Real-Time Stock Market Recommendation & Prediction using Multi Source Data

Kalpana Konety
Real-Time Stock Market Recommendation & Prediction using Multi Source Data

Kalpana Konety

A dissertation submitted in partial fulfilment of the requirements of Technology University Dublin for the degree of M.Sc. in Computing (Data Science)

June 2022
Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Science), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technology University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute’s guidelines for ethics in research.

Signed: Kalpana Konety

Date: June 2022
Abstract

Stock investors must be cognizant of both the current price of their stock and the price at which they want to sell it in the future. This does not stop investors to monitor past price patterns and apply their knowledge to the present. ‘Past performance is not an indicator of future success’, as the saying goes. To put it another way, historical stock data alone isn’t enough to forecast future stock prices. Another key factor to consider in a trading strategy is the impact of market psychology. Financial data, which is a type of multimedia data, provides a wealth of information that has been widely used for data analysis tasks. However, predicting stock prices remains a popular study topic for investors and financial scholars. Forecasting stock prices has become an extremely difficult undertaking because of the significant noise, nonlinearity, and volatility of stock price statistic data. In order to get superior stock price prediction outcomes, we’ve fitted the financial data with Bi-directional deep learning long short-term memory (BiLSTM) neural networks, which train the data going in forward and backward directions. In this dissertation, Uni-directional LSTM is compared to Bi-directional LSTM (BiLSTM) in predicting the adjusted close price of NASDAQ stocks of Google, Apple, Facebook and Amazon. This dissertation compares the LSTM and BiLSTM models over a few time steps, one day, three days, and ten days, using diverse combinations of financial data sources, such as history stock price, technical indicators, News Articles scraped from numerous News sources, and Tweets gathered from verified Twitter accounts. It was possible to fine-tune transformer language models BERT, DistilBERT, RoBERTa, and ALBERT by using labelled financial News articles and labelled Twitter stock Tweets to extract sentiments from unlabelled News and Tweets data in this dissertation, which incorporated both market and investor
sentiment. In both cases, RoBERTa outperformed other models in the BERT family with 88% and 71% accuracy, and with a F1 scores of 0.88 and 0.71 for financial News articles and Twitter stock Tweets, respectively. Technical indicators are one of the most determining factor of how stock prices change. This study uses the technical analysis library (ta-lib) from Python to include technical indicators from two main groups, overlapping studies and statistical functions. A total of 23 technical indicators are introduced in this research. This study addresses the curse of dimensionality by applying PCA on a quantitative stock dataset. Experimental results show dataset combination of ‘Historic Stock Price and Technical Indicators with PCA applied’ achieved better MAPE for different architectural models of both LSTM and BiLSTM. This research incorporated different architectures of LSTM and BiLSTM including single and multi-stacked. When comparing the results of the two models, it was found that BiLSTM outperformed LSTM in forecasting the adjusted closing prices of NASDAQ stocks for Google, Apple, Amazon, and Facebook. LSTM and BiLSTM models were also investigated for the prediction of adjusted close price, which achieved better accuracy when using 4-layer Adam optimized BiLSTM networks with 150 units and 1 day time step, resulting in lower Mean Absolute Percentage Error ranging from 1.86 to 4.36 for different combinations of the data sources compared to 4-layer Adam optimized LSTM models with 150 units and 1 day time step, which achieved higher MAPE ranging from 2.00 to 5.18. Though the training time of BiLSTM was longer than LSTM in every experiment performed with different combinations of datasets, time steps and architectures, still Bi-directional LSTM out performed Uni-directional LSTM in all the experiments. With proper hyper-parameter tuning, BiLSTM models were fairly able to forecast future stock trend with lower MAPE.

**Keywords:** Stock Prediction, Transformer Language Models, Fine-tuning, Sentimental Analysis, Bi-directional LSTM, Uni-directional LSTM, Principle Component Analysis, Adam Optimization, Diebold Marino Tests
I would like to offer my most sincere gratitude to Dr. Giancarlo Salton, my supervisor, for his direction, suggestion, and continuous support during the process of completing my dissertation. I would also want to express my gratitude to Dr. Emma Murphy, my M.Sc. thesis coordinator, for being very helpful in responding to my concerns and providing me with sound advice. In addition, I would like to express my gratitude to Dr. Luca Longo, the professor taught me the fundamentals of research methodology and how to incorporate such principles into research investigations. In addition, I would want to express my gratitude to Dr. Brendan Tierney, my M.Sc. professor, for inspiring me to pursue the most difficult area of stock prediction at a time when I was concerned about my capabilities. I would also want to extend my gratitude to all of the staff members at TU Dublin for assisting me in acquiring the information that was necessary to finish my dissertation. To conclude, I would like to express my gratitude to my parents, Satya Narayana Murthy Konety and Lalitha Konety, as well as to my most loving child, Yashas Vrushank Sreepurushottam, for motivating and encouraging me to pursue a master’s degree, having trust in me to do so, and consistently supporting me throughout the duration of the course, thereby allowing me to realize my goals and realize my dreams.
## Contents

- **Declaration** I
- **Abstract** II
- **Acknowledgments** IV
- **Contents** V
- **List of Figures** X
- **List of Tables** XII
- **List of Acronyms** XIV

### 1 Introduction 1

1.1 Background ............................................. 1
1.2 Research Problem ........................................ 3
1.3 Research Question ....................................... 4
1.4 Research Objectives ..................................... 4
1.5 Research Methodologies .................................. 4
1.6 Scope and Limitations .................................... 5
1.7 Document Outline ....................................... 6

### 2 Literature Review 7

2.1 Review .................................................. 7
2.2 Data Used .............................................. 10
2.2.1 Technical Analysis ............................................. 11
2.2.2 Fundamental Analysis ........................................... 12
2.3 Feature Extraction .................................................. 14
  2.3.1 PCA .......................................................... 14
2.4 Sentimental Analysis .................................................. 15
  2.4.1 Lexicon based ............................................... 15
  2.4.2 Transformers and Language based .......................... 16
2.5 Time Series Predictive Analysis .................................... 18
  2.5.1 Statistical Models .......................................... 18
  2.5.2 Deep Learning Models ...................................... 20
2.6 Model Optimization .................................................. 25
  2.6.1 Adam Technique ........................................... 25
2.7 Research Gaps ...................................................... 27
2.8 Overview .......................................................... 28

3 Experimental design and methodology .................................. 29
  3.1 Design Solution .................................................... 32
  3.2 Design Requirements ............................................. 33
  3.3 Data Understanding ................................................ 33
    3.3.1 Data Collection ........................................... 34
    3.3.2 Data Description ......................................... 35
    3.3.3 Data Exploration ......................................... 36
  3.4 Data Preparation .................................................. 37
    3.4.1 Data Selection ........................................... 37
    3.4.2 Data Cleaning ............................................ 38
    3.4.3 Technical Indicators Computation ......................... 43
    3.4.4 Data Integration ......................................... 47
  3.5 Modeling .......................................................... 48
    3.5.1 PCA ...................................................... 48
    3.5.2 Sentiment Analysis ....................................... 52
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.3 Splitting Time-series Data</td>
<td>55</td>
</tr>
<tr>
<td>3.5.4 Time Series Predictive Analysis</td>
<td>56</td>
</tr>
<tr>
<td>3.5.5 Adam Optimization Technique</td>
<td>57</td>
</tr>
<tr>
<td>3.6 Model Evaluation</td>
<td>58</td>
</tr>
<tr>
<td>3.6.1 Classification Models Evaluation</td>
<td>58</td>
</tr>
<tr>
<td>3.6.2 Time Series Prediction Models Evaluation</td>
<td>59</td>
</tr>
<tr>
<td>3.6.3 Diebold-Mariano (DM) Test</td>
<td>60</td>
</tr>
<tr>
<td>3.7 Overview</td>
<td>61</td>
</tr>
<tr>
<td>4 Experimental Implementation</td>
<td>62</td>
</tr>
<tr>
<td>4.1 Sentiment Extraction for News articles</td>
<td>62</td>
</tr>
<tr>
<td>4.1.1 Data Loading for Fine Tuning</td>
<td>63</td>
</tr>
<tr>
<td>4.1.2 Data Pre-processing</td>
<td>63</td>
</tr>
<tr>
<td>4.1.3 Review length distribution</td>
<td>63</td>
</tr>
<tr>
<td>4.1.4 Train-Validate-Test split</td>
<td>64</td>
</tr>
<tr>
<td>4.1.5 Model training</td>
<td>64</td>
</tr>
<tr>
<td>4.2 Sentiment Extraction for Stock Tweets</td>
<td>66</td>
</tr>
<tr>
<td>4.2.1 VADER sentiment labeling of unlabelled Stock Market Tweets</td>
<td>67</td>
</tr>
<tr>
<td>4.2.2 Data Loading for Fine Tuning</td>
<td>68</td>
</tr>
<tr>
<td>4.2.3 Data Pre-processing</td>
<td>68</td>
</tr>
<tr>
<td>4.2.4 Review length distribution</td>
<td>68</td>
</tr>
<tr>
<td>4.2.5 Train-Validate-Test split</td>
<td>69</td>
</tr>
<tr>
<td>4.2.6 Model training</td>
<td>69</td>
</tr>
<tr>
<td>4.3 Experiment 1 - Predicting Adj close price using Historic Stock Price</td>
<td>71</td>
</tr>
<tr>
<td>4.4 Experiment 2 - Predicting Adj close price using Historic Stock Price and News articles</td>
<td>73</td>
</tr>
<tr>
<td>4.5 Experiment 3 - Predicting Adj close price using Historic Stock Price and Stock Tweets</td>
<td>73</td>
</tr>
</tbody>
</table>
5.7 Evaluation and Results of Experiment 4 - Predicting Adj close price using Historic Stock Price and Technical Indicators ........................................... 90
5.8 Evaluation and Results of Experiment 5 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied ........................................... 92
5.9 Evaluation and Results of 6 - Predicting Adj close price using Historic Stock Price, New Articles and Stock Tweets ........................................... 94
5.10 Evaluation and Results of Experiment 7 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, plus News articles ........................................... 96
5.11 Evaluation and Results of Experiment 8 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, plus Stock Tweets ........................................... 98
5.12 Evaluation and Results of Experiment 9 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets ........................................... 100
5.13 Statistical Evaluation of Hypothesis ........................................... 102
5.14 BiLSTM Prediction of adjusted close price ........................................... 104
5.15 Overview ........................................... 105

6 Conclusion ........................................... 106
6.1 Research Overview ........................................... 106
6.2 Problem Definition ........................................... 107
6.3 Experiment, Evaluation, and Result ........................................... 108
6.4 Contributions and impact ........................................... 110
6.5 Future Work and recommendations ........................................... 111

References ........................................... 112

A Additional content ........................................... 130
List of Figures

2.1 Categorisation of literature review ........................................ 10
2.2 LSTM model Architecture ................................................... 22
2.3 BiLSTM model Architecture ................................................ 24
2.4 Adam pseudo code ............................................................. 26
3.1 CRISP-DM methodology .................................................... 30
3.2 Design Solution ............................................................... 32
3.3 adjusted close price trend from 2020 Jan to 2021 Dec ............... 36
3.4 Normalized adjusted close price trend from 2020 Jan to 2021 Dec . 40
3.5 Most frequent words in the Twitter from 2020 Jan to 2021 Dec . . 42
3.6 Most frequent words in the News from 2020 Jan to 2021 Dec . . . 42
3.7 Stock Technical Features without PCA .................................... 48
3.8 PCA-transformed components’ cumulative explained variance ...... 51
3.9 Stock Technical Features with PCA explained variance of 95% .... 51
3.10 Expanding Window ........................................................... 55
4.1 Financial Phrase Bank dataset Exploratory Analysis ................. 63
4.2 Stock Market Tweets, IEEE Dataport dataset Exploratory Analysis . 68
5.1 Model Train vs Validation Loss of Single vs Stacked BiLSTM models . 80
5.2 MAPE values for Adj close price prediction of Google using 4 Layer LSTM and BiLSTM models with 150 units and 0.2 dropout . . . 81
5.3 Predicted vs Actual adjusted close prices ............................... 104
A.1 RoBERTa sentimental analysis of Apple’s News Articles and Twitter Tweets from 1st Jan 2020 to 9th Jan 2022 ................................. 130
A.2 Adj Close price trend line from 2020 to 2021 ................................. 131
A.3 Frequent of News articles sources ..................................................... 131
A.4 Frequent of Twitter Tweets in a day of 24 hours for 2020 and 2021 ...... 132
A.5 Frequent of Twitter Tweets in a week for 2020 and 2021 .................. 132
A.6 Frequent words in News articles ...................................................... 133
A.7 Frequent words in Twitter Tweets .................................................... 133
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Design and system requirements</td>
<td>33</td>
</tr>
<tr>
<td>3.2</td>
<td>Data description</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Technical Indicators from Pyhton’s ta-lib</td>
<td>46</td>
</tr>
<tr>
<td>3.4</td>
<td>Different Architectures of LSTM and BiLSTM Models</td>
<td>56</td>
</tr>
<tr>
<td>5.1</td>
<td>Experimental Results on the Financial PhraseBank dataset</td>
<td>82</td>
</tr>
<tr>
<td>5.2</td>
<td>Experimental Results on the Stock Market Tweets Dataset</td>
<td>83</td>
</tr>
<tr>
<td>5.3</td>
<td>Learning results using Historic Stock Price</td>
<td>85</td>
</tr>
<tr>
<td>5.4</td>
<td>Learning results using Historic Stock Price and News articles</td>
<td>87</td>
</tr>
<tr>
<td>5.5</td>
<td>Learning results using Historic Stock Price and Stock Tweets</td>
<td>89</td>
</tr>
<tr>
<td>5.6</td>
<td>Learning results using Historic Stock Price and Technical Indicators</td>
<td>91</td>
</tr>
<tr>
<td>5.7</td>
<td>Learning results using Historic Stock Price and Technical Indicators with PCA Applied</td>
<td>93</td>
</tr>
<tr>
<td>5.8</td>
<td>Learning results using Historic Stock Price, New Articles and Stock Tweets</td>
<td>95</td>
</tr>
<tr>
<td>5.9</td>
<td>Learning results using Historic Stock Price and Technical Indicators with PCA Applied, plus News articles</td>
<td>97</td>
</tr>
<tr>
<td>5.10</td>
<td>Learning results using Historic Stock Price and Technical Indicators with PCA Applied and Stock Tweets</td>
<td>99</td>
</tr>
<tr>
<td>5.11</td>
<td>Learning results using Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets</td>
<td>101</td>
</tr>
</tbody>
</table>
5.12 Diebold-Marino test results for BiLSTM and LSTM models in predicting Adj close price using 'Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets' dataset . . . 103
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADX</td>
<td>Average Directional Movement Index</td>
</tr>
<tr>
<td>ALBERT</td>
<td>A Lite BERT</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial neural networks</td>
</tr>
<tr>
<td>Adam</td>
<td>Adaptive Moment Estimation</td>
</tr>
<tr>
<td>ARIMA</td>
<td>Auto-Regressive Integrated Moving Average</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive Moving Average</td>
</tr>
<tr>
<td>AdaGrad</td>
<td>Adaptive Gradient Algorithm</td>
</tr>
<tr>
<td>AdaDelta</td>
<td>Gradient Descent-based Learning Algorithm</td>
</tr>
<tr>
<td>BERT</td>
<td>Bi-directional Encoder Representations from Transformers</td>
</tr>
<tr>
<td>BPE</td>
<td>Byte-Pair Encoding</td>
</tr>
<tr>
<td>BiLSTM</td>
<td>Bi-directional Long-Term Short-Term Model</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CRISP-DM</td>
<td>Cross Industry Standard Process for Data Mining</td>
</tr>
<tr>
<td>CNN-Bi-LSTM</td>
<td>Convolutional Neural Network with Bi-directional Long-Short Term Memory</td>
</tr>
<tr>
<td>DL</td>
<td>Deep Learning</td>
</tr>
<tr>
<td>DM</td>
<td>Diebold-Mariano</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>Distilled version of the BERT</td>
</tr>
<tr>
<td>EMA</td>
<td>Exponential Moving Average</td>
</tr>
<tr>
<td>FinBERT</td>
<td>Financial Sentiment Analysis with BERT</td>
</tr>
<tr>
<td>(GPU/TPU)</td>
<td>Graphics processing unit/Tensor Processing Unit</td>
</tr>
<tr>
<td>GRU</td>
<td>Gated Recurrent Unit</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long-Term Short-Term Model</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>MAPE</td>
<td>Mean Absolute Percentage Error</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-layer perceptron</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum A Posteriori</td>
</tr>
<tr>
<td>MLM</td>
<td>Masked Language Modellin</td>
</tr>
<tr>
<td>NIFTY</td>
<td>National Fifty</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>NSE</td>
<td>National Stock Exchange</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Square Error</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>RSI</td>
<td>Relative Strength Index</td>
</tr>
<tr>
<td>SMA</td>
<td>Simple Moving Average</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>NSP</td>
<td>Next Sentence Prediction</td>
</tr>
<tr>
<td>NASDAQ</td>
<td>National Association of Securities Dealers Automated Quotations</td>
</tr>
<tr>
<td>PCA</td>
<td>Principle Component Analysis</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>Robustly optimized BERT</td>
</tr>
<tr>
<td>RMSProp</td>
<td>Root Mean Squared Propagation</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>TA</td>
<td>Technical Analysis</td>
</tr>
<tr>
<td>LSTM-RNN</td>
<td>Recurrent Neural Network with Long-Short Term Memory</td>
</tr>
<tr>
<td>VADER</td>
<td>Valence Aware Dictionary for Sentiment Reasoning</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

Stock price forecasting can result in gains for both the seller and the broker. It was
difficult to make accurate stock market predictions in prior years due to a lack of tech-
nology and expertise; but, as technology advances, it is becoming easier and easier to
make accurate stock market predictions when compared to years past. Predictions are
often chaotic rather than random, meaning it can be predicted by examining stock
market history. Machine learning is an effective method for representing complicated
processes, such as anticipating market values that are close to the true value. Because
of the increased precision, machine learning is an effective tool for practical applica-
tions. Many researchers have found machine learning to be an efficient and accurate
way to measure stock prices, so it has been popular to introduce the technology to
this field (Usmani, Adil, Raza, & Ali, 2016; Raza, 2017).

The data utilized in machine learning is the most significant aspect. Because little
changes in the data might have a large influence on the conclusion, the data should be
as detailed as feasible. This research employs supervised machine learning on a dataset
obtained from Yahoo Finance. There are five variables in this dataset: open, close,
low, high, and volume. The stock’s open, close, low, and high prices vary throughout
time, with each price name essentially identical to the direct name. The total number
of shares exchanged within the specified time period is referred to as the volume of
shares traded. The model is later validated against the test data set.

Stock market prediction is gaining huge attention in recent times from financial researchers and investors. Since the stock market has a highly nonlinear and fluctuating time-series data, it requires an efficient model which is appropriate to handle such diverse data. Conventional linear models such as ARMA, and ARIMA (Rounaghi & Zadeh 2016; Banerjee 2014; Saxena, Anurag, Chirayath, Bendale, & Kaul 2018) were used previously for predicting the stock market. However, these models work for specific stocks only i.e., the models will be trained for analyzing specific stocks and cannot be applied for other stocks. Because of the ambiguous and stochastic nature of the stock market, it is complex and risky to make any predictions in the stock market using traditional predictive models. In this context, deep learning (DL) holds a prominent role in forecasting the pattern and trends of the stock market (Hiransha, Gopalakrishnan, Menon, & Soman 2018).

Deep learning techniques have the potential to handle highly nonlinear data which makes it an attractive candidate for stock market prediction (Vargas, De Lima, & Evsukoff 2017). Deep learning-based models have been shown to outperform traditional ARIMA-based models in predicting time series, especially for long-term forecasting problems (Siami-Namini, Tavakoli, & Siami Namin 2018). Although LSTMs have been shown to beat ARIMA, it is worth exploring whether their performance can be further improved by including more layers of training data into LSTMs. It’s worth noting that market-related information may be found on a variety of channels, including social media, News, and corporate reviews. Investors rely on this data to forecast stock price movements, and it’s challenging for DL models to interpret such massive amounts of variegated data on their own. Furthermore, the many scales in financial time series complicate conventional methodologies, making them ineffectual at making reliable predictions. Furthermore, any incorrect estimate could result in significant financial losses. Investors seek precisely projected market values in such instances, making projection a difficult task.
1.2 Research Problem

Accurate prediction of the stock market is of utmost importance for the investors to maximize profits and minimize losses. However, designing a deterministic and robust model for predicting stock market behaviour involves a lot of complexity due to its randomness and unpredictability. Despite the availability of different technical and strategic models, there is still a lack of reliable models that can provide accurate stock market forecasts.

Stock market data is made up of both structured and unstructured data, and analyzing such massive amounts of unstructured data is difficult. Traditional data processing technologies are incapable of dealing with the data’s stochasticity. These models don’t represent the stock market’s internal dynamics, which has an impact on their performance and reliability. More advanced RNN-based models such as LSTMs and GRUs (Gated Recurrent Units) have been shown to be more effective due to their ability to quickly learn complex nonlinear stock patterns and produce superior results. Despite these benefits, there are also some disadvantages that must be dealt with.

- Existing studies have mainly focused on the implementation of a single LSTM model and the implementation of other Bi-LSTM is not focused much.

- Not many existing works based on these models considered both investor sentiments and market sentiments.

- Majority of the existing works operate in offline mode and are not capable of predicting real-time stock data.

- Various existing studies have considered technical indicators for on stock market prediction, but haven’t used any feature extraction techniques such as PCA (Principle Component Analysis) to avoid data dimensionality problem.

- Most of the researchers haven’t considered any optimization strategy to enhance the over-all trading performance.

It is essential to address these issues for the successful implementation of deep learning algorithms for predicting the behavior of the stock market with high accuracy.
CHAPTER 1. INTRODUCTION

1.3 Research Question

The research question formulated is as follows:

*How does a model trained with Bi-directional LSTM outperform a model trained using Uni-directional LSTM in terms of accuracy by incorporating both investor and market sentiments?*

1.4 Research Objectives

The primary objective of this research is to build a model that can accurately predict the adjusted close price of NASDAQ stocks for Google, Apple, Amazon and Facebook by combining technical and fundamental analysis, and that can work in real-time.

1.5 Research Methodologies

This study is classified as a secondary study because it gathers information from previous studies and then proposes a new strategy using data from multiple sources testing state-of-the-art models. In this study, the deep learning models (LSTM, BiLSTM) are implemented and evaluated quantitatively based on the MAPE value of the models. Moreover, this study employs an empirical approach, conducting various experiments on models using various datasets to determine, which model performs best in predicting adjusted closing prices. The study is inductive because the models and methods presented in this study were motivated by reviewing existing work and developing research hypotheses, which were then tested using experimental results. This is a quantitative research, in which the Diebold Marino tests are used to evaluate the hypothesis. The data comes from a variety of sources, including historical stock prices obtained from the Yahoo Finance website, News articles obtained from the Google News Library, and Tweets obtained from the Twint Library obtained from Twitter. In order to feed LSTM and BiLSTM models with this data, the data are subjected to a systematically analysed and then translated into numerical data.
1.6 Scope and Limitations

The goal of this study is to use a Deep Learning-based Bi-directional LSTM model for stock market predictions using a variety of data sources, including 'Historic Stock Price', 'Technical Indicators', and 'Twitter Tweets' and 'News articles'. This study uses NASDAQ data of Google, Apple, Amazon, and Facebook stocks from January 1, 2020 to December 31, 2021. This time period includes major events such as 'The COVID-19 pandemic', 'Joe Biden’s Presidency’, and 'China Asserts Itself as Global Leader'. Tweets and News data collected during this period shows investor sentiments about major events and, this dissertation study how those sentiments affect the financial market. Python’s ta-lib, technical indicators library are divided into ten sub-sections, and this study employs 23 technical indicators from 2 sub-sections, namely Overlap Studies groups and Statistic Functions group. The technical indicators in this study are calculated using a 20-day window.

Due to time constraints, and Twitter’s restriction on the number of Tweets collected at a time, the Stock Tweets are collected from verified users and are limited to only 5 users, '@stocktwits', '@MarketWatch', '@IBDinvestors', '@wsjmarkets', and '@cnbc'. The study only looked at English-language News articles and Tweets. Owing to the scarcity of labelled financial datasets, this study considered smaller datasets of less than 5000 rows for fine tuning transformers models for further sentimental analysis of News and Tweets data. Further baseline LSTM models and proposed BiLSTM models developed in this study use a similar architecture to allow for a fair comparison of their performance. The models that were developed as a result of this study are able to function in real time and provide accurate forecasts of adjusted close price. In this research, rather than fully automating the process of live streaming data by enabling scheduler to feed models with real-time data, scripts were manually run to retrieve data for testing in real time. This was done rather than automating the process of live streaming data. This is not within the scope of this study since the primary emphasis is placed on machine learning rather than big data.
1.7 Document Outline

The research paper is divided into the following chapters:

i) **Chapter 2:** "Literature Review" analysis and review of previous works are undertaken to fully comprehend the approaches being used forecast stock markets. This chapter discusses works of literature that incorporated data from multiple sources, data feature extraction, and various prediction models and analyzed their effectiveness to determine the underlying gaps and how this research fills these gaps.

ii) **Chapter 3:** "Experiment design and methodology" describes the CRISP-DM methodology used in this study, with a detailed explanation of each stage of the CRISP-DM methodology. The chapter goes on to describe the design aspects of the models used in this study.

iii) **Chapter 4:** "Experimental Implementation" explains the implementation of models based on the experimental design described in Chapter 3.

iv) **Chapter 5:** "Evaluation and Results" analyzes and evaluates the results obtained by implementing each model in Chapter 4 to determine each model’s performance in the research.

v) **Chapter 6:** "Conclusion" provides an overview of the work done in this study and concludes the performance of the proposed approach used in this study. This chapter also discusses the research’s contribution to stock market prediction and recommends gaps that should be addressed in future research.
Chapter 2

Literature Review

2.1 Review

Extensive research has been conducted in the field of stock market prediction. Many of these studies make use of historical price data and technical indicators as features. Machine learning techniques have been used by some researchers on historical price data, volume, average variance, etc. (T. Chen & Bahsoon, 2013; Kim, 2003; Kouloumpis, Wilson, & Moore, 2011; Pak & Paroubek, 2010; S. Shen, Jiang, & Zhang, 2012).

Previous research on stock prediction has relied mostly on historical or social media data. Using historical data for research means using a technical analysis technique to study data in order to forecast future stock market patterns and prices. (Lien Minh, Sadeghi-Niaraki, Huy, Min, & Moon, 2018). On stock historical price data, researchers employed several machine learning approaches, such as deep learning (X, H, L, J, & X, 2014) and regression analysis (Jeon, Hong, & Chang, 2018). However, these studies did not take into account external influences such as social media. It is critical to use social media data because events communicated on social media can greatly impact stock prices and trends owing to the assumption that prices move according to human behaviour, which social media can represent.

According to (Tim & Bill, 2011) seminal work, word lists produced for other fields misclassify popular terms in financial documents. Consequently, Loughran and McDonald developed an expert annotated vocabulary of positive, negative, and neutral
financial terms that better express moods in financial content, (Terblanche & Mari-vate, 2021). Researchers combined lexicon-based techniques with machine-learning algorithms. The disadvantage of Lexicon-based methodologies for sentiment analysis in the stock market area is that there are no lexicons that incorporate financial terms, causing the model to misread text sentiments. To overcome this challenge, pre-trained models like BERT is used by (Matheus et al., 2019) which gave high accuracy.

FinBERT, an adaption of BERT for the financial arena, achieved significantly higher accuracy than BERT by performing additional pre-training on a financial corpus and refining it for sentiment analysis (Araci & Genc, 2020). The authors of (Sahar, Nicholas, & Dingding, 2018) demonstrate that such combinations are more efficient than single models for sentiment extraction. Regular machine-learning approaches, on the other hand, are incapable of maintaining the order of words in a phrase and extracting complicated features. Deep-learning techniques are required for these tasks, which allow for complicated feature extraction, location identification, and order information (Sahar, Dingding, Anna, & Taghi, 2018).

According to (Mehtab, Sen, & Dutta, 2020) predicting stock prices is a highly complex task and requires sophisticated technology-based models to predict the future movement of stocks. Various mathematical models have been proposed in this context, but the results are quite disappointing (Patel, Shah, Thakkar, & Kotecha, 2015). A deep neural network (DNN) based predictive model was proposed in (X. Liang, Ge, Sun, He, & Chen, 2019) for predicting the movement of stock price. The proposed technique showed satisfactory results compared to conventional techniques in terms of accuracy. Similarly (Stoean, Paja, Stoean, & Sandita, 2019) implemented an LSTM based stock predictive approach for predicting the closing price of the stocks listed on the Bucharest Stock Exchange based on their previous price movements. It was observed that better accuracy can be achieved by combining multiple stock price indicators. Based on the same theory, (Mehtab & Sen, 2020) asserted that there has been the use of social media and non-linear and multivariate regression models to analyze the sentiments of the users.

(Moghar & Hamiche, 2020) aimed to measure the number of epochs required for the
LSTM-RNN model for anticipating the variations or fluctuations in the stock market, price trends and future asset values based on the trading data. Experimental results of the proposed model showed promising results. An improved SVM ensemble with genetic algorithm (GA) used as a predictive model was proposed in (Nti, Adekoya, & Weyori, 2020a). It was inferred that the ensemble algorithms provide superior prediction accuracy. (Nabipour, Nayyeri, Jabani, Shahab, & Mosavi, 2020) provided a comparative analysis of different ML and DL algorithms using continuous and binary data. Nine distinct ML and DL algorithms were subjected for experimental evaluation where the results showed that DL algorithms are best for evaluating the binary data. (Pang, Zhou, Wang, Lin, & Chang, 2020) developed a neural network technique for stock prediction using multi-stock high-dimensional historical data. The method used an integrated LSTM layer and auto encoder for prediction. The integrated deep LSTM method performed better. (Chung & Shin, 2020) used the CNN model with a GA to anticipate stock index changes. GA optimized CNN. Even though the combined form performed better, further study is needed.

In (C. Liang, Tang, Li, & Wei, 2020; Y. Chen, Zhao, Li, & Lu, 2020) the accuracy of prediction is highly affected by the volatility of the stock market and it is difficult to overcome this problem using only technical indicators, which fail to capture all information related to price movements. And this problem emerged as one of the biggest problems and gained huge attention. Also, another problem is the ineffectiveness of the DL models to handle large scale unstructured data. (Di Persio & Honchar, 2016) studied the efficacy of LSTM for financial time series forecasting. (Akita, Yoshihara, Matsubara, & Uehara, 2016) compiled data from newspaper stories to show the influence of past occurrences on stock market opening prices. Their formula sent numerical and written data to LSTM for exact predictions. (Di Persio & Honchar, 2017) used Bi-LSTM to anticipate energy load.

Bi-LSTM and multi-layer LSTM were compared. Bi-LSTM design performed better. No known study has compared LSTM and Bi-LSTM for stock price prediction performance. This study offered a unique methodology using Bi-LSTM to attain the best stock price predicting performance.
CHAPTER 2. LITERATURE REVIEW

Sentiment analysis affects prediction accuracy (Yasir et al., 2020; Nti, Adekoya, & Weyori, 2020b; Mehta, Pandya, & Kotecha, 2021). Incorporating market emotions improves stock market forecast accuracy, but most of existing models have limitations. They don’t include optimization strategies for fine-tuning algorithm parameters, which affects accuracy. Market forecasts are only accurate for the short term and need considerable processing effort due to complexity. Author (Bergstrom & Hjelm, 2019) concluded in his research, variable time steps on a particular LSTM model do not substantially impact the model’s prediction skills; nonetheless, 10 time steps seems to be the optimum number among the numerous choices evaluated. This study uses fewer time steps to evaluate whether stock prediction improves, while (Bergstrom & Hjelm, 2019) only evaluated models for 10 days or longer.

![Figure 2.1: Categorisation of literature review](image)

2.2 Data Used

Quality and quantity of data are needed to predict market trends and choose a forecasting model. Data analysis includes technical and fundamental analysis. Analysts use charts and plots of historical performance and stock technical indicators to anticipate future company performance (Ahmadi et al., 2018). Analysts consider the political party in control, new government laws, investor attitude, a pandemic, and the company’s market position (J. G. Agrawal, Chourasia, & Mittra, 2013). (Nti, Adekoya, & Weyori, 2019), a survey of 122 research publications found that 66% of current research relies on technical analysis and 23% on fundamental analysis.
2.2.1 Technical Analysis

This approach predicts the future using pre-processed stock prices and technical indicators. Data collection frequency determines short- and long-term projections (daily, weekly, or monthly). Historical stock prices are calculated using many factors, including: open (the price at which a stock first trades), high (the stock’s previous 52-week high price per share), low (the stock’s previous 52-week low price per share), closing (the stock’s day-ending price), volume (the number of shares traded), and adjusted closing price (stock price after accounting for dividends). The day’s closing price reveals whether an investment has gained or lost money, thus it’s utilized as an impartial measure in research. Fitting models using EMA, SMA, and Bollinger bands (Borovkova & Tsiamas, 2019; Chan, 2020a). (Borovkova & Tsiamas, 2019) suggests using competitor stock prices to forecast price movements.22 US corporations with market caps above $10 billion and one competitor per share delivered better accurate estimates. Non-linear, non-stationary, and noisy stock data must be translated for prediction models and reduce outlier bias. Centering, standardizing, and normalizing transform data (data is transformed to have a normal distribution by removing outliers). (Chan, 2020b) used log-transformed historical Myanmar Stock Price Index data to stabilize it. Traditional approaches can’t denoise data well. Vargas and dos (Vargas, dos Anjos, Bichara, & Evsukoff, 2018) suggest that judging a company’s market position entirely on quantitative data is impossible since financial markets are impacted by so many other aspects. This despite bullish stock data and technical signs. Many clients acquire stocks after reading about a company’s success in the News and on social media.

Analysts often use at least one technical signal in stock forecasts (Bogullu, Enke, & Dagli, 2002). When choosing an indicator, model correctness is generally considered. Several indications are sometimes disregarded, and a suitable stock model may never be established. Many feel that the more inputs a neural network receives, the better its prediction. If input data is irrelevant to the desired output, it will be harder for the network to link relevant inputs to the output. This research used PCA to reduce the dimensionality of 23 Python ta-lib technical indicators.
CHAPTER 2. LITERATURE REVIEW

2.2.2 Fundamental Analysis

Technical analysis that only uses structured data for financial stock predictions is inadequate because it does not take into account market sentiments obtained from these online sources, which are becoming more and more popular due to people’s reliance on News and the internet for stock market information (Cheng, Huang, & Wu, 2018). (Nagesh, 2021) utilized multi source in his study to anticipate the price of Indian stocks using Facebook Prophet and Attention based-LSTM models. This study employed a similar strategy for data conversions, integration, and evaluation.

Unstructured internet data sources fall into three types, according to (Mao, Counts, & Bollen, 2011):

i. **News websites**: It is the reliability of News websites that people use to decide whether or not to purchase or sell shares in a business. Market sentiment has been proved to have an impact on stock prices in the future. In literature, text analytics and NLP methodologies have proved the value of automatically extracting such unstructured data and their underlying attitudes, as evidenced by many works of literature. From January 2, 2013, to February 4, 2016, (Joshi, N, & Rao, 2016) studied Apple Inc. News from Google, Yahoo Finance, and Reuters sites to estimate the adjusted closing price of the next day. Bag of words technique was used to classify the string of words into either good or negative emotion, according to the author, who used both News headlines and whole News articles as input to execute the bag of words approach. There are two traditional ways to analyze sentiment: positive and negative. (Kumar, Ravi, & Miglani, 2020) proposed an alternative method that classified each text as one of 93 of 241 emotion types, respectively, by extracting psycho-linguistic characteristics from the News articles or headlines using Linguistic Inquiry and Word Count and Tool for the Automatic Analysis of Lexical Sophistication (TAALES) tools. Business standards News site data is used to compile the final feature set, which consists of the top 10 and top 25 extracted psycho-linguistic characteristics from 12 prominent Indian companies across various sectors. The final feature set is then narrowed down to have relevant features using chi-square and Minimum Redundancy Maximum Relevance.
ii. **Social Media:** In addition to News sources, investor opinions made on social media have shown to be an essential component in predicting election results, and it has lately had a big influence on financial market forecast (C. Li et al., 2019). (Malla Reddy, Y, Krishna, & Miranam, 2019) developed a multi-source multi-instance model with a sentiment analyzer to extract sentiments from Tweets gathered from Twitter for the Bombay Stock Exchange (BSE) stock index. Each text is tokenized after being pre-processed to remove punctuation and stop-words. Using a text-blob and a pre-ordered sentiment lexicon, this tokenized text is classified into positive and negative attitudes. To anticipate daily closing prices, the author additionally used historical stock price data, which was put into a stacked LSTM model with two LSTM layers. The final prediction model used is SVM, which uses sentiment value and predicted closing price as inputs to forecast the final daily closing price (Malla Reddy et al., 2019). To obtain more reliable sentiment analysis findings for Tweets, it is necessary to construct a dataset from reputable Tweets and choose an acceptable lexicon based on financial phrases (Oliveira, Cortez, & Areal, 2017).

iii. **Combined Data:** Using a single source to create a model is not practical because both social media and News posts associate with future market movements. Two years’ supply of News postings (2015 and 2016) are retrieved from Chinese News sites hexun and Sina finance social media posts are scraped from Xueqiu, and past stock price data of the Shanghai Composite Index are collected from Wind, by a multi-instance multi-source model developed by (X. Zhang, Qu, Huang, Fang, & Yu, 2018). The author extracts events and sentiment from News and social media messages using an event extraction and sentiment analyzer. The structural information portraying the event of News postings is retrieved using the HnLAP text parser, the result of which is given to the Restricted Boltzmann Machine, which generates a pre-trained vector to be applied as input to the sentence2vec model to represent events (X. Zhang et al., 2018). According to (X. Zhang et al., 2018), the sentiments of social media postings are extracted using the Latent Dirichlet Allocation topic modeling technique, and using data from all three sources (News events, sentiments, and quantitative stock data) a Multi-instance model based on Multiple Instance Learning algorithms is trained.
2.3 Feature Extraction

Feature extraction is a critical tool in pattern recognition and data mining technologies. It isolates the useful feature subset from the original dates using certain criteria to decrease machine training time and space complexity in order to meet the aim of dimensionality reduction. Feature extraction converts the input data into a collection of features, but the resultant reduced representation retains the majority of the original data’s significant information (J. Li & Xie, 2009).

2.3.1 PCA

Principal Component Analysis (PCA) is a data dimensionality reduction method, (Yao, Pan, Yang, Chen, & Li, 2019). PCA is used in a variety of fields, including data compression, pattern recognition, feature extraction, statistical variable analysis (Hotelling, 1933), and visualisation of high-dimensional data (Jolliffe, 2011). PCA finds a minimal number of key components that explain the vast majority of variance in a data set. It may also be used to investigate financial time series (Ince & Trafalis, 2007), dynamic trading techniques (Fung & Hsieh, 1997), financial risk calculations (Fung & Hsieh, 1997; Alexander, 2009), and statistical arbitrage (Shukla R, 1990).

(Yu, Chen, & Zhang, 2014) article employs PCA to extract efficient data with low-dimensional, when implementing the machine-learning technique for developing a stock-selection model capable of nonlinear stock categorization. As mentioned in (Ghorbani & Chong, 2020) research, according to (Zhong & Enke, 2017), PCA, fuzzy robust principal component analysis, and kernel-based PCA, are three mature dimensionality reduction techniques used on a complete data set to reorganize and simplify the original data structure. (J. Wang & Wang, 2015) paper proposes a stochastic function tailored for financial time-series prediction based on PCA. According to (Pasini, 2017), PCA is utilized to improve portfolios of Dow Jones Industrial (DJI) index having three subgroups of businesses.
2.4 Sentimental Analysis

The goal of financial sentiment analysis differs from that of general sentiment analysis. The goal of financial sentiment analysis is often to predict how the markets will respond to the information offered in the text (X et al., 2014).

2.4.1 Lexicon based

Method based on a lexicon To establish polarity, (Taboada, Brooke, Tofiloski, Voll, & Stede, 2011) employs a sentiment lexicon containing opinion terms and matches them with data. They give sentiment ratings to opinion words, which describe how positive, negative, or objective the words in the dictionary are. Lexicon-based techniques, such as the Opinion Finder lexicon, depend primarily on a sentiment lexicon, which is a collection of established and pre-compiled sentiment words, phrases, and even idioms designed for conventional genres of communication. This method is divided into two categories:

Dictionary-based: It is based on the use of phrases (seeds) that are typically manually gathered and annotated. This collection expands by scanning a dictionary’s synonyms and antonyms. WordNet is one such dictionary, and it is used to create SentiWordNet, a thesaurus.

Drawback: Cannot handle domain or context-specific orientations.

Corpus-Based: The corpus-based method aims to provide dictionaries for a certain domain. These dictionaries are created from a collection of seed opinion keywords that develop as a result of the search for related words using either statistical or semantic methodologies.


- Semantic methods, such as the usage of synonyms and antonyms, or relationships from thesauruses such as WordNet, may potentially be an intriguing alternative. According to (Kharde & Sonawane, 2016), performance of machine learning models evaluates accuracy and recall. SVM and Naive Bayes outperform lexicon-based methods.
2.4.2 Transformers and Language based

It is not difficult to see that deep learning outperforms machine learning in numerous fields (W. Yang, Xuemin, Lin, & Wenjie, 2017; Lin, Yang, Xue, & Junbin, 2018; Lin, Richang, Yang, & Meng, 2019; W. Yang et al., 2016; Lin, Yang, Hongzhi, Meng, & Ling, 2019). As a result, it is not an exception in the field of finance. (TIM & BILL, 2016) provides a comprehensive evaluation of current efforts on financial text analysis using machine learning using a "bag-of-words" approach or lexicon-based algorithms, and develop a dictionary of financial phrases with assigned values such as "positive" or "negative" and assess the tone of a document by counting words with a certain dictionary value, (TIM & BILL, 2011). (Mathias & Stefan, 2017) was the first to offer deep learning algorithms for textual financial polarity analysis. They use an LSTM neural network to forecast stock market movements and demonstrate that their techniques outperform established machine learning approaches. (Sohangir, Wang, Pomeranets, & Khoshgoftaar, 2018) applies different generic neural network designs to a StockTwits dataset, determining CNN to be the top performing neural network architecture. However, owing to a shortage of big labeled financial datasets, it is challenging to fully exploit neural networks for sentiment analysis. A more promising approach may be to train practically the whole model using pre-trained values and then fine-tune those values for the classification problem. As a result, this study adopts the concept of migration learning and fine-tunes it using mature pre-training language models, DistilBERT, BERT, RoBERTa, and ALBERT, to perform sentiment analysis. (Henry, 2021) used a restaurant review dataset specific to highlighting different sentiments of positivity and negativity for a comparative study of NLP techniques for sentiment analysis, and demonstrated that distilBERT outperforms other NLP approaches, including VADER and LSTM, in sentiment analysis.

Sentiment analysis has been shown in several studies to be an important tool for predicting stock market movements. In order to use NLP techniques such as VADAR, TextBlob and others to perform sentiment analysis, the data must be labeled before training. A single labeled text data set cannot be utilized to better anticipate the stock market’s behavior, necessitating a tailored collection of textual data for the particular
market in question to be generated (M. Li, Li, Wang, Xiaojun, & Rui, 2020). A new study has shown that pre-trained transformer models are useful in capturing investor emotions (Sousa et al., 2019).

BERT uses a transformer encoder but lacks a decoder, (Vaswani et al., 2017). (Othan, Kilimci, & Uysal, 2019) utilized Turkish Stock Exchange Twitter comments to compare the transformer BERT model to CNN, RNN, and LSTM. TextBlob preprocessed and labeled Tweets before training supervised models (CNN, RNN, and LSTM). The BERT model beat deep learning models in accuracy by 1% to 2%. Despite limited research, BERT was employed for stock forecasting. This research combines the BiLSTM and LSTM algorithms to forecast future stock movements.

(Adoma, Henry, & Chen, 2020) evaluated the performance of pre-trained transformer models BERT, RoBERTa, DistilBERT, and XLNet in identifying emotions from the ISEAR dataset. These models were effective in detecting emotions in text, with RoBERTa achieving the best recognition accuracy. RoBERTa outperformed the other candidate models in identifying emotions on the ISEAR dataset in terms of accuracy, recall, and F1-scores. (Gupta & Chakravarthi, 2021) employed BERT models for fake news detection using transfer learning, which is a binary classification issue (fake or not-fake), and RoBERTa performed better than other BERT models. This study employed a similar strategy to fine-tune BERT models but for multi class problem with positive, negative, and neutral sentiments.

The FinBERT model has been trained on a large set of financial corpora that are indicative of English financial communications, authors (Yi, Mark, & Allen, 2020; Devlin, Chang, Lee, & Toutanova, 2018) compared FinBERT to the original BERT-Base model. And of the three financial sentiment classification tasks employed in these research, FinBERT beats generic BERT models. Those finBERT is the choice when analyzing financial sentiments, it may not classify the Twitter stock Tweets as efficiently, as Tweets text format are different to what finBERT is trained on. For this reason in this study, have trained other models from BERT family on relevant stock data for achieving higher accuracy in sentimental prediction of stock News and stock Tweets.
2.5 Time Series Predictive Analysis

Various models and approaches for forecasting time series have been developed throughout the years. Initially, statistical models such as regression were utilized, but because of the stock market’s non-linear and non-stationary nature, ANN became popular and more appropriate for modeling trends (Patel et al., 2015). ANN has proven successful with nonlinear data for stock markets, however it produces inconsistent forecasts because of noise in the data (W. Shen, Guo, Wu, & Wu, 2011). This study has evolved toward statistical models like as ARIMA and BiLSTM, which are designed specifically for data with strong linearity and seasonality. This section discusses ARIMA research and the recently established BiLSTM model for forecasting stock patterns.

2.5.1 Statistical Models

The ARIMA model, often known as the Box-Jenkins model, is a variant of the ARMA model that incorporates additional differencing components, if the data is not stationary, according to (Ghani & Rahim, 2019). If the data is not consistent, differencing is utilized, and the number of differences is indicated by d (Devi, Sundar, & Alli, 2013). By combining the AR and MA components, the ARIMA model is defined as follows:

\[
ARIMA \quad y_t = \mu + \sum_{i=1}^{m} \phi_i y_{t-i} + \sum_{j=1}^{n} \theta \epsilon_{t-j} \tag{2.1}
\]

Where \( \mu \) is the constant, \( \phi \) is a parameter of AR component, \( \theta \) is a parameter of MA component and \( \epsilon \) is the error term.

The model is formed and deployed in four stages:

i. (J. Liu, Tan, & Wang, 2019), using Autocorrelation and Partial correlational perform charts, confirm the simplest values for Moving average and Autoregression

ii. Statistical tests such as the Philippe-Perron test and Augmented Dickey-Fuller test (d) are used to check for time series data stationarity; the data is considered stationary if the test statistics is greater than the critical value, and differencing
is used by removing the seasonality component and the number of differences in the value of d, to transform the series to be stationary.

iii. The Bayesian Information Criterion (BIC) and the Akaike Information Criterion (AIC) techniques are used to choose the best fitted model among all models, mentioned in (Pietro, 2022).

iv. Using time series data the best fit model is fitted, error metrics or loss functions are used to evaluate the predictions, and residuals are checked for white noise. If the residual is white noise, the procedure is repeated, indicating that the ARIMA model is iterative.

(Devi et al., 2013) argues that stock prices are not always non-stationary and insists on doing a stationarity test instead, while a number of studies have found that stock prices are remarkably volatile and non-stationary; however, It was found that using the augmented Dickey-Fuller test on the four NIFTY 50 businesses indicated that none of closing prices are seasonality, although the data were non-linear, therefore without differencing, just p and q parameters were employed.

From 1995 through 2011, the ARIMA model was used to forecast the daily closing price of Nokia strains (Ariyo, Adewumi, & Ayo, 2014). (Burnham & Anderson, 2004), author used BIC to choose the best model. In (Burnham & Anderson, 2004), however, employing AIC instead of BIC results in more accurate model selection. This is because BIC believes that the best model is among the current models, but AIC assumes that the best model is not among the existing models making to (Y. Yang, 2005) model chosen one. Due to the fact that the ARIMA model in research (R. Zhang et al., 2022) generated lower error values than the LSTM model in monthly and weekly series, it was clear that ARIMA was more effective than LSTM for monthly and weekly forecasting. While the error values provided by LSTM for daily forecasting in rolling forecasting model were lower than those produced by ARIMA, the difference was not statistically significant. The performance of direct forecasting models was inferior to that of rolling forecasting models.
2.5.2 Deep Learning Models

An over-parameterization and over-fitting problem has plagued conventional models such as Artificial neural networks (ANNs) and neural networks (NNs). Over-fitting is an issue that has been successfully addressed by deep models with three or more hidden layers and enough regularization to prevent it (Larochelle, Bengio, Louradour, & Lamblin, 2009). For market prediction, current research has found that merging deep learning models with time series data is useful (Singh & Srivastava, 2017). To use the ARIMA model, one must assume that the data is steady. This is because the ARIMA model assumes stock data is stable and does not change over time, and hence the findings derived by the model cannot be transferred to other stock market data, (Lee & Tong, 2011; Abdoli, MehrAra, & Ardalani, 2020).

A large portion of the time series shows non-linearity if the star based models are used (Melikebildirici & Ersin, 2014). Additionally, it is hard to foresee or estimate the financial exchange conducted without more robust, and exceptionally nonlinear demonstrating techniques (Lendasse, Bodt, Wertz, & Verleysen, 2002). (L.-X. Wang, 2019) examined that deep CNN (DCNN) is increasingly used for resolving the complicated practical issues that are faced while determining future stock prices from the previous data. However, DCNN shows poor performance under high computational load conditions. It suffers from interpretability when operating with more number of model parameters. Using deep neural methods, the future value of stock price can be foreseen with exceptionally nonlinear demonstrating data. A neural network endeavors to map, and highlight the information that is required to be familiar with a function and thus, achieve a better forecasted output (Mandic & Chambers, 2001). It is comprised of a network of neurons with a weighted sum of inputs. Activation functions are used to fit the output from neurons which acquaint non-linearity to the network, and afterward this non-linearity feature is passed to some different neurons. Neural network’s streamlining is typically conducted through back propagation using the gradient descent. Errors are propagated from the output layer to the input layer through this back propagation system. DL models outperforms all other models in forecasting nonlinear data in various time series prediction challenges.
LSTM

The LSTM introduced by (Hochreiter & Schmidhuber, 1997) is one of the variations of RNN models capable of modeling non-linear time series data without the need for data transformation to be stationary, as done with ARIMA. RNNs and LSTM models anticipate by remembering previous data and have therefore shown to be more successful than statistical models like ARIMA for time series forecasting. The research by (Manurung, Budiharto, & Prabowo, 2018) examined the performance of the LSTM and ARIMA models on the stock price of Bank Central Asia from 2013 to 2018. The authors trained and evaluated each model using one, three, and five years of data, and the greatest results were observed with one-year data, where the accuracy of LSTM was 94%, compared to ARIMA’s accuracy of 56%.

The RNN model is similar to NN in that it examines the input sequentially and learns and stores previous knowledge to predict the next item in the sequence. They are made up of numerous hidden layers that retain previous knowledge gained in order to anticipate the next observation in the sequence. The main disadvantage of the RNN model is that it can only detect short-term connections in the sequence and cannot identify long-term relationships, as (Selvin, Vinaya kumar, Gopala krishnan, Menon, & Soman, 2017) points out. The LSTM is an RNN modification that adds features to address the vanishing gradient issue by finding long-term relationships in the sequence. Figure 2.2 depicts the LSTM model, which consists of three gates.

LSTM is made up of cell states that transport data across LSTM units (Siami-Namini & Namin, 2018), and three sigmoid-NN gates are used to remove or add new information to the cell state so that only the necessary information is sent onto the next unit. LSTM model accuracy was improved by (Sarkar, Sahoo, Sah, & Pradhan, 2020) and (X. Li, Wu, & Wang, 2020), who found that basic sentiment analysis approaches including VADER, SentiWordNet 3.0, and SenticNet 5 paired with quantitative stock price data and technical indicators improved LSTM model accuracy, (X. Li et al., 2020).
These are the three gates:

i. **Forget Gate**: The sigmoid NN returns values ranging from 0 to 1, indicating which information should be eliminated. The number 0 indicates that the information should be discarded, whereas the value 1 suggests that the information should be kept.

ii. **Memory Gate**: This gate is made up of two layers: sigmoid and tanh, which aid in determining what fresh information should be added to the cell state. The sigmoid layer determines which values need to be altered, and the next layer generates vector representations of the new values.

iii. **Output Layer**: This gate determines whether pieces of information from the cell state should be output. The sigmoid layer first determines the values of the cell state to output before passing them to the next layer, which outputs the final cell state.

It’s possible that the many benefits supplied by LSTM models are responsible for their growing popularity in the area of stock market forecasting. According to [S. Chen, 2019](https://colah.github.io/posts/2015-08-Understanding-LSTMs/), it is typically more advantageous to adopt a stacked LSTM model than a single-layer LSTM model. This is because single-layer LSTM models need more computational time and power. In addition, the more comprehensive the model is, the more complicated the information that can be learned from the data.
BiLSTM

Deep-Bi-directional LSTMs \citep{Siami-Namini2019, Schuster1997} are an extension of the mentioned LSTM models that use two LSTMs on the input data. The input sequence is subjected to an LSTM in the first round (i.e., forward layer). The reverse version of the input sequence is fed into the LSTM model in the second cycle (i.e., backward layer). Using the LSTM twice improves learning long-term relationships and hence improves model correctness, according to \citep{Siami-Namini2019, Baldi1999}.

By merging two hidden LSTM layers with opposing orientations into a single LSTM cell, Schuster and Paliwal devised Bi-directional recurrent neural networks (BRNN) to solve the single LSTM cell’s limitation of only being capable of capturing prior context but not future context. This structure allows the output layer to incorporate relevant information from previous and future contexts. The input sequence is analyzed by a BiLSTM as mentioned in \citep{Aghdam2021}.

\[ x = (x_1, x_2,..., x_n) \]

\[ \vec{h}_t = (\vec{h}_1, \vec{h}_2,..., \vec{h}_n) \] and a backward hidden sequence \[ \overrightarrow{h}_t = (\overrightarrow{h}_1, \overrightarrow{h}_2,..., \overrightarrow{h}_n) \]. The encoded vector is formed by the concatenation of the final forward and backward outputs, \[ y_t = [\overrightarrow{h}_t, \vec{h}_t] \].

\[ \overrightarrow{h}_t = \sigma(W_{\overrightarrow{h}_x} x_t + W_{\overrightarrow{h}_{\overrightarrow{h}}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}), \]

\[ \vec{h}_t = \sigma(W_{\vec{h}_x} x_t + W_{\vec{h}_{\vec{h}}} \vec{h}_{t+1} + b_{\vec{h}}), \]

\[ y_t = W_{\overrightarrow{h}_y} \overrightarrow{h}_t + W_{\vec{h}_y} \vec{h}_t + b_y \]

where \[ y = (y_1, y_2,..., y_t,..., y_n) \] is the output sequence of the first hidden layer.

Figure 2.3 shows the flow of information from backward and forward levels. Bi-LSTM is often used when sequence-to-sequence operations are required. Text categorization, voice recognition, and forecasting models may all benefit from this kind of network.
As proven in (Q, L, L, & Z, 2020), a deep learning end-to-end model with one BiLSTM layer proved capable of managing stochastic flow characteristics and overcoming overfitting issues when used to estimate future traffic flows. Another study used a combination of BiLSTM and LSTM models to predict traffic speeds throughout the whole network. The stacked model beats both the BiLSTM and the LSTM models, according to (Cui, Ke, Pu, & Wang, 2018).

BiLSTM was used by a number of researchers including (Xu, Chhim, Zheng, & Nojima, 2020; Q. Chen, Zhang, & Lou, 2020; Lu, Li, Wang, & Qin, 2020), in order to address the problem of time series prediction. The final prediction made by the BiLSTM network is more accurate than the one made by the Uni-directional LSTM. This is due to the BiLSTM network’s ability to include information from both the past and the future. Previous research (Zengjian et al., 2017; C. Wang, Yang, Bartz, & Meinel, 2016; Tengfei, Shuangyuan, Baomin, & Hongfeng, 2018) has shown that stacking several BiLSTM in neural networks has the potential to improve classification or regression performance. Bi-directional LSTMs were used in (Zhaowei, Haitao, Zhihui, & Tao, 2022) to minimize the uncertainty in deep network learning and forecasting for traffic flow stochasticity based on the network’s forward and backward contexts approximation. This was accomplished by using the network’s forward and backward contexts approximation.

2.6 Model Optimization

2.6.1 Adam Technique

**Optimization:** Optimization is a procedure that attempts to decrease network error. This is critical for increasing the model’s accuracy. Stochastic Gradient Descent (SGD), Nesterov accelerated gradient, Adagrad, RMSProp, AdaDelta, and Adam are optimizer versions. SGD is an incremental gradient descent method that uses iterations to try to find a minimal error. Models create predictions and compare them to anticipated results in each iteration. The discrepancy between the projected and actual value is referred to as error. This error is utilized to make changes to network weights and internal model parameters. The backpropagation algorithm then follows this update approach. It does not function well with a low learning rate, because it slows down model learning, and it does not work well with a high learning rate since it may cause oscillations (Bikal, 2016). Furthermore, SGD has a difficult time avoiding the saddle spots. Adagrad, Adadelta, RMSprop, and Adam were often used to address saddle points.

The fundamental downside of the AdaGrad optimizer is that it reduces model learning capabilities by making the learning rate infinitesimally tiny (Bikal, 2016). To address the AdaGrad problem, two optimizers, RMSProp and AdaDelta, were created separately. AdaDelta and RMSProp function similarly, with the exception that Adadelta does not need an initial learning rate constant to begin with. As a result, the Adam approach is used to improve DL models in this research.

Adam is an optimization technique that is created by combining the benefits of Adaptive Gradient (AdaGrad) and Root Mean Square Propagation (RMSProp) algorithms, as described in paper (Ruder, 2016). Adam is a stochastic gradient descent variant that has recently been applied in computer vision and natural language processing as deep learning (Diederik P & Jimmy, 2014). Adam is an optimization technique that has recently been applied in computer vision and natural language processing as deep learning (Diederik P & Jimmy, 2014). Adam uses the gradient’s second average in addition to adjusting the amount of learning parameters based on the mean, as
opposed to modifying the amount of learning parameters based on the mean as is
done in RMSProp (uncentered variance). According to (Diederik P & Jimmy, 2014),
the method computes exponential moving averages of gradients and squared gradi-
ents, and the beta 1 and beta 2 parameters govern the rate at which moving averages
decay. Adam is a nice model that may be built automatically throughout the process
of moving up in learning levels, as was mentioned before. Another advantage of using
this method is that it can withstand challenging circumstances, such as those involving
non-convex optimization and non-stationary circumstances. One method that may be
used for automatically calculating the amount of learning in SGD is called the Adam
methodology. This method was proposed by King Ma and Ba. Because every single
one of SGD’s updates follows the same pattern.

The pseudo code following explains the stage of the Adam optimization algorithm:

\begin{verbatim}
Require: \alpha: learning rate; \beta_1, \beta_2 \in [0,1): exponential decay rates for the moment estimates; f(\theta): stochastic
objective function with parameters \theta; \theta_0: initial parameter vector; \lambda \in [0,1): decoupled weight decay.
1: m_0 \leftarrow 0 (Initialize first moment vector)
2: v_0 \leftarrow 0 (Initialize second moment vector)
3: t \leftarrow 0 (Initialize timestep)
4: while \theta_t \not= converged do
5: \ t \leftarrow t + 1
6: g_t \leftarrow \nabla f_t(\theta_{t-1}) (Get gradients w.r.t. stochastic objective at timestep t)
7: m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t (Update biased first moment estimate)
8: v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 (Update biased second raw moment estimate)
9: \hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1} (Compute bias-corrected first moment estimate)
10: \tilde{v}_t \leftarrow \frac{v_t}{1 - \beta_2} (Compute bias-corrected second raw moment estimate)
11: \theta_{t+1} \leftarrow \theta_t - \frac{\alpha}{\sqrt{\tilde{v}_t} + \epsilon} \cdot \hat{m}_t (Update parameters)
12: end while
13: return \theta_t (Resulting parameters)
\end{verbatim}

Figure 2.4: Adam pseudo code \(^3\)

In the study by (Chang, Zhang, & Chen, 2018), an Adam-optimized LSTM neural
network is provided. Findings based on empirical research indicate that the Adam-
LSTM neural network performs better than other popular models now in use, including
ARIMA, ANN, hybrid ARIMA-ANN models, DE-BPNN (Density Estimation-Back
Propagation Neural Network), and DE-LSTM (Density Estimation-LSTM)

\(^3\)https://www.asapp.com/blog/addressing-instabilities-for-few-sample-bert-fine-tuning/
2.7 Research Gaps

There has been intensive research in the segment of stock price forecasting by several scholars and researchers that highlight the use of different conventional approaches such as ARIMA, variable demonstration, linearity estimations, and other approaches. The use of these methodologies provides an estimation of future movement of stock prices in the market but has certain limitations regarding the use of previous data. Issues are also faced in making accurate estimations owing to the inability to assess the economic significance and provide valid outcomes. To overcome these problems, DL-based models such as LSTM are used widely for stock market prediction. Deep learning has achieved good performance in short-term forecasting recently. However, the stochasticity and distribution imbalance are main characteristics to real time forecasting, and these will bring the uncertainty and induce the network overfitting problem during deep learning which needs to be addressed (Zhaowei et al., 2022). There are certain gaps which needs to be addressed:

- Existing studies have mainly focused on the implementation of a single LSTM model and the implementation of other LSTM models such as Bi-directional LSTM and hierarchical Bi-LSTM are not focused much.

- Various existing studies have considered technical indicators for on stock market prediction, but haven’t used any feature extraction techniques to avoid data dimensionality problem.

- Most of the existing research works do not focus on optimization of DL-models to enhance the over-all trading performance

- Not many existing works based on these models considered both investor sentiments and market sentiments.

- Majority of the existing works operate in offline mode and are not capable of predicting real-time stock data, as these models are overfitted with using stock close price which is only available during market non working hours.
2.8 Overview

This chapter evaluated and analyzed many pieces of literature in the subject of stock market prediction in order to better understand the technique used and to improve prediction outcomes. Figure 2.1 depicts the categories of data employed (technical and basic analysis), feature extraction (PCA), emotional analysis (Lexicon based and Transformers and Language based), predictive analysis, statistical model (ARIMA), and deep learning models (LSTM, BiLSTM), and ultimately Adam optimization. Each category’s approaches and strategies are investigated and evaluated in the literature. Finally, gaps in current studies are discovered based on the analysis, which this study has filled.
Chapter 3

Experimental design and methodology

There are a number of elements that impact stock market movements, such as News articles, Tweets, stock technical indicators, and historical stock prices, examined in this research. Bi-LSTM and LSTM for non-linear data are tested to see how well they can forecast stock market changes in the future. The CRISP-DM approach is discussed in this chapter, which is a well-known data analysis tool. In a variety of sectors, it has shown to be a trustworthy and effective method of data analysis. One of the most popular data mining models is the CRISP-DM. Six phases are commonly included in the data mining process. In order to create a thorough project, the developer uses an iterative process like the one shown in Figure 3.1.

Below is a brief description of each stage:

i. Business Understanding: This phase involves understanding the business goal, as well as the technical and business objectives necessary to achieve the goal. This is where the project’s requirements are gathered, such as the technology to be used, software requirements, data requirements, and devising project limitations and contingency plans in case of risks. Chapters 1 and 2 described business goals, objectives to be achieved, previous literature searches to understand gaps, ways in which this study is more useful, and the scope and limitations of this study.
ii. **Data Understanding:** This stage entails comprehending the data required to achieve the goals established in the previous stage, which will be discussed in detail in Section 3.2. Data comprehension is divided into three stages:

a. **Data Collection:** The data is collected from various sources specified in the project requirements and loaded onto software or a tool.

b. **Data Description:** The collected data is described in terms of its type, format, number of records, number of features, and any other data properties.

c. **Data Exploration:** Data is explored in greater detail using visualization or statistical tools to learn about the distribution of features, and statistical analysis is performed to generate insights that aid in the transformation of data for use in the modeling stage.

d. **Data Quality:** The data is checked for completeness, correctness, and errors in the observation that must be addressed during the data preparation stage.

iii. **Data Preparation:** The quality of the data is critical to the model’s output, hence this is a critical step. Section 3.3 discussed about Data preparation strategies

---

Figure 3.1: CRISP-DM methodology

---

https://www.datascience-pm.com/crisp-dm-2/
of this study. Sub-stages are divided into three categories as follows:

a. **Data Selection**: The information gathered in the previous step is narrowed down to only the features and records necessary to complete the project.

b. **Data Cleaning**: As part of the data exploration process, the selected data is cleaned by eliminating inconsistencies, formatting the data types, and applying the proper imputed values to missing values.

c. **Data Construction**: Modifying or creating new features that weren’t there when the data was collected, but are necessary for the experiments; example, technical indicators in this research.

d. **Data Integration**: Multiple data records acquired from various sources are combined according to the necessity in order to generate merged datasets.

iv. **Modeling**: Models are built in this stage using the transformed and cleaned data from the previous stage. Models and their parameters used in this project are discussed in Section 3.4, and their implementation is discussed in Chapter 4. The sub-sections are as follows:

a. **Model Selection**: In order to begin, you must select a model that is compatible with the model assumptions.

b. **Generate Test Design**: Data splitting strategies are utilized based on model selection since the performance and quality of a model are dictated by the test dataset.

c. **Model Building**: The training, validation, and testing sets are used to build the models.

d. **Assess Model**: The findings and domain knowledge are used to evaluate the models’ performance. Models with varied parameters are constructed and their performance evaluated in order to build the best performing models in this iterative process.

v. **Model Evaluation**: In this phase, the optimal model for the company’s needs is selected. Model assessment strategies are discussed in Section 3.5, while Chapter 5 focuses on model evaluation and findings in great depth. The following sections are included:
a. **Evaluate Results:** The final model is selected based on the performance of the models in the preceding stage of the process.

b. **Review Process:** All completed steps are reviewed thoroughly in order to discover any omitted or improperly executed tasks.

c. **Determine Next Step:** It is decided whether or not more iterations are necessary once the findings of the review and modelling phases have been satisfactory and have accomplished the initial business aim.

vi. **Deployment:** CRISP-DM methodology’s last step involves deploying the project to clients via a variety of methods. As a result, this step has been omitted from this project.

### 3.1 Design Solution

Figure 3.2 depicts the design solution for this study.

![Figure 3.2: Design Solution](image-url)
3.2 Design Requirements

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Key Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Historic Stock Price</td>
<td>DataReader 0.10.0</td>
</tr>
<tr>
<td>News Article Extraction</td>
<td>GoogleNews 1.5.1</td>
</tr>
<tr>
<td></td>
<td>Newspaper 0.1.0.7</td>
</tr>
<tr>
<td>Tweets Extraction</td>
<td>twint 2.1.21</td>
</tr>
<tr>
<td>Technical Indicators Computation</td>
<td>ta-lib 0.4.19</td>
</tr>
<tr>
<td>Text Processing</td>
<td>spacy 2.3.5</td>
</tr>
<tr>
<td>Quantitative data processing</td>
<td>scikit-learn-0.23.2</td>
</tr>
<tr>
<td></td>
<td>numpy 1.19.4</td>
</tr>
<tr>
<td></td>
<td>pandas 1.1.5</td>
</tr>
<tr>
<td>Plots for visualization</td>
<td>wordcloud 1.8.1</td>
</tr>
<tr>
<td></td>
<td>matplotlib 3.3.3</td>
</tr>
<tr>
<td></td>
<td>seaborn 0.1.1.0</td>
</tr>
<tr>
<td>Data Splitting</td>
<td>TimeSeriesSplit</td>
</tr>
<tr>
<td>BERT Model Building</td>
<td>transformers 4.0.1</td>
</tr>
<tr>
<td>LSTM Model Building</td>
<td>tensorflow.keras.layers</td>
</tr>
<tr>
<td></td>
<td>tensorflow.keras.models</td>
</tr>
<tr>
<td></td>
<td>keras.callbacks</td>
</tr>
<tr>
<td>BiLSTM Model Building</td>
<td>tensorflow.keras.layers</td>
</tr>
<tr>
<td></td>
<td>tensorflow.keras.models</td>
</tr>
<tr>
<td></td>
<td>keras.callbacks</td>
</tr>
<tr>
<td>System Used</td>
<td>Google Colab with GPU Tesla K80</td>
</tr>
</tbody>
</table>

Table 3.1: Design and system requirements

3.3 Data Understanding

This section explains the collection and investigation of three types of data sources for the firms Google, Apple, Facebook, and Amazon: historical stock price data, News articles, and stock Tweets. To account for recent occurrences like "The COVID-19 epidemic," "Joe Biden Winning the Presidency," and "China Asserting Its Role as Global Leader," all of the data was gathered between January 1, 2020, and December 31, 2021.
3.3.1 Data Collection

i. Historic Stock Price: A total of 505 entries for each company were exported as a csv file, representing the daily NASDAQ stock values for Google, Apple, Facebook, and Amazon. The data was collected via the Yahoo Finance website, with the date range specified as 01/01/2020 to 31/12/2021.

ii. News Article Extraction: Most News APIs include a limit on the amount of articles that can be retrieved, so this research used Google News library. A search term of "GOOG or Google," "AAPL or Apple," "FB or Facebook," and "AMZN or Amazon" is sent to the Google News library, which extracts News Articles from News websites for the data range of 01/01/2020 to 31/12/2021. Article’s title and links are saved into a data-frame. Each of these links is further parsed using the Newspaper library’s Article() method, which returns the article’s title, full text, and publication date. A csv file containing 471 Google records, 1420 Apple records, 3128 Facebook records, and 1785 Amazon records is stored weekly to collect data for 2020 and 2021.

iii. Tweets Extraction from Twitter: Twint, the open source library, which scrapes Tweets with no restrictions, is used since Twitter’s official Twitter API for Developers only provides access to the latest seven days’ of data, but for this research Tweets from two years was needed. The query specifies the username, keyword, and date range from which the Tweets should be collected. Twitter accounts such as @stocktwits, @marketwatch, @ibdinvestors, @wsjmarkets, and @cnbc were used to extract Tweets containing the terms "GOOG or Google," "AAPL or Apple," "FB or Facebook," and "AMZN or Amazon". To extract the most Tweets out of a single day, they are gathered over a 15-day period. A total of 1003 records for Google and 2200 records for Apple, 1578 records for Facebook and 2127 records for Amazon are exported to a csv file, which includes the id and conversation id, date, time and timezone of the creation of the conversation, user id, username and location of the tweet, tweet’s language, mentions, urls, number of replies, reTweets, likes, cashtags, retweet, quote url, video and thumbnail features.
3.3.2 Data Description

The data collected is from various sources, and each raw data collection has different features and data types; a few of the most important features are shown in Table 3.2 below.

<table>
<thead>
<tr>
<th>Data</th>
<th>Features</th>
<th>Feature Description</th>
<th>Feature Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock Price</td>
<td>Date</td>
<td>Date when the market is open</td>
<td>Timestamp</td>
</tr>
<tr>
<td></td>
<td>Open</td>
<td>Price of Apple stocks when market opened</td>
<td>Float</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>Highest price of Apple stocks in a day</td>
<td>Float</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Lowest price of Apple stocks in a day</td>
<td>Float</td>
</tr>
<tr>
<td></td>
<td>Close</td>
<td>Price of Apple stocks when market closed</td>
<td>Float</td>
</tr>
<tr>
<td></td>
<td>adjusted close</td>
<td>Closing price of Apple stocks after considering all applicable splits and dividend distributions</td>
<td>Float</td>
</tr>
<tr>
<td></td>
<td>Volume</td>
<td>Number of Apple shares traded in a day</td>
<td>Float</td>
</tr>
<tr>
<td>News Article</td>
<td>Date</td>
<td>Date when News article was posted</td>
<td>Object</td>
</tr>
<tr>
<td></td>
<td>Media</td>
<td>News source of the article</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Title</td>
<td>Title of the News article</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Article</td>
<td>Complete News article</td>
<td>Object</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>Brief summary of News article</td>
<td>Object</td>
</tr>
<tr>
<td>Stock Tweets</td>
<td>Date</td>
<td>Date when tweet was posted</td>
<td>Object</td>
</tr>
<tr>
<td></td>
<td>Data-item-id</td>
<td>Unique identifier assigned for each tweet</td>
<td>Numeric</td>
</tr>
<tr>
<td></td>
<td>data-conversation-id</td>
<td>Unique identifier assigned for retweet</td>
<td>Numeric</td>
</tr>
<tr>
<td></td>
<td>Username</td>
<td>Username from which Tweets are extracted</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Tweet</td>
<td>Actual tweet of maximum 280 characters</td>
<td>Object</td>
</tr>
<tr>
<td></td>
<td>Date</td>
<td>Date when the tweet is posted</td>
<td>Object</td>
</tr>
</tbody>
</table>

Table 3.2: Data description

The date field is a characteristic that is common for all three of these datasets, and it is one of the crucial features, as this study utilizes the Date field in order to integrate all of the datasets into a single row of a certain dimension or data point in one day. It can be seen from Table 3.2 that the Date field in the historic prices of stocks are shown in Timestamp format, however in News articles and stock Tweets, the relevant information is presented as an Object type.
3.3.3 Data Exploration

i. Stock Price Data: According to the literature review in Chapter 2, the purpose of this study is not to anticipate the close price but rather the adjusted close price of the stocks held by Google, Apple, Facebook, and Amazon. The adjusted close price was chosen as the predictor variable because it is the adjusted closing price after the market has closed, whereas the stock’s closing price is the raw price just before the market closes. Consequently, the adjusted close price is more indicative of future price movements than the stock’s closing price. The negative trend seen in the stock market during the early part of the Pandemic is seen in Figure 3.3.

![Graphs showing adjusted close price trend from 2020 Jan to 2021 Dec for Google, Apple, Facebook, and Amazon.](image)

Figure 3.3: adjusted close price trend from 2020 Jan to 2021 Dec

The data description did not include any missing values. The statistical analysis showed that the open, close, high, and low values all have nearly identical scales, but the Adj close value for Volume has a completely different scale of values. Because of this, the results of the prediction will be affected, and the data will need to be normalized as described in Quantitative Data Transformation, Section 3.4.2 (ii).
ii. News and Tweet Article Data: Tweets and News stories have very similar text structures; both include punctuation, website names, tags, and many spaces in their bodies of text. The procedures involved in preprocessing the text are outlined in Section 3.4.2 (iii). These stages include managing contractions and deleting punctuation, digits, and special characters.

3.4 Data Preparation

In this phase, both quantitative and qualitative data are pre-processed from the very beginning to the very end. This section covers the transformation of data, the preprocessing of textual data (News articles and Tweets), the computation of technical indicators, and the integration of datasets into new datasets, all of which are put to use in the experiments described in Chapter 4.

3.4.1 Data Selection

This information consists of the predictor adjusted closing price, the value of which is derived from the other five components of the historic stock price data, low, high, open, close and volume. Because of their striking similarity to the prediction indicator adjusted close, the modeling process begins with the selection of four stock price data features, low, high, open and volume. This research doesn’t not consider close price, as the model is expected to work in offline and real time mode. And close price is not available in real time mode, and to update model during real time would not be possible if close price is included in model fitting. Since the post’s title is inadequate for accurate sentiment analysis due to the fact that the whole article’s context can only be understood by reading the entire post, only the 'Date' and 'Article' features from the News article dataset are employed. Because the 'Username' feature in the Twitter dataset would not contribute significantly to the accuracy of the stock prediction, just the 'Date' and 'Tweets' characteristics were chosen as the final ones to be included in the analysis.
3.4.2 Data Cleaning

i. Handling Missing Data and Duplicate entries:

Because of the fact that the stock market is closed over the weekend and on major holidays. The following is a list of several ways\(^5\) that may be used to deal with missing data from datasets.

a) Only use common points - Remove all holidays from any index
   - This decreases the sample size.
   - Information loss
   - There is no fabricated data (consistency)

b) Fill forward
   - The problem here is that fluctuations in the market throughout holidays are reported as having no change, which is then followed by a significant spike in change.

c) Linear interpolation - Interpolate the price as a function of time linearly
   - This helps with the jumps in the fill forward method, but it may result in an increase in serial correlation of returns as a side effect of the method.

d) Resampling:
   - resampling the joint distribution of returns using indices in a window that is centered around the gap that has to be filled empirically. Find a nearby day that is as similar as possible in terms of all indices being available in change space (return space), and use that as the change from the previous price in the missing index. In essence, look for a nearby day that is as similar as possible in terms of all indices being available in change space (return space).
   - The issue here is that there may not be a day in the dataset that is very similar.
   - At this point, it also makes sense to be ”making up” data.
   - Should have comparable distributional properties

e) Parametric sampling
- Fit the available data to the joint distribution of returns and sample from it to fill the points.
- As a result of the inherent randomness of the cosmos, it is possible that an atypical sample will be obtained. You could fill with the mean to offset this, but doing so would reduce the variation in the data.

Fill forward was utilized in this study to fill in the missing data in the stock price dataset. To solve the problem of a price spike from zero to high, which would ultimately give inaccurate Adj close price, this study analyzed market sentiment and the stock price to anticipate the adjusted closing value. This study have taken into account the polarity of News articles and Tweets to account for any significant shifts in market sentiment. The price increase will be offset if the aggregate emotion polarity is neither very positive nor extremely negative. In addition, fewer than 1% of the population’s data points are deleted from the dataset when neither stock prices nor sentiment values are available. To get the most data possible for a single day, News articles are scraped every month, while Tweets are scraped every 15 days, resulting in a large number of identical entries that were then removed from final datasets.

ii. Quantitative Data Transformation: Since the scales of the stock price data are not uniform, and since the LSTM and BiLSTM models would provide biased predictions if the data are not scaled, it is necessary to rescale the data. The two methods of data transformation known as standardization and normalization are the most common and frequently utilized of all of them. If the model needs data to be normally distributed, then the data are rescaled such that they have a mean of 0 and a standard deviation of 1, although this is only done if it is necessary. On the other hand, normalization involves rescaling the data between 0 and 1, and it is used in situations when the model does not need regularly distributed data.
Both approaches of rescaling are put to use in this research. It is not necessary to have data that is normally distributed in order to use the LSTM or BiLSTM models; however, in order to use principal component analysis (PCA) on few datasets that contain technical indicators, it is necessary to have data that is normally distributed, with the mean value being 0 and the standard deviation being 1. As a consequence of this, Standardization is implemented when using PCA, and then Normalization is performed on the dataset that was reduced by PCA before it is utilized as input for the LSTM and BiLSTM models. Before feeding data to the models, the StandardScaler function from the sklearn preprocessing package is used when PCA is employed, and the MinMaxScalar function from the scikit learn preprocessing package is used to normalize the data so that it has values that range between 0 and 1 before fitting the models.

Figure 3.4 shows the cleaned and normalized adjusted close price of Google, Apple, Facebook and Amazon stocks.

![Figure 3.4: Normalized adjusted close price trend from 2020 Jan to 2021 Dec](image)
iii. Text Pre-Processing:

a) Handling Contractions

A phrase or term in English that has been reduced to its shortened form is known as a contraction. I'm, for instance, is a condensed form of the word I am. These kinds of contractions may commonly be seen in evaluations made by humans. In order to standardize the evaluations that take place before training the model, these contractions will first be enlarged. Python's contractions library is used in order to remove these contractions.

b) Removing punctuations, numbers, and special characters

The News articles and Tweets consists of punctuation, emojis, special characters, numerals, HTML elements, and undesirable white space. These components produce noise that isn't essential, thus in order to get the most possible performance out of everything, these numbers have been eliminated from the text. The lambda expression and the isdigit() function in Python are used in order to strip text of any numerical characters. Python is used to import the 're library' so that the residual sounds may be removed, and then regular expressions are used to clean the data (regex). The cleaned text is then utilized for further analysis once it has been recorded in a new column in the dataframe. The dataset is investigated to see whether or not there are any duplicate text; if such text are discovered, they are deleted from the dataframe. For this project, transformer-based pre-trained models like BERT, RoBERTa, DistilBERT, and ALBERT will be used. These models make use of self-attention processes to learn their function after being pre-trained with a substantial quantity of data. As a direct consequence of this, conventional text preparation strategies such as the elimination of stopwords, stemming, and lemmatization are not used in order to keep the semantic content of text intact. Tokenizing text before training the model required the use of model-specific tokenizers.
CHAPTER 3. EXPERIMENTAL DESIGN AND METHODOLOGY

Figure 3.5 and Figure 3.6 shows the word cloud representing the top 100 words occurring in News articles and Tweets.

(a) Google

(b) Apple

(c) Facebook

(d) Amazon

Figure 3.5: Most frequent words in the Twitter from 2020 Jan to 2021 Dec

(a) Google

(b) Apple

(c) Facebook

(d) Amazon

Figure 3.6: Most frequent words in the News from 2020 Jan to 2021 Dec
3.4.3 Technical Indicators Computation

Another source of data used in this study is technical indicators, which have shown a significant influence in predicting stock prices in previous works of literature (Oriani & Coelho 2016; M. Agrawal, Khan, & Shukla 2019). There are many technical indicators, and this study used 23 technical indicators as described in Table 3.3. The Python ta-lib library provides a wide range of functions for computing technical indicators, which are used in this study. In general, the window size for computing each of the indicators is set to 20 days.

Technical Indicators are mathematical calculations performed on various stock parameters such as Volume, open, high, and so on. They assist us in identifying various patterns that a stock follows or will follow at a given time.

‘ta-lib’, short for "Technical Analysis Library," is a sophisticated Python financial quantification library that includes over 150 indicators. It is an open-source Python library that is used to analyze historical stock market data such as share price, volume, and so on in order to predict future prices or market direction to aid in investment decisions. ta-lib includes a wide range of technical indicators that were used in this study.

i. Installing ta-lib Python Library

   ta-lib is installed using pip install as given below,

   \texttt{pip install ta-lib}

ii. Creating Technical Indicators using ta-Lib

   ta-lib is used to create various technical indicators. Overlap studies, momentum indicators, cycle indicators, price transform, volume indicators, volatility indicators, statistical functions, pattern recognition, and mathematics transform are the 10 subsections of "ta library". This study will only look at the technical indicators from the Overlap Studies and statistic Functions groups because they cover important indicators of stock trends. Because adjusted close price are the target feature, all technical indicators are calculated for adjusted close price with a window size of 20 days.

\footnote{https://mrjbq7.github.io/ta-lib/}
Below are the technical indicators that are used in this study,

<table>
<thead>
<tr>
<th>Technical Indicator</th>
<th>Description</th>
<th>Math Formula</th>
<th>Python calculation for 20 days timeperiod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bollinger Bands (BBANDS)</td>
<td>Bollinger Bands consist of three lines. The middle band is a simple moving average (generally 20 periods) of the typical price (TP). The upper and lower bands are F standard deviations (generally 2) above and below the middle band. The bands widen and narrow when the volatility of the price is higher or lower, respectively. Bollinger Bands do not, in themselves, generate buy or sell signals; they are an indicator of overbought or oversold conditions. When the price is near the upper or lower band it indicates that a reversal may be imminent. The middle band becomes a support or resistance level. The upper and lower bands can also be interpreted as price targets. When the price bounces off of the lower band and crosses the middle band, then the upper band becomes the price target.</td>
<td>( TP = \frac{\text{high} + \text{low} + \text{close}}{3} ) ( \text{MidBand} = \text{SimpleMovingAverage}(TP) ) ( \text{UpperBand} = \text{MidBand} + F \times \sigma(TP) ) ( \text{LowerBand} = \text{MidBand} - F \times \sigma(TP) )</td>
<td>upper, middle, lower = talib.BBANDS(closing_prices, timeperiod=20, nbdevup=2, nbdevdn=2, matype=0)</td>
</tr>
<tr>
<td>Double Exponential Moving Average (DEMA)</td>
<td>The DEMA is a smoothing indicator with less lag than a straight exponential moving average. DEMA is an acronym for Double Exponential Moving Averages, but the calculation is more complex than just a moving average of a moving average.</td>
<td>( \text{DEMA} = 2 \times \text{EMA(input)} - \text{EMA}(\text{EMA(input)}) )</td>
<td>dema = talib.DEMA(closing_prices, timeperiod=20)</td>
</tr>
<tr>
<td>Exponential Moving Average (EMA)</td>
<td>The Exponential Moving Average is a staple of technical analysis and is used in countless technical indicators. In a Simple Moving Average, each value in the time period carries equal weight, and values outside of the time period are not included in the average. However, the Exponential Moving Average is a cumulative calculation, including all data. Past values have a diminishing contribution to the average, while more recent values have a greater contribution. This method allows the moving average to be more responsive to changes in the data.</td>
<td>( K = \frac{2}{n + 1} ) ( \text{EMA} = \text{EMA}(-1) + K \times (\text{input} - \text{EMA}(-1)) ) or ( \text{EMA} = K \times \text{input} + (1 - K) \times \text{EMA}(-1) )</td>
<td>ema = talib.EMA(closing_prices, timeperiod=20)</td>
</tr>
<tr>
<td>Hilbert Transform - Instantaneous Trendline (HTTRENDLINE)</td>
<td>The Hilbert Transform is a technique used to generate inphase and quadrature components of a detrended real-valued “analytic-like” signal (such as a Price Series) in order to analyze variations of the instantaneous phase and amplitude. HTTrendline (or MESA Instantaneous Trendline) removes the Price Series value after the Dominant Cycle of the analytic signal as generated by the Hilbert Transform has been removed. The Dominant Cycle can be thought of as being the range of 10 to 40 of a sine function of the Price Series. The HTTrendline at a specific bar gives the current Hilbert Transform Trendline as instantaneously measured at that bar. In its Series form, the Instantaneous Trendline appears much like a Moving Average, but with minimal lag compared with the lag normally associated with such averages for equivalent periods. The HTTrendline is formed by removing the Dominant Cycle from the Price Series.</td>
<td>The Hilbert transform of a function ( u(t) ) is given by, ( H(u(t)) = \frac{1}{\pi} \text{p.v.} \int_{-\infty}^{\infty} u(\tau) \frac{t - \tau}{\tau^2} d\tau )</td>
<td>htline = talib.HTTRENDLINE(closing_prices)</td>
</tr>
<tr>
<td>Kaufman Adaptive Moving Average (KAMA)</td>
<td>KAMA is an adaptive moving average, and uses the noise level of the market to determine the length of the trend required to calculate the average. Due to its adaptive nature, the KAMA is more flexible to calculate ( 15 ). The first step is to calculate the market’s efficiency ratio (ER). The efficiency ratio (ER) is defined as the absolute net change in price divided by the absolute sum of the individual price changes over that period. The ER is a measure of a market’s noise level, and oscillates between 0 and 1. With the ER, the smoothing constant (SC) can now be calculated. The SC is calculated for each period, as follows: ( \text{SC} = \frac{\text{ER} \times (\text{fastest} - \text{slowest}) + \text{slowest}}{2} )</td>
<td>kama = talib.KAMA(closing_prices, timeperiod=20)</td>
<td></td>
</tr>
<tr>
<td>Moving Average (MA)</td>
<td>A moving average (MA) is a stock indicator that is commonly used in technical analysis. The reason for calculating the moving average of a stock or to help smooth out the price data by creating a constantly updated average price.</td>
<td>Moving Average = ( \frac{\text{closing prices} \times n}{N+1} ) ( \text{Where,} ) ( C1, C2...Cn ) stands for the closing prices. ( N ) stands for the number of periods for which average is required to be calculated.</td>
<td>ma = talib.MA(closing_prices, timeperiod=20, matype=0)</td>
</tr>
</tbody>
</table>
## CHAPTER 3. EXPERIMENTAL DESIGN AND METHODOLOGY

<table>
<thead>
<tr>
<th>Technical Indicator</th>
<th>Description</th>
<th>Math Formula</th>
<th>Python calculation for 20 days time period</th>
</tr>
</thead>
<tbody>
<tr>
<td>MidPoint Price (MIDPRICE)</td>
<td>The MidPoint calculation is similar to the Midprice, except the highest and lowest values are returned from the same input field. The default indicator calculates the highest high and lowest low within the look back period and averages the two values.</td>
<td>( \text{MIDPRICE} = \frac{\text{HighestHigh} + \text{LowestLow}}{2} )</td>
<td>midprice = talib.MIDPRICE(closing_prices, timeperiod=20)</td>
</tr>
<tr>
<td>Pearson's Correlation Co-efficient (CORREL)</td>
<td>Correlation Analysis compares two array or two samples of data to show you if one sample of data can predict the other. You can analyze correlation between a stock against another stock or a stock against an indicator. Correlation between an indicator and a stock. A value above 0.7 tells you that a change in the indicator will usually predict a change in the security's price. A value below -0.7 tells you that a change in the indicator will usually predict a move of the stock in the opposite direction. A value near 0 tells you that there is no relationship between the security's price and the indicator.</td>
<td>Correlation coefficient ( r ) where ( r ) is the Pearson correlation coefficient between the two variables.</td>
<td>correl = talib.CORREL(High, Low, timeperiod=20)</td>
</tr>
<tr>
<td>Beta (BETA)</td>
<td>The Beta is a measure of how an individual asset moves (on average) when the overall stock market increases or decreases. Thus, Beta is a useful measure of the contribution of an individual asset to the risk of the market portfolio when it is added in small quantity.</td>
<td>Beta formula ( \beta = \frac{\sigma_X \sigma_Y}{\sigma_X} ) where ( \sigma_X ) is the return on the stock and ( \sigma_Y ) is the return on a benchmark index.</td>
<td>beta = talib.BETA(closing_prices, timeperiod=20)</td>
</tr>
<tr>
<td>Weighted Moving Average (WMA)</td>
<td>The Weighted Moving Average (WMA) places more emphasis on recent prices than on older prices. Each period's data is multiplied by a weight, with the weighting determined by the number of periods selected.</td>
<td>Weighted Moving Average (WMA) of time series ( X ):</td>
<td>wma = talib.WMA(closing_prices, timeperiod=20)</td>
</tr>
<tr>
<td>Linear Regression (LINEARREG)</td>
<td>The LinearRegression function calculates the slope and angle of a linear regression line and allows you to identify the price where the projected line crosses a future (or past) bar position.</td>
<td>Linear regression ( y = ax + b )</td>
<td>lineareq = talib.LINEARREG(closing_prices, timeperiod=20)</td>
</tr>
</tbody>
</table>
CHAPTER 3. EXPERIMENTAL DESIGN AND METHODOLOGY

<table>
<thead>
<tr>
<th>Technical Indicator</th>
<th>Description</th>
<th>Math Formula</th>
<th>Python calculation for 20 days time-period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance (VAR)</td>
<td>Variance is a measure of the spread between values in a data set. It is calculated by using the following formula:</td>
<td>$\sigma^2 = \frac{\sum (x_i - \mu)^2}{N}$ where $\sigma^2$ = variance, $x_i$ = value of the data point, $\mu$ = mean of all data points, $N$ = number of data points.</td>
<td>$\text{var} = \text{talib.VAR(close, timeperiod=20)}$</td>
</tr>
<tr>
<td>Time Series Forecast (TSF)</td>
<td>The Time Series Forecast (TSF) is a linear regression calculation that plots each bar’s current regression value using the least square fit method. This indicator is sometimes referred to as a moving linear regression similar to a moving average. For example, the TSF value that covers 10 data bars will have the same value as a 10 bar Time Series Forecast. This differs slightly from the Linear Regression indicator in that the Linear Regression indicator does not add the slope to the ending value of the regression line.</td>
<td>$y = ax + b$ where $y$ = the price for each date $a$ = the intercept for constant, where the value of $a$ is 0 $b$ = slope of the regression line $x$ = the date for each value.</td>
<td>$\text{tsf} = \text{talib.TSF(close, timeperiod=20)}$</td>
</tr>
<tr>
<td>Standard Deviation (STDDEV)</td>
<td>The Standard Deviation function fills the output array with the standard deviation of the last $n$ values of the input array. This function is used to measure the volatility of the market. It can take price or the output of any indicator as its input. Standard Deviation is often used as a measure of volatility.</td>
<td>$SD = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$ where $SD$ = standard deviation, $x_i$ = price for each date, $\bar{x}$ = mean of all data points, $n$ = number of data points.</td>
<td>$\text{stddev} = \text{talib.STDDEV(close, timeperiod=20)}$</td>
</tr>
<tr>
<td>Linear Regression Slope (LINEARREG_SLOPE)</td>
<td>Linear Regression Slope indicator is one of the technical indicators by using the Linear Regression (LR) technique which is applied to technical analysis. The indicator plots the intercept for the trend line.</td>
<td>The slope value $a$ is found by, [ a = \frac{\sum x_iy_i - \sum x_i \sum y_i}{\sum x_i^2 - (\sum x_i)^2} ] where $x_i$ = date for each value $y_i$ = the price for each date $\sum$ denotes the sum of a term or a value from a set of data.</td>
<td>$\text{slope} = \text{talib.LINEARREG_SLOPE(close, timeperiod=20)}$</td>
</tr>
<tr>
<td>Linear Regression Intercept (LINEARREG_INTERCEPT)</td>
<td>Linear Regression Intercept indicator is one of the technical indicators by using the Linear Regression or LR technique which is applied to technical analysis. The indicator plots the intercept for the trend line.</td>
<td>The LRI is calculated as, [ Y = aX + b ] where $Y$ = the price for each date $a$ = the intercept for constant, where the value of $a$ is 0 $b$ = slope of the regression line.</td>
<td>$\text{intercept} = \text{talib.LINEARREG_INTERCEPT(close, timeperiod=20)}$</td>
</tr>
<tr>
<td>Linear Regression Angle (LINEARREG_ANGLE)</td>
<td>Linear Regression Angle is a directional movement indicator which identifies a trend at the moment of its birth, and additionally defines trend weakness. The indicator plots the angle between the trend and horizontal axis in terms of logarithm. The signal line is a simple average of the trend. The angle is the difference between the right and left edges of regression (in points) divided by its period. The angle value above (below) is an uptrend. The higher the value, the stronger the trend. A value below 0 indicates a downtrend. The lower the value, the stronger the downtrend.</td>
<td>The angle ( \theta ) is found by, [ \tan \theta = \frac{\sum x_iy_i - \sum x_i \sum y_i}{\sum x_i^2 - (\sum x_i)^2} ] where $x_i$ = date for each value $y_i$ = the price for each date $\sum$ denotes the sum of a term or a value from a set of data.</td>
<td>$\text{angle} = \text{talib.LINEARREG_ANGLE(close, timeperiod=20)}$</td>
</tr>
</tbody>
</table>

2. https://www.investopedia.com/terms/m/movingaverage.asp
15. https://www.investopedia.com/terms/m/movingaverage.asp

Table 3.3: Technical Indicators from Python's ta-lib
3.4.4 Data Integration

The pre-processed stock price dataset, the computed technical indicators dataset (from section 3.4.3), the technical indicators reduced with PCA dataset (from section 3.5.1) for dimensionality reduction, the final labeled News dataset (from section 5.2.2), and the final labeled stock Tweets dataset (from section 5.3.2) are merged to obtain different dataset combinations using the 'Date' feature, as this is the common feature in all of the datasets. The resulting datasets include different combinations of all the data sources collected initially.

The next sections, 4.3 to 4.11, provide a detailed description of the numerous experiments that were carried out for each of these datasets in order to predict the Adj closing price using the LSTM and BiLSTM models.

Listed below are nine dataset combinations:

i. Historic Stock price
ii. Historic Stock price and New Articles Sentiment Polarity
iii. Historic Stock Price and Stock Tweets Sentiment Polarity
iv. Historic Stock price and Technical Indicators
v. Historic Stock price and Technical Indicators with PCA applied
vi. Historic Stock Price, News articles Sentiment Polarity and Stock Tweets Sentiment Polarity
vii. Historic Stock Price, Technical Indicators with PCA applied and New Articles Sentiment Polarity
viii. Historic Stock Price, Technical Indicators with PCA applied and Twitter Tweets Sentiment Polarity
ix. Historic Stock Price, Technical Indicators Indicators with PCA applied, News Articles Sentiment Polarity and Stock Tweets Sentiment Polarity
3.5 Modeling

This section describes fine tuning pre-trained BERT family transformers with the financial News and Tweets dataset, as well as using the best performing models for sentiment analysis of unlabeled News and Tweets. The splitting of time-series datasets, along with LSTM and BiLSTM models for predicting adjusted close price are also discussed.

3.5.1 PCA

Dimensionality reduction techniques is an approach to reduce the number of variables in a dataset. The number of dimensions, features, or input variables associated with a dataset is referred to as its dimensionality. It is frequently thought of as the number of columns (except the label column, which is adjusted close in the research) in a dataset. The Figure 3.7 depicts 'Historic Stock Price and Technical Indicators Indicators,' dataset, which contains 27 features. As a result, the total number of dimensions is 27. This means, for example, the notation used to demonstrate the first data point in the four-dimensional space is, p1(75.150002, 73.797501, 74.059998, 135480400.0, 73.894318, 73.684567, 68.688135, 63.691702, 79.204298, 68.781716, 72.435024, 68.688135, 69.144756, 70.160000, 68.688135, 67.817076, 76.003786, 68.623005, 70.027602, 0.845264, 0.991178, 72.706535, 22.927854, 64.669734, 0.422990, 2.498216, 73.129525, 6.241084).

Figure 3.7: Stock Technical Features without PCA

Dimensionality reduction refers to reducing the number of features in a dataset. Dimensionality reduction algorithms transform high-dimensional data into a low-dimensional space while preserving as much variation (i.e., salient information) as possible. One method for Dimensionality reduction is principal component analysis (PCA).
The curse of dimensionality

The curse of dimensionality is a prevalent issue in machine learning. Because there are so many dimensions in the feature space, this is a problem that arises. Many Machine Learning issues need the use of tens of thousands of features for a single training instance. It’s tough to discover a viable solution because of these features, which slow down training. It’s impossible for algorithms to train successfully and efficiently on such a large feature space. The curse of dimensionality is the term used to describe this phenomenon.

Dimensionality reduction strategies may relieve dimensionality curse. Data dimensionality decreases using Machine learning algorithms to uncover more patterns in low-dimensional data.

PCA

Principal Component Analysis (PCA) is a technique for reducing linear dimensionality. It reduces a set of correlated variables (p) to a smaller k (kp) number of uncorrelated variables known as principal components while retaining as much of the original dataset’s variation as possible. The PCA’s main idea is to consider the correlation between features. If the correlation between a subset of features is very high, PCA will attempt to combine the highly correlated features and represent this data with a smaller number of linearly uncorrelated features. The algorithm continues to reduce correlations by locating the directions of maximum variance in the original high-dimensional data and projecting them onto a smaller dimensional space. Principal components are the newly derived components.

Using these components, it is feasible to re-create the original features not precisely, but near enough. The PCA algorithm actively strives to reduce reconstruction errors throughout its search for the optimal components. Because PCA decreases the dimensionality of the data, machine learning algorithms may perform better. As an unsupervised method, PCA disregards the target vector in favor of just considering the feature matrix.
Feature scaling in PCA

If there is a large difference in scale between the data’s features, feature scaling is necessary. This can happen, for example, if one feature has a range of 0 to 1 and another has a range of 100 to 1,000. PCA is extremely sensitive to the original features’ relative ranges. The StandardScaler() class from the Scikit-learn library’s preprocessing submodule is used to standardize all features into a common scale.

Using Scikit-learn for PCA

This research’s original feature space contains 27 dimensions, or p dimensions. PCA reduces data to a k-dimensional subspace (kp) while preserving variance. Figure 3.8 shows that the primary components are k=4 dimensions in this research. Some variance (i.e., information) is lost when PCA is used. PCA will reduce the size of the data by reducing its dimensionality. This improves machine learning algorithm performance, as shown in sections 5.6, 5.8, 5.10, and 5.11. This reduces the amount of hardware required and speeds up the training process. In the dataset, the original feature space has 27 dimensions, known as p dimensions. PCA will project the data onto a smaller subspace of k dimensions (where k < p) while retaining as much of the variation as possible. These k dimensions are known as the principal components.

By applying PCA\(^\text{16}\), some of the variance (i.e., information) is lost. By reducing the dimensionality of the data, PCA will reduce the size of the data.

- This will improve the performance of machine learning algorithms.
- This reduces the amount of hardware required and speeds up the training process.
- This will allow us to quickly grasp the data’s underlying structure.
- This will allow us to visualize the data on a two-dimensional or three-dimensional plot (if the number of principal components as 2 or 3 is chosen).

The values in the dataset are not equally scaled, as illustrated in Figure 3.6. As a result, z-score standardization is used to bring all features into the same scale. Scikit-learn StandardScaler() class from Scikit-learn’s preprocessing submodule is used for this.

\(^{16}\) https://towardsdatascience.com/principal-component-analysis-pca-with-scikit-learn-1e84a0c731b0
CHAPTER 3. EXPERIMENTAL DESIGN AND METHODOLOGY

Choose the right number of dimensions (k)

Before applying PCA to the dataset, the appropriate number of dimensions (i.e., the appropriate number of principal components — k) must be determined. The PCA() class in Scikit-learn is used to perform PCA. Scikit-decomposition learn’s submodule contains it. n_components is the most important hyperparameter in that class. It can accept any of the following values: None, int, or float. This study looked at float type values. This study has chosen float type value. PCA will select the number of components that will explain the amount of variance. For example, if n_components=0.95, the algorithm will choose the number of components while preserving 95% of the data variability. For the same variance, either int type value of 4 can be used.

Figure 3.8: PCA-transformed components’ cumulative explained variance

According to Figure 3.8, the first four principal components explain approximately 95 percent of the variance in the dataset. This is the total amount of information represented in comparison to the original data. Using n_components = 5 would result in 100% variance, but this would cause an overfitting problem. Figure 3.9 depicts the reduced dataset after PCA.

Figure 3.9: Stock Technical Features with PCA explained variance of 95%
3.5.2 Sentiment Analysis

Training a large language model, such as a transformer model, from scratch on small data sets will result in overfitting. As a result, manipulating a pre-trained language model that was trained on a different dataset is beneficial. Following that, the model can be fine-tuned by training it on even smaller amounts of data. The hugging face provides a diverse set of pre-trained transformer models, models from the BERT family are being used in this study. Despite being pre-trained on financial datasets, these models will fail to appropriately classify sentiment since the collected data is unlabeled and the models were not trained on financial text data. To address this, the author trained the model on two more annotated financial datasets: One dataset, the Tweets stock market dataset by (Taborda, de Almeida, Carlos Dias, Batista, & Ribeiro 2021), was collected between April 9 and July 16, 2020, using not just the SPX500 tag, but also the top 25 corporations in the index and stocks. A annotator sorted and reviewed 1300 Tweets by hand. In this investigation, further 3900 Tweets were labeled using the VADER approach. The second dataset is the Financial Phrase Bank dataset by (Malo, Sinha, Takala, Korhonen, & Wallenius 2014), which contains 4845 labeled (positive, neutral, and negative) financial headlines data.

One of the biggest drawbacks of employing lexicon-based approaches for sentiment analysis in the stock market sector is that there is no lexicon that incorporates financial terms, causing the model to misread text sentiments. This study uses pre-trained Transformers models to tackle this difficulty. The Transformers architecture was first introduced in the paper "Attention is all you need" by (Vaswani et al., 2017). Transforming robots are typically built using an encoding-decoding architecture. The system has a multi-headed self-attention mechanism that allows for learning sequences one after the other. This eliminates the need for RNN. To understand the location of tokens in a sequence, the input data is encoded in terms of position. The encoder block includes a multi-head attention layer and a feed-forward neural network, while the decoder includes an extra attention layer. Transformer enables parallelization of model training to improve performance. The decoder output is passed through a linear layer followed by a SoftMax layer for anticipating target variable.

52
i. BERT

The Google AI team introduced Bi-directional Encoder Representations from Transformers (BERT), a transformer-based pre-trained model (Devlin et al., 2018). It is pre-trained on a dataset of 2500 million words text paragraphs and 800 million words novels in the English language, both of which are available on Wikipedia. BERT uses Bi-directional learning to learn the context of text by learning right and left word context, as opposed to sequential learning of textual data performed by directional models from either right or left. BERT has shown exceptional results in various areas of natural language processing. The two tasks performed in the BERT’s pre-training phase are language modeling (MLM) and next sentence prediction (NSP). BERT is a modular architecture released in two versions: BERT Base and BERT Large.

This study used the BERT base architecture, which is comprised of 12 transformer-encoder blocks, since the memory and time requirements of the base model are fewer than those of the BERT big model. It is trained on lower-cased English text and has a vocabulary of around 30,000 tokens. Each transformer block is trained on 110M parameters and comprises 768 hidden layers and 12 self-attention heads.

ii. Distill BERT

DistilBERT is a distilled version of the BERT pre-trained model developed by (Sanh, Debut, Chaumond, & Wolf, 2019) that is lighter, cheaper, significantly smaller, and quicker than BERT. DistilBERT employs 40% fewer parameters, runs 60% quicker, and retains 97% of the model’s performance when compared to the BERT basic model. This paradigm, in addition to having minimal computational and resource needs, mitigates a few additional BERT drawbacks, such as word piece embedding and fixed input length size concerns. DistilBERT has kept half of BERT’s layers while removing pooler and token-type embeddings from the BERT architecture.

This model features sinusoidal embedding loss, masked language modeling loss, and random initialization student loss. Before training, the input text must be tokenized, converted, and padded. Distilbert-base-uncased has 6 transformers, 12 self-attention layers, 768 hidden layers, and 66 million parameters.
iii. RoBERTa

The Facebook AI team created the robustly optimized BERT method (RoBERTa), which is a pre-trained model that relies on BERT’s language masking strategy \cite{Liu2019RoBERTa}. RoBERTa surpassed BERT in various NLP tasks by using better training technique. It is trained using more data, higher mini-batch sizes, and longer sequences over longer time periods. It deleted the next sentence prediction function, hence removing the NSP loss issue that was in BERT. RoBERTa is a neural network that has been pre-trained on 160 GB of CCNews datasets and English Wikipedia. During training, it employs dynamic masking to disguise various sets of tokens at each epoch. Instead of BERT’s character-level BPE, it employs Byte-Pair Encoding (BPE) with a vocabulary size of roughly 50,000 tokens for tokenization.

The roberta-base model used to generate the News and Tweets classifier in this study contains 12 layers of transformers, 12 attention heads, 768 hidden layers, and 125 million parameters.

iv. ALBERT

A Lite BERT is a pre-trained model that is based on the BERT architecture but has been modified to overcome its weaknesses, such as longer training time and memory constraints. ALBERT, developed by \cite{Lan2019ALBERT}, employs two-parameter reduction strategies to lower GPU/TPU memory utilization while increasing the speed of the training process to produce equivalent or better outcomes than BERT. This self-supervised learning model predicts sentence order (SOP) rather than next sentence BERT prediction, which reduces NSP loss. It has been trained on the English Wikipedia as well as the Books corpus. ALBERT was able to increase the hidden size while maintaining the size of the vocabularies inclusion parameter within a specific limit, resulting in the transformation of the huge vocabularies inclusion matrix into smaller matrices for incorporating factors. 12 repeating layers, 12 attention heads, 768 hidden layers, 128 embeddings, and 11 million parameters are used in the albert-base-v1 model. Albert lessens the need for extremely high processing power, making it more practical for real-world applications.
The BERT family of transformers, which consists of BERT, DistilBERT, RoBERTa, and ALBERT, is fine-tuned with the financial phrase bank dataset and the stock market Tweets dataset, which are described in sections 4.1 and 4.2, respectively. The model that produces the best results is then saved and loaded with the News articles and Twitter Tweets datasets that were constructed in Section 3.4.2. Each text is analyzed to see if it has a positive, negative, or neutral attitude.

### 3.5.3 Splitting Time-series Data

Traditional ways of randomly partitioning datasets into 80% train and 20% test sets are not practical for time-series data. Each observation in a time-series dataset is reliant on the preceding observations, and arbitrarily dividing the dataset results in data that lacks trend and seasonality. It is also critical for any prediction model to be cross-validated in order to determine the model’s accuracy. The expanding window approach is used in this study to divide time-series data into training and test sets and to perform cross-validation. In this approach, T observations are used as the training set and T+1 observations as the test set for each fold, and subsequent sets are formed by extending the training set and establishing new test sets, as illustrated in Figure 3.10.

![Expanding Window Diagram](https://eng.uber.com/omphalos/)

Figure 3.10: Expanding Window

---

17 https://eng.uber.com/omphalos/
3.5.4 Time Series Predictive Analysis

Following the loading, preprocessing, and splitting of the datasets, LSTM and BiLSTM models are generated and fitted using all nine datasets collected in section 3.4.4. This study used six distinct architectures of the LSTM and BiLSTM models, as indicated in Table 3.4, with three different time steps of 1 day, 3 days, and 10 days. Sections 4.3 through 4.11 go into further detail.

<table>
<thead>
<tr>
<th>Number of layers</th>
<th>Number of neurons</th>
<th>Dropout after layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>150</td>
<td>0.2</td>
</tr>
<tr>
<td>6</td>
<td>75</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 3.4: Different Architectures of LSTM and BiLSTM Models

Imported from Keras sequential LSTM and BiLSTM models are used to create 6 separate LSTM models and 6 different BiLSTM models, with Mean squared error as the loss function and Adam method as the optimizer. Because the models are related with time-series data, the measure employed is mean absolute error.

The architecture details of six different LSTM/BiLSTM models are shown in Table 3.4. The first column shows the number of layers used in each model, the second column is passed as an argument to the models and states the number of units or neurons, which is the dimensionality of the output space in the models, and the last column gives details on dropout used in the models to avoid over-fitting. The first model employs a single LSTM/BiLSTM layer and accepts the return sequences=False argument. The remaining 5 models have multiple LSTM/BiLSTM layers with the argument return sequences=True because stacking LSTM/BiLSTM layers requires a three-dimensional sequence input. The last four models (last four rows) in Table 3.4 have a 20% dropout in between for regularization to avoid over-fitting. All of these models are given an additional argument, input shape, which is the shape of the training dataset. As the final layer of all these models, a Dense layer is added, which specifies an output of one unit.
3.5.5 Adam Optimization Technique

The optimization algorithm is used to finance parameter values (for example, there are models that minimize function error when mapping inputs to outputs). Machine learning is the use of an algorithm to learn and generalize from previous data in order to predict new data. The optimization problem is solved each time a machine learning algorithm is fitted with the training dataset. Adam is a novel optimization algorithm that can be used to more effectively train deep learning models. With stochastic gradient descent learning, the learning rate remains constant (alpha) throughout the training process for all weight updates.

Stochastic gradient descent extensions like Adam are discussed by author in his writing\(^{18}\), as combining the advantages of two earlier extensions.

- By keeping a per-parameter learning rate, the Adaptive Gradient Algorithm (AdaGrad) can improve performance on problems with sparse gradients, such as computer vision issues and natural language processing.

- Root Mean Square Propagation (RMSProp) averages the weights’ recent gradient magnitudes and keeps the learning rates for each parameter (e.g. how rapidly it changes). Clearly, both online and non-stationary issues are no match for the algorithm! (e.g. noisy).

Adam takes use of both the benefits offered by AdaGrad and RMSProp. When adjusting the parameter learning rates, Adam makes use of the average of the second moments of the gradients in addition to the average of the first moments of the gradients, as opposed to utilizing the mean or the average of the first moments as is done in RMSProp (the uncentered variance). Adam is a well-known algorithm in the field of deep learning due to its capacity to give outstanding results in a very short amount of time. The Adam optimizer offers substantially higher performance than earlier models and outperforms them by a considerable margin when it comes to giving an optimum gradient descent. This is based on the strengths of the earlier models. The Adam technique is used in order to get the best possible results from the Predictive deep learning LSTM and BiLSTM models.

\(^{18}\) https://machinelearningmastery.com/Adam-optimization-algorithm-for-deep-learning/
CHAPTER 3. EXPERIMENTAL DESIGN AND METHODOLOGY

3.6 Model Evaluation

The LSTM and BiLSTM prediction models are evaluated using one set of evaluation measures, while the DistilBERT, BERT, RoBERTa and ALBERT sentiment analysis transformers are evaluated with another set of evaluation metrics. Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) are the three assessment criteria that are used in order to conduct an exhaustive analysis of the prediction outcomes produced by the LSTM and BiLSTM models (MAPE).

3.6.1 Classification Models Evaluation

F1-score

When it comes to classification models in machine learning, there are often two metrics that are used to evaluate the quality of the model: the F1-score and the Accuracy. The Accuracy metric, on the other hand, has the drawback of not taking into consideration the manner in which the data is dispersed. Consider that ninety percent of all feelings are negative. If a model just predicted whether the attitude expressed by a statement was positive or negative, then the model would accurately anticipate the result ninety percent of the time for negative words. Despite the fact that this number seems to be rather high, the model is unable to accurately predict positive sentiment in conjunction with semantic context. Take into account the manner in which the data is presented. For situations in which the data are very skewed, for instance, the F1-score will provide a more precise evaluation of the performance of the model. The F1-score is used as an assessment metric for transformer models because of the very unbalanced nature of the data included within the datasets.

The F1-score evaluates the accuracy and recall of sentiment classification models such as DistilBERT, BERT, and RoBERTa by calculating the harmonic mean of the two scores. The F1 score is determined using the following formula:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$  \hspace{1cm} (3.23)
3.6.2 Time Series Prediction Models Evaluation

This research problem is Time Series regression which is a statistical method for predicting a future response based on the response history, and normal accuracy measures do not apply for the time series as its dependent on period of time. The mean absolute percentage error (MAPE) is one of the most popular used error metrics in time series forecasting, which is correlated to accuracy, as the smaller the error the accurate (closer) the results are. So is why this research used MAPE as main evaluating matrix for hypothesis evaluation by comparing the stock prediction models LSTM and BiLSTM.

MAPE

Forecasting models are often evaluated using MAPE, a prominent and recommended measure for this purpose. By subtracting the projected and actual values, the MAPE calculates the average of absolute errors. Because absolute percentage mistakes are taken into account, the cancellation of positive and negative errors is prevented when utilizing MAPE as its principal benefit.

\[
MAPE = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{A_t - P_t}{A_t} \right| 
\]  

(3.24)

\(A_t\) is the actual value and \(P_t\) is the predicted value and \(t\) is the time in time series data.

RMSE

The Root Mean Square Error (RMSE) expresses the expected value of the square of the error caused by the predicted and true values, and its range is \([0, +\infty)\).

\[
RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^{n} (X_t - X'_t)^2} 
\]  

(3.25)

MAE

The mean absolute error (MAE) is the average of the absolute value of the deviation of all single observations, which avoids the problem of mutual cancellation of errors and accurately reflects the size of the actual forecast error, as shown below:

\[
MAE = \frac{1}{n} \sum_{t=1}^{n} |X_t - X'_t| 
\]  

(3.26)
3.6.3 Diebold-Mariano (DM) Test

The hypothesis must be statistically tested to ensure that the difference in MAPE between the BiLSTM and LSTM models is not coincidental and that the forecasts are significantly different. This study will employ a Diebold-Mariano (DM) Test for statistical analysis \cite{Diebold1995}. The prediction accuracy of the LSTM and BiLSTM models will be compared in this test. The DM test will determine the differential loss between the LSTM and BiLSTM models to accomplish this.

\[ d_t = g(e_{1t}) - g(e_{2t}) \] (3.27)

Where \( g(e_{it}) \) is the square (squared-error loss) or the absolute value (absolute error loss) of \( e_{it} \). If and only if the differential loss is zero, both models will have the same prediction accuracy. If the DM test result is positive, model1 has a high accuracy, and vice versa.

The Bi-LSTM model’s results will be compared to the LSTM model’s performance in terms of prediction accuracy using MAPE. A comparative analysis will be performed to determine whether the proposed model outperforms the existing techniques. The alternative and null hypothesis will be evaluated using statistical tests and the error indicator MAPE, i.e.

Assume, \( e_{1t} \) is LSTM average forecast errors and \( e_{2t} \) is BiLSTM average forecast errors,

- If \( d_t > 0 \) i.e., LSTM MAPE is higher than BiLSTM MAPE, reject null hypothesis, accept alternative hypothesis
- If \( d_t < 0 \) i.e., LSTM MAPE is lower than BiLSTM MAPE, accept null hypothesis, reject alternative hypothesis

\( d_t = 0 \); both models have same prediction accuracy

This study aims to provide a satisfactory explanation for the research question by comparing the results of the Bi-LSTM model with the LSTM model.
3.7 Overview

The CRISP-DM approach was presented in detail in this chapter, from data collection through assessment. Adj closing price forecasts for NASDAQ stocks of Google, Apple, Amazon, and Facebook are the research’s business purpose. Research begins with gathering information from a variety of sources, including stock price history, articles from trade publications like The Wall Street Journal, and the Stock Tweets dataset covering the period from January 1, 2020, to December 31, 2021. After that, the data is pre-processed to remove hashtags, usernames, punctuation, and URLs as needed from News articles and Tweets, and the stock price data is normalized. Other major factors that influence stock forecasts include technical indicators. The collection of technical indicators is then reduced in dimension using PCA. Nine alternative dataset combinations will be created from these data sources, and each of these dataset combinations will be used to train a different model. BERT, DistilBERT, RoBERTa, and ALBERT were utilized for sentiment analysis, as well as LSTM and BiLSTM for Adj close price prediction, which were trained and evaluated on nine distinct dataset combinations. A more robust way of extending windows is used to partition the dataset, and these training and test sets are used to fit and evaluate prediction models. BERT models are fine-tuned to provide more accurate sentiment classifications because the prepared textual input is not tagged. A collection of labeled financial headlines and stock Tweets is used to train and evaluate the algorithms. Models that have been previously trained are loaded into the News and Tweets dataset and used to determine whether a sentence is "Positive," "Negative," or "Neutral." Evaluation measures for sentiment analysis are F1-scores, with RoBERTa beating the other BERT models for both financial News and stock Tweets data sets, respectively. Stock prediction models are evaluated using MAPE and the Adj closing price of NASDAQ stocks is forecasted using this method.
Chapter 4

Experimental Implementation

This chapter describes the execution of various experiments carried out in this study. Begin by fine-tuning various transformer language models and determining the best-performing transformer language model for extracting sentiment polarities from collected stock News and stock Tweets. Fitting both LSTM and BiLSTM models with the nine different dataset combinations mentioned in section 3.4.4, and comparing LSTM and BiLSTM performance in predicting the next day adjusted close price of Google, Apple, Facebook, and Amazon.

4.1 Sentiment Extraction for News articles

In the section, two datasets are used as part of two different experiments, one the labelled dataset for fine tuning the transformers and the other, unlabeled data set acquired for 2020 and 2021, as stated in section 3.3.1, this News extracted, is the actual real time news that is not labelled or have sentiment polarity, this will be determined by the transformer models that we will be training in this section by fine tuning them with a dataset of 4845 financial h-indexes. The sole reason why headlines are employed for further fine-tuning is to familiarize the transformers with financial jargons. In the second experiment, the best-performing pre-trained transformer model from the first experiment is used to categorize each unseen text article as 'Positive, Negative, or Neutral' using the unlabeled News articles dataset.
4.1.1 Data Loading for Fine Tuning

The first step in the implementation phase is to load the datasets using Python. The drive is mounted in Google Colaboratory, and the dataset is loaded as a pandas dataframe from a CSV file. The dataset is first checked for null or missing values. Because the data did not contain any missing values, the data exploratory analysis is performed next. The percentage distribution of negative, neutral, and positive sentiments in the dataset is depicted using a bar chart. Figure 4.1 (a) depicts dataset bar charts.

4.1.2 Data Pre-processing

The text preprocessing steps described in Section 3.4.2 are used.

4.1.3 Review length distribution

The input texts for the pre-trained models must all be the same length. To accomplish this, a maximum length is determined based on the dataset used. The dist plot is used to understand the distribution of News articles in the dataset. Figure 4.1 (b) depicts the token length distribution of News articles. The mean length of articles is chosen as the maximum length of the input sequence to minimize the use of computational resources. ‘max seq length’ was set to 100 while training the model.

(a) Sentiment polarities
(b) Token length distribution

Figure 4.1: Financial Phrase Bank dataset Exploratory Analysis
4.1.4 Train-Validate-Test split

To split the dataset, the Python sklearn library is used. Using the train-test split function, each sample of the dataset is divided into Train-Validate-Test sets. 80% of the data is used for model training, and 10% is used for evaluation and testing of the built classifier.

4.1.5 Model training

a) BERT Fine-tuning with Financial Phrase Bank Dataset

With Hugging face as the foundation, more complicated transformer models can be created quickly and easily. News Articles classifier employ the Classification Model, which was developed specifically for binary and multi-class text classification. This dataset, known as the ”Financial Phrase Bank,” is used to fine-tune BERT using two independent models of BERT, bert-base-uncased and bert-base-cased. To encode Negative as 0, Neutral as 1, and Positive as 2, the LabelEncoder() method was used. A BERT tokenizer, conducts article tokenization, and the input text is transformed into tokens. For both of the model’s training functions, the maximum sequence length of 100 is provided as an input. 5 epochs of training, with a batch size of 32 for training and testing, are used for the models. The greater the demand on computing resources, the smaller the batch size should be. So, for training both the models 32 batch size is used since this is the next suggested batch size by Thakur (2020). According to Thakur (2020) study, learning rate has a substantially greater effect on model performance than batch size. Both models are trained with various learning rates, and the ultimate learning rate of 1e-4 is determined by comparing the performance. Two models are trained using categorical crossentropy loss function, since this is a classification issue. Adam optimizer is employed with these models, which is utilized to minimize the losses. The dropout layer is set at 0.2 in order to prevent models from overfitting. F1-score and accuracy are used to rank the models. In the last step, the validated model is evaluated on the test dataset to evaluate how well it performs on new data.
CHAPTER 4. EXPERIMENTAL IMPLEMENTATION

The following are the parameter settings for both the bert-base-uncased and bert-base-cased models:
Learning Rate = 1e-4
Number of Epochs = 5
Batch Size = 32
Maximum Sequence Length = 100
Table 5.1 displays the performance of BERT based on accuracy and F1-score metrics. DistilBERT, RoBERTa and ALBERT models follow same steps as this section of BERT models.

b) DistilBERT Fine-tuning with Financial Phrase Bank Dataset

The distilbert-base-uncased model is used to fine-tune DistilBERT with the 'Financial Phrase Bank dataset.' DistilBERT tokenizes the input text using the WordPiece tokenizer. The best results were obtained with a learning rate of 1e-4. It is important to keep all these parameters consistent in order to ensure that the findings can be compared to the BERT models. Table 5.1 shows DistilBERT’s performance in terms of accuracy and F1-score metrics.

c) RoBERT Fine-tuning with Financial Phrase Bank Dataset

The roberta-base model is used to fine-tune RoBERTa with the 'Financial Phrase Bank dataset.' RoBERTa was trained with a variety of learning rates and batch sizes. The best results were obtained with a learning rate of 1e-4 and a batch size of 32. RoBERTa employs a byte-level Byte-Pair-Encoding tokenizer for tokenization. The maximum sequence length is 100 and the number of epochs for training is 5. The loss function, optimizer, and other parameters remain unchanged from BERT models. Table 5.1 shows RoBERTa’s performance in terms of accuracy and F1-score metrics.

d) ALBERT Fine-tuning with Financial Phrase Bank Dataset

The albert-base-v2 model is used to fine-tune ALBERT with the 'Financial Phrase Bank dataset.' The input text is tokenized using the SentencePiece tokenizer. Keeping the other model parameters consistent with the BERT models ensures a fair comparison with other pre-trained models. Table 5.1 shows ALBERT’s performance in terms of accuracy and F1-score metrics.
4.2 Sentiment Extraction for Stock Tweets

Despite the fact that large amounts of Twitter datasets are freely available online from various sources, it was discovered that labelled datasets on the stock market were few and had a limited number of rows. IEEE DataPort dataset was used in this study\(^\text{19}\). The IEEE DataPort stock market Tweets dataset consist of Tweets between April 9 and July 16, 2020, using the S&P 500 tag (# SPX500), the references to the top 25 companies in the S&P 500 index, and the Bloomberg tag (# stocks). 1,300 out of the 943,672 Tweets were manually annotated as positive, neutral, or negative classes. Firstly, downloaded this dataset from IEEE, which has 'Tweets\_labelled\_09042020\_16072020.csv' file, that consists on 5000 Tweets out of which 1300 are labelled into ‘Positive’, ‘Negative’, or ‘Neutral’. For this experiment to fine tune the transformer models, labelled data is used. So VADER is used to sentiment label the remaining unlabeled 3699 Tweets from 'Tweets\_labelled\_09042020\_16072020.csv' file.

Emotional text sentiment may be analyzed using VADER (Valence Aware Dictionary and Entiment Reasoner). To determine a word’s emotional connotation, a lexical comparison is used. In the NLTK package, it can be used directly with unlabeled text data.

This section describes two main experiments using transformer language models fitted with two stock Tweets datasets, one labelled dataset for fine tuning the transformers and the other an unlabeled collected stock Tweets dataset. In the first experiment, these models are trained and tested using the labeled Stock Market Tweets Data ([Taborda et al., 2021](#taborda2021)). The best-performing pre-trained transformer model from the first experiment is fitted with the unlabeled collected stock Tweets dataset in the second experiment to classify each text as ‘Positive,’ ‘Negative,’ or ‘Neutral.’

\(^{19}\)https://ieee-dataport.org/open-access/stock-market-Tweets-data
4.2.1 VADER sentiment labeling of unlabelled Stock Market Tweets Data

This section describes how VADER is used to sentiment label the remaining unlabelled 3699 Tweets from 'Tweets_labelled_09042020_16072020.csv' file, so that this data is used for fine tuning transformers in further sections.

a) Data Cleaning

After loading the dataset, it is cleaned before being analyzed with VADAR. Many characters and words in this Tweets dataset are unnecessary for sentiment analysis. As a result, the following operations are carried out in the dataset:
- Removed mentions (@username)
- Removed RT (Retweet icon)
- Removed hyperlinks and URLs

Used 're', built-in package of Python for data cleaning process. Once data is cleaned, the sentiment analysis of this data is performed.

b) Sentiment Analysis using VADER

The SentimentIntensityAnalyzer() function of VADER receives a string and returns a dictionary of scores in four categories: negative, neutral, positive, and compound (computed by normalising the values above) (computed by normalising the scores above)

The labels of the text data is determined, by logic, if compound score \( \geq 0.05 \), it marked it positive, if compound score \( \geq -0.05 \), rest all are marked neutral.

Though sentiment analysis is successfully performed on unlabeled Stock Market Tweets Data from \cite{Taborda et al., 2021}, it is important to note that achieving the highest accuracy is always difficult due to a variety of possible factors such as:
- Positive and negative sentiments coexist in the same text data.
- Witty remark (using positive words in negative way).
- Grammatical errors or misspellings may cause the analysis to miss important words.

Now that the fully labelled 5000 stock Tweets data set from \cite{Taborda et al., 2021} is available, the data obtained is used to fine-tune transformers.
CHAPTER 4. EXPERIMENTAL IMPLEMENTATION

4.2.2 Data Loading for Fine Tuning

The first step in the implementation phase is to load the datasets using Python. The drive is mounted in Google Colaboratory, and the dataset is loaded as a pandas dataframe from a CSV file. The dataset is first checked for null or missing values. Because the data did not contain any missing values, the data exploratory analysis is performed next. The percentage distribution of negative, neutral, and positive sentiments in the dataset is depicted using a bar chart. Figure 4.2 (a) depicts dataset bar charts.

4.2.3 Data Pre-processing

The text preprocessing steps described in Section 3.4.2 are followed.

4.2.4 Review length distribution

The input texts for the pre-trained models must all be the same length. To accomplish this, a maximum length is determined based on the dataset used. The dist plot is used to understand the distribution of Stock Tweets in the dataset. The token length distribution of Stock Tweets is depicted in Figure 4.2 (b). The mean length of Tweets is chosen as the maximum length of the input sequence to minimize the use of computational resources. 'max seq length' is set to 90 while training the model.

![Tweets Sentiment Distribution](image1)

(a) Sentiment polarities

![Token length distribution](image2)

(b) Token length distribution

Figure 4.2: Stock Market Tweets, IEEE Dataport dataset Exploratory Analysis
4.2.5 Train-Validate-Test split

The Python module sklearn is used in order to partition the dataset. Each dataset sample is then divided into Train-Validate-Test sets by using the train-test split function. 80% of the data is put to use in the training of the model, while the remaining 10% is put to use in the testing and validation of the classifier.

4.2.6 Model training

a) BERT Fine-tuning with Stock Market Tweets Dataset

Except for the training parameters, the 'Stock Market Tweets Dataset' is subjected to the same experiments as the 'Financial Phrase Bank dataset' in section 4.1. To fine-tune BERT with the 'Stock Market Tweets Dataset,' two different BERT models, bert-base-uncased and bert-base-cased, are implemented separately. The maximum sequence length of 90 is fed into both models’ training functions. The models are trained for 5 epochs with a batch size of 32 for training and evaluation. Both models are trained with different learning rates, and the rate 1e-4 is determined by comparing their performance.

The categorical crossentropy loss function, Adam optimizer, and an epsilon value of 1e-8 are used to train both bert-base-uncased and bert-base-cased models. To prevent the models from overfitting, the dropout layer value is set to 0.2. F1-score and accuracy are used to evaluate the models. Finally, the validated model is tested using the test dataset to determine the model’s performance on unseen collected data. The parameter settings for the bert-base-uncased and bert-base-cased models are as follows.:

- Learning Rate = 1e-4
- Number of Epochs = 5
- Batch Size = 32
- Maximum Sequence Length = 90

Table 5.2 displays the performance of BERT based on accuracy and F1-score metrics.
c) DistilBERT Fine-tuning with Stock Market Tweets Dataset

When working with the 'Stock Market Tweets Dataset,' the distilbert-base-uncased model is used in order to fine-tune DistilBERT. WordPiece is the tokenizer that DistilBERT employs in order to process the input text. The findings were the most successful when a learning rate of 1e-4 was used. For the purpose of ensuring that the results can be compared to the BERT models, it is essential to maintain consistency across all of these factors. The results of DistilBERT’s performance, including its accuracy and F1-score metrics, are shown in Table 5.2.

b) RoBERT Fine-tuning with Stock Market Tweets Dataset

The roberta-base model is applied to the 'Stock Market Tweets Dataset' in order to fine-tune the RoBERTa algorithm. RoBERTa received training using a wide range of different learning rates and group sizes. The optimal learning rate was 1e-4, and the optimal batch size was 32. This combination produced the best results. To complete the tokenization process, RoBERTa makes use of a byte-level Byte-Pair-Encoding tokenizer. The maximum number of sequences that may be in a training set is 100, and the number of epochs is 5. The optimizer, loss function, and other parameters are all kept in the same state as they were in BERT models. This is done in this manner so that several pre-trained models may be compared with one another in an objective manner. The results of RoBERTa’s performance, including its accuracy and F1-score metrics, are shown in Table 5.2.

d) ALBERT Fine-tuning with Stock Market Tweets Dataset

The albert-base-v2 model is applied to the 'Stock Market Tweets Dataset' in order to fine-tune ALBERT’s performance. The SentencePiece tokenizer is used to tokenize the text that was provided as input. A fair comparison with other pre-trained models may be ensured by maintaining consistency across all of the other model parameters with regard to the BERT models. The results of ALBERT’s performance, measured in terms of accuracy and F1-score metrics, are shown in Table 5.2.
4.3 Experiment 1 - Predicting Adj close price using Historic Stock Price

In this section, two experiments are carried out, one using the LSTM model and the other using the BiLSTM model, to predict Google’s next day adjusted close stock price using Historic Stock Price alone. Google data is utilized for all predictive analysis experiments due to time restrictions. Experiments will be complicated if they include Apple, Facebook, or Amazon. The best-performing models from these experiments are subsequently used to forecast the stocks of Apple, Facebook, and Amazon.

i. Packages Imports and Data Loading

As mentioned in section 3.2, all required packages are imported. Following that, the cleaned data from section 3.4.2 is loaded, and data Normalization is performed to scale the data between 0 and 1, as explained in the Quantitative Data Transformation section in 3.4.2.

ii. Incorporating Timesteps Into Data

To feed an LSTM or a BiLSTM, the data must be transformed into a three-dimensional (3-D) array, as [batch_size, time_steps, Features].

* **Batch Size** the weights of the Neural Network are updated based on how many samples of input it has seen.

* **Time Steps** Give specifics on how far back in time the network should go.

* **Features** how many qualities are associated with each time step.

In this experiment, a batch size of 32 is used, which is the batch size recommended by (Thakur, 2020). Three different time steps of 1 day, 3 days, and 10 days are chosen. 1 day time steps indicates that the network has to look back at 1 day of data to anticipate the price on the second day; 3 days time steps imply that the network must rely on data from the previous three days in order to accurately forecast the price for the next fourth day. To anticipate the price on the 11th day, the network has to look back at 10 days of data from the previous 10 days. As discussed in section 3.4.1, four features are taken into account.
iii. Creating and fitting LSTM and BiLSTM Models

First imports are made from Keras for implementing the LSTM and BiLSTM models, 'Sequential' import for initializing the neural network, LSTM for adding the LSTM layer, and Bidirectional in order to incorporate the BiLSTM, Dropout layers are used to prevent overfitting, and Dense layers are used to add a densely connected neural network layer.

Sequential LSTM and BiLSTM models are imported from Keras to generate the six LSTM models and six BiLSTM models shown in Table 3.3, with Mean squared error as the loss function and the Adam technique as the optimizer. Because the models are associated with time-series data, the metric used is mean absolute error.

To reduce overfitting and improve generalization capacity, the number of epochs is set to 100, and loss values are monitored using the Early stopping call back function. This causes training to halt when an increase in loss values is detected. The patience=30 and batch size=32 parameters are set for all six model’s earlystopping callback functions.

iv. Model prediction

'Historic Stock Price' data is fitted separately with six different LSTM/BiLSTM models over three time steps: 1 day, 3 days, and 10 days. With 'Historic Stock Price' data, 36 different models are trained in total. Once fitted, Google’s next day adjusted close price predictions for each of these models are made on the test data set for three different time steps, and validated using expanding window cross validation, as explained in section 3.5.3. The error statistics mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE) are calculated using the actual and predicted values.

v. Result visualization

Following all of these steps, the predicted adjusted close price and the actual adjusted close price for each of the models with three different time steps are plotted. Section 5.4 describes the evaluation and results of these models.
4.4 Experiment 2 - Predicting Adj close price using Historic Stock Price and News articles

Google’s next-day adjusted close stock price is predicted in this section using the LSTM model and the BiLSTM model, using Historic Stock Price and News articles dataset.

Table 3.3 presents the various topologies of the LSTM and BiLSTM models that were utilized in this experiment. These models were fitted with the Historical Stock Price and News Articles datasets that were obtained in section 3.4.4. The models were given three distinct time steps of one day, three days, and ten days separately.

The experimental method described in Section 4.3 is carried out, however the dataset used to make a prediction about the next day’s adjusted closing stock price for Google is Historic Stock Price with News Articles rather than the one described in Section 4.3. The findings of the model assessments are shown in Table 5.5 below.

4.5 Experiment 3 - Predicting Adj close price using Historic Stock Price and Stock Tweets

Google’s next-day adjusted close stock price is predicted in this section using the LSTM model and the BiLSTM model, which are both based on a dataset of Historic Stock Price and Stock Tweets dataset.

Table 3.3 presents the various topologies of the LSTM and BiLSTM models that were utilized in this investigation. These models were fitted with the Historical Stock Price and Stock Tweets datasets that were obtained in section 3.4.4. The models were given three distinct time steps of one day, three days, and ten days respectively.

The experimental method described in Section 4.3 is carried out, however the dataset used to make a prediction for the next day’s adjusted closing stock price for Google is Historic Stock Price with Stock Tweets rather than the one described in Section 4.3. The findings of the model assessments are shown in Table 5.6 below.
4.6 Experiment 4 - Predicting Adj close price using Historic Stock Price and Technical Indicators

Google’s next-day adjusted close stock price is predicted in this section using the LSTM model and the BiLSTM model, which are both based on a dataset of Historic Stock Price and Technical Indicators dataset.

Table 3.3 shows the various topologies of the LSTM and BiLSTM models used in this study, which are fitted with the Historic Stock Price and Technical Indicators obtained in section 3.4.4 with three distinct time steps of 1 day, 3 days and 10 days.

Section 4.3’s experimental technique is followed, but the dataset used to forecast Google’s following day adjusted closing stock price is, Historic Stock Price and Technical Indicators instead of Section 4.3’s. Table 5.7 displays the results of model evaluations.

4.7 Experiment 5 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied

Google’s next-day adjusted close stock price is predicted in this section using the LSTM model and the BiLSTM model, which are both based on a dataset of Historic Stock Price and Technical Indicators with PCA Applied dataset.

The PCA dimentionality reduction of the stock price and technical indicators dataset is described in Section 3.5.1. As shown in Table 3.3, the dataset from section 3.4.4, Merged Historic Stock Price and Technical Indicators with PCA applied, is fitted using six distinct LSTM and BiLSTM models for three different time steps: one day, three days, and ten days. Section 4.3’s experimental technique is followed, but the dataset used to forecast Google’s following day adjusted closing stock price is, Historic Stock Price and Technical Indicators with PCA Applied instead of Section 4.3’s. Table 5.8 displays the results of model evaluations.
CHAPTER 4. EXPERIMENTAL IMPLEMENTATION

4.8 Experiment 6 - Predicting Adj close price using Historic Stock Price, New Articles and Stock Tweets

Google’s next-day adjusted close stock price is predicted in this section using the LSTM model and the BiLSTM model, which are both based on a dataset of Historic Stock Price, New Articles and Stock Tweets dataset.

Table 3.3 displays the topologies of the LSTM and BiLSTM models used in this research, which are fitted using the Historic Stock Price, New Articles, and Stock Tweets from section 3.4.4 with 1 day, 3 days, and 10 days time steps. Following Section 4.3’s experimental approach, Historic Stock Price, New Articles, and Stock Tweets are utilized to anticipate Google’s next day adjusted closing stock price. Table 5.9 shows assessment findings.

4.9 Experiment 7 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, plus News Articles

This section predicts Google’s next-day adjusted closing stock price using the LSTM model and the BiLSTM model, both based on a dataset of Historical Stock Price and Technical Indicators with PCA Applied, plus News Articles.

Section 3.5.1 describes the PCA dimension reduction of stock prices and technical indicators. The dataset from section 3.4.4, Merged Historic Stock Price and Technical Indicators with PCA Applied, plus News articles, is fitted using six LSTM and BiLSTM models for three time steps, 1 day, 3 days, 10 ten days. Repeating Section 4.3’s for dataset, Historic Stock Price and Technical Indicators with PCA Applied, plus News Articles are utilized to anticipate Google’s next day adjusted closing stock price. Table 5.10 shows assessment findings.
4.10 Experiment 8 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, plus Stock Tweets

This section predicts Google’s next-day adjusted closing stock price using the LSTM model and the BiLSTM model, both based on a dataset of Historic Stock Price and Technical Indicators with PCA Applied, plus Stock Tweets.

Section 3.5.1 describes the PCA dimension reduction of stock prices and technical indicators. The dataset from section 3.4.4, Merged Historic Stock Price and Technical Indicators with PCA Applied, including Stock Tweets, is fitted using six LSTM and BiLSTM models for three time steps: one day, three days, and ten days. Repeat Section 4.3’s steps for dataset, Historic Stock Price and Technical Indicators with PCA Applied, plus Stock Tweets are used to anticipate Google’s next day adjusted closing stock price. Table 5.11 shows assessment findings.

4.11 Experiment 9 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets

This section predicts Google’s next-day adjusted closing stock price using the LSTM model and the BiLSTM model, both based on a dataset of Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets.

Section 3.5.1 describes the PCA dimension reduction of stock prices and technical indicators. The dataset from section 3.4.4, Merged Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets, is fitted using six LSTM and BiLSTM models for three time steps: one day, three days, and ten days.
Repeat Section 4.3’s steps for dataset, Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets are utilized to anticipate Google’s next day adjusted closing stock price. Table 5.12 shows assessment findings.

### 4.12 Overview

Experiments were detailed in this section of the study. Data used in this study includes past NASDAQ stock prices, 32 technical indicators from Python’s ta-lib library, sentiment polarity in News articles, and sentiment polarity in Twitter Tweets, as previously indicated in section 3.3.1. To extract sentiments from unlabelled News articles, several BERT models are first trained and evaluated on a labelled Financial Phrase Bank Dataset. An unlabelled dataset of News articles, which is a collection from financial News headlines collected over a two-year period, is used to test BERT models to classify each News article as either "positive," "negative," or "neutral." Next, train and evaluate different types of BERT models on a dataset of labelled financial stock Tweets to extract sentiment polarity. Used the best model to classify each tweet in the Tweets dataset as either positive, negative, or neutral. This is done by saving and loading the model with the unseen datasets. After sentimental analysis of News articles and Tweets is done, these sentiment polarities on any given day may be computed by averaging the sentiment polarities across all the News articles and Tweets. In order to identify the best model, 9 different combinations of datasets formed in section 3.4.4 are fitted with 6 different architectures for each of LSTM and BiLSTM models for 1 day, 3 days and 10 days time steps. These experiments used Google’s data sources to find the best performing model mainly the model that have performed well with multi source dataset in experiment 9, as the best performed model in experiment 9, is used to predict the Apple, Facebook and Amazon Adjusted close price discussed in the next chapter 5. Each model is cross-validated 5 times, with the split size remaining constant throughout all tests, using the expanding window approach. All nine LSTM experiments use the same BiLSTM model architecture for a fair comparisons. The collected data is analyzed and summarized in Chapter 5.
Chapter 5

Evaluation and Results

The evaluations of the data that were acquired from the experimental procedures that were detailed in Chapter 4 are presented in this chapter. The fundamental objective of this study is to estimate the Adj closing price of Google, Apple, Facebook, and Amazon NASDAQ stocks by using the BiLSTM model and multi-source data, as mentioned in Chapter 1. The findings of the BiLSTM model are compared to those of the LSTM model which served as the baseline model, and both LSTM and BiLSTM models are trained and evaluated using the same multi-source data. In this chapter, the findings generated by the LSTM and BiLSTM models are analyzed and evaluated in order to decide which model performs the best. Comparative analysis and Diebold Marino tests are used to compare these models. And finally analyzing the hypothesis and drawing a conclusion on the aim of this research is accomplished.

5.1 Results and Hypothesis Evaluation

According to the efficient-market theory, it is very difficult, if not impossible, to accurately forecast the adjusted closing price based on the previous stock price or data. The purpose of this research is to put this idea to the test by making an effort to forecast the adjusted closing price using historical stock price data, market conditions, and investor sentiment. For the purpose of validating both the alternative and the null hypothesis, Diebold-Mariano (DM) statistical testing are used.
CHAPTER 5. EVALUATION AND RESULTS

Alternative hypothesis ($H_a$): If a model is trained with the Bi-LSTM technique using data from multiple sources and optimized with the Adam algorithm, it will have a lower MAPE than a model trained with LSTM technique using the same set of data in the prediction of adjusted close price of NASDAQ stock prices.

Null hypothesis ($H_0$): If a model is trained with the Bi-LSTM technique using data from multiple sources and optimized with the Adam algorithm, it will have a higher MAPE than a model trained with LSTM technique using the same set of data in the prediction of adjusted close price of NASDAQ stock prices.

Accurate sentiment classification of unseen News articles and Tweets is critical for predicting adjusted close price. As demonstrated in Table 5.1, RoBERTa outperformed BERT, DistilBERT, and ALBERT in sentiment classification of 'Financial Phrase Bank' dataset. According to (Araci, 2019), FinBERT beats state-of-the-art machine learning algorithms, reaching an accuracy of 86% and an F1-score of .84. In this study, RoBERTa model outperformed FibBERT, with an accuracy of 88% and an F1-score of .88 for the 'Financial Phrase Bank' dataset. And as demonstrated in Table 5.2, RoBERTa outperformed BERT, DistilBERT, and ALBERT in sentiment classification of 'Stock Market Tweets' Dataset, reaching 71% and .71 F1-score.

RoBERTa appears to perform better on financial data than other transformer models from BERT family, hence this study employed RoBERTa to sentimentally classify News articles and Tweets.

In predicting adjusted close price of Google for the next day, using multi source dataset, 'Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets' with 1 day time step, the single layer LSTM model achieved MAPE of 12.55, whereas the single layer BiLSTM model achieved MAPE of 12.49. The findings indicated that BiLSTM performed slightly better than LSTM in forecasting the adjusted close price of Google’s NASDAQ stocks.
CHAPTER 5. EVALUATION AND RESULTS

(a) Single Layer BiLSTM  
(b) Stacked 2 Layer BiLSTM

(c) Stacked 4 Layer BiLSTM  
(d) Stacked 6 Layer BiLSTM

Figure 5.1: Model Train vs Validation Loss of Single vs Stacked BiLSTM models

Stacked LSTM and BiLSTM models were also investigated for prediction of adjusted close price, resulting in improved accuracy as number of layers increased until as threshold of 4 layers, after which the model’s performance dropped with 6 layers. In other words, addition of layers helped improve the models, but only up to a certain point, and further addition of layers actually harmed the model’s performance. As shown in Figure 5.1, the loss of models continues to improve (i.e. be minimized) as layers are increased, but for 6 layered model there is no consistency in the loss reduction, as evidenced by the frequent large spikes in loss in Figure 5.1 (d).

Stacked LSTM and BiLSTM models are investigate with different time steps, 1 day, 3 day and 10 days. And results showed that for both stacked LSTM and stacked BiLSTM 1 day time step have given better MAPE results compared to 3 day and 10 days time steps for all the nine combinations of datasets.
CHAPTER 5. EVALUATION AND RESULTS

(a) 10 days Time Steps
(b) 3 days Time Steps
(c) 1 day Time Steps

Figure 5.2: MAPE values for Adj close price prediction of Google using 4 Layer LSTM and BiLSTM models with 150 units and 0.2 dropout

4-layer Adam optimized BiLSTM network with 150 units and 1 day time step resulted in the least Mean Absolute Percentage Error ranging from 1.86 to 4.36 for different data source combinations than the 4-layer Adam optimized LSTM model with 150 units and 1 day time step, which achieved 1.98 to 5.18 MAPE. BiLSTM outperformed LSTM with different architectures, time steps, and data sets, as illustrated in Figure 5.2, tables 5.3 to 5.11 comparing MAPE, and Table 5.12’s DM test results. Although not a fair comparison with [Nagesh, 2021; R. Zhang et al., 2022] outcomes, as using different datasets and targets, but BiLSTM models from this study produced superior MAPE values than combined Facebook Prophet and Attn-LSTM from [Nagesh, 2021] and ARIMA and LSTM from [R. Zhang et al., 2022]. BiLSTM achieves more accurate time series predictions than Uni-directional LSTM [Xu et al., 2020; Q. Chen et al., 2020; Lu et al., 2020].
CHAPTER 5. EVALUATION AND RESULTS

5.2 Evaluation and Results of Sentiment Extraction for News articles

5.2.1 Transformer model evaluation

Metrics such as F1-score and accuracy are used to gauge performance in this section. Because it treats all classes equally, the macro average is an excellent choice for working with an unbalanced dataset, as stated in (Leung, 2022). So, macro F1-score is used in this experiment to classify data as negative, positive, or neutral.

Evaluation Metrics

<table>
<thead>
<tr>
<th>Language Model</th>
<th>All Data</th>
<th>Data with 100% agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Loss</td>
<td>Accuracy</td>
</tr>
<tr>
<td>bert-base-uncased</td>
<td>0.29</td>
<td>0.87</td>
</tr>
<tr>
<td>bert-base-cased</td>
<td>0.38</td>
<td>0.86</td>
</tr>
<tr>
<td>distilbert-base-uncased</td>
<td>0.37</td>
<td>0.85</td>
</tr>
<tr>
<td>roberta-base</td>
<td>0.30</td>
<td><strong>0.88</strong></td>
</tr>
<tr>
<td>albert-base-v2</td>
<td>0.37</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Table 5.1: Experimental Results on the Financial PhraseBank dataset

5.2.2 Labelling new offline and real time News articles dataset

From the results it is observed that Roberta-case model have performed best. So, Roberta-case model is fitted with the new unseen News articles dataset collected from Jan 2020 to Dec 2021 (section 3.3.1), for assigning sentiment polarity for News articles.
5.3 Evaluation and Results of Sentiment Extraction for Stock Tweets

5.3.1 Transformer model evaluation

Metrics such as F1-score and accuracy are used to gauge performance in this section. Because it treats all classes equally, the macro average is an excellent choice for working with an unbalanced dataset, as stated in (Leung, 2022). So, macro F1-score is used in this experiment to classify data as negative, positive, or neutral.

**Evaluation Metrics**

<table>
<thead>
<tr>
<th>Language Model</th>
<th>Labelled Stock Tweets Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Loss</td>
</tr>
<tr>
<td>bert-base-uncased</td>
<td>1.10</td>
</tr>
<tr>
<td>bert-base-cased</td>
<td>1.01</td>
</tr>
<tr>
<td>distilbert-base-uncased</td>
<td>1.06</td>
</tr>
<tr>
<td>roberta-base</td>
<td>0.75</td>
</tr>
<tr>
<td>albert-base-v2</td>
<td>1.15</td>
</tr>
</tbody>
</table>

Table 5.2: Experimental Results on the Stock Market Tweets Dataset

5.3.2 Labelling new offline and real time News articles dataset

From the results it is observed that Roberta-case model have performed best. So, Roberta-case model is fitted with the new unseen Twitter Tweets dataset collected from Jan 2020 to Dec 2021 (section 3.3.1), for assigning sentiment polarity for Twitter Tweets.
5.4 Evaluation and Results of Experiment 1 - Predicting Adj close price using Historic Stock Price

The evaluation metric selected for this research, is MAPE (explained in section 3.6.2). The final MAPE of LSTM and BiLSTM models are obtained by taking the mean of all the MAPE values from cross-validation.

Single layer LSTM with 32 units/neurons, and with 1 day time-step (look-back) achieved MAPE of 4.31 for prediction of Google adjusted close price for next day, whereas the BiLSTM model with 32 units/neurons, and with 1 day time-step (look-back) has shown slight improvement in MAPE of 4.29. For training these models, the resources used by LSTM when measured in terms of 'Train time' is 6.18 seconds, whereas BiLSTM took 7.20 seconds training time. MAPE have improved for both LSTM and BiLSTM by adding an extra layer for each models, and doubling the number of units, i.e., 64 neurons, resulting in MAPE of 3.67 for LSTM and 3.57 for BiLSTM. At the same time adding extra layer has increased in training times for these models, LSTM took 10.17 seconds and BiLSTM took 14.38 seconds. Further these LSTM and BiLSTM models are fitted, by adding a small dropout of 0.2. That is, with number of layer as 2 and number of units as 64 and a dropout of 0.2. Results show, MAPE have improved compared two first two models, with 2.56 for LSTM and 2.53 for BiLSTM. It is observed, still it isn’t big difference in MAPE between both models, but the training time taken for BiLSTM is nearly twice the time taken by LSTM, with 11.63 seconds for LSTM and 22.23 for BiLSTM. A good few experiments are conducted, with different combinations of number of layers and number of units with, dropout of 0.2 and time step of 1 day. Observed MAPE values show, with increase in number of layers, MAPE keeps dropping until a threshold of 4 layers, but with 6 layers architecture, MAPE started increasing for both LSTM and BiLSTM models. Best preformed architecture for both LSTM and BiLSTM is 4 layered, with 150 units and 0.2 dropout with 1 day time step, where LSTM achieved a MAPE of 2.00 and BiLSTM achieved better
MAPE of 1.92. And the training time for these models are 18.49 seconds for LSTM and, 39.04 for BiLSTM. Results shown in Table 5.3, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all architectures, BiLSTM models performed better than LSTM models with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

<table>
<thead>
<tr>
<th>Look-back value</th>
<th>Dropout after layer</th>
<th>Train RMSE</th>
<th>Test MAPE</th>
<th>Train time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.00</td>
<td>2.00</td>
<td>4</td>
<td>8.54</td>
<td>38.17</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>6</td>
<td>4.95</td>
<td>51.26</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>4</td>
<td>3.66</td>
<td>48.57</td>
</tr>
</tbody>
</table>

Table 5.3: Learning results using Historic Stock Price

Exactly same combination of experiments are performed for LSTM and BiLSTM models as mentioned above, with different number of layers and number of units, and a 0.2 dropout for time steps 3 days and 10 days as well, to predict Google’s adjusted close price for next day, and observed that time-steps of 1 day, achieved better MAPE compared to time steps of 3 days and 10 days for both LSTM and BiLSTM models, as shown in Table 5.3. For single layer LSTM with 1 day time step MAPE resulted in 4.31, where as for the same model with 3 days and 10 days, MAPE resulted in 4.94, and 5.48 respectively. And, for single layer BiLSTM resulted in MAPE of 4.29, 4.95 and 5.46 for 1 day, 3 days and 10 days respectively. Table 5.3, shows the results for all architecture with different number of layers, different number of units and 0.2 dropout for both LSTM and BiLSTM models with 1 day, 3 days and 10 days time steps. 1 day time step, achieved better MAPE value. That means, as the time steps increased, MAPE increased. Both LSTM and BiLSTM models predict better with least time steps.
5.5 Evaluation and Results of Experiment 2 - Predicting Adj close price using Historic Stock Price and News articles

The evaluation metric selected for this research, is MAPE (explained in section 3.6.2). The final MAPE of LSTM and BiLSTM models are obtained by taking the mean of all the MAPE values from cross-validation.

Single layer LSTM with 32 units/neurons, and with 1 day time-step (look-back) achieved MAPE of 11.36 for prediction of Google adjusted close price for next day, whereas the BiLSTM model with 32 units/neurons, and with 1 day time-step (look-back) has shown improvement with MAPE of 10.92. For training these models, the resources used by LSTM when measured in terms of 'Train time' is 4.29 seconds, whereas BiLSTM took 4.32 seconds training time. MAPE have improved for both LSTM and BiLSTM by adding an extra layer for each models, and doubling the number of units, ie., 64 neurons, resulting in MAPE of 9.04 for LSTM and 8.36 for BiLSTM. At the same time adding extra layer has increased in training times for these models, LSTM took 6.26 seconds and BiLSTM took 11.47 seconds. Further these LSTM and BiLSTM models are fitted, by adding a small dropout of 0.2. That is, with number of layer as 2 and number of units as 64 and a dropout of 0.2. Results show, MAPE have improved compared two first two models, with 5.07 for LSTM and 4.89 for BiLSTM. It is observed, that the training time taken for BiLSTM is nearly twice the time taken by LSTM, with 6.13 seconds for LSTM and 12.72 for BiLSTM. A good few experiments are conducted, with different combinations of different number of layers and different number of units with, dropout of 0.2 and time step of 1 day. Observed MAPE values show, with increase in number of layers, MAPE keeps dropping until a threshold of 4 layers, but with 6 layers architecture, MAPE started increasing for both LSTM and BiLSTM models. Best preformed architecture for both LSTM and BiLSTM is 4 layered, with 150 units and 0.2 dropout with 1 day time step, where LSTM achieved a MAPE of 2.08 and BiLSTM achieved better MAPE of 2.04. And the training time for
these models are 12.31 seconds for LSTM and, 37.64 for BiLSTM. Results shown in Table 5.4, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all architectures, BiLSTM models performed better than LSTM models with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

<table>
<thead>
<tr>
<th>Number of layers</th>
<th>Number of neurons</th>
<th>Dropout after layer</th>
<th>Look-back value</th>
<th>Train MAE</th>
<th>Test MAE</th>
<th>Train RMSE</th>
<th>Test RMSE</th>
<th>Test MAPE</th>
<th>Train time</th>
<th>Test time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64</td>
<td>0.2</td>
<td>1</td>
<td>18.27</td>
<td>21.36</td>
<td>12.22</td>
<td>12.51</td>
<td>13.28</td>
<td>10.68</td>
<td>37.64</td>
</tr>
<tr>
<td>3</td>
<td>64</td>
<td>0.2</td>
<td>3</td>
<td>15.12</td>
<td>17.41</td>
<td>14.85</td>
<td>15.12</td>
<td>10.92</td>
<td>12.93</td>
<td>37.64</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
<td>0.2</td>
<td>10</td>
<td>10.68</td>
<td>13.48</td>
<td>9.67</td>
<td>10.39</td>
<td>11.53</td>
<td>11.91</td>
<td>37.64</td>
</tr>
</tbody>
</table>

Table 5.4: Learning results using Historic Stock Price and News articles

Exactly same combination of experiments are performed for LSTM and BiLSTM models as mentioned above, with different number of layers and different number of units, and a 0.2 dropout for time steps 3 days and 10 days as well, to predict Google’s adjusted close price for next day, and observed that time-steps of 1 day, achieved better MAPE compared to time steps of 3 days and 10 days for both LSTM and BiLSTM models, as shown in Table 5.4. For single layer LSTM with 1 day time step MAPE resulted in 11.36, where as for the same model with 3 days and 10 days, MAPE resulted in 12.93, and 15.6 respectively. And, for single layer BiLSTM resulted in MAPE of 10.92, 11.53 and 13.28 for 1 day, 3 days and 10 days respectively. Table 5.4, shows the results for all architecture with different number of layers, different number of units and 0.2 dropout for both LSTM and BiLSTM models. 1 day time step, achieved better MAPE value. That means, as the time steps increased, MAPE increased. Both LSTM and BiLSTM models predict better with least time steps.
5.6 Evaluation and Results of Experiment 3 - Predicting Adj close price using Historic Stock Price and Stock Tweets

The evaluation metric selected is MAPE, explained in section 3.6.2. The final MAPE of LSTM models are obtained by taking the mean of all the MAPE values from cross-validation.

Single layer LSTM with 32 units/neurons with 1 day time-step (look-back) achieved MAPE of 10.06 for prediction of Google adjusted close price for next day, whereas the BiLSTM model with 32 units/neurons with 1 day time-step (look-back) has shown improvement in MAPE of 8.06, saying that resources used by LSTM when measured in terms of 'Train time' is 9.60 seconds, where as BiLSTM took 12.7 seconds training time. MAPE have improved for both LSTM and BiLSTM by adding an extra layer for each models, and doubling the number of units, ie., 64 neurons, resulting in MAPE of 6.31 for LSTM and 5.61 for BiLSTM. At the same time adding extra layer has increased in training times for these models. 9.74 seconds and 20.86 seconds for LSTM and BiLSTM respectively. It is observed the training time taken for BiLSTM is more than twice the time taken by LSTM. These LSTM and BiLSTM models are further experimented, by adding a small dropout of 0.2. That is with number of layer as 2 and number of units as 64 and a dropout of 0.2. Results show, MAPE have improved compared two first two models, with 4.93 for LSTM and 4.89 for BiLSTM. A good few experiments are conducted, with different combinations of different number of layers and different number of units with, dropout of 0.2 and time step of 1 day. And results shown in Table 5.3, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models. Best preformed architecture for both LSTM and BiLSTM is 4 layered with 150 units and 0.2 dropout with 1 day time step, where
LSTM achieved a MAPE of 4.22 and BiLSTM achieved better MAPE of 3.8. And results shown in Table 5.5, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models, BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

Table 5.5: Learning results using Historic Stock Price and Stock Tweets

<table>
<thead>
<tr>
<th>Number of layers</th>
<th>Number of neurons</th>
<th>Dropout after layer</th>
<th>Train MAE</th>
<th>Train RMSE</th>
<th>Train MAPE</th>
<th>Test MAE</th>
<th>Test RMSE</th>
<th>Test MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64</td>
<td>0.2</td>
<td>21.37</td>
<td>4.78</td>
<td>7.55</td>
<td>15.92</td>
<td>6.58</td>
<td>7.89</td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>0.2</td>
<td>15.92</td>
<td>6.58</td>
<td>7.89</td>
<td>15.92</td>
<td>6.58</td>
<td>7.89</td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>0.2</td>
<td>9.04</td>
<td>3.03</td>
<td>5.51</td>
<td>9.04</td>
<td>3.03</td>
<td>5.51</td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>0.2</td>
<td>12.25</td>
<td>7.20</td>
<td>8.33</td>
<td>12.25</td>
<td>7.20</td>
<td>8.33</td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>0.2</td>
<td>10.06</td>
<td>4.93</td>
<td>6.11</td>
<td>10.06</td>
<td>4.93</td>
<td>6.11</td>
</tr>
</tbody>
</table>

Exactly same combination of experiments are performed for LSTM and BiLSTM models as mentioned above, with different number of layers and different number of units, and a 0.2 dropout for time steps (look back) 3 days and 10 days to predict Google’s adjusted close price for next day, and observed that time-steps of 1 day, achieved better MAPE compared to time steps of 3 days and 10 days for both LSTM and BiLSTM models, as shown in Table 5.5. For single layer LSTM with 1 day time step MAPE resulted in 10.06, where as for the same model with 3 days and 10 days, MAPE resulted in 13.98, and 14.4 respectively. And, for single layer BiLSTM resulted in MAPE of 8.06, 9.72 and 12.53 for 1 day, 3 days and 10 days respectively. Other combinations of different number of layers, different number of units and 0.2 dropout for both LSTM and BiLSTM models, shown in Table 5.5, shows 1 day time step, achieved better MAPE value. That means, as the time steps increased, MAPE increased. Both LSTM and BiLSTM models predict better with least time steps.
5.7 Evaluation and Results of Experiment 4 - Predicting Adj close price using Historic Stock Price and Technical Indicators

The evaluation metric selected is MAPE, explained in section 3.6.2. The final MAPE of LSTM models are obtained by taking the mean of all the MAPE values from cross-validation. Single layer LSTM with 32 units/neurons with 1 day time-step (look-back) achieved MAPE of 17.73 for prediction of Google adjusted close price for next day, whereas the BiLSTM model with 32 units/neurons with 1 day time-step (look-back) has shown tiny improvement in MAPE of 17.68, saying that resources used by LSTM when measured in terms of 'Train time' is 5.80 seconds, where as BiLSTM took 6.29 seconds training time. MAPE have improved for both LSTM and BiLSTM by adding an extra layer for each models, and doubling the number of units, ie., 64 neurons, resulting in MAPE of 16.94 for LSTM and 16.85 for BiLSTM. At the same time adding extra layer has increased in training times for these models. 7.64 seconds and 12.54 seconds. It is observed, still it isn’t big difference in MAPE between both models, but the training time taken for BiLSTM is higher than the time taken by LSTM. These LSTM and BiLSTM models are further experimented, by adding a small dropout of 0.2. That is with number of layer as 2 and number of units as 64 and a dropout of 0.2. Results show, MAPE have improved compared two first two models, with 13.34 for LSTM and 11.94 for BiLSTM. A good few experiments are conducted, with different combinations of number of layers and number of units with, dropout of 0.2 and time step of 1 day. And results shown in Table 5.3, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models, BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models. Best preformed architecture for both LSTM and BiLSTM is 4 layered with 150 units and 0.2 dropout with 1 day time step, where LSTM achieved a MAPE of
3.83 and BiLSTM achieved better MAPE of 3.66. And results shown in Table 5.6, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

<table>
<thead>
<tr>
<th>Layer</th>
<th>MAPE 1 day</th>
<th>MAPE 3 days</th>
<th>MAPE 10 days</th>
<th>Train time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17.68</td>
<td>17.73</td>
<td>18.45</td>
<td>15.08</td>
</tr>
<tr>
<td>2</td>
<td>16.53</td>
<td>17.41</td>
<td>18.41</td>
<td>28.12</td>
</tr>
<tr>
<td>3</td>
<td>15.37</td>
<td>16.48</td>
<td>18.96</td>
<td>46.91</td>
</tr>
</tbody>
</table>

Table 5.6: Learning results using Historic Stock Price and Technical Indicators

Exactly same combination of experiments are performed for LSTM and BiLSTM models as mentioned above, with different number of layers and number of units, and a 0.2 dropout for time steps (look back) 3 days and 10 days to predict Google’s adjusted close price for next day, and observed that time-steps of 1 day achieved better MAPE compared to time steps of 3 days and 10 days for both LSTM and BiLSTM models, as shown in Table 5.6. For single layer LSTM with 1 day time step MAPE resulted in 17.73, where as for the same model with 3 days and 10 days, MAPE resulted in 18.45, and 19.74 respectively. And, for single layer BiLSTM resulted in MAPE of 17.68, 18.41 and 18.96 for 1 day, 3 days and 10 days respectively. Other combinations of different number of layers, different number of units and 0.2 dropout for both LSTM and BiLSTM models, shown in Table 5.6, shows 1 day time step, achieved better MAPE value. That means, as the time steps increased, MAPE increased. Both LSTM and BiLSTM models predict better with least time steps.
5.8 Evaluation and Results of Experiment 5 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied

The evaluation metric selected for this research, is MAPE (explained in section 3.6.2). The final MAPE of LSTM and BiLSTM models are obtained by taking the mean of all the MAPE values from cross-validation. Single layer LSTM with 32 units/neurons, and with 1 day time-step (look-back) achieved MAPE of 10.91 for prediction of Google adjusted close price for next day, whereas the BiLSTM model with 32 units/neurons, and with 1 day time-step (look-back) has shown improvement in MAPE of 10.42. For training these models, the resources used by LSTM when measured in terms of 'Train time' is 3.59 seconds, whereas BiLSTM took 4.46 seconds training time. MAPE have improved for both LSTM and BiLSTM by adding an extra layer for each models, and doubling the number of units, i.e., 64 neurons, resulting in MAPE of 8.33 for LSTM and 8.06 for BiLSTM. At the same time adding extra layer has increased in training times for these models, LSTM took 5.59 seconds and BiLSTM took 10.45 seconds. Further these LSTM and BiLSTM models are fitted, by adding a small dropout of 0.2. That is, with number of layer as 2 and number of units as 64 and a dropout of 0.2. Results show, MAPE have improved compared two first two models, with 4.53 for LSTM and 4.44 for BiLSTM. It is observed, still it isn’t big difference in MAPE between both models, but the training time taken for BiLSTM is nearly twice the time taken by LSTM, with 4.80 seconds for LSTM and 15.81 for BiLSTM. A good few experiments are conducted, with different combinations of number of layers and number of units with, dropout of 0.2 and time step of 1 day. Observed MAPE values show, with increase in number of layers, MAPE keeps dropping until a threshold of 4 layers, but with 6 layers architecture, MAPE started increasing for both LSTM and BiLSTM models. Best preformed architecture for both LSTM and BiLSTM is 4 layered, with 150 units and 0.2 dropout with 1 day time step, where LSTM achieved a MAPE of 1.98
and BiLSTM achieved better MAPE of 1.86. And the training time for these models are 9.86 seconds for LSTM and, 21.86 seconds for BiLSTM. Results shown in Table 5.7, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all architectures, BiLSTM models performed better than LSTM models with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

<table>
<thead>
<tr>
<th>Look-back Value</th>
<th>Number of neurons</th>
<th>Train MAE</th>
<th>Test MAE</th>
<th>Train MAPE</th>
<th>Test MAPE</th>
<th>Test RMSE</th>
<th>Train time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>19.44</td>
<td>12.01</td>
<td>10.18</td>
<td>11.95</td>
<td>17.54</td>
<td>36.88</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>20.96</td>
<td>17.68</td>
<td>11.36</td>
<td>11.01</td>
<td>16.22</td>
<td>20.39</td>
</tr>
<tr>
<td>6</td>
<td>93</td>
<td>15.56</td>
<td>12.66</td>
<td>2.17</td>
<td>12.89</td>
<td>22.35</td>
<td>26.82</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>20.21</td>
<td>15.01</td>
<td>2.45</td>
<td>11.03</td>
<td>20.39</td>
<td>20.97</td>
</tr>
<tr>
<td>1</td>
<td>32</td>
<td>16.13</td>
<td>11.33</td>
<td>2.13</td>
<td>11.01</td>
<td>19.14</td>
<td>11.09</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>15.48</td>
<td>16.71</td>
<td>2.22</td>
<td>12.89</td>
<td>19.14</td>
<td>15.44</td>
</tr>
<tr>
<td>6</td>
<td>93</td>
<td>12.33</td>
<td>14.07</td>
<td>2.84</td>
<td>10.42</td>
<td>23.08</td>
<td>26.37</td>
</tr>
</tbody>
</table>

Table 5.7: Learning results using Historic Stock Price and Technical Indicators with PCA Applied

Exactly same combination of experiments are performed for LSTM and BiLSTM models as mentioned above, with different number of layers and number of units, and a 0.2 dropout for time steps 3 days and 10 days as well, to predict Google’s adjusted close price for next day, and observed that time-steps of 1 day, achieved better MAPE compared to time steps of 3 days and 10 days for both LSTM and BiLSTM models, as shown in Table 5.7. For single layer LSTM with 1 day time step MAPE resulted in 10.91, where as for the same model with 3 days and 10 days, MAPE resulted in 11.03, and 12.89 respectively. And, for single layer BiLSTM resulted in MAPE of 10.42, 11.01 and 11.95 for 1 day, 3 days and 10 days respectively. Table 5.7, shows the results for all architecture with number of layers, number of units and 0.2 dropout for both LSTM and BiLSTM models. 1 day time step, achieved better MAPE value. That means, as the time steps increased, MAPE increased. Both LSTM and BiLSTM models predict better with least time steps.
5.9 Evaluation and Results of 6 - Predicting Adj close price using Historic Stock Price, New Articles and Stock Tweets

The evaluation metric selected is MAPE, explained in section 3.6.2. The final MAPE of LSTM models are obtained by taking the mean of all the MAPE values from cross-validation. Single layer LSTM with 32 units/neurons with 1 day time-step (look-back) achieved MAPE of 12.82 for prediction of Google adjusted close price for next day, whereas the BiLSTM model with 32 units/neurons with 1 day time-step (look-back) has shown tiny improvement in MAPE of 12.46, saying that resources used by LSTM when measured in terms of 'Train time' is 4.58 seconds, whereas BiLSTM took 5.33 seconds training time. MAPE have improved for both LSTM and BiLSTM by adding an extra layer for each models, and doubling the number of units, i.e., 64 neurons, resulting in MAPE of 9.03 for LSTM and 8.34 for BiLSTM. At the same time adding extra layer has increased in training times for these models, 6.94 seconds and 17.74 seconds. It is observed, still it isn’t big difference in MAPE between both models, but the training time taken for BiLSTM is nearly twice the time taken by LSTM. These LSTM and BiLSTM models are further experimented, by adding a small dropout of 0.2. That is with number of layer as 2 and number of units as 64 and a dropout of 0.2. Results show, MAPE have improved compared two first two models, with 5.46 for LSTM and 5.4 for BiLSTM. A good few experiments are conducted, with different combinations of number of layers and number of units with, dropout of 0.2 and time step of 1 day. And results shown in Table 5.3, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

Best preformed architecture for both LSTM and BiLSTM is 4 layered with 150 units and 0.2 dropout with 1 day time step, where LSTM achieved a MAPE of 2.34
and BiLSTM achieved better MAPE of 2.31. And results shown in Table 5.8, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

<table>
<thead>
<tr>
<th>Number of Layers</th>
<th>Dropout</th>
<th>Look-back</th>
<th>Train MAPE</th>
<th>Test MAPE</th>
<th>Train RMSE</th>
<th>Test RMSE</th>
<th>Train time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2</td>
<td>1</td>
<td>12.82</td>
<td>19.10</td>
<td>16.45</td>
<td>17.18</td>
<td>14.59</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>1</td>
<td>12.46</td>
<td>13.72</td>
<td>12.82</td>
<td>12.96</td>
<td>14.59</td>
</tr>
<tr>
<td>3</td>
<td>0.2</td>
<td>1</td>
<td>12.39</td>
<td>13.69</td>
<td>12.65</td>
<td>12.82</td>
<td>14.59</td>
</tr>
<tr>
<td>4</td>
<td>0.2</td>
<td>1</td>
<td>12.39</td>
<td>13.72</td>
<td>12.82</td>
<td>12.96</td>
<td>14.59</td>
</tr>
<tr>
<td>5</td>
<td>0.2</td>
<td>1</td>
<td>12.39</td>
<td>13.69</td>
<td>12.65</td>
<td>12.82</td>
<td>14.59</td>
</tr>
<tr>
<td>6</td>
<td>0.2</td>
<td>1</td>
<td>12.39</td>
<td>13.72</td>
<td>12.82</td>
<td>12.96</td>
<td>14.59</td>
</tr>
</tbody>
</table>

Table 5.8: Learning results using Historic Stock Price, New Articles and Stock Tweets

Exactly same combination of experiments are performed for LSTM and BiLSTM models as mentioned above, with different number of layers and number of units, and a 0.2 dropout for time steps (look back) 3 days and 10 days to predict Google’s adjusted close price for next day, and observed that time-steps of 1 day, achieved better MAPE compared to time steps of 3 days and 10 days for both LSTM and BiLSTM models, as shown in Table 5.8. For single layer LSTM with 1 day time step MAPE resulted in 12.82, where as for the same model with 3 days and 10 days, MAPE resulted in 16.45, and 19.1 respectively. And, for single layer BiLSTM resulted in MAPE of 12.46, 13.72 and 17.18 for 1 day, 3 days and 10 days respectively. Other combinations of number of layers, number of units and 0.2 dropout for both LSTM and BiLSTM models, shown in Table 5.8, shows 1 day time step, achieved better MAPE value. That means, as the time steps increased, MAPE increased. Both LSTM and BiLSTM models predict better with least time steps.
5.10 Evaluation and Results of Experiment 7 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, plus News articles

The evaluation metric selected is MAPE, explained in section 3.6.2. The final MAPE of LSTM models are obtained by taking the mean of all the MAPE values from cross-validation. Single layer LSTM with 32 units/neurons with 1 day time-step (look-back) achieved MAPE of 14.96 for prediction of Google adjusted close price for next day, whereas the BiLSTM model with 32 units/neurons with 1 day time-step (look-back) has shown tiny improvement in MAPE of 14.41, saying that resources used by LSTM when measured in terms of ‘Train time’ is 10.65 seconds, where as BiLSTM took 17.25 seconds training time. MAPE have improved for both LSTM and BiLSTM by adding an extra layer for each models, and doubling the number of units, ie., 64 neurons, resulting in MAPE of 11.19 for LSTM and 9.6 for BiLSTM. At the same time adding extra layer has increased in training times for these models. 9.48 seconds and 14.58 seconds. It is observed, still it isn’t big difference in MAPE between both models, but the training time taken for BiLSTM is much higher than the time taken by LSTM. These LSTM and BiLSTM models are further experimented, by adding a small dropout of 0.2. That is with number of layer as 2 and number of units as 64 and a dropout of 0.2. Results show, MAPE have improved compared two first two models, with 6.49 for LSTM and 6.1 for BiLSTM. A good few experiments are conducted, with different combinations of number of layers and number of units with, dropout of 0.2 and time step of 1 day. And results shown in Table 5.3, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models, BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models. Best preformed architecture for both LSTM and BiLSTM is 4 layered with
150 units and 0.2 dropout with 1 day time step, where LSTM achieved a MAPE of 4.69 and BiLSTM achieved better MAPE of 4.22. And results shown in Table 5.9, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

<table>
<thead>
<tr>
<th>Number of layers</th>
<th>Number of neurons</th>
<th>Dropout after layer</th>
<th>Look-back value</th>
<th>Train MAPE</th>
<th>Test MAE</th>
<th>Test RMSE</th>
<th>Test MAPE</th>
<th>Train time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>0.2</td>
<td>1</td>
<td>7.5</td>
<td>7.08</td>
<td>6.95</td>
<td>45.08</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>0.2</td>
<td>1</td>
<td>8.29</td>
<td>7.4</td>
<td>6.67</td>
<td>36.41</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>150</td>
<td>0.2</td>
<td>1</td>
<td>9.05</td>
<td>7.24</td>
<td>6.97</td>
<td>47.45</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.9: Learning results using Historic Stock Price and Technical Indicators with PCA Applied, plus News articles

Exactly same combination of experiments are performed for LSTM and BiLSTM models as mentioned above, with different number of layers and number of units, and a 0.2 dropout for time steps 3 days and 10 days to predict Google’s adjusted close price for next day, and observed that time-steps of 1 day, achieved better MAPE compared to time steps of 3 days and 10 days for both LSTM and BiLSTM models, as shown in Table 5.9. For single layer LSTM with 1 day time step MAPE resulted in 14.96, where as for the same model with 3 days and 10 days, MAPE resulted in 15.75, and 16.93 respectively. And, for single layer BiLSTM resulted in MAPE of 14.41, 15.64 and 15.83 for 1 day, 3 days and 10 days respectively. Other combinations of number of layers, number of units and 0.2 dropout for both LSTM and BiLSTM models, shown in Table 5.9, shows 1 day time step, achieved better MAPE value. That means, as the time steps increased, MAPE increased. Both LSTM and BiLSTM models predict better with least time steps.
5.11 Evaluation and Results of Experiment 8 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, plus Stock Tweets

The evaluation metric selected is MAPE, explained in section 3.6.2. The final MAPE of LSTM models are obtained by taking the mean of all the MAPE values from cross-validation. Single layer LSTM with 32 units/neurons with 1 day time-step (look-back) achieved MAPE of 7.81 for prediction of Google adjusted close price for next day, whereas the BiLSTM model with 32 units/neurons with 1 day time-step (look-back) has shown improvement in MAPE of 6.36, saying that resources used by LSTM when measured in terms of 'Train time’ is 14.90 seconds, where as BiLSTM took 17.75 seconds training time. MAPE have improved for both LSTM and BiLSTM by adding an extra layer for each models, and doubling the number of units, ie., 64 neurons, resulting in MAPE of 6.2 for LSTM and 5.83 for BiLSTM. At the same time adding extra layer has increased in training times for these models. 10.88 seconds and 15.56 seconds. It is observed, still it isn’t big difference in MAPE between both models, but the training time taken for BiLSTM is higher than the time taken by LSTM. These LSTM and BiLSTM models are further experimented, by adding a small dropout of 0.2. That is with number of layer as 2 and number of units as 64 and a dropout of 0.2. Results show, MAPE have improved compared two first two models, with 4.98 for LSTM and 4.79 for BiLSTM. A good few experiments are conducted, with different combinations of number of layers and number of units with, dropout of 0.2 and time step of 1 day. And results shown in Table 5.3, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM, models BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models. Best preformed architecture for both LSTM and BiLSTM is 4 layered with 150 units
and 0.2 dropout with 1 day time step, where LSTM achieved a MAPE of 3.92 and BiLSTM achieved better MAPE of 3.24. And results shown in Table 5.10, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

| LSTM | BiLSTM | LSTM | BiLSTM | LSTM | BiLSTM | LSTM | BiLSTM | LSTM | BiLSTM | LSTM | BiLSTM | LSTM | BiLSTM | LSTM | BiLSTM | LSTM | BiLSTM | LSTM | BiLSTM | LSTM | BiLSTM |
|------|--------|------|--------|------|--------|------|--------|------|--------|------|--------|------|--------|------|--------|------|--------|------|--------|------|--------|------|--------|
| 0.2  | 50     | 0.2  | 50     | 0.2  | 50     | 0.2  | 50     | 0.2  | 50     | 0.2  | 50     | 0.2  | 50     | 0.2  | 50     | 0.2  | 50     | 0.2  | 50     | 0.2  | 50     | 0.2  | 50     |

Table 5.10: Learning results using Historic Stock Price and Technical Indicators with PCA Applied and Stock Tweets

Exactly same combination of experiments are performed for LSTM and BiLSTM models as mentioned above, with different number of layers and number of units, and a 0.2 dropout for time steps (look back) 3 days and 10 days to predict Google’s adjusted close price for next day, and observed that time-steps of 1 day, achieved better MAPE compared to time steps of 3 days and 10 days for both LSTM and BiLSTM models, as shown in Table 5.10. For single layer LSTM with 1 day time step MAPE resulted in 7.81, where as for the same model with 3 days and 10 days, MAPE resulted in 8.26, and 11.41 respectively. And, for single layer BiLSTM resulted in MAPE of 6.36, 6.96 and 8.69 for 1 day, 3 days and 10 days respectively. Other combinations of number of layers, number of units and 0.2 dropout for both LSTM and BiLSTM models, shown in Table 5.10, shows 1 day time step, achieved better MAPE value. That means, as the time steps increased, MAPE increased. Both LSTM and BiLSTM models predict better with least time steps.
5.12 Evaluation and Results of Experiment 9 - Predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets

The evaluation metric selected is MAPE, explained in section 3.6.2. The final MAPE of LSTM models are obtained by taking the mean of all the MAPE values from cross-validation. Single layer LSTM with 32 units/neurons with 1 day time-step (look-back) achieved MAPE of 12.55 for prediction of Google adjusted close price for next day, whereas the BiLSTM model with 32 units/neurons with 1 day time-step (look-back) has shown tiny improvement in MAPE of 12.49, saying that resources used by LSTM when measured in terms of 'Train time' is 10.19 seconds, where as BiLSTM took 13.50 seconds training time. MAPE have improved for both LSTM and BiLSTM by adding an extra layer for each models, and doubling the number of units, ie., 64 neurons, resulting in MAPE of 9.28 for LSTM and 9.07 for BiLSTM. At the same time adding extra layer has increased in training times for these models. 8.08 seconds and 12.95 seconds. It is observed, still it isn’t big difference in MAPE between both models, but the training time taken for BiLSTM is nearly twice the time taken by LSTM. These LSTM and BiLSTM models are further experimented, by adding a small dropout of 0.2. That is with number of layer as 2 and number of units as 64 and a dropout of 0.2. Results show, MAPE have improved compared two first two models, with 5.24 for LSTM and 5.14 for BiLSTM. A good few experiments are conducted, with different combinations of number of layers and number of units with, dropout of 0.2 and time step of 1 day. And results shown in Table 5.3, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models, BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models. Best preformed architecture for both LSTM and BiLSTM is 4 layered with
150 units and 0.2 dropout with 1 day time step, where LSTM achieved a MAPE of 4.85 and BiLSTM achieved better MAPE of 4.36. And results shown in Table 5.11, as number of layers increases and adding little dropout, the MAPE improves, but at the same time, the time to train the models increased with adding more layers. In all comparison to LSTM models BiLSTM models performed better with lower MAPE, but as number of layers increased BiLSTM training time increased drastically compared to LSTM models.

![Table 5.11: Learning results using Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets](image)

Exactly same combination of experiments are performed for LSTM and BiLSTM models as mentioned above, with different number of layers and number of units, and a 0.2 dropout for time steps (look back) 3 days and 10 days to predict Google’s adjusted close price for next day, and observed that time-steps of 1 day, achieved better MAPE compared to time steps of 3 days and 10 days for both LSTM and BiLSTM models, as shown in Table 5.11. For single layer LSTM with 1 day time step MAPE resulted in 12.55, where as for the same model with 3 days and 10 days, MAPE resulted in 15.69, and 18.37 respectively. And, for single layer BiLSTM resulted in MAPE of 12.49, 13.06 and 16.31 for 1 day, 3 days and 10 days respectively. Other combinations of number of layers, number of units and 0.2 dropout for both LSTM and BiLSTM models, shown in Table 5.11, shows 1 day time step, achieved better MAPE value. That means, as the time steps increased, MAPE increased. Both LSTM and BiLSTM models predict better with least time steps.
5.13 Statistical Evaluation of Hypothesis

From the error statistics results of BiLSTM and LSTM in predicting the adjusted closing price of Google NASDAQ stocks shown in Tables 5.3.3 to 5.3.11, BiLSTM performed better than LSTM, with BiLSTM exhibiting a smaller error measurement than LSTM. However, these statistics are insufficient to determine whether BiLSTM is superior to LSTM, as the difference in error statistics may be due to chance. Therefore, the Diebold-Marino (DM) test (Section 3.6.5) is used in this study to evaluate the prediction accuracy of both models and to differentiate the prediction performance of the BiLSTM and LSTM models. In certain circumstances, it’s clear to choose the model with the lesser MAPE error measurement. Is it necessary to assess if this difference is substantial (for time series redaction) or due to sample data values.

Unlike NASDAQ stocks of Google, where 324 experiments with 12 different LSTM and BiLSTM models with 3 different time steps and 9 different combinations of datasets are performed, due to time constraints, all of these experiments are not repeated for Apple, Facebook, and Amazon with different time steps, architectures, and datasets (which would be another 972 experiments), but rather this research, used the best performed model with multi source data to predict adjusted close price which is the main objective of the research. To predict the NASDAQ stock for Apple, Facebook, and Amazon, 4-layer Adam optimized LSTM and BiLSTM networks with 150 units, 0.2 dropout, and 1 day time step are used, along with Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets dataset. The BiLSTM model outperformed the LSTM model in predicting adjusted close for Apple, Facebook, and Amazon, with lower MAPE values of 3.98, 3.15, and 4.02, compared to LSTM MAPE values of 5.95, 5.91 and 6.31. The predicted vs. actual adjusted close price plot is shown in Figure 5.3. The DM test is used in this section to compare the forecasting performance of the BiLSTM and LSTM models. The DM test is run on the forecasting results of BiLSTM and LSTM in predicting Adj close price using Historic Stock Price and Technical Indicators with PCA Applied, as well as the News articles and Stock Tweets dataset.
CHAPTER 5. EVALUATION AND RESULTS

<table>
<thead>
<tr>
<th>Stock Label</th>
<th>DM test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$dt$</td>
</tr>
<tr>
<td>Google</td>
<td>1.98</td>
</tr>
<tr>
<td>Apple</td>
<td>1.99</td>
</tr>
<tr>
<td>Facebook</td>
<td>2.76</td>
</tr>
<tr>
<td>Amazon</td>
<td>2.18</td>
</tr>
</tbody>
</table>

Table 5.12: Diebold-Marino test results for BiLSTM and LSTM models in predicting Adj close price using 'Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets’ dataset

This research uses the Diebold-Mariano(DM) test to determine whether the two forecasts are significantly different. Python function dm_test is used from (Tsang, 2017), which implements the (Diebold & Mariano, 1995b) to statistically identify forecast accuracy difference. From the DM test results in table 5.12, observed $dt$ scores are $> 0$ that is LSTM MAPE are higher than BiLSTM as explained in section 3.6.5, also the null hypothesis is rejected at the 5 percent level of significance, indicating that the observed differences are significant since the DM test statistic for Google is $(dt) = 1.98 > 1.96$ and $p - Value < 0.03$, Apple is $(dt) = 1.99 > 1.96$ and $p - Value < 0.04$, Amazon is $(dt) = 2.18 > 1.96$ and $p - Value < 0.035$, and Facebook is $(dt) = 2.76 > 1.96$ and $p - Value < 0.028$. Furthermore, the positive value of the Diebold–Mariano test indicates that the absolute value of errors of the LSTM model is greater than that of BiLSTM models, implying that BiLSTM models outperform LSTM models in predicting the adjusted close price for Google, Apple, Amazon and Facebook.

Based on the findings, this study rejects the null hypothesis and accepts the alternative hypothesis, concluding that, ”If a model is trained with the Bi-LSTM technique using data from multiple sources and optimized with the Adam algorithm, it will have a lower MAPE than a model trained with LSTM technique using the same set of data in the prediction of adjusted close price of NASDAQ stock prices”.  

103
5.14 BiLSTM Prediction of adjusted close price

Figure 5.3: Predicted vs Actual adjusted close prices

Figure 5.3, shows the 'Predicted vs Actual adjusted close prices' using Historic Stock Price and Technical Indicators with PCA, plus News articles and Stock Tweets of NASDAQ stocks of Google, Apple, Facebook, and Amazon, using a 1 day time step and a 4 Layered BiLSTM model with 150 units and 0.2 dropout.

While the exact price points from the predicted adjusted close price were not always very close to the actual price, these models did still indicate overall trends such as going up or down, demonstrating that both LSTMs and BiLSTMs can be effective in times series forecasting. Almost in every experiment, BiLSTM outperformed LSTM in a similar architecture by achieving a higher MAPE, demonstrating that BiLSTM outperforms LSTM in time series forecasting. Figure 5.3 shows that the BiLSTM model is capable of accurately predicting the direction of stock movement; this model could be used as a recommendation model for future next day stock direction prediction, that could be used for suggesting the stock investors whether to sell or buy the stocks.
5.15 Overview

This section evaluated the outcomes of the experimental implementations described in Chapter 4. The baseline LSTM model results, demonstrated that the model was capable of fitting the underlying overall trend of Adj close price but produced a higher MAPE value than BiLSTM across all dataset combinations. The proposed BiLSTM model, when trained and tested with combinations of historic stock prices, technical indicators, News articles, and tweet datasets, produced highly noised predictions and high MAPE values when compared to other experiments with the BiLSTM model on other datasets. The stacked 4 layer BiLSTM model with 150 units and 0.2 dropout performed best with the stock price and technical indicators datasets with PCA where the predictions were not noised and achieved the lowest MAPE of 1.86. The same model trained with multiple data sources produced a higher MAPE value of 4.36 than the model trained with only historic stock prices, which produced an overall MAPE of 1.92. Finally, the chapter concludes by defining the research hypothesis; because the proposed BiLSTM model outperformed the baseline LSTM model, the null hypothesis is rejected, and the alternate hypothesis is accepted, thereby satisfying the research objective.
Chapter 6

Conclusion

This chapter provides an overview of the research objective, design, and methodology used, experiments carried out to achieve the objective, and evaluation of the experimental results. Finally, the chapter concludes by recommending future works that can improve this research and its contributions to the field of stock market prediction.

6.1 Research Overview

Stock market prediction is popular among academics and financial gurus, but no research has established a 100% successful model or technique. This research used a Bi-directional LSTM model to forecast the adjusted closing price of stocks with the lowest MAPE value. Researchers reviewed and analyzed the literature to learn about the market and investor sentiment, as well as the essential data. According to study, News, social networking sites like Twitter, stock data, and technical indicators affect the market. Several research use supervised learning to train and evaluate NLP models on labeled datasets to extract sentiments from News articles and Tweets. This study employs an fine-tuning technique, to extract attitudes from unlabeled News and Tweets using pre-trained BERT, DistilBERT, RoBERTa, and ALBERT transformer models. Then, it chooses the optimal transformer to classify unseen News articles and Tweets.
According to the literature reviews, just a few technical indicators were used to anticipate stock values. This study used 28 technical indicators and employed PCA, to solve the dimensionality curse which has been observed in literature reviews with large number of features. This research used 1-day, 3-day, and 10-day time steps to evaluate stock forecast, which had limited study in past. 1 day time step forecasted the adjusted closing price better than 3 days and 10 days steps. This study improved deep learning models using Adam optimization.

### 6.2 Problem Definition

The purpose of this dissertation is to investigate the possibility of predicting stock market movements using 1 day ahead financial time series and to make recommendations on whether to sell or buy stocks based on the stock direction movement. The research looks at recurrent neural networks with bi-directional long short-term memory (BiLSTM). The goal of this dissertation is to evaluate BiLSTM’s performance on financial time series data, as well as how market and investor sentiment affect stock prediction. Also, how the number of time steps for BiLSTM affects the model’s predictive power. This dissertation believes that there have been many studies that use models to predict stock market trends offline, but the results were not satisfactory. Consider developing a model that can predict stock prices both offline and in real-time, that is, during market hours and non-market hours.

This dissertation’s research question, including the secondary question, can be formalized as follows:

* How does a model trained with Bi-directional LSTM outperform a model trained with Uni-directional LSTM in terms of accuracy by incorporating both investor and market sentiments?
* Is this worthwhile to fine tune transformer language models BERT, DistilBERT, RoBERTa and ALBERT with relevant financial datasets than using just FinBERT for sentimental analysis of News articles and Tweets?
What effect does the number of time steps have on the BiLSTM model’s ability to predict the adjusted close price of NASDAQ stocks?

How well do the BiLSTM models that have been built predict financial time series?

Would incorporating multi-source than just historical price data produce better prediction results?

Is the PCA feature extraction technique capable of improving model prediction results?

Would stacked BiLSTM models outperform a single BiLSTM model in predicting the adjusted close price of NASDAQ stocks?

6.3 Experiment, Evaluation, and Result

In this research, all steps of the CRISP-DM approach are included except deployment, which is out of scope. Google, Apple, Facebook, and Amazon’s NASDAQ stock prices will be scarped from January 1, 2020, through December 31, 2021. As part of feature engineering, the collected data is pre-processed, and 23 technical indicators are computed. To extract the unlabeled News and Tweet sentiments of Google, Apple, Facebook and Amazon, this research used the RoBERTa model that performed better than BERT, DistilBERT, and ALBERT when fine-tuned using a labeled financial News and Tweet dataset.

Following the creation of a baseline LSTM model and a proposed BiLSTM model, all models are evaluated based on their MAPE. Both the LSTM and BiLSTM models are trained and evaluated using various dataset combinations with 1 day, 3 day, and 10 day time steps. By varying the number of layers and units with and without dropout, several architectures of LSTM and BiLSTM models, both single and stacked, are created. Experiments are run on nine dataset combinations produced by combining five datasets namely, historic stock price, News sentiment polarity, tweet sentiment polarity, technical indicators without PCA, and technical indicators with PCA. In this work, expanding window validation was used to divide time series data into train
CHAPTER 6. CONCLUSION

and test datasets, which is more robust than traditional train-test splitting procedures.

The stacked 4 layer Uni-directional LSTM model with 150 units and 0.2 dropout produced higher MAPE values (1.98 to 5.18) than the stacked 4 layer Bi-directional LSTM model (1.86 to 4.36) for all nine experiments with different dataset combinations and 1 day time step. The proposed BiLSTM model performed best with past stock price data and technical indicators with PCA with resulted MAPE of 3.68 and 4.03, respectively. In experiments using News and Tweets data, the suggested BiLSTM scored the highest MAPE due to an unbalanced dataset in which "Neutral" sentiment data outweighed "Positive" and "Negative" sentiment data. 1 day time step yielded greater MAPE than 3 days and 10 days for all dataset combinations and LSTM and BiLSTM architectures. In other words, the shorter the time step value, the higher the prediction accuracy of the adjusted close price. According to the findings, stacked LSTM and BiLSTM models outperformed single layer models for all dataset combinations up to a threshold of 4 layers, beyond which the MAPE began to increase for 6 layered models. In other words, increasing layers only enhanced the models up to a degree, while adding more layers worsened the model’s performance. The loss of models continues to improve (i.e. be reduced) as the number of layers grows, as seen in Figure 5.1, however there is no consistency in the loss reduction for the 6 layered model, as demonstrated by the frequent big spikes in loss. BiLSTM also took longer to train than LSTM, which might be explained by the fact that BiLSTM traverses the input data twice, once left-to-right and once right-to-left. And, as the number of layers grows, BiLSTM training time increases significantly faster than LSTM; in some cases, BiLSTM training time nearly twice that of LSTM, yet the gain in MAPE was not always remarkable when compared to LSTM. The findings demonstrated that the proposed BiLSTM models had the lowest MAPE in all experiments with all combinations of datasets compared to the LSTM models, which met the study aim. Thus, the null hypothesis is rejected and concluded that, “If a model is trained with the Bi-LSTM technique using data from multiple sources and optimized with the Adam algorithm, it will have a lower MAPE than a model trained with LSTM technique using the same set of data in the prediction of adjusted close price of NASDAQ stock prices”.

109
6.4 Contributions and impact

Uni-directional and Bi-directional LSTM networks were developed in this dissertation to predict the Adj close price of NASDAQ stocks of Google, Apple, Amazon, and Facebook using multi-source data, 'Historic Stock Price and Technical Indicators with PCA Applied, plus News articles and Stock Tweets'. These models are designed to function both offline and in real time, as the 'Close price' feature from the stock price is not used to train the models, as its only available during market hours. RoBERTa outperformed other transformers from BERT family including finBERT, by employing fine-tuning technique. The models were evaluated using multi source dataset. To evaluate the suitability of various architectures and modeling parameters, a comprehensive and rigorous procedure was used. The Bi-directional LSTM outperformed the Uni-directional LSTM in the results. Multiple LSTM and BiLSTM layers architectural models were used for Adj close price prediction with different time steps and data sources. The findings also highlighted the significance of time steps in predicting the Adj close price. It was discovered that using time steps of 1 day yielded more accurate results than time steps of 3 days and 10 days. Simultaneously, in some architectures with stacked layers, the training time of BiLSTM is nearly twice that of LSTM. So, while BiLSTM outperforms LSTM in terms of prediction accuracy, it consumes a lot of resources due to its bidirectional processing.

Experiments revealed that the 4-layered BiLSTM with 150 units, 0.2 dropout, and 1 day time step outperformed other BiLSTM and LSTM architectures for various data source combinations. Another contribution of this work was to investigate model validation and its transferability. The examination used many datasets. Historic stock price with technical indicators and PCA applied dataset produced the lowest MAPE values. Stock prices and technical indicators without PCA have a greater MAPE than datasets with PCA. After optimizing the model using Adam, it’s tested on test data. Adam technique is used to optimize the model, and findings demonstrate BiLSTM effectively predict future NASDAQ Adj closing prices.
6.5 Future Work and recommendations

Although this research has taken new approaches, there are still some areas that need to be explored further:

i. The XLNet Language model may be compared to RoBERTa for sentiment classification of financial qualitative data since this research considers that models from the BERT family are the best performing transformer models for sentiment extraction.

ii. To get denoised forecasts and enhanced MAPE values with the proposed models, several sampling strategies, such as over or under-sampling, may be applied.

iii. Hybrid models could be used to determine whether prediction accuracy improves further.

iv. Because the focus of this research was on machine learning and deep learning, and due to the time constraints of 16 weeks, this research did not look into implementing Big Data concepts such as streaming live data and continuously updating the model. This would be an excellent addition to keep the model up to date.

v. This study employed two groups of ta-lib technical indicators groups, totaling 23 in number. ta-lib contains ten groups and hundreds of technical indicators. It will be interesting to see if including all of the technical indicators by applying PCA on them, will improve model accuracy or not.

vi. This study employs the Adam model optimization technique; it would be interesting to see if Root Mean Square Propagation optimization techniques would improve model accuracy.

vii. This study employed a 20-day window size to calculate technical indicators; shorter window size, such as 5 days, may be studied to see how the forecast accuracy of the models improves.
References


REFERENCES


REFERENCES


REFERENCES

Henry, G. (2021). Improved sentiment analysis using a customized distilbert nlp configuration. In (Vol. 3(2)). Retrieved from https://www.academia.edu/56705388/Improved_Sentiment_Analysis_using_a_Customized_Distilbert_NLP_Configuration


REFERENCES


Kouloumpis, E., Wilson, T., & Moore, J. (2011, 01). Twitter sentiment analysis: The good the bad and the omg!. Retrieved from https://www.researchgate.net/publication/221297835_Twitter_Sentiment_Analysis_The_Good_the_Bad_and_the_OMG


REFERENCES


Selvin, S., Vinaya kumar, R., Gopala krishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. *Inter-
REFERENCES

national Conference on Advances in Computing, Communications and Informatics (ICACCI), 1643-1647. doi: 10.1109/ICACCI.2017.8126078


REFERENCES


127


doi: 10.1093/biomet/92.4.937

doi: 10.3390/su11061579
REFERENCES


Appendix A

Additional content

(a) News Articles Sentiment Extraction

(b) Twitter Tweets Sentiment Extraction

Figure A.1: RoBERTa sentimental analysis of Apple’s News Articles and Twitter Tweets from 1st Jan 2020 to 9th Jan 2022
Figure A.2: Adj Close price trend line from 2020 to 2021

Figure A.3: Frequent of News articles sources
Figure A.4: Frequent of Twitter Tweets in a day of 24 hours for 2020 and 2021

Figure A.5: Frequent of Twitter Tweets in a week for 2020 and 2021
APPENDIX A. ADDITIONAL CONTENT

Figure A.6: Frequent words in News articles

(a) Google  
(b) Apple  
(c) Facebook  
(d) Amazon

Figure A.7: Frequent words in Twitter Tweets

(a) Google  
(b) Apple  
(c) Facebook  
(d) Amazon