Intelligent Prediction of Stock Market Using Text and Data Mining Techniques

By

Mohammad Raahemi

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Science in Electronic Business Technologies

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

uOttawa

University of Ottawa

Abstract

The stock market undergoes many fluctuations on a daily basis. These changes can be challenging to anticipate. Understanding such volatility are beneficial to investors as it empowers them to make inform decisions to avoid losses and invest when opportunities are predicted to earn funds. The objective of this research is to use text mining and data mining techniques to discover the relationship between news articles and stock prices fluctuations. There are a variety of sources for news articles, including Bloomberg, Google Finance, Yahoo Finance, Factiva, Thompson Routers, and Twitter. In our research, we use Factive and Intrinio news databases. These databases provide daily analytical articles about the general stock market, as well as daily changes in stock prices. The focus of this research is on understanding the news articles which influence stock prices. We believe that different types of stocks in the market behave differently, and news articles could provide indications on different stock price movements. The goal of this research is to create a framework that uses text mining and data mining algorithms to correlate different types of news articles with stock fluctuations to predict whether to "Buy", "Sell", or "Hold" a specific stock. We train Doc2Vec models on 1GB of financial news from Factiva to convert news articles into vectors of 100 dimensions. After preprocessing the data, including labeling and balancing the data, we build five predictive models, namely Neural Networks, SVM, Decision Tree, KNN, and Random Forest to predict stock movements (Buy, Sell, or Hold). We evaluate the performances of the predictive models in terms of accuracy and area under the ROC. We conclude that SVM provides the best performance among the five models to predict the stock movement.

To my mom, my first and best friend in my life and my father, my forever hero

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Liam Peyton for his invaluable guidance and supports, and sharing his insights not only for my thesis but for these two wonderful years of my study in University of Ottawa.

I would also like to extend my deepest appreciation to Professor Bijan Raahemi of the University of Ottawa, for his valuable guidance and supports. This dissertation would not have been completed successfully without his continuous support and guidance throughout my thesis work. I would like to express my deepest gratitude to the committee members, Professor Nancy Samaan and Professor Hussein Al Osman for their invaluable comments and suggestions which have greatly improved my dissertation.

I thank all my colleagues at Knowledge Discovery and Data mining Laboratory of University of Ottawa for their continual support and encouragement.

Finally, I thank my mother and my sister and her beautiful daughter for their support, patience and encouragement throughout my research years.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
AMH	Adaptive Market Hypothesis
API	Application Program Interface
APPL	Apple Inc.
AUC	Area Under Curve
C-BOW	Continuous Bag of Words
CNN	Convolutional Neural Networks
CSV	comma-separated values
DJIA	Dow Jones Industrial Average
DNN	Dynamic Neural Network
DT	Decision Tree
EMH	Efficient Market Hypothesis
IIR	Infinite Impulse Response
KNN	K-Nearest Neighbor
LSTM	Long–Short Term Memory
MA	Moving Average
ML	Machine Learning
NLTK	Natural Language Tool Kit
NN	Neural Network
OLS	Ordinary Least Square
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
PV-DBOW	Paragraph Vectors Distributed Bag of Words Model
PV-DM	Paragraph Vectors Distributed Memory Model (PV-DM)
RF	Random Forest
RNN	Recurrent Neural Network

ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
UKF	Unscented Kalman Filter

CHAPTER 1
INTRODUCTION

News articles written on business activities serve the purpose of spreading information about the companies. When trading in the stock market, this information influence people either consciously or unconsciously in their decision process. Annual and quarterly earnings, dividend announcements, acquisitions, mergers, tender offers, stock splits, and major management changes, and any substantive items of unusual or non-recurrent nature are examples of news items that are useful for traders in making their trading decisions. These types of news are usually published immediately as breaking news and are often given to the press directly from the companies. News articles with other kinds of information (e.g. political news) about companies are also important for traders, but they do not necessarily originate from the companies themselves. With the immense growth of the internet in the last decade, the amount of published news articles has experienced a similar rate of growth. This has increased the amount of both useful and not so useful information about each company.

Information published in news articles influence, in varying degrees, the decisions of the stock traders, especially if the given information is unexpected. It is important to analyze this information as fast as possible so it can be used as a foundation for trading decisions by traders before the market has had time to adjust itself to the new information. This is a humongous task if done manually because of the immense amount of information and the speed of which new information is published. This means that an automatic system for analyzing news articles is potentially needed.

Due to the rapid expansion of the internet, as mentioned above, there has been a huge growth of easily available textual information over the last decade in the form of documents, news, blogs, forums, emails, and etc. This increased amount of available textual information has led to a research field devoted to knowledge discovery in unstructured data (textual data) known as text mining (Konchady, 2006). Text mining originates from the related field of data mining, which mines patterns from structured data instead of unstructured. It is also related to other fields like information retrieval, web mining, statistics, computational linguistics and natural language processing. One important application of text mining is text sentiment analysis, also referred to as opinion mining. This technique tries to discover the sentiment of a written text. This can be used to categorize text documents into a set of predefined sentiment categories (e.g. positive or negative sentiment categories), or it can be used to give the text a grade on a given scale (e.g. giving a text about a movie review a score on a grade from one to ten). Text analysis seems like a logical place to start when applying text mining to analyze news articles. This is because some news articles should have a higher probability of positively influencing the stock price, while the opposite is true for other articles.

To have more concrete definition: Business / financial news stories can contain information (words, sequence of words or overall story opinion) which has an economic value which is not reflected in the current price of a specific share or market index. Therefore, business/ financial news can be used as a basis of a successful trading strategy. In summary, the underlying objective of this research is that it is possible to improve an existing trading strategy by adding information from a financial news text mining system.

1.1 Objectives

The objectives of this research are derived by two research questions:

RQ1: Can we extract features- in the form of Doc2Vec numerical vector- of the news texts from reliable sources to be used in building predictive models to forecast when to buy, sell, or hold of a specific stock?

RQ2: If yes, then can we train various machine learning predictive models based on numerical vector representation of news text to determine which ones perform the best in terms of accuracy and area under ROC curve? (E.g. Decision Tree, Random Forest, Neural Networks, KNN, SVM)

Accordingly, the objectives of this study are to extract features of the text news related to a specific stock, label them using the reliable data from Dow Jones and other sources, then build machine learning models to make decisions on trading of that specific stock (whether to buy, sell, or hold)

We will then evaluate the performances of the predictive models in terms of accuracy and area under ROC to choose the best possible model.

1.2 Methodology

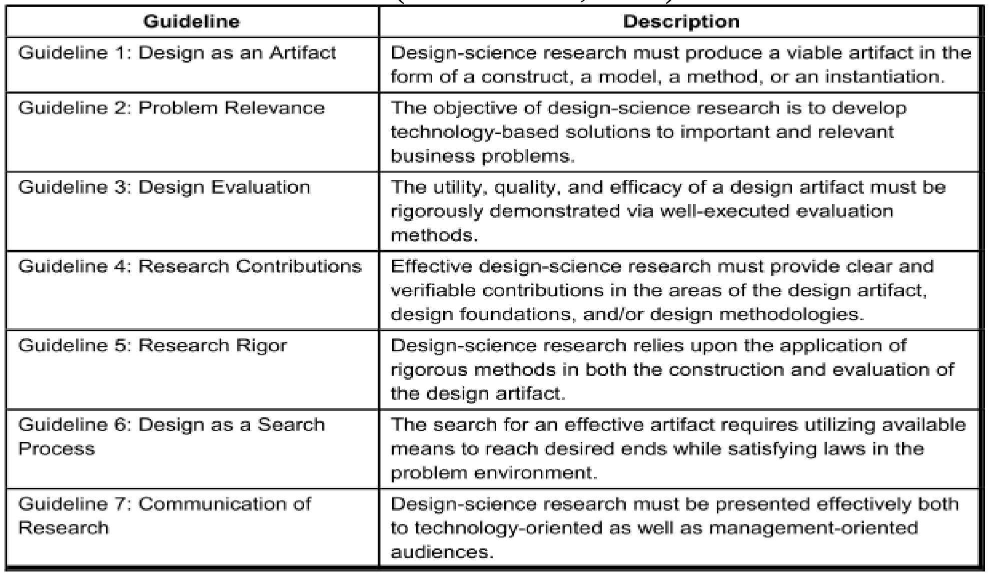
The Design Science Research Methodology (DSRM) for Information System (IS) incorporates principles, practices, and procedures required to carry out Information System researches and meets objectives (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). Design research is inherently a problem solving process that creates and evaluates IT artifacts intended to meet business needs, and solve the identified organizational problems (Hevner, March, Park, & Ram, 2004).

The field is still evolving after almost two decades of being adopted by Design Science Research searchers. The guidelines for the DS Research were presented by Hevner et al (2004), to assist researchers understand Design Science Research in IS, and determine when, where, and how to apply the guidelines in a research as illustrated in Table 1.

A formalized DS Research process (DSRP) model is presented by Peffers et al (2007), which consists of six activities in a normal sequence as shown in Figure 2. The first activity of DSRP is problem identification and motivation where specific research problem is defined, and the value of the potential solution is justified. This activity requires knowledge about the state of the problem, and the importance of the solution. The second activity is to define the objectives of the solution. This is where the objectives should be inferred rationally from the problem definition and specifications.

TABLE 1. DESIGN SCIENCE RESEARCH GUIDELINES

(Hevner et al., 2004)



Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

The third activity is the design and development of IT artifactual solution. This includes identifying artifact’s functionality, its architecture and then developing the actual artifact. In this phase we followed CRISP DM Data Mining Methodology which includes five main steps: data collection, data preprocessing, building models, evaluating models, and deployment (prediction).

Following the CRISP DM methodology, we conducted the following stages in this research, as shown in Figure 1:

Stage 1: Data Collection

Financial news and headlines are retrieved from Intrinio news database, and a list of features is selected, and Daily stock prices are collected separately on a daily basis. In this research, first we utilize a 4-year financial news corpus comprising over 1,000 articles collected from Intrinio website for Apple Inc. stock in Dow Jones Index (DJI) to train a directional stock price prediction system based on news headlines.

Next, we collect over 750,000 news corpuses that mention at least one of the 30 DJI stock symbols in Technology stream. We utilize this collected news to train our Word2Vec and Doc2Vec word embedding models to map our news headlines to the vectors of real numbers and make it readable for our machine learning models.

Then we proceed to test our hypothesis to determine if information in articles lead to a statistically significant boost in the daily directional prediction accuracies for the prices of APPL stocks.

Stage 2: Preprocessing

A Word2Vec and a Doc2Vec Models are trained using 1 GB Financial news collected from the Factiva database. News texts are tokenized, stop words removed, lemmatized, converted into lowercase, and then converted to vectors with 100 dimensions using the trained Doc2Vec. The process of collecting and labeling news headlines and stock data is shown. Text mining and natural language processing conducted with tokenizing, removing stop words, lemmatizing, and applying using text mining and machine learning on news headlines to predict the stock market lowercase text to all news headlines prior to their semantic analysis. The Doc2Vec is used for the semantic analysis of headlines, valuing each word's context as well as transforming each headline into a vector of 100 elements to complete the preprocessing prior to inputting within the machine learning models.

Daily stock prices are used to label each news vectors into 3 groups of Buy, Sell, and Hold. Since the data is imbalanced (90 % hold), we employed SMOTE to balance data within the 3 groups.

Stage 3: Modeling

The preprocessed labeled data is used to build 5 predictive models: K Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), Neural Networks (NN), and Support Vector Machine (SVM).

Grid searches are used for each machine learning model chosen including, k-nearest neighbour, decision tree, and random forest. Optimal parameters are discovered from the grid search and are used to run each model in order to analyze their results.

Stage 4: Evaluation

A 10-fold cross validation is performed. Models are evaluated based on their accuracy, as well as areas under the ROC curves Models are compared using T-test on 10 folds data to select the best one. T-tests are conducted to discover if statistical significance is present between the model classifiers. From here, the ROC curves, accuracy, as well as micro averages of each classifier are evaluated to determine the optimal path to predicting the Apple stock from relevant news headlines.

Stage 5: Iteration Cycle

Feedback according to the results obtained from the “Evaluation” and “Deployment” stages, some parameters and thresholds need to be adjusted, and the models should be retrained to improve their performances. According to the results, feedback obtained from the “Evaluation” and “Deployment” stages.

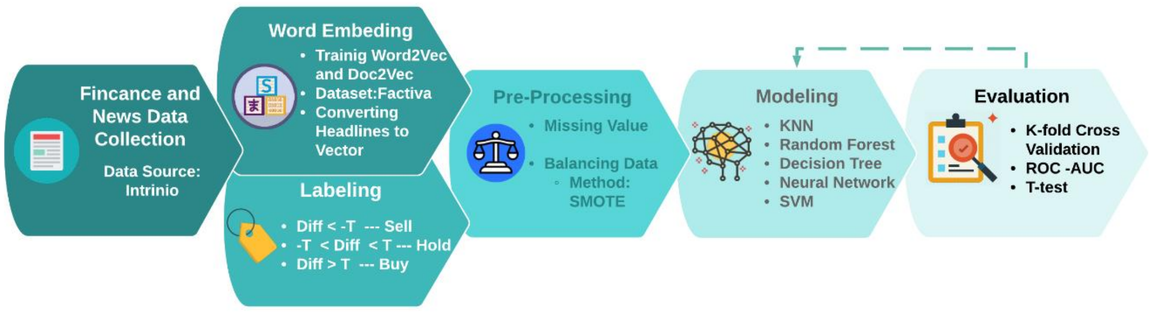
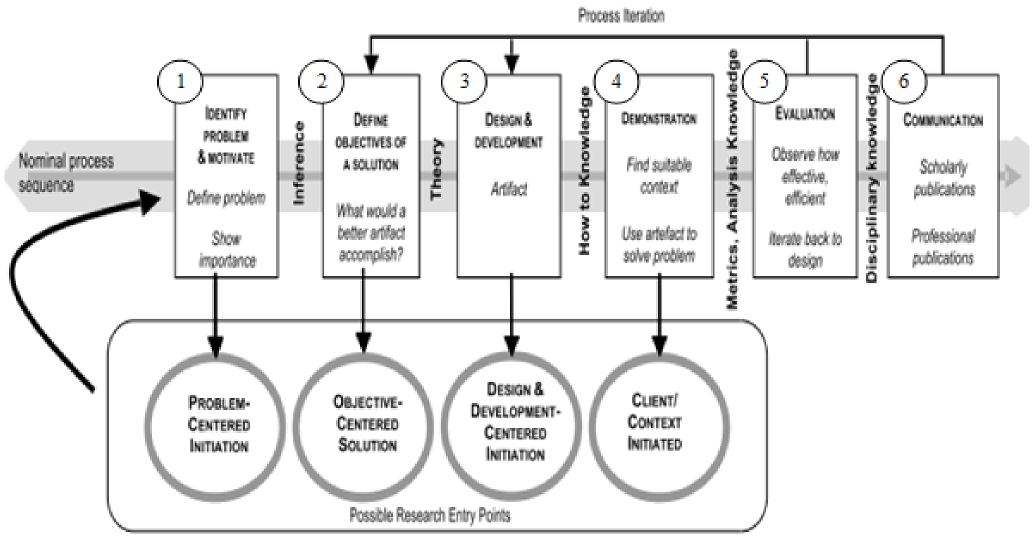


FIGURE 1. IMPLEMENTATION OF THE THIRD ACTIVITY OF THE DSR METHODOLOGY (DEVELOPING ARTIFACT) USING CRISP METHODOLOGY

The fourth activity is to demonstrate the efficacy of the developed artifact in solving one or more instances of the research problem. This may involve its use of experimentation, proof, simulation, case study or any other appropriate activity. The fifth activity is to evaluate and observe how the developed artifact will support a solution to the research problem. This activity includes comparing the objectives of the solution to the actual results from using the artifact in the demonstration activity. The sixth activity is to communicate the problem, the artifacts, its importance, its utility, and its effectiveness to researchers and other audiences.

According to Peffers et al (2007), the DSRP is structured in a nominally sequential order, but researchers can actually start at any step, and move to the next. The basis of the nominal sequence is the problem-centered approach, which starts with activity number one. If the idea for the research resulted from observation of a certain problem or from suggested future research work in a paper or a prior research, researches can proceed in the sequence of the problem-centered approach. The nature of the research can help researchers identify their DS Research possible entry points.



Process Iteration
Nominal process sequence
1
Identify problem & motivate
Define problem
Show importance
Inference
2
Define objectives of a solution
What would a better artifact accomplish?
Theory
3
Design & development
Artifact
How to Knowledge
4
Demonstration
Find suitable context
Use artefact to solve problem
Metrics, Analysis Knowledge
5
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
6
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development-Centered Initiation
Client/Context Initiated
Possible Research Entry Points

FIGURE 2. THE DS RESEARCH METHODOLOGY

(Peffers et al., 2007)

Although we implement this methodology using Factiva and Intrinio high-tech related datasets, and Apple Inc. as an example, the methodology itself is general and applicable to any other high-tech related stock market prediction.

1.3 Contributions of the Thesis

To achieve the objectives of this research, we employed the methods outlined below. After ensuring proper understanding of the domain and performing a literature review to study the backgrounds and understand the field (conducting a comprehensive literature review on predicting stock market movements using reliable sources of textual financial news), we designed the architecture of the intelligent market prediction using text and data mining techniques.

Then, we proceeded with data collection by defining the features and format of the data. To collect the data, we used Factiva and Intrinio databases to retrieve historical prices and news articles. The reason for choosing these databases is that they provide real and reliable data, which is crucial for the research. The news vectors were then labeled into 3 classes of “buy”, “Hold”, and “Sell”. To label the news data, we used the closing prices of the stock for two consecutive days. Such labeled data was then balanced into 3 groups of almost equal sizes and prepared for the next phase to build predictive models. (Collected and pre-processed data from Factiva and Intrino databases)

The news data was pre-processed, tokenized, lemmatized, and converted into a vector of 100 dimension using Doc2Vec embedding technique. (Trained Word2Vc and Doc2Vec models on 1GB of financial news from Factiva)

Then, the data were split into training, testing, and validation datasets. Training data is used to train the predictive models. Testing is used to verify the accuracy of the models. Finally, validation data, is used as unseen data to measure the accuracy of the models and validate them. We built and trained five different machine learning models. (SVM, Neural Network, Random Forest, KNN, and Decision Tree)

At the evaluation phase, we conducted performance analysis (accuracy, area under ROC), as well as combative study of the models (using T-test) to identify the best predictive models.

1.4 Thesis Organization

This thesis consists of five chapters. Chapter 2 presents a background and reviews some related researches in the field. Chapter 3 describes data collection and data preprocessing steps taken in this thesis. Chapter 4 presents the five predictive models built on the labeled data together with analysis of the experimental results. Chapter 5 includes a summary of the research, contributions, as well as recommended future research directions. Appendix-A includes the link to the Github page where the Python Codes implemented for this research are shared.

CHAPTER 2
LITERATURE REVIEW AND BACKGROUND STUDY

Companies, after establishing their initial requirements, need to raise capital through a joint venture to expand their business. For companies sharing their stock is a way to access new financial resources. For shareholders, a share represents fractional ownership of a company, which allows investors to share the company's profits and properties.

Although developed countries depend on the stock market for their growth, and individuals pursue their financial goals through the stock market, such means is a double-edged sword. People may gain profit through proper investments in stock markets or may lose their entire life savings by wrong investments. Stock market prediction is the act of trying to determine the future value of a company’s stock. The successful prediction of a stock’s future price could yield a significant profit.

One of the primary debates among economists is stock market behavior. Random Walk theory (Bollen, Mao, & Zeng, 2011) proclaims that past movement or trend of a stock price or market cannot be used to predict its future. For instance, growth and inflation are the two main factors that create uncertainty in stock prediction. This theory was introduced by Eugene Fama (1970) and is well known as the Efficient Market Hypothesis (EMH) (Fama, 1970). For many years EMH has been the dominant theory; however, Adaptive Market Hypothesis (AMH), followed by Lo (2005), combines EMH principles with psychology-based theories. AMH states that people are mainly rational, but sometimes can overreact during periods of market volatility. AMH asserts that people are stimulated by their self-interests, make errors, and tend to adapt and learn from their mistakes. (Lo, 2005)

The stock market prediction includes multiple disciplines, such as computer science, statistics, economics, and finance. For the last decades, researchers of these fields are investigating which of these declines can make the stock market better predictable. Yet, this is one of the most popular and challenging topics in these research areas. (Patel, Quadros, Patil, Pawale, & Saxena, 2017)

2.1. Fundamental versus Technical Analysis

The nonlinear and nonstationary features of the stock market make it a complicated system (Bisoi & Dash, 2014). As observed by Ticknor (2013), the complexity of the stock market is related to a significant number of elements such as political events, market news, quarterly earnings reports, international influence, and conflicting trading behavior. Two major schools model the stock market prices, the schools of *Technical* and *Fundamental* approaches. (Gerencser et al., 2009) The former posses the price of a share of the company is based on the interaction of demand and supply forces only. *Technical analysis* has been taking into account Mathematic-statistical modeling as a set of tools that allow for predicting future returns in financial assets by concentrating past market data, mostly stock price and volume.

As a different approach, the *Fundamental analysis* is a method of measuring a security's intrinsic value by factors that influence the interest of the economy, industry, and company. In general, fundamental analysis has the essence of Graham's work as its cornerstone. In his well-known book, Intelligent Investor, Graham recommends buying stocks based on a set of stock-selection rules. Several authors, such as Oppenheimer &

Schlarbaum (1981) and Metghalchi, Chang, & Marcucci (2008) tested these rules and applied them and concluded that it is possible to achieve returns higher than those obtained by the buy-and-hold strategy. This policy is a conservative investment strategy in which an investor purchases stocks and retains them for an extended period regardless of market fluctuations, considering even the semi-strong form of the efficient market hypothesis. Nevertheless, the fundamental data is available in the unstructured text, such as social media, news, blogs, and forums. This is the most challenging aspect of the research.

2.2. Technical Analysis using Machine Learning Techniques

Most of the traditional stock prediction approaches characterize the stock market by historical trading prices. This technique is called Technical Analysis. They tried to find signs that indicate rising or dropping prices based on this historical time series. (Li, Jin, Xi, Liu, & Luo, 2019) Recent studies have mixed traditional technical analysis with intelligent system techniques and mathematical models. Some of these ML technologies are neural networks, fuzzy systems, evolutionary computation, and genetic algorithms. Borovkova & Tsiamas (2019) proposed an ensemble of long–short-term memory (LSTM) neural networks for intraday stock predictions, using a large variety of technical analysis indicators as network inputs. Pongsena, Ditsayabut, Kerdprasop, & Kerdprasop, (2020) merged time-series data analytics to an image processing-based approach to do forecast the Forex or Foreign Exchange financial trend. Bisoi & Dash, (2014) present a simple Infinite impulse response (IIR) filter based Dynamic Neural Network (DNN) and an innovative, optimized adaptive unscented the Kalman filter for estimating stock price indices of four different Indian stocks. Dynamic neural information system weights are modified by four different learning techniques, including gradient calculation, Unscented Kalman Filter (UKF), differential evolution (DE), and hybrid technique (DEUKF) by alternately executing DE and UKF for several generations. After predicting stock price indices one day to one week ahead of time horizon, the stock market trend was evaluated using several significant technological indicators, such as the moving average (MA), stochastic oscillators. (Bisoi & Dash, 2014)

Wei, Chen, & Ho (2011), in their paper, have proposed a model based on an adaptive-network-based fuzzy inference system which uses multi-technical indicators to predict stock price trends. Three sophisticated processes have been proposed in their hybrid forecasting model: select key technical indicators from the standard correlation matrix; use the frequency modulation clustering approach to divide the technical indicator value into linguistic values based on the data discretion approach in the second step; Finally, a fuzzy inference method (FIS) is used to extract the rules of linguistic terms from the data set of technical indicators and to refine the FIS parameters based on an adaptive network for the development of forecasts. Six years of the Taiwan Capitalization Weighted Stock Index (TAIEX) was used as an experimental database to test the proposed performance predictor model. (Wei et al., 2011)

A Bayesian regularized artificial neural network was intended as a novel approach to forecast financial market behavior by Ticknor (2013). Another technique to predict financial market behavior is the application of Bayesian regularized artificial neural network proposed by Ticknor. In this technique, daily market rates and financial, technical indicators are utilized as inputs to predict the one-day future closing price of individual stocks. (Ticknor, 2013)

Using past price and volume data, with the help of machine learning techniques can be implemented along with technical trading rules to shape a trading framework that can forecast the future direction of security prices. (Gorgulho, Neves, & Horta, 2011)

Machine learning techniques can use past price and volume data to build a framework for forecasting future movements (directions) within the trading rules.

2.3 Fundamental Analysis using Machine Learning and Text Mining Techniques

Predicted stock prices based on mere technical data has proven to be insufficient. For instance, recent studies have shown that there is a strong correlation between news articles related to a company and its stock price movements. In these studies, it is evident that financial news has a significant influence on the stock market. Therefore, the financial market prediction based on news disclosures is attracting more and more attention in recent years. (Li et al., 2019) However, it is complicated to predict the financial market accurately based on news disclosures due to the complexity and ambiguity of natural languages used (Manning, Schütze, & Weikurn, 2002) Therefore, there is a massive amount of textual data that can be used as a new source of information on this task (Merello, Picasso Ratto, Ma, Oneto, & Cambria, 2019)

Various data sources can be used to predict market behavior. Some studies have used news articles(Meyer, Bikdash, & Dai, 2017; Vargas, De Lima, & Evsukoff, 2017; Yang, B, Raza, Herbert, & Kang, 2018), social media(Li et al., 2019), message boards (Gálvez & Gravano, 2017; Long, Song, & Tian, 2019) and company's ad-hoc announcements (Kraus & Feuerriegel, 2017; Pröllochs, Feuerriegel, & Neumann, 2016) to predict the stock market and others have used hybrid methods which consider both historical market prices and external sources of text data for prediction (Merello et al., 2019; Mohan, Mullapudi, Sammeta, Vijayvergia, & Anastasiu, 2019; Z. Wang, Ho, & Lin, 2019). The data source in this approach has a critical role and profoundly impacts prediction performance. However, techniques that are applying to extract the information from this data is also essential.

2.4 Review of the Prior Arts on using Machine Learning and Text Mining for Stock Prediction

In this subsection, we present the popular machine learning models adopted in the reviewed studies. A summary of these algorithms, along with the details of the methods, are provided. Depending on the characteristics of the stock market datasets and situations, each algorithm has its strengths and weaknesses. This implies that among learning-based prediction methods, no one algorithm is always better than others, and methods will perform well when the problem is identified and optimized successfully. The Support Vector Machine (SVM) is widely used across the articles. Note that some studies applied more than one machine learning technique. The other most used algorithms are Naive Bayes, Long Short-term Memory (LSTM), Neural Networks, Random Forest (RF), Regression models, K Nearest Neighbour, Decision Tree (DT), and Reinforcement Le arning respectively.

Nearly all the machine learning algorithms that have been used are classification algorithms. Basically, the systems are using input data to learn and classify the output. The goal of the models usually is to categorize the outputs. They categorize the data in terms of the financial market movements in classes such as Up, Down, and Steady. However, there is also a group of works that use regression analysis for making predictions classification. Regression models are especially suitable for impact analysis. Sometimes, a primary linear regression is directly used with Ordinary Least Square (OLS) to estimate coefficients (Khadjeh Nassirtoussi, Aghabozorgi, Ying Wah, & Ngo, 2014). After a brief overview of all phases, we will review some of the relevant articles in detail below.

2.4.1 Using Machine Learning and Text Mining on the News for Stock Prediction

Wang et al. (2019) proposed an enhanced learning-based approach toward stock price prediction by using Neural Network (NN) as an initial learning-based process and applying it to evaluate stock price time series data. In the data collection phase, two types of datasets were used. To collect the financial market value dataset, daily DJIA data was downloaded from *Yahoo! Finance*. The time interval taken was from 1/1/2007 to 31/12/2016 - ten years in total. And for news articles dataset collection, New York Times news articles with the help of a python API. The articles were split monthly. The articles were time sensitive, hence, were divided monthly. 80% of the data was used for training purposes, and 20% of the data was used as testing data.

In order to preprocess the news articles, a filter was applied to articles in the section of Business', 'National', 'World', 'U.S.', 'Politics', 'Opinion', 'Tech', 'Science', 'Health' and 'Foreign', which caused 400,000 articles to remain in the dataset for the analysis.

After filtering and preprocessing the news articles dataset, sentiment analysis is applied. The Natural Language Toolkit Package (NLTK) was used to perform sentiment analysis with the help of another open-source tool named Vader Sentiment Analyzer. To calculate the sentiment scores, SentiMo software (through voting methods) was used. The scores obtained on a daily basis contain four parts: mixed score, negative score, neutral score, and positive score.

In the end, Stock market values merged to news article sentiment analysis to be given to the NN system. This system trained in four different methods: Bayesian regularization backpropagation (trainbr), Levenberg-Marquardt backpropagation (trainlm), Conjugate gradient backpropagation with Powell-Beale restarts (traincgb), and Gradient descent with adaptive learning rate backpropagation (traingda).

To observe lasting impactfulness of the news on the stock price, a window method was used. Authors claimed the best windows size is six; however, they were not able to explain when the window size is set larger than six days, such as ten days, why the MSE will be increased. In their case study, MSE of a ten days observation is almost doubled in comparison with six days of observation.

The results of this research work demonstrated that the news sentiments relevant to the stock market could be used to improve the performance of the learning-based predictor. This work also suggests that an improved output of the NN method can be achieved when the appropriate size of the window is chosen. The efficiency of the learning-based method in this study is further improved when the effects of the news sentiments are considered. (Z. Wang et al., 2019)

In their work, Kraus & Feuerriegel (2017), instead of financial news, targeted Company disclosures specifically to propose a decision support system to predict expected price changes in stock markets subsequent to the announcements. As an alternative to the traditional bag-of-words approach, as a simplifying representative, this paper used sequence modeling based on deep neural networks to improve the predictive power of their decision support system.

Their corpus comprised of 13,135 regulated German ad hoc announcements in English since they have shown a strong influence on financial markets. In order to reduce their noise from the dataset, they omitted the disclosures of the Penny Stock Company, which yields a 10,895 observations survey labeled as Positive or Negative. To prepare the company disclosure, they performed tokenization and stemming techniques, then preprocessed content into numerical feature vectors by utilizing the TF-IDF approach, which puts more robust weights on specific terms.

As their Deep learning architectures, they introduced the recurrent neural network (RNN), followed by its extension, the LSTM, which could better memorize the information. Here the input vector xi consists of the words (or stems) in a single hot encoding. Mathematically, this method specifies a vector composed of zeros, except for a single element with a one that refers to the i-th word in the sequence. This produces high-dimensional but sparse vectors as input. To improve their prediction accuracy, they tuned the LSTM hyper parameters. This was used to find the best-performing parameters based on time-series cross-validation that employs a rolling forecasting origin, which caused 57.8% prediction accuracy at the end.

To tune all parameters of the basic method, a grid search was performed with a 10-fold time-series cross-validation on the training set. Their results show that long short-term memory models (LSTM) can outperform all traditional machine learning models based on the bag-of-words approach, especially when they further pre-train word embeddings with transfer learning. (Kraus & Feuerriegel, 2017)

To improve Kraus and Feuerriegel's results, Chiong et al. (2018) recommended a sentimental analysis-based approach for financial market prediction. They used news disclosures, including extracting sentiment-related features from financial news and the historical stock market as an input. They employed a support vector machine to predict the stock market trend and optimized the output with the help of Particle Swarm Optimization (PSO). The novelty of this work was improving the efficiency of models in comparison to deep learning by applying transferred learning models via high dimensional input features.

The dataset used in this study was precisely the same as Kraus and Feuerriegel's research, including 13,135 regulated German ad hoc announcements in English. To preprocess the data, they removed acknowledgments of stock, which was published on “non-trading” days manually, that remain as a gap in their research since they did not mention which days consider as non-trading. After this stage, a semantic analysis was applied by using TextBlob python library. This caused two features - polarity and subjectivity – of disclosures. They classified each disclosure as negative or positive based on polarity and subjectivity criteria. The contrasted research achieved 57.8% prediction accuracy with the same input by Kraus & Feuerriegel, (2017).

In the prediction stage, an SVM model was implemented. The kernel function for the SVM is a Gaussian, determined by trial-and-error. Three hyper-parameters of this function, the penalty parameter of the error term, the bandwidth of Gaussian kernel, and the epsilon-tube, were optimized through PSO. The swarm size and maximum iteration of PSO were set to 5 and 100, respectively.

After implementing and optimizing their framework, the accuracy of 59.15% obtained by their SVM and PSO-based prediction model replaced the previously proposed models. However, they could not determine whether variations between the findings were statistically significant. (Chiong et al., 2018)

In contrast with the recent researchers mentioned above, Schumaker and Chen (2009) used their study to predict a discreet stock price twenty minutes after a news article had been published.

In the word embedding phase, as they assumed that integrating more precise textual representations with past stock pricing data would result in enhanced predictability, this paper experiments with the use of various linguistic text representations, including Bag of Words, Noun Phrases which is accomplished through the use of a syntax where parts of speech are identified through the aid of a lexicon and aggregated using syntactic rules and Named Entities approaches.

In their methodology design, each financial news article is represented using three predefined textual analysis techniques. Such a representation defines and stores the essential words of the article in the database. The Entity extractor portion goes one step further by assigning hybrid semantic/syntactic tags to document terms and phrases in one of the seven predefined categories of date, location, money, organization, percentage, person, and time. These entities are then identified through the usage of a lexicon.

Stock quotes are gathered on a per-minute basis for each stock. Upon publication of a news article, they predict what the actual stock price will be 20 minutes after the publication of the item. To do so, they performed a linear regression on the quotation data using an arbitrary 60 minutes before article release and extrapolate what the stock price should be in 20 minutes.

To test the types of information that need to be included, they developed four different models and gave varied the data to the models. The first model, Regress, was a simple linear regression estimate of the 20+ minute stock price. Assuming that breaking financial news articles have no impact on the movement of stock prices, they expect a consistent performance from this model. The other three models used supervised learning of SVM regression to compute their -20+ minute predictions. Model M1, used only extracted article terms for its prediction. While no baseline stock price exists within this model. They chose it because of its frequent usage in prior studies on the directional classification of stock prices. Model M2, used extracted article terms and the stock price at the time the article was released. They feel that a Named Entity representation would generate better performance because of its ability to abstract the article terms and discard the noise of terms picked up by both Bags of Words and Noun Phrases. Model M3, used extracted terms and a regressed estimate of the minute+ stock price. This model led to better predictive results.

As a result, by using a specially adapted support vector machine (SVM) derivative specially tailored for discrete numeric prediction and models containing different stock-specific variables they showed that the model containing both the terms of the article and the stock price at the time of article publication had the best output in closeness proximity to the real stock price (MSE 0.04261), the same direction of price movement as the future price (57.1% directional accuracy) and the highest return using a simulated trading engine (2.06% return). (Schumaker & Chen, 2009)

Ding, Zhang, Liu, & Duan (2015) argued that because research by Schumaker and Chen (2009) is unable to capture established relationships, their ability restricts. For example, representing the event {“Microsoft sues Barnes & Noble.” using term-level features “Microsoft”, “sues”, “Barnes”, “Noble”} alone, can be difficult to accurately predict the price movements of Microsoft Inc. and Barnes & Noble Inc., respectively, as the unstructured terms cannot differentiate the accuser (“Microsoft”) and defendant (“Barnes & Noble”).

Thus, using Open Knowledge Extraction (Open IE) to obtain organized representations of events, they found that the participants and object of events could be captured better. For example, a structured representation of the event above can be (Actor = Microsoft, Action = sues, Object = Barnes & Noble). This led them to design an event embedding instead of word embedding. They learned event embedding using a Novel neural Tensor Network (NTN), which can learn the semantic compositionality over event arguments by combining them instead of only implicitly, as with standard neural networks. The input of the neural tensor network is word embedding, and the output is event embedding. By using the skip-gram algorithm used in the Word2Vec word embedding model captured from Reuters financial news and Bloomberg financial news, they trained the initial word representation of d-dimensions (d = 100) from the massive financial news corpus. Then passed it to the neural tensor network to extract the event embedding. They modeled long-term events as past month events, mid-term events as past week events, and short-term events as stock price change events on the past day. The predictive model learned the impact of these three different time ranges on stock prices based on a CNN framework. The input to the model is an embedded sequence of events, where events are ordered in chronological order. Embedding of the events on each day is averaged as a single input unit (U). The output of the model is in a binary class, where Class +1 represents that the stock price will increase, and Class -1 represents that the stock price will decrease.

To experiment with their model, they collected financial news from Reuters and Bloomberg over the period from October 2006 to November 2013. They derived events from news titles only and performed experiments to forecast the 500 stock index of Standard & Poor (S&P 500) and individual stocks, and collected indices and prices from Yahoo Finance.

To make detailed analysis, they construct the following five models: WB-NN: word embedding input and standard neural network prediction model, WB-CNN: word embedding input and convolutional neural network prediction model, E-CNN: structured events tuple input and convolutional neural network prediction model (this paper), EB-NN: event embedding input (this paper) and standard neural network prediction model, EB-CNN: event embedding input and convolutional neural network prediction model. They used Accuracy and MCC parameters to measure their models' results. This showed EB-CNN is working better than others (Accuracy: 65.08%, MCC: 0.4357) (Ding et al., 2015)

In Merello et al., (2019) work, two possible causes of the stock price movements, are evaluated. First, they related the stock price trend with the single news. They considered a collection of the recently published news as input. Their proposed model is supposed to have the ability to select the most critical items in the collection and to track the flow of the extracted information. Secondly, they investigated the aggregation of multiple news as input. A unique vector is used to represent a set of news recently published (and a collection of them) is fed to the model. As before, the model was required to select essential representations and track their evolution through time. Their model had three stages. The first stage was needed to extract the most informative element in the window. The second stage modeled the evolution of the information over different time steps. Finally, the third stage was needed to make the prediction with the LSTM algorithm.

According to the results of this paper, the information in news articles could be used as an accurate predictor of past stock price movements. However, the forecast performance decreases rapidly. They have investigated two different agents that can possibly drive stock market movements: aggregate news and new information. In the results, in both cases, they have shown how even if the proposed models achieve good performances in predicting the past price, and when they increase the size of the window they are not able to perform significantly better than random guessing. (Merello et al., 2019)

2.4.2 Using Machine Learning and Text Mining on the Social Media for Stock Prediction

As we stated at the beginning of this chapter, the news is not the only fundamental database to predict the stock. A survey by the Reuters Institute (2016) finds that 51% of respondents use social media to access news every week, and 12% cited it as their main source of news. Moreover, a significant share of social media content relates to stock markets. This is observed by an emerging company that extracts and sells market-relevant indicators from social media. Bogle & Potter (2019) investigate the sentiments expressed on Twitter and their predictive impact on the Jamaica Stock Exchange in the short term. In this study, the qualitative data collection involves collecting data from the social network of Twitter using a customized framework built in the programming language of the open-source R and the programming interface of the Twitter application (API). Of note, Twitter's data was relatively readily accessible via a link to its API barring its API restrictions. Tweets were cleaned in order to determine the sentiments being expressed about marijuana and its legalization. The qualitative data pre-processing involved the removal of duplicate tweets, numbers, punctuation, and symbols. A pre-processing function developed in R allowed for the initial filtering of unwanted or unnecessary verbiage from the tweets while being extracted from Twitter. It also included swapping certain emoticons with words (e.g. :-) with "happy"). This population reflected a collection of tweets that conveyed polarities about potential positive, negative, or neutral sentiments about medical marijuana prohibition.

Further pre-processing of the tweets was performed to extract values that the R tweet cleaner did not catch. This pre-processing phase involved translating the data string by applying a StringToWordVector function into word vectors. This operator transforms string attributes into a collection of attributes (depending on the tokenizer), which reflect word occurrence. It also sets parameters that were important to the retrieved information. These parameters include restrictions imposed on the number of terms repeated (term frequency); the number of words to output (word count); the tokenizer that delimits words within the string; and the stemmer that facilitates the conversion of terms to their base types, (for example, the base term love for the words like lovely, lovable, loving).

In order to assess tweet sentiment, several machine learning classifiers were tested to define the data mining classification model best suited for the question. Three classificators in machine learning were explored: the Naive Bayes Multinomial Text Classifier, the Support Vector Machine (SVM), and the J48 Decision Tree. After training the three classification models to properly classify the tweets into positive, negative, and neutral categories, they were explored to validate both their accuracy and performance. Naive Bayes Multinomial Text emerged from these classifiers as the best performing model and was applied to unknown instances of tweets derived from Twitter. As a consequence, the market's bullish behavior can be predicted by the twitter sentiments, but the bearish action of the market does not seem to suit the sentiments generated from those tweets. Although they can predict the flow of market by 87% and their correlation coefficiency is 0.99 for price prediction, the biggest challenge they face is to find out the proper sense of terms such as "ur," which substitutes people for "your" etc. (Bogle & Potter, 2019)

Most of the existing studies focus on one source of news at a time. The open question, therefore, is whether the stock markets react systematically differently to social media and news media. In their paper, Jiao & Walther (2016) directly compared social and news coverage and thus provide new stylized facts about their relationship with the stock markets. They used a panel dataset for media coverage from the Thomson Reuters MarketPsych Indices database (TRMI). Their data aggregated a wide range of news media sources and the most popular social media coverage indicators, which facilitate a like-for-like comparison between social and news media. These data are merged with stock prices, turnover, and stock-specific characteristics.

Their main result is that coverage in social and news media are associated with markedly different patterns of subsequent return volatility and trading volume per share (or turnover). A high social media buzz around a given stock predicts a statistically significant increase in idiosyncratic return volatility and trading activity over the following month. A high news media buzz predicts a significant decrease in volatility and trading activity. These empiric patterns are robust for the inclusion of stock and time-fixed effects, the time difference in stock characteristics, and the measure of asset value disagreement (dispersion in the opinions of financial analysts). Their results on volatility also apply at the market level: having a high social media buzz around the stock market as a whole predicts higher return volatility, while news media buzz predicts the opposite.

They further evaluated the mechanisms in effect, using the VAR stock-level stand, which includes news media buzz, social media buzz, uncertainty, and turnover as endogenous variables. In the end, they found that an increase in news media buzz predicts an increase in subsequent social media buzz in the form of Granger's causality. Conversely, the rise in social media does not indicate shifts in the ensuing news coverage. The key contribution of this paper was creating these robust stylized details. Together, they say that financial markets' social media and news coverage are connected in a number of ways. In fact, it seems that the news media is a crucial indicator of social media. This is consistent with the belief that social media content is created by repeating and sharing (e.g., re-tweeting) current news. Nevertheless, their observations do not have a causal significance (they do not detect exogenous differences in social or news media coverage). (Jiao & Walther, 2016)

2.4.3 Review of the Prior Arts on using hybrid Models for Stock Prediction

Lien Minh et al. (2018) suggested a novel method to forecast the course of stock prices by using both the financial news and the sentiment dictionary. The original contributions of their paper included a proposal for a novel two-stream recurring unit network and a sentiment word embedding developed on the financial news dataset and Harvard IV-4. The proposed approach consisted of four parts. First, in the document preprocessing phase, the extracted articles are preprocessed to eliminate unnecessary details such as stop words, punctuation. Second, the next step required two tasks: the marking of news articles on the basis of stock prices. In the third stage, in the Stock2Vec embedding segment, they developed a new sentiment word embedding model based on the financial dataset and sentiment dictionary. In the final stage, the TGRU network introduced on the financial news dataset, followed by multiple scenarios.

To preprocess the news articles, since news articles were crawled out of the internet and in HTML format, all unnecessary tags were dropped. After that, the goods were exported in plain text. Next, each sentence was tokenized into a series of tokens; each token was a single phrase. Finally, the following steps were taken: all stop words have been removed based on the English stop word dictionary. All punctuation and numbers have also been removed because they are unnecessary and sometimes occur, which affect the position of other meaningful words. Raw daily S&P 500 Index stock prices downloaded from Yahoo Finance was in CSV format to collect the daily stock price.

To label their documents they used open-to-close return (daytime return) instead of a close-to-close return (overnight return) since the records of the open-to-close are more similar to the total of the same stock, suggesting that the open-to-close return contributes more to the total return and at the end they labeled each document as Positive or Negative.

After the labeling step, a labeled dataset was generated; it was then used to train the proposed Stock2Vec model. The model was implemented using Python programming language. Each word was presented by a vector with a maximum length of 300, with a total number of words in the Stock2Vec being limited to 5,000.

In addition to the features derived from the news dataset, they used another set of features containing three metrics that are widely used in technical research. Technical indicators are quantitative measurements that are focused on the price, volume, or open interest of a security or contract. Technical analysts use metrics to forecast potential market changes by examining historical evidence. They apply them to evaluate whether the efficiency of the performance is enhanced. These three technical indicators included the Stochastic oscillator, William, and Relative Strength Index.

The stochastic oscillator is a measure of momentum that compares the closing price of a stock to its price range over a period of time. It can be used to foreshadow reversals when the predictor indicates bullish or bearish differences. William is a measure of the momentum indicator that allows investors to identify over-purchased and oversold conditions. It is based on a comparison of the current close to the highest high for a user-defined lookback time. Relative Strength Index is a measure of the momentum oscillator that evaluates the speed and trend of price movements. The Relative Strength Index (RSI) ranges between 0 and 100. In reality, investors typically sell RSIs with a value greater than 80 and buy RSIs with a value of less than 20.

The proposed two-stream Gated Recurrent Unit (TGRU) overall accuracy was 66.32 percent, which outperformed the output of the previous models, including GRU and LSTM. The proposed model had two learning states, including backward and forwards processing, so that it can learn more valuable information, especially on text processing issues. The sentimental embedding of Stock2Vec is then generated by using financial datasets and the Harvard IV-4 sentiment dictionary. Because it took into account the sentiment value of the word embedding of Stock2Vec, in experiments, it proved to be more successful than the original embedding process, such as Glove and Word2Vec. Also, financial indicators, which are one of the most significant variables in the financial analysis, are included. The model reveals its robustness in both the S&P 500 index and the individual market trend forecast. In addition, a simulation framework has been placed in place to measure the actual profits that investors make when they use our method. It has been shown to be resilient against market fluctuations and the ability to respond to real market risk. This mechanism can also be built into an automated system to assist investors in the trading of individual stocks.(Lien Minh et al., 2018)

Vargas et al. (2017) in their research, used deep learning models to predict frequent directional movements of stock prices using financial news titles and technical indicators as inputs. A contrast is made between two separate sets of technical indicators. Set 1: Stochastic (a momentum indicator comparing the closing price of a security to the range of its prices over a certain period), Momentum (the rate of acceleration of a security's price), Rate of Change (the speed at which a variable changes over a specific period); set 2: Exponential Moving Average Convergence-Divergence, Relative Strength Index, On Balance Volume and Bollinger Bands. Deep learning methods can detect and analyze complex patterns and data interactions. Experiments showed that the Convolutional Neural Network (CNN) can be better than the Recurrent Neural Networks (RNN) to capture text semantics. Also, RNN is better at capturing contextual information and modeling complex temporal characteristics for stock market forecasting. There are. Therefore, two models compared in this paper: a hybrid model composed of CNN for financial news and a Long Short-Term Memory (LSTM) for technical indicators, called SI-RCNN; and an LSTM network for technical indicators only, called I-RNN. The output of each model is used as input for a trading agent who purchases stocks on the current day and sells on the next day. When the model predicts that the price will increase, then the agent will sell stocks on the current day and buy the next day. The suggested approach reveals that financial news plays a key role in stabilizing the performance and almost no change in comparing various sets of technical indicators. The results of this paper showed that the SI-RCNN architecture would make a profit (13.94 % in 8 months) compared to a buy-and-hold strategy of 3.22 % over the era. This is shown by the lower output of the I-RNN model, which uses only a collection of technological inputs, compared to the SI-RCNN model that uses a hybrid input.

Furthermore, the experiments showed that it is important which set of technical indicators is chosen as input for the model since the SI-RCNN model is proved to be better than the SI-RCNN-2, which uses the first and second set of indicators. The I-RNN model is better than the I-RNN-2. Besides, tests have shown that the accuracy of more than 50 percent will not automatically contribute to a successful series of forecasts, as SI-RCNN-2 has reached an accuracy of 51.08 percent and a loss of $935.03. This may be occurring due to the difference among daily price variations, brokerage cost, and also because our models do not weight volatility. So, both 0.1% up and 3.5% up is labeled as [1, 0], possibly generating mispredictions on major movements.

It is important to note that deep learning models need a large amount of data, but forward-looking stock market prices do not automatically correlate with future actions. (Vargas et al., 2017)

Zhang et al. (2018) leverage the consistency between different data sources and created a multi-source multiple instance model that can effectively integrate incidents, emotions, and quantitative data into a comprehensive structure. They apply a new event extraction and representation approach to capture news events effectively. Therefore, their method would evaluate the value of each data source automatically and classify the key input information that is considered to guide the movements, making the predictions interpretable.

Their contributions were 1) they provide reliable and accurate stock market trends forecasts. They extended the multi-instance learning model to incorporate heterogeneous information like web articles, social media messages, and quantitative information. 2) In their context, the latent consistencies between different data sources are modeled by sharing the common, estimated true mark among the hinge losses at instance level of the various data sources. 3) They proposed a novel event representation model by first extracting organized events from the news text and then training them to obtain dense vectors using deep learning methods involving RBM and Sentence2vec. 4) Test results on two-year datasets (2015 and 2016) indicate that their plan will outperform the state-of-the-art baselines. In addition, it is possible to obtain the impacts of various sources and the main factors that drive the movements.

From January 1, 2015, to Dec. 31, 2016, they gathered stock market-related information and split the data into two data sets, one for 2015 and the other for 2016. Their data consists of three sections, the historical quantitative data, the news articles, and the social network posts which are summarized as follows in detail.

Quantitative data: sourced from Wind, a commonly used supplier of financial information services in China. The gathered data included the Shanghai Composite Index's average rates, market level adjustment, and turnover rate on each trading day.

News data: news articles about the macro economy were collect via Wind and receive 38,727 and 39,465 news articles respectively in 2015 and 2016. The news articles from major financial news websites in China are aggregated by Wind. Instead of the entire papers, they process the news titles to extract the facts, since the main subject of a news article is always summed up in the title.

Social media data: sentiments are extracted from the posts crawled from an ordinary Chinese investor social network named Xueqiu.3 for 2015 and 2016, a total of 6,163,056 posts are collected. They got time stamp for posting and the material for each post. When the stock market index raised for each trading day, this considered to be a positive example; otherwise, it would be a negative case. They used the data from the first ten months as the training set for each year, and the last two months as the test set. They evaluate their model's efficiency with similar lead days and various historical days. Lead days refers to the number of days the model makes forecasts in advance. The historical days reflected the number of days the multi-source information is used over. The performance criteria that they used were F1-score and accuracy (ACC).

The full implementation of their framework is named as Multi-source Multiple Instance (M-MI) model. (Zhang et al., 2018)

2.5 Gap Analysis

Although, in recent years, the wide adoption of machine learning techniques in predicting stock prices has led to the emergence of many articles on the topic, but there are still some areas to improve the stock prediction with the help of data mining and text mining.

Most recent works suffer from an inefficient word feature-extraction functions, or their accuracy was not sufficient enough to handle the dynamic nature of the stock since their databases were mostly synthetic.

Prior works mainly used TF-IDF or N-Gram method to filter out the candidates of keywords that is most relevant to stock price. Among proposed methods for feature selection and weighting, a few of them followed semantic approaches. The rest of the methods used Bag of words as their feature selection in which basically TF-IDF technique is applied in term selection and weighting. However, the results show that these methods are not sufficient enough to forecast the stock movements accurately. Table 2 shows a summary of the recent prior arts referring to the gaps in each research.

In this thesis, we applied Doc2Vec as our feature selection method from news texts. Doc2Vec is rather new numerical vector presentative as discussed in this chapter, and to the best of our knowledge, only a couple of previous works deployed Doc2Vec to vector the financial news. Recent efforts to address the obstacle of sentence level sentiment analysis of financial news builds on foundational work by Google with Doc2Vec(Le & Mikolov, 2015)

2.5 Chapter Summary

In this chapter, we summarized the different schools of thought according to stock behavior. We discussed fundamental and technical analysis to forecast the stock price. We reviewed prior arts on the fundamental analysis using machine learning and text mining methods on the news article and social media as the resource. Then we looked at the papers using the hybrid models to predict the stock. The main articles are listed in table 2.

The scope of this thesis is using fundamental analysis on the news titles to predict the daily stock prices. To achieve this goal, news articles and stock quotes are collected. Doc2Vec and Word2Vec word embedding models are implemented, and news are labeled based on the closing prices of the selected stock. After pre-processing, the data is used to train various models to predict daily stock prices.

Table 2. A Summary of The main prior arts

Paper	Data	Word Featuring	ML models	Gap Analysis
(Z. Wang et al., 2019)	New York Times News	Semantic Analysis	NN	Not able to define an efficient threshold for windows size
(Kraus & Feuerriegel, 2017)	Company Disclosure	Sequence Modeling base on Deep NN	RNN + LSTM	Accuracy < 55%
(Chiong et al., 2018)	Company Disclosure	TextBlob Python Library	SVM	A small increase in accuracy (≈ 2 %)
(Schumaker & Chen, 2009)	Breaking Financial News	Bag of Words	SVM	Only predict a fairly stable set of companies, unable to capture established relationship of the news
(Ding et al., 2015)	Yahoo! Finance	(Actor, Action, Object) + Word2Vec	CNN	Small dataset
(Merello et al., 2019)	News (Individually, group)	-	RNN+ LSTM	Accuracy decrease in big-time windows
(Bogle & Potter, 2019)	Twitter	StringToWordVector Function	SVM	Accuracy < 50%
(Lien Minh et al., 2018)	News	Stock2Vec	Hybrid	Using GRU since framework requires long training time and massive computational resources
(Vargas et al., 2017)	News	CNN	Dual LSTM	Model only focus in events with significant variation on the price
(Ding et al., 2015)	News on the Reddit	Sentiment Analysis	Hybrid	Lack of trusted News articles

CHAPTER 3

Data Collection and Preprocessing

The work described in this chapter aims to set out the original study that facilitated the collection of the data used in this thesis, and our process to parse the data into a final dataset for machine learning. We then present the word embedding structure, which is one of our main contributions. We include summary statistics on the final dataset.

3.1 Data Collection

For this research, two types of data are required; (I) time-stamped news articles, (II) stock prices. These data acquisition steps performed at the beginning. The following sections describe how these two types of data are gathered.

3.1.1 Collection of News Articles

Outperforming the market requires constant innovation in gathering the available data by investors. News is a potential new data source that is not fully exploited as yet. Many financial data service providers such as Thomson Reuters, Bloomberg, and Dow Jones are now available to provide sentiment signals to investors. We explored many options, including Event Registry[1] , Bloomberg[2], Reuters[3] , Yahoo![4], Google[5], Webhose.io[6], Intrinio[7], S&P Capital IQ[8], and Factiva[9]. Many sources could not be extracted easily or at a high cost. While exploring news content, it was evident that some sources lacked a fair representation of company news or contains irrelevant news. With the intention of including all news relating to a specific company, whether bad or good, minor or significant, it was clear that some sources were unable to provide such data. Finding valuable financial news data that could be extracted was a challenge. Intrinio is the source used for the news data required received through an API call. Bloomberg is the source of the daily close stock price used, collected at a Bloomberg terminal for the dates associated with the news release dates.

In this study, the news article dataset was gathered from the API of Intrinio by applying Python's library of pandas-DataReader. Intrinio is a financial data company with a mission of empowering fintech innovation by providing affordable, high-quality data to developers and engineers from fintech companies, large institutions, hedge funds, startups, and universities. The Intrinio platform consists of an application programming interface (API) supported by over 200 types of financial data feeds that can be easily integrated into applications, websites, widgets, algorithms, and other financial tools. We chose Apple sock news articles released by the Intrinio Financial Data Platform, ranging from May 24th, 2016

[1] https://eventregistry.org/
[2] https://www.bloomberg.com/canada
[3] https://ca.reuters.com/
[4] https://ca.finance.yahoo.com/
[5] https://www.google.ca/finance
[6] https://webhose.io/
[7] https://intrinio.com/
[8] https://www.capitaliq.com/
[9] https://professional.dowjones.com/factiva/

to January 9th, 2020. There were altogether 1325 trading days. With access to the API, Python can automatically download the daily time series data, and to assign news to dates within stock trading hours, financial news was released after 4 pm (when the stock market closes), the news affects the next day's stock price.

After collecting the raw data via calling the API, all the news articles converted to the CSV (comma-separated values) format files. Each data contains URL, Titles, Date, and summary of the news that shows when and where the news was published. Since Radinsky et al. (2012) and Ding et al. (2015) show that news titles are more useful for prediction compared to news contents, we extracted the date and titles from the dataset. Table 3 shows an instance of what we have in the final CSV:

TABLE 3. COLLECTED NEWS DATA SAMPLE

URL	Title	date	summary
https://finance.yahoo.com/news/greenpeace-calls-alibaba-tencent-maiden-020000610.html?.tsrc=rss	Greenpeace Calls Out Alibaba, Tencent in Maiden China Scorecard	2020-01-09 02:00:00+00:00	(Bloomberg) -- Greenpeace, calling attention to the Chinese technology …

3.1.2 Collection of Stock Quotes

Daily Apple's stock prices and quotes were gathered. A program in Python was written for collecting daily quotes from the Intrinio API, which has daily quotes for the thirty top DOW JONES companies. Daily quotes consist of the date, opening price, high price, low price, and the closing price for that day. In our initial sample of the news articles, a total of 1325 news were obtained between May 24th, 2016, and January 9th, 2020. This provided a sufficient amount of observations to analyze the effect of daily news on market characteristics. Figure 3 is the candlestick chart with Apple's stock fluctuation as an example. This figure is used to give a brief depiction of the gathered data. For this figure, which has a large scale of data, it could be hard to recognize the candle shape. However, we can still see the trends by noticing that the green candle charts mean rising in the price, and red means dropping. The blue line represents the mean price line. 'Close' values are used to show stock price data in this research.

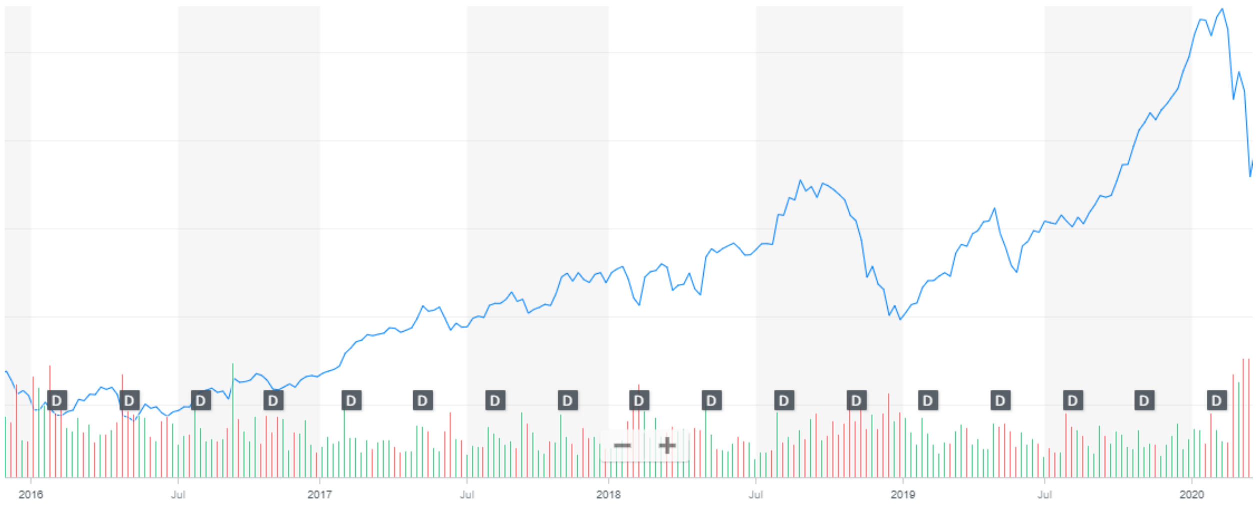


FIGURE 3. APPLE INC. (AAPL) STOCK HISTORICAL PRICES & DATA

🟩 Increasing in the price 🟥 Decreasing in the price

For the first attempt, the difference in stock price on each day is calculated in this phase as follows, and saved to the same CSV file which contains other daily stock quotes:

The difference of stock price on each day is calculated and saved to the same CSV file, which contains other daily stock quotes.

$$Difference = Closing\ price\ (n) - Closing\ price\ (n-1) \quad (1)$$

Where, n indicates the day.

The following table shows an example of the historical finance data stored in the CSV format.

TABLE 4. COLLECTED STOCK QUOTES SAMPLE

close	high	low	open	date	difference
309.63	310.43	306.2	307.235	1/9/2020	8.36

3.2 Data Labeling

Following the data collection, we merged two datasets. This step was taken to combine multiple data tables with various information into one dataframe. To do that, first, we converted the date type of the News Articles from (date, time) into date the only format. Second, we used the INNER MERGE that keeps only the common values in both left and right dataframes based on the date. This process also detects the corrupt or inaccurate records from the dataset and identifies incomplete, incorrect, inaccurate, or irrelevant parts of the data and delete dirty or coarse data. In the final CSV, each row carries the news article and stock quotes data together.

There are two strategies used to label news articles using historical stock prices: an open-to-close return (day-time return) and a close-to-close return (overnight return). The open-to-close approach is used to evaluate the document's label. The reason we chose open-to-close return rather than close-to-close return originated from the realistic point of view mentioned in Wang et al. (2009). The close-to-close records are more comparable to the total price of the same stock, implying that the close-to-close return contributes more to the overall return. In this study, the close-to-close return is stored in the "difference" column.

The goal of article labeling is to classify each of them into either sell, (which drives the price up), or buy, (which leads to a downward trend of the price), or hold, (which says it is not convinced to buy or hold the stock share according to the released article). In this study, we choose a three-classes approach (hold, buy, sell) to reflect the direction of the stock prices.

To pursue this goal, we set out a threshold to distinguish various classes from each other. For the first try, we assumed the threshold as the following:

$$T = K * [Closing\ price\ (n) - Closing\ price\ (n-1)] \quad + C \qquad (2)$$

Where K represents the percentage in prices difference which warrants a change in the decision (Buy, Sell, Hold), and C is a constant parameter as a bias to offset for the flat cost of making a transaction to buy or sell the stock. T represents the threshold is used to make the decision, i.e. to distinguish the three classes (Buy, Sell, Hold). We further realized that, Equation (2) cannot be a good representative to label the articles since the effect of one article can last for many days (improving long term stock market prediction with text analysis). This leads us to a new formula that includes two days before of the nth date with damping factors for each day as shown in Equation (3):

$$\mathrm{T} = \mathrm{K} \times \left[Closing\ price\ (\mathrm{n}) - \left(\alpha \times Close\ price\ (\mathrm{n}-1) + \beta \times Close\ price\ (\mathrm{n}-2)\right)\right] + C \quad (3)$$

Where T represents the threshold to make a decision (to buy, sell, hold), K represents the percentage in prices difference which warrants a change in the decision (Buy, Sell, Hold),

C is a constant parameter as a bias to offset for the flat cost of making a transaction to buy or sell the stock, α and β are damping factors carrying and incorporating the significance of the history of the news from previous days when calculating the threshold. Without losing generality, in this thesis we set damping factors α and β empirically to $\alpha=0.6$, and $\beta=0.4$. We also set the percentage factor $K=0.02$. We chose 2% of the change in daily stock price since, in comparison with *Apple's* share price in our study period, this is what a shareholder can be charged as a typical trade fee whenever a share is sold or bought.

To label the news article in our dataset, a return less than -*T* is considered as class 0 (Sell), a return between -T and T is labeled as class 1 (Hold), and a return higher than T is classified as class 2 (Buy).

$$\begin{cases} Difference < -T & \rightarrow Sell \\ -T < Difference < T & \rightarrow Hold \\ T < Difference & \rightarrow Buy \end{cases} \qquad (4)$$

Where Difference is calculated from Equation (1), and T from Equation (3).

This step is done after the data acquisition step and before the document preparation. The reason why this step is performed before any news article documents are prepared is that this step only needs the IDs of news articles, publication date and time, and stock prices. It does not need the news content. This process gave a dataset with 1076 news articles labeled as *buy*, 7611 published news which tagged as *hold*, and at the end 536 articles in the *sell* class.

3.3 Word Embedding

In machine learning, it is always vital to preprocess the data. The better quality of data we feed to a machine learning model, the better quality of results we will get. The data was cleaned upon creating a language model. Compared to other natural language processing tasks, stop words were not removed because they are essential for understanding negativity and positivity. This is one of the reasons why Word2Vec is often used in sentiment analysis tasks. Since it is possible, to sum up, or average the word embedding of all words of a sentence to form a new sentence, the final embedding of a positive sentence and negative sentence will differ. Several steps were taken to clean the data. First, all the words of the articles were lowercased. This helps reduce the number of words in the vocabulary and increases the number of times the words appear in context. This helps the algorithm to learn better embedding. To reduce the words, there are two well-known techniques, namely stemming and lemmatization. The stemming process will reduce the word length by cutting out the suffix. Because this technique might create words that do not exist, and we want to create a language model, the lemmatization technique was chosen, which brings the words to its base. The data was lemmatized using the WordNet lemmatizer. This process brings the word to its base form, which has the same benefits as the lowercase process by reducing the number of unique words in the vocabulary and allowing more times for the words to be seen in context (Prabhakaran, 2018). For example, usually, a word will appear in the same context, no matter if it is plural or not. We need to convert the high-dimensional representation of the news into shorter vectors to apply our classification methods. This approach is based on the word embedding method, which maps the news articles into vectors of low dimensions. We have experimented with our word embedding with two methods: one is using the number of word vectors, where each word vector is derived from word2vec (Mikolov et al., 2013). The other, using doc2vec implemented at (Le & Mikolov, 2015)

3.3.1 Word2vec

The cutting-edge algorithm for learning these embedded words is word2vec, created in 2013 by Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean (Mikolov et al., 2013). In this model, single-hot encoded vectors representing sampled words from a text extract are passed through a shallow neural network to predict neighboring words from the excerpt. As this network's input weight matrix is modified by back-propagation to optimize the probability of predicting correct term co-occurrences, each column in the matrix reflects the "value" of one of the terms in the vocabulary within the sense of its surrounding terms in the corpus. Therefore, such weights can be used as distributed representations of vectors for words.

The word2vec algorithm can be configured to use either a Skip-Gram or a Continuous-Bag-of-Words process. Each consisting of only one input layer, one projection layer, and one output layer (Figure x). It allows training of high-dimensional vectors from vast data sets with excellent computational efficiency due to the lack of multiplications of a dense matrix. Indeed, the main aim of the authors was to build strategies that would be used for large-scale vocabulary.

Most contextual words are taken as inputs in the Continuous-Bag-of-Words (C-BOW) model to predict a single output word. This method is called so because it does not matter the order of the terms used to predict the output. In this sense, each word in the vocabulary, V, is represented in a matrix W with a unique column vector. N context words are selected from about a target word in the input layer and encoded in one-hot vectors or vectors of size V × 1where an index is allocated to each individual word in the vocabulary, and words are encoded by putting a 1 in their respective index and a 0 elsewhere. The input layer is projected to an N × V space with a standard weight matrix at the projection layer. Finally, a log-linear classifier is used to predict the target word.

The objective for the C-BOW model is to maximize the log probability, given a sequence of training words $w_1, ..., w_T$:

$$\frac{1}{T} \sum_{t=T}^{T-N} logp(w_t \mid w_{t-N}, w_{t-N+1}, \dots , w_{t-1}, w_{t+1}, w_{t+2}, \dots, w_{t+N}) \quad (5)$$

Where

$$p(w_t \mid w_{t-N}, w_{t-N+1}, \dots , w_{t-1}, w_{t+1}, w_{t+2}, \dots, w_{t+N}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \quad (6)$$

And y_i are the long-probabilities for each output word i.

Thus, the C-BOW model is trained in predicting a word w_t given a window of N-words that exists before and after it. When the model training is complete, the matrix of weights can be split into vectors representing each word. Since these weights have been trained to transform the one-hot encoded vectors to enhance the probability of their appropriate adjacent words for each word in the vocabulary, the weight vectors accurately reflect the "meaning" of each word. The weights matrix encodes how each hot vector needs to be altered to represent its place most thoroughly in the original text. Hence, its distributed embedding among the other terms. These vectors form the representations of the words in a corpus, which are highly useful as inputs to other machine learning algorithms.

Conversely, a single input word is used in the Continuous-Skip-Gram Model to predict other words which have high probabilities of occurring in the sense of the input (within N words before or after the input). Again, it employs a continuous layer of projection accompanied by a log-linear classifier. In order to train the Skip-Gram model, given a sequence of training words w_1, w_2, ..., w_T from a size V vocabulary and a training scope N, the average log probability of each word is maximized with the objective given input word.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-N<j\leq N, j\neq 0} logp\left(w_{t+j}\middle|\, w_t\right) \quad (7)$$

where $p(w_{t+j}\ |w_t)$ is given by the softmax function:

$$p(w_0|w_I) = \frac{\exp(\mathrm{v}'^{T}_{w_O} v_{w_I})}{\sum_{w=1}^{V}\exp(\mathrm{v}'^{T}_{w_O} v_{w_I})} \quad (8)$$

and v_{w_I} and v'_w are the input and output vector representations of a word w, respectively.

Terms further from the input word are often less tested during training due to their supposedly decreased significance. The authors show that using the models described previously, the use of simple vector algebra can draw analogous relationships between vocabulary terms. Using cosine distance as a metric, X = vector("biggest")vector("big) "+ vector("small") yields a vector X that is very close to the qualified embedding for "smallest" other relationships which can be identified by the algorithm include Man-Woman, City-in-State, Singular-Plural, and Opposite.

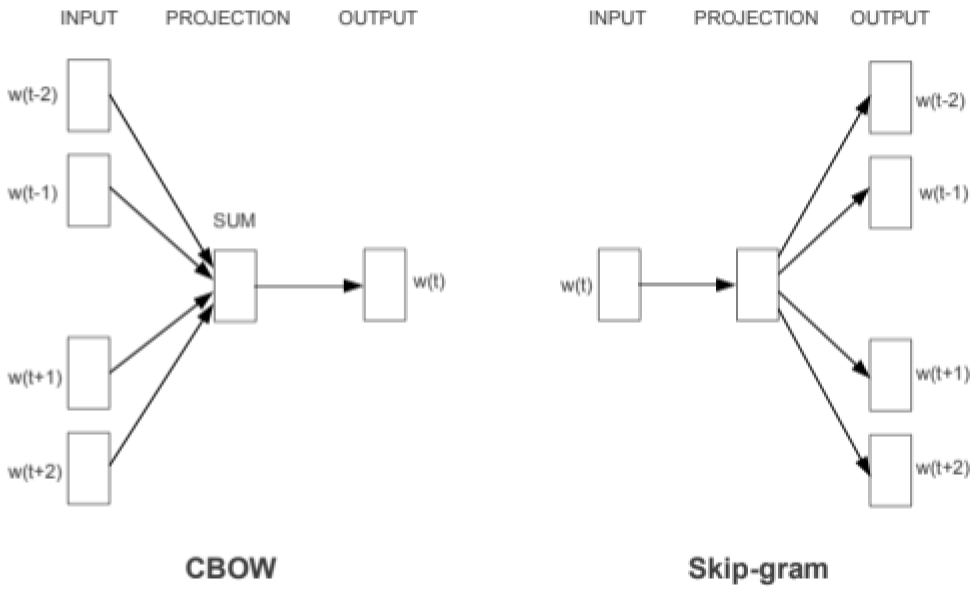


FIGURE 4. WORD2VEC MODEL ARCHITECTURES (MIKOLOV ET AL., 2013)

With over 2 million distinct words in our corpus' vocabulary, word2vec is the perfect solution for this study to establish word representations. For data of this size, more conventional methods like one-hot encoding are impractical, if not prohibitive, in computational terms. However, the model's ability to extract semantic similarity and analogous grouping of vocabulary terms provides a solution to confronted shortcomings when applying LDA to such results.

3.3.2 Doc2vec

Paragraph Vector, or doc2vec, is commonly known as implemented as an extension of word2with that trains a vector representing the full training case or text along with embedded word representations. This algorithm is particularly effective because it allows variable-length text pieces to be interpreted as fixed-length vectors. Using texts as inputs are required for most machine learning algorithms too. The doc2vec algorithm's architecture can again take on two versions, the Paragraph Vectors Distributed Memory Model (PV-DM) or the Paragraph Vectors Distributed Bag of Words Model (PV-DBOW). These architectures are shown in Figure 3.4. In the Paragraph Vector paradigm, each piece of text in the corpus is mapped to a unique vector in a matrix D. Thus, for a corpus of p documents, D = {d_1, d_2, …, dp}.

For PV-DM, training takes place in the same way as for word2vec in the C-BOW model; several background words are passed as input to the model to predict the next word given the above probability equations. Nevertheless, in this case, a paragraph vector, d_a, is taken from D to represent the text from which the words at issue were drawn.

As an input to the model, d_a is averaged or concatenated with the term vectors. Consequently, as vectors are modified during training, the paragraph vector comes to represent a collective "memory" of the states of all the vectors of its words. Importantly, all paragraphs share the word vector matrix, but the matrix of paragraphs is only shared across contexts created from the same paragraph. Trained word vectors thus reflect the sense in which words are presented in the corpus. However, paragraph vectors reflect particular instances of words throughout each paragraph and the co-occurrence of words. The only adjustment that needs to be made to the C-BOW structure during training is to expand the input combination form, h, to concatenate or average the document vector with the word vectors as input meaning. So, the model is equipped by maximizing the average probability of logs:

$$\frac{1}{T}\sum_{t=T}^{T-N} logp(w_t \mid w_{t-N}, w_{t-N+1}, \dots, w_{t-1}, w_{t+1}, w_{t+2}, \dots, w_{t+N}, d_a) \quad (9)$$

In order to predict the word w_t given meaning w_{t-N}, ..., w_{t+N}, where all words were taken from the document d_a. The term vectors and vectors for paragraphs are equipped with stochastic gradient descent and back-propagation. A new vector d_a is applied to the paragraph matrix D to determine a vector for a new paragraph. The word vectors and model weights are set while the gradient descent is implemented to modify only the paragraph matrix.

For the Paragraph Vector model, the following format, PV-DBOW, imitates word2vec as the Skip-Gram model. A single paragraph vector is taken as an input, and the model is asked to predict multiple terms from that paragraph as being randomly sampled. Therefore, as input, the model is given a single paragraph vector d_a from D and is trained to predict several vectors w from W. A word from a fragment of a text from one of the paragraphs is sampled during gradient descent. Then, despite the paragraph from which it was derived, the word is marked as one of the vectors in W. As this format focuses on classifying words, it is not necessary to store the vectors for the individual words during the training, making this approach less computationally intensive than the other one. The authors note that PV-DM alone appears to perform well, but a concatenation of the vectors learned by each approach leads to the best results, with an error rate of 7.42 percent on a standard data set for sentiment analysis, and only 3 percent on a custom information-retrieval task (Le & Mikolov, 2015)

PV-DM is an unsupervised learning algorithm that is an important feature of the Paragraph Vector model. Algorithms use co-occurrences of input data in unsupervised learning, rather than marks, to conclude the statistical structure implicit in the input space (Le & Mikolov, 2015). Doc2vec is unsupervised in that no specific target outputs or identifiers are needed to be associated with input documents. Alternatively, it draws its predictions from sliding windows of word meanings that exist within the input texts.

Doc2vec has been developed with the intention of training on huge datasets. Removing the constraint of having labeled data makes it possible to use such algorithms in cases where labeled data is scarce, but unlabeled data are readily accessible. The paragraph vector algorithm can be trained on a complete dataset of which a subset is named. Then, only the vectors for which labels are available can be transferred to a predictive task classifier (Le & Mikolov, 2015). The unsupervised nature of the training phase enables researchers to capitalize on as many similar data as they have at their hands while keeping only labeled data for classification models. This may require fewer training samples to perform well. The applicability of that algorithm dimension is discussed in the section below.

3.3.3 Training Word Embedding Models

The raw material for this research was news articles. There was no large, publicly available collection of news articles to use for this research and so there was a need to scrap a wide number of documents from the Internet. This section explains the system's crawler, extraction of text, and addition of meta-data parts.

3.3.3.1 Data Collection

The functional requirements of the crawler were to: 1. scrape news stories from the Factiva news database, 2. autonomously extract the story text from the original HTML, 3. to identify the headline, 4. to identify the news body 5. to identify duplicate news stories from the same source.

This information was available in the search engine provided by the Dow Jones Factiva. Factiva is a Dow Jones & Company owned Market Knowledge and Analysis Platform. Dow Jones' Factiva.com integrates over 32,000 outlets to provide 28 languages for students, faculty, and librarians to access premium content from 200 countries. Users have access to a wide variety of newspaper content, newswires, business newsletters, blogs, company reports, and more. The broad range of content provides both local insight and a global perspective on business issues and current events. This especially important for research requiring up-to-date information on companies, industries, and financial markets. Factiva search engine provides news articles from different resources categorized by Date, Industry type, and language. Each news includes a published date, headline, and a uniform resource locator (URL), which provides a link to the news story. All the results would be expected to contain financial stories since they were gathered by Dow Jones.

The process of scraping news stories had two distinct parts: a crawler that fetched news story HTML and a story text extractor that extracts story text from the news story HTML. The crawler relied upon a list of searched news articles in the Technology industry from the Factiva. The web crawler crawls the site to look for news articles fitting these parameters. Each news article is stored in a text file under the folder named after the company it belongs to. The naming convention of the text file is presented as: "date time company-ticker news-source title.txt". The news articles information is stored inside the CSV file with additional information in a structured way that makes it easier to be reused later on:

```
<doc>
<date></date>
<time></time>
```

<source_link></source_link>

<source></source>

<author></author>

<title></title>

<body></body>

</text>

</doc>

However, to train our models, we gathered news titles and bodies in CSV format separately. In the end, we came up with less than 1 GB finance news data.

3.3.3.2 Preprocessing

Text preprocessing is the procedure of clarifying each language structure and eliminating the language-dependent factors as much as possible. There are many different tasks under preprocessing, but some of the most common ones are tokenization, stop-word removal and word stemming.

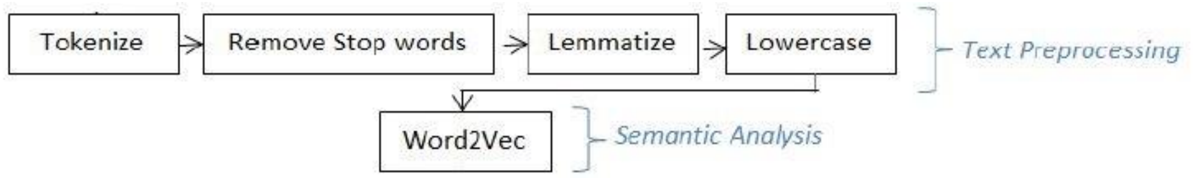


FIGURE 5. DATA PREPROCESSING

I) Tokenization

Tokenization is the method of splitting a text stream into symbols, words, phrases, or other significant elements called tokens. Further text mining techniques are used for these tokens. Usually, word tokens are sent to preprocessing stages such as stop-word removal and stemming, which are later identified. These also serve as inputs for the extraction processes of functionality. There are several ways Text streams can be tokenized into tokens. A simple approach would be only breaking the text into blank spaces, but more advanced methods also take into account punctuation and other sings.

The tokenizing method used in this research would tokenize the following text string:

"Facebook's official Twitter and Instagram accounts were hacked!"

First, splitting it on blank space. Then, followed by splitting it on most special characters. The tokenized string would then consist of the following tokens.

[' Facebook ', ' 's ', ' official ', ' Twitter ', ' and ', ' Instagram ', ' accounts ', ' were ', ' hacked ', '!']

II) Stop-Word Removal

Stop words are a language's high-frequency words that do not hold any critical information independently. Throughout the preprocessing stage, these terms are often omitted to reduce the number of features, thereby reducing the amount of noise. Stop words may, however, contain a large amount of information along with other words. That implies that in some cases, such as when looking for sentences or names, some of the stop-words are often retained and not eliminated. Closed words of the class, such as articles, pronouns, prepositions, and conjunctions, are typically included in the lists of stop words. This also contains some of the most commonly used open class phrases, such as auxiliary verbs. It is also possible to construct domain-dependent stop-word lists by filtering out words of high and low frequency or by using some statistical measures like information gain or chi-square to filter out less informative terms. All terms that occur in the given stop word list are deleted from the source documents during the deletion process.

III) Lemmatization

Many types of research used stemming before train their model. Stemming, in linguistic morphology, means the reduction of a word from its inflected form to its root, stem, or base form. The stem does not have to be the morphological root words. Instead, it is usually enough to map similar words to the same stem. Even though this stem is not a real root in itself. Use in information retrieval, and Stemming is applied in information retrieval from natural language processing and other methods dealing with text analysis. It is regarded as a standard technique for discovering the semantic similarity between the different morphological variants of a word. Because this might create words that do not exist while the aim is to create a language model. Thus, the lemmatization technique was chosen. The data was lemmatized using the WordNet lemmatizer. This process brought the word to their base form, which has the same benefits as the lowercase process by reducing the number of unique words in the vocabulary and allowing more times for the words to be seen in context (Prabhakaran, 2018). For example, usually, a word will appear in the same context, no matter if it is plural or not.

As a result of lemmatization, they both gained a similar feature instead of having two features that the computer would see as entirely different even though they obviously have a robust semantic similarity for humans. Supporters for word stemming claim that it has the benefit of reducing the dimensionality of items, making the data less fragmented and more comfortable to work with. Nevertheless, some experimental studies have shown that stemming may often be detrimental to a text classifier's effectiveness (Baker & McCallum, 1998).

Some example of words that are or might be stemmed:

Stemmer, stemming, stemmed → stem

Financial, quasi-financial, prefinance→ finance

There are many algorithms for performing stemming. One of the efficient algorithms is Porter stemmer. It uses a series of special language rules to turn a word into its base form (Ghosh, Modak, & Mondal, 2014). However, a different stemmer for English is written in the language of Python created by Porter in this article.

3.3.3.2 Training the Word2Vec

The Word2Vec algorithm comes with the package *genism* in Python. By specifying it to use fast computation, it can be trained so fast. The total number of words in the corpora is a total of 8 million words. This number is much smaller than using the whole Wikipedia dataset, which can contain up to two billion words in English. Different size of embeddings was trained to create different language models. Different windows sizes were tested. A window of size five was used after looking at the different similarities; it was what offered the most results.

One way to test our model is by comparing the similarity of words. In this case, a higher similarity means a more accurate model. For example, the model was tested on the Tesla Company and its CEO Elon Musk. The model captured 0.55 cosine similarity between the company and its CEO (this is considered a high similarity rater based on the -1 to +1 continuum) cosine similarity. There is also a function in the *genism* package for calculating the cosine similarity known as "similarity", where you provide those two words that you wish to have the similarity calculated. Two identical words will have a cosine similarity of 1, and the similarity will range from -1 to 1. It is hard to understand exactly what value represents very similar words and what threshold should use to validate the similarity measures. However, it is at least possible to verify if, in general, the similarity between two similar words should be higher than two irrelevant words. This concept is the basis of how semantic relationships are captured in the geometric position of the words. However, as these models are insensitive to word order, embeddings built using these models are suboptimal for tasks involving syntaxes, such as part of speech tagging or dependency parsing. This is because syntax defines "what words go where?" while semantics defines "what words go together". Obviously, in a model where word order is discarded, the many semantics relations between words cannot be captured properly. For instance, while most words occur with the word "the", only nouns tend to occur exactly afterward (e.g., the cat). This is supported by empirical evidence that suggests that order-insensitivity does indeed lead to substandard syntactic representations. (Lau & Baldwin, 2016)The problem of word2vec is that although the vectors can contain some position information during training word2vec models, the word order has been lost during summing up. This results in the possibility of different tweets sharing the same vector as long as the words they used are the same. Moreover, this may also cause two tweets that contain very different words with the same vectors or in a closer distance. This problem can be solved by using doc2vec.

3.3.3.3 Compiling Training Documents for doc2vec

The process of training distributed vector representations for variable-length texts provides a unique opportunity for our research to create communication representations based on the similarities and nonlinearities between the messages themselves. As word vector representations are trained together with the doc2vec model, we adopt an effective training method that yields multifaceted data for multiple tasks. In sum, for the research, we used doc2vec with the following techniques to compile documents for transformation. First, all of the news bodies and names were concatenated into one single text. The next step was to "tokenize" the document for each of the document compilation methods mentioned above or to divide it into a list of its component pieces, such as words and emoticons, again using the TweetTokenizer. Prior to passing it as input to the doc2vec model, we did not perform any further preprocessing on the data. For our corpus, one of the key benefits of doc2vec is its ability to discern meaning for all context-dependent tokens. However, filtering out misspellings, stop words, or slang phrases was unnecessary as the model would learn to equate such phrases with their proper English counterparts.

3.3.3.4 Training doc2vec Model

The Doc2Vec implementation from the widely used *gensim* library was used to implement the Paragraph Vector model for our data. We picked the model's PV-DM implementation based on the original authors' remarks that for most tasks appears to function well by itself. We also induced the model to simultaneously train word-vectors in Skip-Gram fashion to train paragraph vectors in order to exploit the distributed representations of the terms for future testing within the text documents. We initialized the model, using a window size of 5 to learn vectors of size 100. The relevant parameters of the Doc2Vec model tuned for our research are the size of the window, the size of the vector, and the epoch number. The window size is the cumulative distance within a document between the word predicted and the reference terms used to forecast it. In their original Doc2Vec experiments with the IMDB dataset, Le and Mikolov used dimensionality of 400, but 100 dimensions were used in this research to reduce the computational cost. The number of epochs is set at 600 based on the work of Lau and Baldwin (2016), who defined it as the optimum value for the task of semantic similarity. Besides, the min count parameter used to determine the minimum threshold frequency to disregard those terms with a total frequency lower than the threshold set to one because papers are not repeated in the dataset, so the labels would only appear once. We maintained the architecture for comparability across document compilation methods across all training datasets. Every doc2vec model was trained on the full-text message corpus to allow the algorithm to capture the vernacular text on a larger scale and the word representations that inform the embedding of paragraphs. The conceptual basis for the doc2vec training algorithm makes these changes resilient for our methodology. If a new slang term replaces an old favorite utterance, then the word2vec mechanism that underlies doc2vec can learn how to associate it with both words of the same meaning. Thus, identical vector representations.

Figure 3 illustrates how document vectors are generated using the Doc2Vec PV-DM model. The news articles are labeled and shuffled with their respective specific IDs to get them in random order. The randomized tagged documents are then fed to the Doc2Vec model, which implicitly builds the vocabulary out of the documents received. After that, a sanity test procedure is conducted to ensure the model is adequately trained and behaved as planned.

In that, a vector is deduced from its text using the trained model for each of the documents. The document vectors initially created during training are ranked in the order of similarity with this newly inferred vector. This procedure leads the vector representation generated by Doc2Vec to preserve the semantic meaning of the words.

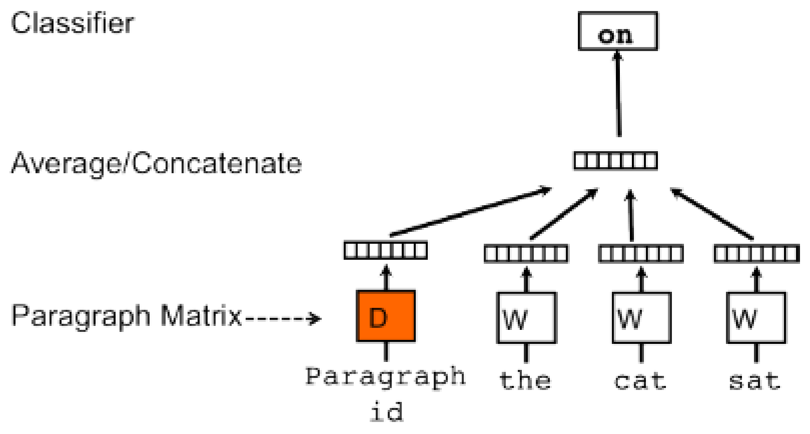
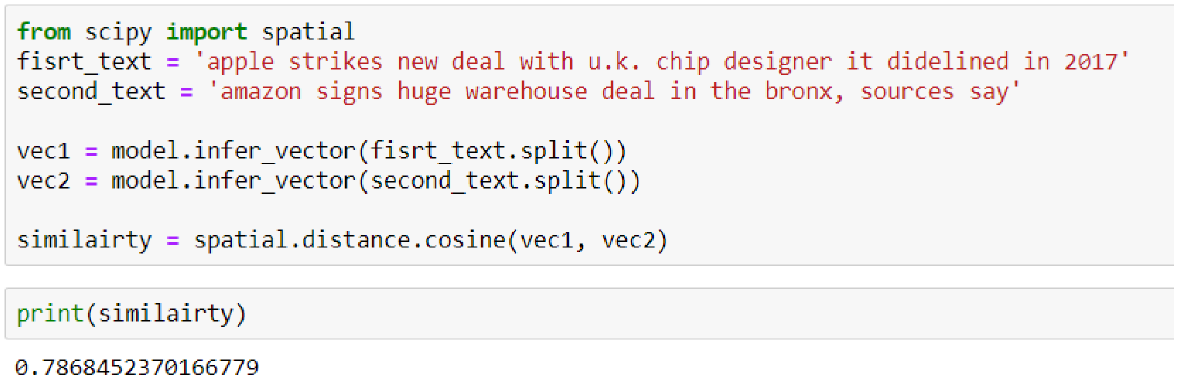


FIGURE 6.A FRAMEWORK FOR LEARNING PARAGRAPH VECTOR (LE & MIKOLOV, 2015)

3.3.4 Vectorization

The Doc2vec model proceeded to convert each subset into its word embedding representation (also known as an inferred vector). Each news title is represented by a vector of numeric values where each value was calculated via a trained Doc2Vec vector and resulted in 9222 × 101 matrices, where the last column allocated to the labels calculated from section 3.3.

We first begin with a search for the optimal parameters of the Doc2Vec model. Here, we train the model on about one million high-tech related financial news data. The motivation behind this decision was to use a security that represents the wide range of American companies that may be sensitive to the phrases which can move the stock. We observe that smaller vector sizes tend to have lower validation loss, with the lowest validation loss belonging to a vector size of 100. We observe the opposite relation with number of training epochs, where training our Doc2Vec models for 100 epochs resulted in the lowest validation error. Using these optimal parameters, we then tested our model by using similarity function. To inspect relationships between documents a bit more numerically, we can calculate the cosine distances between their inferred vectors by using the similarity_unseen_docs() function. This function takes as its parameters the Doc2Vec model we just trained and the two documents to be compared. As a measure of the documents' similarity, the function then returns a value between 0 and 1, where the larger the value, the more similar the documents are. Figure 7 shows an examples of similarity measures between two documents after our Doc2Vec was trained.



```
from scipy import spatial
fisrt_text = 'apple strikes new deal with u.k. chip designer it didelined in 2017'
second_text = 'amazon signs huge warehouse deal in the bronx, sources say'

vec1 = model.infer_vector(fisrt_text.split())
vec2 = model.infer_vector(second_text.split())

similairty = spatial.distance.cosine(vec1, vec2)
```

```
print(similairty)
```

```
0.7868452370166779
```

FIGURE 7. SIMILARITY MEASURED BETWEEN TWO TITLES

3.4 Data Balancing

The machine learning (ML) classification algorithms work well when the number of instances of each response variable is roughly equal. In a real dataset, it is almost rare to get a balanced dataset. In stock prediction studies, it is widespread to have an imbalanced dataset that may have a ratio of 1:100. Such an imbalance generally causes a classifier to be biased towards the majority class (Longadge, Dongre, & Malik, 2013). In machine learning, the problem refers to the class imbalance problem, where the number of instances of one class significantly exceeds the other classes. Class imbalance problem is well recognized by many researchers in various domains such as fraud detection, medical diagnosis, credit scoring, telecommunication as well as text classification. While the class imbalance problem is primarily attributed to the highly skewed distribution of instances in different classes, it may have other additional characteristics. Various characterizes of an imbalanced dataset are presented below:

- **Between class imbalance:** if the number of instances of one class is significantly less than the instances of other classes, it is termed as the *between class imbalance* in the dataset. (Sobhani, Viktor, & Matwin, 2015) posit that regardless of the extent of between class imbalance, if the classes are distinctly separated, standard classification techniques should be able to classify different instances correctly (Japkowicz and Stephen, 2002). However, once the between class imbalance is associated with other types of complexity (such as within-class imbalance and class overlapping), it can lead to misclassification of instances, and this problem is more acute with minority class instances.
- **Within class imbalance:** if a class has several smaller sub-classes that represent separate subconcepts, it may lead to within-class imbalance (Jo and Japkowicz, 2004).

"Subconcepts with limited representatives are called "small disjuncts". Classification algorithms are often not able to learn small disjuncts. This problem is more severe in the case of undersampling techniques. This is due to the fact that the probability of randomly selecting an instance from small disjuncts within the majority class is very low. These regions may thus remain unlearned" (Sobhani et al., 2015).

- **Class overlapping:** in an imbalanced dataset, if the instances belong to different classes overlap, it increases the possibility of misclassification by various standard learning algorithms. The difficulties associated with separating overlapping instances make a dataset more complex and often classify instances belong to minority class as the one in majority class (Lin et al., 2017; Batista et al., 2004).

3.4.1 Undersampling

One of the most straightforward methodologies to create a balanced dataset is random undersampling. To create a balanced distribution of majority and minority class, a subset of the majority class is randomly removed from the tuple (Han, Kamber and Pei, 2012). This technique reduces the data size. Hence, the run-time of the classification task reduces significantly. On the other hand, random undersampling is subject to high variance (Wallace et al., 2011). The majority class may include some sub-classes. Random undersampling may completely exclude instances from some sub-classes or may select only a few cases from a specific sub-class. This induces the risk of losing valuable information and may lead to poor predictive models.

As we have discussed above, random undersampling has the risk of overfitting the minority class, and often, valuable information from the majority class gets discarded (Han et al., 2012). When random undersampling removes a large number of instances from a sample, it can create small disjuncts. As Holte et al. (1989) argue this can make classification algorithms error-prone. Akbani et al. (2004) also made a similar argument. Besides that, since undersampling reduce the number of classes to minor class and it decreases the number of records to training the machine learning models, we decide to overlook this method

3.4.2 Oversampling

Oversampling in data mining techniques adjusts the training class distribution so that there is an equal distribution of minority and majority classes. Both oversampling and undersampling work with a similar concept but uses different techniques. In Oversampling, the minority tuples are resampled so that the resulting training set contains an equal number of positive and negative tuples (Han, Kamber and Pei, 2012). Random oversampling produces or replicates minority tuples by sampling with replacement. While it alleviates the class imbalance problem, it increases the risk of overfitting (Batista et al., 2004). To address this issue, new instances to the minority class need to be added. This is challenging and deems less practical due to the limited size of the minority class sample.

3.4.3 Synthetic Minority Oversampling Technique (SMOTE)

Chawla, Bowyer, Hall, and Kegelmeyer (2002) proposed the Synthetic Minority Oversampling Technique (SMOTE), which generates random synthetic instances of the minority class on the feature space rather than replicating the existing instances of the minority sample. Thus, it improves the bias. SMOTE algorithm uses a k-nearest neighbor technique to create these synthetic minority examples. For each minority class sample X_i, SMOTE finds the k-nearest neighbors, N_i for that instance. Then, it calculates the difference between the feature sample and its nearest neighbors, $\Delta = Xi - Ni$. Next, after multiplying this difference by random number 0 or 1, it adds it to the feature sample. As a result, it creates a set of random points along with the line segments between two specific features (Chawla, Bowyer, Hall and Kegelmeyer, 2002).

The random points are generated as

$$F_{new} = \mathrm{X}_i + \mathrm{X}_i - \mathrm{N}_i \times rand\ (0,1)$$

Where F_{new} the generated random point and rand (0, 1) is the random variable generated a random number either 0 or 1. In order to create a balanced dataset, different amounts of synthetic oversampling may be needed for different datasets.

The synthetic minority over-sampling technique ("SMOTE") is an academically accepted method to address class imbalance issues (Chawla et al., 2002). The SKlearn Imblearn implementation of SMOTE was used to create an over-sampled (on minority class) and under-sampled (on majority class) version of the Base dataset called the SMOTE dataset.

TABLE 5. NUMBER OF RECORDS IN EACH CLASS AFTER AND BEFORE BALANCING

Classes	# Before Balancing	# After Balancing
Hold	7611	7611
Buy	1076	7611
Sell	536	7611

3.5 Chapter Summary

This chapter has presented strategies for data collection and data preprocessing. These strategies attempted to collect data at two levels: news articles and stock quotes. Stock market data level used to classify the news articles to three classes. News articles were vectorized via two different word embedding methods: Word2Vec and Doc2Vec. After preprocessing phase the dataset has been balanced with help of SMOTE technique. These steps were taken to answer this question Can we extract features (in the forms of numerical vector) of the news texts from reliable sources to be used in building predictive models to forecast when to buy, sell, or hold of a specific stock (RQ1).

CHAPTER 4

MODEL BUILDING AND EVALUATION

The experiments conducted for this research consist of collecting financial news articles related to the DJI (Dow Jones Index), a well-known index of the New York Stock Exchange, and extracting certain features that influence the prices of the DJI. The movements of the DJI prices are needed to be learned by machine learning models in order to predict the unseen prices. Since the desired outputs are either buy, sell, or hold, the triple classification machine learning models are needed to classify the data set in order to figure out whether there is a strong enough correlation between the relevant financial news and the DJI price movements. If this condition is met, then it could predict the movement of the DJI unseen prices. Some of the methods commonly used for classification are Decision Trees, Random Forests, Bayesian Networks, Support Vector Machines, Neural Networks, and Logistic Regression. Each classifier is only advantageous in a selected domain based on the number of observations, the dimensionality of the feature vector, the noise in the data, and other factors. Some tools to evaluate the performances of the models, besides some techniques to optimize them, are required for such cross-validation and grid search. The goal of this experiment is to evaluate how the DJI prices change accordingly to relevant news and to see if there is an existing link between these two.

4.1 Model Selection

The first objective of this research is to find a meaningful correlation between the DJI related News Sentiments and the DJI price movements through specific machine learning models. The second objective is to determine if there is indeed a way to predict the movements of the DJI unforeseen prices using the News Sentiments.

The performances of classification models are analyzed using 10-Fold Cross-Validation techniques. The K-Fold Cross Validation Score function repeatedly and randomly splits the training set into K-folds and then makes predictions based on each fold using a model trained on the remaining folds. Also, the performance of each model has been evaluated using a T-test.

In order to do binary classification and considering the size of the data set, K-Nearest Neighbor (KNN), Decision Tree, MLP Classifier, Random Forest Classifier, and Support Vector Machine (SVM), models were chosen for the experiments of this research. As we discussed in section 2.4 previous studies mainly focused on SVM, KNN, Decision Tree, MLP Classifier, and Random Forest Classifier to predict the stock movements. Accordingly, we selected these five models in our thesis research as well.

The hyperparameters of the models which have a significant influence on the performance of the classification models have been optimized by using a loop function. This function exhaustively tries every combination of all the hyper-parameters for each model.

Data collection and preprocessing steps were described in the previous chapter. To learn the patterns in each data set, there should be enough samples of the three desired outputs (hold, buy, sell). To do so, a SMOTE oversampling method was used in order to balance the dataset. To train the predictive models, the News Vectors (created by Doc2Vec) and the desired outputs of the train set of each sample set were used.

4.1.1 The Tools

For conducting the analysis, the author applied Scikit[10]. Scikit-learn is a free software machine-learning library for the Python programming languages. It features various classifications, regression, and clustering algorithms such as SVM, random forests, gradient boosting, k-means, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The following are the results of the experiments conducted to get an accurate classification and determine if a significant correlation exists between the News vectors and the movements of the DJI prices from May 24th, 2016 to January 9th, 2020 exists. The preprocessed data set, which is explained in the previous chapter, has first been fit to all the classification models, and then their hyper-parameters were optimized to get the best accurate results.

4.2 Random Forest Classifier

4.2.1 Model Building

The third predictive model experimented is the random forest classifier. This is a meta-estimator that fits several decision tree classifiers on various sub-samples of the

[10] Scikit-learn was initially developed by David Cournapeau as a Google summer of code project in 2007. Its' name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately developed and distributed third-party extension to SciPy. The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, took leadership of the project and made its first public release on February 1st, 2010. As of 2017, scikit-learn is under active development (Scikit Website).

dataset. This predictor uses averaging to improve the predictive accuracy and controls its over-fitting (Shah, 2007). The sub-sample size is always the same as the original input sample size but, the samples are drawn with a replacement if bootstrap=True (default) (Shah, 2007). The following parameters have resulted in the best predictive accuracy for this data. Similar to the other models tested, the missing parameters have little to no effects on the accuracy and therefore have been changed to their default values.

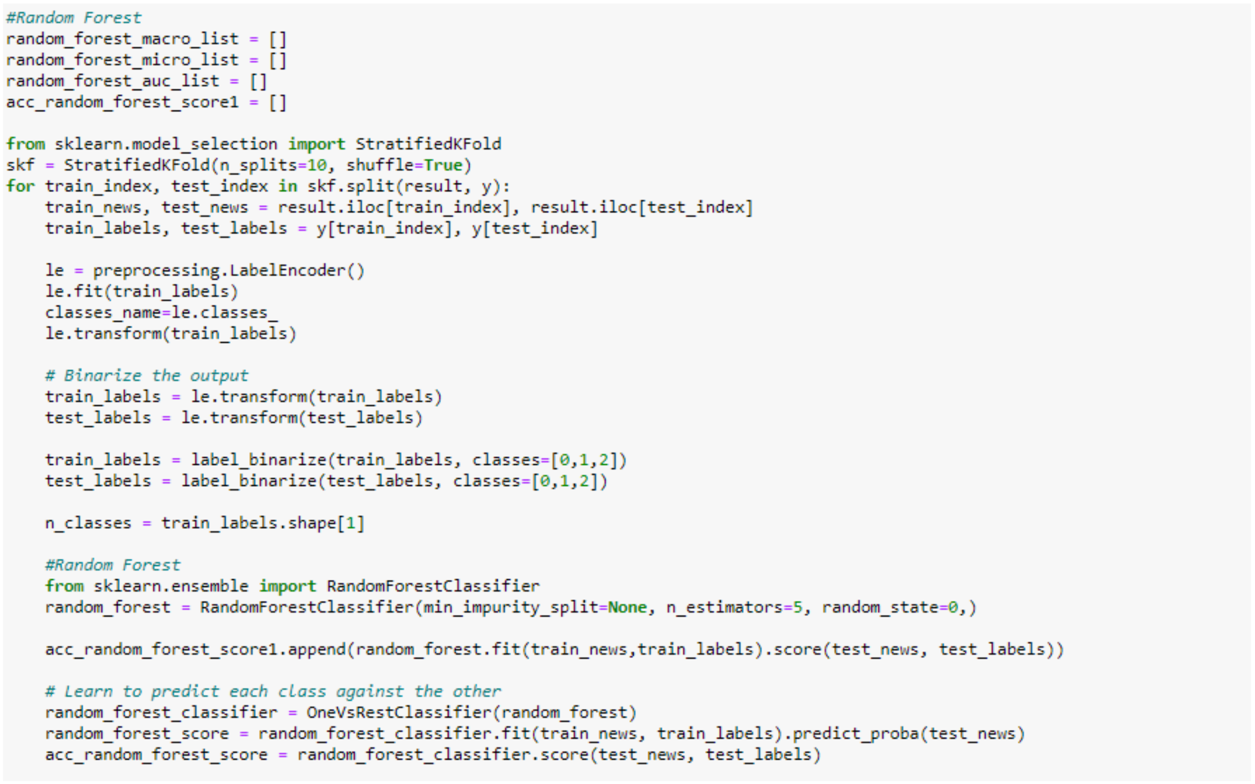
Therefore, the best parameters of Random Forest Classifier for this data set are: bootstrap=True, class_weight=None, criterion='gini', max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=5, random_state=0, n_jobs=-1, oob_score=False.

TABLE 6. DESCRIPTIONS OF THE PARAMETERS PF THE RANDOM FOREST CLASSIFIER

Parameter	Description
n_estimators	The number of trees in the forest.
Criterion	The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
max_features	The number of features to consider when looking for the best split: If int, then consider max_features features at each split. If float, then max_features is a percentage, and int(max_features × n_features) features are considered at each split. If "auto", then $max_{features} = \sqrt{n\_features}$ if "sqrt", then max_features can be calculated same as "auto".

	If “log2”, then $max\_features = log\ log\ (n\_features)$. If None, then $max\_features\ =\ n\_features$.
max_depth	The maximum depth of the tree. If None is given as a variable, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
min_samples_split	int, float, optional (default=2) The minimum number of samples required to split an internal node: if int, then considers min_samples_split as the minimum number. If float, then min_samples_split is a percentage and ceil ($min_{samples_{split}} \times\ n_{samples}$) are the minimum number of samples for each split.
min_samples_leaf	The minimum number of samples required to be at a leaf node: if int, then considers min_samples_leaf as the minimum number. If float, then min_samples_leaf is a percentage and ceil ($min\_samples\_leaf\ \times\ n\_samples$) are the minimum number of samples for each node.
min_weight_fraction_leaf	The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
max_leaf_nodes	Grow trees with max_leaf_nodes in best-first fashion. Best nodes are defined as a relative reduction in impurity. If None then unlimited number of leaf nodes.
min_impurity_split	The threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise, it is a leaf. min_impurity_decrease A node will be split if this

	split induces a decrease of the impurity greater than or equal to this Value. The weighted impurity decrease equation is the following: $$\frac{N\_t}{N * (impurity - N\_t\_R \,/\, N\_t * right\_impurity - N\_t\_L \,/\, N\_t * left\_impurity)}$$ where N is the total number of samples, N_t is the number of samples at the current node, N_t_L is the number of samples in the left child, and N_t_R is the number of samples in the right child. N, N_t, N_t_R, and N_t_L all refer to the weighted sum, if sample_weight is passed.
Bootstrap	Whether bootstrap samples are used when building trees.
oob_score	Whether to use out-of-bag samples to estimate the generalization accuracy.



```
#Random Forest
random_forest_macro_list = []
random_forest_micro_list = []
random_forest_auc_list = []
acc_random_forest_score1 = []

from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=10, shuffle=True)
for train_index, test_index in skf.split(result, y):
    train_news, test_news = result.iloc[train_index], result.iloc[test_index]
    train_labels, test_labels = y[train_index], y[test_index]

    le = preprocessing.LabelEncoder()
    le.fit(train_labels)
    classes_name=le.classes_
    le.transform(train_labels)

    # Binarize the output
    train_labels = le.transform(train_labels)
    test_labels = le.transform(test_labels)

    train_labels = label_binarize(train_labels, classes=[0,1,2])
    test_labels = label_binarize(test_labels, classes=[0,1,2])

    n_classes = train_labels.shape[1]

    #Random Forest
    from sklearn.ensemble import RandomForestClassifier
    random_forest = RandomForestClassifier(min_impurity_split=None, n_estimators=5, random_state=0,)

    acc_random_forest_score1.append(random_forest.fit(train_news,train_labels).score(test_news, test_labels))

    # Learn to predict each class against the other
    random_forest_classifier = OneVsRestClassifier(random_forest)
    random_forest_score = random_forest_classifier.fit(train_news, train_labels).predict_proba(test_news)
    acc_random_forest_score = random_forest_classifier.score(test_news, test_labels)
```

FIGURE 8. RANDOM FOREST MODEL BUILDING

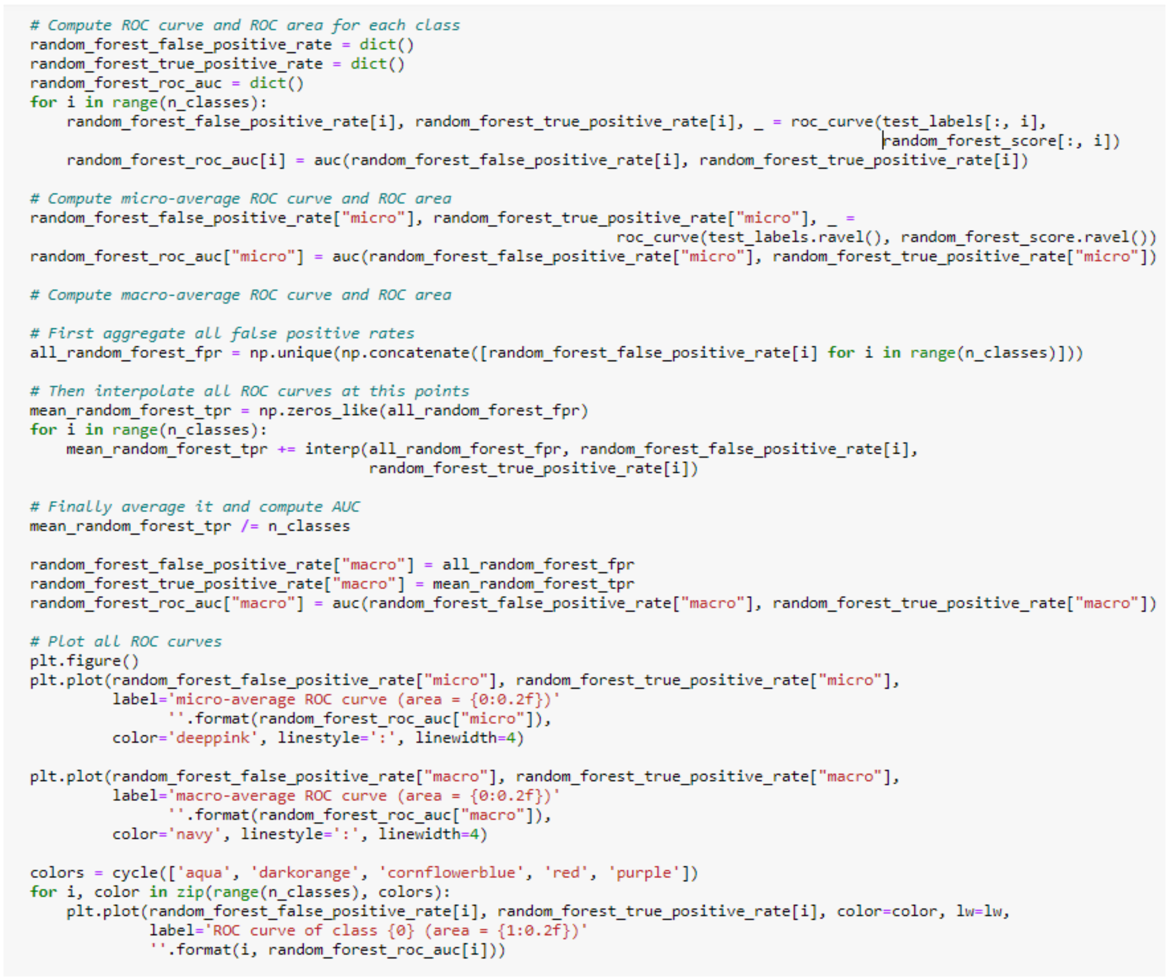
4.2.2 Classificaton Performance measures

One of the crucial points in the utilization of different classifiers is to measure the quality of the classification method to determine when to use a particular technique to achieve the best results. There are different evaluation measures for discrete and probabilistic classifiers. In discrete classifiers, a class is returned as a result of classification, whereas probabilistic classifiers return the probability that a sample belongs to a class. For discrete classifiers, evaluation measures such as precision, accuracy, recall, and F-score are used. For probabilistic classifiers, the Receiver Operating Characteristic (ROC) curve is typically used. In order to assess the performance of a classifier, two sets of data called to train and test datasets are needed. The training dataset is used to build the classification model, and the test dataset is used to evaluate the built classifier. The classifier labels the samples in the test dataset, and the result of the classification is measured using evaluation measures. The training dataset is more significant than the test data set (Wagner, 1979). The most significant disadvantage of this scheme is that the evaluation is performed on a small test set, which may not be an unbiased sample of the entire dataset. In order to alleviate these challenges, a method called K-fold Cross-Validation (Stone, 1974) divides the dataset into K equal subsets.

A receiver operating characteristics (ROC) graph is a technique for visualizing, organizing and selecting classifiers based on their performance. ROC graphs have long been used in signal detection theory to depict the trade-off between hit rates and false alarm rates of classifiers (Swets et al., 2000). ROC analysis has been extended for use in visualizing and analyzing the behavior of diagnostic systems. The medical decision-making community has an extensive literature on the use of ROC graphs for diagnostic testing(Bowers & Zhou, 2019). Swets et al. (2000) brought ROC curves to the attention of the broader public with their *Scientific American* article.

One of the earliest adopters of ROC graphs in machine learning, Spackman (1989), demonstrated the Value of ROC curves in evaluating and comparing algorithms. The recent years have seen an increase in the use of ROC graphs in the machine learning community. This is due in part to the realization that simple classification accuracy is often a poor metric for measuring performance (Provost & Fawcett, 2001). In addition, to be a generally useful performance graphing method, they have properties that make them especially useful for domains with skewed class distribution and unequal classification error costs. These characteristics have become increasingly important as research continues into the areas of cost-sensitive learning and learning in the presence of unbalanced classes.

In this study, we used the ROC curve to measure our model’s classification performance. Receiver Operating Characteristics (ROC) curve was conducted with coding from Sci-Kit Learn Machine Learning in Python for each of the ten folds in the stratified k-fold. Figure x illustrates one of the 10-folds of the Random Forest Multi-classifier model ROC curve, where class 0 shows the *Buy* class, Class 1 demonstrations the *sell* class, and class 2 demos the *hold* class.



```
# Compute ROC curve and ROC area for each class
random_forest_false_positive_rate = dict()
random_forest_true_positive_rate = dict()
random_forest_roc_auc = dict()
for i in range(n_classes):
    random_forest_false_positive_rate[i], random_forest_true_positive_rate[i], _ = roc_curve(test_labels[:, i],
                                                                                            random_forest_score[:, i])
    random_forest_roc_auc[i] = auc(random_forest_false_positive_rate[i], random_forest_true_positive_rate[i])

# Compute micro-average ROC curve and ROC area
random_forest_false_positive_rate["micro"], random_forest_true_positive_rate["micro"], _ =
                                                                roc_curve(test_labels.ravel(), random_forest_score.ravel())
random_forest_roc_auc["micro"] = auc(random_forest_false_positive_rate["micro"], random_forest_true_positive_rate["micro"])

# Compute macro-average ROC curve and ROC area

# First aggregate all false positive rates
all_random_forest_fpr = np.unique(np.concatenate([random_forest_false_positive_rate[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_random_forest_tpr = np.zeros_like(all_random_forest_fpr)
for i in range(n_classes):
    mean_random_forest_tpr += interp(all_random_forest_fpr, random_forest_false_positive_rate[i],
                                     random_forest_true_positive_rate[i])

# Finally average it and compute AUC
mean_random_forest_tpr /= n_classes

random_forest_false_positive_rate["macro"] = all_random_forest_fpr
random_forest_true_positive_rate["macro"] = mean_random_forest_tpr
random_forest_roc_auc["macro"] = auc(random_forest_false_positive_rate["macro"], random_forest_true_positive_rate["macro"])

# Plot all ROC curves
plt.figure()
plt.plot(random_forest_false_positive_rate["micro"], random_forest_true_positive_rate["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(random_forest_roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(random_forest_false_positive_rate["macro"], random_forest_true_positive_rate["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(random_forest_roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'red', 'purple'])
for i, color in zip(range(n_classes), colors):
    plt.plot(random_forest_false_positive_rate[i], random_forest_true_positive_rate[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, random_forest_roc_auc[i]))
```

FIGURE 9. RANDOM FOREST CLASSIFICATION PERFORMANCE MEASUREMENT

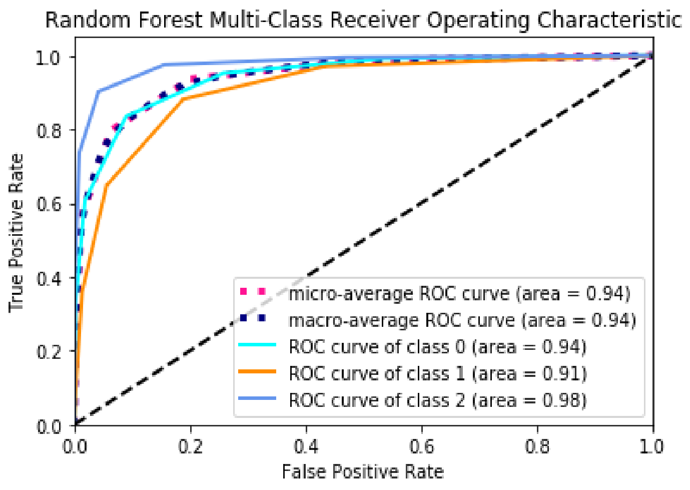


FIGURE 10. RANDOM FOREST ROC CURVE

Classification effectiveness is measured in terms of the classic IR notations of precision and recall, adapted to the case of document categorization. (Sebastiani, 1999) According to Zheng and Srihari (2003) classification effectiveness has been evaluated in terms of the standard precision, recall, F-measure Rijsbergen (1979) and precision/recall breakeven point. All of these measures can be calculated based on two conventional methods, namely macro-averaging and micro-averaging. (Aas and Eikvil, 1999) Macro-average performance scores are determined by first computing the performance measures per category and then averaging these by the number of categories. Micro-average performance scores are determined by first computing the totals of contingency table for all categories and then use these totals to compute the performance measure of each category. There is an important distinction between the two types of averaging. Micro and macro average of ROC Curves for our first model is shown in the table 7.

TABLE 7. MICRO - MACRO AVERAGE OF ROC CURVES TABLE FOR RANDOM FOREST

Classification Method	Micro Average	Macro Average	AUC Class 0 Average	AUC Class 1 Average	AUC Class 2 Average
Random Forest	0.7548	0.7738	0.9454	0.9137	0.969

4. 3 K Nearest Neighbors Classifier

The first predictive classifier trained is KNN, which is used as the first predictive classifier. This is a non-parametric method that is used for classification and regression. The following parameters have been determined to get the best predictive accuracy for this data. The parameters which were not mentioned in the tables are considered to be equal to their default value.

Best parameters of K Neighbors Classifier for this data set: algorithm='auto', leaf_size=10, metric='minkowski', metric_params=None, n_jobs=1, n_neighbors=4, p=2, weights='distance'

TABLE 8. DESCRIPTIONS OF THE PARAMETERS PF THE KNN

Parameter	Description
n_neighbors	Number of neighbors to use by default for kneighbors() queries
weights	weight function used in prediction algorithm ‘auto’, ‘ball_tree’, ‘kd_tree’, ‘brute’
leaf_size	Leaf size passed to BallTree or KDTree. This can effect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.
p	Power parameter for the Minkowski metric. When p = 1, this is equivalent to using *manhattan_distance* (l1), and *euclidean_distance* (l2) for p = 2. For arbitrary p, *minkowski_distance* (l_p) is used.
metric	the distance metric to use for the tree. The default metric is *minkowski*, and with p=2 is equivalent to the standard Euclidean metric.

After the model built and we deployed the 10-fold cross validation on it we got 10-fold cross validation results for the KNN. The obtained macro and micro averages showed in the table 9.

TABLE 9. MICRO - MACRO AVERAGE OF ROC CURVES TABLE FOR KNN

Classification Method	Micro Average	Macro Average	AUC Class 0 Average	AUC Class 1 Average	AUC Class 2 Average
KNN	0.7425	0.7574	0.9575	0.7468	0.9628

4. 4 Support Vector Machine

The second predictive classifier used is SVM. Support Vector Machine is supervised learning models that working with learning algorithms that analyze data used for classification and regression. The SVM training algorithm builds a model that assigns new examples to the category (Madge, 2015). The following parameters have resulted in the most accurate predictive data. The missing parameters are insignificant, so their values have been changed to their default values.

SVM is based on *libsvm* (Cao, 2003). *libsvm* is an integrated software for support vector classification, regression, and distribution estimation. It supports the multi-class classification. The appropriate time complexity is more than quadratic with the number of samples. Therefore, making it hard to scale to a dataset with more than a couple of 10000 samples (Cao, 2003). The multi-class support is handled based on a one-vs-one scheme. The following is the optimum estimator for the dataset.

Best parameters of Support Vector Machine for this data set: C=1, cache_size=200,

class_weight=None, coef0=0.0, _function_shape='ovr', degree=3, gamma=0.001, kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False.

TABLE 10. DESCRIPTIONS OF THE PARAMETERS PF THE SVM

Parameter	Description
c	Penalty parameter
Kernel	Specifies the kernel type to be used in the algorithm. It must be one of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘pre-computed’ or a callable. If none is given, ‘rbf’ will be used. If a callable is given, it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples).
Degree	The degree of the polynomial kernel function (‘poly’). Ignored by all other kernels.
Gamma	Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’. If gamma is ‘auto’ then $\frac{1}{n\_features}$ will be used instead.
Coef	Independent term in kernel function. It is only significant in ‘poly’ and ‘sigmoid’.
Probability-Whether	to enable probability estimates. This must be enabled before calling fit and will slow down that method.
Shrinking	Whether to use the shrinking heuristic.
Tol	Tolerance for stopping criterion.
cache_size	Specify the size of the kernel cache (in MB).
class_weight	Set the parameter C of class i to $class\_weight[i] * C$ for SVC. If not given, all classes are supposed to have weight one. The “balanced” mode uses the values of y to

	automatically adjust weights inversely proportional to class frequencies in the input data as $\frac{n_{samples}}{(n_{classes} * np.bincount(y))}$
decision_function_shape	Whether to return a one-vs-rest (‘ovr’) decision function of shape (n_samples, n_classes) as all other classifiers, or the original one-vs- one (‘ovo’) decision function of libsvm which has shape $\frac{(n\_samples, n\_classes * (n\_classes - 1)}{2}$.
max_iter	Hard limit on iterations within solver, or -1 for no limit.

To measure our SVM model performance we deployed 10-fold cross validation and displayed its micro and macro averages obtained from the area under ROC curves in the table below. It seems SVM is an efficient algorithm for regression as well as classification purpose. It draws a hyperplane to separate classes. This algorithm works extremely well with regression. The effect of SVM increases as we increase dimensional space. SVM also performs well when the dimension number is larger than the sample number (İşeri, et al. 2017). There exists a drawback too: it does not perform well on huge datasets. SVM extensively uses cross-validation to increase its computational efficiency. SVM tries to finds the “best” margin (distance between the line and the support vectors) that separates the classes, therefore it can have different decision boundaries with different weights that are near the optimal point. Studies show SVM works well with unstructured and semi-structured data like text and the risk of overfitting is less in SVM. (Verma, Jain and Prakash, 2020). Accordingly, we also observe that SVM performs the best on our dataset.

TABLE 11. MICRO - MACRO AVERAGE OF ROC CURVES TABLE FOR SVM

Classification Method	Micro Average	Macro Average	AUC Class 0 Average	AUC Class 1 Average	AUC Class 2 Average
SVM	0.9115	0.9325	0.9692	0.8419	0.987

4.5 Multi-Layer Perceptron Classifier

The fourth predictive model attempted for classification is the Multi-Layer Perceptron Classifier.

This is a class of feedforward artificial neural networks. It consists of at least three layers of nodes. Each node, except the input node, is a neuron that uses a nonlinear activation function. This model optimizes the log-loss function using LBFGS or stochastic gradient descent. The following parameters result in the most accurate predictive results and the missing parameters are considered to be so insignificant that their values are changed to their default values. MLP which is one of the most commonly used classifier, is a feed-forward, supervised neural network topology. Back propagation algorithm is used for minimizing the error between network output and the target value. According to classification process, MLP structure and learning parameters, which are used in back propagation algorithm, are needed to decide for increasing the test accuracy. Commonly these variables are chosen randomly, so finding the values that give maximum test accuracy is a time-consuming process. In this thesis, we used IBM SPSS Modeller to ameliorate and optimize the hyper-parameters.

Best parameters of MLP Classifier for this data set: activation='relu', alpha=0.0001,

batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(10,20), learning_rate='invscaling', learning_rate_init=0.001, max_iter=200, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=None, shuffle=True, solver='lbfgs', tol=0.0001, validation_fraction=0.1.

TABLE 12. DESCRIPTIONS OF THE PARAMETERS PF THE MLP CLASSIFIER

Parameter	Description
hidden_layer_sizes	The ith element represents the number of neurons in the ith hidden layer.
Activation	Activation function for the hidden layer: ‘identity’, no-op activation, useful to implement linear bottleneck, returns f(x) = x. ‘logistic’, the logistic sigmoid function, returns f(x) = 1 / (1 + exp(- x)). ‘tanh’, the hyperbolic tan function, returns f(x) = tanh(x). ‘relu’, the rectified linear unit function, returns f(x) = max(0, x) solver The solver for weight optimization: ‘lbfgs’ is an optimizer in the family of quasi-Newton methods. ‘sgd’ refers to stochastic gradient descent. ‘adam’ refers to a stochastic gradient-based optimizer alpha L2 penalty (regularization term) parameter.
batch_size	Size of mini-batches for stochastic optimizers. If the solver is ‘lbfgs’, the classifier will not use minibatch. When set to “auto”, batch_size=min (200, n_samples)
learning_rate	‘constant’ is a constant learning rate given by ‘learning_rate_init’. ‘invscaling’ gradually decreases the learning rate learning_rate_ at each time step ‘t’ using an inverse scaling exponent of ‘power_t’.
effective_learning_rate	= learning_rate_init / pow (t, power_t) ‘adaptive’ keeps the learning rate constant to ‘learning_rate_init’ as long as training loss keeps decreasing. Each time two consecutive

	epochs fail to decrease training loss by at least tol, or fail to increase validation score by at least tol if ‘early_stopping’ is on, the current learning rate is divided by 5.
learning_rate_init	The initial learning rate used. It controls the step-size in updating the weights. Only used when solver=’sgd’ or ‘adam’.
power_t	The exponent for inverse scaling learning rate. It is used in updating effective learning rate when the learning_rate is set to ‘invscaling’. Only used when solver=’sgd’. max_iter Maximum number of iterations. The solver iterates until convergence (determined by ‘tol’) or this number of iterations. For stochastic solvers (‘sgd’, ‘adam’), note that this determines the number of epochs (how many times each data point will be used), not the number of gradient steps.
shuffle	Whether to shuffle samples in each iteration. Only used when solver=’sgd’ or ‘adam’.
random_state	If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.
tol	Tolerance for the optimization. When the loss or score is not improving by at least tol for two consecutive iterations, unless learning_rate is set to ‘adaptive’, convergence is considered to be reached and training stops.

Momentum	Momentum for gradient descent update. Should be between 0 and 1. Only used when solver='sgd'.
nesterovs_momentum	Whether to use Nesterov's momentum. Only used when solver='sgd' and momentum > 0.
early_stopping	Whether to use early stopping to terminate training when validation score is not improving. If set to true, it will automatically set aside 10% of training data as validation and terminate training when validation score is not improving by at least tol for two consecutive epochs. Only effective when solver='sgd' or 'adam'
validation_fraction	The proportion of training data to set aside as validation set for early stopping. Must be between 0 and 1. Only used if early_stopping is True beta_1 Exponential decay rate for estimates of the first moment vector in adam, should be in [0, 1]. Only used when solver='adam' beta_2 Exponential decay rate for estimates of second moment vectors in adam, should be in [0, 1]. Only used when solver='adam' epsilon: float, optional, default 1e-8 Value for numerical stability in adam. Only used when solver='adam'

Micro and Macro average of 10-fold cross validation for our MLP model can be see in the table 13.

TABLE 13. MICRO - MACRO AVERAGE OF ROC CURVES TABLE FOR MLP

Classification Method	Micro Average	Macro Average	AUC Class 0 Average	AUC Class 1 Average	AUC Class 2 Average
MLP	0.3708	0.6223	0.5048	0.9414	0.9673

4.6 Decision Tree

The Decision Tree is the last predictive model experimented in this research. By its simplest description, decision tree analysis is a divide-and-conquer approach to classification (and regression). Decision trees can be used to discover features and extract patterns in large databases that are important for discrimination and predictive modeling. These characteristics, coupled with their intuitive interpretation, are some of the reasons decision trees have been used extensively for both exploratory data analysis and predictive modeling applications for more than two decades.

One advantage that decision tree modeling has over other pattern recognition techniques lies in the interpretability of the constructed model. Owing to this interpretability, information relating to the identification of important features and interclass relationships can be used to support future experiments design and data analysis. Microarray data analysis is an example of a biochemical application that is suitable for pattern recognition techniques such as decision trees where the emphasis of the analysis resides in the interpretation and the identification of important biological probes rather than the predictive accuracy of classification (at least initially).(Myles et al., 2004)

Best hyper parameters of decision tree Classifier for this data set: criterion = ”gini”, max_features='auto', min_impurity_split = None, min_samples_split = 2, min_samples_leaf = 1.

TABLE 14. DESCRIPTIONS OF THE PARAMETERS PF THE DECISION TREE

Parameter	Description
Criterion	The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.
max_features	The number of features to consider when looking for the best split: If int, then consider max_features features at each split. If float, then max_features is a percentage, and int(max_features × n_features) features are considered at each split. If “auto”, then $max_{features} = \sqrt{n\_features}$ if “sqrt”, then max_features can be calculated same as “auto”. If “log2”, then $max\_features = log\, log\, (n\_features)$. If None, then $max\_features = n\_features$.
min_impurity_split	Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise, it is a leaf.
min_samples_split	int, float, optional (default=2) The minimum number of samples required to split an internal node: If int, then consider min_samples_split as the minimum number. If float, then min_samples_split is a percentage and ceil ($min_{samples_{split}} \times n_{samples}$) are the minimum number of samples for each split.
min_samples_leaf	The minimum number of samples required to be at a leaf node: If int, then considers min_samples_leaf as the minimum number. If float, then min_samples_leaf is a

	percentage and ceil ($min\_samples\_leaf \times n\_samples$) are the minimum number of samples for each node.

Besides that, the area under ROC curve for our Decision Tree can be found in the table 15.

TABLE 15. MICRO - MACRO AVERAGE OF ROC CURVES TABLE FOR DECISION TREE

Classification Method	Micro Average	Macro Average	AUC Class 0 Average	AUC Class 1 Average	AUC Class 2 Average
Decision Tree	0.7451	0.7636	0.8275	0.7764	0.8896

To sum up our models' performance, mean accuracy of our models stratified by 10-fold can be seen in table 16, which was calculated from the Area under the ROC Curve.

TABLE 16. MODELS PERFORMANCE MEASUREMENTS

Classifier	10-fold Mean Accuracy
KNN	0.750
Decision Tree	0.757
Random Forest	0.765
MLP	0.512
SVM	0.921

However, we cannot trust the accuracies to compare the models; therefore, we need T-tests to discover if statistical significance is present in differences between the model classifiers.

4.7 T-Tests

T-tests are conducted to discover if statistical significance is present in differences between the model's classifiers. The mean of the difference between classifiers' accuracies is used in the t-test formulas. The ‘diff’ within the formulas represents the difference in 10-fold accuracies between the two classifiers stated. Using a confidence level of 95 percent and a degree of freedom of 9, the coefficient used from the probability table for t-distribution is 2.26 (Tan, Steinbach, & Kumar; 2015). This coefficient is multiplied by the standard deviation of the difference. The coefficient multiplied by the standard deviation is added and subtracted from the mean of the difference between classifiers resulting in a confidence interval used for interpreting statistical significance.

The Student’s t-Test is a statistical hypothesis test for testing whether two samples are expected to have been drawn from the same population. The test works by checking the means from two samples to see if they are significantly different from each other. It does this by calculating the standard error in the difference between means, which can be interpreted to see how likely the difference is, if the two samples have the same mean (the null hypothesis).

The t statistic calculated by the test can be interpreted by comparing it to critical values from the t-distribution. The critical value can be calculated using the degrees of freedom and a significance level with the percent point function (PPF).

We can interpret the statistic value in a two-tailed test, meaning that if we reject the null hypothesis, it could be because the first mean is smaller or greater than the second mean. To do this, we can calculate the absolute value of the, as follows:

$$d_t^{cv} = d \pm \ t_{(1-\alpha),k-1}\sigma_{d^{cv}}$$

In this equation, d is the mean, t is specified in the t-distribution table, and σ is the standard deviation. According to our models we will have the following formula:

$$CFI = Mean \pm 2.26 * STDV$$

If the range of confidence intervals spans zero, it means the difference between the two models is not statistically significant, i.e. the two models are performing the same and For example, here, Random Forest and Decision Tree have this situation. If the range of confidence intervals does not span zero, it means the difference between the two models is statistically significant, i.e. the two models are not performing the same. For example, Random Forest and Decision Tree have this situation.

TABLE 17. MEASURED T-TEST RANGES

	NN	KNN	RandomForest	DecisionTree	SVM
NN	■				
KNN	**(0.02 , 1.37)**	■			
RandomForest	**(0.01 , 1.35)**	**(-0.11,1.41)**	■		
Decision Tree	**(0.02 , 1.37)**	**(-0.007 , 1.2)**	**(-0.2 , 1.25)**	■	
SVM	**(0.02 , 1.39)**	**(0.23 , 1.21)**	**(0.17 , 1.16)**	**(0.02 , 1.33)**	■

Table 17 shows the measured T-test ranges, which shows making the difference between the accuracies of MLP classifiers and SVM in comparison with other classifiers statistically significant. However, we can not decide KNN performs better or decision tree or Random Forest. Therefore, we argue the 10-fold accuracy mean (calculated in the table 16) can be trusted.

4.8 Chapter Summary

This chapter presented five models building strategies. K-nearest neighbour, decision tree, MLP classifier, SVM, as well as random forest were chosen as the machine learning models. To determine the optimal parameters to use, a Grid Search was conducted for each model. Optimal parameters were found and used to run each classifier. T-tests were then performed to compare models determining statistical significance of differences between accuracies of the models. The accuracies, ROC curves of each of the ten folds were evaluated. These chapter served addressing the question which machine learning model perform the best in terms of accuracy and area under ROC (RQ2).

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Research Summary

It is argued that in the near future the machine learning methods will provide us with a promising future in learning and mastering stock market price change. We acknowledge that currently some may hold negative view on using machine learning to predict stock price. They argue stock market is too volatile to predict. Yet, scholars and big company traders from are working to explore more indicators and models to accurately forecast the economic market. However, in the last few decades, the efforts of various researchers all over the world have shown that the stock market is predictable. The only issue is that the existing prediction methods are not good enough. Thus, the stock market prediction is still one of the most challenging topics in this research area and has always been one of the most popular research topics in the world. (Patel, Quadros, Patil, Pawale, & Saxena, 2017)

This thesis provides background information and summarizes relevant trading theory and some important and relevant text mining and data mining techniques in the stock market as a necessary point for understanding the rest of the thesis. A review of relevant news based stock prediction systems is provided. Most of the prior researches used different classifiers that categorizes news articles depending on how they affect the price actions. In order to classify the news, they have to generate a labeled training set which differ based on their researches' strategy.

Most of the existing approaches to predict the stock distinguished the stock market by historic trading prices, which called technical methods. Based on this historical time series, they tried to find indications that suggest rising or dropping prices. Predicted stock prices based on technical data alone has proven to be insufficient. Recent studies have shown that there is a strong correlation between news articles related to a company and its stock price movements. In these studies, it is evident that financial news has a significant influence on the stock market. Correspondingly, the prediction of the financial market based on news releases has attracted more and more attention in recent years (Li et al., 2019). However, due to the complexity and uncertainty of the natural languages used, it is difficult to forecast the financial market correctly based on news releases. There is a vast amount of textual data that can be used as a new source of information on this challenge.

Different data sources can be used to predict market behavior. Some studies have used news articles, social media, message boards, and company's ad-hoc announcements to predict the stock market and others have used hybrid methods which consider both historical market prices and external sources of text data for prediction. The data source is critical and has a high impact on prediction performance. However, techniques that are applying to extract the information from this data is also important. That is the area where text mining and natural language processing (NLP) comes into play. Since textual data are unstructured and scattered from different sources, it is a field where natural language processing (NLP) techniques are frequently employed. NLP techniques are widely used to extract the most informative knowledge from the unstructured textual data. When the pre-processing phase completed and text transformed into several features with a numeric representation, machine learning algorithms can be involved. Each algorithm has its strengths and weaknesses, depending on the characteristics of the stock market datasets and situations.

This implies that among learning-based prediction methods, no one algorithm is always better than the other, and the method will perform well when the problem is identified and optimized successfully. The Support Vector Machine (SVM) is widely used across the articles. Note that some studies applied more than one machine learning technique. The other most used algorithms are Naive Bayes, Long Short-term Memory (LSTM), Neural Networks, Random Forest (RF), Regression models, K Nearest Neighbor, Decision Tree (DT), and Reinforcement Learning respectively.

In the practical part of this thesis, we implanted news-based forecasting system in order to predict the daily stock price direction. The prediction is designed based on the textual analysis of news articles' headlines. We have investigated five different agents that can possibly drive stock market movements: Support Vector Machine, Decision Tree, Random Forest, k-nearest neighbors and Multi-layer Perceptron classifier performed on financial stock trend prediction in our prediction phase.

For conducting the analysis, we used Apple Inc. stocks trading from NASDAQ to compare the accuracy rate yielded by various models. We collected headlines form the Intrinio. In the preprocessing phase, news headlines converted to vectors by deploying Doc2Vec and Word2vec, a new word embedding step in an attempt to improve headlines concertation. Our Doc2Vec and Word2Vec models trained on news datasets collected from Factiva. After labeling our dataset, since we faced disparity in three predefined classes: *Buy*, *Hold*, and *Sell*, we employed SMOTE balancing technique to level our classes. Following the data preprocessing phase, ML models parameters are optimized and experiment evaluation methods are described.

The thesis then goes on to describe the experiments that are performed. At the end, Cross-Validation and Ensemble methods were utilized to improve prediction performances in forecasting the daily movement direction of one of the most popular New York Stock Exchange indices (APPL). Evaluations showed SVM can have around 15% improvement of accuracy rate compare to other methods that showed 75 % accuracy in average. It has transpired that SVM did the best comparing to KNN, DT, RF and MLP via consulting a T-test. This may suggest that in this type of prediction, Support Vector Machine outstands among other methods.

5.2 Contributions

- The purpose of this study is to predict the stock market prices using text mining and machine learning on historical financial news.
- Financial news articles about the company Apple ™ were collected using Intrinio and Factiva API calls for the past four years. These two datasets are reliable and well-established in terms of financial news which are accessible through the University of Ottawa.
- Training a Word2Vec model using preprocessed (Tokenization, Stop Words Removal, Lemmatization, Converting to Lowercase) Financial/Business News corpus and headlines. (RQ1)
- Training Doc2Vec model on 1 GB Financial/Business News corpus and headlines in textual format. (RQ1)
- A trained Word2Vec and Doc2Vec was used on the news article to evaluate their semantics and convert each word into a vector of dimension of 100. (RQ1) Using

Doc2Vec as a recent numerical representative vector for each title is one of the nobilities of this thesis.

- Apple's closing stock prices was collected from Factiva dataset.
- The difference of the date the article was released close price from the closing price of the next two day was used to label each news article into three groups of Buy, Sell, and Hold.
- The dataset was preprocessed, and balanced to be used for building the machine learning models. The stratified 10 fold cross validation was used to split the data into training and testing records.
- Five machine learning models were built, namely, KNN, DT, RF, NN, and SVM.
- The models were evaluated using performance analysis (accuracies and the area under the ROC curve, and micro averages of each of the ten folds). (RQ2)
- A comparative study (T tests) was performed to compare models determining statistical significance of differences between accuracies of the models. Tests show that the Support Vector Machine model performs better. (RQ2)

Although we chose Apple Inc. stock as an example in this research, our Methodology is generic and can be applied on any other high-tech related stock prediction.

5.3 Future Works

Automatic news-based trading decision systems are a small but growing research field. Thus, there are many aspects that can be investigated further. With regard to system developed in this thesis the following issues can be considered to be the primary ones that should be investigated in future works:

1. Building correlation and relationship among related companies to account for their impacts on each other by generating dictionary based word featuring models.

2. The classifier method used in this thesis classifies, buy, hold, and sell articles. A possible improvement to this could be implementing a clustering method to see if this classes can be adjusted or new classes can be added to the study.

3. Prediction could be performed with intraday data. The system developed in this thesis is ready for evaluations with intraday data, but it needs more data to be performed. This could be done by using intraday data over a period.

4. The trading strategy could be improved. The strategy used in this thesis only acquire a stock for a fixed time duration then liquidates it. This strategy could be improved by using different windows sizes to observe lasting the impact of the news.

5. Conduction comparative studies against other types of labeling method. Compare other technical indicator (Stochastic oscillator, William, Relative Strength Index, etc.) to label the news. This technique could increase the accuracy of forecasting.

6. It would have been desired to evaluate the models on the unseen data. With limited number of data we had for Apple stock, we did not test the models on unseen data. However, as future work, we suggest to do back-testing and validating the models on unseen data.

7. We also suggest building a hybrid model combining the fundamental analysis with technical analysis to evaluate the prediction results.

REFERENCES

Arora, S., Liang, Y., & Ma, T. (2019). A simple but tough-to-beat baseline for sentence embeddings. Paper presented at 5th International Conference on Learning Representations, ICLR 2017, Toulon, France.

Baker, L. D., & McCallum, A. K. (1998). Distributional clustering of words for text classification. *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, 96–103. https://doi.org/10.1145/290941.290970

Bisoi, R., & Dash, P. K. (2014). A hybrid evolutionary dynamic neural network for stock market trend analysis and prediction using unscented Kalman filter. *Applied Soft Computing Journal*, *19*, 41–56. https://doi.org/10.1016/j.asoc.2014.01.039

Bogle, S. A., & Potter, W. D. (2019). Sentamal- A sentiment analysis machine learning stock predictive model. *Proceedings of the 2015 International Conference on Artificial Intelligence, ICAI 2015 - WORLDCOMP 2015*, 610–615.

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, *2*(1), 1–8. https://doi.org/10.1016/j.jocs.2010.12.007

Borovkova, S., & Tsiamas, I. (2019). An ensemble of LSTM neural networks for high-frequency stock market classification. *Journal of Forecasting*, *38*(6), 600–619. https://doi.org/10.1002/for.2585

Bowers, A. J., & Zhou, X. (2019). Receiver Operating Characteristic (ROC) Area Under the Curve (AUC): A Diagnostic Measure for Evaluating the Accuracy of Predictors of Education Outcomes. *Journal of Education for Students Placed at Risk*, *24*(1), 20–46. https://doi.org/10.1080/10824669.2018.1523734

Cao, L. (2003). Support vector machines experts for time series forecasting. *Neurocomputing*, *51*, 321–339. https://doi.org/10.1016/S0925-2312(02)00577-5

Chiong, R., Adam, M. T. P., Fan, Z., Lutz, B., Hu, Z., & Neumann, D. (2018). A sentiment analysis-based machine learning approach for financial market prediction via news disclosures. *GECCO 2018 Companion - Proceedings of the 2018 Genetic and Evolutionary Computation Conference Companion*, 278–279. https://doi.org/10.1145/3205651.3205682

Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015). Deep Learning for Event-Driven Stock

Prediction. *Journal of Scientific and Industrial Research*, *61*(9), 719–725.

Fama, E. F. (1970). American Finance Association Efficient Capital Markets : A Review of Theory and Empirical Work Author (s): Eugene F . Fama Source : The Journal of Finance , Vol . 25 , No . 2 , Papers and Proceedings of the Twenty- Eighth Annual Meeting of the American. *The Journal of Finance*, *25*(2), 383–417.

İ. İşeri, Ö. F. Atasoy and H. Alçiçek, "Sentiment classification of social media data for telecommunication companies in Turkey," 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, 2017, pp. 1015-1019

Gálvez, R. H., & Gravano, A. (2017). Assessing the usefulness of online message board mining in automatic stock prediction systems. *Journal of Computational Science*, *19*, 43–56. https://doi.org/10.1016/j.jocs.2017.01.001

Ghosh, S., Modak, S., & Mondal, A. C. (2014). A Model to Compute Degree of Polarity of Review Titles, *7*(1), 65–70.

Gorgulho, A., Neves, R., & Horta, N. (2011). Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition. *Expert Systems with Applications*, *38*(11), 14072–14085. https://doi.org/10.1016/j.eswa.2011.04.216

Jiao, P., & Walther, A. (2016). Social Media, News Media and the Stock Market. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2755933

Khadjeh Nassirtoussi, A., Aghabozorgi, S., Ying Wah, T., & Ngo, D. C. L. (2014). Text mining for market prediction: A systematic review. *Expert Systems with Applications*, *41*(16), 7653–7670. https://doi.org/10.1016/j.eswa.2014.06.009

Kraus, M., & Feuerriegel, S. (2017). Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, *104*, 38–48. https://doi.org/10.1016/j.dss.2017.10.001

Lau, J. H., & Baldwin, T. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation, 78–86. https://doi.org/10.18653/v1/w16-1609

Le, Q., & Mikolov, T. (2015). Distributed Representations of Sentences and Documents Quoc. *Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion*, *32*, 29–30. https://doi.org/10.1145/2740908.2742760

Li, Y., Jin, T., Xi, M., Liu, S., & Luo, Z. (2019). Massive Text Mining for Abnormal Market Trend Detection. *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 4135–4141. https://doi.org/10.1109/BigData.2018.8622450

Lien Minh, D., Sadeghi-Niaraki, A., Huy, H. D., Min, K., & Moon, H. (2018). Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. *IEEE Access*, *6*, 55392–55404. https://doi.org/10.1109/ACCESS.2018.2868970

Lo, A. (2005). The Journal of Arthroplasty. *The Journal of Arthroplasty*, *19*(8), A6. https://doi.org/10.1016/s0883-5403(04)00612-6

Long, W., Song, L., & Tian, Y. (2019). A new graphic kernel method of stock price trend prediction based on financial news semantic and structural similarity. *Expert Systems with Applications*, *118*, 411–424. https://doi.org/10.1016/j.eswa.2018.10.008

Longadge, R., Dongre, S., & Malik, L. (2013). Class Imbalance Problem in Data Mining: Review. *International Journal of Computer Science and Network (IJCSN)*, *2*(1), 227–231. https://doi.org/10.1016/j.ejim.2013.08.659

Madge, S. (2015). Predicting Stock Price Direction using Support Vector Machines. *Neurocomputing*, *55*, 307–319.

Manning, C. D., Schütze, H., & Weikurn, G. (2002). Foundations of Statistical Natural Language Processing. *SIGMOD Record*, *31*(3), 37–38. https://doi.org/10.1145/601858.601867

Merello, S., Picasso Ratto, A., Ma, Y., Oneto, L., & Cambria, E. (2019). Investigating timing and impact of news on the stock market. *IEEE International Conference on Data Mining Workshops, ICDMW*, *2018-Novem*, 1348–1354. https://doi.org/10.1109/ICDMW.2018.00191

Metghalchi, M., Chang, Y. H., & Marcucci, J. (2008). Is the Swedish stock market efficient? Evidence from some simple trading rules. *International Review of Financial Analysis*, *17*(3), 475–490. https://doi.org/10.1016/j.irfa.2007.05.001

Meyer, B., Bikdash, M., & Dai, X. (2017). Fine-grained financial news sentiment analysis. *Conference Proceedings - IEEE SOUTHEASTCON*, 1–8. https://doi.org/10.1109/SECON.2017.7925378

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2016). Distributed Representations ofWords and Phrases and their Compositionality. *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 1389–1399. https://doi.org/10.18653/v1/d16-1146

Mohan, S., Mullapudi, S., Sammeta, S., Vijayvergia, P., & Anastasiu, D. C. (2019). Stock price prediction using news sentiment analysis. *Proceedings - 5th IEEE International Conference on Big Data Service and Applications, BigDataService 2019, Workshop on*

Big Data in Water Resources, Environment, and Hydraulic Engineering and Workshop on Medical, Healthcare, Using Big Data Technologies, 205–208. https://doi.org/10.1109/BigDataService.2019.00035

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., & Brown, S. D. (2004). An introduction to decision tree modeling. *Journal of Chemometrics*, *18*(6), 275–285. https://doi.org/10.1002/cem.873

Oppenheimer, H. R., & Schlarbaum, G. G. (1981). AN EX ANTE TEST OF THE, *XVI*(3), 341–360.

Patel, S. D., Quadros, D., Patil, V., Pawale, M., & Saxena, H. (2017). www.ijemr.net Stock Prediction using Neural Networks. *International Journal of Engineering and Management Research*, *7*(2), 490–493. Retrieved from http://www.ijemr.net/DOC/StockPredictionUsingNeuralNetworks.PDF

Pongsena, W., Ditsayabut, P., Kerdprasop, N., & Kerdprasop, K. (2020). Deep Learning for Financial Time-Series Data Analytics: An Image Processing Based Approach. *International Journal of Machine Learning and Computing*, *10*(1), 51–56. https://doi.org/10.18178/ijmlc.2020.10.1.897

Pröllochs, N., Feuerriegel, S., & Neumann, D. (2016). Negation scope detection in sentiment analysis: Decision support for news-driven trading. *Decision Support Systems*, *88*, 67–75. https://doi.org/10.1016/j.dss.2016.05.009

Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, *42*(3), 203–231. https://doi.org/10.1023/A:1007601015854

Radinsky, K., Davidovich, S., & Markovitch, S. (2012). Learning causality for news events prediction. *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web*, 909–918. https://doi.org/10.1145/2187836.2187958

Schumaker, R., & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Trans. Inf. Syst.*, *27*. https://doi.org/10.1145/1462198.1462204

Shah, V. (2007). Machine Learning Techniques for Stock Prediction. *Talanta*, *71*(2), 676–682. https://doi.org/10.1016/j.talanta.2006.05.013

Sobhani, P., Viktor, H., & Matwin, S. (2015). Learning from imbalanced data using ensemble methods and cluster-based undersampling. *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, *8983*, 69–83. https://doi.org/10.1007/978-3-319-17876-9_5

Spackman, K. A. (1989). Signal Detection Theory: Valuable Tools for Evaluating Inductive Learning. In *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 160–163). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, *36*(2), 111–133. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x

Swets, J. A., Dawes, R. M., & Monahan, J. (2000). Better decisions through science. *Scientific American*, *283*(4), 82–87. https://doi.org/10.1038/scientificamerican1000-82

Ticknor, J. L. (2013). A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, *40*(14), 5501–5506. https://doi.org/10.1016/j.eswa.2013.04.013

Vargas, M. R., De Lima, B. S. L. P., & Evsukoff, A. G. (2017). Deep learning for stock market prediction from financial news articles. *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications, CIVEMSA 2017 - Proceedings*, 60–65. https://doi.org/10.1109/CIVEMSA.2017.7995302

Verma, S., Jain, K. and Prakash, C., 2020. An Unstructured To Structured Data Conversion Using Machine Learning Algorithm In Internet Of Things (Iot). Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020, Available at SSRN: https://ssrn.com/abstract=3563389 or http://dx.doi.org/10.2139/ssrn.3563389

Wang, F., Shieh, S. J., Havlin, S., & Stanley, H. E. (2009). Statistical analysis of the overnight and daytime return. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, *79*(5), 1–7. https://doi.org/10.1103/PhysRevE.79.056109

Wang, Z., Ho, S. B., & Lin, Z. (2019). Stock market prediction analysis by incorporating social and news opinion and sentiment. *IEEE International Conference on Data Mining Workshops, ICDMW*, *2018-Novem*, 1375–1380. https://doi.org/10.1109/ICDMW.2018.00195

Wei, L. Y., Chen, T. L., & Ho, T. H. (2011). A hybrid model based on adaptive-network-based fuzzy inference system to forecast Taiwan stock market. *Expert Systems with Applications*, *38*(11), 13625–13631. https://doi.org/10.1016/j.eswa.2011.04.127

Yang, W., B, S. G., Raza, A., Herbert, D., & Kang, B. (2018). *Stock Price Movement Prediction*. *15th Pacific Rim Knowledge Acquisition Workshop* (Vol. 11016). Springer International Publishing. https://doi.org/10.1007/978-3-319-97289-3

Zhang, Z., He, X., & Ge, J. (2018). News-oriented Stock Price Trend Prediction.

Appendix

The collected datasets and code implemented for this study can be found on GitHub at the following link:

GitHub Link