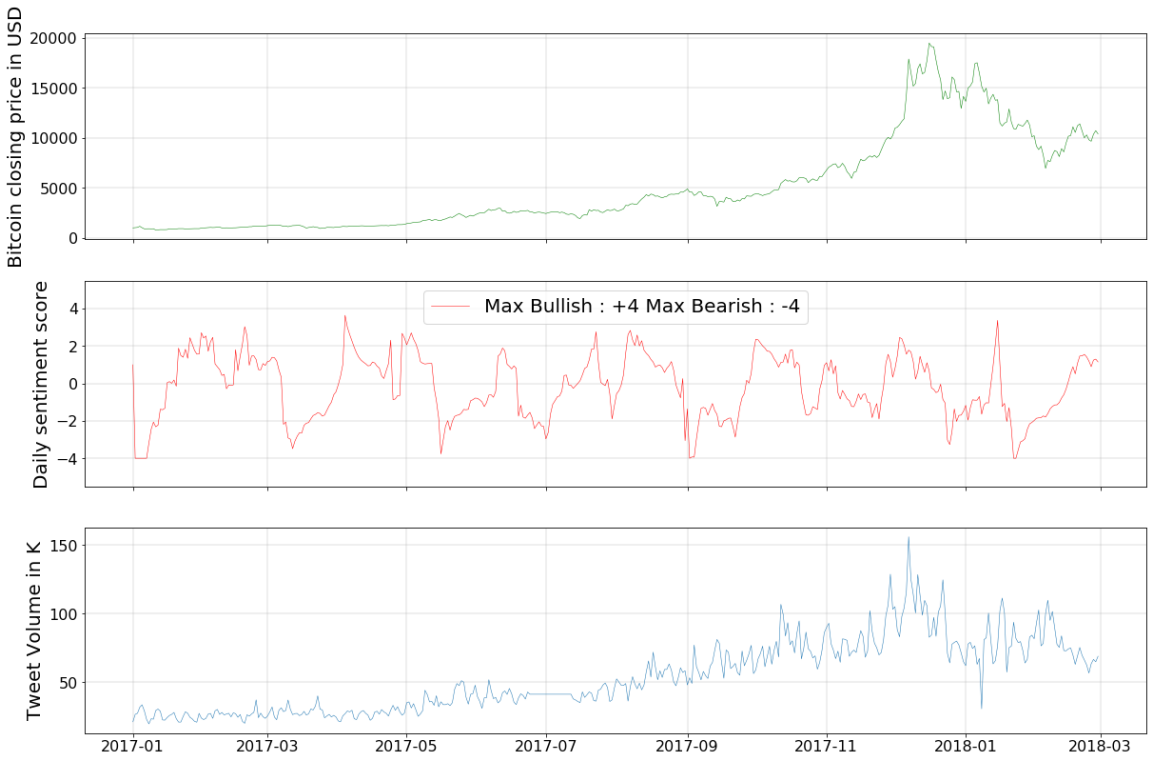

Figure 38: daily twitter tweet volume vs bitcoin closing price vs daily final sentiment

5.3 Predictive Analysis

In this section, we predict the closing price of bitcoin; for this, we first set up a base model using dataset 2 closing price, as shown in Figure 40. For this task, we use the Prophet model as discussed in the earlier section. We use the closing price of the previous day to predict the price of the next day, as shown in Figure 41.

Since this model uses sliding window N to predict the price, we test different N values varying from almost one year to one week to understand its impact on the R2 score. As seen in Figure 44, when the window is 394 days, the training R2 score was -19.7 as we decreased the window size to 274, the R2 score improved to 0.22. Further, when we reduce the window size to 7 days, we notice the best training R2 score of 0.79. We see a trend where decreasing the sliding window increases the R2 score.

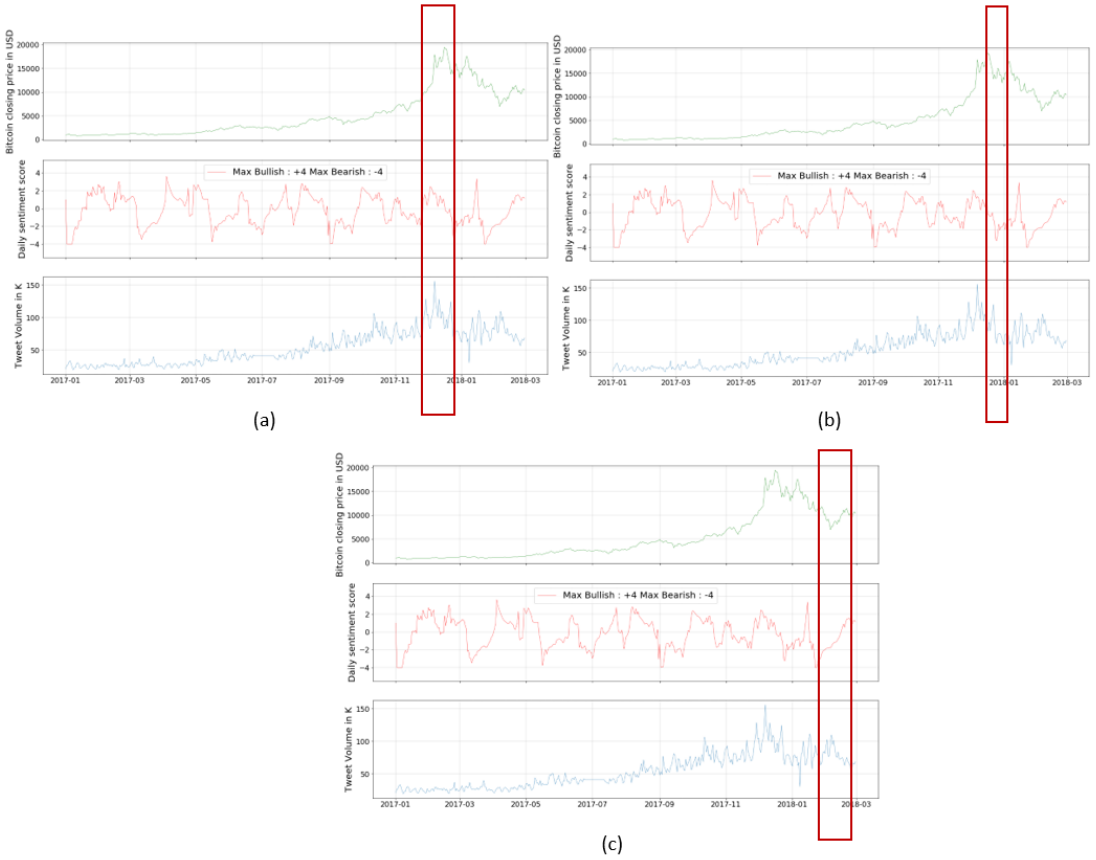

Figure 39: (a) ,(c) When tweet volume increased and the sentiment trend was bullish, bitcoin price increased ,(b)When sentiment trend was bearish, and tweet volume was low, bitcoin price decreased

Date	Close_price
2017-01-01	998.325
2017-01-02	1021.750
2017-01-03	1043.840
2017-01-04	1154.730
2017-01-05	1013.380

Figure 40: bitcoin closing price in USD

When the window size is 7 (one week), the model captures only daily trends while predicting the closing price of bitcoin, but while predicting the closing price, we want

actual_price	predicted_price
9813.07	12871.214803
9664.73	12669.150745
10366.70	12226.157064
10725.60	11921.638803
10397.90	11810.070003

Figure 41: bitcoin closing price predictions in USD

to consider weekly and monthly trends as well. We try different values for window N, as shown in Figure 42 based on the experiment results, we take 90 days (approx. 3 months) as the window size, capturing 3 months, 12 weeks price trends. So our final baseline model will have a training window size of 90 and a training time of 300 days. We test all the models on February 2018 predictions, so our test set would have 28 values.

Figure 42: Training R2 score for N window sizes

We see the training and test results of our baseline model in Figure 43 and Figure 44. The red line is the actual price, and the blue line is our model's predicted price, as we can see in Figure 43 for the first 150 predictions, our model predicts well since there are no exponential spikes. Later, when there were exponential rises, our model was able to catch the changing trend, but it failed when there was an exponential fall.

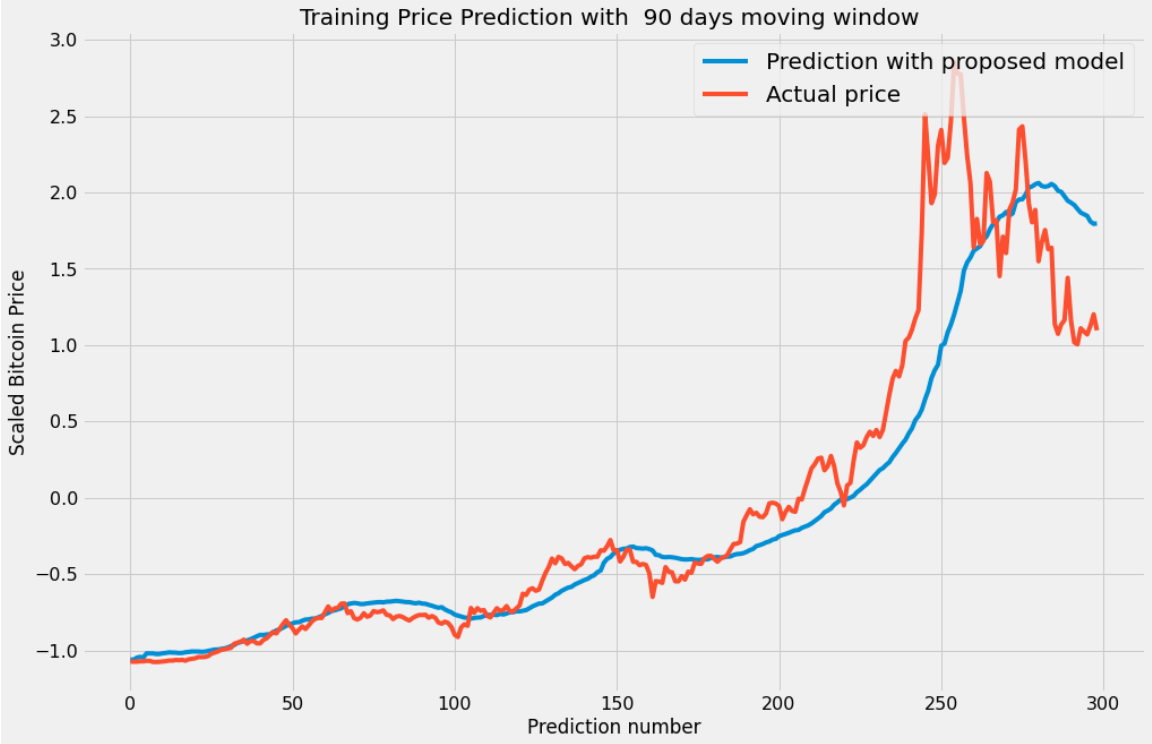

Figure 43: Training Price Predictions for N = 90 window size

We can see in Figure 44 that our model performs poorly on the Test set, it achieved an R2 score of -19.93. This score becomes our base model score that we try to improve by adding the sentiment, and additional features from dataset 2.

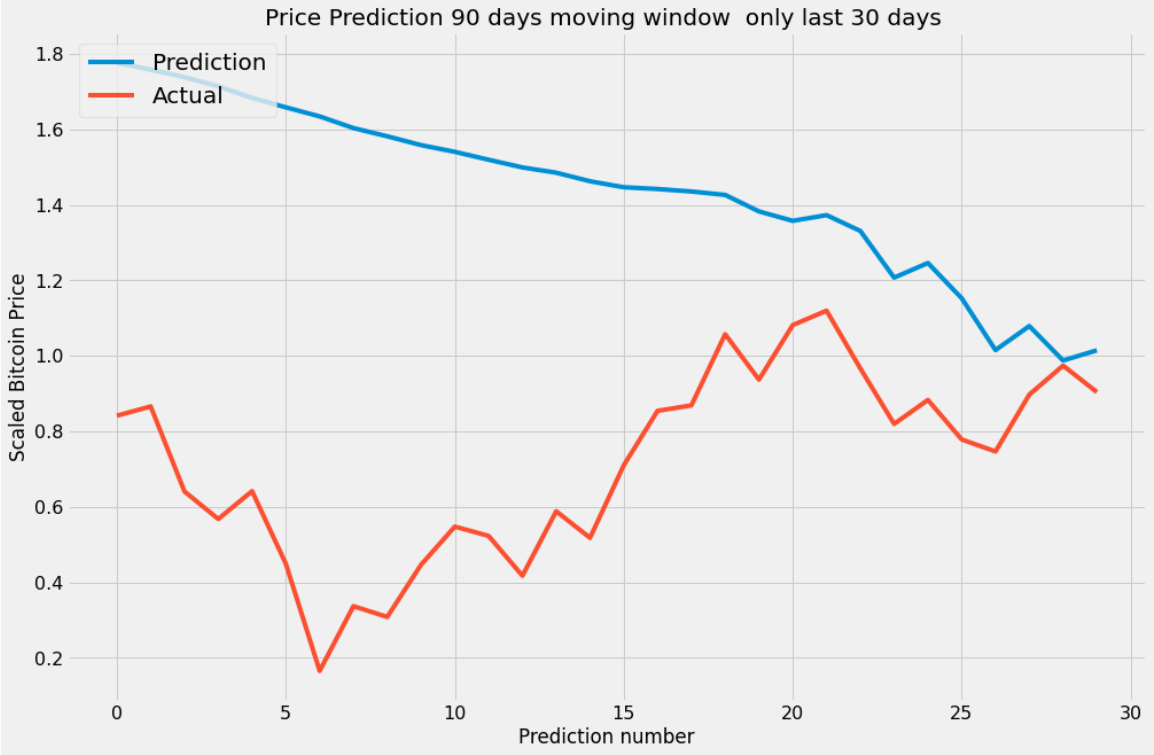

Figure 44: Test R2 score for N = 90 window size

5.4 Proposed Model

Using dataset 2 and Machine 3, we follow the recursive additive approach, where we first choose a combination of features, then using these features we train the SVM model and predict for test data. Using gridsearchCV, we tune the SVM hyperparameters to optimize and get the best R2 score.

As shown in Figure 45, we try RBF and sigmoid SVM kernels, try different values of C, gamma, and epsilon values to find the optimal values.

```
parameters = {'kernel':['rbf', 'sigmoid'],
              'C':np.logspace(np.log10(0.001), np.log10(200), num=20),
              'gamma':np.logspace(np.log10(0.00001), np.log10(20), num=30),
              'epsilon':np.logspace(np.log10(0.00001), np.log10(20), num=30)}
```

Figure 45: SVM Parameter Hypertuning

Sentiment and predicted prices are fixed features for all models; we keep adding additional features to them. As we see in Figure 46,

1. When we use only one extra feature, Average Transaction Fee, the best R2 score obtained is -7.67.

2. When we choose two features, Average Transaction Fee and Tweet Volume, the best R2 score obtained is -2.9.

3. When we choose three additional features, Average Transaction Fee, Twitter Volume, and Average Transaction Value, we obtained the best R2 score of -1.938.

From Figure 47, adding the sentiment and additional features improved our R2 score. Comparing it to Figure 44, the R2 score for Figure 47 is approximately 90 percent lesser than our base model R2 score. It reduced from -19.8 is -1.938.

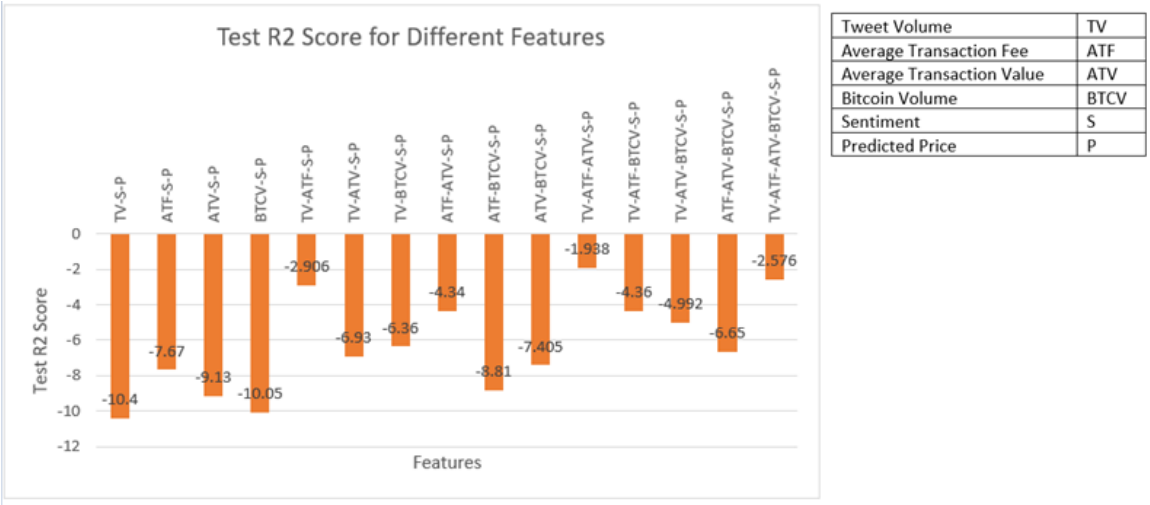

Figure 46: Test R2 Score for Different Additional Features

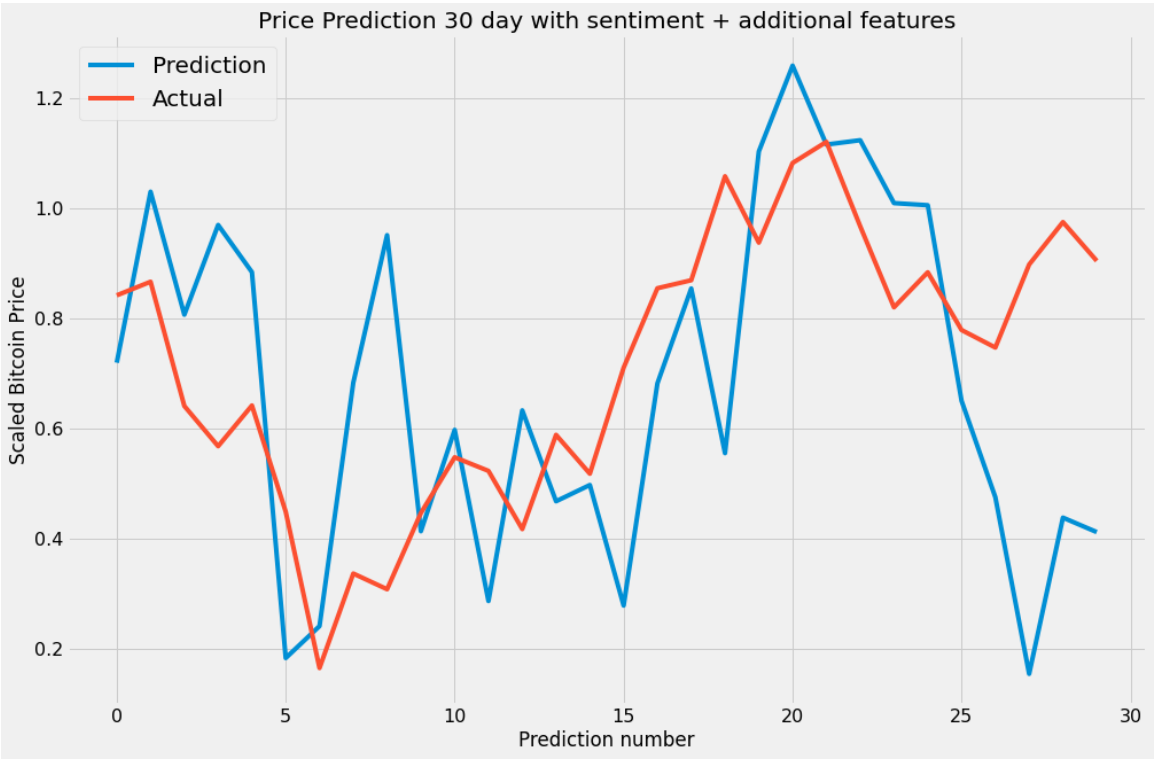

Figure 47: Test R2 score for N = 90 window size with best features

CHAPTER 6

Conclusion and Future Work

In this paper, we discussed the types of Text classification models, their advantages, and disadvantages. We performed experiments to compare the statistical model and advanced BERT model for sentiment classification. For dataset 1 twitter tweets, the results of the experiments show that the statistical model performed well, whereas BERT took comparatively larger training times with less accuracy.

We have performed experiments on a novel approach to predict the bitcoin prices using information from both predictive analysis and sentiment analysis. We performed the numerical analysis using Facebook's prophet model, where a base model gave us an R2 score of -19.8 for a sliding window of N = 90. We observed that using twitter tweet feed related to bitcoin to our proposed model, we were able to improve the R2 score. Also, utilizing additional features of dataset 3 improved the R2 score to -1.938 R2, as shown in Figure 48. Hence we observe that adding textual information from twitter to the bitcoin price data can greatly improve the prediction accuracy.

For our experiments, we took a time frame of 14 months; the results of predictive analysis can be improved by using more training data having larger time frames. We can try considering weekly, monthly moving averages and see if this improves the bitcoin price prediction. Also, in textual analysis, improved training data can be collected; for example, instead of randomly extracting tweets from API, we can try collecting opinions of certain users only and see how their sentiment is changing over the time. Additionally, instead of just looking at the current day's sentiment as used in our model, we can try using a wider window sentiment and see if it improves the model.

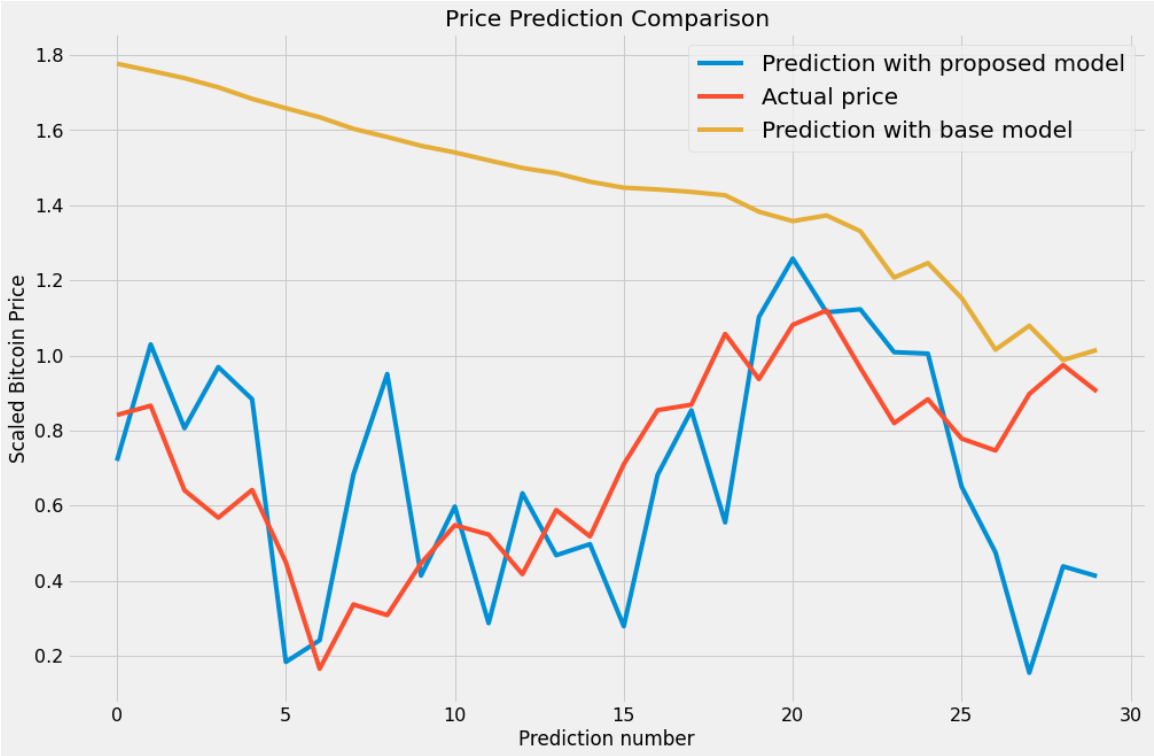

Figure 48: Comparing Base Prediction Vs Actual Vs Proposed Model Prediction

LIST OF REFERENCES

[1] [Online; accessed 6-April-2020]. [Online]. Available: http://api-dev.augmento.ai/v0.1/documentationsources

[2] D. Kaloti, ‘‘The rise fall of bitcoin,’’ [Online; accessed 6-April-2020]. [Online]. Available: https://medium.com/datadriveninvestor/the-rise-fall-of-bitcoin-cebe115eb17

[3] Wikipedia contributors, ‘‘Facebook --- Wikipedia, the free encyclopedia,’’ 2020, [Online; accessed 6-April-2020]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Facebook&oldid=949275207

[4] ‘‘Cs224d stanford,’’ 2020, [Online; accessed 6-April-2020]. [Online]. Available: https://cs224d.stanford.edu/lectures/CS224d-Lecture2.pdf

[5] Oinkina, ‘‘Understanding lstm networks,’’ 2015, [Online; accessed 6-April-2020]. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] ‘‘Cs224d stanford 2,’’ 2020, [Online; accessed 6-April-2020]. [Online]. Available: https://cs224d.stanford.edu/lectures/CS224d-Lecture5.pdf

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, ‘‘Attention is all you need,’’ in *Advances in neural information processing systems*, 2017, pp. 5998--6008.

[8] J. Alammar, ‘‘The illustrated transformer,’’ [Online; accessed 6-April-2020]. [Online]. Available: http://jalammar.github.io/illustrated-transformer/

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘‘Bert: Pre-training of deep bidirectional transformers for language understanding,’’ *arXiv preprint arXiv:1810.04805*, 2018.

[10] S. I. S. (WS19/20), ‘‘Bidirectional encoder representations from transformers (bert),’’ 2020, [Online; accessed 6-April-2020]. [Online]. Available: https://humboldt-wi.github.io/blog/research/information_systems_1920/bert_blog_post/

[11] mathworks, ‘‘Predictive analytics 3 things you need to know,’’ [Online; accessed 6-April-2020]. [Online]. Available: https://www.mathworks.com/discovery/predictive-analytics.html

[12] WSL, [Online; accessed 6-April-2020]. [Online]. Available: https://www.wsj.com/market-data/stocks/marketsdiary

[13] S. J. Taylor and B. Letham, ‘‘Forecasting at scale,’’ *The American Statistician*, vol. 72, no. 1, pp. 37--45, 2018.

[14] V. N. Vapnik, ‘‘The nature of statistical l earning t heory,’’ *New York: Springer2V erlag*, 1995.

[15] S. RAY, [Online; accessed 6-April-2020]. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[16] S. Girgin, ‘‘Support vector regression in 6 steps with python,’’ [Online; accessed 6-April-2020]. [Online]. Available: https://medium.com/pursuitnotes/support-vector-regression-in-6-steps-with-python-c4569acd062d

[17] E. Rieuf, ‘‘How to interpret r-squared and goodness-of-fit in regression analysis,’’ [Online; accessed 6-April-2020]. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/regression-analysis-how-do-i-interpret-r-squared-and-assess-the

[18] A. Deebadi, [Online; accessed 6-April-2020]. [Online]. Available: https://medium.com/datadriveninvestor/the-rise-fall-of-bitcoin-cebe115eb17

[19] ‘‘Cryptocurrency statistics,’’ [Online; accessed 6-April-2020]. [Online]. Available: https://bitinfocharts.com/

[20] [Online; accessed 6-April-2020]. [Online]. Available: https://finance.yahoo.com/quote/BTC-USD/

[21] J. Henrique, ‘‘Get old tweets programatically,’’ [Online; accessed 6-April-2020]. [Online]. Available: https://github.com/Jefferson-Henrique/GetOldTweets-python