

Sentiment Analysis & Time Series Analysis on Stock Market

by

Aniket K Singh

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master

of

Computing and Information Systems

Youngstown State University

May 2023

Sentiment Analysis & Time Series Analysis on Stock Market
Aniket K Singh

I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Aniket Student, Student Date

Approvals:

Dr. John R Sullins, Thesis Advisor Date

Dr. Feng George Yu, Committee Member Date

Dr. Alina Lazar, Committee Member Date

Salvatore A. Sanders, PhD, Dean, College of Graduate Studies Date

Abstract

Investors are always looking for ways to make profit in the stock market. Predicting this highly volatile market has been historically challenging. This study explores the use of the social media platform, Twitter, and Machine Learning Algorithm for Time Series Analysis. Our findings suggested that Twitter's data may not be the best for Sentiment Analysis, while other machine learning techniques for Time Series Analysis such as LSTM would be effective. This could potentially help an investor with higher returns.

Keywords: Stock Market, Sentiment Analysis, Opinion Mining, Twitter Sentiment Analysis, Time Series Analysis, Sentiment & Time Series Model

Table of Contents

Chapter I.....	9
1.1 INTRODUCTION.....	9
1.2 Motivation.....	10
1.3 Research Questions	10
1.4 Contributions.....	11
1.5 Organization	11
2 Chapter II.....	12
2.1 LITERATURE REVIEW.....	12
3 Chapter III.....	13
3.1 DATASET.....	13
3.2 Twitter Data Scraping	13
3.3 Yahoo Stock Data.....	14
3.4 Preprocessing.....	15
3.5 Stock Data Preprocessing	15
3.6 Twitter Data.....	15
4 Chapter IV.....	17
4.1 Overview of Natural Language Processing.....	17
4.2 Sentiment Analysis	17
4.3 Sentiment Analysis Pre-Trained Models	18

4.4	Implementation of Twitter Sentiment Analysis Models	19
5	<i>Chapter V</i>	25
5.1	Introduction to Time Series Analysis	25
5.2	Recurrent Neural Networks (RNNs)	25
5.3	Long Short-Term Memory.....	27
5.4	LSTM Models	29
6	<i>Chapter VI</i>	35
6.1	Summary.....	35
6.2	Potential Future Work.....	35
	<i>References</i>	37
7	<i>Appendix: Important Code From The Study</i>	39

List of Figures

<i>Figure 1: RMSE for each model</i>	20
<i>Figure 2: VADER Sentiment Score & Returns</i>	22
<i>Figure 3: Return vs Score plot</i>	23
<i>Figure 4: Simple Neural Network</i>	26
<i>Figure 5: RNNs</i>	27
<i>Figure 6: LSTM Architecture[13]</i>	28
<i>Figure 7: Basic LSTM Model Result</i>	30
<i>Figure 8: Actual vs Predicted with 160 epochs</i>	31
<i>Figure 9: Actual vs Predicted for 25 epochs</i>	33

List of Abbreviations:

NLP	Natural Language Processing
VADER	Valence Aware Dictionary and sEntiment Reasoner
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory Networks
LibSVM	Library for Support Vector Machines
SNS	Social Network Sites
BERT	Bidirectional Encoder Representation from Transformers

ACKNOWLEDGEMENT

I would like to take this opportunity to thank my advisor Dr. John R Sullins for his constant support and motivation. This work would not be possible without him, and his guidance. I would also like to thank my thesis committee members Dr. Alina Lazar and Dr. Feng George Yu; I am very grateful for both of you for becoming a committee member in my thesis.

I would also like to thank my friends and colleagues who kept me motivated to do this study.

1 Chapter I

1.1 INTRODUCTION

Stocks and Stock Market has been around for hundreds of years. Investors, traders, mathematicians, researchers, everyone is attracted by the stock market. Prediction of stock price and trend has been a problem of interest since the stock market has been around. Historically, people have used the massive data that stock market provides to predict the prices, which hasn't been a very successful approach. There are two major types of analysis that has been historically used to predict prices. Fundamental and Technical Analysis, which uses a company's fundamental information such as understanding what company does, how much debt they have, how much cash they have, what's their cash flow, competitions they have, balance sheets and other documents that explains how the company is doing. This type of analysis has been significantly good for a long-term investor, because a stock might be doing well temporarily in the market but might not be a good company to invest, similarly a stock may not do well but the company's fundamentals are good then most likely they would start to see growth. Technical Analysis uses the historical stock data such as volume, charts, indicators, moving averages, trendlines and similar other indicators. This information is mostly used by short term traders. Both methods are not reliable for short-term predictions and has been unsuccessful in many cases. Stock prices are said to be unpredictable because it follows a random walk pattern, and it is deemed to not be predicted with accuracy of 50% or more. [1]

1.2 Motivation

Mathematicians, researchers, investors, and traders have been trying to predict the stock prices for years. I started to learn about stocks and stock market in 2016, right after I completed my High School, and since then I have tried multiple ways to predict the stocks. I have learned both fundamental analysis, and technical analysis, and later after learning Time Series Analysis I learned new methods to predict the prices of stock. Most of the methods were deemed to be unsuccessful but as I was always interested in this market, I started to believe that the stock market can't be predicted without using sentiments. In this thesis, I will be exploring Natural Language Processing techniques to predict the human sentiment, incorporate this with the Time Series Model in hopes of improving the prediction accuracy of the stock prices.

1.3 Research Questions

We are hoping to answer these following questions with this study.

RQ1: Are Twitter posts sentiment scores correlated with stock returns?

RQ2: Using an LSTM model, can we generate profits in Stock Market?

We will explore all these research questions and answer them to conclude this research. The first question here is to understand if at all the sentiment scores that we compute using Natural Language Processing model, is significantly correlated to the price of that stock. The second question is to use see if Time Series models can help to predict stock prices and generate profits in the market.

1.4 Contributions

The objective of this study is to:

- i. To collect sentiment data from Twitter (<https://twitter.com/>) posts for the stocks that we will be studying.
- ii. To analyze the sentiment data and find the correlation score between sentiment score and the stock prices, then determine if the correlation is significant.
- iii. To collect the historical price data and use different time series models to forecast the stock prices with lowest Mean Squared Error.

1.5 Organization

This study has been divided into 6 chapters, second chapter includes literature review on sentiment analysis, chapter three explain the process of data collection and the types of data that are used in the study. Chapter 4 discusses and shows implementation of sentiment model and NLP, followed by Chapter 5 that discusses Neural Networks and Deep Neural Networks like Long Short Term Memory Networks for Time Series Prediction. Chapter 6, the final chapter contains conclusion of the study, including potential future work.

2 Chapter II

2.1 LITERATURE REVIEW

Predicting stock prices/moments has been area of research since the stock market has been around. Many researchers and mathematicians have tried different prediction methods to predict the stock moments. But with Machine Learning becoming more and more popular there are many studies done using these new techniques. There are many blogs and research papers presented in the similar areas. A study presented in [2], also used Sentiment Analysis of Twitter data to predict stock market movements. They used two different methods to analyze tweets, Word2Vec and N-grams. The study employed sentiment analysis and supervised machine learning techniques to find the correlation between stock price and public sentiment expressed on Twitter. They concluded their study with 70.49% accuracy for sentiment analysis using N-grams. They also concluded with 71.82% accuracy with sentiment and stock-price using LibSVM model. Similarly, another extensive study was done by [3], this study also used Twitter for their Sentiment Analysis to predict stock movements. This paper uses Self Organizing Fuzzy Neural Networks (SOFNN) for financial data and obtains 75.56% accuracy on Dow Jones Industrial Index (DJIA). Similarly, there are many other studies has been done using public sentiment to predict the stock movements. This study will reference many other similar studies as we do proceed with this study.

3 Chapter III

3.1 DATASET

The stock market produces large amounts of data every second [4], mostly numbers such as prices, volume, adj close, close, open, high, low. But another form of data that these numbers can't show is human emotions about the stock. Insights of business or fundamental analysis can provide a good overview of the business in long term and how strong the business is fundamentally but when doing predictions, even a business with strong fundamentals and great numbers can start doing bad in short term, and vice versa. This happens because of sudden shift in human emotions about the stock or about stock market as whole. With growing technology, humans can share their emotions on multiple platforms, and Twitter is the most popular among all the platforms where many investors, public figures, retail investors, and stock analyst are sharing their views daily about different stocks.

3.2 Twitter Data Scraping

To scrap the Twitter data, there are several methods including **Twitter's API**. But after I started my research, I started to look for a library that can perform this task with minimal effort. After some research, I learned about **snsrape (SNS)**[5], a well-known scraping library for many social media platforms, including information such as user profiles, hashtags, and specific contents for searches. I scraped data for \$SPY stock between (01/01/2020 to 01/01/2023) with all the contents that included \$SPY, SPY S&P500, \$AAPL, \$META, \$AMZN, and related words.

The python library that we're using is sncscrape, this library does not require any Twitter API at all, to perform the scraping, it follows these methods:

- i. URL construction: sncscrape constructs the appropriate URLs based on the search query or user profile you want to scrape.
- ii. Sending requests: It sends HTTP requests to the constructed URLs. This step mimics a browser accessing the web page, allowing the scraper to retrieve the content without using the official API.
- iii. HTML parsing: Once the content is retrieved, sncscrape parses the HTML using Python's BeautifulSoup4 library. It navigates through the HTML structure to locate the information you're looking for, such as tweets, user profiles, or other related data.
- iv. Data extraction: After locating the desired information, sncscrape extracts it from the HTML and returns it in a structured format, such as a list of dictionaries or custom objects.
- v. Pagination: If the platform requires pagination (loading additional content when scrolling down), sncscrape handles this by sending more requests and repeating the previous steps to collect the new data.

3.3 Yahoo Stock Data

To perform the Time Series Analysis, we also needed a stock's historical data, Yahoo Finance is well known for all the stock's historical data. I retrieved \$SPY stock data for x years range between (01/01/2010 to 03/01/2023).

3.4 Preprocessing

We know that before performing any analysis on data, we need to make sure data is prepared according to our needs. As we have two different datasets, one the stock dataset and the other Twitter's data, so we utilized different preprocessing techniques for each. Let's dive into the preprocessing techniques used for both data.

3.5 Stock Data Preprocessing

As the data was retrieved from Yahoo finance, we did not see any major issues with stock data except for missing values. If we encountered missing values, we used an average of previous day (X_{prev}), and the next day (X_{next}). [6]

$$y = (X_{prev} + X_{next}) / 2$$

3.6 Twitter Data

We used SNS to scrape twitter data, as this framework allows us to search for all the tweets containing the required contents that we're looking for, and this allows us to save the retrieved data in form of CSV files. After retrieval of all the data, we encountered multiple issues with Twitter data, where we saw symbols like '@', '#', '!' and many other symbols but removed all symbols except for '#' because '#' or hash may contain important information about the topic [6]. Also, we removed long tweets with too many characters (>140 characters). We also removed the tweets that contained URLs because that would also be worthless for sentiment analysis, as we can't get emotions from the URL unless we access the information from that URL which is beyond the scope of this study. After cleaning of data, we manually labeled 400 tweets as 'positive' or 'negative' since that information is not available in the dataset. After this, we had a training dataset for our analysis.

After getting all the Twitter data, and performing sentiment analysis, which we will be discussing more in further chapters. We aggregated the scores of all the data for each day and used that as the score for the day. Also, since the stock market is closed on weekends, we used aggregate score of whole weekends to do analysis for Monday, when the stock market opens.

4 Chapter IV

4.1 Overview of Natural Language Processing

We use computers on daily basis, it may be in form of a laptop, mobile phone, calculator, or even small appliance like a microwave has a computer in built. These computers may be different in functionality and may look different but all of them speak the same language which is Binary. We all know what binary is, it is a number system based on '1' and '0'. Computers show us texts, videos, music, and so much more but it is all based on '1' and '0'. When we use alphabets, that is also ultimately converted into series of zeros and ones.

But recently we have started to see many technologies that are becoming better and better at understand human texts and speeches, that is possible because of Natural Language Processing (or NLP). Natural Language Processing is an area of Machine Learning that deals with human languages and processing them in a way that can help computers understand and respond with a human language. Natural Language Processing began in 1950s as an intersection of Artificial Intelligence and Linguistics[7]. There are many approaches to NLP, which has been discussed in [7], such as using Support Vector Machines, Hidden Markov Models, Conditional Random Fields, N-grams, etc.

4.2 Sentiment Analysis

Sentiment Analysis is a Natural Language Processing (NLP) application. But what is Sentiment Analysis? We have seen a massive growth of social media and online businesses, with this growth people come to these platforms and share their thoughts and comments. Consider Amazon for example, People buy products on Amazon, and they comment their reviews. That review can be good or bad or neutral if we choose to

differentiate them into 3 levels. While it is possible for a person to go read the review and address them accordingly, but in a mass scale it becomes an extremely human intensive labor and very expensive. This is an area where Sentiment Analysis can be used, to understand the sentiment of the comment provided by the customer and that would ultimately help the company cater to the customers.

Similarly, people also share their thoughts and ideas about many things on social media platforms and Twitter is widely popular for investors and traders compared to other big platforms.

4.3 Sentiment Analysis Pre-Trained Models

There are many pre-trained models that are available for free, so instead of training with our own model, we choose to use a pre-trained model as they are trained with huge datasets and have proven record as excellent sentiment analysis models. We investigated multiple options, but we choose three models to initiate our study, these models are RoBERTa[8], FinBERT[8], and VADER Sentiment Model.

Before discussing these models, we need to make sure we understand two things about these models.

First, Transformer based models[9] uses Neural Network Architectures widely used in NLP [9]. These use an attention layer or attention mechanism that allows the model to selectively focus on different parts of the input while making predictions. The model assigns weight to each word and then this helps the model to determine relationship between words that were inputted.

Second, the model based on Lexicon of sentiment [10] related words is pre-programmed with sentiment related words. We can say that these types of models employ a sentiment dictionary and use that to determine sentiment score.

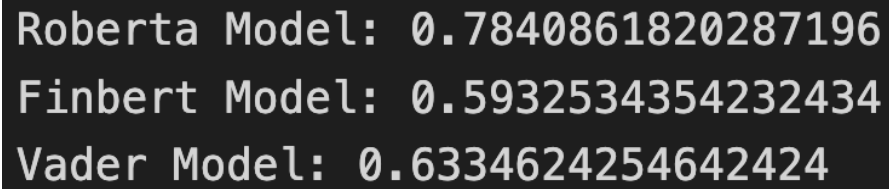
RoBERTa: RoBERTa model is a pre-trained transformer-based model developed by Facebook AI research. This model is considered state-of-the-art model for many benchmarks in NLP.

FinBERT: FinBERT model is also a pre-trained transformer-based model and in fact a derivative of RoBERTa model that has been fine tuned for financial data. This model outperforms many other NLP models in financial data.

VADER: Valence Aware Dictionary and sEntiment Reasoner or VADER is a Lexicon based model trained on emotions, emojis, social media posts. [10]

4.4 Implementation of Twitter Sentiment Analysis Models

After collection of data from Twitter, we started to work on the collected data. First as mentioned earlier, we started to clean the data and keep it under 140 characters. After this cleaning process, we manually scored 400 rows out of about 30,000 rows, because that was doable manually and to ensure that we use the model with less error when compared with a human scored sentiment. Using a Roberta model, FinBERT model and VADER Sentiment Analyzer, we calculated Root Mean Squared Error for each model and got the result shown in Figure 1..



Roberta Model: 0.7840861820287196
 Finbert Model: 0.5932534354232434
 Vader Model: 0.6334624254642424

Figure 1: RMSE for each model

Root Mean Squared Error (RMSE) is statistical method of calculating error between actual and predicted values. In our case, human calculated sentiment scores are being considered a benchmark and all 3 models are predicted scores. [11]

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Where y_i : This represents the i^{th} actual or observed value of the dependent variable.

\hat{y}_i : This represents the i^{th} predicted value of the dependent variable.

n: is total number of observations.

Figure 1 shows that the FinBERT model performs best compared to other two models. However, performing sentiment analysis using Roberta and FinBERT model require a lot more computing power and computational time (20+ hours for our dataset). To compensate for the time and computation, we did further study using VADER Model, as it was somewhere in between both models and is also an extremely fast model, which allowed me to perform sentiment analysis on different datasets as well.

We also retrieved user information such as their followers count, verification status, and information about the tweet such as like count and retweet count.

We performed some minor data analysis such as correlation score and found that ‘Like Count’ and ‘Followers Count’ does not have strong correlation with each other, similarly

‘Like Count’ and ‘Retweets’ also did not have strong correlation, and we did not observe any strong correlation between all the features except in case of Verified users having high follower count. After this analysis, we choose to use ‘Like Count’, ‘Followers Count’, ‘Retweet’ for our new scores with different weights. In hopes of getting some better insight of data, I used different weights for each feature such as square root and cube root, as some of these numbers were too high compared to other features. After getting new scores, I normalized the scores between -1 to 1 scale. These scores were then merged with stock data with ‘Date’ as common value for both tables, that also eliminated the issue to deal with weekends.

Instead of comparing with prices, I choose to use returns instead, as returns would indicate percentage change in stock price each day.

$$\text{Return}_i = \frac{\text{Close}_i - \text{Close}_{i-1}}{\text{Close}_{i-1}}$$

Return_i : This is the return on the stock for day

Close_i : This is the closing price of the stock on day i .

Close_{i-1} : This is the closing price of the stock on the previous day, i.e., day $i - 1$.

Figure 2 shows the relationship between sentiment scores using the VADER algorithm and the stock return over a period of five days in March.

Date	Return	vader_sentiment_scores
2020-03-03	-0.028632	-0.105375
2020-03-04	0.042033	-0.226460
2020-03-05	-0.033242	0.000000
2020-03-06	-0.016531	0.166120
2020-03-09	-0.078094	-0.091067

Figure 2: VADER Sentiment Score & Returns

We created similar data frames with other scores using weights for followers count and like count. The data frame had Returns and different version of the score. We then tested for Pearson correlation with Score and Returns. We know that correlation does not prove causation[12], but it was important to know if there is any correlation at all.

We compared all scorers with 'Returns' in hopes of seeing correlation but, we did not observe any correlation with any score and 'Returns'. Here is the plot to visualize the 'Returns' and 'Score'.

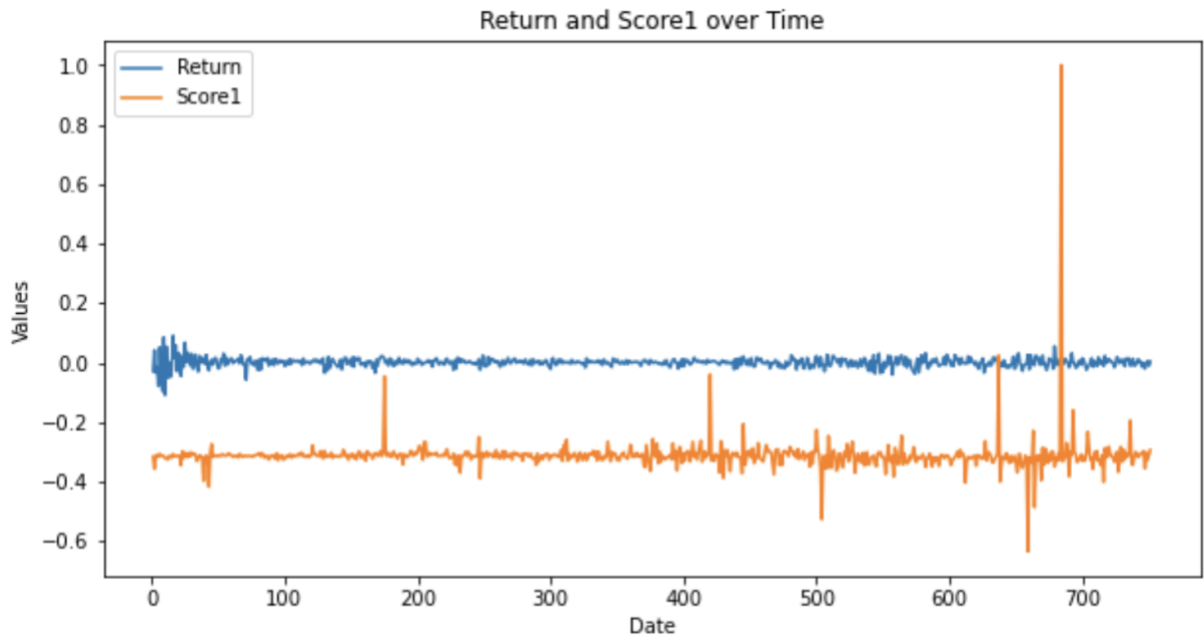


Figure 3: Return vs Score plot

We saw similar plots for each of the scores, which was surprising as we expected them to at least be correlated. Correlation would not have proved causality, but having a correlation was an important part of the prediction.

To investigate this further, I manually checked a lot more tweets and found many of the Tweets were not related to S&P500, or just not good enough to be considered for the analysis. Upon reviewing many of the rows manually, I discovered a lot of noise and irrelevant tweets related to SPY or people just talking about stocks without any knowledge at all. Noise most likely impacted model's effectiveness and resulted in irrelevant sentiment scores.

We also performed similar analysis using only Top 50 Financial accounts on Twitter as well as some famous names such as Elon Musk and other famous individuals within the stock market, giving a total of 65 Twitter accounts. Unfortunately, this model

had issue with collecting data that was related to S&P500, as these people do not always post related to \$SPY stock or even related to stocks that I scraped data for earlier.

Because of this, we adjusted scraping so that if these accounts did not post anything then we should impute with data from other account that has at least 100 likes and has queries that we are looking for. We again got similar result and did not find a strong correlation.

This time, we had Pearson correlation of 0.34, so we choose to not investigate further and concluded the sentiment study. Results from Twitter data suggest that sentiment scores from Twitter does not help to predict stock market or has any correlation with returns from SPY stock.

5 Chapter V

5.1 Introduction to Time Series Analysis

Stock Market prediction has been popular topic for years, there are many ways mathematicians, traders, researchers, investors, hedge funds etc. has been trying to predict stock market movements. Some popular methods historically have been Fundamental Analysis and Technical Analysis. Both methods have their own other methods and theories, Technical Analysis is the idea we're currently interested into because it's potential to predict short time prices with some degree of accuracy.

Technical Analysis uses multiple methods such as use of indicators, graphs, plots, and charts, using mathematical models such as regression models, etc. to perform Technical Analysis. We will be discussing one of the most reliable models called Long Short Term Memory Networks (aka LSTMs), but before discussing LSTMs it is important to learn about Recurrent Neural Networks (aka RNNs).

5.2 Recurrent Neural Networks (RNNs)

Before we discuss Recurrent Neural Networks, it's important to just provide a brief description of a Neural Network. The idea of Neural Network comes from our brain, we try to mimic a brain by using Neural Networks, these networks are then used to train a model to perform certain task with accuracy like or even better than humans in some cases.

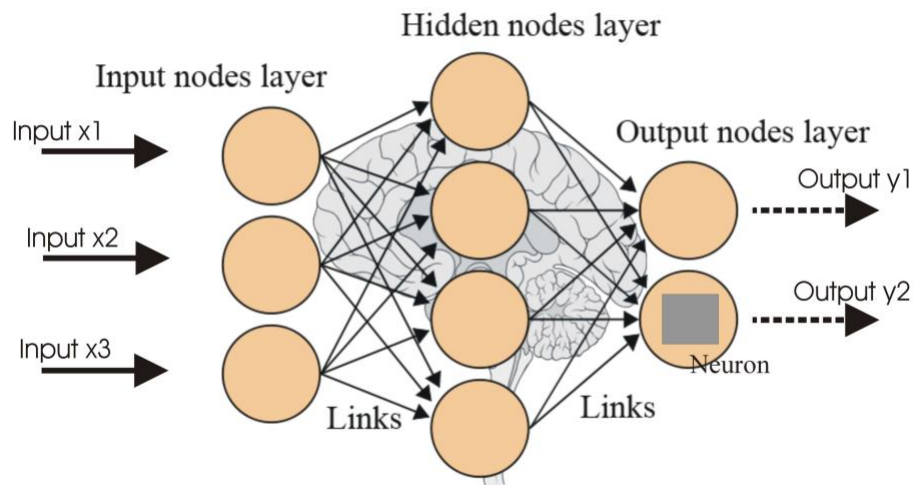


Figure 4: Simple Neural Network

Now that we have a general idea of Neural Networks, let's try to understand Recurrent Neural Networks or RNNs[9]. RNNs are Neural Networks but with loops, as we see in the diagram above, if we must process something we put an input into the Neural Network model and it processes, then results with an output. But if we give the same input, it again must process the input, doesn't have a memory to store the information on previous inputs. But brains don't work like that, we do have a memory, we use this memory for everything we do almost. If we're trying to read something, to understand each sentence it is important that we remember what came before each word to make a sense out of it. RNNs are still widely used in speech recognition, translation and other technologies where sequential learning is useful. But there is a major drawback of RNNs, that will be discussed further with LSTMs.

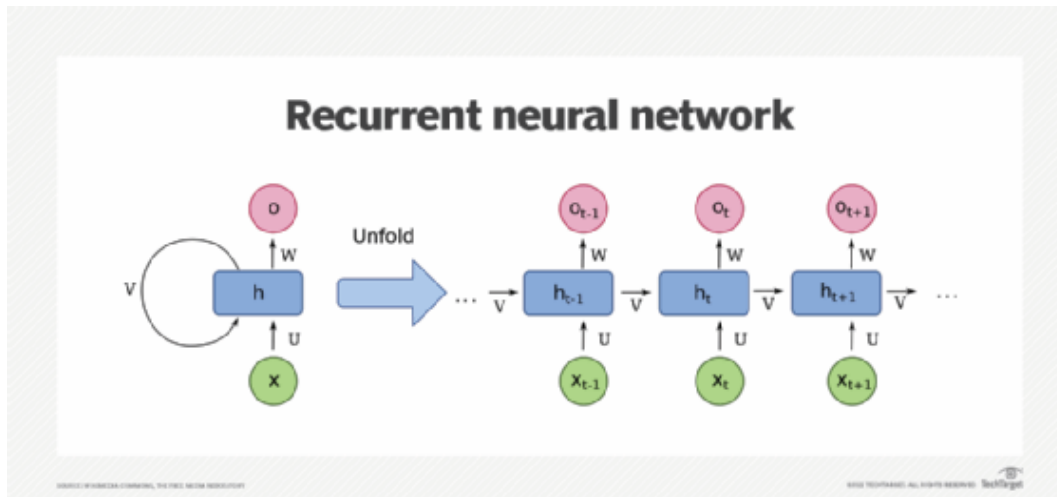


Figure 5: RNNs

5.3 Long Short-Term Memory

As we already have RNNs, do we really need LSTMs? What problems does LSTMs solve? RNNs are somewhat good at predicting next thing depending on the current context. Let's understand this with an example:

Examples 1: We *are* learning about LSTMs.

RNN would know that we need 'are' when using a trained RNN.

Example 2: I grew up in Nepal, I speak *Nepali*.

RNN would know that we need 'Nepali' because it does not remember the context of growing up in Nepal.

This is not a problem that LSTM model would have; it would easily know that 'Nepali' is what needs to be there.

Long Short-Term Memory (LSTMs) were introduced by Hochreiter & Schmidhuber in 1997. LSTM models have become much popular since then and it is

widely being used to process sequential data such as speech, text, and time series data. This model solves the issue that an RNN model has which is vanishing gradient problem. LSTM models are successfully implanted in wide range of applications such as speech recognition, natural language processing, image captioning [13] To understand how LSTM works and remembers all the important information, it is important for us to understand brief overview of the LSTM Architecture.

There are key components of an LSTM that helps it to overcome the drawbacks of RNN.

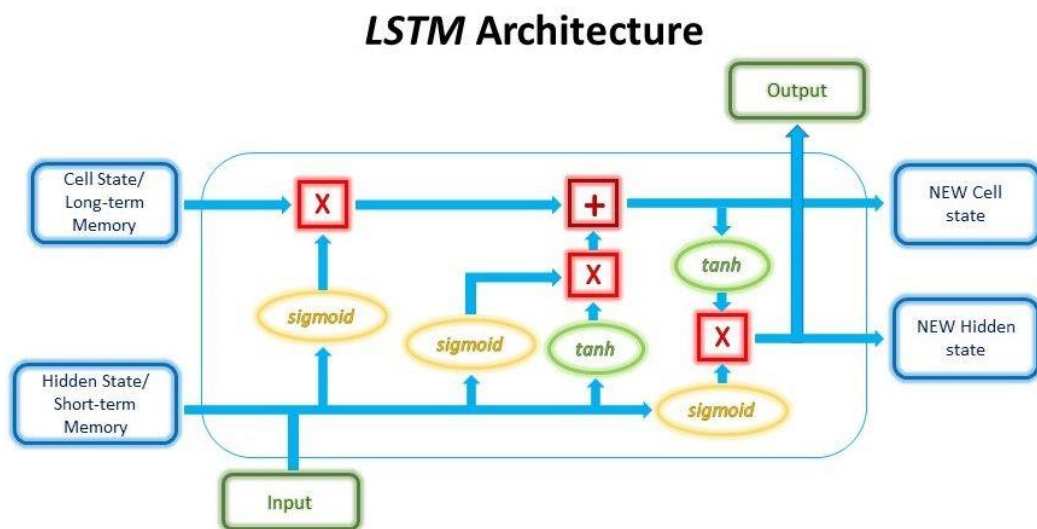


Figure 6: LSTM Architecture[14]

Components of LSTM network:

- i. Input Layer: This layer is responsible for inputs, and this is the first layer of LSTM. Inputs can be text, audio, or any sequential data.
- ii. LSTMS cells: These are the major building blocks for the LSTM networks. Each cell is more like a memory unit, which stores information overtime. This model

also uses 3 gates to process information: input, output and forget gate. By utilizing these gates, cells regulate the flow of input and output for each cell.

- iii. Hidden Layer: Hidden layers or states processes the output of LSTM cells which is responsible for applying a non-linear activation function to the output. These non-linear activation functions introduce non-linearities into the network, which allows it to learn more complex and abstract representation of the input. These non-linear activations functions are commonly 'Sigmoid', 'Tanh', 'ReLU', 'SoftMax'.
- iv. Output Layer: This is the final layer, where output can be in form of a continuous value or in a form of probability distribution.

During the process of training a LSTM Network, we try to minimize the differences between predicted output and true input which is done using backpropagation through time (or BPTT), which is a variant of backpropagation that is used in training of RNNs.

5.4 LSTM Models

We just learnt about LSTMs, and we have a general idea of how great they are for sequential data and time series data. But can we use this for predicting Stock Market? LSTMs are great models that may have potential to predict the stock prices as they're good for time series data. As we obtained data from Yahoo, for \$SPY.

We first started with a basic LSTM model; We used 70% of the data to train the LSTM model and 30% to test the dataset. Which is a standard practice in Machine Learning to use a 70-30 train-test split. After training was completed, we got Root Mean

Squared Error (RMSE) of 73.31. Which is not a great model, especially when we're dealing with money it can be very risky. We started with 30 epochs to train this model. Before moving forward, it is better to discuss Epochs.

Epochs: When we say 30 epochs, this means that the LSTM model was presented with the training data 30 times. During each epoch, we try to minimize the difference between input and output. And the difference between input and output is called 'Loss', the goal is to minimize the difference.

Here is the plot that is from the basic model, with no tuning.

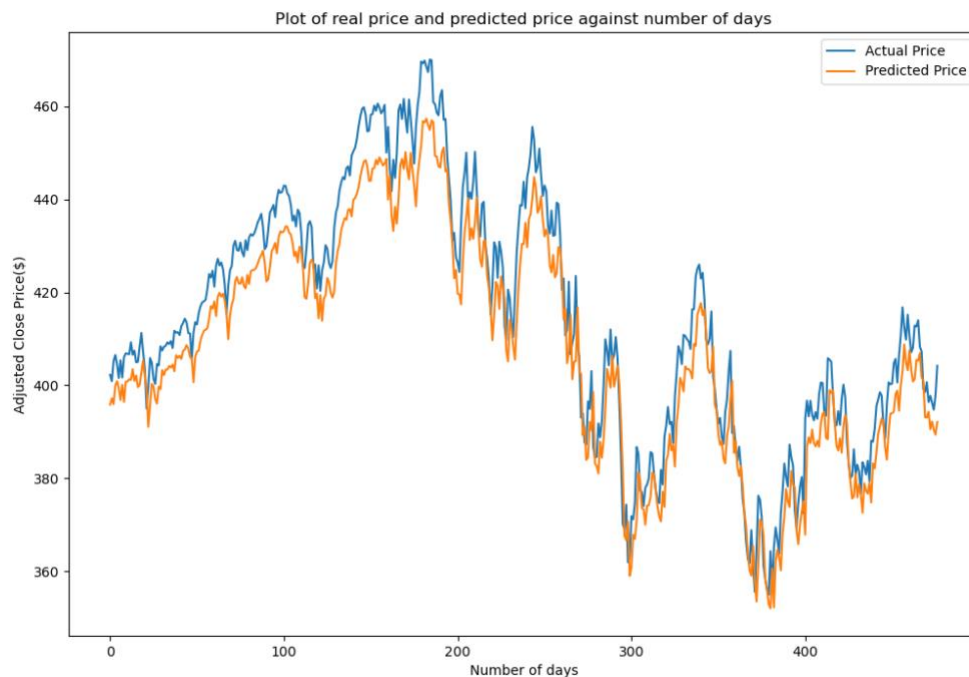


Figure 7: Basic LSTM Model Result

After this model, as we were unsatisfied by the result, we did some hyper-parameter tuning which basically means to adjust parameters in hopes of making the model better.

After some hyper-parameter tuning, we used 160 epochs, as we had minimal loss at 160 epochs.

To perform this hyper-parameter tuning, we trained the model multiple times with different input values and stores the loss for each epoch. At 160 epochs, we saw a minimal loss, after this point we saw fluctuation in loss value which is an indication of over-fitting. We must try to avoid over-fitting, because then our test set would not provide us a reasonable output.

Below, we can see the Actual vs Predicted plot for test set, using 160 epochs. We got a RMSE of 62.51.

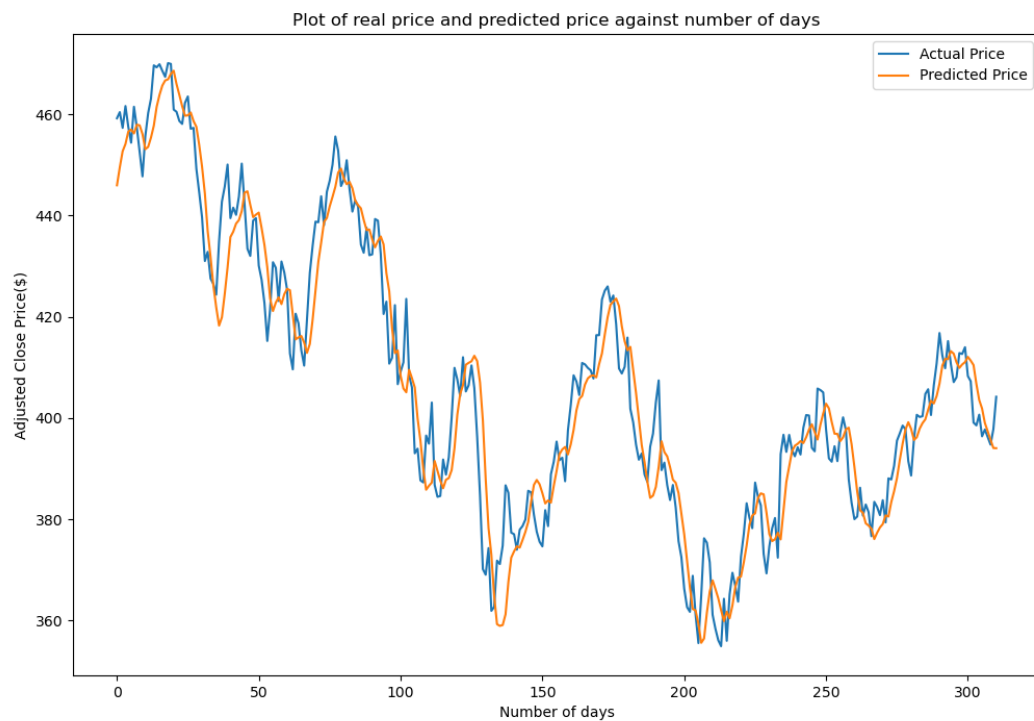


Figure 8: Actual vs Predicted with 160 epochs.

This is getting better, but still this error is not worth believing in a LSTM model for stock price prediction.

We needed better results, as suggested in this article [15] we then focused on a seasonality model and while also using raw stock prices as input. To get the seasonality component, we use stock price differences as an input and previous day's price. When we combined both, we trained an LSTM model with more layers and only 25 epochs, because we do not want to overfit the model. We selected the epoch value, after performing a hyper-parameter tuning. When adding more layers, there's chance that the model may be overfit to the training data. After this training, when we tested the model on test set of data, we got a RMSE of 4.74.

LSTM Seasonality model overview:

- i. Model input: An input layer is created to accept the input data. This layer has a specific shape based on the number of data points in the input sequence.
- ii. LSTM layers: Two LSTM layers are added to the model with 21 and 32 units each. The first LSTM layer is designed to return the output sequence, while the second one does not.
- iii. Dropout layers: To avoid overfitting the model, dropout layers are added after each LSTM layer. These layers have dropout rates of 0.1 and 0.05.
- iv. Dense layers: Two dense (fully connected) layers are added to the model with 32 and 1 units each.
- v. Output layer: An activation layer with a linear activation function is added to serve as the output layer of the model.

- vi. Model definition: The model is put together using the input and output layers. The Adam optimizer is set up with a learning rate of 0.002.
- vii. Model compilation: The model is compiled using the optimizer configured earlier and Mean Squared Error (MSE) as the loss function.
- viii. Model training: The model is trained using the input data (X_{train} and y_{train}) with a batch size of 15 and 25 epochs. The dataset is shuffled, and 10% of the data is used for validation.

Actual vs Predicted plot for model with 25 epochs:

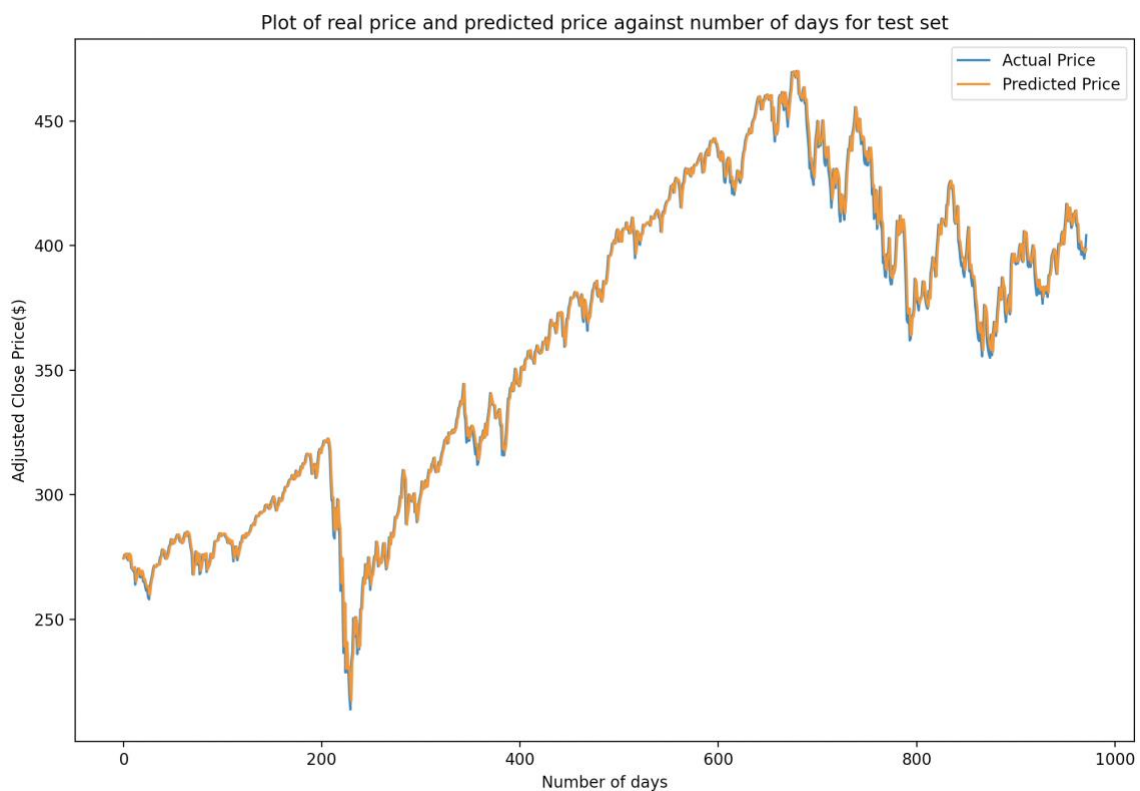


Figure 9: Actual vs Predicted for 25 epochs.

This model is potentially a model that may generate some profits. Even if this is not a perfect model, LSTM models with seasonality and raw stock prices are one of the best methods to train an LSTM model.

6 Chapter VI

6.1 Summary

In this study we explored two different ways Machine Learning can be utilized in stock market price prediction. We explored two different areas of Machine Learning, Natural Language Processing and Neural Networks. Both are growing field of Machine Learning, and we are seeing endless possibilities of these Machine Learning areas. While in this study, Twitter's data did not provide great insights that could help us predict potentially predict stock market, we still managed to perform sentiment analysis that was very close to a human sentiment. This may or may not help predict stocks in future, but we do know that NLP can be utilized in many ways to understand human language, and the possibilities of that is just endless.

We also performed Time Series prediction using Long Short Term Memory Networks, that did perform well enough for an investor to take some risk. There is always risk, because no model would be perfect to predict something that is deemed to be unpredictable. But, as researcher it is the goal to try to increase the predicting accuracy by some degree, while it may not always be able to predict but it is worth a try to use different mathematical and computational ways to predict the market.

6.2 Potential Future Work

While doing this study, we found that there is too much noise or irrelevant data in Twitter. While Twitter is considered a benchmark, this study showed otherwise even after extensive cleaning and manipulation of data. There are few things we suggest future

work, we realized while doing this that may be employing these for future work may improve correlation or even predictions using sentiment. They're listed below:

- i. Using StockTwits might be a better alternative, as they're focused much more on people who trade stocks and there will be less chance of noise.
- ii. Potentially News headlines might also be an alternative, as they're not as biased as people commenting and posting on social media platforms.
- iii. If successful in obtaining sentiment scores that are correlated to the stock returns, combining both sentiment and LSTM model might be able to provide even better predictions.

7 References

1. Qian B, Rasheed K (2007) Stock market prediction with multiple classifiers. *Applied Intelligence* 26:25–33. <https://doi.org/10.1007/s10489-006-0001-7>
2. Pagolu VS, Reddy KN, Panda G, Majhi B (2016) Sentiment analysis of Twitter data for predicting stock market movements. In: 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPEs). IEEE, pp 1345–1350
3. Mittal A (2011) Stock Prediction Using Twitter Sentiment Analysis
4. Understanding charts for stock/index movements - efinacademy.com. <https://efinacademy.com/understanding-charts-for-stock-index-movements/>. Accessed 18 Mar 2023
5. Snsrape link:<https://github.com/JustAnotherArchivist/snsrape>
6. Kordonis J, Symeonidis S, Arampatzis A (2016) Stock Price Forecasting via Sentiment Analysis on Twitter. In: Proceedings of the 20th Pan-Hellenic Conference on Informatics. Association for Computing Machinery, New York, NY, USA
7. Nadkarni P, Ohno-Machado L, Chapman W (2011) Natural language processing: An introduction. *J Am Med Inform Assoc* 18:544–551. <https://doi.org/10.1136/amiajnl-2011-000464>
8. Arslan Y, Allix K, Veiber L, et al (2021) A Comparison of Pre-Trained Language Models for Multi-Class Text Classification in the Financial Domain. In: The Web Conference 2021 - Companion of the World Wide Web Conference, WWW 2021. Association for Computing Machinery, Inc, pp 260–268

9. Gillioz A, Casas J, Mugellini E, Khaled OA (2020) Overview of the Transformer-based Models for NLP Tasks. In: Proceedings of the 2020 Federated Conference on Computer Science and Information Systems, FedCSIS 2020. Institute of Electrical and Electronics Engineers Inc., pp 179–183
10. Hutto CJ, Gilbert E (2014) VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text
11. Chai T, Draxler RR (2014) Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature. *Geosci Model Dev* 7:1247–1250. <https://doi.org/10.5194/gmd-7-1247-2014>
12. Bollen J, Mao H, Zeng X (2011) Twitter mood predicts the stock market. *J Comput Sci* 2:1–8. <https://doi.org/10.1016/j.jocs.2010.12.007>
13. Olah C (2015) Understanding LSTM Networks -- colah's blog
14. Loya G (2019) Long Short-Term Memory: From Zero to Hero with PyTorch
15. Lee YH Analysis of Stock Price Predictions using LSTM models | by Yu Hao Lee | Analytics Vidhya | Medium. <https://medium.com/analytics-vidhya/analysis-of-stock-price-predictions-using-lstm-models-f993faa524c4>. Accessed 14 Mar 2023

8 Appendix: Important Code From The Study

```
def lstm_model(X_train, y_train, data_set_points, epochs):
    #Setting of seed (to maintain constant result)
    tf.random.set_seed(10)
    np.random.seed(10)

    lstm_input = Input(shape=(data_set_points, 1), name='input_for_lstm')

    inputs = LSTM(21, name='first_layer', return_sequences = True)(lstm_input)

    inputs = Dropout(0.1, name='first_dropout_layer')(inputs)
    inputs = LSTM(32, name='lstm_1')(inputs)
    inputs = Dropout(0.05, name='lstm_dropout_1')(inputs) #Dropout layers to prevent
overfitting
    inputs = Dense(32, name='first_dense_layer')(inputs)
    inputs = Dense(1, name='dense_layer')(inputs)
    output = Activation('linear', name='output')(inputs)

    model = Model(inputs=lstm_input, outputs=output)
    adam = optimizers.Adam(learning_rate = 0.002)

    model.compile(optimizer=adam, loss='mse')
    model.fit(x=X_train, y=y_train, batch_size=15, epochs=epochs, shuffle=True,
validation_split = 0.1)

    return model
```

