University of East London

Pioneering Futures Since 1898

SCHOOL OF ARCHITECTURE, COMPUTING AND ENGINEERING

Department of Engineering and Computing

Cryptocurrency Prediction using social media and machine learning

Rohan Parekh

1930306

Presentation Link: https://youtu.be/YvrRuRsmpEQ

A report submitted in part fulfilment of the degree of

BSc (Hons) in *Computer Science*

Supervisor**:** Dr Mustansar Ali Ghanzafar

CN6000

9 May 2022

Abstract

As we see that today's dynamics are changing in the space of finance and technologies where we see everything converting to digital, where trading follows the digital method and traditional banks are changing to web-based and application-based. I realised that cryptocurrency had become the word of the market, primarily because of the hype bitcoin created, which made me realise that it is not the same as an everyday currency. Also, it had high volatility to it. During the pandemic, I found that everybody had jumped the social media wagons where all the tiniest thoughts and details from everyone's daily life were posted. It felt like it played a significant role in all the decisions made by the people, which is why we see social media marketing playing more prominent roles better than ever to get hold of customers from their interaction throughout. This project focuses more towards the impact of sentiment towards bitcoin prices with different algorithms. For research method I follow quantitative analysis where I first search for dataset then process it and for bitcoin prices I fetch it using API, for the development of the model I follow Agile SDLC. Based on that scenario, I create multiple machine learning models to predict bitcoin prices, with the inputs being historical data and tweets. I compare the output accuracy for different models and present the input features' impact on the model. The model I used are based on TensorFlow API with keras conjunction and use of Sequential model .Overall I create four other models where. Three are multivariable regression models, and one is a binary classification model. In all the models, the result concluded that the tweets impacted the total output of the prediction model. Although we see the overall accuracy getting a dip in the last two models, it shows the impact of tweets having higher accuracy than the model without the tweets. From all the results, I found out that twitter sentiments impact the model without any feelings. We see that the first two regression models that follow same day prediction have the highest accuracy for TextBlob and the latter two models have the highest precision from Vader Sentiments. Only the last model tries to predict the future difference between the close price. This project in future can be improved to create better accurate predictions , while at the same time improve the understanding towards the use of machine learning and impact of it on economy .

Acknowledgements

I want to thank my supervisor Dr Mustansar Ali Ghazanfar, for guiding me in the right direction for this project and continuously pushing me to create the best possible output.

Contents

Chapter 1: **Introduction**

There are different ways in the market to pay for the items. Wherein few of them are via mobile phone, internet and digital storage card. These payment methods are based on fiat currency, or one may say it as a stable currency, but as we see the drastic growth of digital currency, which allows for faster transfer along with many other innovative ways to pay for the goods, is starting to have a significant impact around the globe. (Nian, L.P. and Chuen, D.L.K., 2015)

Cryptocurrency markets have grown drastically and at a tremendous rate in a short time. Since the arrival of the trailblazer anarchic digital money, Bitcoin, to the general population in January 2009, a more significant number than 550 digital currencies have been created, the more significant part with just a bit of achievement. Research on the business is still scant. Most of it is independently centred around Bitcoin rather than a more assorted spread of digital forms of money and is consistently being outperformed by liquid industry advancements, counting new coins, mechanical movement, and expanding unofficial law of the business sectors (Farell, R., 2015)

1.1 Purpose of the Project

We realise that people spend more and more amount of time on social media, where the big companies use multiple algorithms to influence people's decisions on a day to day basis. As researched by Lee E., in the year 2013 where it was found that social media creates a two-way relation between the consumer and the company as thus this can have a chance of impacting people's decisions, although it was found that they still go through all the stages of decision making and that did not increase the speed of decision making to access this more relation information, it has a higher chance to affect the decision of a consumer. I believe that there is an influence on people's sentiment of engagement towards the price of the cryptocurrency, which gets more hyped on social media platforms as compared to the stock market and other exchange markets.

1.2 Problem Statement

From all these years, we see that the cryptocurrency industry has become a place of high volatility in nature, which makes it more challenging the prediction of prices. And the inclusion of social media in the day to day life of people can have an impact on the way decisions are made.

1.3 Aim of the project

The aim of this project is to design and implement a web-based prediction system for cryptocurrency and the stock market based on understanding the emotions and feelings of people, getting that to a numerical value and using that as a major input for prediction. With this goal, we will be able to find out if the sentiments of tweets have any impact on our prediction model or not.

1.4 Objectives of the Project

- To research and find a dataset that needs to be used in the project and different software related to prediction

The preparation before the implementation where the collection or fetching of data, finding and learning out what will be required for the implementation.

- To investigate and identify the most appropriate tools and API required for the project

Most appropriate tools and APIs are required to understand and deploy the algorithm, as without understanding what an API is doing or running, it might get difficult for the model to process data. Such as, there might be an instance where there is some background processing going on while

- To research and identify the relationship between social media sentiments for crypto market prediction

Social media in today's world is growing rapidly, and so identifying and researching the impact, especially in the crypto market, is volatile.

- To develop a working demo of the project and to analyse and evaluate the demo providing the possible chances of being accurate

With the help of a working project, I can find and evaluate the impact, if there is any, especially by counting tweets as one of the input features along with other historical data.

- To conclude the model, show what could be the possible outcome if implemented in a real-world scenario

Providing the outcome of this project and giving what could be changed or implemented.

Chapter 2: **Literature Review**

Cryptocurrency term is an excitement that is constantly evolving and exciting in today's world. It can be briefly described as digital currency based on blockchain. This concept of having something digital to buy and spend has drastically changed the way a person thinks. The rapid growth and usage have increased value in the current world scenario by a large quantity. There have been many options in the present day, but the first one was Bitcoin, basically digital cash without any governing body. Because of their high value, they have their own trading market and are not seen in online merchant transactions. (Pant, D.R., et al., 2018,)

2.1 Cryptocurrency until now

Satoshi Nakomoto was the founder of the program which had the possibility to run most of the ongoing cryptocurrencies along with world famous bitcoin. The concept behind this was to create a payment platform which could perform n number of transactions without the need of an intermediary like a bank. It was the most secure way to do the transactions as it did not required any sensitive data input from the user and still perform the transaction securely (Lenton G, 2021).

Cryptocurrency is the most unpredictable type of payment in the market as it is not been surveillance by the investor regulators or not related to fundamental values it follows into the direct path of Hierarchical Risk Parity (Papenbrock J, Schwender P,and Sandner PG, 2021)

Because of cryptocurrency's dynamic volatility it becomes slightly difficult to create an accurate prediction (Martin J, Cunliffe, et.al 2019)

2.1.1 Bitcoin in essence

Bitcoin being first introduced in 2008, was one of the first peers to peer open-sourced digital currencies, which was developed by a pseudo developer named Satoshi Nakomoto. In addition to that, it was, in all essence a chain of digital signatures which created a blockchain to remove the middleman which use to trust and verify the authenticity of the transaction which added the benefit of fewer transaction fees and less latency.(Farell, R., 2015)

Recently Bitcoin has received so much of attention on various fronts. It is based on exciting technological answer to various problems and function in a distinct way than the traditional methods. Based on current day scenario the usage is low as of now and the upcoming time looks very uncertain although being very interesting (Segendorf, B., 2014). Apart from relying on peer to peer network, it depends on open source software, i.e. a, software code which is generally available or has almost no copyright restriction. They are created to provide the answer to a computation problem by identities known as 'miners'. Being able to provide the solution to the question gives proof of work which authenticates that the miner was able to do the job. Other identities are able to authenticate at low cost that the answer has been given, but duplicating the work is not for low cost. The increase in complexity of algorithm is set to increase the cost over time, with an eventual restriction on the number of coins that can be generated.(Dwyer, G.P., 2015)

2.1.2 Other Cryptocurrencies apart from Bitcoin

Even though bitcoin was one of the first cryptocurrency it was not a complete one, it had its own flaws which got solved by a whitepaper which was released by Vitalik Buterin in the year 2014 known as Ethereum which incorporated A Turing Complete Programming Language, allowing users to make a logic which could run on open source blockchain. Ethereum covered the flaw of allowing the code to be executed and ownership to be recorded which was not in the case of Bitcoin. With this

powers developers were able to design and implement pieces of codes known as smart contracts that could work on blockchain nodes around the world. these smart contracts are supposed to be pre-defined code specified agreements without the need of trust from any single party or authority.(Phillips, R.C., 2019)

Dogecoin was launched being the 'joke currency' in 2013 currently having a market capitilization of 10 billion in 2021 up from 60 million USD in 2014. The construction of Dogecoin is based on the ongoing cryptocurrency known as Luckycoin which follows Litecoin based on scrypt technology. Previously it provided a randomised award for mining but later in 2014 it was converted to static block reward. These coins use scrypt technology for proof of work verification as oppose to SHA-256 bitcoin mining. The original launch supply for Dogecoin was constricted to 100 billion coins which was afterwards increased to infinite number which could result in inflationary momentum. (Chohan, U.W., 2021)

2.1.3 Why choose Bitcoin or why it is so hyped

Cryptocurrency such as Bitcoin have a bigger issue regarding bits which are easy to create and copy on digital devices. As these are harder to recreate it becomes easy with the physical ones, an answer to this could be having a central authority that looks after or governs all the transactions of the currency and authenticates them. Bitcoin solves this problem by having a peer to peer network where even though there is no central authority for verification but is an open source method from where it makes it alike peer to peer network as everyone can look and edit the code and the progress are done by participants and not a central governing body(Dwyer, G.P., 2015) and because of this the verification and authentication problem is been resolved by Bitcoin. Two other reason that Bitcoin has value is because that it has low transaction charges and it has an anonymity. (Árnason, S.L., 2015)

From the beginning of 2009 being the open source cryptocurrency bitcoin has grabbed the notice of economists, traders and policy makers which assisted it in dominating the financial market which caused exponential surge in market value and transactions. As the time passed bitcoin gained tremendous market value as in the year it was below $ 7 billion in 2015 and $ 16 billion in 2016 but later surpassed $ 216 billion in the year end of 2017. (Bouri,et al ,.2019)

Bitcoin is the most openly used and available type of digital crypto currency with approximately 13 billion total bitcoins in circulation and these can never exceed the limit of 21 million as this is the total limit sent to mine the about of bitcoins.(Luther, W.J. and White, L.H., 2014)

2.2 Cryptocurrency trading market

As the cryptocurrency attracts many new customers there is an urgency to launch a self-maintained market to handle these virtual assets or as one call it currencies whose amount is determined by the social conseus has gained an interest of the research community. (Alessandretti, L., et al, 2018)

We have seen in the last 2 years the burst of cryptocurrencies as there was sudden creation of many cryptocurrencies and in the year of 2018 it was mainly guided by the chance created by Initial coin offering technique used by the organisations as a means to support their growth and innovation. Along that the expansion follows the new business models based on the cryptocurrencies and their relative blockchains. In the last few years the market is been dominated by these 5 cryptocurrencies named Bitcoin, Bitcoin Cash, Ethereum, Litecoin and Ripple. In total there are around 15 currencies with a total capitalisation of over 1 billion USD. Being the new and volatile market it creates more confusion and after some sudden rise in the values, often a quick decrease in their values is seen till failure. This is a market which is frequently repeated in social media with some substantial expectations and rapid change in sentiments, hard beliefs and rough debates. (Aste, T., 2019)

2.3 Influence of Online Indicators on Economy

Social media acts as a tool for the conversations to reach at a higher level following the concept off long tail which describes as communications that can be transferred to different channels. There can be many channels in an organisation via different route such as messages, calls and emails .They have their own disadvantages such as unable to take notes in a meeting or not following up on a message, or scrubbing through the long pile of emails in search for a particular information. On oppose to those downsides there are some upsides as well to these communications such as it increases the engagement and collaboration between the employees of the company. It can be associated as a generator of different and unique ideas, through the right use of this technology and collaboration one can increase the effectiveness of the whole group. Once everything works in perfect sync it helps in transformation into a reputed brand name. Accordingly social media can assist in forming a valuable recognition for an organisation, small number of words are required to explain the brand either in the consumer market or in the business area. With the guide of creating a brand it assists in strengthening the brand name, thus increasing the brand value. Customer feels the brand a luxury while consuming that product or service and also around the experience of the company. (Carraher, S.M., et al., 2006) To make the consumers aware of the brand, social media can act as a powerful tool to enhance the channel to express the brand value and their attributes as they provide open conversations, hence providing a better chance for the business to grow. (Edosomwan, S. et al., 2011)

2.3.1 Twitter's influence on overall crypto markets and stock markets

Twitter being the microblogging site for the users to communicate and social network. It allows them to share small text often described as Tweets, which are available at Twitter's own website, tags such #cryptocurrency and #bitcoins are used to associate tweets regarding cryptocurrency. (Laskowski, M. and Kim, H.M., 2016) Twitter was launched in 2006 in the month of July by Jack Dorsey, by now it has more than 645750000 registered users and is now in the top 10 of the most visited sites. As the business industry is related it can be used to broadcast news, posts, articles, comments of customers. (Matta, M., Lunesu, I. and Marchesi, M., 2015)

During the presidential campaign of Barack Obama in the US social media proved to be integral components for the campaign tools. Just sometime after the victory Mr. Obama expressed himself stating "This is history", this example makes it clear of how much Twitter has grown to become a recognised means for the communication channels around the world. (Tumasjan, A., et al, 2010)

2.3.2 Evaluating the sentiments on twitter for the use of prediction

Understanding people's emotion has been one of the recognised topics for research and sentiment analysis is utilised to acquire helpful knowledge across business industries and most quite in financial analysis and digital marketing. (Stenqvist, E. and Lönnö, J., 2017)

In a research presented by Georgoula et al in year 2015 with the help of sentiment analysis, by creating various regression models and Support Vector Machine were implemented to forecast price changes for Bitcoin. Writers found only a limited time period of co-relation between positive twitter sentiment and value of Bitcoin, along while achieving 89.6 percent of accuracy for their model.

In a different research identified by Mata et al there was a connection found between the value of Bitcoin and the quantity of tweets or the final results of Web Search Media. (Sattarov, O., et al.,2020)

2.4 Different Neural Language Processing

The procedure to build computer applications for neural language processing involves 3 major issues at the core the first one is related to the thinking process, second one is towards expression and the importance of language input and the third can be described as knowledge of world.

2.4.1 TextBlob

TextBlob can be defined as a python library used to mine text, process them and further analyse it for the developers in python environment. It follows sentence level analysis. Initially it takes the whole dataset as input then it breaks into sentences. A normal way of finding the polarity of positive and negative sentences and decide the overall outlook of the data can be done by counting the total positive and negative sentences. With the help of sentiment() function polarity and subjectivity is figured out. The range of polarity is from -1 to 1 and for subjectivity is 0 to 1, 0 being more objective and 1 being more subjective. (Bonta, V. and Janardhan, N.K.N., 2019)

2.4.2 Vader

Vader which stands for Valence Aware Dictionary for Sentiment Reasoning. This lexicon provides one of the best performances specially for the use in social media domain. A sentiment lexicon is a collection of lexical features (for example, words) that are classified as positive or negative depending on their semantic orientation. Manually constructing and validating such lists of opinion-bearing attributes is one of the most time-consuming approaches for generating accurate sentiment lexicons, even though they are sturdy and powerful methods. (Liu, 2010)

The final values from Vader's basic rule-based approach are on line with the sophisticated benchmarks. The simplicity of Vader overrules these advance machine learning formulas having multiple advantages. First it is fast and computationally low budget, without loosing much towards accuracy. Implementing it on average modern laptop a corpus that a part of a second to analyse with Vader, can take hours when implementing with more complex models similar to SVM (Support Vector Machnie) if model has been trained previously. Apart from that the lexicon and the regulations followed by Vader are not contained in a machine accessible black-box, instead directly available making it easily inspectable, modifiable or even scalable. It allows everyone to more access to the inner working of sentiment analysis engine by exposing lexicon and rule based model, allowing access to wider audience which can go above the computer science groups and society. Along to that people included from different occupation such as psychologist, sociologist, linguistic experts who are familiar with LIWC (Linguistic Inquiry and Word Count) should have no problem understanding Vader. Third benefit is by facilitating a general sentiment lexicon, and rules using grammar and 224 syntax, it is considered a self contained. VADER should be accessible to sociologists, psychologists, marketing researchers, and linguists who are familiar with LIWC. Third, VADER is both self-contained and domain agnostic, since it uses a broad (human-validated) sentiment lexicon and general grammar and syntax rules. It does not need a large collection of training data, yet it works well in a variety of domains. We want to be clear that we don't want to imply that elaborate or advanced procedures are inherently harmful or improper. Instead, we demonstrate that a straightforward, human-centric, interpretable, and computationally economical technique may provide high-quality findings, even beating individual human raters. (Hutto, C. and Gilbert, E., 2014)

2.5 Machine Learning in Cryptocurrency

Machine learning helps in finding out the trends in the current ongoing market. There have been many areas on how the machine learning functions and performs in the current market where some of them. As we see in this article the author tries to forecast the market trend pattern instead of the future prices of that particular stock due to which the accuracy rises to 79%. (Velankar, S., Valecha, S. and Maji, S., 2018)

There have not been much research for predicting the value of bitcoin with the application of machine learning algorithms, Devarat Shah along with his partners in 2014 created a model that created to forecast the price of Bitcoin providing 89% in return in 50 days, this model also had a sharpe ratio of 4:1. Along with this there are many other articles that have been here to evaluate the price of bitcoin with the assist of sentiment analysis (McNally, S., Roche, J. and Caton, S., 2018)

2.5.1 TensorFlow API with Keras

TensorFlow can be described as a toolkit that can be used to create and run machine learning algorithms. Tensorflow.js models run in a web browser and in node.js environment. TensorFlow library is a set of APIs and it can be used to interchange models between these two languages. There can be machine learning models build and deployed by community which can facilitate new type of on device computing. (Smilkov, D., et al, 2019)

Keras can be described as a neural network library written in python that can be used for high level deep learning. It is free and open source and it is built on top of TensorFlow and Theano. It was created with the goal to empower people to write their own programs without requiring to write backend in deep. Some of the features provided by Keras framework are used to provide simple interface giving advantage of easy and quick prototyping, making it run flawlessly on both the CPU and GPU, other advantages include easily extensible and scalable, user friendly, modular design. It adds additional support for all models of neural network along with the ability to combine different models to combine into a single complex one. Uses for Keras include classification, text generation, speech recognition. Major drawback for Keras include limited support as it only works with TensorFlow, Theano and CNTK backend also it has less functionalities as compared over TensorFlow, thus making it difficult to design a specialised model, and so to decrease the effect it is recommended to used Keras plus TenorFlow. (Nagisetty, A. and Gupta, G.P., 2019)

2.5.2 LSTM

It is the dependent learning variant of the current RNN which stands for recurrent neural network architecture. LSTM memory block consist of memory cell storing information taken from previous time step through self recurrent connections. There are three main gates governing the cell which are input gate, forget gate and output gate. Sigmoid layer is used by forget gate to choose the information preserved or removed from the cell, on the other side the input layer use the tanh layer and a sigmoid to handle value updates. The output gate uses a sigmoid function to find out memory contribution to the cell output then followed by tanh activation function. The capability of LSTM are much higher when it comes to learn non linear statistical and temportal relationships of real world time-series data. (Masum, M., et al., 2020)

2.5.3 Linear Regression Model

First there is regression analysis where it can be sub divided into two different categories wherein the first can be linear regression analysis. To understand we can start with the simple formula where we can see in the figure that Y represents the dependent variable which will be the predicted outcome and b represents the slope which is the dependent input that is directly proportional to the output and a being the intercept or slope.

$$Yi = a + bXi$$

Fig 2.1 *Formula of Linear Regression Source (Alexopoulos, E.C., 2010.)*

Linear Regression generally has just one input and one output but it can be adapted to take multiple inputs and produce one output which is also known as multivariable regression model.

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta\rho X\rho + \sigma(Y), \; sd(Y) = \sigma \text{ (independent of X's)}$$

Fig 2.2 Multiple Linear regression Analysis (Alexopoulos, E.C., 2010.)

In this fig 2 which represents with the multiple input where Y is the mean of the normal distribution, X are the inputs and β1 βρ are the coefficients of the regression inputs. As seen in this model X1, X2 can be the multiple inputs which can help in determining the output of the Y (Alexopoulos, E.C., 2010)

2.5.4 Classification Model

Classification is one of the popular machine learning approaches where a model learn the model uses the training data to predict the class of the new samples in general classification is a process where the data is labelled interpreted to mind classes in a data set it is generally a two step process well first the model is built from a data set then the data from which classification function or model is learned is known as the training and the model is implemented on the new set where it is known as testing in data mining literature many classification algorithms are suggested to have capabilities of prediction these kind of facilities can be realised in other fields then context aware mobile services such an example would be cyber security with relevant data, Internet of Things or healthcare services (Sarker, I.H., Kayes, A.S.M. and Watters, P., 2019.)

2.6 Related Work

The public opinion of Bitcoin was examined by analysing around 2.27 million od tweets related to bitcoin for changes in their sentiments that might have an impact on price in near future. This was made possible by a method of solely attributing rise and falls of the aggregated twitter sentiments from period of 5 mins to 4 hours then shifting the predictions forwards in time to 1,2,3 or 4 time the corresponding interval. The model accuracy was 79% when sentiments were aggregated over 30 mins period. (Stenqvist, E. and Lönnö, J., 2017.)

The method used to process the data was with the help of supervised learning algorithms like logistic regression, naïve bayes and support vector outputs based on hour to hour or day to day with an accuracy over 90%. To gain this outcome a through out error analysis is used to verify each stage. Over the average this research resulted in a 25% boost over the accuracy. (Colianni, S., Rosales, S. and Signorotti, M., 2015)

It was discovered that tweet volume was main predictor of price instead of the sentiment of tweets by examining them. Forecasts were effective by linear method to understand direction of price movements considering the tweets and google trends data as input. (Abraham, J., et al., 2018)

The process of capturing weibo posts was detailed, describing how to create cryptospecific lexicon and give a recurrent neural network based on LSTM to suggest future price movements and forecast the trends, this technique beats auto regressive based model by 18.5% in precision and 15.4% overall (Huang, X., et al., 2021)

2.7 Gap in Literature

From all the literature mentioned above there are some research which are done in specific area of sentiment by twitter for prediction of prices but still they lack the comparison between different algorithms and overall comparison between different regression and classification models. Where I will research the impact of tweet sentiments based on the outcome of different models. My research will identify the gap between Vader Sentiment impact on the prediction model of Bitcoin and TextBlob Sentiment on the prediction model with the one where there is no sentiment input, which is not seen in the literature mentioned above.

Chapter 3: **Project Methodology**

3.1 Introduction

Projects based on research have a clear guideline which is followed throughout the timeline of this project. As the project is based on the dataset it was a clear secondary research which does not requires any active surveys or interviews and can be continued with the secondary available dataset.

For a project based on software application or any IT related project methodology are designed to carry out specific set of responsibilities. Often, this set of responsibilities that the device will carry out offers clearly explained results which contain complex computation and processing it is consequently a harsh and tedious task to manipulate the complete improvement method to make certain that the end product accommodates of excessive diploma of integrity and robustness an addition to user acceptance. Thus, a significant scientific improvement method that's capable of emphasising at the knowledge of the scope and complexity of the whole improvement method is crucial to attaining the said characteristics of the success device. (Leau, Y.B., et al., 2012).

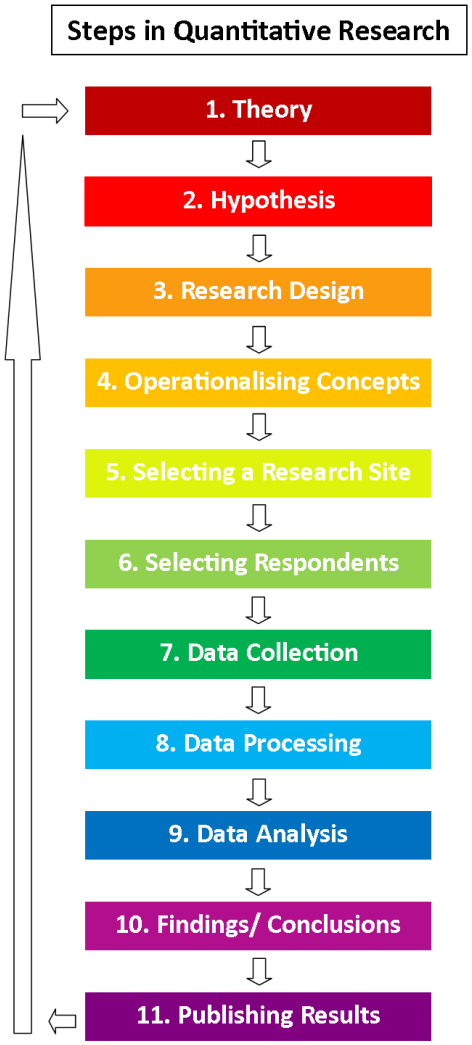

Fig 3.1 Flow chart for quantitative analysis (ReviseSociology. 2022)

3.2 Research Methodology

3.2.1 Data types

In the research aspects data is generally categorised into structured and unstructured where structured data is more represented as organised data and being labelled, on the other hand unstructured data can be described as unorganised and unlabelled where most of it can be random values or have the values but possibly in different measuring units. Data is generally referred to as structured and unstructured type, wherein structured data is more like labelled one and more organised. Unstructured data type is more like unorganised data with some random values, which is more towards unrelation form of the storage. While in the structured data storage it is more towards relational type of the storage. Data source can be categorised as the primary and secondary type where primary is defined as the data being collected by the researcher itself and not using a pre collected information wherein Once the data is understood we start planning and get to the point of data collection.

3.2.2 Data Collection

After determining the sort of data necessary for our model, we begin the search for the dataset. For the bitcoin price, we first check at several APIs to obtain data for that certain time period. Data gathering may be done in two distinct ways. The first is referred to as primary data, while the second is referred to as secondary data. Primary data is retrieved using a variety of methods, including quantitative analysis and qualitative analysis. In this type, we primarily use interview questions and focus groups, in which a question is asked and a discussion is held about various topics or aspects, and data is collected from selected individuals during the interview. In quantitative analysis, surveys are often conducted in sufficient numbers to allow for data processing and distribution in accordance with the study objectives. In the other section, secondary data is utilised to verify the study. This means that anybody who has previously conducted research or gathered data may use the same data for this current research.

For my project I found a secondary dataset regarding the bitcoin tweets. (Bitcoin Tweets | Kaggle. 2022) There is another possibility that the tweets can be retrieved directly from twitter with tweepy API such as tweepy for which first the user needs to be registered with twitter developer account and then get the keys and tokens which can be input for tweepy API that can retrieve the tweets with the required specification or hashtags.

3.2.3 Data Analysis

It is considered to be one of the most key element in terms of attainment of research. In this project there is first requirement of pre processing the data for which initially we need to clear and normalise the data as sometimes the data might not have the same unit in the columns or the data type might be different like the values could be integer but it is identifying as string which again requires the processing and the conversion to compatible data type. There are various paths towards data analysis like scaling and normalisation. Then there are different techniques to analysis the data like regression analysis, factor analysis, classification analysis. In the case for my project I am using multiple regression model as I have minimum of 3 inputs for all the models and multiple regression model is one where there can be 3 inputs and one output.

For my project I have used multiple linear regression analysis where first I load the secondary data then followed by train in test splitting and then I scale the data which is later normalised and used in the model first I create the model with the help of tensor flow sequential API in that I create multiple layers and then I predict the data once the data is predicted please check the accuracy and tweak the model as per my needs. Once the model is create I continue with the creation of other model and use the similar method along with the inclusion of different sentiment values that are generated from Vader and TextBlob algorithms and in the end I print the final accuracy for all three models in the actual value and in the bar chart form. With the help of this I can analyse the accuracy

3.3 Implementation

3.3.1 SDLC

As per the project description I chose Agile model because of the level of uncertainty and continuous requirement of the development around the project. First we collect and find what will be required to initiate this project. So we start with the basic research and find about the current scenario in this area where I do the literature research and later I start working on my model. I followed Agile because when I was done with low accuracy and after moving to another model I realised that there could be an improvement on the previous one so I would go back to the previous implementation which is one o the reason to choose Agile where it has the freedom to go back and forth as per the needs of the project. As for the requirement is concerned we can see that it is doable without any special requirements as for cost a regular computer, a stable internet connection is required and the rest of task can be done on the available cloud services viz. for Gantt chart and for task board and Gantt Chart I use ClickUp web services which I found easy to understand and handle along for running actual python notebook I used Visual Studio Code with jupyter and python because it would provide me to run jupyter notebook locally.

3.3.2 Method

I used the Kanban method where I create cards and individualise the task and try to achieve the target of task by sorting out them in space of completed in progress, and need to be developed. In the fig 3.3 I have displayed the old version of Kanban Board where I have allocated the task based on the timeline and the requirement of the project.

3.4 Project Planning

As per the given requirement I am suppose to follow a timeline based on the sub tasks divided for which I created a Gantt chart as with Gantt chart it is easy to create and follow the project . Based on the Gantt Chart

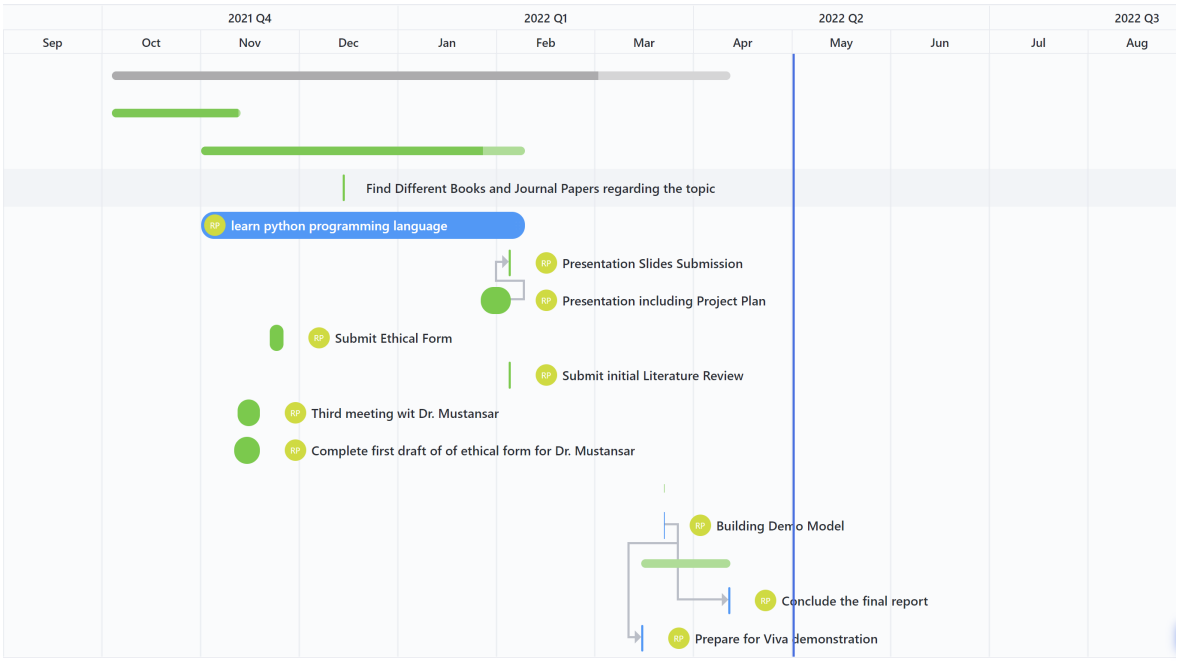

Fig 3.2 Gantt Chart for the timeline for this project

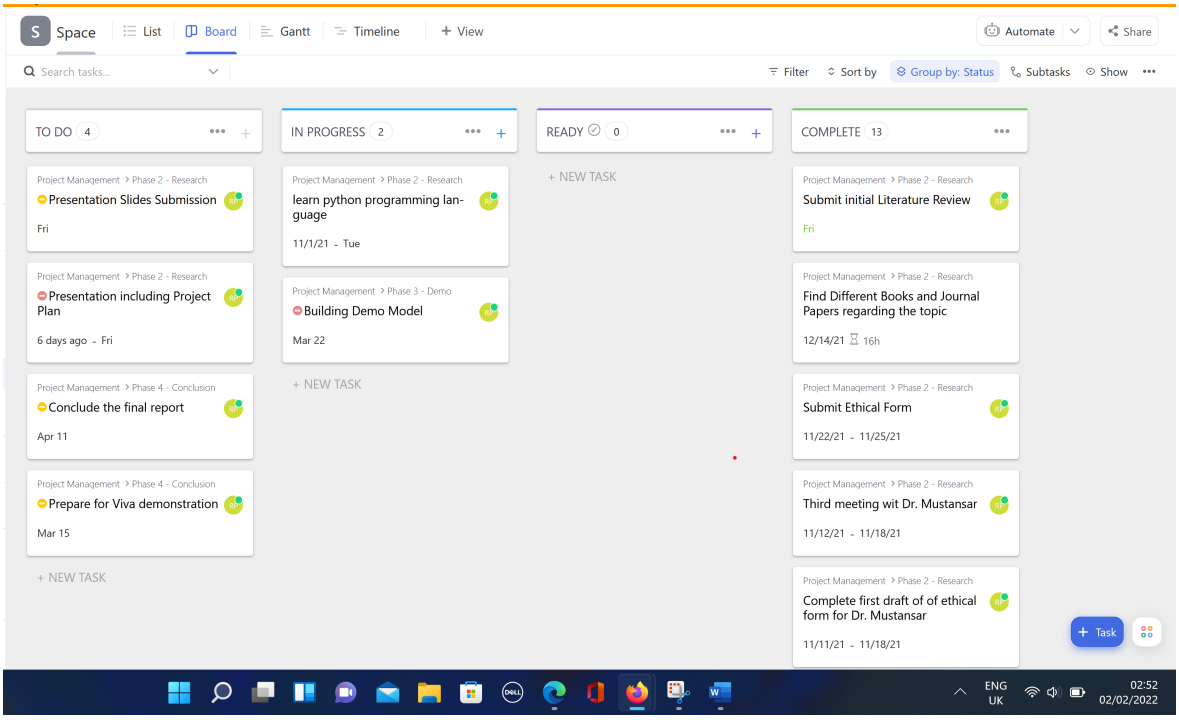

Fig 3.3 Task board for the description of Gantt chart old version

3.5 Challenges and Limitations

Initially the main challenge for me was that I had never used or learned the python programming language before the start of this project. So initially I had to allocate some time for that learning process and for me to get comfortable with the use of this language. Other major challenge was time limitation as because having not only this project but also to invest time in other modules along with the extra time for gaining the required skills to complete this project for which I created a Gantt chart for this project displayed above which clears the time allocation and provides me with an idea to complete this project without delaying every part of this project for last moment.

3.6 Ethical Considerations

This project contains secondary public tweets which do not contain any personal data. The use of this project is only to identify aggregate impact of sentiments of tweets over the bitcoin price. Although there can be a possible use case where this can be used to receive monetary gains by a large entity which can have negative effect on the overall economy.

Chapter 4: **Experiments and Results**

4.1 Introduction

In this part I display the code along with the output and the final results of the outcome there are 3 minimum inputs in all the models viz. high low and volume.

4.2 Research Outcome

First I retrieve the bitcoin prices for the required timeline with the help of Fastquant API

The output is shown below

```
import matplotlib. pyplot as plt
#visualise the closing history
plt.figure(figsize=(16,8))
plt.title('Close price History for bitcoin')
plt.plot(df['close'])
plt.xlabel('dt', fontsize=18)
plt.ylabel('Close prize GBP(£)', fontsize=18)
plt.show()
```

Fig 4.1 Bitcoin Price Graph for the months

dt	open	high	low	close	volume
2021-02-10	33753.00	34297.20	31712.00	32596.25	1181.160162
2021-02-11	32588.63	35360.73	32031.61	34945.27	1018.010015
2021-02-12	34955.21	35598.00	33516.12	34425.12	879.890038
2021-02-13	34377.80	34950.00	33681.11	34346.96	642.220822
2021-02-14	34336.73	36100.00	34231.02	35342.51	771.386256
...	...	...	...	...	...
2021-08-14	32514.60	32959.04	31762.31	32191.11	321.380926
2021-08-15	32203.04	32590.75	31720.01	32334.40	223.187002
2021-08-16	32350.69	33065.28	32226.62	32604.78	242.737023
2021-08-17	32646.96	32948.02	31910.89	32143.11	286.657226
2021-08-18	32133.01	32491.62	31717.13	32047.71	251.151725

190 rows × 5 columns

Fig 4.2 Bitcoin price table

Here I display the output of the neural language distribution in different formats. The outputs below are specially for VADER algorithm in different ways where in the table it is displayed row wise, there is month aggregation line chart, bar chart for distribution, percent based pie chart and histogram which displays tweets as per their sentiment score distribution.

```
dfbit[['date', 'text','hashtags','scores','compound','sentiment_type']]
```

Python

	date	text	hashtags	scores	compound	sentiment_type
0	2021-02-10 23:59:04	Blue Ridge Bank shares halted by NYSE after #b...	['bitcoin']	{'neg': 0.0, 'neu': 0.855, 'pos': 0.145, 'comp...	0.2960	POSITIVE
1	2021-02-10 23:58:48	😎 Today, that's this #Thursday, we will do a "...	['Thursday', 'Btc', 'wallet', 'security']	{'neg': 0.0, 'neu': 0.728, 'pos': 0.272, 'comp...	0.8225	POSITIVE
2	2021-02-10 23:54:48	Guys evening, I have read this article about B...	NaN	{'neg': 0.0, 'neu': 0.793, 'pos': 0.207, 'comp...	0.5719	POSITIVE
3	2021-02-10 23:54:33	$BTC A big chance in a billion! Price: \487264...	['Bitcoin', 'FX', 'BTC', 'crypto']	{'neg': 0.0, 'neu': 0.859, 'pos': 0.141, 'comp...	0.3164	POSITIVE
4	2021-02-10 23:54:06	This network is secured by 9 508 nodes as of t...	['BTC']	{'neg': 0.0, 'neu': 0.895, 'pos': 0.105, 'comp...	0.4019	POSITIVE
...	...	...	...	...	...	...
1048570	2021-08-18 10:57:20	10 Best NFT Stocks to Buy Now - Cashdasher htt...	['psychedelicart', 'btc', 'art', 'animation', ...	{'neg': 0.0, 'neu': 0.769, 'pos': 0.231, 'comp...	0.6369	POSITIVE
1048571	2021-08-18 10:57:20	Very good project\n@Naderi_Trader \n@Zdavari19...	['YieldFarming', 'Airdrop', 'PancakeSwap', 'BT...	{'neg': 0.0, 'neu': 0.842, 'pos': 0.158, 'comp...	0.4927	POSITIVE
1048572	2021-08-18 10:57:05	DeFi will seriously compete with TradFi. \n#Ea...	['EasyCrypto10', 'BTC', 'XRP', 'bnb', 'ADA', '...	{'neg': 0.108, 'neu': 0.892, 'pos': 0.0, 'comp...	-0.1779	NEGATIVE
1048573	2021-08-18 10:57:03	@BITCOIN_AL_DIA I expect a fall to $43k, retes...	['btc']	{'neg': 0.079, 'neu': 0.789, 'pos': 0.132, 'co...	0.6369	POSITIVE
1048574	2021-08-18 10:56:51	@CornDeFi it's really amazing project i lik...	['YieldFarming', 'Airdrop', 'Giveaway', 'Panca...	{'neg': 0.0, 'neu': 0.742, 'pos': 0.258, 'comp...	0.7645	POSITIVE

1048575 rows × 6 columns

Fig 4.3 Output of Vader NLP processing with the sentiments values

```
monthvad.plot()
```

```
<AxesSubplot:xlabel='date'>
```

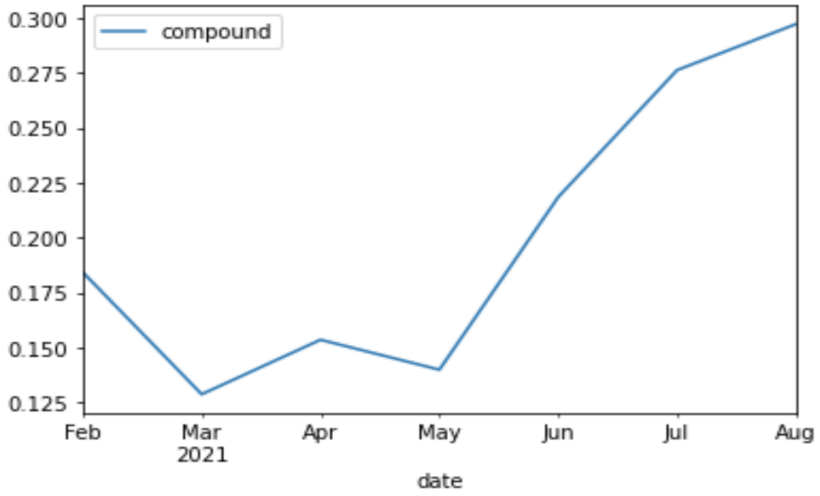

Fig 4.4 Monthly Graph for sentiments

```
dfvad.sentiment_type.value_counts().plot(kind='bar',title="sentiment analysis")
```

```
<AxesSubplot:title={'center':'sentiment analysis'}>
```

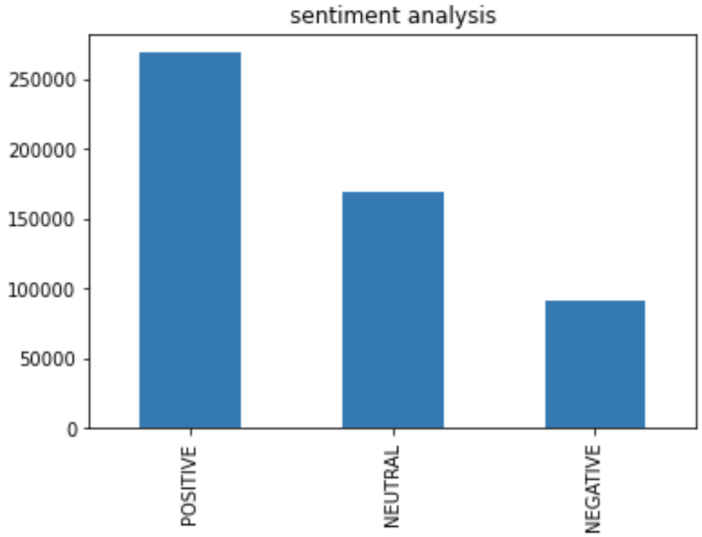

Fig 4.5 Bar chart for tweet distribution based on Vader NLP

```
import matplotlib. pyplot as plt
import matplotlib
plt.pie(dfbit.sentiment_type.value_counts()
plt.show()
```

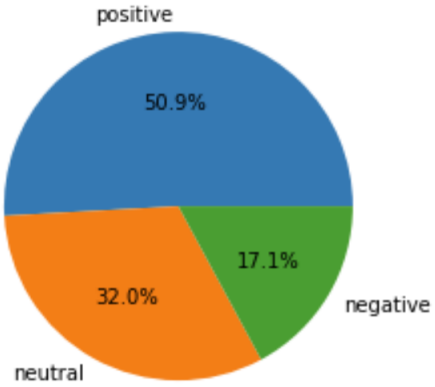

Fig 4.6 Percent Distribution for tweets based on Vader NLP.

Fig 4.6 shows tweets distribution based on Vader neural language algorithm where 50.9% of total were positive tweets, 32 % were neutral tweets, and 17.1% were negative.

```
dfbit.compound.hist()
```

```
<AxesSubplot:>
```

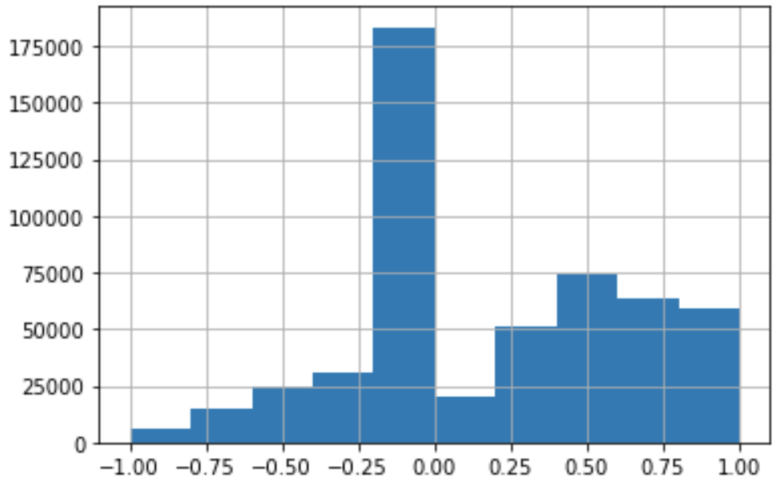

Fig 4.7 Histogram for tweets based on Vader NLP

```
import numpy
import matplotlib.pyplot as plt
plt.figure(figsize=(16,8))
#plt.title('Close price History for bitcoin')
plt.plot(dfvad['date'],dfvad['compound'])
plt.xlabel('date', fontsize=18)
plt.ylabel('polarity', fontsize=18)
plt.show()
```

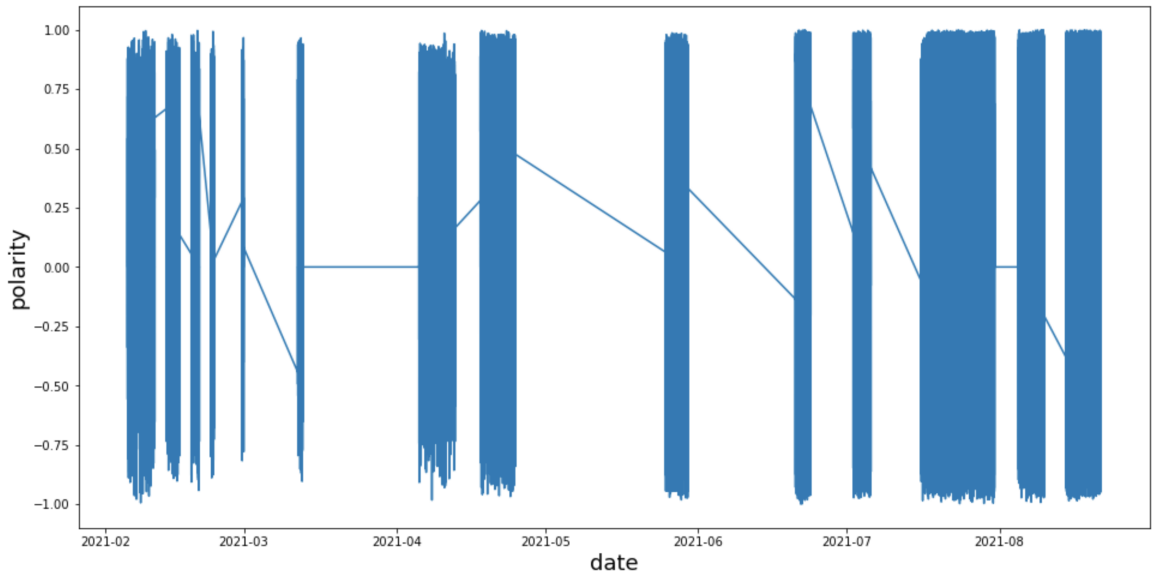

Fig 4.8 Line Chart of tweets based on Vader NLP

Next I display output of TextBlob Algorithm in different ways.

```
dfbit[['date', 'text','hashtags','Polarity','Sentiment_Type']]
```

✓ 1.7s Pytho

	date	text	hashtags	Polarity	Sentiment_Type
0	2021-02-10 23:59:04	Blue Ridge Bank shares halted by NYSE after #b...	['bitcoin']	0.000000	NEUTRAL
1	2021-02-10 23:58:48	😎 Today, that's this #Thursday, we will do a "...	['Thursday', 'Btc', 'wallet', 'security']	0.000000	NEUTRAL
2	2021-02-10 23:54:48	Guys evening, I have read this article about B...	NaN	0.000000	NEUTRAL
3	2021-02-10 23:54:33	$BTC A big chance in a billion! Price: \487264...	['Bitcoin', 'FX', 'BTC', 'crypto']	0.000000	NEUTRAL
4	2021-02-10 23:54:06	This network is secured by 9 508 nodes as of t...	['BTC']	0.000000	NEUTRAL
...	...	...	...	...	...
1048570	2021-08-18 10:57:20	10 Best NFT Stocks to Buy Now - Cashdasher htt...	['psychedelicart', 'btc', 'art', 'animation', ...	1.000000	POSITIVE
1048571	2021-08-18 10:57:20	Very good project\n@Naderi_Trader \n@Zdavari19...	['YieldFarming', 'Airdrop', 'PancakeSwap', 'BT...	0.910000	POSITIVE
1048572	2021-08-18 10:57:05	DeFi will seriously compete with TradFi. \n#Ea...	['EasyCrypto10', 'BTC', 'XRP', 'bnb', 'ADA', '...	-0.333333	NEGATIVE
1048573	2021-08-18 10:57:03	@BITCOIN_AL_DIA I expect a fall to $43k, retes...	['btc']	0.092500	POSITIVE
1048574	2021-08-18 10:56:51	@CornDeFi it's really amazing project i lik...	['YieldFarming', 'Airdrop', 'Giveaway', 'Panca...	0.600000	POSITIVE

1030260 rows × 5 columns

Fig 4.9 Output table for sentiments based on TextBlob algorithm

```
monthblob.plot()
```
✓ 0.3s

```
<AxesSubplot:xlabel='date'>
```

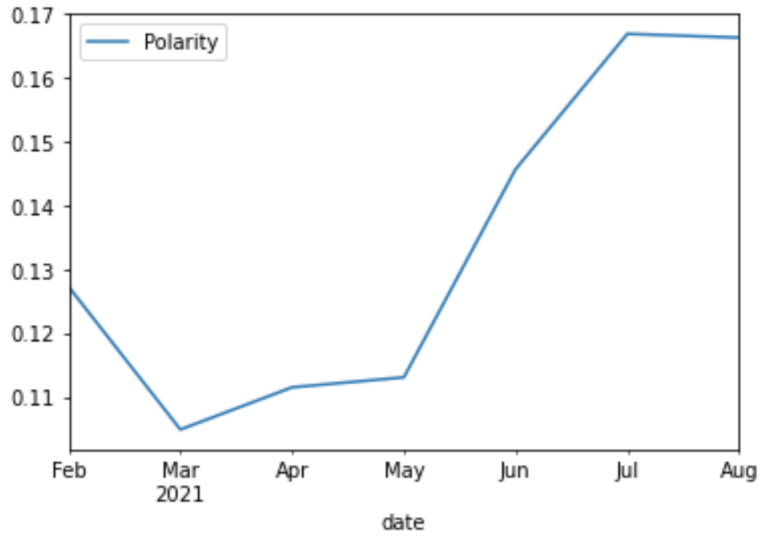

Fig 4.10 Monthly Sentiment score line chart based on TextBlob

```
import matplotlib. pyplot as plt
import matplotlib
plt.pie(dfbit.Sentiment_Type.value_counts()
plt.show()
```
✓ 0.1s

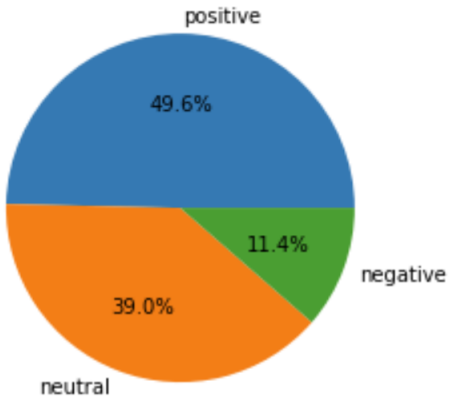

Fig 4.11 Percent Distribution for Sentiments based on TextBlob

Fig 4.11 is the percent distribution based on TextBlob algorithm for sentiment where 49.6% of all tweets were positive, 39% were neutral tweets and 11.4% were negative tweets

4.3 Implementation Outcome

4.3.1 Model 1 Multivariable Regression with sentiment score

The fig 4.10 represents the line chart with all the values both the features and targets where first 3 are features and Y is target

```
df.plot(subplots=True)
```

✓ 1.1s

```
array([<AxesSubplot:xlabel='dt'>, <AxesSubplot:xlabel='dt'>,
       <AxesSubplot:xlabel='dt'>, <AxesSubplot:xlabel='dt'>], dtype=object)
```

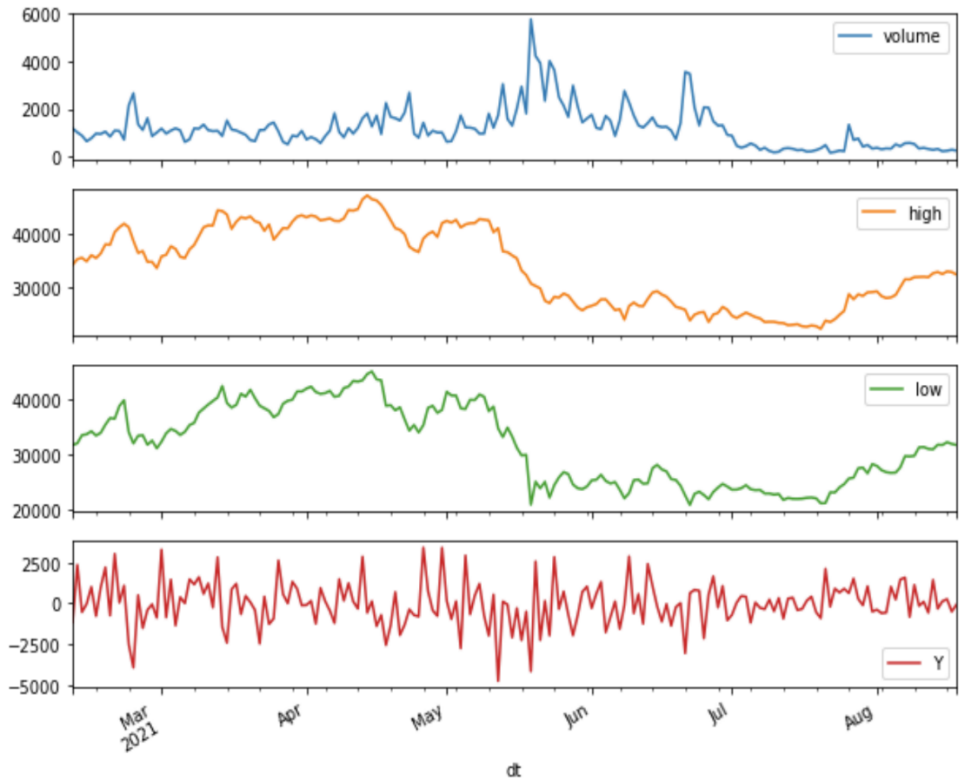

Fig 4.12 Line Chart for BTC price

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape= (win_length, num_features), return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.LSTM(128, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(64, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(128, kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64, kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(32, kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))
```

✓ 1.2s

```
model.summary()
```

✓ 0.8s

```
Output exceeds the size limit. Open the full output data in a text editor
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 5, 128)            67584

 leaky_re_lu (LeakyReLU)     (None, 5, 128)            0

 lstm_1 (LSTM)               (None, 5, 128)            131584

 leaky_re_lu_1 (LeakyReLU)   (None, 5, 128)            0

 dropout (Dropout)           (None, 5, 128)            0

 lstm_2 (LSTM)               (None, 64)                49408

 dropout_1 (Dropout)         (None, 64)                0

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 32)                2080

 dense_3 (Dense)             (None, 1)                 33
...
Total params: 267,265
Trainable params: 267,265
Non-trainable params: 0
```

Fig 4.13 Model Creation along with summary

In the model shown above, for first multivariable regression the model contains 11 layers where the last layer is the output. The model remains same across the first program.

```
df_final
```
✓ 0.4s

dt	volume	high	low	Y	App_Pred
2021-06-28	1302.977806	25333.33	23921.38	-246.48	-3693.837956
2021-06-29	1314.614957	26453.88	24596.30	1059.31	-3467.942538
2021-06-30	910.616418	25934.99	24125.24	-1036.02	-3625.606107
2021-07-01	880.650499	24804.49	23550.00	-701.58	-3818.138674
2021-07-02	453.865458	24385.16	23568.32	46.23	-3812.006978
2021-07-03	373.405407	24900.00	23824.16	450.55	-3726.377446
2021-07-04	436.082707	25399.18	24366.61	410.38	-3544.819677
2021-07-05	551.012291	24960.58	23632.67	-1181.45	-3790.469063
2021-07-06	464.249891	24560.97	23471.60	51.88	-3844.379119
2021-07-07	282.807617	24306.50	23512.00	-271.92	-3830.857257

Fig 4.14 Output of Model 1 without sentiment input

In the table above the model has considered only the historical data of bitcoin.

```
df_final[['App_Pred','Y']].plot()
```
✓ 0.2s

```
<AxesSubplot:xlabel='dt'>
```

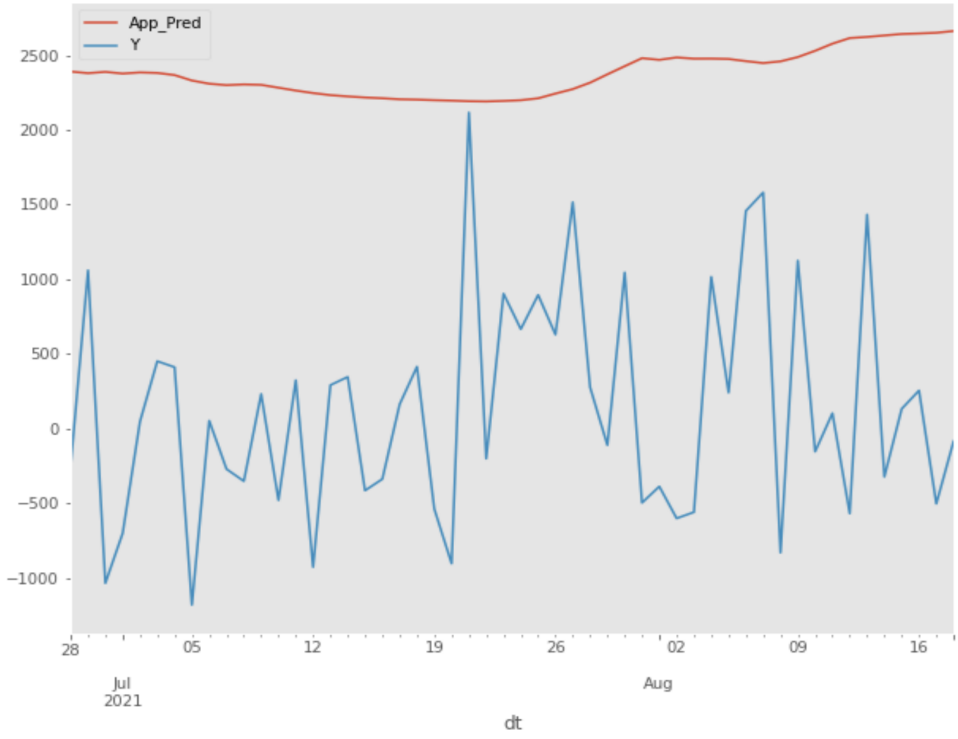

Fig 4.15 Actual Prices (Y) vs Prediction line chart

```
df_final[['App_Pred','high','low','volume','Y']].plot()
```
✓ 0.3s

```
<AxesSubplot:xlabel='dt'>
```

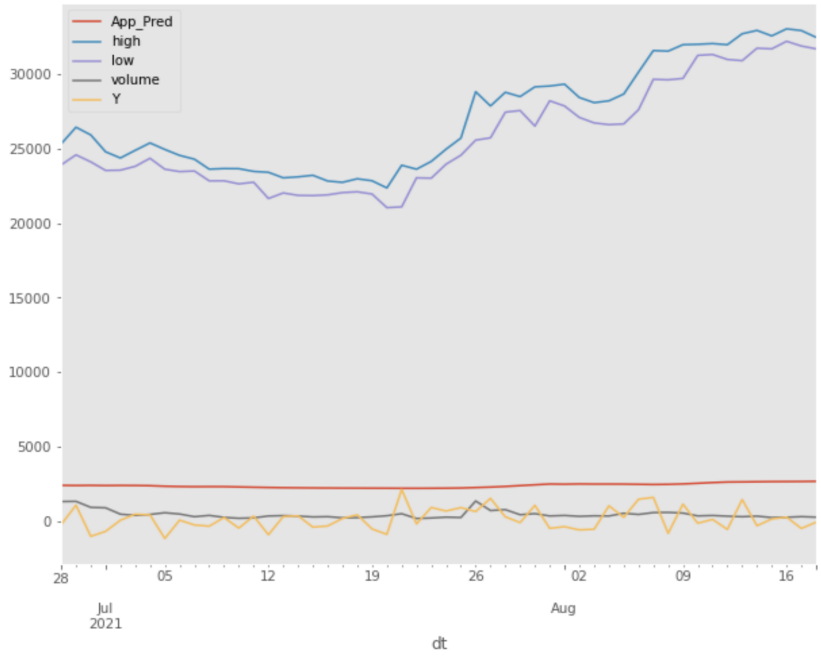

Fig 4.16 All data line chart along with prediction

The figures shown below are for the model with Vader Sentiment as one more input.

```
df1.plot(subplots=True)
```
✓ 0.7s

```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>], dtype=object)
```

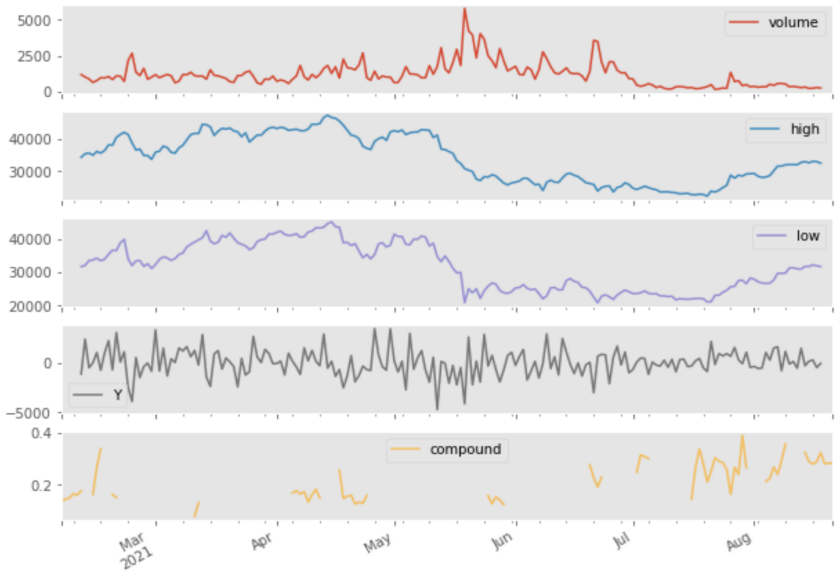

Fig 4.17 Data for model with Vader Sentiment

```
df_input1
```
✓ 0.7s

	volume	high	low	compound	Y
2021-02-10	1181.160162	34297.20	31712.00	0.177385	-1156.75
2021-02-13	642.220822	34950.00	33681.11	0.163294	-30.84
2021-02-14	771.386256	36100.00	34231.02	0.269409	1005.78
2021-02-15	965.418037	35560.29	33450.00	0.337748	-776.42
2021-02-18	840.049697	38030.00	36639.55	0.163850	-740.15
...	...	...	...	...	...
2021-08-14	321.380926	32959.04	31762.31	0.324488	-323.49
2021-08-15	223.187002	32590.75	31720.01	0.289495	131.36
2021-08-16	242.737023	33065.28	32226.62	0.279850	254.09
2021-08-17	286.657226	32948.02	31910.89	0.289264	-503.85
2021-08-18	251.151725	32491.62	31717.13	0.323912	-85.30

65 rows × 5 columns

Fig 4.18 Datafrme in tabular format with the Vader sentiment

```
df_final[['App_Pred','Y']].plot()
```
✓ 0.2s

```
<AxesSubplot:>
```

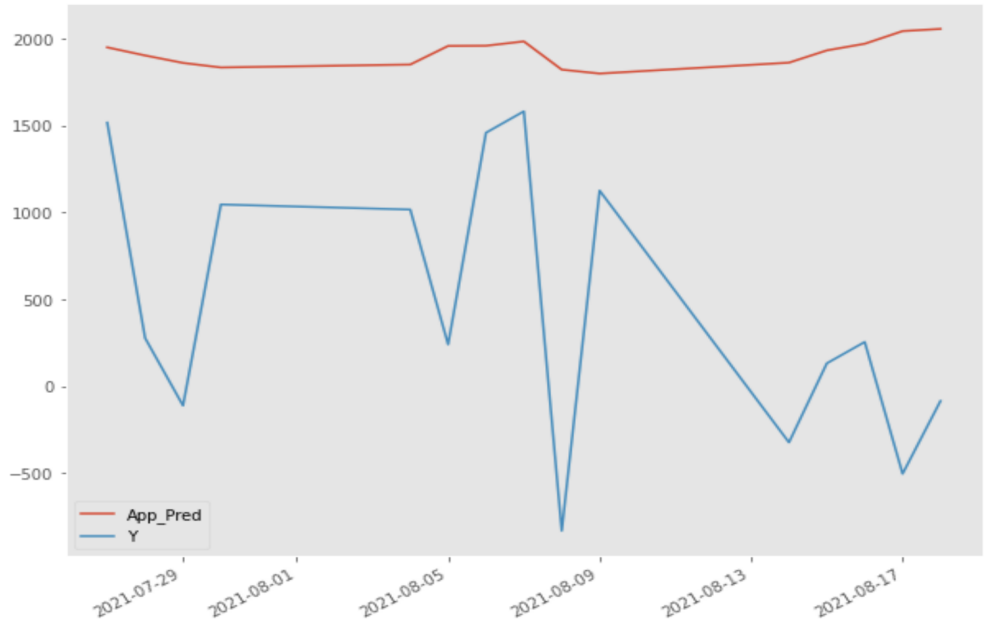

Fig 4.19 Actual value Y vs Prediction with Vader Sentiments

```
df_final[['App_Pred','high','low','volume','Y']].plot()
```
✓ 0.2s

```
<AxesSubplot:>
```

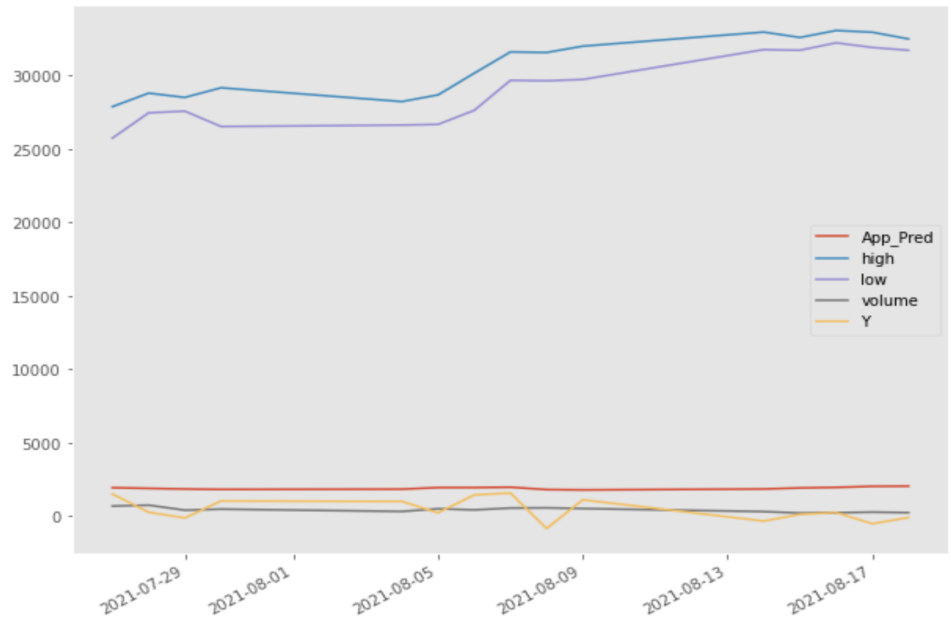

Fig 4.20 All the data line chart

The figures shown below are for the model with the TextBlob generated sentiments.

```
df_input1 = df_input1.dropna()
df_input1
```
✓ 0.4s

	volume	high	low	Polarity	Y
2021-02-10	1181.160162	34297.20	31712.00	0.109335	-1156.75
2021-02-13	642.220822	34950.00	33681.11	0.108791	-30.84
2021-02-14	771.386256	36100.00	34231.02	0.172871	1005.78
2021-02-15	965.418037	35560.29	33450.00	0.219700	-776.42
2021-02-18	840.049697	38030.00	36639.55	0.106882	-740.15
...	...	...	...	...	...
2021-08-14	321.380926	32959.04	31762.31	0.189500	-323.49
2021-08-15	223.187002	32590.75	31720.01	0.178322	131.36
2021-08-16	242.737023	33065.28	32226.62	0.157109	254.09
2021-08-17	286.657226	32948.02	31910.89	0.166063	-503.85
2021-08-18	251.151725	32491.62	31717.13	0.190544	-85.30

65 rows × 5 columns

Fig 4.21 Scaled values of the dataset

In table above there are all the values both the features and target.

```
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                  patience=2,
                                                  mode= 'min')
model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit_generator(train_generator, epochs=50,
                    validation_data=test_generator,
                    shuffle=False,
                    callbacks=[early_stopping], verbose=1)
```

✓ 9.9s Python

```
Epoch 1/50

C:\Users\parek\AppData\Local\Temp/ipykernel_17252/1778312446.py:7: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future
version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(train_generator, epochs=50,

4/4 [==============================] - 9s 629ms/step - loss: 0.2347 - mean_absolute_error: 0.4423 - val_loss: 0.2505 - val_mean_absolute_error:
0.4843
Epoch 2/50
4/4 [==============================] - 0s 34ms/step - loss: 0.1384 - mean_absolute_error: 0.3238 - val_loss: 0.1036 - val_mean_absolute_error:
0.2928
Epoch 3/50
4/4 [==============================] - 0s 35ms/step - loss: 0.0546 - mean_absolute_error: 0.1758 - val_loss: 0.0216 - val_mean_absolute_error:
0.1320
Epoch 4/50
4/4 [==============================] - 0s 36ms/step - loss: 0.1173 - mean_absolute_error: 0.2788 - val_loss: 0.0199 - val_mean_absolute_error:
0.1253
Epoch 5/50
4/4 [==============================] - 0s 38ms/step - loss: 0.0741 - mean_absolute_error: 0.2078 - val_loss: 0.0242 - val_mean_absolute_error:
0.1228
```

Fig 4.22 Loss output for model with TextBlob sentiments

	volume	high	low	Polarity	Y	App_Pred
2021-07-27	696.799504	27882.60	25741.74	0.164128	1515.33	1851.329947
2021-07-28	758.954605	28800.00	27464.02	0.144288	276.36	1823.560825
2021-07-29	417.142048	28510.32	27576.06	0.166982	-111.72	1777.170479
2021-07-30	488.119281	29166.39	26530.98	0.143572	1044.71	1750.143836
2021-08-04	330.635564	28228.30	26627.04	0.127996	1015.51	1687.042788
2021-08-05	513.347409	28682.59	26675.00	0.137004	240.04	1781.487755
2021-08-06	436.380723	30159.35	27630.00	0.170495	1457.20	1741.713609
2021-08-07	561.001297	31605.21	29673.05	0.144028	1580.38	1740.694813
2021-08-08	573.071552	31565.98	29639.51	0.161139	-831.63	1694.407035
2021-08-09	527.310511	32000.00	29732.27	0.157640	1124.66	1707.736230
2021-08-14	321.380926	32959.04	31762.31	0.189500	-323.49	1773.301242
2021-08-15	223.187002	32590.75	31720.01	0.178322	131.36	1835.653735
2021-08-16	242.737023	33065.28	32226.62	0.157109	254.09	1832.729380
2021-08-17	286.657226	32948.02	31910.89	0.166063	-503.85	1887.370939
2021-08-18	251.151725	32491.62	31717.13	0.190544	-85.30	1908.133436

Fig 4.23 Input table along with actual and predicted output

```
df_final[['App_Pred','Y']].plot()
```

✓ 0.2s

```
<AxesSubplot:>
```

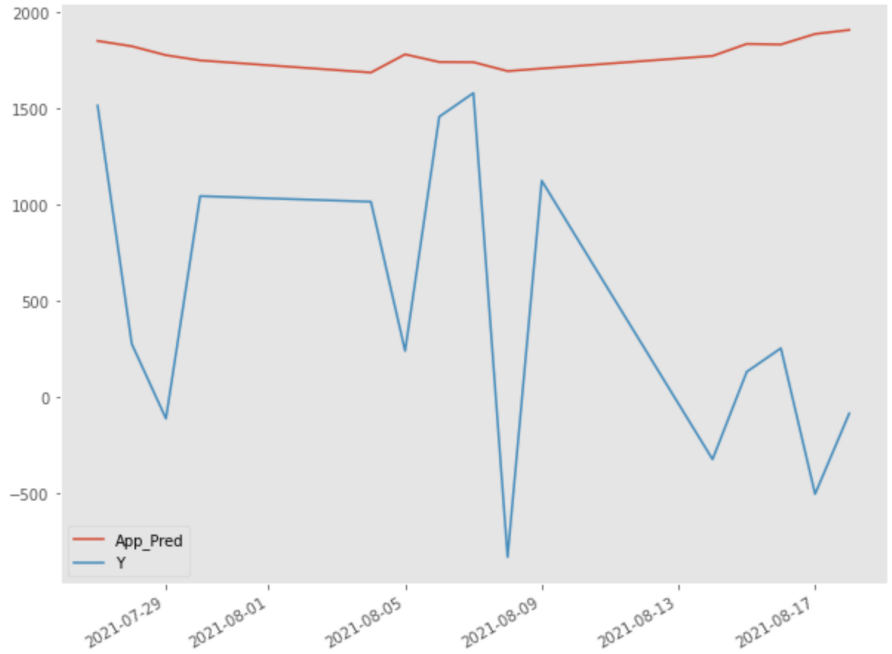

Fig 4.24 Predicted vs actual output line chart based on Textblob algorithm

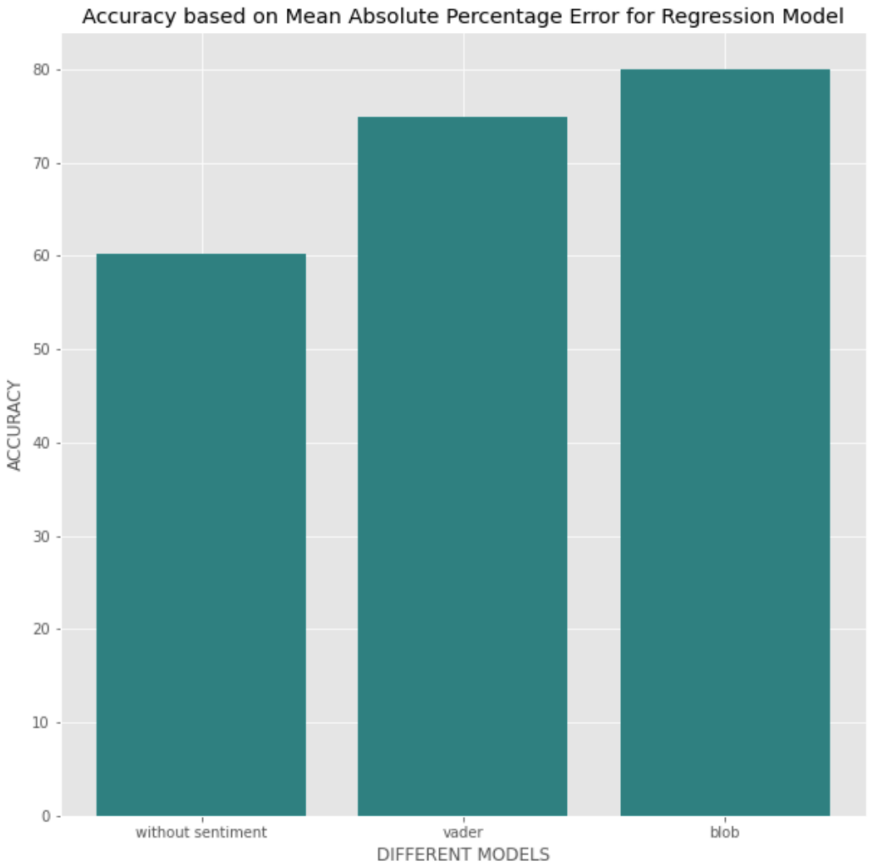

Fig 4.25 Accuracy for all three inputs

```
print('Error margin without sentiments:',mae1)
print('Error margin Vader Model:       ',mae2)
print('Error margin Textblob model:    ',mae3)

print('Accuracy without sentiments:    ',acc1)
print('Accuracy Vader Model:           ',acc2)
print('Accuracy Textblob model:        ',acc3)
```

✓ 0.6s

```
Error margin without sentiments: 39.76859353053503
Error margin Vader Model:        25.105253609558204
Error margin Textblob model:     20.027108484700452
Accuracy without sentiments:     60.23140646946497
Accuracy Vader Model:            74.8947463904418
Accuracy Textblob model:         79.97289151529955
```

Fig 4.26 Numerical Value for all three accuracy along with error margin

4.3.2 Model 2 Multivariable regression

```
df_input.describe()
```
✓ 0.7s

	volume	high	low	Trend
count	60.000000	60.000000	60.000000	60.000000
mean	1113.511208	31734.157167	29697.966167	-160.342333
std	886.915188	7082.403631	6664.392225	1181.526139
min	162.375809	22374.720000	20767.620000	-3048.550000
25%	344.305474	25643.020000	23932.250000	-749.217500
50%	893.493156	28891.680000	27288.215000	-104.530000
75%	1679.191008	37769.922500	35363.087500	637.887500
max	3559.823258	45438.940000	43554.990000	3029.850000

Fig 4.27 Data frame description

```
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.30, random_state=42, shuffle = False)
```
4] ✓ 0.6s

```
x_train.shape
```
5] ✓ 0.6s

```
(42, 3)
```

```
x_test.shape
```
6] ✓ 0.7s

```
(18, 3)
```

```
win_length=5
batch_size=13
num_features=3 #this number will change when polarity is added
train_generator = TimeseriesGenerator(x_train, y_train, length=win_length, sampling_rate=1, batch_size=batch_size)
test_generator = TimeseriesGenerator(x_test, y_test, length=win_length, sampling_rate=1, batch_size=batch_size)
```
7] ✓ 0.7s

Fig 4.28 Train Test data split along with Time Series generator

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape= (win_length, num_features), return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.LSTM(128, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(64, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(128, kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64, kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(32, kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))
```

✓ 0.5s

+ Code + Markdown

Fig 4.29 Model for model with both sentiment and binary sentiment score input.

```
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',   patience=5,  mode= 'min')

model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit_generator(train_generator, epochs=100,
                    validation_data=test_generator,
                    shuffle=False,
                    callbacks=[early_stopping], verbose=1)
```

✓ 6.1s

```
Epoch 1/100

C:\Users\parek\AppData\Local\Temp/ipykernel_22072/4153657523.py:6: UserWarning: `Model.fit_generator` is deprecated and will be removed in a fu
version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(train_generator, epochs=100,

3/3 [==============================] - 5s 526ms/step - loss: 0.2050 - mean_absolute_error: 0.4006 - val_loss: 0.1884 - val_mean_absolute_error:
0.4179
Epoch 2/100
3/3 [==============================] - 0s 39ms/step - loss: 0.1046 - mean_absolute_error: 0.2604 - val_loss: 0.0616 - val_mean_absolute_error:
0.2098
Epoch 3/100
3/3 [==============================] - 0s 37ms/step - loss: 0.0644 - mean_absolute_error: 0.2017 - val_loss: 0.0246 - val_mean_absolute_error:
0.1257
Epoch 4/100
3/3 [==============================] - 0s 41ms/step - loss: 0.0863 - mean_absolute_error: 0.2323 - val_loss: 0.0300 - val_mean_absolute_error:
0.1361
Epoch 5/100
3/3 [==============================] - 0s 44ms/step - loss: 0.0585 - mean_absolute_error: 0.1916 - val_loss: 0.0495 - val_mean_absolute_error:
0.1868
```

Fig 4.30 Loss for the model training without sentiment input

```
df_final
```
✓ 0.6s

	volume	high	low	Trend	App_Pred
2021-07-28	758.954605	28800.00	27464.02	276.36	1329.152015
2021-07-29	417.142048	28510.32	27576.06	-111.72	1358.832873
2021-07-30	488.119281	29166.39	26530.98	1044.71	1387.620994
2021-08-04	1034.951917	42442.16	40718.71	1477.40	1416.249239
2021-08-05	962.859033	42825.04	40950.00	1196.90	1396.757655
...	...	...	...	...	...
2021-08-14	321.380926	32959.04	31762.31	-323.49	1530.786923
2021-08-15	223.187002	32590.75	31720.01	131.36	1496.335888
2021-08-16	242.737023	33065.28	32226.62	254.09	1458.712938
2021-08-17	286.657226	32948.02	31910.89	-503.85	1371.207061
2021-08-18	251.151725	32491.62	31717.13	-85.30	1393.628063

13 rows × 5 columns

Fig 4.31 Output of the testing data without sentiments

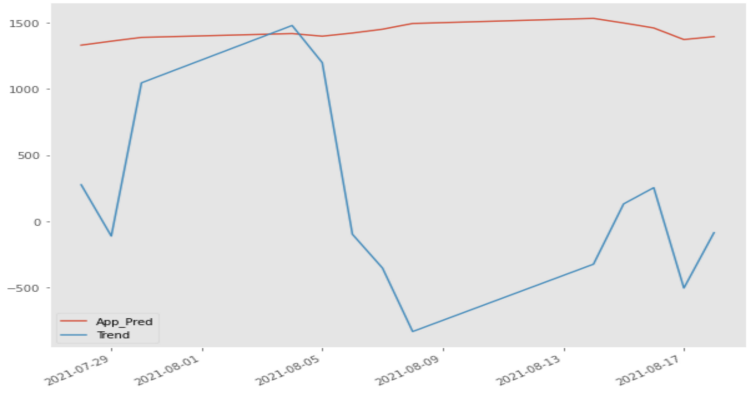

Fig 4.32 Predicted vs actual value

```
df_input1 = df_input1[['volume','high','low','sentiment_mag','sentiment','Trend' ]]
df_input1
```

✓ 0.4s

	volume	high	low	sentiment_mag	sentiment	Trend
2021-02-05	646.527211	42161.21	40778.59	0.134685	1	-960.19
2021-02-06	1202.602896	26996.48	25379.30	0.145868	1	600.30
2021-02-07	453.865458	24385.16	23568.32	0.149594	1	46.23
2021-02-08	307.734154	28451.25	27112.41	0.165777	1	-601.86
2021-02-13	642.220822	34950.00	33681.11	0.163294	1	-30.84
...	...	...	...	...	...	...
2021-08-14	321.380926	32959.04	31762.31	0.324488	1	-323.49
2021-08-15	223.187002	32590.75	31720.01	0.289495	1	131.36
2021-08-16	242.737023	33065.28	32226.62	0.279850	1	254.09
2021-08-17	286.657226	32948.02	31910.89	0.289264	1	-503.85
2021-08-18	251.151725	32491.62	31717.13	0.323912	1	-85.30

60 rows × 6 columns

Fig 4.33 Input for the Vader model

```
df_input1['sentiment']=df_input1['sentiment_mag'].astype(int)
df_input1.loc[df_input1.sentiment_mag>0,'sentiment']=1
df_input1.loc[df_input1.sentiment_mag==0,'sentiment']=0
df_input1.loc[df_input1.sentiment_mag<0,'sentiment']=0
```

✓ 0.4s

Fig 4.34 Binary conversion of Sentiment Score

	volume	high	low	sentiment_mag	sentiment	Trend	App_Pred
2021-07-28	758.954605	28800.00	27464.02	0.240599	1	276.36	1501.897448
2021-07-29	417.142048	28510.32	27576.06	0.389384	1	-111.72	1497.989938
2021-07-30	488.119281	29166.39	26530.98	0.265796	1	1044.71	1509.987469
2021-08-04	1034.951917	42442.16	40718.71	0.213507	1	1477.40	1544.375221
2021-08-05	962.859033	42825.04	40950.00	0.224732	1	1196.90	1600.686018
...	...	...	...	...	...	...	...
2021-08-14	321.380926	32959.04	31762.31	0.324488	1	-323.49	1648.689913
2021-08-15	223.187002	32590.75	31720.01	0.289495	1	131.36	1631.214456
2021-08-16	242.737023	33065.28	32226.62	0.279850	1	254.09	1612.067938
2021-08-17	286.657226	32948.02	31910.89	0.289264	1	-503.85	1588.111959
2021-08-18	251.151725	32491.62	31717.13	0.323912	1	-85.30	1648.324191

13 rows × 7 columns

Fig 4.35 Input with the predicted output for Vader

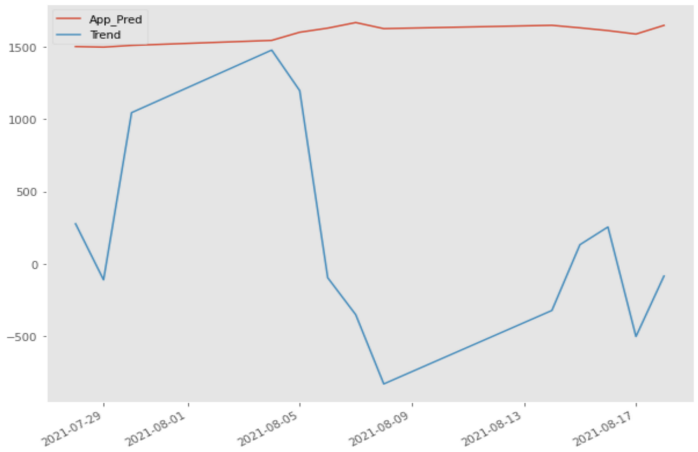

Fig 4.36 Predicted vs Actual Output based on Vader Sentiments Value

```
df_input1 = df_input1 [['volume','high','low','sentiment_blob','sentiment','Trend']]
df_input1
```

✓ 0.7s

	volume	high	low	sentiment_blob	sentiment	Trend
2021-02-05	646.527211	42161.21	40778.59	0.104351	1	-960.19
2021-02-06	1202.602896	26996.48	25379.30	0.119142	1	600.30
2021-02-07	453.865458	24385.16	23568.32	0.156801	1	46.23
2021-02-08	307.734154	28451.25	27112.41	0.112676	1	-601.86
2021-02-13	642.220822	34950.00	33681.11	0.108791	1	-30.84
...	...	...	...	...	...	...
2021-08-14	321.380926	32959.04	31762.31	0.189500	1	-323.49
2021-08-15	223.187002	32590.75	31720.01	0.178322	1	131.36
2021-08-16	242.737023	33065.28	32226.62	0.157109	1	254.09
2021-08-17	286.657226	32948.02	31910.89	0.166063	1	-503.85
2021-08-18	251.151725	32491.62	31717.13	0.190544	1	-85.30

60 rows × 6 columns

Fig 4.37 Input and the Actual output for TextBlob model

	volume	high	low	sentiment_blob	sentiment	Trend	App_Pred
2021-07-28	758.954605	28800.00	27464.02	0.144288	1	276.36	1670.949418
2021-07-29	417.142048	28510.32	27576.06	0.166982	1	-111.72	1660.077503
2021-07-30	488.119281	29166.39	26530.98	0.143572	1	1044.71	1649.231104
2021-08-04	1034.951917	42442.16	40718.71	0.127996	1	1477.40	1640.834290
2021-08-05	962.859033	42825.04	40950.00	0.137004	1	1196.90	1628.906927
...	...	...	...	...	...	...	...
2021-08-14	321.380926	32959.04	31762.31	0.189500	1	-323.49	1544.003627
2021-08-15	223.187002	32590.75	31720.01	0.178322	1	131.36	1590.655304
2021-08-16	242.737023	33065.28	32226.62	0.157109	1	254.09	1648.986075
2021-08-17	286.657226	32948.02	31910.89	0.166063	1	-503.85	1625.598523
2021-08-18	251.151725	32491.62	31717.13	0.190544	1	-85.30	1586.705470

13 rows × 7 columns

Fig 4.38 Data frame with the predicted output

```
df_final[['App_Pred','Trend']].plot()
```

✓ 0.2s

```
<AxesSubplot:>
```

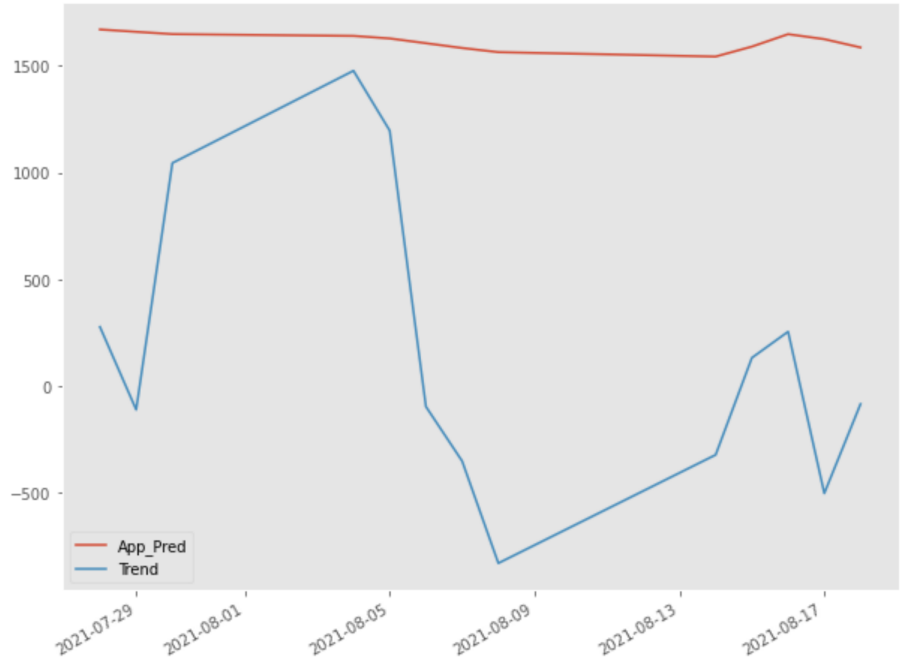

Fig 4.39 Predicted vs Actual output for Textblob model

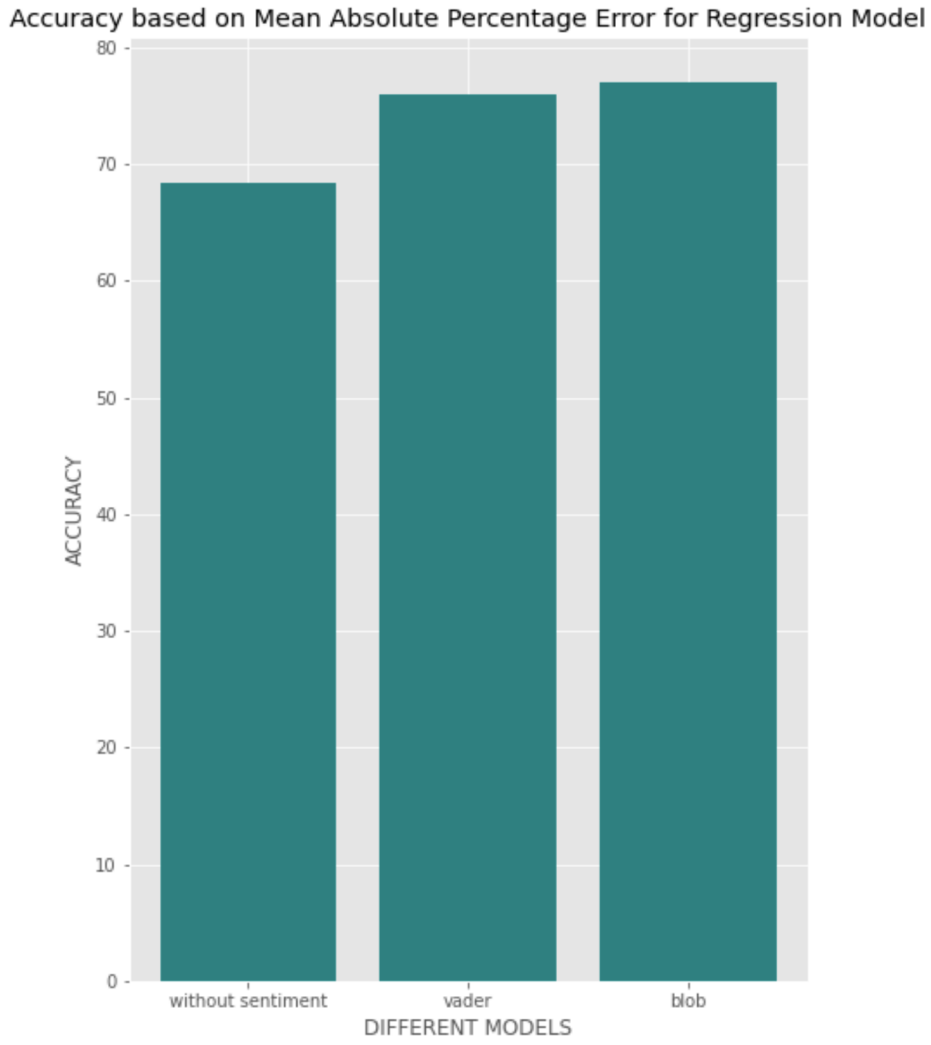

Fig 4.40 Accuracy for overall all 3 models

```
Error margin without sentiments: 31.580421081147776
Error margin Vader Model:        23.972154657991137
Error margin Textblob model:     22.981888227242315
Accuracy without sentiments:     68.41957891885222
Accuracy Vader Model:            76.02784534200886
Accuracy Textblob model:         77.01811177275769
```

Fig 4.41 Error Margins and Accuracy for all 3 models

4.3.3 Model 3 Binary Classification

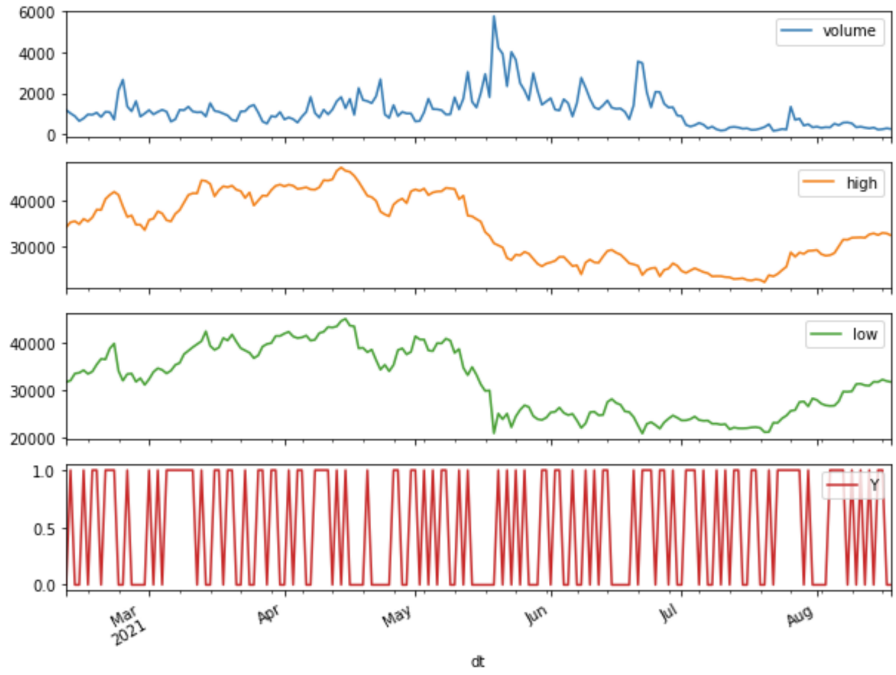

Fig 4.42 Input and the Actual Output line chart for Binary Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense , LeakyReLU
model=Sequential()
model.add(Dense(16, kernel_initializer='HeUniform', activation='relu'))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(Dense(16, kernel_initializer='HeUniform', activation='relu'))
model.add(Dense(32, kernel_initializer='HeUniform', activation='relu'))


model.add(Dense (1, activation='sigmoid'))
model.compile(optimizer='SGD',loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train_scaled,y_train,epochs=250,verbose=0)
J_list = model.history.history['loss']
plt.plot (J_list)
```

✓ 1.7s

```
[<matplotlib.lines.Line2D at 0x2246dc49ac0>]
```

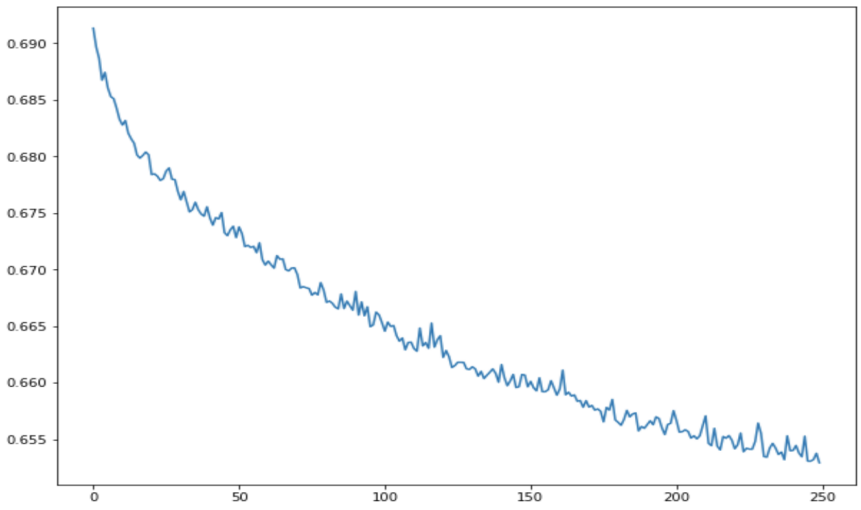

Fig 4.43 model with the loss graph

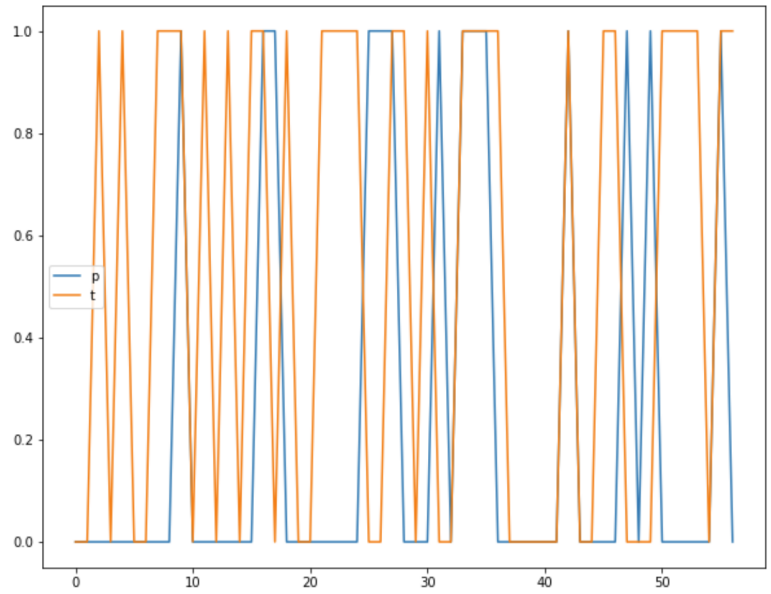

Fig 4.44 Predictions (p) actual (t) line chart

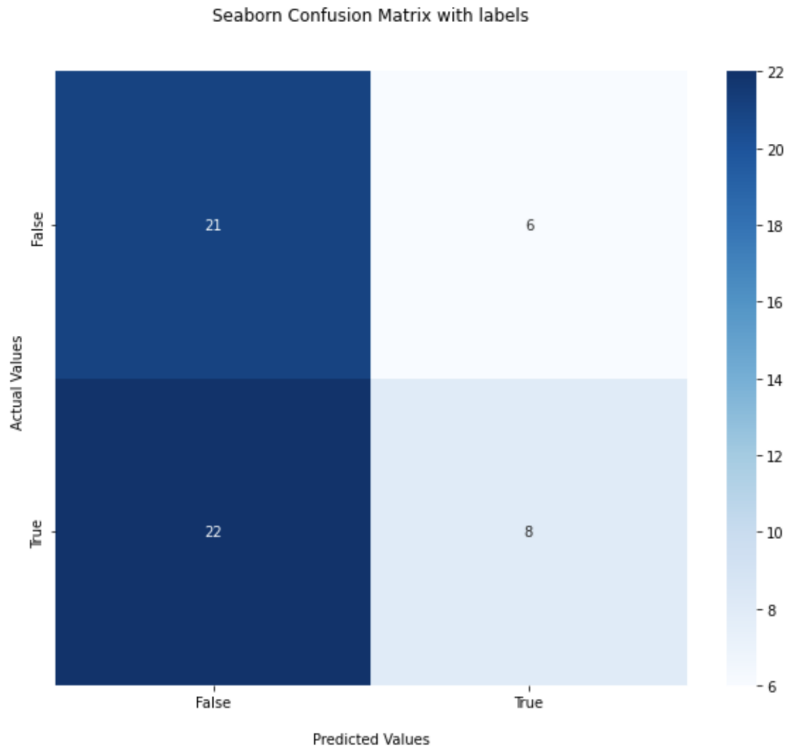

Fig 4.45 Confusion Matrix for model without sentiments

	volume	high	low	Y	sentiment_mag	sentiment
2021-02-10	1181.160162	34297.20	31712.00	0.0	0.177385	1
2021-02-13	642.220822	34950.00	33681.11	0.0	0.163294	1
2021-02-14	771.386256	36100.00	34231.02	1.0	0.269409	1
2021-02-15	965.418037	35560.29	33450.00	0.0	0.337748	1
2021-02-18	840.049697	38030.00	36639.55	0.0	0.163850	1
...	...	...	...	...	...	...
2021-08-14	321.380926	32959.04	31762.31	0.0	0.324488	1
2021-08-15	223.187002	32590.75	31720.01	1.0	0.289495	1
2021-08-16	242.737023	33065.28	32226.62	1.0	0.279850	1
2021-08-17	286.657226	32948.02	31910.89	0.0	0.289264	1
2021-08-18	251.151725	32491.62	31717.13	0.0	0.323912	1

Fig 4.46 data for the model with Vader sentiments

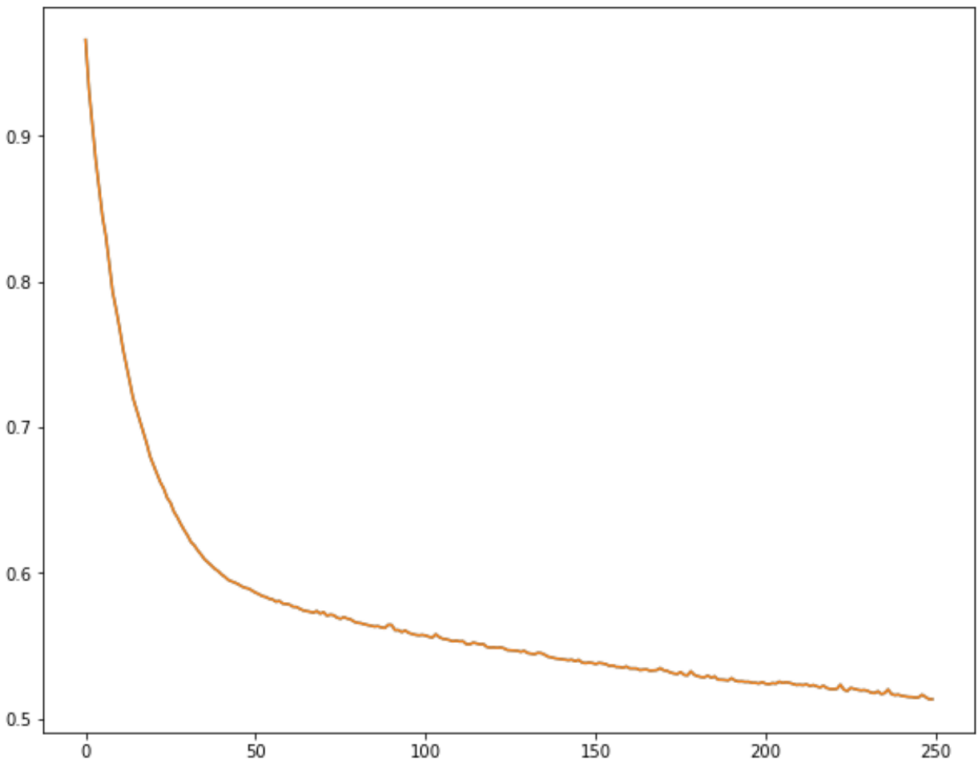

Fig 4.47 Loss for Model Training with Vader Sentiments

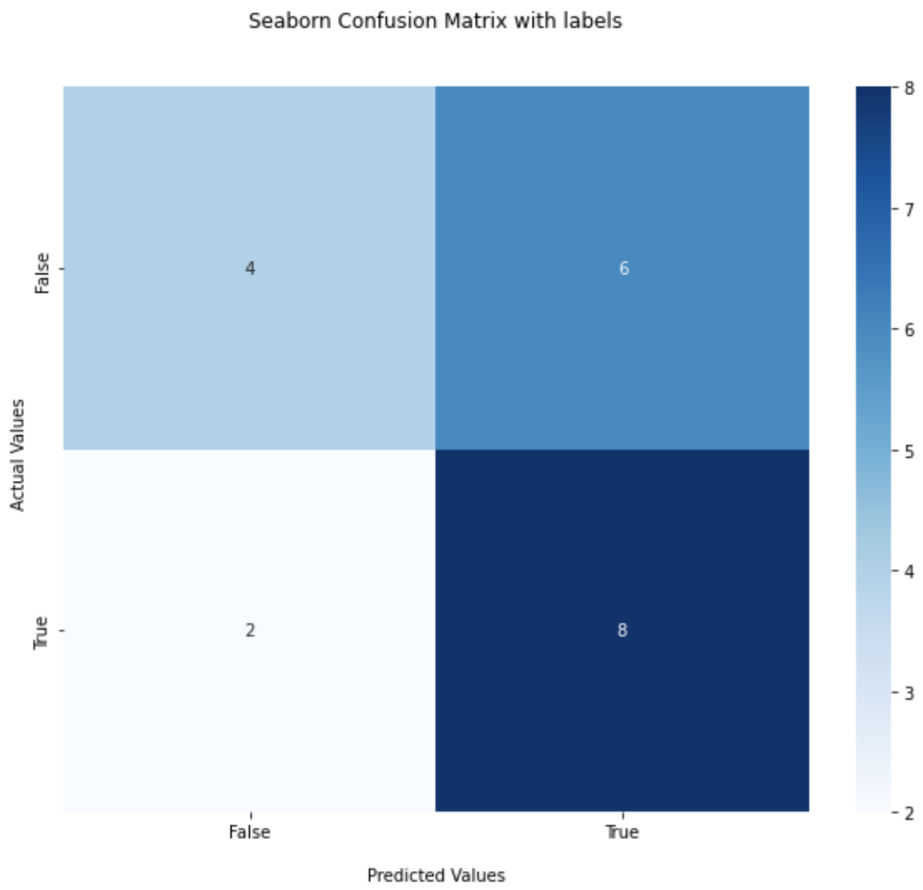

Fig 4.48 Confusion Matrix for Model with Vader Sentiments

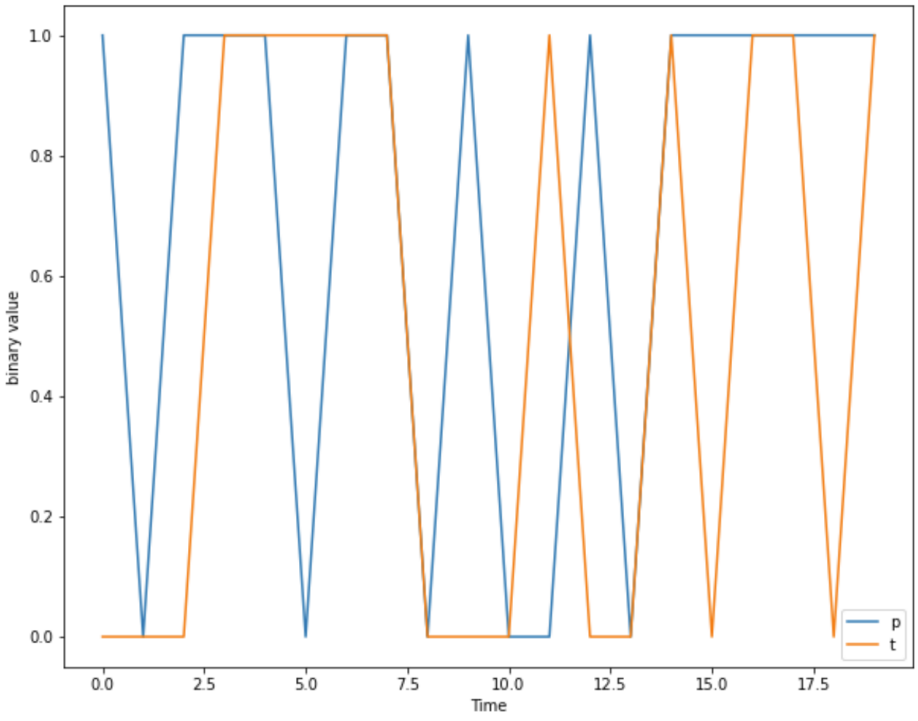

Fig 4.49 Predicted (p) vs Actual (t) values

	volume	high	low	Y	sentiment_mag	sentiment
2021-02-10	1181.160162	34297.20	31712.00	0.0	0.109335	1
2021-02-13	642.220822	34950.00	33681.11	0.0	0.108791	1
2021-02-14	771.386256	36100.00	34231.02	1.0	0.172871	1
2021-02-15	965.418037	35560.29	33450.00	0.0	0.219700	1
2021-02-18	840.049697	38030.00	36639.55	0.0	0.106882	1
...	...	...	...	...	...	...
2021-08-14	321.380926	32959.04	31762.31	0.0	0.189500	1
2021-08-15	223.187002	32590.75	31720.01	1.0	0.178322	1
2021-08-16	242.737023	33065.28	32226.62	1.0	0.157109	1
2021-08-17	286.657226	32948.02	31910.89	0.0	0.166063	1
2021-08-18	251.151725	32491.62	31717.13	0.0	0.190544	1

Fig 4.50 Data for Model with TextBlob Algorithm

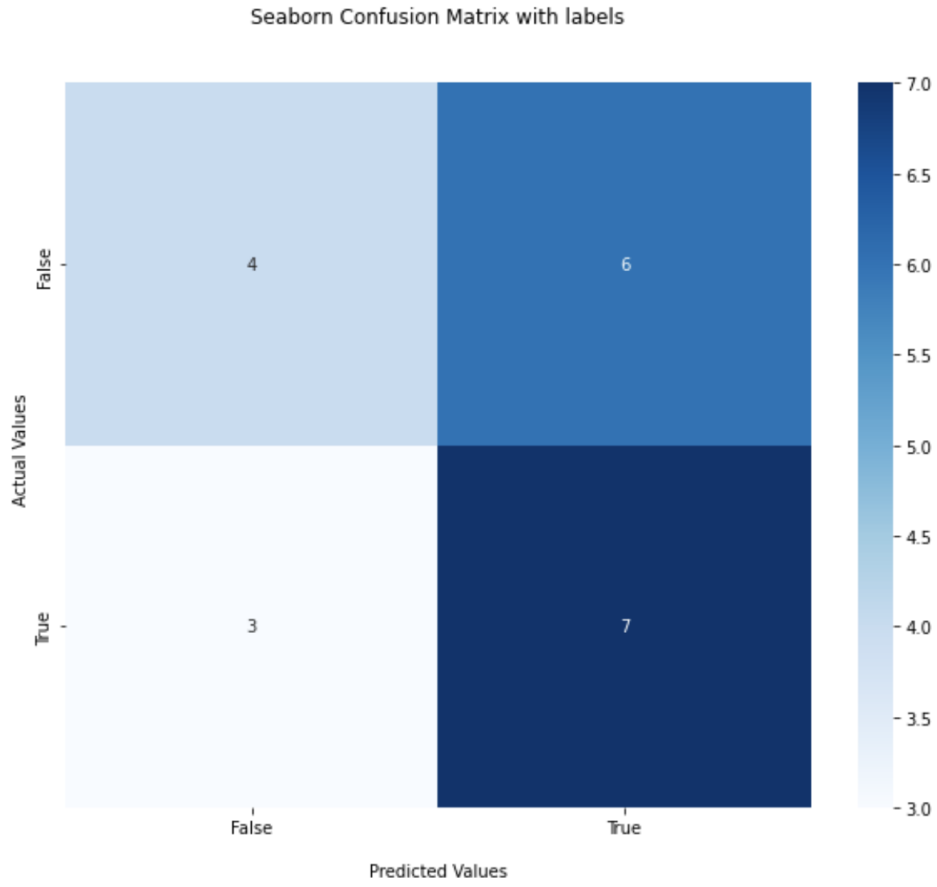

Fig 4.51 Confusion Matrix for Model with TextBlob algorithm

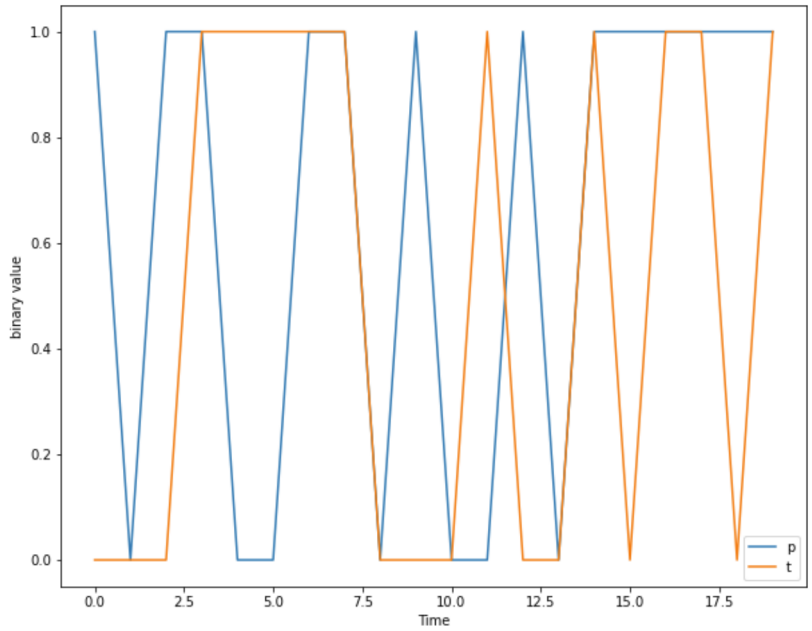

Fig 4.52 Actual (t) vs Predicted (p) line chart

```
Accuracy without sentiment:  50.877192982456144 %
Accuracy with vader:  60.0 %
Accuracy with blob:  55.00000000000001 %
```

Fig 4.53 Accuracy score for binary classification model

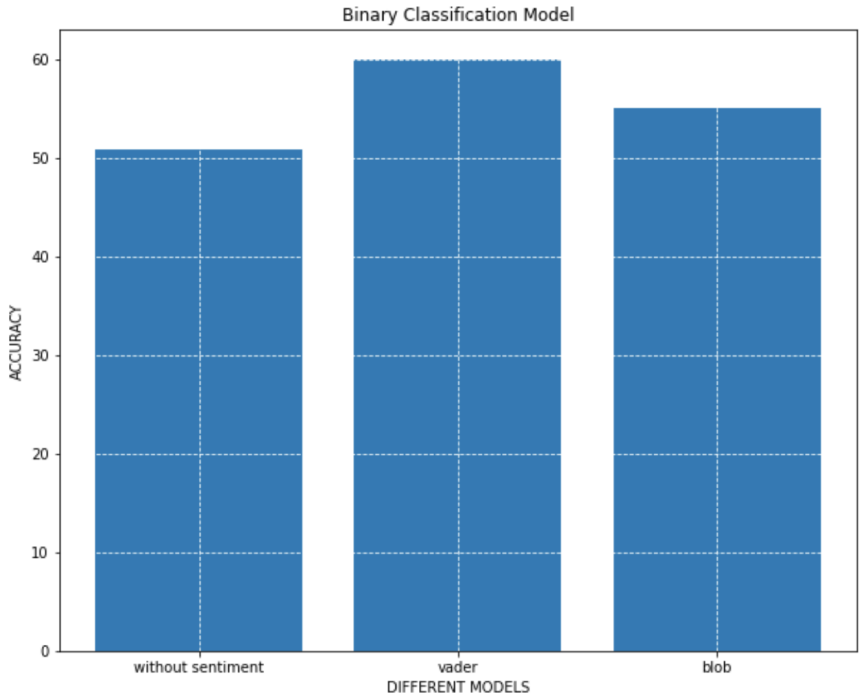

Fig 4.54 Accuracy bar chart for binary classification model

4.3.4 Model 4 Future Trends Regression

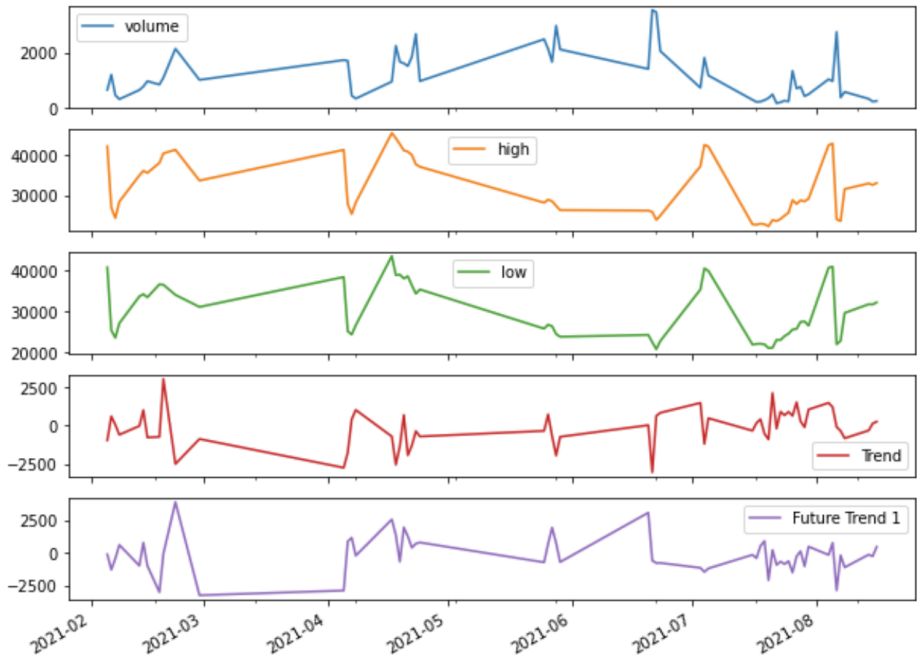

Fig 4.55 Data values for Future trends model

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(8, input_shape= (win_length, num_features), return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(tf.keras.layers.LSTM(16, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(16, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(23, kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(20, kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64, kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))

model.summary()
```

✓ 0.5s

```
Output exceeds the size limit. Open the full output data in a text editor
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_18 (LSTM)              (None, 5, 8)              416

 leaky_re_lu_12 (LeakyReLU)  (None, 5, 8)              0

 lstm_19 (LSTM)              (None, 5, 16)             1600

 leaky_re_lu_13 (LeakyReLU)  (None, 5, 16)             0

 dropout_12 (Dropout)        (None, 5, 16)             0

 lstm_20 (LSTM)              (None, 16)                2112

 dropout_13 (Dropout)        (None, 16)                0

 dense_24 (Dense)            (None, 23)                391

 dense_25 (Dense)            (None, 20)                480

 dense_26 (Dense)            (None, 64)                1344

 dense_27 (Dense)            (None, 1)                 65
...
Total params: 6,408
Trainable params: 6,408
Non-trainable params: 0
```

Fig 4.56 Creation of model with the summary description

```
plt.plot(predictions)
plt.plot(y_test)
plt.legend(['Predicted', 'Actual'])
plt.show()
```

✓ 0.1s

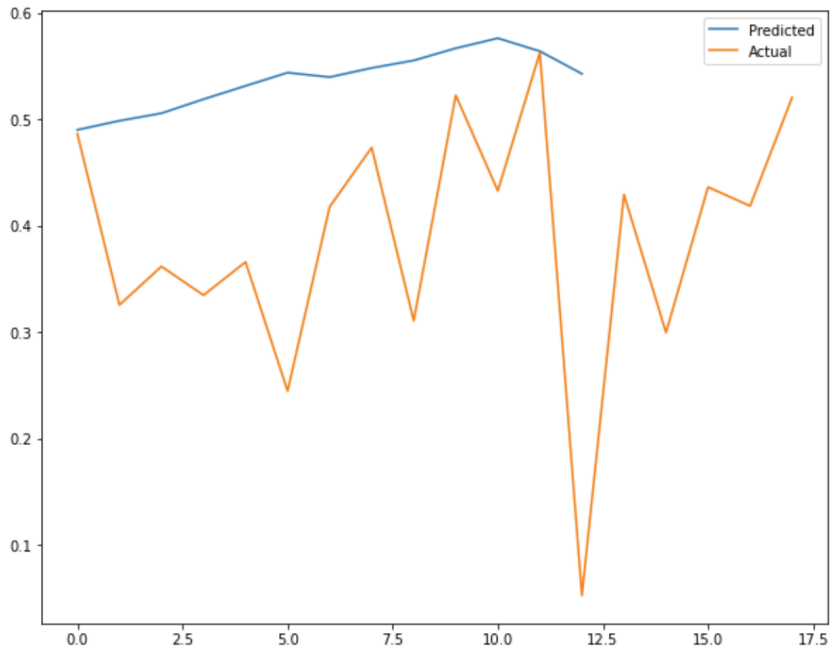

Fig 4.57 Actual vs Predicted Value

	volume	high	low	Trend	Future Trend 1	Predictions
2021-07-26	1339.283655	28839.56	25587.26	628.68	-1519.01	1826.797947
2021-07-27	696.799504	27882.60	25741.74	1515.33	-276.36	1855.671524
2021-07-28	758.954605	28800.00	27464.02	276.36	123.44	1879.647450
2021-07-29	417.142048	28510.32	27576.06	-111.72	-1044.70	1924.510716
2021-07-30	488.119281	29166.39	26530.98	1044.71	475.90	1967.081640
...	...	...	...	...	...	...
2021-08-07	370.966298	23635.29	22849.00	-352.90	-193.60	2048.280136
2021-08-08	573.071552	31565.98	29639.51	-831.63	-1124.66	2087.166709
2021-08-14	321.380926	32959.04	31762.31	-323.49	-143.29	2119.149325
2021-08-15	223.187002	32590.75	31720.01	131.36	-270.38	2077.695816
2021-08-16	242.737023	33065.28	32226.62	254.09	461.67	2005.700099

Fig 4.58 Output after inverse scaling without sentiments

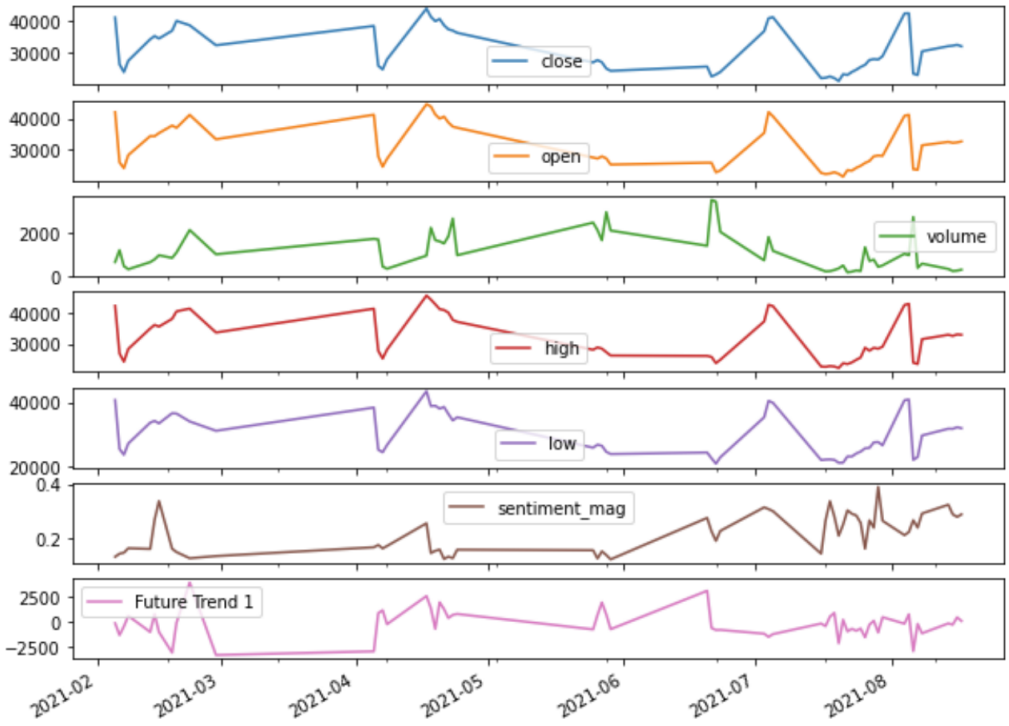

Fig 4.59 Data with the sentiment value from Vader NLP

	volume	high	low	sentiment_mag	Trend	Future Trend 1
2021-02-05	646.527211	42161.21	40778.59	0.134685	-960.19	-127.32
2021-02-06	1202.602896	26996.48	25379.30	0.145868	600.30	-1303.95
2021-02-07	453.865458	24385.16	23568.32	0.149594	46.23	-444.95
2021-02-08	307.734154	28451.25	27112.41	0.165777	-601.86	604.07
2021-02-13	642.220822	34950.00	33681.11	0.163294	-30.84	-995.55
...	...	...	...	...	...	...
2021-08-08	573.071552	31565.98	29639.51	0.292357	-831.63	-1124.66
2021-08-14	321.380926	32959.04	31762.31	0.324488	-323.49	-143.29
2021-08-15	223.187002	32590.75	31720.01	0.289495	131.36	-270.38
2021-08-16	242.737023	33065.28	32226.62	0.279850	254.09	461.67
2021-08-17	286.657226	32948.02	31910.89	0.289264	-503.85	95.40

59 rows × 6 columns

Fig 4.60 Data Table with Vader Nlp column

```
plt.plot(predictions2)
plt.plot(y_test2)
plt.legend(['Predicted', 'Actual'])
plt.show()
```

✓ 0.1s

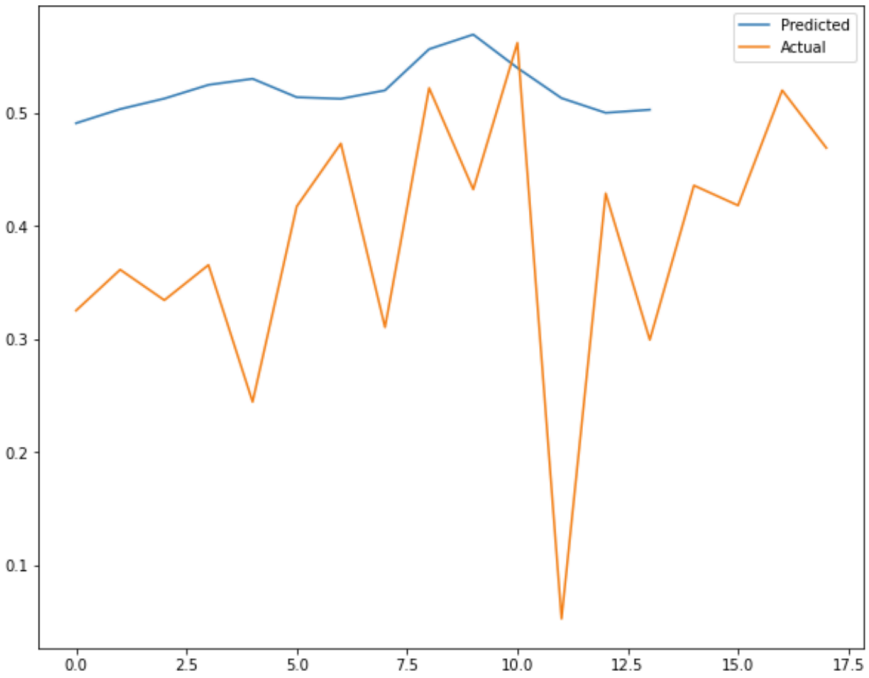

Fig 4.61 Actual vs Predicted for model with Vader Nlp input

	volume	high	low	sentiment_mag	Trend	Future Trend 1	Predictions
2021-07-26	1339.283655	28839.56	25587.26	0.164167	628.68	-1519.01	1831.387388
2021-07-27	696.799504	27882.60	25741.74	0.267390	1515.33	-276.36	1873.723611
2021-07-28	758.954605	28800.00	27464.02	0.240599	276.36	123.44	1905.399029
2021-07-29	417.142048	28510.32	27576.06	0.389384	-111.72	-1044.70	1946.609939
2021-07-30	488.119281	29166.39	26530.98	0.265796	1044.71	475.90	1964.958997
...	...	...	...	...	...	...	...
2021-08-08	573.071552	31565.98	29639.51	0.292357	-831.63	-1124.66	2098.017664
2021-08-14	321.380926	32959.04	31762.31	0.324488	-323.49	-143.29	1997.698572
2021-08-15	223.187002	32590.75	31720.01	0.289495	131.36	-270.38	1906.670347
2021-08-16	242.737023	33065.28	32226.62	0.279850	254.09	461.67	1862.529412
2021-08-17	286.657226	32948.02	31910.89	0.289264	-503.85	95.40	1871.836479

Fig 4.62 Data table with the predicted output

```
    plt.plot(predictions2)
    plt.plot(y_test2)
    plt.legend(['Predicted', 'Actual'])
    plt.show()
```

1] ✓ 0.1s

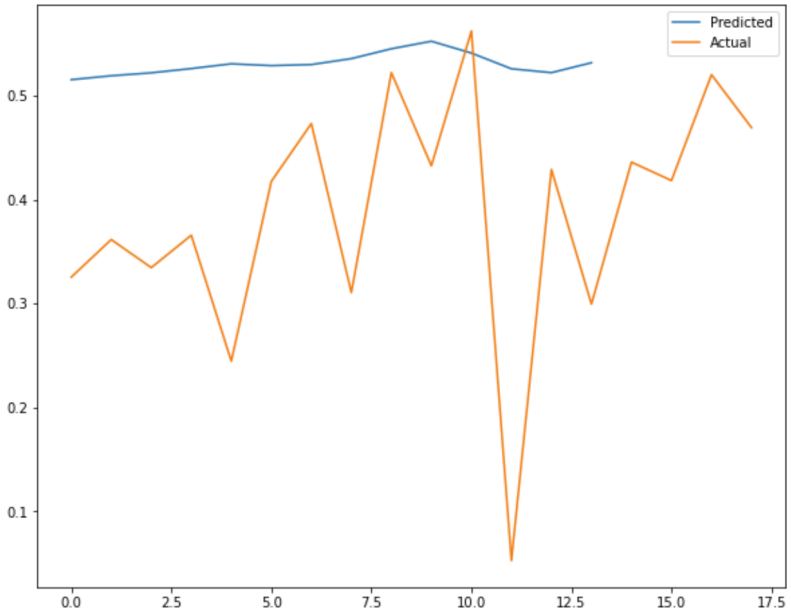

Fig 4.63 Actual vs Predicted with TextBlob algorithm

```
    plt.bar(['without sentiments', 'Vader', 'Textblob'] ,[acc1, acc2, acc3])
    plt.xlabel('DIFFERENT MODELS')
    plt.ylabel('ACCURACY')
    plt.grid(color='white', linestyle='dashed')
    plt.show()
```

✓ 0.1s

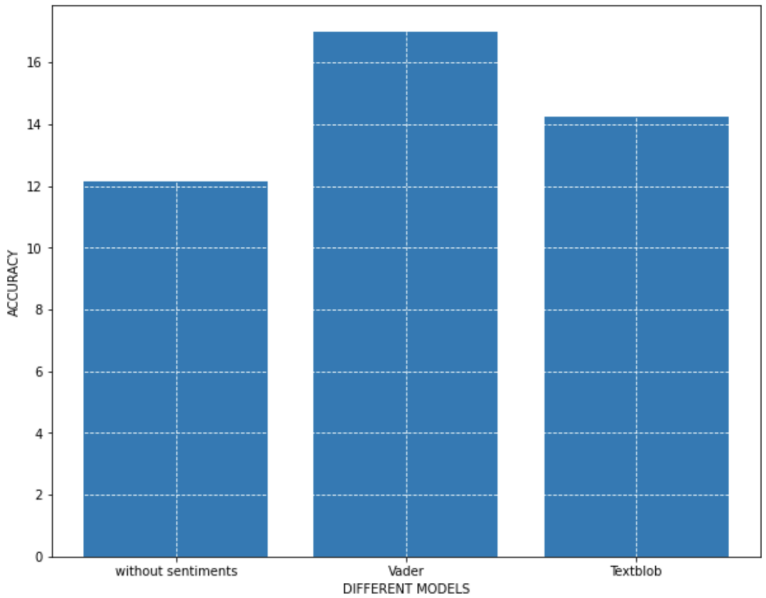

Fig 4.64 Final accuracy for all three models

4.4 Conclusion

From the graphs and figures portrayed above I was able to create and implement multiple models with different types and all of them showed higher accuracy of prediction with sentiments with either one as compared to models predicting without sentiments. All the graphs shown above have some kind of error margin and none of them have achieved the accuracy of 100%. All the models were working fine.

Chapter 5: **Discussion**

5.1 Introduction

In this part I explain the code and reason behind the use of this particular code along with the explanation for the output

There are multiple ways for the prediction of the data as we are in this project use multiple different method and algorithms to find out difference in percent between different ways of model prediction.. Regression model works in a way where it generates a relation between the dependent and independent variables. For the scope of this project we have at least three inputs for all the models and sometimes more so for that reason we use multivariable regression and binary classification model to find out the impact of the sentiments.

5.2 Reflection on Research

In this part after we retrieve the secondary data which is still in raw formal and requires pre processing as per the needs of the project where some places there are null values and other places there are values missing. Once the data is processed then we run twitter dataset through the sentiment analysis which can gives us the numerical values that can be counted towards the input for the model prediction. (Bitcoin Tweets | Kaggle. 2022)

In figure 4.1 and 4.2, we can see I have used fastquant API to retrieve the bitcoin prices for the required timeline in the bitcoin to GBP format as we can see from the output. I believe that was the quickest way to fetch the prices as it does not require keys or tokens for the process. Where we can see that the prices were going higher initially then there was sudden drop in the middle of may and starting of June as we can see from fig 4.1 downward slope and the gradual rise during the ending of July and starting of august.

In figure 4.3 there is the twitter secondary dataset which is already having converted values and the sentiments which can be seen in the last two columns, rest of the columns include the user data that is collected by twitter. In the next figure 4.4 we can see the score of the sentiments calculated where initially till may it was low but after may it gets sky rocket high. Fig 4.5 to fig 4.8 shows multiple ways to display the distribution of sentiments in all the tweets wherein in the figure 4.5 bar chart is displayed which shows the number of positive, negative and neutral tweets. In fig 4.6 pie chart is displayed which portrays the percent distribution of the tweet, 4.7 shows the histogram of the tweets where the tweets are segregated based on their score value. And the last fig 4.8 shows the line chart for the tweets for their timeline. These all tweets are converted with the Vader Neural language processing algorithm. In all the graphs we can see that maximum number of tweets were talking positive about bitcoin then after that 32% of the overall tweets were neutral which were not praising or condemning about the bitcoin.

Once we had sentiments for Vader algorithm next we wanted same conversion but with TextBlob algorithm. The fig displayed from fig 4.9 to fig 4.11 are all the outputs for the TextBlob sentiments. As shown in fig 4.9 the output is in the last two columns where the sentiment score i.e. Polarity and sentiment type are shown. Fig 4.10 exhibits the monthly polarity score and from the graph we can see it is very similar to the one from the Vader algorithm and depicts the same result. Last fig 4.11 shows the percent distribution where majority of them are positive then there are neutral sentiments and the last one are the negative ones.

5.3 Reflection on Implementation

5.3.1 Model 1 Multivariable Regression key outcomes

In this model the training and testing is done for the same day with and without the consideration of sentiment values. It can help us identify if the tweet sentiments have an impact on the outcome of prediction or not.

As we see from fig 4.12 where all the input values are displayed in the line chart and we can see all the features and target in the line format where Y is the difference between close and open of the same day. And in the next fig 4.13 we can see the creation of model where it includes 11 layers and the last one is the output layers and this models remains the same for across this first program. Once the model is trained then we predict the output and compare it with the actual value and we can see that from the fig 4.14 where it represents the table output along with all the data, although we can see there is an error margin which is expected from the model as it is at an early stage and it is difficult to achieve 100% accuracy. From fig 4.15 we can see the line graph for the actual value towards the predicted value and it shows a slight gap between the actual and predicted values. Fig 4.16 represent all the values in the line chart format where it even includes the predicted value which on the overall graphs is very close to the actual value showing chances of improvement in the near future.

On comparing fig 4.15 to fig 4.19 we can see that even the actual price i.e. 'Y' is different which is because some of the values in those date were counted null for sentiment conversion due to which it removed all of its data and completely ignored other aspects of that row. It is even seen in the fig 4.24 which is even different to the Vader because of different fundamentals of the algorithm.

The description above is for model without sentiment input, as we move on to model with sentiments first we start of with the Vader scores as an additional input with the historical data of volume, high, low as shown in fig 4.17 where we can see the values along with the compound which is the sentiment score for vader nlp algorithm, we can see in the figure that the line is broken in some part which is due to the null values found and the algorithm was unable to create a score for that null values. On the fig 4.18 it depicts the table where volume, high, low along with compound i.e. sentiment scores, will be used for prediction features or in other words as inputs for the model. Fig 4.19 similar to 4.15 which shows the graph for the actual price to the predicted price. Next for the fig 4.20 we see the overall plot for all the columns where we can compare all the values with the predicted outcome.

After the Vader sentiments we consider the TextBlob sentiments first we see fig 4.21 which is the input table along with the expected actual output value. Next in 4.22 it is the model fitting, compiling and early stopping feature where if the model is not learning then it will stop training and continue towards next step. Fig 4.23 and fig 4.24 are different formats of the output where one is the table format and the other one is line based graph format although.

After the overall accuracy is produces based on the given error margin percent we can see that model with TextBlob algorithm was having the highest accuracy of all the models in this set of models with multivariable regression with just sentiment score.

5.3.2 Model 2 Multivariable regression

In this model we consider the sentiment score along with the binary conversion of sentiments where we convert the score if it is greater than 0 then it is counted as 1 or if it is equal to or less than zero then considered 0. Based on this a table is generated which has the sentiment values as binary converted data. In fg 4.27 we see that there is a description of the data values like total count, mean, standard values, 25% value of that column and the rest, the trend column here is the difference between the close and open price for the same day. This explains that the dataset is working fine and function of it. Next in the fig 4.28 is the code snippet for the train test splitting and the time series generator for train and test where it is checking the train data and test data, putting common window length which is the initial gap to skip the prediction and start the prediction. Batch size is defined so

that it trains quicker and efficient in small batches. Next in 4.29 the model creation is shown where I have 11 layers and the last one is the output layers. In fig 4.30 it shows the early stopping along with the model compile and model fit, these syntax are used to compile and train the model as we can see from the epochs showing the loss and other information regarding the training. Fig 4.31 and fig 4.32 both are outputs in different formats where the first one is in tabular format and the second one is the graph comparison between this and the actual values.

From fig 4.33 it is the start for creation of model with Vader sentiments input. As shown in the table it has both the score value along with the binary value for the sentiments. Fig 4.34 shows the logic snippet for the binary conversion. This has the same model as fig 4.29 and run through the same process just the inputs got varied and fig 4.35 displays the output table where the Trend is actual value and 'app_pred' is the predicted value. Fig 4.36. shows the predicted vs actual in the line chart formart where as we can see in the predicted value there is slight beds up and down but on an overall it do not have as much as the actual value.

As coming from Vader we move on towards the TextBlob fig 4.37 shows the input features along with the actual Trend price. As it follows the same model so run through the train test and model predicting phase and in fig 4.38 shows the predicted value labelled AppPPred next to the actual value labelled Trend. Fig 4.39 shows the line graph and it displays the similar effect between the predicted and actual line where margin is present between both the values. Fig 4.40 and 4.41 are the final outcome of this project where the accuracy between all three inputs is shown. Model with TextBlob was the highest with the accuracy of 77% next was with Vader at 76% while the model without any sentiment s was behind at 68% this accuracy is based on the mean absolute percent error.

5.3.3 Model 3 Binary Classification

In this model instead of sentiments being binary the output is converted to binary value which if implemented on a rea world scenario can ideally inform if the price will go high or low In this model we take the historical values like volume, high and low and then along with the Vader Sentiment Value and TextBlob sentiment value. Fig 4.42 shows the output Y which is the difference between close and open and it is shown in graph that it is continuously up or down. This fig is the graphical representation of the data table. In fig 4.43 we can see the creation of model with the loss graph. The model has 5 layers where the last one being the output. In this model we have used sigmoid activation in the last layer to give binary impact. In the image there is loss graph, it is decreasing showing that the model is training decreasing the loss thus showing an improvement. Fig 4.44 shows the binary output of both the predicted and the actual prices of trends. Fig 4.45 shows the confusion matrix where in this matrix false positives are the highest then there are false negatives showing not the best accuracy, fig 4.46 shows the table with the Vader sentiments and fig 4.47 show the loss graph for that model having one of the input as Vader sentients. Fig 4.48 shows the confusion matrix for that model where true positives are the highest. Fig 4.49 shows the graph of actual vs predicted values where in some lines it is overlaying on top of each other showing the ability of correct prediction. Next is the model with the TextBlob input the fig 4.50 represents, fig 4.51 and fi 4.52 depict the accuracy of that model in confusion matrix and line chart respectively. The final two figures fig 4.53 and fig 4.54 show the comparison between the accuracy of models without the sentiments to Vader sentiment input model to TextBlob input model. Here the Vader model received the highest accuracy as compared to TextBlob which achieved that in the previous two model.

5.3.4 Model 4 Future Trends Regression

In this model it is the same as the previous regression model just the difference here is I try to predict the future trends which is technically the difference between today's close and upcoming tomorrow's close. The use of this is if the difference is positive then the price tomorrow will be decrease by the margin of the predicted amount and if it is negative then it will be higher with the margin of that predicted amount. Fig 4.55 shows the plot of the data table where the future trend 1 is the difference between today's close and tomorrow's close, next the fig 4.56 shows the model with the summary

of the model. Fig 4.57 and fig 4.58 show the graph and the table for the predicted values in comparison to actual values where we see there is some margin but still trying to close the gap. Fig 4.59 to fig 4.61 show the output for model with the Vader sentiment as one of the input. Where fig 4.59 and 4.60 represent the input and the actual values while fig 4.61 shows the predicted to the actual chart, the predicted value is based on the same model mentioned before. Fig 4.62 is the input table for input with TextBlob values and fig 4.63 is the output for that model based on the same prediction model i.e. fig 4.56. fig 4.64 shows the final accuracy for all 3 models where Vader has the highest accuracy, TextBlob comes in second and model without sentiments input is the last one. This is the only model of all that focuses on future prediction and has low accuracy. This model predicts the difference between the close of today to close of tomorrow.

5.4 Reflection on Objectives

- To research and find a dataset that needs to be used in the project and different software related to prediction

As mentioned above I was able to retrieve the bitcoin prices and received a secondary twitter dataset which accomplishes this object

- To investigate and identify the most appropriate tools and API required for the project

On implementing the model all the required APIs were identified and used to make the most of them.

- To research and identify the relationship between social media sentiments for crypto market prediction

From the results and discussion it was found that tweets do have an impact on prediction as we see the improvement in accuracy for the models with sentiments as compared to model without sentiments.

- To develop a working demo of the project and to analyse and evaluate the demo providing the possible chances of being accurate

With my implementation the project is successfully able to predict the outcome for that dataset giving out the accuracy as per the model, where in some model have higher accuracy while other have lower accuracy.

- To conclude the model, show what could be the possible outcome if implemented in a real-world scenario

Chapter 6: **Conclusion**

6.1 Introduction

In this part I briefly summarise the issues, major outcome and what are the possible implications for the future where how can it be used to improve the economy, provide people with a better understanding about the relation between social media and crypto market

6.2 Summary of Key findings

The accuracy charts which are fig 4.25 , fig 4.40, fig 4.50 and fig 4.64 are the main outcomes of this project where we can see in the fig 4.25 TextBlob got the highest accuracy of 79% and in the next model as well TextBlob got the highest accuracy of 77% where in the last two mode Vader got the highest accuracy of 60% and 17% respectively which concludes the research that sentiments have a higher impact on prediction as compare to model with just historical data.

6.3 Project Limitations

This project had some limitations as the dataset was too big and required to be processed. Next was the tight schedule of time which required higher time management to make sure I learned the python programming language upto a level which would require me to build a neural network. Along with that it is difficult to search for secondary dataset which are required as per project.

6.4 Future Research and Recommendations

This project can be extended and improved in future where the prediction accuracy can be improved, along with addition of more layers and including a large scale data which could essentially assist in improving the accuracy and this could be deployed over a network where beginners can learn and understand use of machine learning towards economy and it an provide a safer option for investing to public.

Reference List

Abraham, J et al., 2018. Cryptocurrency price prediction using tweet volumes and sentiment analysis. SMU Data Science Review, 1(3), p.1.

Alessandretti, L., et al, 2018. Machine learning the cryptocurrency market. Available at SSRN 3183792.

Alexopoulos, E.C., 2010. Introduction to multivariate regression analysis. Hippokratia, 14(Suppl 1), p.23.

Árnason, S.L., 2015. Cryptocurrency and Bitcoin. A possible foundation of future currency: why it has value, what is its history and its future outlook (Doctoral dissertation).

Aste, T., 2019. Cryptocurrency market structure: connecting emotions and economics. Digital Finance, 1(1), pp.5-21.

Bitcoin Tweets | Kaggle. 2022. Bitcoin Tweets | Kaggle. [ONLINE] Available at: https://www.kaggle.com/datasets/kaushiksuresh147/bitcoin-tweets [Accessed 09 May 2022].

Bonta, V. and Janardhan, N.K.N., 2019. A comprehensive study on lexicon based approaches for sentiment analysis. Asian Journal of Computer Science and Technology, 8(S2), pp.1-6.

Bouri,et al ,.2019. Trading volume and the predictability of return and volatility in the cryptocurrency market. Finance Research Letters, 29, pp.340-346.

Buterin, V., 2014. A next-generation smart contract and decentralized decentralised application platform. white paper, 3(37).

Carraher, S.M., et al., 2006. Customer service, entrepreneurial orientation, and performance: A study in health care organizations organisations in Hong Kong, Italy, New Zealand, the United Kingdom, and the USA. Journal of Applied Management and Entrepreneurship, 11(4), p.33.

Chen, G.H., Nikolov, S. and Shah, D., 2013. A latent source model for nonparametric time series classification. arXiv preprint arXiv:1302.3639.

Chohan, U.W., 2021. A history of Dogecoin. Discussion Series: Notes on the 21st Century.

Colianni, S., Rosales, S. and Signorotti, M., 2015. Algorithmic trading of cryptocurrency based on Twitter sentiment analysis. CS229 Project, 1(5).

Dwyer, G.P., 2015. The economics of Bitcoin and similar private digital currencies. Journal of financial stability, 17, pp.81-91.

Edosomwan, S., et al., 2011. The history of social media and its impact on business. Journal of Applied Management and entrepreneurship, 16(3), p.79.

Farell, R., 2015. An analysis of the cryptocurrency industry. Wharton Research Scholars. 130. https://repository.upenn.edu/wharton_research_scholars/130

Georgoula, I et al., 2015. Using time-series and sentiment analysis to detect the determinants of bitcoin prices. Available at SSRN 2607167.

Huang, X., et al., 2021, April. Lstm based sentiment analysis for cryptocurrency prediction. In International Conference on Database Systems for Advanced Applications (pp. 617-621). Springer, Cham.

Laskowski, M. and Kim, H.M., 2016, July. Rapid prototyping of a text mining application for cryptocurrency market intelligence. In 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI) (pp. 448-453). IEEE.

Leau, Y.B., Loo, W.K., Tham, W.Y. and Tan, S.F., 2012. Software development life cycle AGILE vs traditional approaches. In International Conference on Information and Network Technology (Vol. 37, No. 1, pp. 162-167).

Lee, E., 2013. Impacts of social media on consumer behavior: decision making process.

Lenton, G. (2021). A short history of cryptocurrencies. Central Penn Business Journal, Retrieved from https://www.proquest.com/trade-journals/short-history-cryptocurrencies/docview/2555245923/se-2?accountid=17234

Liu, B. (2010). Sentiment Analysis and Subjectivity. In N. In-durkhya & F. Damerau (Eds.), Handbook of Natural Language Processing (2nd ed.). Boca Raton, FL: Chapman & Hall.

Luther, W.J. and White, L.H., 2014. Can bitcoin become a major currency?. GMU Working Paper in Economics No. 14-17, Available at SSRN: https://ssrn.com/abstract=2446604 or http://dx.doi.org/10.2139/ssrn.2446604

Martin, J., Cunliffe, J. and Munksgaard, R., 2019. A Modern-day History of Cryptomarkets. In Cryptomarkets: A Research Companion. Emerald Publishing Limited.

Masum, M., et al., 2020, December. r-LSTM: Time Series Forecasting for COVID-19 Confirmed Cases with LSTMbased Framework. In 2020 IEEE International Conference on Big Data (Big Data) (pp. 1374-1379). IEEE.

Matta, M., Lunesu, I. and Marchesi, M., 2015, June. Bitcoin Spread Prediction Using Social and Web Search Media. In UMAP workshops (pp. 1-10).

McNally, S., Roche, J. and Caton, S., 2018, March. Predicting the price of bitcoin using machine learning. In 2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP) (pp. 339-343). IEEE.

Nagisetty, A. and Gupta, G.P., 2019, March. Framework for detection of malicious activities in IoT networks using Keras deep learning library. In 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC) (pp. 633-637). IEEE.

Nian, L.P. and Chuen, D.L.K., 2015. Introduction to bitcoin. In Handbook of digital currency (pp. 5-30). Academic Press.

Papenbrock, J., Schwendner, P. and Sandner, P.G., 2021. Can Adaptive Seriational Risk Parity Tame Crypto Portfolios?. Available at SSRN 3877143.

Phillips, R.C., 2019. The Predictive Power of Social Media within Cryptocurrency Markets (Doctoral dissertation, UCL (University College London)).

ReviseSociology. 2022. The Steps of Quantitative Research – ReviseSociology. [ONLINE] Available at: https://revisesociology.com/2017/11/26/the-steps-of-quantitative-research/. [Accessed 08 May 2022].

Sarker, I.H., Kayes, A.S.M. and Watters, P., 2019. Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage. Journal of Big Data, 6(1), pp.1-28.

Sattarov, O., et al., 2020, November. Forecasting Bitcoin price fluctuation by Twitter sentiment analysis.

Segendorf, B., 2014. What is bitcoin. Sveri gesRiksbankEconomicReview, 2014, pp.2-71.

Shah, D. and Zhang, K., 2014, September. Bayesian regression and Bitcoin. In 2014 52nd annual Allerton conference on communication, control, and computing (Allerton) (pp. 409-414). IEEE.

Smilkov, D., et al, 2019. Tensorflow. js: Machine learning for the web and beyond. Proceedings of Machine Learning and Systems, 1, pp.309-321.

Stenqvist, E. and Lönnö, J., 2017. Predicting Bitcoin price fluctuation with Twitter sentiment analysis.

Tandon, C., et al., 2021. 'How can we predict the impact of the social media messages on the value of cryptocurrency? Insights from big data analytics', International Journal of Information Management Data Insights, vol. 1, no. 2, pp. 1-8.

Tumasjan, A., et al, 2010, May. Predicting elections with twitter: What 140 characters reveal about political sentiment. In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 4, No. 1).

Velankar, S., Valecha, S. and Maji, S., 2018, February. Bitcoin price prediction using machine learning. In 2018 20th International Conference on Advanced Communication Technology (ICACT) (pp. Pant, D.R., Neupane, P., Poudel, A., Pokhrel, A.K. and Lama, B.K., 2018, October. Recurrent neural network based bitcoin price prediction by twitter sentiment analysis. In 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS) (pp. 128-132). IEEE.144-147). IEEE.

Appendix A - Initial Project Proposal

Project (CN6000) Initial Proposal Form

Programme: BSc(Hons.) Computer Science Year: 2021

Student Number: 1930306

Proposed Title: Crypto Currency and Stock Market Prediction

Proposed Aim: Explain the reasoning and provide a working demo of the model for the given topic based on real life situation

Rationale: In current world where social media is some what giving out the guesses for the world economy there could be an algorithm which could help in identifying these trends and predict to the near future outcomes which can assist financial mega giants handle their finances in a way where if the markets are booming then everyone could get a share of it while it is booming and if it is falling everyone can be alerted in time.

Supervisor: Dr. Mustansar Ali Ghazanfar

Appendix B - Final Project Proposal

Project (CN6000) Final Proposal Form

Programme: BSc(Hons.) Computer Science Year: 2021

Student Number: 1930306

Proposed Title: Cryptocurrency Prediction based on Social Media and machine learning

Proposed Aim: To design and implement a cryptocurrency prediction based on the relation between cryptocurrency and social media.

Objectives: • - To research and find a dataset that needs to be used in the project and different software related to prediction

- - To investigate and identify the most appropriate tools and API required for the project
- - To research and identify the relationship between social media and crypto market
- - To develop a working demo of the project.
- - To analyse and evaluate the demo providing the possible chances of being accurate
- - To conclude the model, show what could be the possible outcome if implemented in a real-world scenario

Rationale: In the current world where social media is somewhat giving out the guesses for the world economy there could be an algorithm that could help in identifying these trends and predict the near future outcomes which can assist financial mega giants handling their finances in a way where if the markets are booming then everyone could get a share of it while it is booming and if it is falling everyone can be alerted in time. The project will be based on a historical dataset along with identifying the current situation with machine learning.

Facilities required: A working PC, Google Colab or Jupyter Notebooks for Python, different APIs to gather the data for the relation, academic journals or research papers on machine learning.

Supervisor: Dr. Mustansar Ali Ghazanfar

Appendix C – Source Code

Link to secondary dataset: https://www.kaggle.com/datasets/kaushiksuresh147/bitcoin-tweets

Link to access whole code is:

For OneDrive: https://uelac-my.sharepoint.com/:f:/g/personal/u1930306_uel_ac_uk/EszW9xV66XRLo_11Mpzen6EB3pT1LyR89uwYQVxP1EjB0w

From the link provided anyone can look into the project and download and clone it along with the data set

6.5 Code for retrieving Bitcoin data

All the Files were save in jupyter notebook i.e. ipynb format

```
# %%
!pip install fastquant
import pandas as pd
from fastquant import get_crypto_data
import fastquant as fq
#get the stock quote
start = '2021-02-10'
end = '2021-08-18'

df = get_crypto_data('BTC/GBP', start, end)
df

# %%
import matplotlib. pyplot as plt
#visualise the closing history
plt.figure(figsize=(16,8))
plt.title('Close price History for bitcoin')
plt.plot(df['close'])
plt.xlabel('dt', fontsize=18)
plt.ylabel('Close prize GBP(£)', fontsize=18)
plt.show()

# %%
#saving data as csv file
df.to_csv('C:/Users/parek/OneDrive - University of East London/Crypto
Prediction with machine learning and sentiment
analysis/Dataset/btc_value.csv')# ,  dtype={'text': 'str', 'date':
'int'})#,sep='\t')
```

6.6 Vader Sentiment Conversion

```
# %% [markdown]
# Vadel NLP sentiment analysis
# References
# #
#

# %%
%pip install vaderSentiment
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
obj = SentimentIntensityAnalyzer();


# %% [markdown]
# import the main file

# %%
import pandas as pd

dfbit = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/btc_refined.csv')  # ,  dtype={'text': 'str', 'date':
'int'})#,sep='\t')
#dfbit= pd.read_csv('/Users/rohanparekh/UEL/UEL Research/btc_refined.csv')
#for macos its different
low_memory = False

# %% [markdown]
# checking if the file is running

# %%
dfbit.head()

# %%
dfbit

# %%
# removing NA values from data frame
#dfbit.dropna(inplace=True)

# %%
sentiment_dict = obj.polarity_scores(dfbit.iloc[0]['text'])
#print(sentiment_dict)
sentiment_dict

# %%
dfbit['scores'] = dfbit['text'].apply(lambda
review:obj.polarity_scores(review)) #avg time 4 mins
```

```
dfbit['compound'] = dfbit['scores'].apply(lambda score_dict:
score_dict['compound'])

dfbit['sentiment_type']=''
dfbit.loc[dfbit.compound>0,'sentiment_type']='POSITIVE'
dfbit.loc[dfbit.compound==0,'sentiment_type']='NEUTRAL'
dfbit.loc[dfbit.compound<0,'sentiment_type']='NEGATIVE'

dfbit.head()

# %%
dfbit[['date', 'text','hashtags','scores','compound','sentiment_type']]

# %% [markdown]
# Calculate daily average from hourly data

# %%
dfvad=dfbit[['date','compound']]
dfvad['date'] = pd.to_datetime(dfvad.date)#, errors='ignore')
dfvad.set_index('date', inplace=True)
dayvad = dfvad.resample('D').mean()
monthvad = dfvad.resample('M').mean()

# %%
dayvad

# %%
monthvad

# %%
import matplotlib.pyplot as plt
plt.figure(figsize=(16,8))
plt.plot(dayvad['compound'])

# %%
monthvad.plot()

# %%
dayvad.to_csv('C:/Users/parek/UEL OneDrive/Research Google
Colab/dayvad.csv')
# %% [markdown]
# //Code Dump
# %%
#mount to the drive
from google.colab import drive
drive.mount('/content/drive')

# %%
sentence = " you are happy and very happy"#trial with one sentence
sentiment_dict = obj.polarity_scores(sentence)
```

```
print(sentiment_dict)

# %%
%pip install matplotlib

# %%
import matplotlib.pyplot as plt
dayvad.plot()

# %%
dayvad.compound.hist()

# %%
dfvad.info()

# %%
dfvad.sentiment_type.value_counts().plot(kind='bar',title="sentiment
analysis")

# %%
import matplotlib. pyplot as plt
import matplotlib
plt.pie(dfbit.sentiment_type.value_counts()
,labels=['positive','neutral','negative'], autopct='%1.1f%%')
plt.show()

# %%
dfbit.compound.hist()

# %%
dfbit.sentiment_type.hist() #same as above

# %%
dfvad['date'] = pd.to_datetime(dfvad.date)#, errors='ignore')

# %%
import numpy
import matplotlib.pyplot as plt
plt.figure(figsize=(16,8))
#plt.title('Close price History for bitcoin')
plt.plot(dfvad['date'],dfvad['compound'])
plt.xlabel('date', fontsize=18)
plt.ylabel('polarity', fontsize=18)
plt.show()

# %%
dfvad.to_csv('C:/Users/parek/UEL OneDrive/Research Google
Colab/btc_vader.csv')
dfvad.info
```

6.7 TextBlob Sentiment Conversion

```
# %% [markdown]
# Sentiment analysis for csv file

# %%
%pip install textblob
from textblob import TextBlob
from nltk import tokenize
import pandas as pd

dfbit = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/btc_refined.csv')#,sep='\t')

low_memory=False

# %%
dfbit[['user_name','user_location','date','text','hashtags']]

# %%
dfbit

# %%
dfbit.drop_duplicates(subset ="text", keep = "first" , inplace = True)
dfbit['text'] = dfbit['text'].astype('str')


# %%
#avg 6 min runtime
def get_polarity(text):
  return TextBlob(text).sentiment.polarity
dfbit['Polarity'] = dfbit['text'].apply(get_polarity)

# %%
dfbit['Sentiment_Type']=''
dfbit.loc[dfbit.Polarity>0,'Sentiment_Type']='POSITIVE'
dfbit.loc[dfbit.Polarity==0,'Sentiment_Type']='NEUTRAL'
dfbit.loc[dfbit.Polarity<0,'Sentiment_Type']='NEGATIVE'

# %%
dfbit.head()

# %%

dfbit.Sentiment_Type.value_counts().plot(kind='bar',title="sentiment
analysis")

# %%
dfbit[['date', 'text','hashtags','Polarity','Sentiment_Type']]
```

```
# %%
dfbit.Sentiment_Type.value_counts()

# %%
dfbit.Polarity.value_counts()

# %%
dfbit.hist()

# %%
import matplotlib. pyplot as plt
import matplotlib
plt.pie(dfbit.Sentiment_Type.value_counts()
,labels=['positive','neutral','negative'], autopct='%1.1f%%')
plt.show()

# %%
dfbit.Polarity.hist()

# %%
dfblob=dfbit[['date','Polarity']]
dfblob['date'] = pd.to_datetime(dfblob.date)#, errors='ignore')
dfblob.set_index('date', inplace=True)
dayblob = dfblob.resample('D').mean()
dayblob

# %%
monthblob = dfblob.resample('M').mean()
monthblob

# %%
dayblob.plot()

# %%
monthblob.plot()

# %%
dayblob.Polarity.hist()

# %%
import matplotlib. pyplot as plt
#not working yet
plt.figure(figsize=(16,8))
plt.plot(dayblob['Polarity'])
plt.legend(['date','Polarity'])
plt.xlabel('date')
plt.ylabel('sentiment')
plt.show()
```

```
# %%
dayblob.to_csv('C:/Users/parek/UEL OneDrive/Research Google
Colab/dayblob.csv')

# %% [markdown]
# //Code Dump

# %%
#mount to the drive
from google.colab import drive
drive.mount('/content/drive')

# %%
from textblob import TextBlob

# %%
blob = TextBlob("Analytics Vidhya is a great platform to learn data
science.")
print (blob)
blob.sentiment
#test statement

# %%
def sentiment_analysis(tweet):
  def getSubjectivity(text):
   return TextBlob(text).sentiment.subjectivity

# %%
#Create a function to get the polarity
def getPolarity(text):
  return TextBlob(text).sentiment.polarity
```

6.8 Model 1 Multivariable Regression

```
# %% [markdown]
# // model without sentiments MODEL 1

# %%
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

import tensorflow as tf
```

```
df=pd.read_csv('C:/Users/parek/OneDrive - University of East London/Crypto
Prediction with machine learning and sentiment
analysis/Dataset/btc_value.csv')
#df = pd.read_csv('/Users/rohanparekh/UEL/UEL Research/btc_value.csv')

df['dt']=pd.to_datetime(df.dt)
df=df.set_index(pd.DatetimeIndex(df['dt']))
#
#df.iloc[:,1]
df['Y'] = df['close'] - df['open']
df = df[['volume','high','low','Y']]
df

# %%
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import TimeseriesGenerator
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import tensorflow as tf
mpl.rcParams['figure.figsize'] = (10, 8)
mpl.rcParams['axes.grid'] = False

# %%
df.info()

# %%
df_input = df [['volume','high','low','Y']] #add polarity in here when added

# %%
df.plot(subplots=True)

# %%
df_input.describe()

# %%
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(df_input)

# %%
data_scaled

# %%
data_scaled[:,0:] #change here

# %%
features=data_scaled[:,0:3]  #change heere
target=data_scaled[:,3]

# %%
```

```
TimeseriesGenerator(features, target, length=2, sampling_rate=1,
batch_size=1)[0]

# %%
x_train, x_test, y_train, y_test = train_test_split(features, target,
test_size=0.30, random_state=42, shuffle = False)

# %%
x_train.shape

# %%
x_test.shape

# %%
win_length=5
batch_size=13
num_features=3 #this number will change when polarity is added
train_generator = TimeseriesGenerator(x_train, y_train, length=win_length,
sampling_rate=1, batch_size=batch_size)
test_generator = TimeseriesGenerator(x_test, y_test, length=win_length,
sampling_rate=1, batch_size=batch_size)



# %%
train_generator[0]

# %% [markdown]
# model creation without sentiments

# %%
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape= (win_length, num_features),
return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.LSTM(128, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(64, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(128,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(32,
kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))

# %%
```

```
model.summary()

# %%
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                  patience=2,
                                                 mode= 'min')
model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit_generator(train_generator, epochs=50,
                    validation_data=test_generator,
                    shuffle=False,
                    callbacks=[early_stopping], verbose=1)

# %%
model.evaluate_generator(test_generator, verbose=0)

# %%
predictions=model.predict_generator(test_generator)

# %%
predictions.shape[0]

# %%
predictions

# %% [markdown]
# accuracy for model without sentiments

# %%
import numpy as np

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mae1 = mean_absolute_percentage_error(y_test, predictions)
acc1 = 100-mae1
acc1

# %%
y_test

# %%
x_test

# %%
x_test[:,0:][win_length:]

# %%
```

```
df_pred=pd.concat([pd.DataFrame(predictions),
pd.DataFrame(x_test[:,0:3][win_length:])],axis=1) #change here

# %%
df_pred

# %%
rev_trans=scaler.inverse_transform(df_pred)

# %%
rev_trans

# %%
df_final=df_input[predictions.shape[0]*-1:]

# %%
df_final.count()

# %% [markdown]
# App pred vs Y

# %%
df_final['App_Pred']=rev_trans[:,0] #change here

# %%
df_final

# %%
df_final[['App_Pred','Y']].plot()

# %%
df_final[['App_Pred','high','low','volume','Y']].plot()

# %% [markdown]
# // 1ST MODEL WTIH SENTIMENTS MODEL 2

# %%
# for polarity
dfvad = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/dayvad.csv')
dfvad['date']=pd.to_datetime(dfvad.date)
dfvad=dfvad.set_index(pd.DatetimeIndex(dfvad['date']))

dfvad=dfvad['compound']
dfvad

# %%
df1 = pd.concat([df, dfvad], axis=1)#, join='outer')
df1
```

```
# %%
df1.head(30)

# %%
df_input1 = df1 [['volume','high','low','compound','Y']] #add polarity in
here when added

# %%
df_input1

# %%
df1.plot(subplots=True)

# %%
df_input1.describe()

# %%
df_input1 = df_input1.dropna()

# %%
df_input1

# %%
data_scaled = scaler.fit_transform(df_input1)
data_scaled

# %%
data_scaled[:,0:] #change here

# %%
features=data_scaled[:,0:4]  #change heere
target=data_scaled[:,4]

# %%
TimeseriesGenerator(features, target, length=2, sampling_rate=1,
batch_size=1)[0]

# %%
x_train, x_test, y_train, y_test = train_test_split(features, target,
test_size=0.30, random_state=42, shuffle = False)

# %%
win_length=5
batch_size=13
num_features=4 #this number will change when polarity is added
train_generator = TimeseriesGenerator(x_train, y_train, length=win_length,
sampling_rate=1, batch_size=batch_size)
test_generator = TimeseriesGenerator(x_test, y_test, length=win_length,
sampling_rate=1, batch_size=batch_size)
```

```
# %% [markdown]
# model creation vader

# %%
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape= (win_length, num_features),
return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.LSTM(128, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(64, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(128,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(32,
kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))

# %%
model.summary()

# %%
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                  patience=2,
                                                 mode= 'min')
model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit_generator(train_generator, epochs=50,
                    validation_data=test_generator,
                    shuffle=False,
                    callbacks=[early_stopping], verbose=1)

# %%
model.evaluate_generator(test_generator, verbose=0)

# %%
predictions=model.predict_generator(test_generator)

# %%
predictions

# %% [markdown]
# accuracy for vader model
```

```
# %%
import numpy as np

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mae2 = mean_absolute_percentage_error(y_test, predictions)
acc2 = 100 - mae2
acc2

# %%
y_test

# %%
x_test

# %%
x_test[:,0:][win_length:]

# %%
df_pred=pd.concat([pd.DataFrame(predictions),
pd.DataFrame(x_test[:,0:4][win_length:])],axis=1) #change here

# %%
df_pred

# %%
rev_trans=scaler.inverse_transform(df_pred)
rev_trans

# %%
df_final=df_input1[predictions.shape[0]*-1:]

# %%
df_final.count()

# %% [markdown]
# App pred vs Y

# %%
df_final['App_Pred']=rev_trans[:,0] #change here
df_final

# %%
df_final[['App_Pred','Y']].plot()

# %%
df_final[['App_Pred','high','low','volume','Y']].plot()
```

```
# %% [markdown]
# WIth flair 2nd Sentiments MOdel // MODEL 3

# %%
# for polarity
dfvad = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/dayblob.csv')
dfvad['date']=pd.to_datetime(dfvad.date)
dfvad=dfvad.set_index(pd.DatetimeIndex(dfvad['date']))

dfblob=dfvad['Polarity']
dfblob

# %%
df1 = pd.concat([df, dfblob], axis=1)#, join='outer')
df1
df1.head(30)

# %%
df_input1 = df1 [['volume','high','low','Polarity','Y']] #add polarity in
here when added
df_input1

# %%
df1.plot(subplots=True)

# %%
df_input1.describe()

# %%
df_input1 = df_input1.dropna()
df_input1

# %%
data_scaled = scaler.fit_transform(df_input1)
data_scaled

# %%
data_scaled[:,0:] #change here

# %%
features=data_scaled[:,0:4]  #change heere
target=data_scaled[:,4]

# %%
TimeseriesGenerator(features, target, length=2, sampling_rate=1,
batch_size=1)[0]

# %%
```

```
x_train, x_test, y_train, y_test = train_test_split(features, target,
test_size=0.30, random_state=42, shuffle = False)

# %%
win_length=5
batch_size=13
num_features=4 #this number will change when polarity is added
train_generator = TimeseriesGenerator(x_train, y_train, length=win_length,
sampling_rate=1, batch_size=batch_size)
test_generator = TimeseriesGenerator(x_test, y_test, length=win_length,
sampling_rate=1, batch_size=batch_size)

# %% [markdown]
# model creation textblob

# %%
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape= (win_length, num_features),
return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.LSTM(128, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(64, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(128,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(32,
kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))

# %%
model.summary()

# %%
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                  patience=2,
                                                 mode= 'min')
model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit_generator(train_generator, epochs=50,
                    validation_data=test_generator,
                    shuffle=False,
                    callbacks=[early_stopping], verbose=1)

# %%
```

```
model.evaluate_generator(test_generator, verbose=0)

# %%
predictions=model.predict_generator(test_generator)
predictions

# %% [markdown]
# accuracy for textblob

# %%
import numpy as np

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mae3 = mean_absolute_percentage_error(y_test, predictions)
acc3 = 100 - mae3
acc3

# %%
y_test

# %%
x_test

# %%
x_test[:,0:][win_length:]

# %%
df_pred=pd.concat([pd.DataFrame(predictions),
pd.DataFrame(x_test[:,0:4][win_length:])],axis=1) #change here
df_pred

# %%
rev_trans=scaler.inverse_transform(df_pred)
rev_trans

# %%
df_final=df_input1[predictions.shape[0]*-1:]
df_final.count()

# %% [markdown]
# App pred vs Y

# %%
df_final['App_Pred']=rev_trans[:,0] #change here
df_final

# %%
```

```
df_final[['App_Pred','Y']].plot()

# %% [markdown]
# final plot

# %%
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
plt.figure(figsize=(10,10))
mae = [acc1, acc2, acc3]
plt.title('Accuracy based on Mean Absolute Percentage Error for Regression
Model')
x= ['without sentiment', 'vader','blob']
plt.xlabel('DIFFERENT MODELS')
plt.ylabel('ACCURACY')
x_pos = [i for i, _ in enumerate(x)]
plt.bar(x_pos, mae, color='teal')
plt.xticks(x_pos, x)
plt.show()

# %%
print('Error margin without sentiments:',mae1)
print('Error margin Vader Model:       ',mae2)
print('Error margin Textblob model:    ',mae3)

print('Accuracy without sentiments:    ',acc1)
print('Accuracy Vader Model:           ',acc2)
print('Accuracy Textblob model:        ',acc3)
```

6.9 Model 2 Multivariable Regression

```
# %% [markdown]
# // model without sentiments MODEL 1

# %%
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import TimeseriesGenerator
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import tensorflow as tf
mpl.rcParams['figure.figsize'] = (10, 8)
mpl.rcParams['axes.grid'] = False
```

```
import numpy as np
import pandas as pd

df=pd.read_csv('C:/Users/parek/OneDrive - University of East London/Crypto
Prediction with machine learning and sentiment
analysis/Dataset/btc_valueextra.csv')
#df = pd.read_csv('/Users/rohanparekh/UEL/UEL Research/btc_value.csv')

df['dt']=pd.to_datetime(df.dt)
df=df.set_index(pd.DatetimeIndex(df['dt']))

# imported vader sentiment values into main dataframe
dfvad = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/dayvad.csv')
dfvad = dfvad.rename(columns={'compound':'sentiment_mag'})

dfvad['date']=pd.to_datetime(dfvad.date)
dfvad=dfvad.set_index(pd.DatetimeIndex(dfvad['date']))

dfvad=dfvad['sentiment_mag']

df = pd.concat([df, dfvad], axis=1, join='outer')
#print (df.head())
# df['Trend'] = df['close'] - df['open']

# imported textblob sentiments into main dataframe
dfblob = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/dayblob.csv')
dfblob = dfblob.rename(columns={'Polarity':'sentiment_blob'})

dfblob['date']=pd.to_datetime(dfblob.date)
dfblob=dfblob.set_index(pd.DatetimeIndex(dfblob['date']))

dfblob=dfblob['sentiment_blob']

df = pd.concat([df, dfblob], axis=1, join='outer')

df = df[[
'close','open','volume','high','low','sentiment_mag','sentiment_blob','Trend
','Future Trend 1','Future 2']]
df = df.dropna()

df2 = df[[ 'volume','high','low','sentiment_mag','Trend','Future Trend 1']]
df3 = df[[ 'volume','high','low','sentiment_blob','Trend','Future Trend 1']]


df1 = df[['volume','high','low','Trend','Future Trend 1']]
pd.set_option('display.max_rows', 10)
```

```
df1


# %%
pd.set_option('display.max_rows', 10)

# %%
df.info()

# %%
df_input = df1 [['volume','high','low','Trend']] #add polarity in here when
added

# %%
df.plot(subplots=True)

# %%
df_input.describe()

# %%
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(df_input)

# %%
data_scaled

# %%
data_scaled[:,0:] #change here

# %%
features=data_scaled[:,0:3]  #change heere
target=data_scaled[:,3]

# %%
TimeseriesGenerator(features, target, length=2, sampling_rate=1,
batch_size=1)[0]

# %%
x_train, x_test, y_train, y_test = train_test_split(features, target,
test_size=0.30, random_state=42, shuffle = False)

# %%
x_train.shape

# %%
x_test.shape

# %%
win_length=5
batch_size=13
```

```
num_features=3 #this number will change when polarity is added
train_generator = TimeseriesGenerator(x_train, y_train, length=win_length,
sampling_rate=1, batch_size=batch_size)
test_generator = TimeseriesGenerator(x_test, y_test, length=win_length,
sampling_rate=1, batch_size=batch_size)



# %%
train_generator[0]

# %%
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape= (win_length, num_features),
return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.LSTM(128, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(64, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(128,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(32,
kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))

# %%
model.summary()

# %%
early_stopping =
tf.keras.callbacks.EarlyStopping(monitor='val_loss',   patience=5,  mode=
'min')

model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit_generator(train_generator, epochs=100,
                    validation_data=test_generator,
                    shuffle=False,
                    callbacks=[early_stopping], verbose=1)



# %%
model.evaluate_generator(test_generator, verbose=0)
```

```
# %%
predictions=model.predict_generator(test_generator)

# %%
predictions.shape[0]

# %%
predictions

# %%
import numpy as np

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mae1 = mean_absolute_percentage_error(y_test, predictions)
print("Mean absolute percentage error: ", mae1)
acc1 = 100-mae1
print ("Accuracy based on mean absolute error: ", mae1)

# %%
y_test

# %%
x_test

# %%
x_test[:,0:][win_length:]

# %%
df_pred=pd.concat([pd.DataFrame(predictions),
pd.DataFrame(x_test[:,0:3][win_length:])],axis=1) #change here

# %%
df_pred

# %%
rev_trans=scaler.inverse_transform(df_pred)

# %%
rev_trans

# %%
df_final=df_input[predictions.shape[0]*-1:]

# %%
df_final.count()
```

```
# %%
df_final['App_Pred']=rev_trans[:,0] #change here

# %%
df_final

# %% [markdown]
# App pred vs Trend

# %%
df_final[['App_Pred','Trend']].plot()

# %%
df_final[['App_Pred','high','low','volume','Trend']].plot()

# %% [markdown]
# // 1ST MODEL WTIH SENTIMENTS MODEL 2

# %%
df_input1 = df2 [['volume','high','low','sentiment_mag','Trend']] #add
polarity in here when added

# %%
df_input1

# %%
df1.plot(subplots=True)

# %%
df_input1.describe()

# %%
df_input1 = df_input1.dropna()

# %%
df_input1

# %%
df_input1['sentiment']=df_input1['sentiment_mag'].astype(int)
df_input1.loc[df_input1.sentiment_mag>0,'sentiment']=1
df_input1.loc[df_input1.sentiment_mag==0,'sentiment']=0
df_input1.loc[df_input1.sentiment_mag<0,'sentiment']=0

# %%
df_input1 =
df_input1[['volume','high','low','sentiment_mag','sentiment','Trend' ]]
df_input1

# %% [markdown]
# Vader data scaling
```

```
# %%
data_scaled = scaler.fit_transform(df_input1)
data_scaled

# %%
data_scaled[:,0:] #change here

# %%
features=data_scaled[:,0:5]  #change heere
target=data_scaled[:,5]

# %%
TimeseriesGenerator(features, target, length=2, sampling_rate=1,
batch_size=1)[0]

# %% [markdown]
# Vader splitting train and test

# %%
x_train, x_test, y_train, y_test = train_test_split(features, target,
test_size=0.30, random_state=42, shuffle = False)

# %%
win_length=5
batch_size=13
num_features=5 #this number will change when polarity is added
train_generator = TimeseriesGenerator(x_train, y_train, length=win_length,
sampling_rate=1, batch_size=batch_size)
test_generator = TimeseriesGenerator(x_test, y_test, length=win_length,
sampling_rate=1, batch_size=batch_size)

# %% [markdown]
# Vader Model creation

# %%
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape= (win_length, num_features),
return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.LSTM(128, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(64, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(128,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64,
kernel_initializer='HeUniform',activation='relu'))
```

```
model.add(tf.keras.layers.Dense(32,
kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))

# %%
model.summary()

# %%
early_stopping =
tf.keras.callbacks.EarlyStopping(monitor='val_loss',  patience=5, mode=
'min')

model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit_generator(train_generator, epochs=100,
                    validation_data=test_generator,
                    shuffle=False,
                    callbacks=[early_stopping], verbose=1)

# %%
model.evaluate_generator(test_generator, verbose=0)

# %%
predictions=model.predict_generator(test_generator)

# %%
predictions

# %% [markdown]
# Vader Accuracy Calculation

# %%
import numpy as np

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mae2 = mean_absolute_percentage_error(y_test, predictions)
acc2 = 100 - mae2
acc2

# %%
y_test

# %%
x_test
```

```
# %%
x_test[:,0:][win_length:]

# %%
df_pred=pd.concat([pd.DataFrame(predictions),
pd.DataFrame(x_test[:,0:5][win_length:])],axis=1) #change here

# %%
df_pred

# %%
rev_trans=scaler.inverse_transform(df_pred)
rev_trans

# %%
df_final=df_input1[predictions.shape[0]*-1:]

# %%
df_final.count()

# %%
df_final['App_Pred']=rev_trans[:,0] #change here
df_final

# %% [markdown]
# Vader prediction plot

# %% [markdown]
# App pred vs Trend

# %%
df_final[['App_Pred','Trend']].plot()

# %%
df_final[['App_Pred','high','low','volume','sentiment','sentiment_mag','Tren
d']].plot()

# %% [markdown]
# WIth Textblob 2nd Sentiments MOdel // MODEL 3

# %%
df_input1 = df3 [['volume','high','low','sentiment_blob','Trend']] #add
polarity in here when added
df_input1

# %%
df1.plot(subplots=True)

# %%
df_input1.describe()
```

```
# %%
df_input1 = df_input1.dropna()
df_input1

# %%
df_input1['sentiment']=df_input1['sentiment_blob'].astype(int)
df_input1.loc[df_input1.sentiment_blob>0,'sentiment']=1
df_input1.loc[df_input1.sentiment_blob==0,'sentiment']=0
df_input1.loc[df_input1.sentiment_blob<0,'sentiment']=0

# %%
df_input1 = df_input1
[['volume','high','low','sentiment_blob','sentiment','Trend']] #add polarity
in here when added
df_input1

# %%
from sklearn.preprocessing import MinMaxScaler, StandardScaler
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(df_input1)
data_scaled

# %%
data_scaled[:,0:] #change here

# %%
features=data_scaled[:,0:5]  #change heere
target=data_scaled[:,5]

# %%
TimeseriesGenerator(features, target, length=2, sampling_rate=1,
batch_size=1)[0]

# %% [markdown]
# blob splitting train and test

# %%
x_train, x_test, y_train, y_test = train_test_split(features, target,
test_size=0.30, random_state=42, shuffle = False)

# %%
win_length=5
batch_size=13
num_features=5 #this number will change when polarity is added
train_generator = TimeseriesGenerator(x_train, y_train, length=win_length,
sampling_rate=1, batch_size=batch_size)
test_generator = TimeseriesGenerator(x_test, y_test, length=win_length,
sampling_rate=1, batch_size=batch_size)
```

```
# %% [markdown]
# Blob model Creation

# %%
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape= (win_length, num_features),
return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.LSTM(128, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.5))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(64, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(128,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(32,
kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# %%
model.summary()

# %%
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=5, mode= 'min')
model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit_generator(train_generator, epochs=50,
                    validation_data=test_generator,
                    shuffle=False,
                    callbacks=[early_stopping], verbose=1)

# %%
model.evaluate_generator(test_generator, verbose=0)

# %%
predictions=model.predict_generator(test_generator)
predictions

# %% [markdown]
# Blob accuracy calculation

# %%
import numpy as np

def mean_absolute_percentage_error(y_true, y_pred):
```

```
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mae3 = mean_absolute_percentage_error(y_test, predictions)
acc3 = 100 - mae3
acc3

# %%
y_test

# %%
x_test

# %%
x_test[:,0:][win_length:]

# %%
df_pred=pd.concat([pd.DataFrame(predictions),
pd.DataFrame(x_test[:,0:5][win_length:])],axis=1) #change here
df_pred

# %%
rev_trans=scaler.inverse_transform(df_pred)
rev_trans

# %%
df_final=df_input1[predictions.shape[0]*-1:]
df_final.count()

# %%
df_final['App_Pred']=rev_trans[:,0] #change here
df_final

# %% [markdown]
# blob prediction plot

# %% [markdown]
# App pred vs Trend

# %%
df_final[['App_Pred','Trend']].plot()

# %% [markdown]
# Final comparison accuracy plot

# %%
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
plt.figure(figsize=(7,10))
```

```
mae = [acc1, acc2, acc3]
plt.title('Accuracy based on Mean Absolute Percentage Error for Regression
Model')
x= ['without sentiment', 'vader','blob']
plt.xlabel('DIFFERENT MODELS')
plt.ylabel('ACCURACY')
x_pos = [i for i, _ in enumerate(x)]
plt.bar(x_pos, mae, color='teal')
plt.xticks(x_pos, x)
plt.show()

# %%
print('Error margin without sentiments:',mae1)
print('Error margin Vader Model:       ',mae2)
print('Error margin Textblob model:    ',mae3)

print('Accuracy without sentiments:    ',acc1)
print('Accuracy Vader Model:           ',acc2)
print('Accuracy Textblob model:        ',acc3)
```

6.10 Model 3 Binary Classification

```
# %%
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import TimeseriesGenerator
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import tensorflow as tf
mpl.rcParams['figure.figsize'] = (10, 8)
mpl.rcParams['axes.grid'] = False

import numpy as np
import pandas as pd

df=pd.read_csv('C:/Users/parek/OneDrive - University of East London/Crypto
Prediction with machine learning and sentiment
analysis/Dataset/btc_value.csv')
#df = pd.read_csv('/Users/rohanparekh/UEL/UEL Research/btc_value.csv')

df['dt']=pd.to_datetime(df.dt)
df=df.set_index(pd.DatetimeIndex(df['dt']))
#
#df.iloc[:,1]
df['Y'] = df['close'] - df['open']
```

```
df = df[['volume','high','low','Y']]


# %%
df['Y']=df['Y']
df.loc[df['Y']>0,'Y']=1
df.loc[df['Y']<=0,'Y']=0

# %%
df

# %%
df.plot(subplots = True)

# %% [markdown]
# initial model with sequential

# %%
X=df.iloc[:,0:3].values  #change heere
y=df.iloc[:,3].values

# %%
X

# %%
y

# %% [markdown]
# Splitting train and test

# %%
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,
random_state = 0)

# %%
#from sklearn import preprocessing
#scaler=preprocessing.StandardScaler()
#scaler.fit(X_train)

# %%
#X_train_scaled=scaler.transform(X_train)
#X_test_scaled= scaler.transform(X_test)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

X_train_scaled=(X_train)
```

```
X_test_scaled= (X_test)

# %% [markdown]
# Model for Prediction

# %%
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense , LeakyReLU
model=Sequential()
model.add(Dense(16, kernel_initializer='HeUniform', activation='relu'))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(Dense(16, kernel_initializer='HeUniform', activation='relu'))
model.add(Dense(32, kernel_initializer='HeUniform', activation='relu'))


model.add(Dense (1, activation='sigmoid'))
model.compile(optimizer='SGD',loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X_train_scaled,y_train,epochs=250,verbose=0)
J_list = model.history.history['loss']
plt.plot (J_list)


# %%
J_train=model.evaluate (X_train_scaled,y_train)
J_test = model.evaluate (X_test_scaled, y_test)
print (J_train,J_test)


# %%
from sklearn.metrics import classification_report,confusion_matrix,
accuracy_score

predictions = model.predict(X_test_scaled).round()

Y_hat_test =  model.predict (X_test_scaled)

score = accuracy_score(y_test, predictions)
print('accuracy = ', score)
print('Accuracy percent: ', (score*100),'%')

#print (classification_report(y_test,predictions))

# %%
#predictions =  model.predict(X_train_scaled) # predict_class is depricated
#Y_hat_train =  model.predict(X_train_scaled)
#print (classification_report(y_train,predictions))

# %% [markdown]
# Prediction Plot
```

```
# %%
plt.plot(predictions.round())
plt.plot(y_test)
plt.legend('pt')

# %% [markdown]
# Confusion Matrix

# %%
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, predictions.round())
import seaborn as sns

ax = sns.heatmap(conf_matrix, annot=True, cmap='Blues')

ax.set_title('Seaborn Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()

# %% [markdown]
# Creating model with sentiment analysis

# %%
# for polarity
dfvad = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/dayvad.csv')
dfvad = dfvad.rename(columns={'compound':'sentiment_mag'})
dfvad['date']=pd.to_datetime(dfvad.date)
dfvad=dfvad.set_index(pd.DatetimeIndex(dfvad['date']))

dfvad=dfvad['sentiment_mag']
dfvad

# %%
df1 = pd.concat([df, dfvad], axis=1)#, join='outer')
df1

# %%
df1.plot(subplots=True)

# %%
```

```
df1 = df1.dropna()

# %%
df1['sentiment']=df1['sentiment_mag'].astype(int)
df1.loc[df1.sentiment_mag>0,'sentiment']=1
df1.loc[df1.sentiment_mag==0,'sentiment']=0
df1.loc[df1.sentiment_mag<0,'sentiment']=0

df1

# %%
df1 = df1 [['volume','high','low','sentiment_mag','Y']] #add polarity in
here when added

# %%
X=df1.iloc[:,0:4].values  #change heere
y=df1.iloc[:,4].values

# %% [markdown]
# Vader splitting train and test

# %%
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,
random_state = 0)

# %%
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

# %% [markdown]
# Vader Model creation

# %%
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense , LeakyReLU
model=Sequential()
model.add(Dense(16, kernel_initializer='HeUniform', activation='relu'))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(Dense(16, kernel_initializer='HeUniform', activation='relu'))
model.add(Dense(32, kernel_initializer='HeUniform', activation='relu'))

model.add(Dense (1, activation='sigmoid'))
model.compile(optimizer='SGD',loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X_train,y_train,epochs=250,verbose=0)
J_list = model.history.history['loss']
plt.plot (J_list)
```

```
J_list = model.history.history['loss']
plt.plot (J_list)

# %%
model.summary()

# %%
y_pred = model.predict(X_test)

# %%
y_pred

# %%
y_test

# %%
from sklearn.metrics import confusion_matrix, accuracy_score
conf_matrix2 = confusion_matrix(y_test, y_pred.round())
score2 = accuracy_score(y_test, y_pred.round())
print('accuracy = ', (score2*100),'%')

# %% [markdown]
# vader Confusion Matrix

# %%
import seaborn as sns

ax = sns.heatmap(conf_matrix2, annot=True, cmap='Blues')

ax.set_title('Seaborn Confusion Matrix with labels\n\n')
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ')

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()

# %%
plt.plot(y_pred.round())
plt.plot(y_test)
plt.xlabel("Time")
plt.ylabel("binary value")
plt.legend("pt")

# %% [markdown]
# adding blob sentiment analysis
```

```
# %%
# for polarity
dfvad = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/dayblob.csv')
dfvad = dfvad.rename(columns={'Polarity':'sentiment_mag'})
dfvad['date']=pd.to_datetime(dfvad.date)
dfvad=dfvad.set_index(pd.DatetimeIndex(dfvad['date']))

dfvad=dfvad['sentiment_mag']
dfvad

# %%
df1 = pd.concat([df, dfvad], axis=1)#, join='outer')
df1

# %%
df1.plot(subplots=True)

# %%
df1 = df1.dropna()

# %%
df1['sentiment']=df1['sentiment_mag'].astype(int)
df1.loc[df1.sentiment_mag>0,'sentiment']=1
df1.loc[df1.sentiment_mag==0,'sentiment']=0
df1.loc[df1.sentiment_mag<0,'sentiment']=0

df1

# %%
df1 = df1 [['volume','high','low','sentiment_mag','Y']] #add polarity in
here when added

# %%
X=df1.iloc[:,0:4].values  #change heere
y=df1.iloc[:,4].values

# %%
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,
random_state = 0)

# %%
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

# %% [markdown]
```

```
# blob model

# %%
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense , LeakyReLU
model=Sequential()
model.add(Dense(16, kernel_initializer='HeUniform', activation='relu'))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(Dense(16, kernel_initializer='HeUniform', activation='relu'))
model.add(Dense(32, kernel_initializer='HeUniform', activation='relu'))

model.add(Dense (1, activation='sigmoid'))
model.compile(optimizer='SGD',loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(X_train,y_train,epochs=250,verbose=0)
J_list = model.history.history['loss']
plt.plot (J_list)
J_list = model.history.history['loss']
plt.plot (J_list)

# %%
model.summary()

# %%
y_pred = model.predict(X_test)

# %%
y_pred

# %%
y_test

# %%
from sklearn.metrics import confusion_matrix, accuracy_score
conf_matrix2 = confusion_matrix(y_test, y_pred.round())
score3 = accuracy_score(y_test, y_pred.round())
print('accuracy = ', (score2*100),'%')

# %% [markdown]
# Blob Confusion Matrix

# %%
import seaborn as sns

ax = sns.heatmap(conf_matrix2, annot=True, cmap='Blues')

ax.set_title('Seaborn Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');
```

```
## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])

## Display the visualization of the Confusion Matrix.
plt.show()

# %%
plt.plot(y_pred.round())
plt.plot(y_test)
plt.xlabel("Time")
plt.ylabel("binary value")
plt.legend("pt")

# %%
print('Accuracy without sentiment: ', (score*100),'%')
print('Accuracy with vader: ', (score2*100),'%')
print('Accuracy with blob: ', (score3*100),'%')

# %% [markdown]
# Final accuracy plot

# %%
plt.title('Binary Classification Model')
plt.bar(['without sentiment','vader','blob'], [(score*100), (score2*100),
(score3*100)])
plt.xlabel('DIFFERENT MODELS')
plt.ylabel('ACCURACY')
plt.grid(color='white', linestyle='dashed')
plt.show()
```

6.11 Model 4 Future Trends Regression

```
# %% [markdown]
# Initalisation and import all the required file

# %%
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import TimeseriesGenerator
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import tensorflow as tf
mpl.rcParams['figure.figsize'] = (10, 8)
mpl.rcParams['axes.grid'] = False
```

```
import numpy as np
import pandas as pd

df=pd.read_csv('C:/Users/parek/OneDrive - University of East London/Crypto
Prediction with machine learning and sentiment
analysis/Dataset/btc_valueextra.csv')
#df = pd.read_csv('/Users/rohanparekh/UEL/UEL Research/btc_value.csv')

df['dt']=pd.to_datetime(df.dt)
df=df.set_index(pd.DatetimeIndex(df['dt']))

# imported vader sentiment values into main dataframe
dfvad = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/dayvad.csv')
dfvad = dfvad.rename(columns={'compound':'sentiment_mag'})

dfvad['date']=pd.to_datetime(dfvad.date)
dfvad=dfvad.set_index(pd.DatetimeIndex(dfvad['date']))

dfvad=dfvad['sentiment_mag']

df = pd.concat([df, dfvad], axis=1, join='outer')
#print (df.head())
# df['Trend'] = df['close'] - df['open']

# imported textblob sentiments into main dataframe
dfblob = pd.read_csv('C:/Users/parek/OneDrive - University of East
London/Crypto Prediction with machine learning and sentiment
analysis/Dataset/dayblob.csv')
dfblob = dfblob.rename(columns={'Polarity':'sentiment_blob'})

dfblob['date']=pd.to_datetime(dfblob.date)
dfblob=dfblob.set_index(pd.DatetimeIndex(dfblob['date']))

dfblob=dfblob['sentiment_blob']

df = pd.concat([df, dfblob], axis=1, join='outer')

df = df[[
'close','open','volume','high','low','sentiment_mag','sentiment_blob','Trend
','Future Trend 1','Future 2']]
df = df.dropna()
df = df.iloc[:-1,:]

df2 = df[[ 'volume','high','low','sentiment_mag','Trend','Future Trend 1']]
df3 = df[[ 'volume','high','low','sentiment_blob','Trend','Future Trend 1']]
df
```

```
df1 = df[['volume','high','low','Trend','Future Trend 1']]
df1

# %%
pd.set_option('display.max_rows', 10)

# %%
df1= df1.iloc[:-1,:] # to remove the last row from the dataframe as it has
an impact on the prediction
df1

# %%
df1.info()

# %%
df1.plot(subplots=True)

# %%
print(df1.describe())

# %%
print(df1.iloc[:,0:4])
print(df1.iloc[:,4])

# %% [markdown]
# Scaling the dataset

# %%
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(df1)

# %%
data_scaled[:,0:] #change here

# %%
features=data_scaled[:,0:4]  #change heere
target=data_scaled[:,4]

# %%
TimeseriesGenerator(features, target, length=2, sampling_rate=1,
batch_size=1)[0]

# %% [markdown]
# Splitting in train and test dataset

# %%
x_train, x_test, y_train, y_test = train_test_split(features, target,
test_size=0.30, random_state=42, shuffle = False)

# %%
```

```
x_train.shape

# %%
x_test.shape

# %%
win_length=5
batch_size=13
num_features=4 # this number will change when polarity is added
train_generator = TimeseriesGenerator(x_train, y_train, length=win_length,
sampling_rate=1, batch_size=batch_size)
test_generator = TimeseriesGenerator(x_test, y_test, length=win_length,
sampling_rate=1, batch_size=batch_size)

# %%
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(8, input_shape= (win_length, num_features),
return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(tf.keras.layers.LSTM(16, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(16, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(23,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(20,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64,
kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))

model.summary()

# %%
early_stopping =
tf.keras.callbacks.EarlyStopping(monitor='val_loss',  patience=2,  mode=
'min')
model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit_generator(train_generator, epochs=50,
                    validation_data=test_generator,
                    shuffle=False,
                     verbose=0, callbacks=[early_stopping])

# %% [markdown]
# Prediction the targeted values
```

```
# %%
predictions=model.predict_generator(test_generator)

# %%
y_test

# %%
y_test[5:]

# %% [markdown]
# Checking the accuracy of the predicted values

# %%
yt3=y_test[5:]
from sklearn.metrics import mean_squared_error

rmse = mean_squared_error(yt3, predictions)
print ('RMSE: ', rmse)

# %%
import numpy as np

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mae1 = mean_absolute_percentage_error(y_test, predictions)
print("Mean absolute error margin: ", mae1)
acc1 = 100-mae1
print ("Accuracy based on mean absolute error: ", acc1, "%")

# %%
plt.plot(predictions)
plt.plot(y_test)
plt.legend(['Predicted', 'Actual'])
plt.show()

# %%
plt.bar(['accuracy'] ,mae1)
plt.show()

# %%
df_pred=pd.concat([pd.DataFrame(predictions),
pd.DataFrame(x_test[:,0:4][win_length:])],axis=1) # change here

# %%
rev_trans=scaler.inverse_transform(df_pred)

# %%
df_final=df1[predictions.shape[0]*-1:]
```

```
# %%
df_final['Predictions']=rev_trans[:,0] #change here
df_final

# %%
mean_absolute_percentage_error(df_final['Future Trend 1'],
df_final['Predictions'])

# %%
df_final[['Predictions','Future Trend 1']].plot()

# %% [markdown]
# Import Sentiment File

# %%
df2 = df[[ 'close','open','volume','high','low','sentiment_mag','Future
Trend 1']]

# %%
df2.plot(subplots=True)

# %%
df2 = df2.dropna()

# %%
df2 = df[['volume','high','low','sentiment_mag','Trend', 'Future Trend 1' ]]

# Future trend 1 = today's close - tomorrow's close
df2

# %%
# df2 = df2[['volume','high','low','sentiment_mag', 'Trend', 'Future Trend
1' ]]

# %% [markdown]
# Scaling for sentiment dataset

# %%
data_scaled2 = scaler.fit_transform(df2)
data_scaled2

# %%
features2=data_scaled2[:,0:5]  #change here
target2=data_scaled2[:,5]

# %%
TimeseriesGenerator(features2, target2, length=2, sampling_rate=1,
batch_size=1)[0]
```

```
# %% [markdown]
# Splitting train and test for sentiments

# %%
x_train2, x_test2, y_train2, y_test2 = train_test_split(features2, target2,
test_size=0.30, random_state=42, shuffle = False)

# %%
x_train2.shape

# %%
x_test2.shape

# %%
win_length=4
batch_size=13
num_features=5 #this number will change when polarity is added
train_generator2 = TimeseriesGenerator(x_train2, y_train2,
length=win_length, sampling_rate=1, batch_size=batch_size)
test_generator2 = TimeseriesGenerator(x_test2, y_test2, length=win_length,
sampling_rate=1, batch_size=batch_size)

# %%
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(8, input_shape= (win_length, num_features),
return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(tf.keras.layers.LSTM(16, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(16, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(23,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(20,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64,
kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))

model.summary()

# %%
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=2, mode= 'min')
model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit(train_generator2, epochs=100,
```

```
                    validation_data=test_generator2,
                    shuffle=False,
                    verbose=0 , callbacks=[early_stopping])

# %% [markdown]
# Predicting values for sentiment model

# %%
predictions2=model.predict_generator(test_generator2)

# %%
yt4=y_test2[4:]
from sklearn.metrics import mean_squared_error

rmse2 = mean_squared_error(yt4, predictions2)
print ('RMSE: ', rmse2*100)

# %%
import numpy as np

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mae2 = mean_absolute_percentage_error(y_test2, predictions2)
print("Mean absolute error margin: ", mae2)
acc2 = 100-mae2
print ("Accuracy based on mean absolute error: ", acc2, "%")

# %%
plt.plot(predictions2)
plt.plot(y_test2)
plt.legend(['Predicted', 'Actual'])
plt.show()

# %% [markdown]
# Final plot graph

# %%
plt.bar(['without sentiments', 'with sentiments'] ,[mae1, mae2])
plt.xlabel('DIFFERENT MODELS')
plt.ylabel('ACCURACY')
plt.grid(color='white', linestyle='dashed')
plt.show()

# %%
df_pred2=pd.concat([pd.DataFrame(predictions2),
pd.DataFrame(x_test2[:,0:5][win_length:])],axis=1) # change here

# %%
```

```
rev_trans2=scaler.inverse_transform(df_pred2)

# %%
df_final2=df2[predictions2.shape[0]*-1:]

# %%
df_final2['Predictions']=rev_trans2[:,0] #change here
df_final2

# %%


# %%
mean_absolute_percentage_error(df_final2['Future Trend 1'],
df_final2['Predictions'])

# %%
df_final2[['Predictions','Future Trend 1']].plot()

# %% [markdown]
# Creating textblob model

# %%
df3.plot(subplots=True)

# %% [markdown]
# Scaling for sentiment dataset

# %%
data_scaled2 = scaler.fit_transform(df3)
data_scaled2

# %%
features2=data_scaled2[:,0:5]  #change here
target2=data_scaled2[:,5]

# %%
TimeseriesGenerator(features2, target2, length=2, sampling_rate=1,
batch_size=1)[0]

# %% [markdown]
# Splitting train and test for sentiments

# %%
x_train2, x_test2, y_train2, y_test2 = train_test_split(features2, target2,
test_size=0.30, random_state=42, shuffle = False)

# %%
x_train2.shape
```

```
# %%
x_test2.shape

# %%
win_length=4
batch_size=13
num_features=5 #this number will change when polarity is added
train_generator2 = TimeseriesGenerator(x_train2, y_train2,
length=win_length, sampling_rate=1, batch_size=batch_size)
test_generator2 = TimeseriesGenerator(x_test2, y_test2, length=win_length,
sampling_rate=1, batch_size=batch_size)

# %%
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(8, input_shape= (win_length, num_features),
return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(tf.keras.layers.LSTM(16, return_sequences=True))
model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.LSTM(16, return_sequences=False))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(23,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(20,
kernel_initializer='HeUniform',activation='relu'))
model.add(tf.keras.layers.Dense(64,
kernel_initializer='HeUniform',activation='relu'))

model.add(tf.keras.layers.Dense(1))

model.summary()

# %%
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=2, mode= 'min')
model.compile(loss=tf.losses.MeanSquaredError(),
              optimizer=tf.optimizers.Adam(),
              metrics=[tf.metrics.MeanAbsoluteError()])
history = model.fit(train_generator2, epochs=100,
                    validation_data=test_generator2,
                    shuffle=False,
                    verbose=0, callbacks=[early_stopping])

# %% [markdown]
# Predicting values for sentiment model

# %%
predictions2=model.predict_generator(test_generator2)
```

```
# %%
yt4=y_test2[4:]
from sklearn.metrics import mean_squared_error

rmse2 = mean_squared_error(yt4, predictions2)
print ('RMSE: ', rmse2*100)

# %%
import numpy as np

def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mae3 = mean_absolute_percentage_error(y_test2, predictions2)
print("Mean absolute error margin: ", mae3)
acc3 = 100-mae3
print ("Accuracy based on mean absolute error: ", acc3, "%")

# %%
plt.plot(predictions2)
plt.plot(y_test2)
plt.legend(['Predicted', 'Actual'])
plt.show()

# %% [markdown]
# Final plot graph

# %%
plt.bar(['without sentiments', 'Vader', 'Textblob'] ,[acc1, acc2, acc3])
plt.xlabel('DIFFERENT MODELS')
plt.ylabel('ACCURACY')
plt.grid(color='white', linestyle='dashed')
plt.show()

# %%
print('Error margin without sentiments:',mae1)
print('Error margin Vader Model:       ',mae2)
print('Error margin Textblob model:    ',mae3)

print('Accuracy without sentiments:    ',acc1)
print('Accuracy Vader Model:           ',acc2)
print('Accuracy Textblob model:        ',acc3)

# %%
df_pred2=pd.concat([pd.DataFrame(predictions2),
pd.DataFrame(x_test2[:,0:5][win_length:])],axis=1) # change here

# %%
df_pred2
```

```
# %%
rev_trans2=scaler.inverse_transform(df_pred2)

# %%
df_final2=df3[predictions2.shape[0]*-1:]

# %%
df_final2['Predictions']=rev_trans2[:,0] #change here
df_final2

# %%


# %%
mean_absolute_percentage_error(df_final2['Future Trend 1'],
df_final2['Predictions'])

# %%
df_final2[['Predictions','Future Trend 1']].plot()
```