Stock Price Forecasting and Volatility Analysis

A THESIS REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING

IN

DIVISION OF COMPUTER ENGINEERING

Submitted by:

Nipun Jain, 2017UCO1543

Harshit Mahajan, 2017UCO1565

Mohit Motiani, 2017UCO1569

Under the supervision of

Dr. Preeti Kaur

DIVISION OF COMPUTER ENGINEERING

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF DELHI

DECEMBER, 2020

Department of Computer Engineering

University of Delhi

Delhi-110007, India

CERTIFICATE OF ORIGINALITY

We, Nipun Jain (2017UCO1543), Harshit Mahajan (2017UCO1565) and Mohit Motiani (2017UCO1569) students of B. Tech. Department of Computer Engineering, hereby declare that the Project-Thesis titled "Stock Price Forecasting and Volatility Analysis" which is submitted by us to the Department of Computer Engineering, Netaji Subhas Institute of Technology, Delhi (University of Delhi) in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is original and not copied from source without proper citation. The manuscript has been subjected to plagiarism check by Turnitin software. This work has not previously formed the basis for the award of any Degree.

Place: Delhi

Date: 01/01/2021

Nipun Jain, 2017UCO1543

Harshit Mahajan, 2017UCO1565

Mohit Motiani, 2017UCO1569

Department of Computer Engineering

Netaji Subhas University of Technology

Dwarka, Delhi-110078, India

CERTIFICATE OF DECLARATION

This is to certify that the Project-Thesis titled " Stock Price Forecasting and Volatility Analysis" which is being submitted by Nipun Jain (2017UCO1543), Harshit Mahajan (2017UCO1565) and Mohit Motiani (2017UCO1569) to the Department of Computer Engineering, Netaji Subhas Institute of Technology (University of Delhi) in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the thesis work carried out by the students under my supervision and guidance. The content of this thesis, in full or in parts, have not been submitted for any other degree or diploma.

Place: Delhi

Date: 01/01/2021

Dr. Preeti Kaur

SUPERVISOR

ABSTRACT

It is widely acknowledged that stock price prediction is a job full of challenges due to the highly unpredictable existence of financial markets. Many market participants or analysts, however, attempt to predict stock prices using different mathematical, econometric, or even neural network models in order to make money or understand the nature of the equity market. In the past few years, a lot of models based on deep learning have been gaining popularity for predicting the volatility of the stock market prices.

In this project, the outcomes of many classical deep learning models such as LSTMs, GRUs, CNNs and their several common variants are contrasted with two distinct stock price prediction targets: absolute stock price and volatility. The aim of the comparative study is to find out which model is the best fit for stock market prediction.

We also attempt to research the relationship between news and stock trends, believing that news stories have an impact on the stock market by incorporating sentiment analysis into our model. Our methodology was to scrape news articles of a particular stock and use the corpus gathered to generate a sentiment score which is further used as an input to the model.

ACKNOWLEDGEMENT

We would like to take this opportunity to acknowledge the support of all those without whom the project wouldn’t have been possible.

We sincerely thank our mentor, **Dr. Preeti Kaur** for her guidance, criticism and encouragement which led to the completion of the project. The regular interactions with her gave us a deep insight into the topic and evolved our ideas. Her enthusiastic attitude has helped us to sail through and stay on track. We thank her for giving us this opportunity and supporting us throughout.

We wish to express heartfelt gratitude to our college, Netaji Subhas Institute of Technology for giving us this opportunity for research and development. Lastly, we would also like to thank our parents who supported us through our thick and thin. Their trust in our capabilities helped us in giving our best.

Place: Delhi

Date: 01/01/2021

Nipun Jain, 2017UCO1543

Harshit Mahajan, 2017UCO1565

Mohit Motiani, 2017UCO1569

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS, SYMBOLS AND NOMENCLATURE

RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
GRU	**G**ated **R**ecurrent **U**nit
CNN	**C**onvolutional **N**eural **N**etwork
Seq2Seq	**S**equence **t**o **S**equence
EMH	**E**fficient **M**arket **H**ypothesis
API	**A**pplication **P**rogramming **I**nterface
NLP	**N**atural **L**anguage **P**rocessing
SI	**S**entiment **I**ndicator
KNN	**K** **N**earest **N**eighbour
SVM	**S**upport **V**ector **M**achine
GPU	**G**raphics **P**rocessing **U**nit
CSV	**C**omma **S**eparated **V**alues
BRNN	**B**idirectional **R**ecurrent **N**eural **N**etwork

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 DETAILED PROBLEM STATEMENT

A stock also referred to as equity, is a security representing the holding of a fraction of the company. This grants the shareholder the right to a share of the assets and earnings of the corporation equal to how much of the stock they own. The stock market refers to the set of stocks and trades in which shares in publicly listed companies are acquired, exchanged and released on a daily basis.

A stock market prediction is an attempt to forecast the future trend of an individual stock, a particular sector of the market, or the market as a whole. These forecasts generally use fundamental analysis of a company or economy, or technical analysis of charts, or a combination of the two.

In the past few years, a lot of models based on deep learning have been gaining popularity for predicting the volatility of the stock market prices[1]. In this project, we try to implement several classical deep learning models like RNNs, CNNs, and their many variants by implementing them using the Tensorflow framework in Python[2].

Then we perform a comparative study of all the aforementioned models with our aim being to find out which model is the best fit for volatility prediction.

The next objective of this research is to observe how well the movements in a company's stock prices, the rises, and falls, are associated with the public views expressed in news articles regarding that company[3].

Therefore we will improve the predictive model's potential to correlate the public meaning of stock markets with investor opinion by adding a sentiment analysis module for news records. In order to forecast stock price movement for the next day, we use news sentiment and the values of the previous day.

1.2 MOTIVATION

The prediction of stock prices is a popular and significant problem. We can gain insight into market behavior over time with a good model for stock prediction, identifying patterns that would otherwise not have been observed. Machine learning would be an effective way to solve this issue with the rising computing capacity of the computer. Forecasting stock volatility is, therefore, an essential application of current knowledge and resources in the field of deep learning.

In an attempt to further increase the accuracy of our classical models we try to incorporate sentiment analysis into our models. One of the most enticing realistic uses of sentiment analysis is stock market forecasting. Another motivation to use sentiment analysis is that, according to the Efficient Market Hypothesis (EMH)[4], stock volatility cannot be predicted by historical prices alone in the long term as investors are guided by fear and greed[5].

Thus begins the second part of our project. For a given company's stock, we scrape news articles for it each day and run sentiment analysis on the articles to get a sentiment score which is an input to the model. We try to find out whether adding sentiment analysis gives us a reasonable increase in accuracy.

We plan to achieve this along with a few of our goals being:

1. Leverage sentiment analysis to improve stock volatility prediction
2. Learning deep learning paradigms
3. Skills for Debugging and Checking
4. Seeking and utilizing datasets that are correctly labeled

5. Writing a good quality code
6. Using the newest and reliable innovations

1.3 Objectives

Following are the objectives of this project:
I. Compare the performances of the following deep learning models on two different forecast targets of stock price: absolute stock price and volatility

- RNN
    - Vanilla RNN
    - Bidirectional RNN
    - 2-Path RNN
- LSTM
    - Vanilla LSTM
    - LSTM Bidirectional
    - LSTM 2-Path
    - LSTM Sequence to Sequence
    - LSTM Bidirectional Sequence to Sequence
- GRU
    - Vanilla GRU
    - GRU Bidirectional
    - GRU 2-Path
    - GRU Sequence to Sequence
    - GRU Bidirectional Sequence to Sequence
- CNN
    - Sequence to Sequence CNN
    - Dilated Sequence to Sequence CNN

II. Create an LSTM model encompassing historical stock prices and sentiment analysis. Compare the performance with and without using sentiment analysis to determine if using sentiment analysis enhances the performance of stock volatility prediction significantly.

1.4 Challenges

- To achieve fair precision, the dataset should be large enough.
- The models should be ideally trained using a GPU for lower training time and faster training iterations.
- The code for the models should be flexible enough to incorporate future design changes.
- Historical news article collection is a major bottleneck in this project.
- Selection of sentiment score generation method from news articles.
- Selection of model architecture and hyperparameters.
- Creating an adaptive, scale feature extraction proved to be difficult.

CHAPTER 2

THEORETICAL KNOWLEDGE

2.1 Neural Networks

Neural Nets are a cognitive model of the human brain that can handle complex problems and identify unknown correlations and patterns using statistical algorithms and mathematics. They are capable of interpreting input data by means of machine perception, clustering or labeling. The examples used for training these nets are typically hand-labeled beforehand, if supervised learning is employed. For example, an object recognition system could be fed thousands of marked images of vehicles, buildings, coffee cups, and so on, and visual patterns would be identified in images that correspond closely with similar labels.

Neural networks consist of several nodes articulated in layers, known as neurons. Similar to the way, neurons function in our brain, these nodes are also connected to several other nodes and trigger the functioning of each other. These interconnections are represented using weights which can be excitatory (or +ve) or inhibitory (or -ve) depending on the extent to which they influence each other's result. Neural Networks have gained widespread adoption in applications ranging from healthcare, financing to autonomous vehicles and other specialized expert tasks.

2.2 Recurrent Neural Networks

A recurrent neural network[6] widely used in the processing of natural language and speech recognition (NLP) is a type of artificial neural network. RNNs are programmed to understand the temporal features of the data and use trends to forecast the next possible scenario.

RNNs are used in the creation of models that replicate neuron function in the human brain. In situations where the context is important for predicting the result, they are incredibly powerful and are distinguished from other types of artificial neural networks because they use feedback loops to process a data sequence that informs the final outcome, which may also be a data sequence. These feedback loops cause knowledge to continue; the effect is often represented as memory.

Cases of Recurrent Neural Network use tend to be connected to language models in which the interpretation of the next letter in a word or the next word in an expression relies on the detail that comes before it.

2.3 Long Short Term Memory Networks

LSTM[7] networks are a category of Recurrent Neural Network which uses standard units and special units. LSTM units have a 'memory cell' that can store data over long periods of time in memory. A gate series is used to track where info arrives at the memory when it's produced, and when it's discarded..There are three kinds of viz gates, gate input, gate exit, and gate forget. The input gate defines how much information is stored in memory from the last sample; the output gate governs the sum of data transferred to the next layer and the tearing rate of the stored memory is controlled by forgetting gates. This design helps them to consider longer-term dependencies.

2.4 GRU

A Gated Recurrent Unit (GRU)[8] is a specific variant of the RNN i.e recurrent neural network which is commonly used for machine learning tasks related to memory and clustering. Gated recurrent units help to alter the weight of the input of the neural network to overcome the vanishing gradient problem common to recurrent neural networks.

The gated recurrent units include two gates namely, a reset gate and an update gate as an improvement of the overall recurrent neural network structure. The algorithm refines outputs of these two vectors and regulates the data flow into the model. Models with gated recurrent units can hold information over time, as most recurrent network models – which is why it is a "memory-centered" type of Neural network and is one of the easier means of defining these forms of technology. Other types of neural networks, in comparison, are also not able to hold information without a gated recurrent unit.

The uses for neural networks with gated recurrent units include but are not limited to audio recognition, text recognition, handwriting, OCR and many more tasks. In stock market analytics and government service, some of these networks are also used.

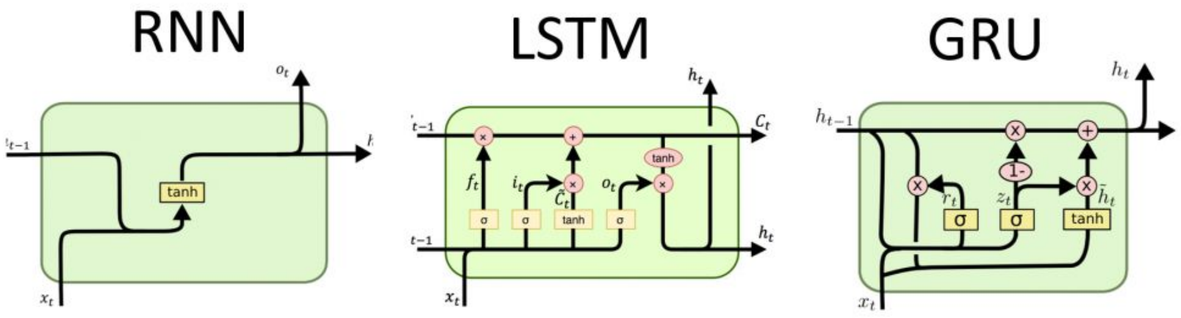


Figure 2.1 Comparison of RNN, LSTM and GRU architectures

2.5 Convolutional neural network

A Convolutional Neural Network[9] is a machine learning algorithm that takes input data, usually images, calculates weights that are defined on the pretext of some untold feature signals about several elements in the input given, and can easily carry out such applications over these inputs. In the areas of facial and object recognition, image detection, etc., CNNs have shown outstanding results.

CNNs also benefit from fewer restrictions than a fully linked network, making it simpler to train and house the same number of secret units. In an image, by converting them into a type suitable for achieving the desired goal, it can effectively capture spatial and temporal dependencies.

The CNN architecture is somewhat close to the connectivity of the human brain. A CNN is a sequence of layers that are convolutionary and sub-sampling, optionally followed by entirely related layers.

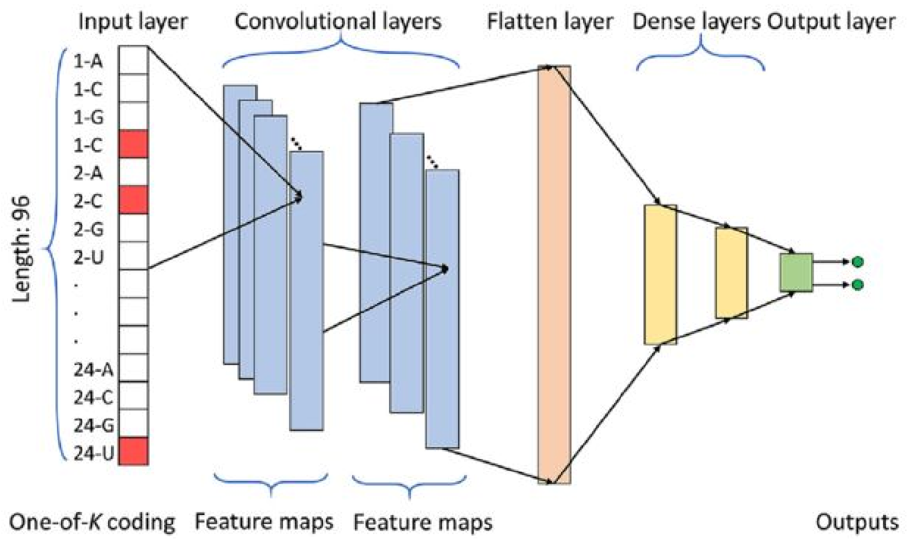


Figure 2.2 CNN Architecture

2.6 Variations

2.6.1 Bidirectional RNN (BRNN)

The bi-directional recurrent neural networks, BRNN[10], links two hidden layers going in opposite directions to the same output, allowing the algorithm to receive feedback from both the forward and the backward states.

BRNN's are trained to concurrently anticipate both the positive and negative paths of time, unlike traditional recurrent neural networks. The neurons of a normal RNN are divided by BRNN into two ways, one for forward states i.e the positive direction of time and one for backward states i.e the negative direction of time.

2.6.2 2-Path Networks

2-path networks[11] are an easy but efficient way to organize and model RNN layers in a deep structure. They break the long sequential input into smaller chunks and iteratively implement intra- and inter-chunk operations, where the input length can be made equal in each operation to the square root of the initial sequence length.

2.6.3 Sequence to Sequence models

Sequence-to-Sequence[12] (Seq2Seq) modeling is about training models that can translate sequences, such as Hindi to Russian, from one domain to sequences from another domain. The LSTM encoder and decoder carry out this Seq2Seq modeling.

Combining both the above approaches we also get the Bidirectional Sequence to Sequence variant of the LSTM and GRU.

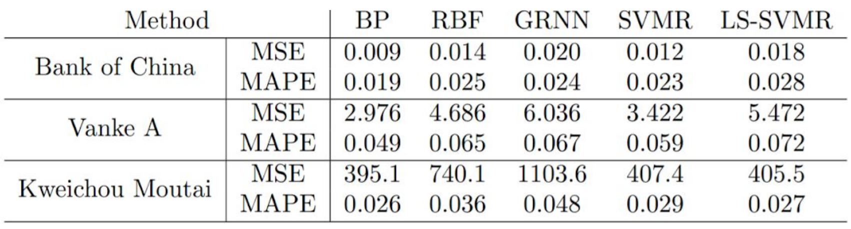
CHAPTER 3

REVIEW OF LITERATURE

3.1 Prior Work

In the paper ***"Neural networks for stock price prediction (2018)"* by Ren-Jie Han, Yu-Long Zhou, Yue-Gang Song**[13] , five neural network models, namely, back propagation (BP) neural network, radial base function (RBF) neural network, general regression neural network (GRNN), support vector machine regression (SVMR), least square support vector machine regression, are surveyed and compared with predictive capacity (LS-SVMR). They conclude that the BP neural network reliably and robustly outperforms the other four models by following mean square error and average absolute percentage error as parameters.

Over all the given stocks, the results show that the Back Propagation Neural Network's performance exceeds that of the other four models in terms of both Mean Squared Error i.e MSE and Mean Absolute Percentage Error i.e MAPE.



Method		BP	RBF	GRNN	SVMR	LS-SVMR
Bank of China	MSE	0.009	0.014	0.020	0.012	0.018
	MAPE	0.019	0.025	0.024	0.023	0.028
Vanke A	MSE	2.976	4.686	6.036	3.422	5.472
	MAPE	0.049	0.065	0.067	0.059	0.072
Kweichou Moutai	MSE	395.1	740.1	1103.6	407.4	405.5
	MAPE	0.026	0.036	0.048	0.029	0.027

Table 3.1 Results of BP, RBF, GRNN, SVMR, LS-SVMR models

Another paper "***Stock Prices Prediction using Deep Learning Models (2019)"* by Jialin Liu, Fei Chao, Yu-Chen Lin, and Chih-Min Lin**[14], talks about the challenges of using deep learning models for predicting stock prices. This is a challenge, since there is a lot of noise and confusion in stock price-related details. In order to denoise the data, this work utilizes sparse autoencoders with one-dimensional (1-D) residual convolutional networks. In order to forecast the stock price, long-short term memory, LSTM is then used. Prices, indexes and macroeconomic factors in the past are the attributes used to estimate the expense of the next day.

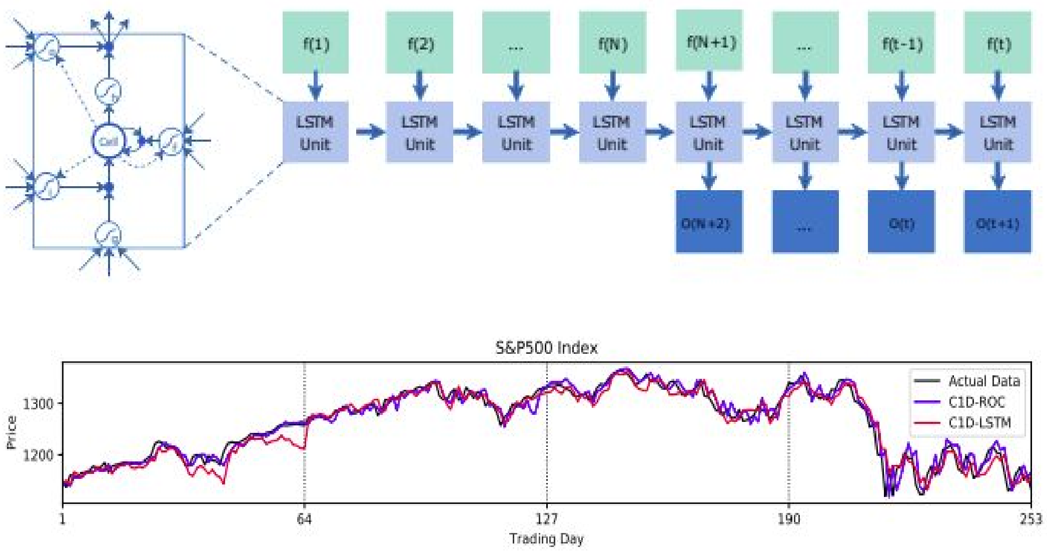
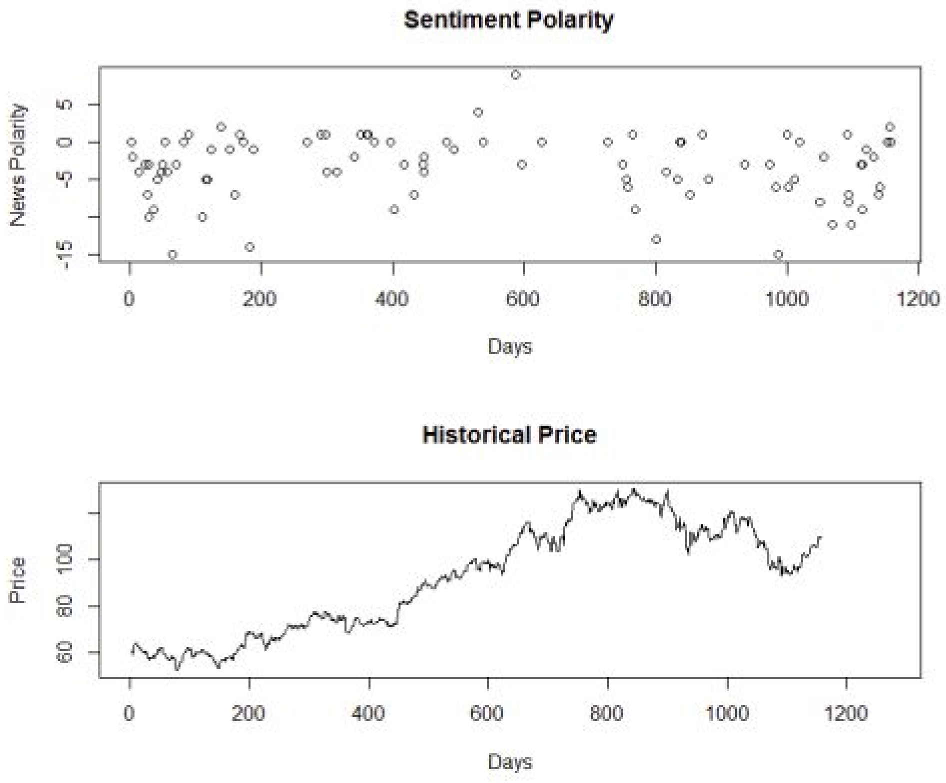
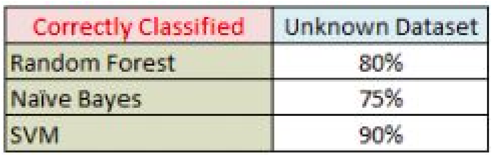


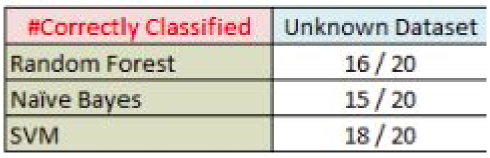
Figure 3.1 Results of LSTM on S&P500 Index

***"Stock Trend Prediction using News Sentiment Analysis"* by Kalyani Joshi, Prof. Bharathi H. N., Prof. Jyothi Rao**[15] focuses on the famous theory of stock prediction i.e the Efficient Market Hypothesis. This paper includes aggregation of unquantifiable information such as financial news reports from a company and estimating the potential trend in the market using a news sentiment categorization. This is an attempt to research the relationship between news and stock trends, believing that news stories have an impact on the stock market. They developed three distinct classification models to explain this which indicate that the polarity of news articles is positive or negative.





Correctly Classified	Unknown Dataset
Random Forest	80%
Naïve Bayes	75%
SVM	90%



#Correctly Classified	Unknown Dataset
Random Forest	16 / 20
Naïve Bayes	15 / 20
SVM	18 / 20

Figure 3.2 Graphs of Sentiment Polarity and Historical Price with model accuracies

Observations show that in all forms of testing, random forest and support vector machines perform well. Naive Bayes performs well, but not in contrast to the other two. The accuracy of the forecast model is over 80 percent and 50 percent accuracy relative to news random labeling; the model has improved accuracy by 30 percent.

***"Sentiment Analysis of Twitter Data for Predicting Stock Market Movements (2016)"* by Venkata Sasank Pagolu**[16] highlights that social media now-a-days are a true reflection of public perception and opinion on current affairs. An fascinating area of research has been stock market forecasting based on public opinions shared on Twitter. Earlier studies have shown that Twitter's aggregate public sentiment may well be linked to the Dow Jones Industrial Average Index. (DJIA).Two separate textual representations, Word2vec and N-gram, have been used in the present paper to examine public feelings in tweets.

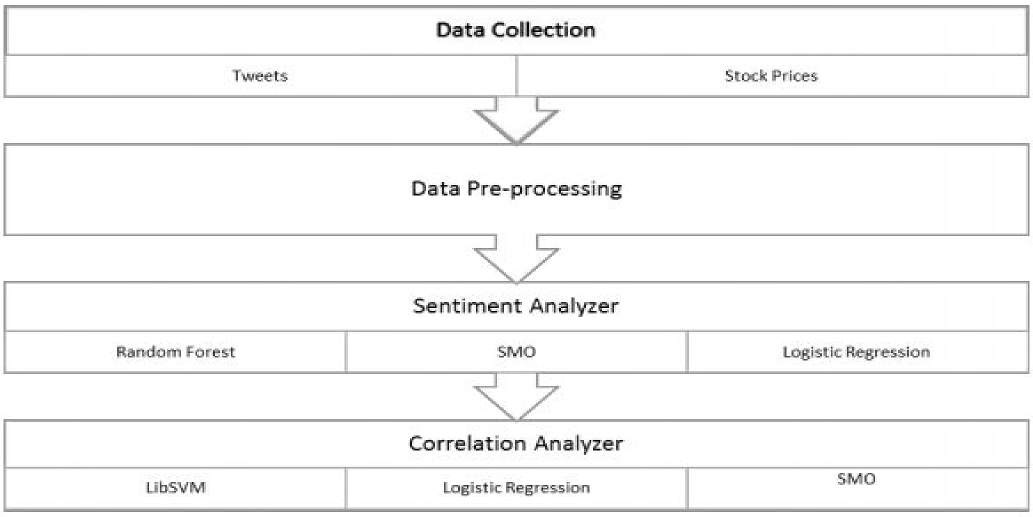
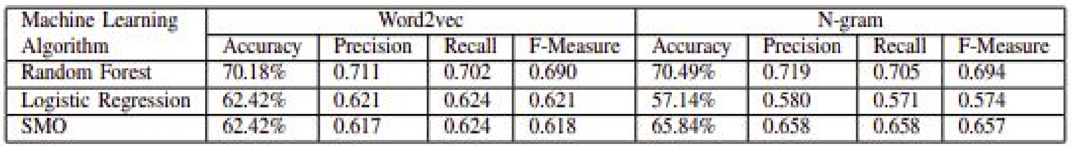


Figure 3.3 Flowchart of proposed architecture



Machine Learning Algorithm	Word2vec				N-gram			
	Accuracy	Precision	Recall	F-Measure	Accuracy	Precision	Recall	F-Measure
Random Forest	70.18%	0.711	0.702	0.690	70.49%	0.719	0.705	0.694
Logistic Regression	62.42%	0.621	0.624	0.621	57.14%	0.580	0.571	0.574
SMO	62.42%	0.617	0.624	0.618	65.84%	0.658	0.658	0.657

Table 3.2 Sentiment analysis results

CHAPTER 4

TECHNOLOGIES AND LIBRARIES USED

4.1 Keras

Keras is a common deep learning programming system that simplifies the development of deep learning applications. It uses TendFlow or Theano behind the scenes and has a standard, streamlined programming interface on top, rather than having all the features per se. Keras is a simple to use, extensible and flexible library that simplifies and encourages prototyping. It supports convolutionalary networks, recurrent networks and even the fusion of both.

Today there are countless deep-learning systems available, but Keras has proven better in some areas than other frameworks. Keras emphasises on the need for minimum user behaviour in common usage scenarios, even where the user commits a mistake, transparent and input is given. This enables the learning and usage of keras.

If you want to use the Keras models on any application, you need to deploy them on other systems that are pretty simple if you are using Keras. It also supports several backends and also facilitates backend portability, i.e. you can train and load with one backend.

4.2 Python

It is a high-level, object-oriented, dynamically typed scripting language that is interpreted. Python interpreters read one line of code at a time, translate it and then execute it in a low-level machine language (byte code). As a consequence, errors in run time are commonly observed.

It can be tricky to incorporate AI and ML algorithms into your software and is a time consuming process. It is important to have a structured and tested environment to enable faster development and quicker iterations.

Nowadays programmers are moving to a range of Python frameworks and libraries to implement these algorithms. Python provides a comprehensive collection of libraries for artificial intelligence and machine learning including but not limited to Keras, Tensorflow, Numpy, Scikit-Learn, Pandas, Seaborn, Pytorch etc.

4.3 Web Scraping - Scrapy

Scrapy is an open source and interactive framework to retrieve from websites the data you need. It is fast, powerful, easily extensible and portable. It provides you with all the tools you need to collect data from websites effectively, process it as you like, and store it in your desired structure and format.

In collecting data from sites, there is no single strategy, since the internet is diverse. Many ad hoc methods are taken and you will ultimately end up building your own scraping system if you start writing code for each small task you do. That framework is Scrapy.

4.4 Rosetta Sentiment Analyzer

Rosetta Sentiment Analyzer is a tool used in texts about corporations, individuals, and items to identify emotional hotspots. In Rosette, machine learning models are trained on tweets and feedback to identify strong positive and negative feelings in an overall document and against specific entities. Rosette applies the entity's extraction process to classify entities and decides the sentiment of each entity by referring the sentiment in the analysis to each entity.

4.5 Data Manipulation and Visualization tools

1. Pandas: Pandas is a software library in Python that focuses on quick and simple data handling and analysis. In specific, it includes high-level data structures (such as DataFrame and Sequence) and data techniques for numerical tables and time series data management and visualisation. It's built on top of NumPy with essential code paths written in C and is incredibly optimised for performance

2. NumPy: With efficient data structures, NumPy enriches the programming language Python, introducing multi-dimensional arrays and matrices. With matrices and arrays, these data structures guarantee efficient calculations. The implementation also focuses on enormous matrices and arrays, best known under the heading of "big data" Moreover, to work on these matrices and arrays, the module offers a wide library of high-level mathematical functions.

3. Sklearn: Scikit-learn is an open source Python machine learning library. The library offers specialised algorithms like Support Vector Machine, K-Nearest Neighbour, random forests and XGBoost. It's built on Numpy's base. In the kaggle competition, popular technological firms commonly use Scikit-learn. Scikit-learn assists with classification, model selection, dimensional reduction, regression, clustering, and pre-processing.

4. Matplotlib: Matplotlib is one of Python's most common data display packages. It is an interdisciplinary library that renders 2D plots in arrays of data. In Python, Matplotlib uses NumPy, the numerical extension of Python's math. Provides a Python GUI toolkit, such as WxPython Tkinter and PyQt for object-orienting API that lets integrate plots through applications. It can also be found in web application servers, Jupyter notebooks, Python and IPython shells.

4.6 Dataset

1. Historical Stock Price Data: Procured from Yahoo! Finance.Yahoo! Finance is a domain of the media and is part of Yahoo! Networks. Includes stock quotes, press releases, financial reports, and original programming, it offers financial news, data and commentary. It also provides some personal finance management online resources.

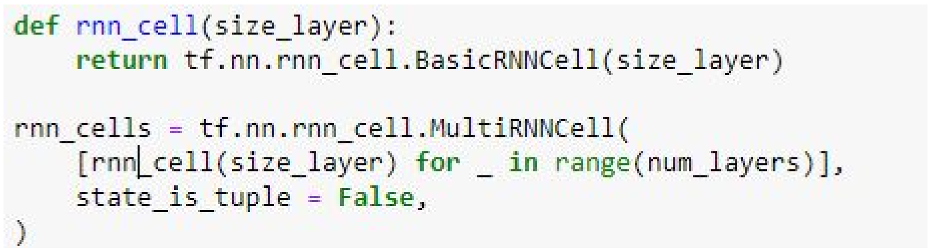
2. Company News Articles: Procured from reuters.com. Reuters news delivers trustworthy intelligence in an impartial way, with unparalleled coverage of news in more than 16 languages, and touching billions of people worldwide every day.

CHAPTER 5

CLASSICAL METHODOLOGY

5.1 Model Construction

5.1.1 Vanilla RNN



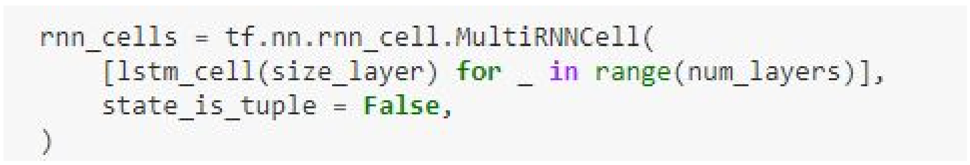
```
def rnn_cell(size_layer):
    return tf.nn.rnn_cell.BasicRNNCell(size_layer)

rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [rnn_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

rnn_cells contain a collection of layers, each layer is composed of rnn_cell with some number of basicRNNCell in each rnn_cell. Here num_layers=1 and size_layer=128.So a RNN is constructed with 1 layer and that layer contains an RNNCell with 128 RNN Units.

Here the rnn_cell is constructed using the inbuilt BasicRNNCell function of tensorflow library.

5.1.2 Vanilla LSTM



```
rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [lstm_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

Creates a collection of LSTM cells for each of the num_layers number of layers. Here num_layers=1, so it is a single layer structure in terms of layers of lstm cells. The function lstm_cell(size_layer) creates a single lstm cell with size_layer number of lstm basic blocks/memory units. The function is defined as follows:- Uses the inbuilt Tensorflow function LSTMCell for a lstm cell construction. Here the lstm cell is constructed with 128 units.

5.1.3 Vanilla GRU

Similar to LSTM model, just in place of using a LSTMCell function we use the GRUCell function of tensorflow

```
    def gru_cell(size_layer):
        return tf.nn.rnn_cell.GRUCell(size_layer)
```

size_layer =128 that is it has 128 GRU basic units taken here.

These cells are used to make a layer using the MultiRNNCell function as in above:-

```
rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [gru_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

Here also num_layers =1 and gru_cell function defined above is used for the contents of the layer

5.1.4 LSTM Bidirectional and Bidirectional RNN

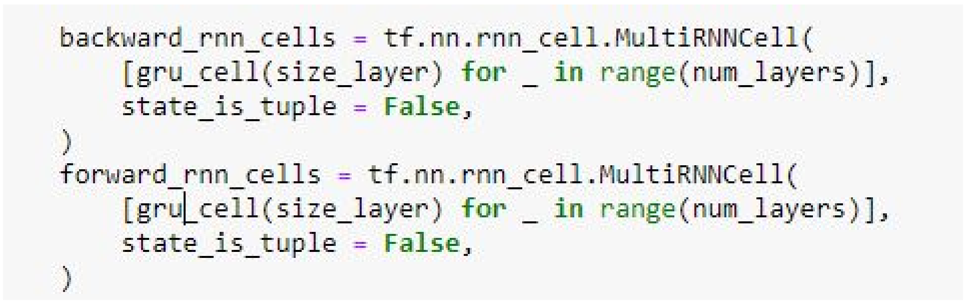
```
backward_rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [lstm_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
forward_rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [lstm_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

Creates a collection of LSTM cells each for forward path and backward path. Here num_layers =1, so it's a single layer structure for both the directions. Now apart from storing info from past data we also take info from the future data and dependencies generated by the LSTM path in backward_rnn_cells,thus improving our prediction.

The function lstm_cell(size_layer) creates a single lstm cell with size_layer number of lstm basic blocks/memory units. The function is defined as follows:- Uses the inbuilt Tensorflow function LSTMCell for a lstm cell construction. Here the lstm cell is constructed with 128 units.

The code for the Bidirectional RNN model is similar except for having half the number of units as the LSTM Bidirectional model.

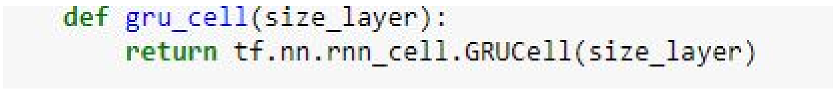
5.1.5 GRU Bidirectional



```
backward_rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [gru_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
forward_rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [gru_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

Like Bi-LSTM, this Creates a collection of GRU cells each for forward path and backward path. Here num_layers =1, so it's a single layer structure for both directions. Now apart from storing info from past data we also take info from the future data and dependencies generated by the GRU path in backward_rnn_cells,thus improving our prediction.

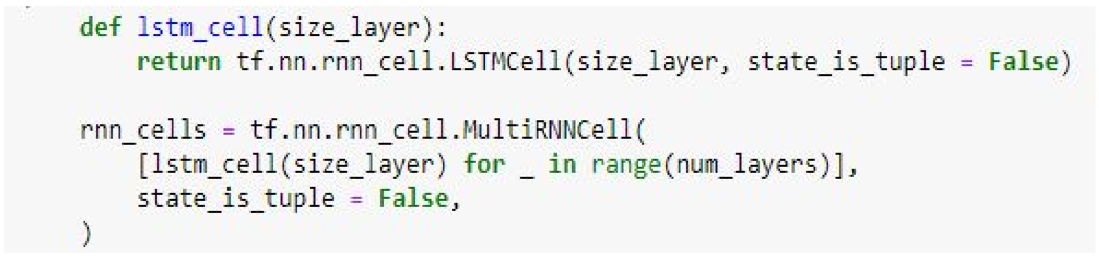
The function gru_cell(size_layer) creates a single lstm cell with size_layer number of lstm basic blocks/memory units .The function is defined as follows:-



```
def gru_cell(size_layer):
    return tf.nn.rnn_cell.GRUCell(size_layer)
```

Similar to the LSTM model, just in place of using a LSTMCell function we use the GRUCell function of tensorflow size_layer=128 that has 128 GRU basic units taken here.

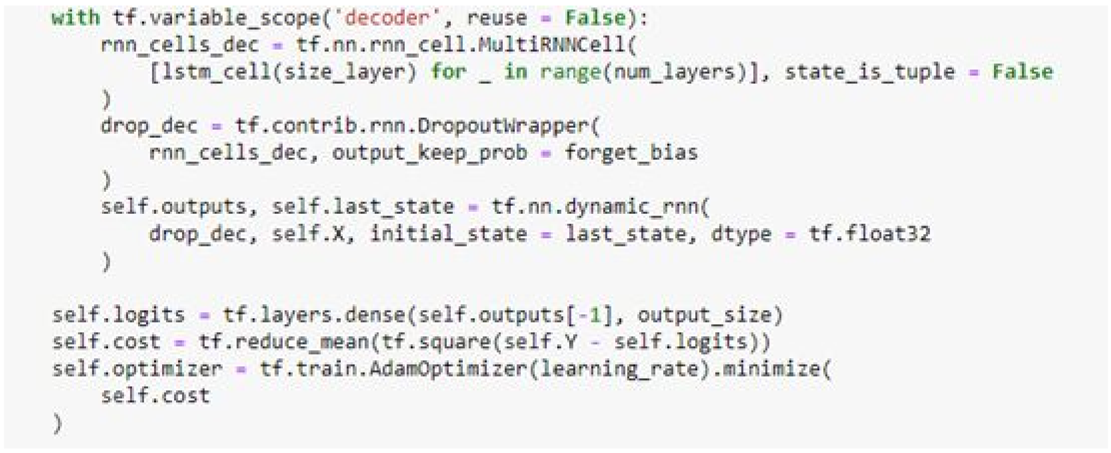
5.1.6 LSTM Sequence to Sequence



```
def lstm_cell(size_layer):
    return tf.nn.rnn_cell.LSTMCell(size_layer, state_is_tuple = False)

rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [lstm_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

rnn_cells is a layer of lstm cells. Here num_layers=1 and the size_layer is the number of basic LSTM blocks in each LSTM cell ,here =128. lstm_cell function uses the inbuilt LSTMCell model/function to generate the lstm cell with appropriate parameters.



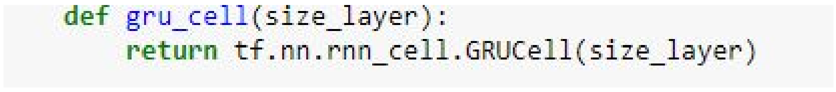
```
with tf.variable_scope('decoder', reuse = False):
    rnn_cells_dec = tf.nn.rnn_cell.MultiRNNCell(
        [lstm_cell(size_layer) for _ in range(num_layers)], state_is_tuple = False
    )
    drop_dec = tf.contrib.rnn.DropoutWrapper(
        rnn_cells_dec, output_keep_prob = forget_bias
    )
    self.outputs, self.last_state = tf.nn.dynamic_rnn(
        drop_dec, self.X, initial_state = last_state, dtype = tf.float32
    )

self.logits = tf.layers.dense(self.outputs[-1], output_size)
self.cost = tf.reduce_mean(tf.square(self.Y - self.logits))
self.optimizer = tf.train.AdamOptimizer(learning_rate).minimize(
    self.cost
)
```

This RNN layer serves as the output RNN layer of the sequence to sequence model. Here the last_state generated by the previous rnn layer containing the context info of the inputs is fed as the initial_state so that it reflects in the output and then it calculates the output series, here series of stock prices.

5.1.7 GRU Sequence to Sequence

Similar to the above sequence to sequence LSTM model, the only difference is in place of the LSTMCell, GRUCell given below is used



```
def gru_cell(size_layer):
    return tf.nn.rnn_cell.GRUCell(size_layer)
```

5.1.8 Sequence to Sequence CNN

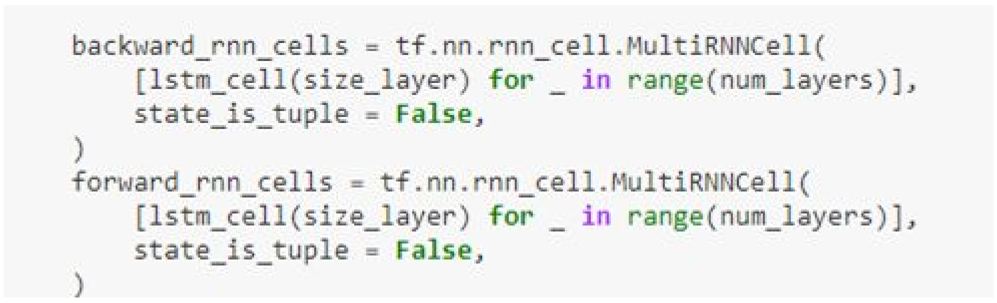
This involves Gated Linear Units and residual connections that were already in operation. It also applies separate attention to each decoder layer and shows that very little overhead is added by each attention. Multi-layered CNN can learn hierarchical characteristics with relatively close dependencies of lower layer learning, where higher layers learn long-range interactions.

5.1.9 Dilated Sequence to Sequence CNN

Dilated convolution is just a convolution for the input with specific deficiencies. Dilated setup is a means of exponentially and linearly accretion of receptive vision (global view) of the network. It uses it to this end mostly to incorporate awareness of the larger environment with less expense in applications.

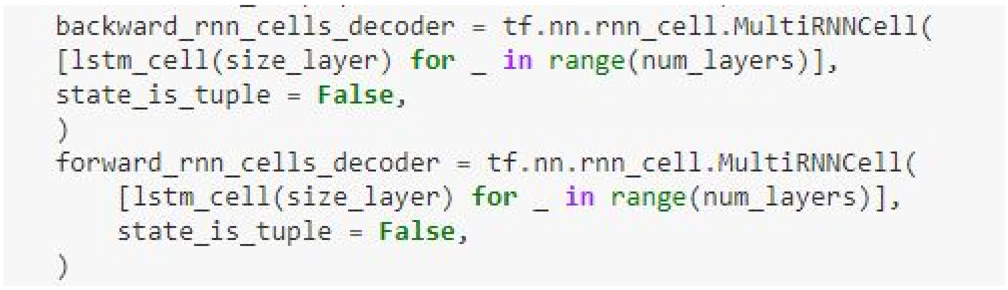
5.1.10 LSTM Bidirectional Sequence to Sequence

Couples the features of both sequence to sequence and Bidirectional models each containing the LSTM cells. Like GRU based one here also each side has an encoder and a decoder so that the input sequence can be of any of the 2 forms in consideration and can be input at both sides.



```
backward_rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [lstm_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
forward_rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [lstm_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

Constructed the encoder for both sides



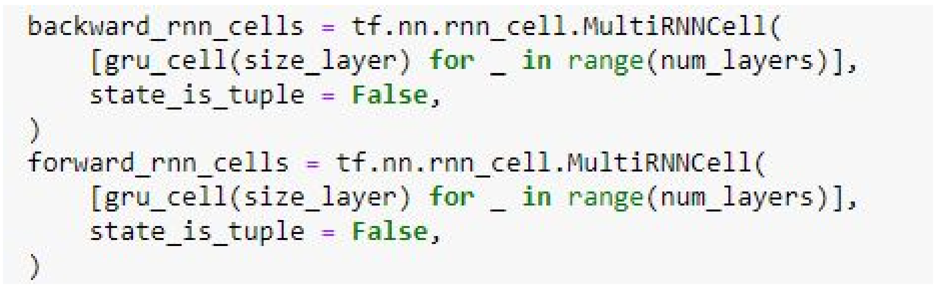
```
backward_rnn_cells_decoder = tf.nn.rnn_cell.MultiRNNCell(
[lstm_cell(size_layer) for _ in range(num_layers)],
state_is_tuple = False,
)
forward_rnn_cells_decoder = tf.nn.rnn_cell.MultiRNNCell(
    [lstm_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

Decoder model for both sides

Each MultiRNNCell Layer on both sides has a single layer tho that single layer contains 128 units of LSTM Cells.

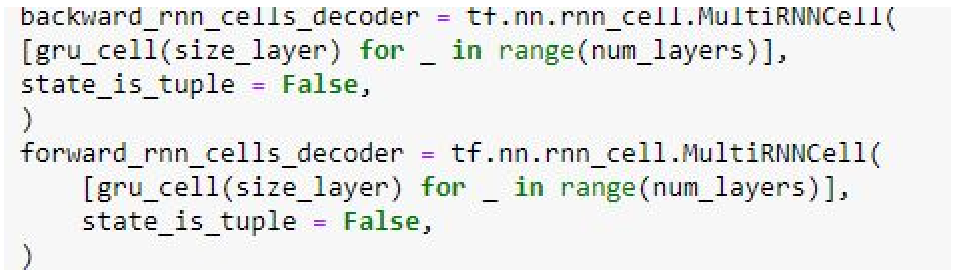
5.1.11 GRU Bidirectional Sequence to Sequence

Combines the features of Bidirectional and Sequence to Sequence models for the GRU Unit. Sequence can be now transformed from both Type 1 to Type 2 and vice versa i.e. both ends act as encoder and decoder.



```
backward_rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [gru_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
forward_rnn_cells = tf.nn.rnn_cell.MultiRNNCell(
    [gru_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

These 2 layers serve as the encoders for both the sides.



```
backward_rnn_cells_decoder = tf.nn.rnn_cell.MultiRNNCell(
[gru_cell(size_layer) for _ in range(num_layers)],
state_is_tuple = False,
)
forward_rnn_cells_decoder = tf.nn.rnn_cell.MultiRNNCell(
    [gru_cell(size_layer) for _ in range(num_layers)],
    state_is_tuple = False,
)
```

These 2 serve as the decoders at the respective sides.

Each MultiRNNCell Layer on both sides has a single layer tho that single layer contains 128 units of GRU Cells.

5.1.12 2-Path RNN, LSTM and GRU

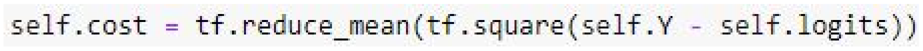
2-path networks organize and model RNN layers in a deep structure. They break the long sequential input into smaller chunks and iteratively implement intra and inter-chunk operations, where the input length can be made equal in each operation to the square root of the initial sequence length.

5.2 Model training, testing and simulations

Steps taken to train the model were:-

1) After normalization of data we take the 4the column of the dataset i.e. "Close" indicates the closing stock price data for each specific date.We will use only this stock price across all models to maintain uniformity.
2) Out of the 252 data points 222 are used for training and the most recent 30 days are used for testing during the train test split.
3) The 222 data points are divided into batch sizes of 5 stored in the variable batch_X and sent as a placeholder for X in the model constructed using the feed_dict object. batch_Y is also formed of 5 data points but the data starts from the position 1 ahead of the starting position of that of batch_X. These are

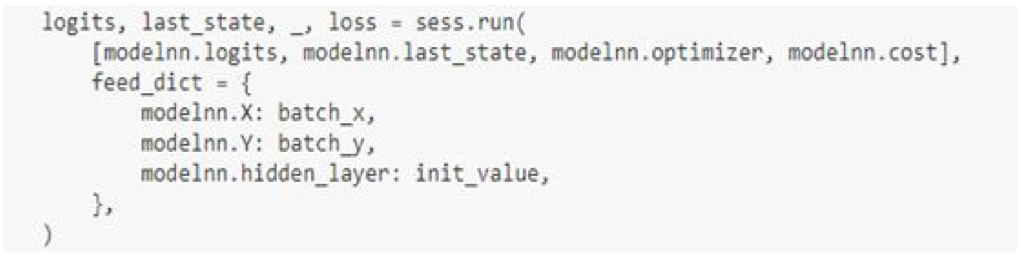
also sent to the model object and are used in the calculation of loss function.



```
self.cost = tf.reduce_mean(tf.square(self.Y - self.logits))
```

Where, cost = loss function (mean of squared errors), Y = batch_Y(containing real values), logits = containing the predicted values using the batch_X.

4) The model is run on the batch_X and batch_Y values and other inputs given and it gives 4 outputs variables/vectors



```
logits, last_state, _, loss = sess.run(
    [modelnn.logits, modelnn.last_state, modelnn.optimizer, modelnn.cost],
    feed_dict = {
        modelnn.X: batch_x,
        modelnn.Y: batch_y,
        modelnn.hidden_layer: init_value,
    },
)
```

Where, modelnn.logits = containing the predicted values using the batch_X, modelnn.last_state = containing outputs for each data point from last layer only, modelnn.cost = containing the loss function calculated above

5) The outputs obtained in the last_state are set to the initial state variable and supplied to the model for the next 5 data points as it contains the memory/important information obtained from the previous 5 data points training.

Steps taken to test the model were:-

1) From the total 252 data points final 30 points are taken for testing purposes.
2) Again here the test data is input in batch size of 5, the answer calculated is stored in an array “output_predict”.

```python
for i in range(future_day):
    o = output_predict[-future_day - timestamp + i:-future_day + i]
    out_logits, last_state = sess.run(
        [modelnn.logits, modelnn.last_state],
        feed_dict = {
            modelnn.X: np.expand_dims(o, axis = 0),
            modelnn.backward_hidden_layer: init_value_backward,
            modelnn.forward_hidden_layer: init_value_forward,
        },
    )
```

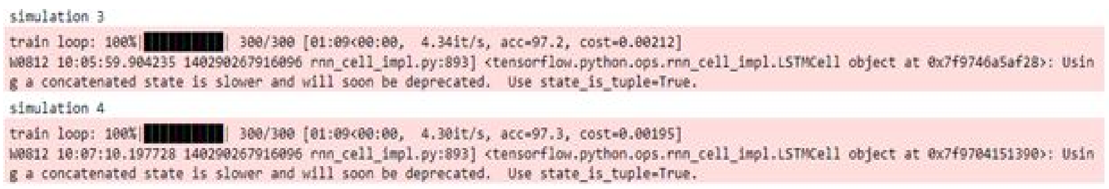
3) The input in the model is given through the standard feed_dict object like done in training to fill the placeholder.
4) Inverse Normalisation is applied using the “minimax.inverse_transform” function to get the data according to the input scale, not the standard Gaussian scale.

```python
output_predict = minmax.inverse_transform(output_predict)
```

5) The output is now returned and this is later used to compare with the original output and calculate the accuracy accordingly.

Steps taken to run simulations were:-

We ran 10 simulations of the model to ensure the perfect training and testing accuracies. Each simulation trains over the training data for epoch(10) times i.e. we train over the data 100 times and we calculate the accuracy and cost side by side in each simulation using the functions of the tqdm library. Here’s one of the simulation runs, this shows the run of simulation 3 & 4 and shows their respective costs and accuracies.



```
simulation 3
train loop: 100%|██████████| 300/300 [01:09<00:00,  4.34it/s, acc=97.2, cost=0.00212]
W0812 10:05:59.904235 140290267916096 rnn_cell_impl.py:893] <tensorflow.python.ops.rnn_cell_impl.LSTMCell object at 0x7f9746a5af28>: Usin
g a concatenated state is slower and will soon be deprecated.  Use state_is_tuple=True.
simulation 4
train loop: 100%|██████████| 300/300 [01:09<00:00,  4.30it/s, acc=97.3, cost=0.00195]
W0812 10:07:10.197728 140290267916096 rnn_cell_impl.py:893] <tensorflow.python.ops.rnn_cell_impl.LSTMCell object at 0x7f9704151390>: Usin
g a concatenated state is slower and will soon be deprecated.  Use state_is_tuple=True.
```

Figure 5.1 Example simulation run

5.3 Graph Plots and Results

A 'results' vector contains the result of all the 10 training/forecasts for the 10 epochs. We plot each of the forecasts on the same graph and we also plot the actual true trend plot on the same graph colored black. This graph can be used to check where the model has undergone overfitting and where its still underfit and thus also determine a perfect forecast which most closely matches the true trend.

We have the two following evaluation metrics

1. Volatility Accuracy:- Calculated by taking the percentage of predicted prices that are predicted to swing in the correct direction i.e. the same direction as the original stock price for that day .
2. Prediction Accuracy:- Calculated by the formula of relative error as we have both the predicted and the original values.

5.3.1 Graph Plots

Here the black line indicates the true trend and others are forecast trends.

The average accuracy mentioned on the graphs is the volatility prediction accuracy.

1) RNN

- Vanilla RNN: Accuracy on Stock Price = 91.46%, on Volatility=53%

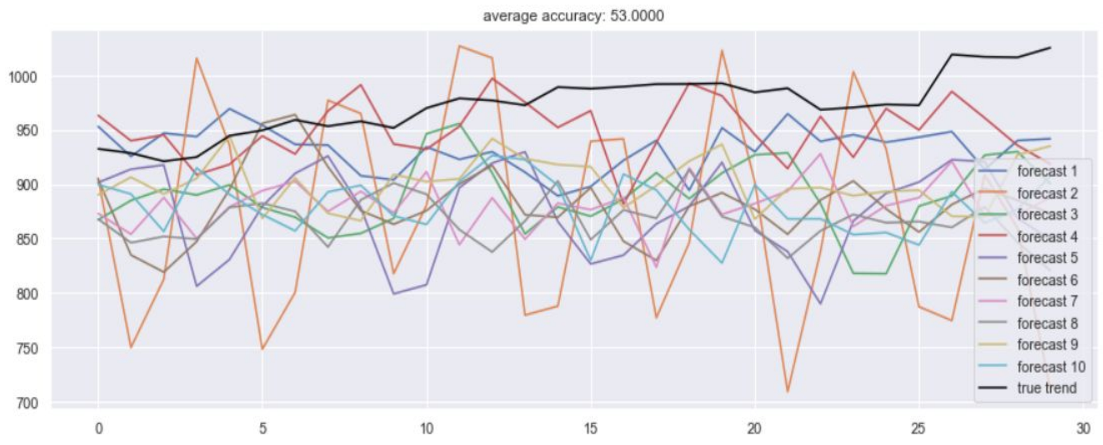


Figure 5.2 Vanilla RNN Forecast Graph

- Bidirectional RNN: Accuracy on Stock Price = 88.99%, on Volatility=52.43%

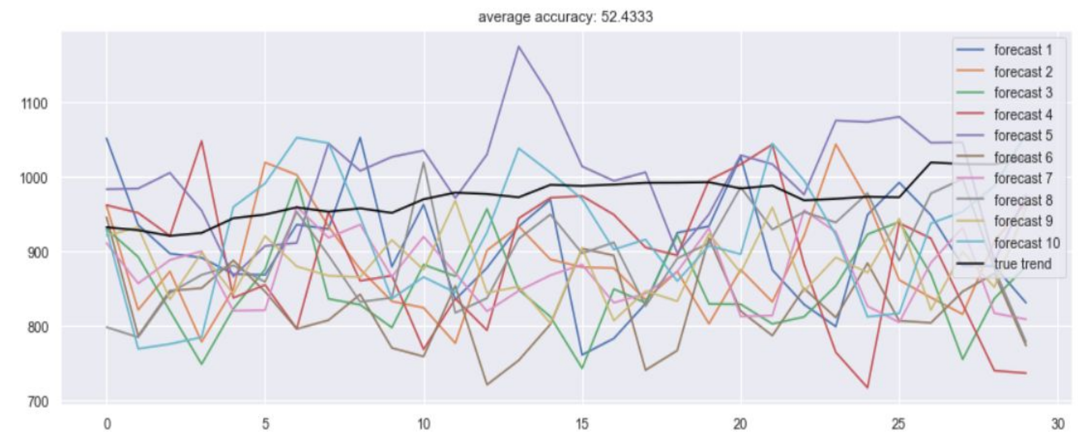


Figure 5.3 Bidirectional RNN Forecast Graph

- 2-Path RNN: Accuracy on Stock Price = 91.54%, on Volatility=53.9%

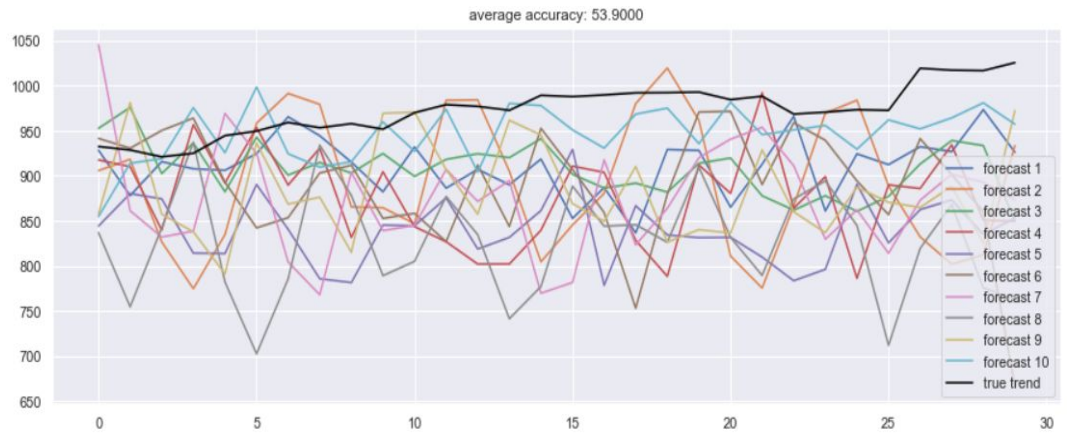


Figure 5.4 2-Path RNN Forecast Graph

2) LSTM

- Vanilla LSTM: Accuracy on Stock Price = 94.37%, on Volatility=55.55%

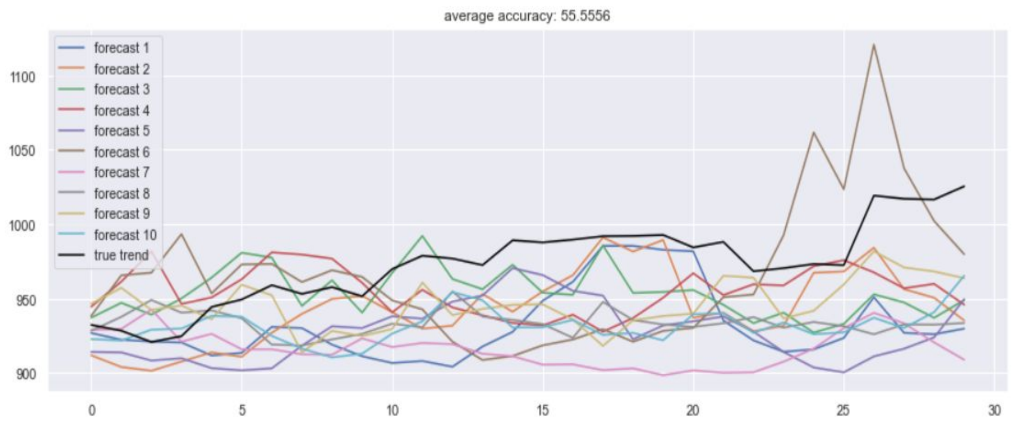


Figure 5.5 Vanilla LSTM Forecast Graph

- LSTM Bidirectional: Accuracy on Stock Price = 94.38%, on Volatility=51.7%

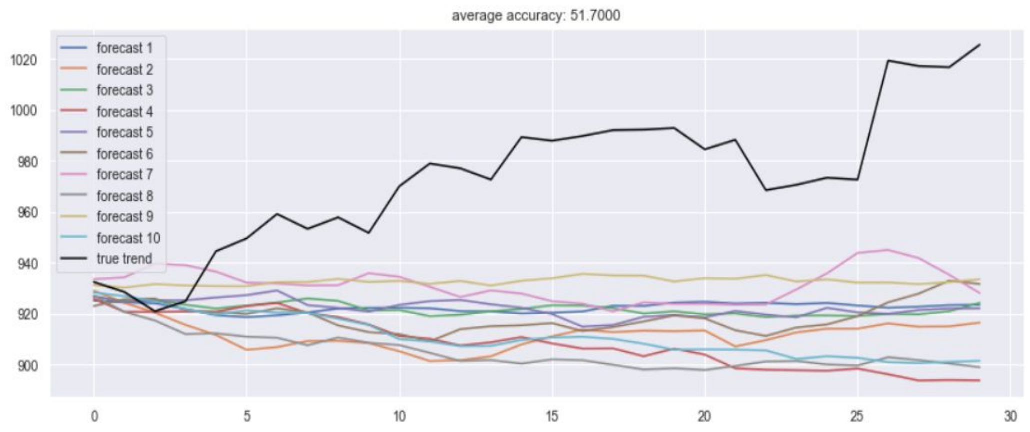


Figure 5.6 LSTM Bidirectional Forecast Graph

- LSTM 2-Path: Accuracy on Stock Price = 94.63%, on Volatility=52.2%

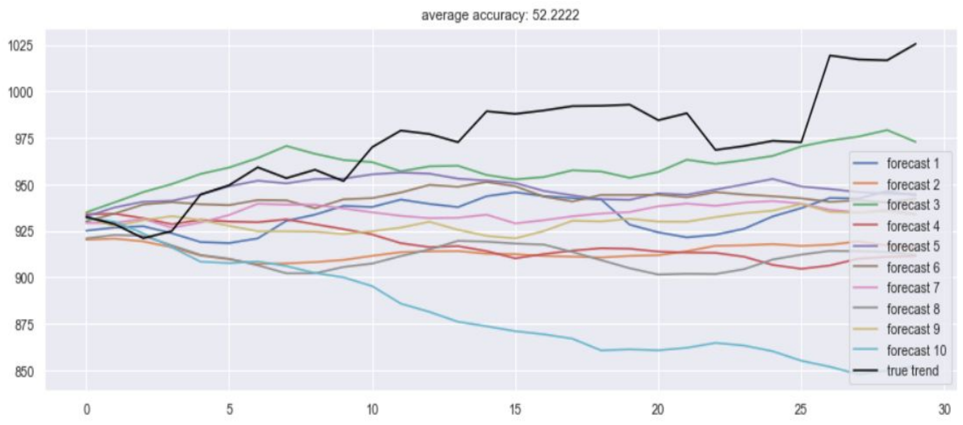


Figure 5.7 LSTM 2-Path Forecast Graph

- LSTM Sequence to Sequence: Accuracy on Stock Price = 94.98%, on Volatility=51.3%

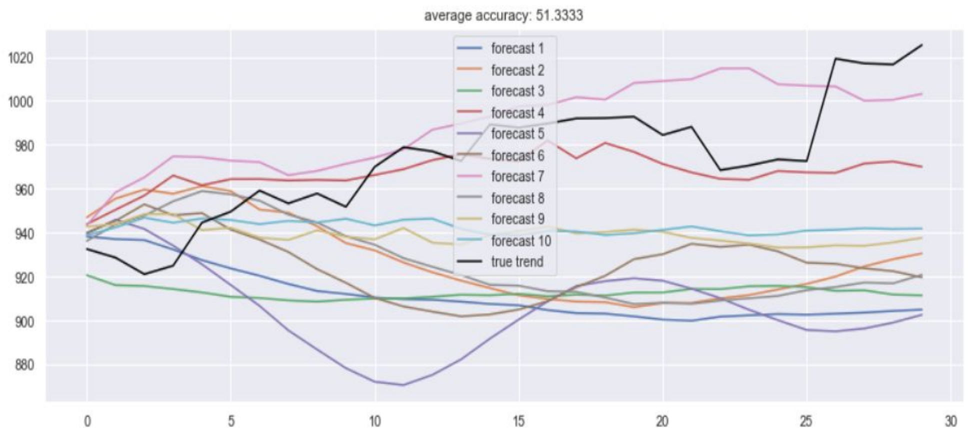


Figure 5.8 LSTM Sequence to Sequence Forecast Graph

- LSTM Bidirectional Sequence to Sequence: Accuracy on Stock Price = 94.51%, on Volatility=53.53%

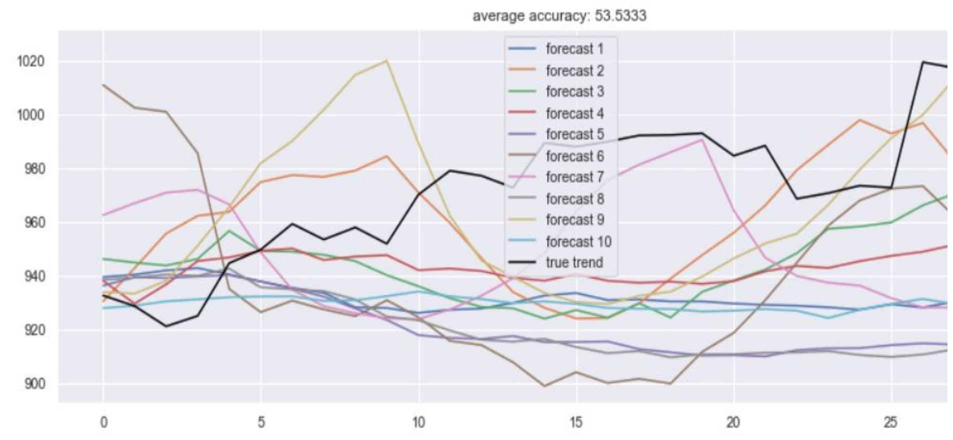


Figure 5.9 LSTM Bidirectional Sequence to Sequence Forecast Graph

3) GRU

- Vanilla GRU: Accuracy on Stock Price = 94.63%, on Volatility=54%

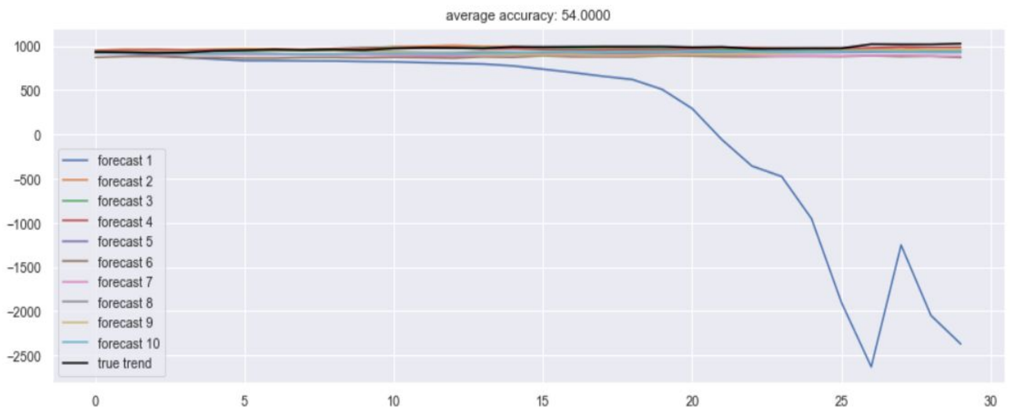


Figure 5.10 Vanilla GRU Forecast Graph

- GRU Bidirectional: Accuracy on Stock Price = 92.56%, on Volatility=56.4%

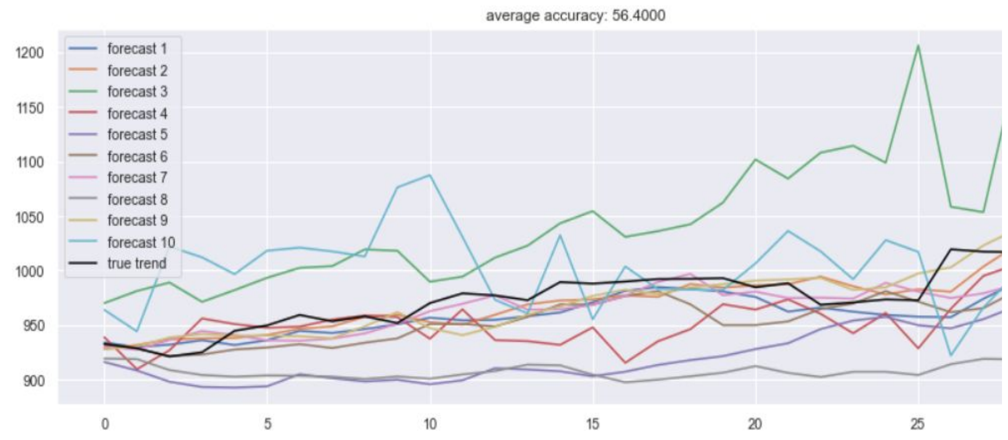


Figure 5.11 GRU Bidirectional Forecast Graph

- GRU 2-Path: Accuracy on Stock Price = 93.21%, on Volatility=51.1%

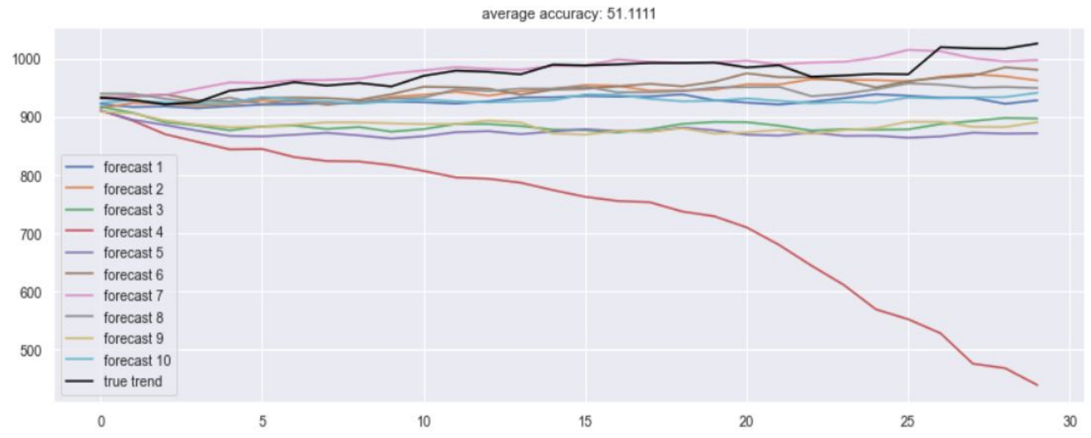


Figure 5.12 GRU 2-Path Forecast Graph

- GRU Sequence to Sequence: Accuracy on Stock Price = 90.88%, on Volatility=53.3%

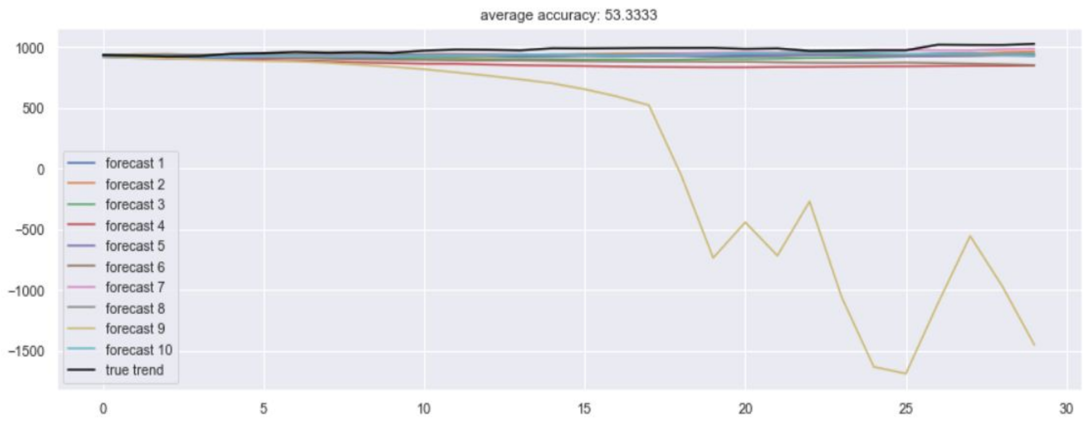


Figure 5.13 GRU Sequence to Sequence Forecast Graph

- GRU Bidirectional Sequence to Sequence: Accuracy on Stock Price = 67.99%, on Volatility=56%

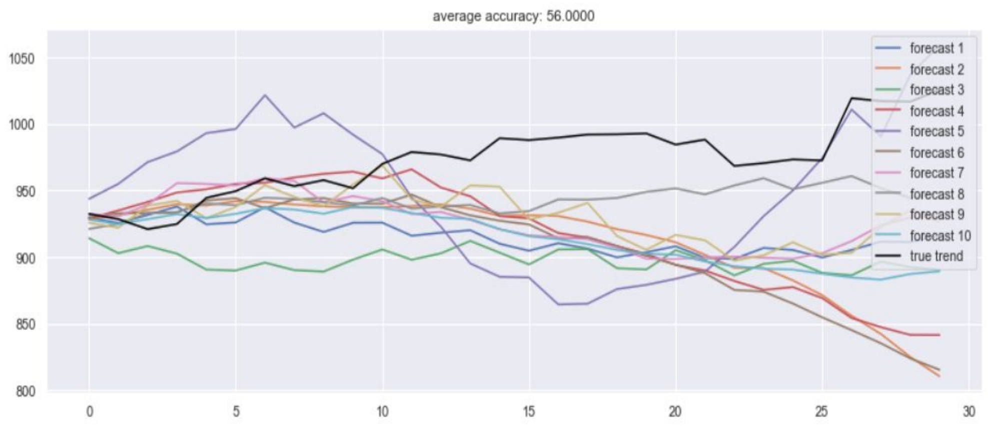


Figure 5.14 GRU Bidirectional Sequence to Sequence Forecast Graph

4) CNN

- Sequence to Sequence CNN: Stock Price Accuracy 90.73

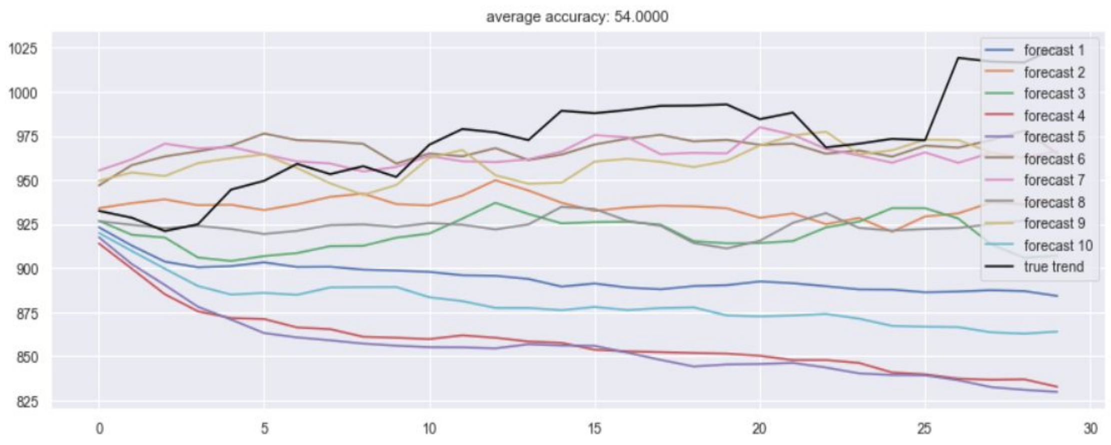


Figure 5.15 Sequence to Sequence CNN Forecast Graph

- Dilated Sequence to Sequence CNN: Stock Price Accuracy 95.86

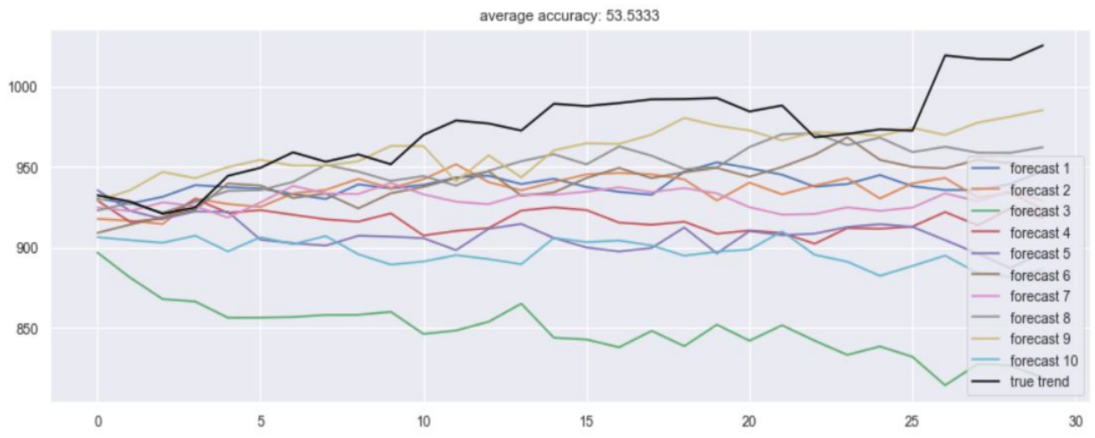


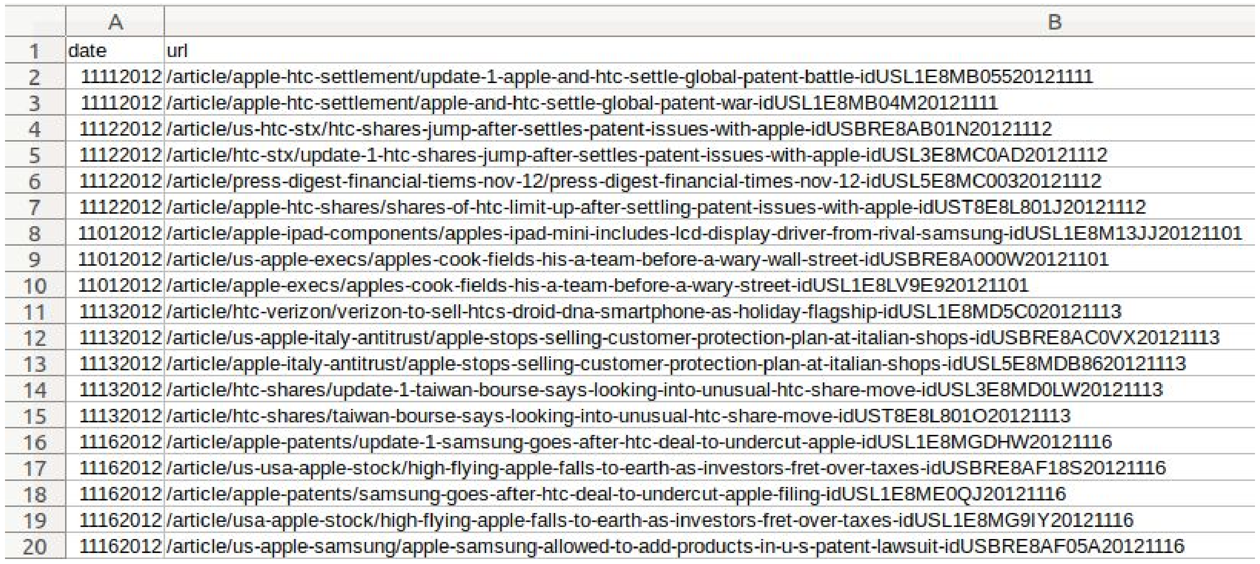
Figure 5.16 Dilated Sequence to Sequence CNN Forecast Graph

CHAPTER 6

SENTIMENT ANALYSIS MODULE

6.1 News Article Generation

Web Scraper made in Python is used to get the list of links of news articles. This uses the python module ‘scrapy’. We use the website ‘reuters.com’ for this purpose. For each day we have a corresponding page on the website which lists all the news articles of the given company on that day. We use scrapy to extract the links of the news articles from the html anchor tags on the page. Then we output it in a csv file with two columns, the date and url of the news articles.



	A	B
1	date	url
2	11112012	/article/apple-htc-settlement/update-1-apple-and-htc-settle-global-patent-battle-idUSL1E8MB05520121111
3	11112012	/article/apple-htc-settlement/apple-and-htc-settle-global-patent-war-idUSL1E8MB04M20121111
4	11122012	/article/us-htc-stx/htc-shares-jump-after-settles-patent-issues-with-apple-idUSBRE8AB01N20121112
5	11122012	/article/htc-stx/update-1-htc-shares-jump-after-settles-patent-issues-with-apple-idUSL3E8MC0AD20121112
6	11122012	/article/press-digest-financial-tiems-nov-12/press-digest-financial-times-nov-12-idUSL5E8MC00320121112
7	11122012	/article/apple-htc-shares/shares-of-htc-limit-up-after-settling-patent-issues-with-apple-idUST8E8L801J20121112
8	11012012	/article/apple-ipad-components/apples-ipad-mini-includes-lcd-display-driver-from-rival-samsung-idUSL1E8M13JJ20121101
9	11012012	/article/us-apple-execs/apples-cook-fields-his-a-team-before-a-wary-wall-street-idUSBRE8A000W20121101
10	11012012	/article/apple-execs/apples-cook-fields-his-a-team-before-a-wary-street-idUSL1E8LV9E920121101
11	11132012	/article/htc-verizon/verizon-to-sell-htcs-droid-dna-smartphone-as-holiday-flagship-idUSL1E8MD5C020121113
12	11132012	/article/us-apple-italy-antitrust/apple-stops-selling-customer-protection-plan-at-italian-shops-idUSBRE8AC0VX20121113
13	11132012	/article/apple-italy-antitrust/apple-stops-selling-customer-protection-plan-at-italian-shops-idUSL5E8MDB8620121113
14	11132012	/article/htc-shares/update-1-taiwan-bourse-says-looking-into-unusual-htc-share-move-idUSL3E8MD0LW20121113
15	11132012	/article/htc-shares/taiwan-bourse-says-looking-into-unusual-htc-share-move-idUST8E8L801O20121113
16	11162012	/article/apple-patents/update-1-samsung-goes-after-htc-deal-to-undercut-apple-idUSL1E8MGDHW20121116
17	11162012	/article/us-usa-apple-stock/high-flying-apple-falls-to-earth-as-investors-fret-over-taxes-idUSBRE8AF18S20121116
18	11162012	/article/apple-patents/samsung-goes-after-htc-deal-to-undercut-apple-filing-idUSL1E8ME0QJ20121116
19	11162012	/article/usa-apple-stock/high-flying-apple-falls-to-earth-as-investors-fret-over-taxes-idUSL1E8MG9IY20121116
20	11162012	/article/us-apple-samsung/apple-samsung-allowed-to-add-products-in-u-s-patent-lawsuit-idUSBRE8AF05A20121116

Table 6.1 News Article Generation

6.2 Sentiment Score Generation

The output csv file from Step A is used as an input in this step. Here we use the 'Rosette' API for text analysis for the contents for each article outputting a confidence label corresponding to each article.

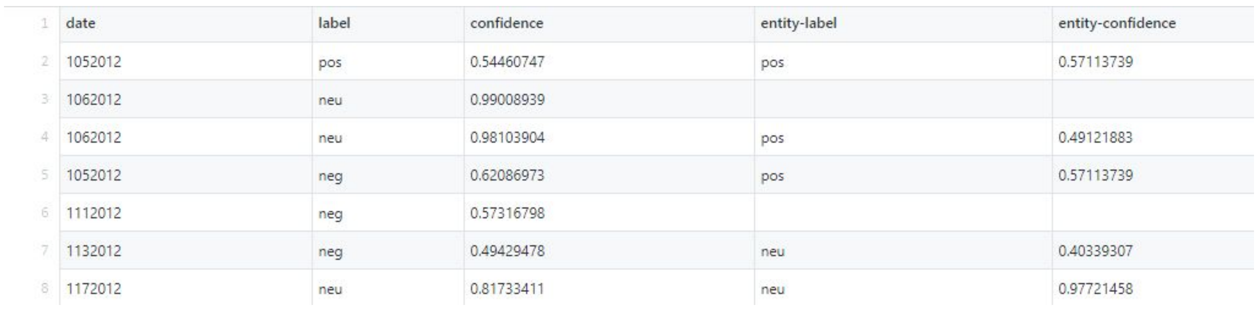
The confidence column presents the probability/confidence with which the label is decided for that specific date ie. label defines the attitude/emotion of the whole article,while entity_label defines the emotion towards a specific entity here in this case is the company name. Similarly confidence is the probability for the label while entity_confidence is probability for entity_label

This file outputs a csv file which contains the columns date, label,confidence,entity-label and entity-confidence. The column label can take 3 values:-

1. pos:-expressing the article is positive for company and its share
2. neg:- expressing the article is negative for company and its share
3. neu:- expressing the article is neutral for company and its share

The column confidence can range from -1: very negative outlook to 1: very positive outlook.

Below is a subset of sentiment scores generated for Apple news articles:-



date	label	confidence	entity-label	entity-confidence
1052012	pos	0.54460747	pos	0.57113739
1062012	neu	0.99008939		
1062012	neu	0.98103904	pos	0.49121883
1052012	neg	0.62086973	pos	0.57113739
1112012	neg	0.57316798		
1132012	neg	0.49429478	neu	0.40339307
1172012	neu	0.81733411	neu	0.97721458

Table 6.2 Sentiment Score Generation

6.3 Bullishness

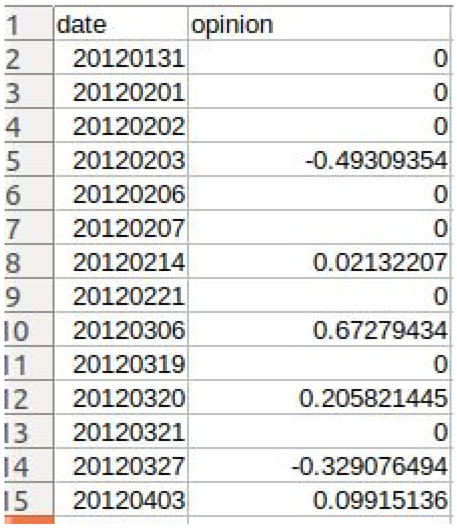
Here we calculate Bullishness and sort it in chronological order.

Using the sentiment score calculated for each article in the previous step, We calculate the bullishness score[17] of the selected stock on a per day basis. Bullishness for each day is calculated as:

$$Bullishness\ score = \frac{sum\ of\ confidence\ scores}{Number\ of\ articles}$$

Bullishness score represents the average perspective of the particular stock on the particular date.

In the next file, we arrange these scores in a chronological order.

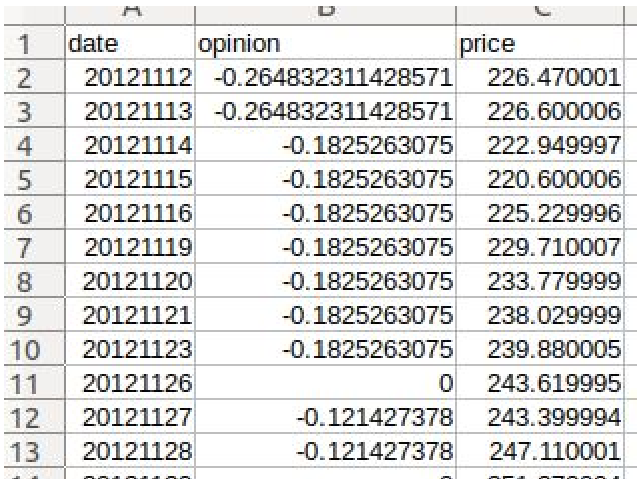


1	date	opinion
2	20120131	0
3	20120201	0
4	20120202	0
5	20120203	-0.49309354
6	20120206	0
7	20120207	0
8	20120214	0.02132207
9	20120221	0
10	20120306	0.67279434
11	20120319	0
12	20120320	0.205821445
13	20120321	0
14	20120327	-0.329076494
15	20120403	0.09915136

Table 6.3 Bullishness Score sorted in chronological order

6.4 Combining Sentiment Analysis with Stock Prices

For each day, combine the sentiment analysis of that day with the corresponding stock price. In case, the sentiment score of a day is not available, mark the sentiment score as the score of the latest previous day available. The output file after this step contains sentiment score and stock price for each day.



	A	B	C
1	date	opinion	price
2	20121112	-0.264832311428571	226.470001
3	20121113	-0.264832311428571	226.600006
4	20121114	-0.1825263075	222.949997
5	20121115	-0.1825263075	220.600006
6	20121116	-0.1825263075	225.229996
7	20121119	-0.1825263075	229.710007
8	20121120	-0.1825263075	233.779999
9	20121121	-0.1825263075	238.029999
10	20121123	-0.1825263075	239.880005
11	20121126	0	243.619995
12	20121127	-0.121427378	243.399994
13	20121128	-0.121427378	247.110001

Table 6.4 Sentiment Analysis output csv file

6.5 Deep Learning Model

For training our model, the sentiment score goes z score normalization followed by min-max normalization[18].

$$\boxed{z = \frac{x - \mu}{\sigma}} \quad \boxed{x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}}$$

Then, we construct a 2 layer LSTM followed by 2 Dense layers. Input of which will be previous k days stock price and its corresponding sentiment score where k is the window size. The output is the prediction of stock price.

```
model = Sequential()

model.add(LSTM(
    neurons[0],
    input_shape =(layers[1],layers[0]),
    return_sequences=True))
model.add(Dropout(d))

model.add(LSTM(
    neurons[1],
    input_shape=(layers[1],layers[0]),
    return_sequences=False))
model.add(Dropout(d))

model.add(Dense(neurons[2], activation='relu', kernel_initializer='uniform'))
model.add(Dense(neurons[3], activation='linear', kernel_initializer='uniform'))
```

Hyperparameters of this model are as follows

- batch_size=128
- epochs=200
- Train: test split- 80:20
- validation_split=0, verbose=0

Data set properties are

- 4319 news articles for AAPL stock
- 1261 dates considered

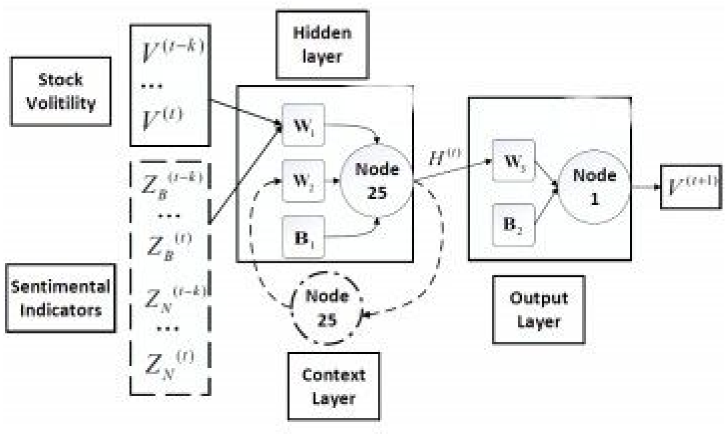


Figure 6.1 Deep learning model with sentiment indicators

6.6 Results

We used RMSProp optimizer[19] to minimize mean square error. Our objective is to maximise stock volatility prediction accuracy.

Accuracy and the best k for RNN+EMM and RNN					
	AAPL	AMZN	FB	GOOG	MSFT
LSTM	0.6129	0.5968	0.5484	0.5968	0.5726
LSTM+SI	0.621	0.6129	0.6048	0.6371	0.621

k(LSTM)	4	7	6	7	6
k(LSTM+SI)	4	9	7	7	5

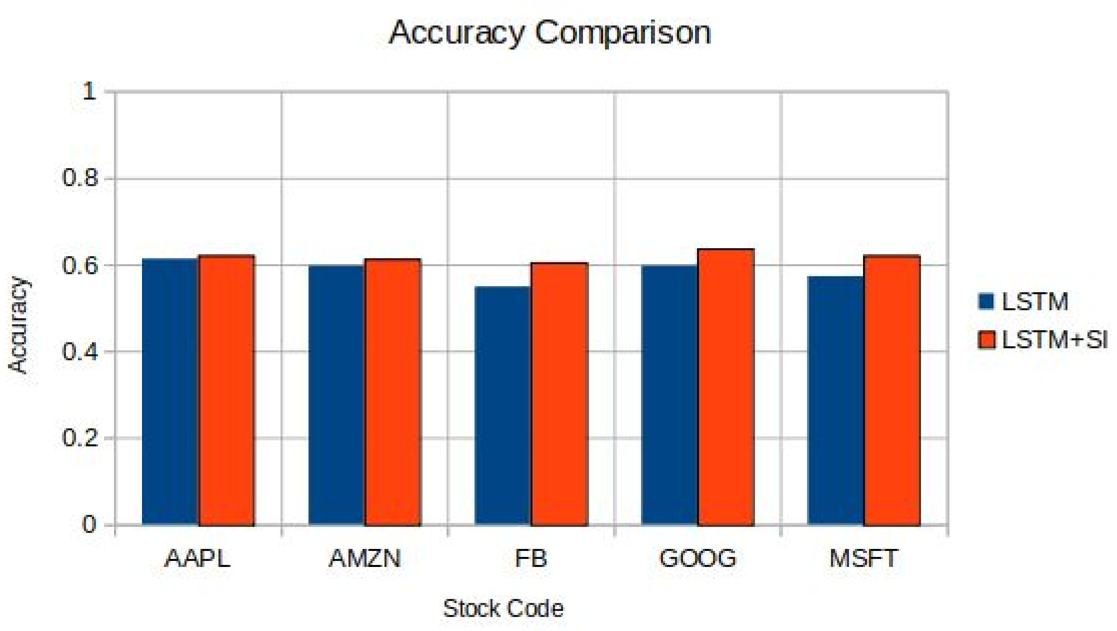


Table 6.5 Sentiment Analysis Results

Here we can see that accuracy is best increased in Facebook stock by 0.056 to achieve an overall accuracy of 0.6048, while the minimum increase in accuracy is shown in Apple stock of 0.0082. Average increase in accuracy among these 5 stocks is 0.03386.

After completing the prediction, we can clearly observe that the results using sentiment analysis offer marginally better results as compared to predicting without them. Hence, we conclude that analyzing sentiments in order to predict the volatility in the stock market is not worth the effort.

CHAPTER 7

SUMMARY AND CONCLUSIONS

We implemented 15 classical deep learning models including RNNs, LSTMs, GRUs and CNNs along with their variations and compared them with two distinct stock price prediction targets: absolute stock price and volatility. In a bid to further increase the accuracy of our predictions we incorporated sentiment analysis using news articles into our model as an input.

7.1 Limitations and Future Scope

The drawbacks and restrictions were as follows in the execution of this project:

- Lack of historical news articles: We were not able to find any news website available for getting historical news articles on a particular company due to which, we had to stick to the biggest known companies for our research.
- Lack of computational resources: The models should be ideally trained using a GPU for lower training time and faster training iterations but due to limited resources we used CPU for training the models.
- Limited number of calls allowed to Rosette API: Under the free version of Rosette API, we were allowed to make only 500 calls largely limiting our power to process all our articles in one go. Therefore we had to spread this part to multiple days to process sentiment analysis of all the articles.

Scope of further research include:

- Building the sentiment analysis module from scratch instead of outsourcing it to Rosette.
- Merging multiple social media sources for sentiment analysis.
- Observing a cut-off sentiment score to make a prediction for stronger results.

- Optimizing the current model's architecture, hyperparameters, etc.

Our future efforts will concentrate on the established shortcomings and on looking into how our stock market prices and volatility projections can be further enhanced.

7.2 Concluding Discussion

In this report, we have tried to predict stock market price and volatility. We created 15 classical deep learning models. On analyzing the prediction results from the classical deep learning models we found that all the models performed almost equally for volatility prediction but LSTMs outperform the others for predicting the stock market price. We try to incorporate sentiment analysis in our LSTM model to further increase volatility prediction. We observed that using sentiment analysis offered only a marginal increase in accuracy and hence concluded that given the time and effort required to incorporate sentiment analysis into the models it is not worth the effort. Our analysis outlines that there is still scope for future research work.

REFERENCES

[1] Frank Z. Xing, Erik Cambria, and Roy E. Welsch. 2018. Intelligent Bayesian Asset Allocation via Market Sentiment Views. IEEE Computational Intelligence Magazine (2018).

[2] White, "Economic prediction using neural networks: the case of IBM daily stock returns," IEEE 1988 International Conference on Neural Networks, San Diego, CA, USA, 1988, pp. 451-458 vol.2, doi: 10.1109/ICNN.1988.23959.

[3] Yauheniya Shynkevich, T.M. McGinnity, Sonya Coleman, Ammar Belatreche, Predicting Stock Price Movements Based on Different Categories of News Articles, 2015 IEEE Symposium Series on Computational Intelligence

[4] Malkiel B.G. (1989) Efficient Market Hypothesis. In: Eatwell J., Milgate M., Newman P. (eds) Finance. The New Palgrave. Palgrave Macmillan, London. https://doi.org/10.1007/978-1-349-20213-3_13

[5] Gang Wang, Tianyi Wang, Bolun Wang, Divya Sambasivan, Zengbin Zhang, Haitao Zheng, and Ben Y Zhao. 2015. Crowds on wall street: Extracting value from collaborative investing platforms. In Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing. ACM, 17–30

[6] Gu, Yanlei & Shibukawa, Takuya & Kondo, Yohei & Nagao, Shintaro & Kamijo, Shunsuke. (2020). Prediction of Stock Performance Using Deep Neural Networks. Applied Sciences. 10. 8142. 10.3390/app10228142.

[7] Zhou, Sijie. (2021). A Stock Prediction Method Based on LSTM. 10.1007/978-3-030-63784-2_24.
[8] Teoh, T.-T & Lim, W. & Koh, K. & Soh, J. & Tan, T. & Liu, S.Y. & Nguwi, Y.-Y. (2019). From Technical Analysis to Text Analytics: Stock and Index Prediction with GRU. 496-500. 10.1109/CIS-RAM47153.2019.9095772.

[9] Zhang, Rui & Wu, Zi'ang & Wang, Siqi. (2020). Prediction of Stock Based on Convolution Neural Network. 3175-3178. 10.1109/CCDC49329.2020.9164222.

[10] Jia, Mingzhu & Huang, Jian & Pang, Lihua & Zhao, Qian. (2019). Analysis and Research on Stock Price of LSTM and Bidirectional LSTM Neural Network. 10.2991/iccia-19.2019.72.

[11] Chiewhawan, Tanawat & Vateekul, Peerapon. (2020). Stock Return Prediction Using Dual-Stage Attention Model with Stock Relation Inference. 10.1007/978-3-030-41964-6_42.

[12] Mootha, Siddartha & Sridhar, Sashank & Seetharaman, Rahul & Gopalan, Chitrakala. (2020). Stock Price Prediction using Bi-Directional LSTM based Sequence to Sequence Modeling and Multitask Learning. 10.1109/UEMCON51285.2020.9298066.

[13] Song, Yue-Gang & Zhou, Yu-Long & Han, Ren-Jie. (2018). Neural networks for stock price prediction.

[14] Liu, Jialin & Chao, Fei & Lin, Yu-Chen & Lin, Chih-Min. (2019). Stock Prices Prediction using Deep Learning Models.

[15] Joshi, Kalyani & N, Bharathi & Rao, Jyothi. (2016). Stock Trend Prediction Using News Sentiment Analysis. International Journal of Computer Science and Information Technology. 8. 67-76. 10.5121/ijcsit.2016.8306.

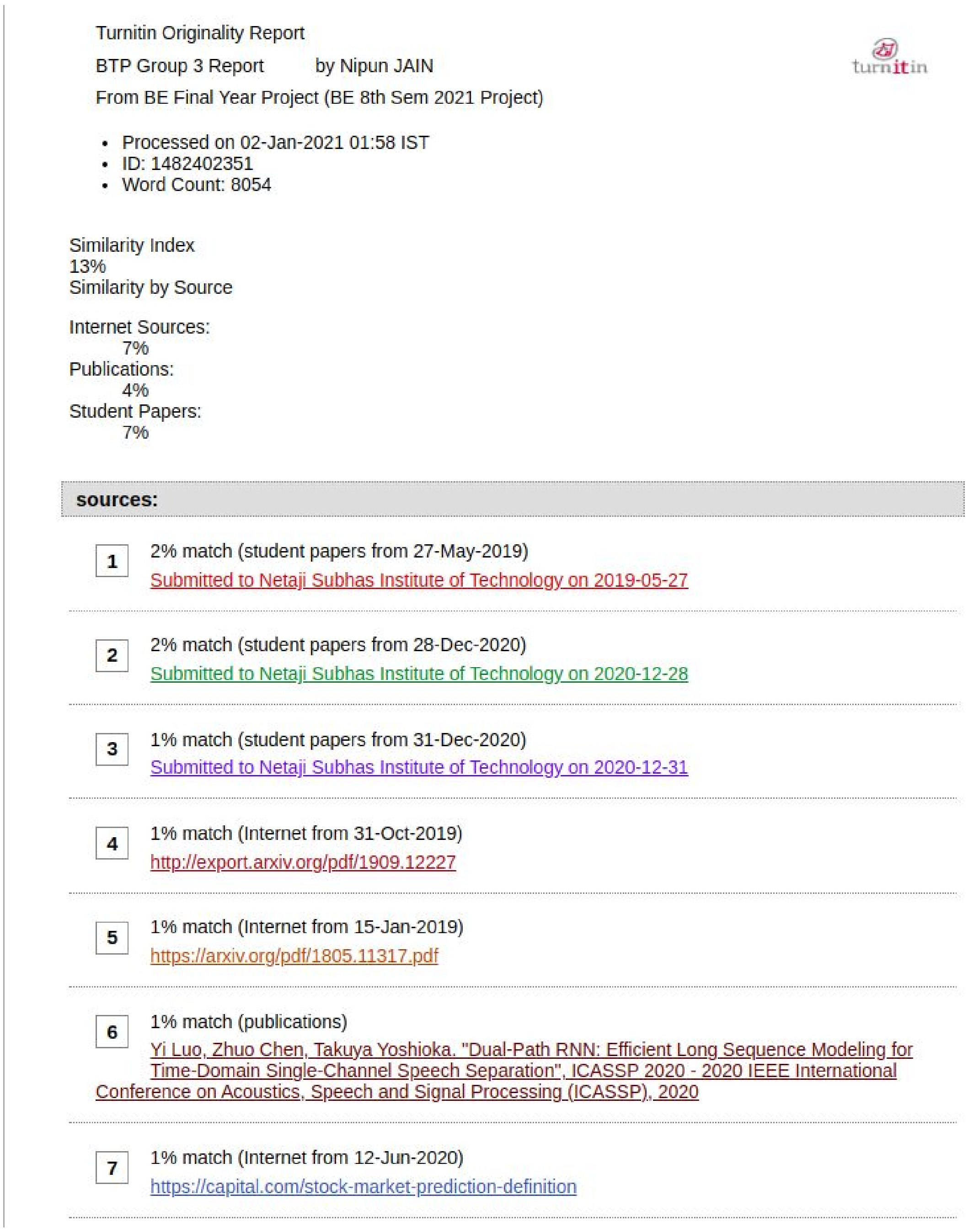
[16] Pagolu, Sasank & Challa, Kamal & Panda, Ganapati & Majhi, Babita. (2016). Sentiment Analysis of Twitter Data for Predicting Stock Market Movements.

[17] Usman, Berto. (2016). The Phenomenon of Bearish and Bullish in The Indonesian Stock Exchange. ESENSI Jurnal Bisnis dan Manajemen. 6. 181-198. 10.15408/ess.v6i2.3750.

[18] Willard, Cheryl. (2020). z-Scores and Other Standard Scores. 10.4324/9780429261039-5.

[19] Zou, Fangyu & Shen, Li & Jie, Zequn & Zhang, Weizhong & Liu, Wei. (2018). A Sufficient Condition for Convergences of Adam and RMSProp.

PLAGIARISM REPORT



Turnitin Originality Report

BTP Group 3 Report by Nipun JAIN

From BE Final Year Project (BE 8th Sem 2021 Project)

- Processed on 02-Jan-2021 01:58 IST
- ID: 1482402351
- Word Count: 8054

Similarity Index
13%
Similarity by Source

Internet Sources:
7%
Publications:
4%
Student Papers:
7%

sources:

1 2% match (student papers from 27-May-2019)
Submitted to Netaji Subhas Institute of Technology on 2019-05-27

2 2% match (student papers from 28-Dec-2020)
Submitted to Netaji Subhas Institute of Technology on 2020-12-28

3 1% match (student papers from 31-Dec-2020)
Submitted to Netaji Subhas Institute of Technology on 2020-12-31

4 1% match (Internet from 31-Oct-2019)
http://export.arxiv.org/pdf/1909.12227

5 1% match (Internet from 15-Jan-2019)
https://arxiv.org/pdf/1805.11317.pdf

6 1% match (publications)
Yi Luo, Zhuo Chen, Takuya Yoshioka. "Dual-Path RNN: Efficient Long Sequence Modeling for Time-Domain Single-Channel Speech Separation", ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020

7 1% match (Internet from 12-Jun-2020)
https://capital.com/stock-market-prediction-definition