Mathias Grotøy Bjørgum
Aasmund Groven Lindtveit

Cleaning Up Intraday Noise

Investigating Whether Textual Data Enhances the Understanding of Intraday Stock Price Patterns

Master's thesis in Economics and Business Administration
Supervisor: Arild Brandrud Næss
May 2023

Mathias Grotøy Bjørgum
Aasmund Groven Lindtveit

Cleaning Up Intraday Noise

Investigating Whether Textual Data Enhances the Understanding of Intraday Stock Price Patterns

Master's thesis in Economics and Business Administration
Supervisor: Arild Brandrud Næss
May 2023

Norwegian University of Science and Technology
Faculty of Economics and Management
NTNU Business School

Abstract

Financial markets are complex, and stock data is noisy. After the introduction of the efficient market hypothesis, which states that the market reacts immediately to all available information, additional data sources have been used to predict the movement of stock prices. Financial news articles could be one of these. Several studies have explored the relationship between textual data and stock price movements on a day-to-day basis, this thesis explores the relationship on an intraday basis.

We explore the relationship between news articles published on the NASDAQ website and the directional movement of the stock price on the stocks mentioned in the articles. The goal of this thesis is not to beat the market, but rather to explore whether news articles can be shown to affect the stock price in a time frame of 20 minutes after its publication. Textual features are extracted from news articles and used alongside short-term historical stock price movements. These features are combined in a feed-forward neural network, random forests, XGBoost, and logistic regression to predict the holding period return (HPR) 20 minutes after an article has been published. The results are then compared with the same models, excluding textual features, to see whether the textual data matter or not.

The feature extraction from text is done by using BERT-based language models. We differentiate between the headline and the content of the articles and extract the same sort of features from both. The features extracted include sentiment classification, achieved by employing FinBERT, which is a BERT model specifically trained for analyzing sentiment in financial text. Additionally, we conduct text classification using a fine-tuned DistilBERT model. The DistilBERT model is tuned to classify whether the HPR 20 minutes after the publication of a news article went *up* or not. The results of the pure text classification are also discussed to see whether the text can help explain the stock market.

DistilBERT achieves a validation set accuracy of 76.23% by considering an *up* prediction as true when DistilBERT correctly predicts an *up* movement on either headlines or content. It is important to note that this accuracy is a theoretical result and assumes that a classifier utilizing the model's correct *up* predictions could achieve an accuracy of 76.23%. In the context of our DistilBERT models, the validation set represents out-of-sample predictions.

The results of our analysis are in alignment with the results of previous work. When used alongside historical stock prices, evidence suggests that textual data do not help predict HPR in a time frame of 20 minutes. However, pure text classification indicates that textual data can help explain the stock market. Our findings then suggest that intraday stock data is too noisy when predicting short-term stock price movements, but textual data could be an explanatory variable.

Sammendrag

Finansmarkeder er komplekse, og aksjedata inneholder mye støy. Etter introduksjonen av hypotesen om effisiente markeder, som sier at markedet reagerer umiddelbart på all tilgjengelig informasjon, har det blitt brukt flere datakilder enn rene aksjedata for å predikere aksjeprisbevegelser. En av disse datakildene er finansielle nyheter. Ettersom flere studier har utforsket forholdet mellom tekstdata og aksjeprisbevegelser på dagsbasis, søker vi å utforske forholdet på intradagsbasis.

Vi undersøker forholdet mellom artikler publisert på NASDAQ-nettstedet og bevegelsen til aksjeprisen på aksjene som nevnes i artiklene. Målet med denne avhandlingen er ikke å slå markedet, men heller å undersøke om nyhetsartikler kan påvirke aksjeprisen i et tidsvindu på 20 minutter etter publisering. Tekstvariabler genereres fra nyhetsartiklene, som brukes sammen med kortsiktige historiske aksjeprisbevegelser. Dette samles i et feed-forward nevralt nettverk, random forest, XGBoost og logistisk regresjon for å predikere avkastningen 20 minutter etter at en artikkel har blitt publisert. Resultatene sammenlignes deretter med de samme modellene uten tekstvariabler for å se om nyhetsartiklene har betydning eller ikke.

Tekstvariablene lages ved å bruke BERT-baserte språkmodeller. Vi skiller mellom overskrift og innhold i artiklene og lager de samme type variablene fra begge deler. Tekstvariablene som brukes er sentiment klassifikasjon hentet ved hjelp av FinBERT, en BERT-modell spesialtrent for å analysere sentimentet i finansiell tekst. I tillegg gjennomfører vi også en finjustert DistilBERT modell. Denne modellen er justert til å klassifisere om avkastningen til en aksje 20 minutter etter at en nyhetsartikkel ble publisert har gått opp eller ikke. Resultatene fra den rene tekstklassifikasjonen er også diskutert for å se om tekst kan hjelpe til å forklare aksjemarkedet.

DistilBERT oppnår en nøyaktighet på 76,23 % på valideringssettet når vi anser en opp-prediksjon som sann når DistilBERT har gjort en riktig opp-prediksjon på enten overskrift eller innhold. Det er viktig å merke seg at denne nøyaktigheten er et teoretisk resultat, og indikerer at en klassifikator som benytter modellenes riktige opp-prediksjoner kunne oppnådd en nøyaktighet på 76,23 %. I konteksten av våre DistilBERT modeller representerer valideringssettet usett data.

Resultatene fra analysen stemmer overens med resultater fra tidligere arbeid. Når tekstdata brukes sammen med historiske aksjepriser tyder evidens på at tekstdataen ikke hjelper med å predikere avkastning i et tidsvindu på 20 minutter. Den rene tekstklassifiseringen indikerer derimot at tekstdata kan bidra til å forklare deler av aksjemarkedet. Våre funn antyder dermed at aksjedata på intradagsbasis inneholder for mye støy, når det gjelder å predikere kortsiktige aksjeprisbevegelser, men at tekst kan være en forklaringsvariabel.

Preface

This master's thesis is the culmination of the authors' five-year journey to a Master of Science degree in Economics and Business Administration, with specialization in Business Analytics, at the Norwegian University of Science and Technology (NTNU), in the spring of 2023.

First and foremost, we thank our supervisor Arild Brandrud Næss for invaluable feedback and support in our project. Second, we also thank Sander Ruud and Ane Hjort-Larsen for invaluable discussions around the thesis and predictions about the stock market in genreal. We also thank Rayan Ayari for taking the time to contribute his insight within the field of stock predictions.

We also thank our families and friends for their support through our studies and for their tolerance when this has been the only topic of discussion for the last few months. Special thanks also go to Caroline and Sara, whom we have had the pleasure of sharing an office with for the duration of writing this thesis.

The authors take full responsibility for the content of this thesis. Interested parties can direct an email to either of the authors to request access to our codebase.

Trondheim, May 2023

[illegible]
Mathias Grotøy Bjørgum
`mathias.bjorgum@gmail.com`

[illegible]
Aasmund Groven Lindtveit
`aagl@online.no`

This page intentionally left blank.

Contents

List of Figures

List of Tables

List of Abbreviations

BERT	Bi-directional Encoder Representations from Transformers.
EMH	efficient market hypothesis.
FNN	feed-forward neural network.
FPR	false positive rate.
HPR	holding-period return.
ML	machine learning.
NLP	natural language processing.
NN	neural network.
NND	NASDAQ news dataset.
RSD	Refinitiv stock dataset.
TPR	true positive rate.

Note Abbreviations only used within one chapter are omitted from this list.

1 Introduction

Stock data is inherently noisy data. It is noisy on a daily basis and even more on an intraday basis. Fama (1965) formulated the efficient market hypothesis (EMH), which states that the market reacts immediately to all available information. If this holds, it must imply that the noise in stock return data is explained in variables. By finding these variables, we can expand our understanding of the stock market. Techniques such as fundamental and technical analysis have come a long way but are still not perfect.

News articles have been identified as a potentially valuable variable in explaining stock market movements. Each day, many news articles are published in various sources, such as Twitter, Reddit, and other web-based news sources. These sources of textual data contain valuable information that can influence stock market movements. Previous studies have found various amounts of correlations between news articles and stock movements, but the relationship remains complex and is not yet fully understood.

In recent years, natural language processing (NLP) has made massive gains with language models, such as Bi-directional Encoder Representations from Transformers (BERT) and GPT, built on the transformer architecture introduced by Vaswani et al. (2017). These models have achieved unprecedented results on many NLP tasks and have made researchers think about how the techniques can be applied in finance. News articles could be a valuable variable in clearing up some of the noise in stock data.

There is little doubt that solving the puzzle of intraday returns and news articles could create massive financial gains. It is, however, also an interesting correlation to find to better understand the market. If it is so that news articles drive stock movements, malicious actors could drive the market in whatever direction they desire based on what articles they write. This also raises the question of whether the market shapes news or whether that news shapes the market. It also sparks a discussion around the performative characteristics of economical models as mentioned by Callon (2007). We touch on this subject, but it is a bit beyond the scope of this thesis.

Willding et al. (2018) have published a report on the impact of digital platforms and journalistic content. In this report, they discuss how the migration of news to digital channels has caused a shift from mass communication to more personalized and customized news consumption. They state that the capacity to personalize and customize news consumption has been made possible by the growth of online news access. This suggests that, with the Internet, the news is not being published for a wide audience but is more tailored toward specific audiences. Adding the argument that malicious actors could drive the market in whatever direction they desire by tailoring news for specific audiences, it is more important than ever to understand how news articles and stock price movements relate to each other.

In their report, Willding et al. (2018) further suggests that artificial intelligence (AI) is being used to generate summaries and headlines for news articles. This indicates that the headlines in modern news may be outputs from language models, and the content is the part that is human written. How AI-generated headlines influence stock prices is interesting on its own and it is also important to understand how these headlines influence the short-term movement of stock prices.

Adding to the argument on how malicious actors could influence the market is an analysis of the impact of behavioral finance of fake news on financial markets by Fong (2021). The results of this article suggest that stock prices overreact to fake news and underreact to real news. They also suggest that fake news in a security amplifies underreactions to subsequent real news for the security. These suggestions are based on the importance of understanding the relationship between news and stock price movements.

With the rise of novel NLP techniques and models, such as transformers, this day and age is an interesting time to study the relationship between stock movements and news articles. Transformer-based models could be the missing link to a better understanding of what drives stock movements.

The field where economics/finance and data science intersect is an interesting field of study and a field with a lot left unknown. Traditional economists have been reluctant to take new methods to heart, especially new models that are difficult to understand and interpret. For instance, traditional econometrics is largely based on linear regressions. That being said, regression works in many contexts, but not in all. This is where newer and more advanced machine learning (ML) methods and neural networks (NNs) come into play, and this is what we find motivating in this area of research.

A large part of our motivation for writing this thesis includes the desire to contribute to the literature in the field. We have noticed that meaningful research is somewhat lacking and the research that has been conducted seems to hold back on some of their results.

1.1 Research Questions

This thesis explores the potential of NLP in finance. Specifically, we explore how NLP can enhance the understanding of intraday stock price patterns. To investigate the relationship between stock prices and financial news, we have formulated the following research questions:

RQ 1: Does textual data help predict the direction of stock returns on intraday data?

RQ 2: Can a machine learning model that includes textual features predict the direction of a stock price movement 20 minutes after the news has been published?

RQ 3: Does textual data clear up some of the noise in stock data?

The difference between the first and third research questions is that the third is more of a conceptual question. From the EMH, we know that stock prices, in theory, *should* reflect all available information, and the third question is tailored to spark a discussion of whether news data actually help clear up some of the noise.

We emphasize that the goal of our thesis is not to beat the market, but rather to understand what drives it. The research questions are designed to investigate whether news articles actually contain information that can explain short-term price movements of stocks.

1.2 Thesis Structure

In the next chapter, Chapter 2, we present the theoretical background required to understand both the significance of this thesis and previous work in the field. This includes a description of stocks and how stock exchanges work, and an introduction to NLP, with a focus on the models used. Furthermore, we introduce how the evaluation of different classifiers is conducted before we describe the NNs and other ML classifiers used in this thesis.

Chapter 3 presents the previous work in the field. This includes previous work regarding both interday and intraday predictions, with and without textual features. We note that this chapter is rather short as there is a lack of meaningful work in the field.

In Chapter 4 we start by describing our different data sets and the preprocessing that was required to perform analyses on them. We then perform an exploratory data analysis with respect to our target variable and the feature variables. Rounding up the chapter, we present our research design.

To extract features from our texts, we use two BERT-based models, FinBERT and DistilBERT. These models, with related methodology, are presented in Chapter 5. We also evaluate how our fine-tuned DistilBERT model performs on training and validation data, as well as fine-tuning a model on a single stock.

In Chapters 6 and 7 we present the methodology and experimental results related to the predictions. This is because each model has its own methodology. At the end of each of those chapters, we also discuss our findings. We start by explaining how NNs have been used in Chapter 6. Here we run multiple tests with different network typologies and evaluate them. Chapter 7 describes how other ML classifiers have been used. We describe how the hyperparameters were set up and also evaluate different prediction thresholds.

Chapter 8 contains the discussion of the overall results of our experiments. This discussion includes the significance of our findings and a comparison of our different results. The chapter ends with a discussion around real-world applications and ethical considerations.

Finally, Chapter 9 concludes our thesis. We answer our three research questions and make some suggestions for further work.

1.3 Contributions

The goal of this thesis is to expand our understanding of the stock market. The field of using news articles to predict intraday directional movements on stock prices is largely new. As a consequence of this, we found it difficult to locate suitable existing frameworks to build our research on. We hope that this thesis contributes to the field by building a foundation and framework that can be further researched.

Another contribution of this thesis, which we have not found in previous work, is the fact that we have investigated if we can predict a *significant* rise in stock price. We have defined this significance as greater than 0.001. This increases the possibility of using the models for trading, as it also considers transaction costs.

Unlike much of the previous work in the field, this thesis presents all relevant metrics from our models. The hope is that, by doing this, future researchers can learn from what we have done. In this way, this thesis can serve as a basis for further research.

The codebase used in this thesis is also a contribution to future researchers. If they use similar news and stock data sets, they can, with some small tweaks, explore similar relationships. Expanding the prediction time frame and incorporating additional stocks mentioned in the news data set is a straightforward process.

2 Theoretical Background

This chapter introduces the theoretical background required to understand the foundation of this thesis. The presented theory includes basic theory about the stock market, an introduction to NLP, what a classification problem is, and how to measure the performance of classification models. In the later sections of this chapter, we introduce and describe NNs and other ML methods.

2.1 Financial Markets

This section explains the relevant theory concerning the financial market with regard to stocks. The theory includes how to perform trades, the NASDAQ stock exchange, and some economic theories regarding stock price movements. Analysis strategies and some considerations concerning investors' risk are also introduced. Information on the stock market is fundamental to understanding how NLP can be applied to further understand the stock market.

Stocks and the holding period return "Common stocks, also known as equity securities or equities, represent ownership shares in a corporation" (Bodie et al., 2021). There are several ways to earn money when investing in stocks, but the key aspect is to have a positive return on investment. The return is the amount of profit on a share over a period of time. We define this as holding-period return (HPR) and calculate it as

$$\text{HPR} = \frac{\text{Ending price of a share} - \text{Beginning price} + \text{Cash dividend}}{\text{Beginning price}}. \tag{2.1}$$

We note that there is no cash dividend to be considered in the data we use in this thesis.

Another aspect of stocks is volatility. Volatility is considered the standard deviation of the HPR. It models how much the predicted HPR will deviate from the actual return. However, volatility is not considered in this thesis.

Purchase and short sales of stocks In traditional stock trading, an investor will first buy a stock and then later sell it. With a short sale, this order is reversed, and the stock is first sold, then bought. The investor first borrows a share of stock from a broker and sells it with the promise of buying the stock back to the broker at a later time. If the stock then goes down in price, the investor gains a profit. It is important to note that short-selling stocks have no limit on how much money the investor can lose. Table 2.1 shows the profit calculation of purchases and short sales. We also note that not all investors have the possibility to short-sell.

Table 2.1: Overview of profit from the purchase and short sale of stocks. IP is the initial price, EP is the ending price and D is the dividend.

	Purchase		Short Sale	
Time	Action	Cash Flow	Action	Cash Flow
0	Buy share	− IP	Sell share	+ IP
1	Sell share	EP + D	Buy share	−(EP + D)
Profit	(EP + D) − IP		IP − (EP + D)	

Volume in stock markets As well as the open, close, high, and low price of a stock, *volume* is also relevant. This is the number of stocks traded over a given period of time.

NASDAQ Stock Exchange This is a particular stock exchange located in New York (US). NASDAQ computes a composite index of more than 3,000 companies traded on the NASDAQ market/exchange. It started as an over-the-counter dealer market, but over time, it has become a primarily electronic market (Bodie et al., 2021). The exchange has opening hours Monday through Friday from 9:30 am to 4:00 pm Eastern Standard Time (GMT−05:00)[1]. In UTC this is 14:30–21:00[2].

The NASDAQ Stock Exchange has extended hours of trading. The pre-trading session is from 4:00 am to 9:30 am. The post-trade session is from 4:00 pm to 8:00 pm. We note that the New York Stock Exchange has the same trading hours.

Trading costs There are costs to consider when trading. These costs are transaction costs and represent the broker's commission. There are two different types of brokers that traders can choose from, namely full-service brokers and discount brokers. A full-service broker typically relies on a research staff that prepares, analyzes, and forecasts general economic conditions, as well as industry and company trends. They often provide specific buy or sell recommendations based on their analysis. These brokers can also make trades on behalf of the investor (Bodie et al., 2021).

On the other hand, discount brokers provide "no frills" services. They carry out buy and sell orders, offer margin loans, and facilitate short sales. Compared to full-service brokers, discount brokers demand a lower commission (Bodie et al., 2021). The discount broker asks for a fee when placing orders, and we can consider this to be the transaction cost when trading.

[1]GMT−04:00 during daylight savings time.
[2]13:30–20:00 during daylight savings time.

Random walks and efficient markets Stock prices seem to follow a random walk. A random walk means that something is unpredictable. Changes in stock price can be surprising and, therefore, also hard to predict using statistical models. However, even though the argument that stock prices follow random walks is strong, they also respond swiftly to new information. Another theory to consider is the EMH. It states that markets are efficient and that stock prices reflect all available information relating to the stock. If the EMH and the random walk theory both hold, then predicting future stock price movements using historical price movements should be impossible (Bodie et al., 2021).

Technical and fundamental analysis A technical analysis is when investors use historical prices and other technical indicators to predict movements in future stock prices. Investors look for patterns in historical data to predict how the stock price will move in the future. The fundamental analysis is when investors use more available information. This information can include the prospects for the earnings and dividends of firms, the expectations for the future, and interest rates. Sources of this kind of information can be stock market announcements, tweets, and news articles. Fundamental analysis represents an attempt to determine the present value of all payments that a stockholder will receive from each share of stock (Bodie et al., 2021).

Considering investor's appetite for risk Investors handle risk differently when investing. Some investors seek risk, and others are risk-averse. Their risk appetite can be explained as a utility function as

$$U = E(r) - \frac{1}{2}A\sigma^2, \tag{2.2}$$

where A is a positive integer referencing the investor's risk aversion and a higher value means a higher risk aversion. U is the utility score from the equation and the investor seeks to maximize this number. σ^2 is the variance of returns and $E(r)$ is the expected return on the investment (Bodie et al., 2021). It is most common that investors are risk averse.

2.2 Natural Language Processing

This section explains the NLP methods applied to textual data, as well as explaining how NLP is valuable when trying to understand the market. In general, NLP is a branch of computer science in which a computer is used to model and analyze human-written text. Human-written text is a form of noisy data and is available almost everywhere. Examples of programs that use NLP are e-mail spam filters and chatbots. Before a computer can understand the textual data, the data must be transformed into numbers.

2.2.1 NLP in Finance

Since financial news articles contain a lot of information about the market, financial news should also influence the market. News can affect stock prices through a variety of means (Basilone, 2021). What makes NLP interesting in the field of finance boils down to the EMH. As mentioned earlier in this chapter, the EMH states that the market reacts immediately to all available information. In addition to reacting immediately to all available information, the market also reflects all available information (Bodie et al., 2021). Since financial news is a source of information concerning stocks, it should in theory help explain the market.

2.2.2 Word Embeddings and Tokenizers

The process of transforming human-written text into numbers is called embedding. Generally speaking, the goal of word embedding is mapping words in unlabeled text data to a continuously valued low-dimension space to capture internal semantic and syntactic information (Li and T. Yang, 2018). The essence of word embeddings is to represent words as vectors of numbers.

The NLP techniques used in this thesis include specific tools for embedding and tokenization. Tokenization is the process of splitting sentences into words. The words can then be embedded using an embedding algorithm. The BERT tokenizer performs the embedding and tokenization of the text before it is input into the model. We describe the BERT tokenizer in Section 2.4.6.

2.2.3 Text Classification

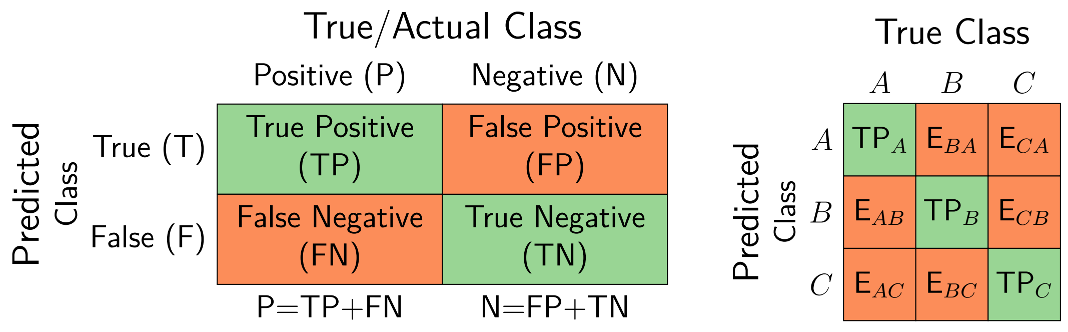
Text classification is a fundamental task in NLP, aimed at automatically assigning categories or labels to textual data. It is comparable to classification tasks where the input is numbers. The premise for classification is that, given a categorical target variable, a model learns patterns that exist between instances composed of independent variables and their relationship to the target (Bengfort, 2018). Within the domain of NLP, some classification tasks are more common than others. One of the most common tasks is sentiment analysis.

Sentiment analysis Sentiment analysis or opinion mining refers to the application of natural language processing, computational linguistics, and text analytics to identify and extract subjective information from source materials. Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document (Hovy, 2015). It is a form of specified text classification. A model trained to perform a sentiment analysis can return a sentiment score, for example, a continuous number from -1 (negative) to 1 (positive). It can also return a sentiment class or label. This label can be extracted from the sentiment score and be placed in 2, 3, or 5 different categories. These categories symbolize the sentiment whether it is negative, somewhat negative, neutral, somewhat positive, or positive. In the three-class method, we consider positive, neutral, and negative sentiments. In a financial context, the sentiment may be used to predict investors' behavior and from this help with predicting directional stock price movement.

2.3 Evaluation of Classifiers

Within ML, classification problems are problems in which the model predicts a categorical variable based on input variables. When evaluating different classification algorithms, some common metrics are calculated from a confusion matrix. The examples in this section are based on Tharwat (2020).

Confusion matrix The confusion matrix, also called *Contingency Table*, was introduced by Pearson (1904). The matrix displays how many observations the model predicts in a class plotted against the actual class of the observations. Figure 2.1 displays both a $2 \cdot 2$ and a $3 \cdot 3$ matrix.



(a) $2 \cdot 2$ confusion matrix. There are two true classes P and N. The output of the predicted class is true or false.

(b) $3 \cdot 3$ confusion matrix. There are three true classes A, B and C.

Figure 2.1: An illustrative example of a $2 \cdot 2$ and a $3 \cdot 3$ confusion matrix. The green diagonal represents correct predictions and the orange squares represents the incorrect predictions.

The *false negative (FN)* displayed in Figure 2.1a is also known as *Type II error*, and the *false positive* is also called *false alarm* or *Type I error*.

2.3.1 Classification Metrics

To compare the performance of different models, classification metrics are used. The formulas presented here target the $2 \cdot 2$ confusion matrix, but they can also be applied to multiclass problems with some tweaks to the formulas. The most common performance metrics are described here.

Accuracy (Acc) and error rate (ERR) Accuracy is one of the most commonly used measures for classification performance. It is defined as the ratio between the correctly classified samples to the total number of samples (Sokolova et al., 2006). ERR is the complement of accuracy. It is also known as *misclassification rate*. The metric represents the number of misclassified samples and is defined as $1 - \text{Acc}$. The formula for accuracy is

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.3}$$

Accuracy is a measure of how many correct predictions a model makes, and it does not take into consideration imbalanced datasets or differentiate between classes.

True positive rate and false positive rate True positive rate (TPR), also called *Sensitivity, hit rate*, or *recall*, represents the number of correctly classified positive samples of the total number of positive samples. TPR can be viewed as the accuracy for a given class. TPR and false positive rate (FPR) are calculated as

$$\text{TPR} = \text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{P}, \tag{2.4}$$

$$\text{FPR} = \frac{FP}{FP + TN} = \frac{FP}{N}. \tag{2.5}$$

The FPR represents the observations misclassified as positive.

Positive prediction value (PPV) PPV, also called *precision*, is the number of correctly classified positive samples to the total number of predicted positive samples. As in TPR, the precision is calculated for each class as

$$\text{PPV} = \text{Precision} = \frac{TP}{FP + TP}. \tag{2.6}$$

***F*-measure** This metric represents the harmonic mean of precision (PPV) and recall (TPR). The value ranges from zero to one, where a higher number indicates better performance. The metric is often called the F_1 *score*. The F_1 *score* is extracted from the F_β *score*. The β weights precision and recall differently, where a lower β places more weight on precision and less on recall, while a higher β places more weight on recall and less precision. The F_β *score* is calculated as

$$F_\beta = \frac{(1 + \beta^2) \cdot PPV \cdot TPR}{\beta^2 \cdot PPV + TPR}. \tag{2.7}$$

The F_1 *score* is the F_β *score* with a $\beta = 1$.

ROC Curve and AUC The final metric that we include here is the ROC (Receiver Operating Characteristic) curve. An illustration of the ROC curve and AUC is displayed in Figure 2.2. Starting from the lower left, the ROC curve plots the pairs {TPR, FPR} as the cutoff value decreases from 1 to 0. Better curves are reflected by curves that are closer to the top-left corner. The orange diagonal represents the average performance of a guessing classifier that has no information about the predictors or the outcome variable. This guessing classifier guesses that a proportion α of the records is 1's and therefore assigns each record an equal probability $P(Y = 1) = \alpha$. A common metric to summarize a ROC curve is *area under the curve (AUC)*, which ranges from 1 to 0.5, where 1 is perfect discrimination between classes and 0.5 is no better than random guessing (Shmueli et al., 2020).

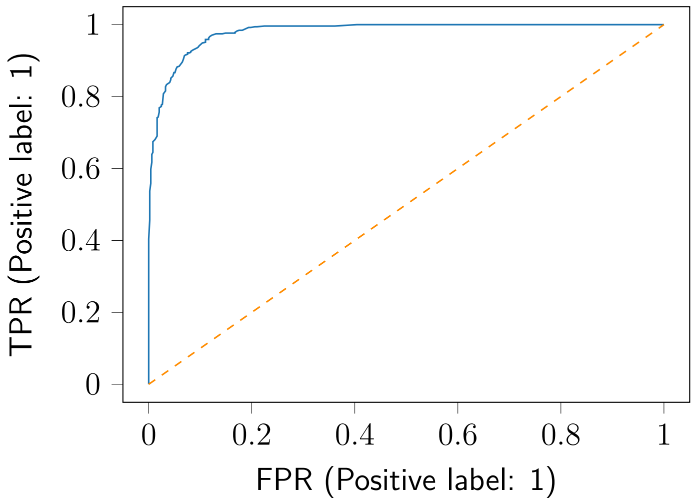


Figure 2.2: Illustration of a ROC curve. The AUC here is 0.98

2.3.2 Cross-Validation

Cross-validation (CV) is a technique in which we use some of the data in the full dataset to tune hyperparameters. There are multiple ways to do this, you could have a separate validation set to tune the hyperparameters, or you could use *k-fold cross-validation*. This method divides the set of observations into k-folds (groups). Then, one of the folds is used as the validation set, and the method is adapted to the remaining $(k - 1)$ folds (James et al., 2021).

Grid search This is a cross-validation form in which we cycle through all combinations of parameters. The general algorithm is shown in Algorithm 1. We loop through and calculate the score for each combination of hyperparameters.

Algorithm 1 Grid Search Cross-Validation

```
1: procedure GridSearchCV(data, hyperparameter_grid, k)
2:     best_params ← None
3:     best_score ← $-\infty$
4:     for params in hyperparameter_grid do
5:         scores ← []
6:         for fold in k-fold cross-validation(data, k) do
7:             train, val ← split(data, fold)
8:             model ← train_model(train, params)
9:             score ← evaluate(model, val)
10:            append(scores, score)
11:        end for
12:        avg_score ← mean(scores)
13:        if avg_score > best_score then
14:            best_params ← params
15:            best_score ← avg_score
16:        end if
17:    end for
18:    return best_params
19: end procedure
```

2.3.3 McNemar's Test

McNemar's test, introduced by McNemar (1947), is a test to determine if there is a significant difference between the proportions of two related binomial distributions. The following description is based onKavzoglu (2017). The test is also a well-known test to analyze the statistical significance of differences in classifier performance (Dietterich, 1998). The test is a Chi-square (χ^2) test for goodness of fit comparing the distribution of counts expected under the null hypothesis with the observed counts (Kavzoglu and Colkesen, 2013). The test is applied to a $2 \cdot 2$ confusion matrix. The matrix includes the number of samples correctly and incorrectly identified by both methods and the number of samples correctly classified by only one method.

The test statistic with continuity correction is estimated from the following equation with 1 degree of freedom as

$$\chi^2 = \frac{(|n_{ij} - n_{ji}| - 1)^2}{n_{ij} + n_{ji}}, \tag{2.8}$$

where n_{ij} indicates the number of observations misclassified by method i but classified correctly by method j, and n_{ij} indicates the number of observations misclassified by method j but not by method i. This value is then compared to the desired significance level in the χ^2 table.

2.4 Neural Networks

Since stock data is inherently noisy data, we need to use a model that has the ability to handle noisy data. Artificial neural networks (ANN) are machines designed to perform specific tasks by imitating how the human brain works, and build a neural network made up of hundreds or even thousands of artificial neurons or processing units (Montesinos López et al., 2022). The power of the human brain is superior to many information-processing systems since it can perform highly complex, non-linear, and parallel processing by organizing its structural constituents (neurons) to perform tasks such as accurate predictions, pattern recognition, perception, and motor control (Montesinos López et al., 2022). In this section, the basics of how a feed-forward neural network (FNN) works are described. This includes activation and loss functions, as well as some limitations of NNs.

2.4.1 Activation Functions

The goal of activation functions is to capture non-linear relationships in data. In NNs, they are used between layers to capture these non-linear relationships. Activation functions have different shapes and are used to capture different relationships. It is the shape of the activation function that decides what non-linear relationship it will capture. Some of the most commonly used activation functions are briefly explained in this subsection.

The Sigmoid

There are several sigmoid activation functions. In this thesis, we focus on the logistic sigmoid and the hyperbolic tangent function. The common denominator of sigmoid functions is that they are monotonically increasing functions that asymptotically increase at some finite value as infinity approaches (LeCun et al., 2012).

The logistic sigmoid This is the standard logistic function. It is a function formulated as

$$g(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}. \tag{2.9}$$

The standard logistic function is one of the most common sigmoid activation functions. It converts a value, z into a value between 0 and 1. This return can be interpreted as a probability, and it is a commonly used function in the output layer of a NN to perform a binary classification problem. In some cases, it is also used as an activation function between layers in regular FNNs.

The hyperbolic tangent A different variant of the logistic sigmoid function is the hyperbolic tangent function, tanh, formulated as

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \tag{2.10}$$

The tanh returns a number between −1 and 1, not 0 and 1 as the logistic function returns. Symmetric activation functions, such as the tanh, are preferred for the same reason that the inputs should be normalized. They produce outputs that are on average close to zero, in contrast to the logistic function whose outputs are always positive, and so they must have a mean that is positive (LeCun et al., 2012).

Other Activation Functions

There exist other activation functions that do not undergo the sigmoid family. Some of these functions are introduced here.

ReLU (Rectified Linear Unit) This activation function is an alternative to the functions in the sigmoid family. The function is stated as

$$g(z) = (z)_+ = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise} \end{cases}, \tag{2.11}$$

meaning that the function returns 0 if the input z is less than or equal to 0. If the z is greater than 0 the output is equal to the input.

Leaky ReLU A variant of ReLU is the leaky ReLU. It has a small slope, for instance, 0.01, where $z < 0$ to return a small negative number if the input is lower than 0. The leaky ReLU is stated as

$$g(z) = (z)_+ = \begin{cases} 0.1z & \text{if } z < 0 \\ z & \text{otherwise} \end{cases}. \tag{2.12}$$

Leaky ReLU can be applied if the user is struggling with the dying ReLU problem. The dying ReLU problem exists when the input into the ReLU is always lower than 0. If this is the case, using a regular ReLU will only return 0 throughout the network. Having a slight slope when $z < 0$ may help solve the dying ReLU problem by allowing the ReLU to return small negative numbers as well.

Softmax The last activation function discussed here is the softmax function. It was introduced by Bridle (1989), and is (for $b = e^{-\beta}$) defined as

$$\sigma(\mathbf{z})_i = \frac{e^{-\beta z_i}}{\sum_{j=1}^{K} e^{-\beta z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K. \tag{2.13}$$

The softmax can, as the logistic function, be used as the activation function in the output layer in a NN. This function is used mainly when there is a multiclass classification problem. It produces a probability for each class and the sum of these probabilities adds up to 1. When used for prediction, it predicts the class with the highest probability of being true.

The different activation functions are displayed in Figure 2.3.

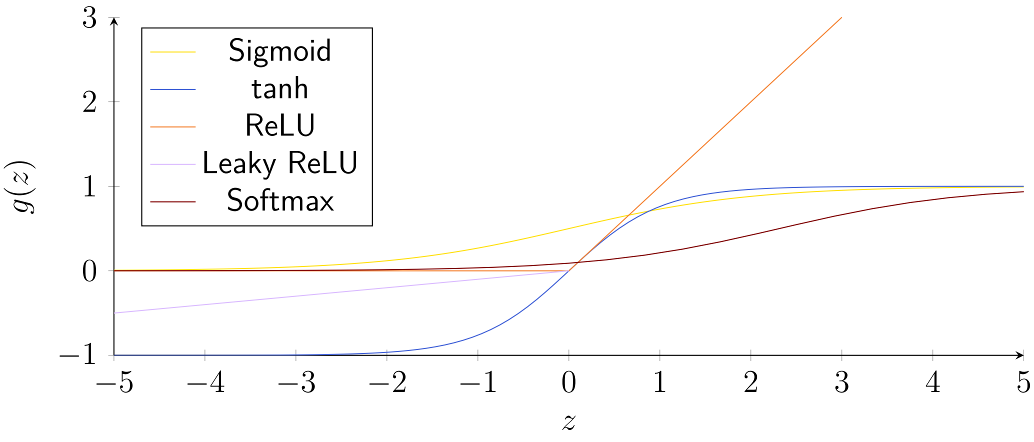


Figure 2.3: Illustration of different activation functions. Note that the leaky ReLU is equivalent to the normal ReLU when $z > 0$.

2.4.2 Loss Functions

A loss function is a function that compares the target and the predicted output values. In NNs, it measures how well the model matches the training data. When training, our goal is to minimize this loss between the predicted and target outputs. In classification problems, we use loss functions that given an input, the NN produces a vector of probabilities of the input belonging to pre-set categories, and then selects the category with the highest probability. An important attribute of loss functions is that they are differentiable.

Cross-entropy loss This is a loss function commonly used in classification problems. A binary cross-entropy loss is defined as

$$-\sum_{j} \mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}, \tag{2.14}$$

where $\mathbf{y}$ is the true label as a one-hot encoding, $\mathbf{o}$ is the output of the last layer of the network, $\cdot^{(j)}$ denotes the jth dimension of a given vector and $\sigma(\cdot)$ denotes the probability estimate (Janocha and Czarnecki, 2017).

2.4.3 Optimizers and Learning Rate

When optimizing a neural network, backpropagation with gradient-based learning is mostly used to update the parameters in each layer. Gradient-based learning is also known as an optimizer. There are different optimizers that are used in deep learning. They are used to optimize network parameters during backpropagation (UFLDL Tutorial, 2023).

Stochastic gradient descent The stochastic gradient descent (SGD) algorithm is inspired by the gradient descent algorithm. Gradient descent uses the following equation, in which it updates the parameters θ of the target $J(\theta)$ as

$$\theta = \theta - \alpha \nabla_{\theta} E\left[J(\theta)\right], \tag{2.15}$$

where the expectation in the above equation is approximated by evaluating the cost and gradient over the full training set.

SGD however, is simpler because it computes the gradient of the parameters using fewer examples from the training set. The formula of SGD is given as

$$\theta = \theta - \alpha \nabla_{\theta} J\left(\theta; x^{(i)}, y^{(i)}\right), \tag{2.16}$$

where $x^{(i)}, y^{(i)}$ is a pair of observations from the training set. θ is subtracted by ∇ multiplied by the learning rate α. α defines how large the steps each iteration of the algorithm will take. If the learning rate is set too high there is a risk that the algorithm diverges from the optimum. Setting the learning rate too low there is a risk that the SGD does not reach the global minimum, but a local minimum. The parameter has reached its optimal value when SGD has reached the global minimum of the derivative function of the loss with subject to the parameter it wants to optimize. Overall, this process has the goal of minimizing the total loss of the entire network.

2.4.4 Feed-Forward Neural Network

This network takes an input vector of p variables $X = (X_1, X_2, \cdots, X_p)$, and then builds a non-linear function $f(X)$ to predict the response. This description is based on James et al. (2021). Figure 2.4 displays a simple FNN with $p = 3$ predictors.

The predictors make up the *input layer*. All nodes in the input layer are connected to all nodes in the *hidden layer*. The figure has $K = 5$ nodes in the first hidden layer, note that this number can increase or decrease. The network has form as

$$\begin{aligned} f(X) &= \beta_0 + \sum_{k=1}^{k} \beta_k h_k(X) \\ &= \beta_0 + \sum_{k=1}^{k} \beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right). \end{aligned} \tag{2.17}$$

Here, it is built in two steps. First, the K activations $A_k, k = 1, \cdots K$, in the hidden layer, are calculated as functions of the input features $X_1, \cdots, X_p$ as

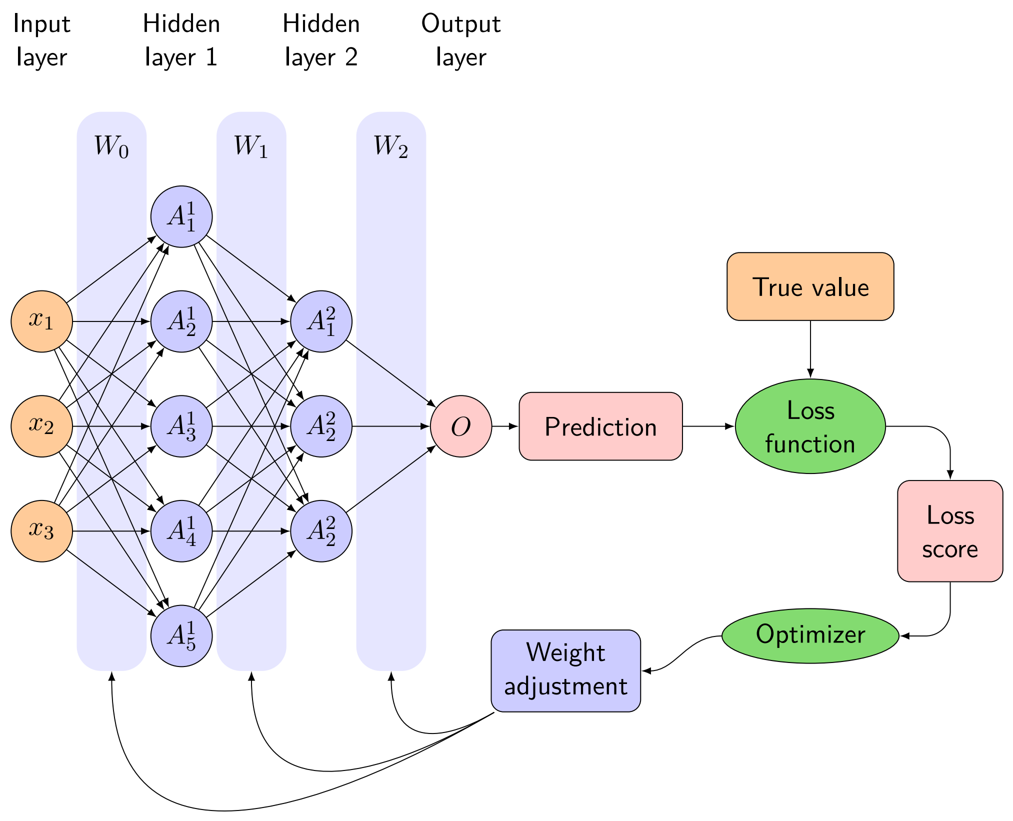


Figure 2.4: Illustration of a feed-forward neural network with two hidden layers. The hidden layers compute activations $A_k^j = H_k(X)$ that are nonlinear transformations of linear combinations of the inputs $X_1, X_2, \cdots, X_p$. The prediction and true value combine in the loss function, which produces the loss score. Then the optimizer adjusts the weights $W_0, W_1, \cdots, W_n$.

$$A_k = h_k(X) = g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right), \tag{2.18}$$

where $g(z)$ is a nonlinear activation function specified in advance. A_k can be thought of as a different transformation, $h_k(X)$, of the original features. The K activations then feed into the output layer resulting in a linear regression model in the $K = 5$ activations, as shown in

$$f(X) = \beta_0 + \sum_{k=1}^{k} \beta_k A_k. \tag{2.19}$$

All parameters, $\beta_0, \cdots, \beta_K$ and $w_{10}, \cdots, w_{Kp}$, need to be estimated from the data.

2.4.5 Transformer Models

Transformer models are used to extract textual features from news articles. These models are based on the NN encoder-decoder architecture. They are considered state-of-the-art when used for NLP tasks, and they also give great performance for sequential data in general. The architecture of the transformer model was first introduced by Vaswani et al. (2017). The models are very effective when training, given that the user has enough computing power. The effectivity comes from parallelization. Parallelization means that it can effectively read a whole sentence at a time if used for NLP. It also takes account of each word's position in a sentence by using a positional encoding mechanism. However, the most important aspect of the transformer model is *attention*. Attention is a mechanism that allows for the modeling of dependencies without regard to their placement in the input or output sequences. The transformer architecture is displayed in Figure 2.5.

Attention as a linguistic term As humans, we can understand what words other words refer to, or *attending*. Consider the following two sentences:

1. The AI executed the swap because it was an effective hedge.
2. The AI executed the swap because it had been trained to do so.

In the first sentence, we understand that *it* is referring to *the swap* because it is an effective hedge. In the second sentence, we understand that *it* is referring to *the AI* that has been trained.[3] This differentiation is easy for a human, but much harder for a machine. Attention mechanisms attempt to solve this problem.

[3]Example retrieved from *Lucidate: Attention is all you need explained* on Youtube, `https://www.youtube.com/watch?v=sznZ78HquPc&t=605s`

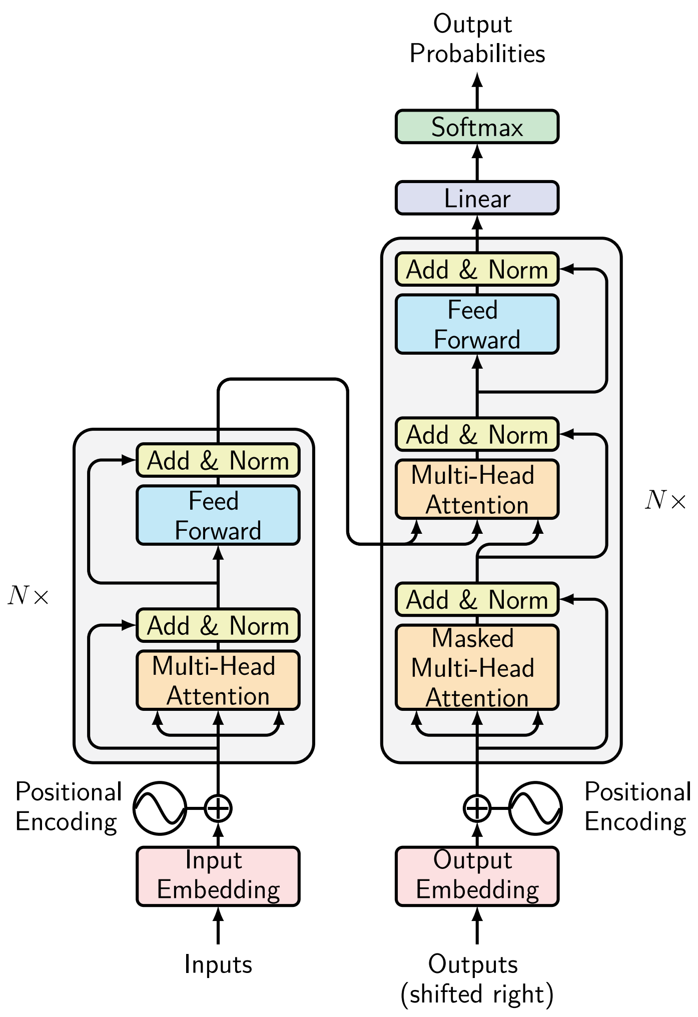


Figure 2.5: Illustration of a transformer model architecture. It illustrates how textual inputs are fed through several layers before output probabilities are calculated. The left unit is the encoder and the right is the decoder. BERT do not use the decoder part of the network and the output from the encoder is fed straight into the Linear and Softmax. Figure adapted from Vaswani et al. (2017).

Attention in more technical detail Attention is a function that can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is then calculated as a weighted sum of the values, where the weight assigned to each value is calculated using a compatibility function of the query with the corresponding key (Vaswani et al., 2017).

Furthermore, the transformer model uses *multi-head attention*. This form of attention uses multiple *scaled dot-product attentions*. Here, the input consists of queries and keys of dimension d_k, and values of dimension d_v. Then the dot products of the query with all keys are computed, and divided by $\sqrt{d_k}$, and applied with a softmax function. The output matrix is computed as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2.20}$$

This attention is then used in the *multi-head attention* as

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \cdots, \text{head}_h)W^O \\ \text{where } \text{head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right), \end{aligned} \tag{2.21}$$

where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \cdot d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \cdot d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \cdot d_v}$ and $W_i^O \in \mathbb{R}^{hd_v \cdot d_{\text{model}}}$.

The encoder maps an input sequence of symbol representations $(x_1, \cdots, x_m)$ to a sequence of continuous representations $\mathbf{z} = (z_1, \cdots, z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, \cdots, y_n)$ of symbols one element at a time. The output of the transformer model is a softmax probability distribution for all classes, where the predicted class is the one with the highest probability.

This framework has then given rise to multiple transformer-based models. Many of these are pre-trained models, and as such only require one additional output layer to fine-tune to a particular task. Transformer models have made BERT models possible.

2.4.6 BERT Models

BERT was developed and presented in "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Devlin et al. (2019). This is a model designed to pre-train deep bidirectional representations from unlabeled text by joint conditioning on both left and right contexts in all layers. It is based on the encoder section of the transformer architecture. The base BERT model uses a stack of 12 encoders, giving it a total of 110M parameters. BERT also has many related models that build on it.

Pre-trained BERT models Before a BERT model can be fine-tuned, it must be trained to understand the context of language. A pre-trained model is a model that has gone through the first part of training. This first part is where the model learns the dependencies of language using Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM is that random words in a sentence are masked. The model then tries to predict what the masked words are. NSP makes the model try to predict the next sentence. This is done so that the model can be used for question answering and other natural language inference tasks. It gives the model the ability to see a relationship between two sentences.

Fine-tuning BERT models After pre-training, a BERT model can be fine-tuned. Fine-tuning is when we train the model for specific tasks. For example, text classification, sentiment analysis, or question answering. It trains the model to alter its understanding of the language to perform more specific tasks.

FinBERT FinBERT is a common name used for BERT models fine-tuned to perform different tasks on financial data. There are several FinBERT models available on the Huggingface Hub[4]. Since there are several possible options, the choice of FinBERT fell on a version with well-documented performance for sentiment analysis. This is the FinBERT-tone developed by Huang et al. (2020). The model is fine-tuned to be a sentiment analysis model that predicts positive, negative, or neutral sentiments on financial textual data.

DistilBERT Sanh et al. (2019) has developed as a smaller model than the BERT model. This is called DistilBERT and is a distilled version of the regular BERT model. DistilBERT has been created because the developers wanted to pre-train a smaller general-purpose language representation model. Being smaller in size, DistilBERT can be more easily fine-tuned with good performance in a wide range of tasks such as the BERT model. The resulting DistilBERT model has reduced the size of BERT by 40%, while retaining 97% of its language understanding capabilities. The reduction in size has given the model the ability to train 60% faster than BERT. For researchers with hardware limitations, having a model like DistilBERT that can be easily fine-tuned and still perform well is valuable. In this thesis, we will fine-tune the DistilBERT model to predict HPR 20 minutes after a news article was published.

The BERT tokenizer The BERT tokenizer applies a WordPiece embedding, developed by Google. It makes sure that every sentence starts with a special classification token `[CLS]` and the sentences are separated using a `[SEP]` token. Huggingface states that the WordPiece training algorithm is not open source. This explanation is largely based on Hugginface's explanation of the tokenizer[5].

The WordPiece algorithm starts from a small vocabulary that includes the special tokens presented above and the alphabet. It then identifies subwords by adding a prefix, `##` for BERT. Each word is initially split by adding this prefix to all characters

[4]`https://huggingface.co/search/full-text?q=finbert` (search date 2023/05/12)
[5]`https://huggingface.co/learn/nlp-course/chapter6/6?fw=pt` (visited 2023/05/12)

within the word. The word *word* is split as follows: `(w ##o ##r ##d)`. The characters at the beginning of a word are all contained in the alphabet, and the characters are present inside a word preceded by the WordPiece prefix, `##`.

WordPiece then learns specific merge rules. It computes a score for each character pair as

$$\text{score} = \frac{\text{freq\_of\_pair}}{\text{freq\_of\_first\_element} \cdot \text{freq\_of\_second\_element}}. \tag{2.22}$$

By dividing the frequency of the pair by the product of the frequencies of each of its parts, the algorithm prioritizes the merging of pairs where the individual parts are less frequent in the vocabulary.

2.4.7 Limitations of a Neural Network

Although previous work indicates that NNs have high performance, there are some limitations and disadvantages when using them. These limitations consider interpretability, computational power, and data structures.

The black-box problem One of the main limitations when using NNs is the black-box problem. NNs are artificial intelligence (AI) that uses deep learning, an algorithmic system of deep neural networks, which on the whole remain opaque or hidden from human comprehension. This situation is commonly referred to as the black-box problem in AI. Without understanding how AI comes to its conclusions, it is an open question as to what extent we can trust these systems (Eschenbach, 2021).

The black-box problem is common when using complex models, and it is a problem with regard to interpretability. Complex models are similar to black boxes because it is difficult to understand what happens inside the models. The variables are put into the black box and the black box returns a prediction. The essence is that black-box models are more difficult to interpret than more basic models, such as regression models and decision trees. This is especially relevant if the NN is deep, indicating that it has many hidden layers and many nodes in each layer. The deeper the network, the harder it is to interpret the results.

Computational power and training data Another limitation of NNs is that they require a lot of data and computational power for training. Even training a NN for a simple task often requires large amounts of training data (Aggarwal, 2018). The number of parameters that must be optimized during training increases with the size of the network. A larger network will require more computational power to train effectively. This may be a problem if the model underfits. Underfitting occurs when the model is incapable of capturing the variability of the data (Jabbar and Khan, 2015).

A way of solving an underfitting model is to increase the complexity of the network or adjust the input features. There are techniques such as normalizing or standardization that can be applied to help the NN find patterns more easily. If the input features cannot be adjusted, then the network needs to be more complex. For a NN this means a larger, deeper, and wider network. The problem then is that the larger network requires even more computational power to train effectively.

Overfitting and how to overcome it If we work too hard to find the very best fit to the training data, there is a risk that we will fit the noise in the data by memorizing various peculiarities of the training data rather than finding a general predictive rule. This phenomenon is usually called overfitting (Dietterich, 1995). When training NNs, there are several tools available to help with overfitting. One way is to track the network during training with a logger to see the development of training and validation metrics. The point where the validation metrics have stopped improving and are getting worse, and the training metrics continue improving, is a sign that the network has started overfitting. If checkpoints are saved, then the network can be loaded from the optimal point in training and be used on testing data.

Overfitting can also be controlled by using early stopping, which is a tool that stops training when the metric being monitored has not improved significantly for a set number of training epochs. The network can also be made less complex or dropout layers can be used. Dropout is a form of regularization that deactivates a random set percentage of neurons in each layer. By using dropout, the network can still be complex to ensure that it does not underfit, and the dropout helps regulate overfitting.

2.5 Other Classifiers

Other classifiers are applied to further explore the relationship between movements in HPR and textual data. What is defined as other classifiers are ML models that do not use deep learning. The non-deep learning models used in this thesis are described in this section.

2.5.1 Logistic Regression

This is a variant of a regression model. It applies a technique that models the posterior probability of the classes K through linear functions of x. At the same time, it ensures that the probabilities of K sum up to one, and remains in $[0, 1]$. This description is largely based on Hastie et al. (2009). Logistic regression is formulated as

$$\begin{aligned}
\log \frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)} &= \beta_{10} + \beta_1^T x \\
\log \frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)} &= \beta_{20} + \beta_2^T x \\
&\vdots \\
\log \frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)} &= \beta_{(K-1)0} + \beta_{K-1}^T x.
\end{aligned} \tag{2.23}$$

The model is specified in terms of $K-1$ log odds or logit transformations. The regression is fit with maximum likelihood, using the conditional likelihood of G given X. The log-likelihood for N observations is

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i;\theta), \tag{2.24}$$

where $p_k(x_i;\theta) = \Pr(G=k|X=x_i;\theta)$.

The two-class problem is described in a bit more detail here, as the algorithms are considerably simplified. Let $p_1(x;\theta) = p(x;\theta)$, and $p_2(x;\theta) = 1 - p(x;\theta)$. The log-likelihood is then written as

$$\begin{aligned}
\ell &= \sum_{i=1}^{N} \Big\{ y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta)) \Big\} \\
&= \sum_{i=1}^{N} \Big\{ y_i \beta^T x_i - \log\Big(1 + e^{\beta^T x_i}\Big) \Big\}.
\end{aligned} \tag{2.25}$$

Here $\beta = \{\beta_{10}, \beta_1\}$, and we assume that the vector of inputs x_i includes the constant term 1 to accommodate the intercept. We then maximize the log-likelihood by setting the derivatives to zero.

2.5.2 Decision Trees

Decision trees create the foundation for a family of ML methods referred to as tree-based methods. This is a family of regression and classification methods. The models *stratify* or *segment* the predictor space into a number of simple regions. To make a prediction for a given observation, we typically use the mean or mode response value for the training observations in the region to which it belongs. The set of splitting rules can be summarized in a tree, also known as a *decision tree*. This is great for interpretability, but it cannot usually compete with more sophisticated supervised learning methods. For this reason, we also introduce *random forest* and *XGBoost*, which often have better predictive power, at the expense of some loss in interpretation. The following sections are largely based on Hastie et al. (2009) and James et al. (2021).

Fitting a regression tree We have a data set consisting of inputs p and a response, for each of the observations N. The algorithm needs to automatically decide on the splitting variables and split points, as well as on what typology (shape) the tree should have. We have a partition into M regions $R_1, R_2, \ldots, R_M$, and we model the response as a constant c_m in each region

$$f(x) = \sum_{m=1}^{M} c_m I\left(x \in R_m\right). \tag{2.26}$$

In the regression tree, we minimize the sum of squares $\sum(y_i - f(x_i))^2$. We see that the best $\hat{c}_m$ is the average of y_i in region R_m

$$\hat{c}_m = \mathrm{ave}(y_i | x_i \in R_m). \tag{2.27}$$

Finding the best binary partition in terms of the minimum sum of squares is usually computationally infeasible, and we use a greedy algorithm. Starting with all the data, we use a splitting variable (j) and a splitting point (s) and define the pair of half-planes

$$R_1(j,s) = \{X | X_j \leq s\} \text{ and } R_2(j,s) = \{X | X_j \ > s\}. \tag{2.28}$$

Then we look for the splitting variable (j) and (s) that solves

$$\min_{j,s}\left[\min_{c_1} \sum_{x_i \in R_1(j,s)} \left(y_i - c_1\right)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} \left(y_i - c_2\right)^2\right]. \tag{2.29}$$

For all choices of j and s, the inner minimization is solved by

$$\hat{c}_1 = \mathrm{ave}\left(y_i | x_i \in R_1(j,s)\right) \text{ and } \hat{c}_2 = \mathrm{ave}\left(y_i | x_i \in R_2(j,s)\right). \tag{2.30}$$

For each splitting variable, the determination of the split point s can be done quickly and hence by scanning through all of the inputs, the determination of the best pair j, s is feasible. Having found the best split, we repeat this split in all the resulting regions.

To avoid overfitting, we also need to set a tree size as a tuning parameter. The optimal tree size should be chosen adaptively from the data. The preferred strategy is to grow a large tree T_0, stopping the splitting process when a minimum node size is reached. We then prune this tree using *cost-complexity pruning.* This pruning works by defining a subtree $T \subset T_0$ as any tree that can be obtained by pruning T_0, that is, collapsing any number of its internal nodes. We index the terminal nodes by m, with node m representing region R_m. Let $|T|$ denote the number of terminal nodes in T. Letting

$$\begin{aligned} N_m &= |\{x_i \in R_m\}|, \\ \hat{c}_m &= \frac{1}{N_m} \sum_{x_i \in R_m} y_i, \\ Q_m(T) &= \frac{1}{N_m} \sum_{x_i \in R_m} \left(y_i - \hat{c}_m\right)^2, \end{aligned} \tag{2.31}$$

we define the cost complexity criterion

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|. \tag{2.32}$$

The goal is to find, for each α, the subtree $T_\alpha \subseteq T_0$ to minimize $C_\alpha(T)$. The tuning parameter $\alpha \geq 0$ governs the trade-off between the size of the tree and its goodness of fit to the data. α can be adaptively chosen.

For each α there is a unique smallest subtree T_α that minimizes $C_\alpha(T)$. To find this subtree, we can use *weakest link pruning*. This works by successively collapsing the internal node that produces the smallest per-node increase in $\sum_m N_m Q_m(T)$, and continues until we produce the single node tree. This gives a finite sequence of subtrees, and this sequence must contain T_α. We then estimate α by cross-validation,and we choose the value $\hat{\alpha}$ that minimizes the sum of squares, resulting in $T_{\hat{\alpha}}$.

Converting the regression tree to a classification tree Where we in the regression tree used the impurity measure $Q_m(T)$, we need to use another measure. In a node m, representing a region R_m with N_m observations, let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k), \tag{2.33}$$

the proportion of class k observations in node m. The observations in node m gets classified to class $k(m) = \arg\max_k \hat{p}_{mk}$, the majority class in node m. We then have different measures for the node impurity $Q_m(T)$:

$$\text{Misclassification error: } \frac{1}{N_m} \sum_{i \in R_m} I\left(y_i \neq k(m)\right) = 1 - \hat{p}_{mk(m)} \tag{2.34}$$

$$\text{Gini index: } \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}\left(1 - \hat{p}_{mk}\right) \tag{2.35}$$

$$\text{Cross-entropy or deviance: } -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}. \tag{2.36}$$

For two classes, if p is the proportion in the second class, these three measures are $1 - \max(p, 1-p)$, $2p(1-p)$ and $-p \log p - (1-p)\log(1-p)$, respectively. All of the measures are similar, but cross-entropy and Gini index are differentiable, and hence are more amendable to numerical optimization.

2.5.3 Random Forests

Random forests are built on the decision tree. The forest is constructed by creating a selection of decision trees and combining the prediction of these trees to make a prediction. A central idea for random forests is bagging.

Bagging (Bootstrap aggregation) This builds on the concept of bootstrapping. A given set of n independent observations $Z_1, \cdots, Z_n$, each with variance σ^2, the variance of the mean $\overline{Z}$ of the observations is given by σ/n. This shows that averaging a set of observations reduces variance. A natural way to reduce variance and increase the accuracy on the test set is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. We can then obtain a low-variance learning model given by

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{b}(x). \tag{2.37}$$

This method is not practical, as we usually don't have multiple training sets. A solution to this is to bootstrap, meaning that we take repeated samples from the training data set. With this approach, we generate B different bootstrapped training sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all the prediction to obtain

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{2.38}$$

Fitting the random forests. The random forest provides an improvement over bagged trees by way of a small tweak that decorrelates the trees. As we do with bagging, we build a number of decision trees on bootstrapped training samples. The difference is that each time a split is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors. At each split, a fresh sample of m predictors is taken, and usually, we choose $m \approx \sqrt{p}$. This means that at each split in the trees of a random forest, the model is not allowed to consider the majority of the available predictors. The main difference between a random forest and a bagged tree is the choice of predictor subset m.

2.5.4 XGBoost

eXtreme Gradient Boosting (XGBoost), introduced by Chen and Guestrin (2016), is another method that builds on the foundation of decision trees. The model uses clever penalization and regularization methods. It is based on a *gradient boosting* algorithm and decision tree ensembles. The description here is largely based on the description provided by Chen and Guestrin (2016) in their Github repository.[6] We include this model since it “achieves state-of-the-art results on many machine learning challenges” (Chen and Guestrin, 2016).

Decision Tree Ensembles This is one of the foundations of XGBoost. We can formulate the model as

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}, \tag{2.39}$$

where K is the number of trees, f_k is a function in the functional space $\mathcal{F}$, and $\mathcal{F}$ is the set of all possible decision tree ensembles. The objective function to be optimized is given by

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \omega(f_k), \tag{2.40}$$

where $\omega\left(f_k\right)$ is the complexity of the tree f_k defined in detail later.

Boosting trees We can now consider how the model trains. We start by defining an objective function that we optimize. For this model, we consider the objective function

$$\text{obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \omega(f_i). \tag{2.41}$$

After defining the objective function we need to find the parameters of the tree.

Additive training is a way to find these parameters. We need to learn the functions f_i, where each one is containing the structure of the tree and the leaf scores. Learning tree structure is much harder than traditional optimization problems where you can simply take the gradient. It is intractable to learn all the trees at once. Instead, we use an additive strategy: fixing what we have learned and adding one new tree at a time. We can write the prediction value at step t as $\hat{y}_i^{(t)}$. Then we have the following

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\vdots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i).
\end{aligned} \tag{2.42}$$

[6] https://github.com/dmlc/xgboost/blob/21d95f3d8f23873a76f8afaad0fee5fa3e00eafe/doc/index.rst

With this, we can find which tree we want at each step. This can be solved by adding the tree that optimizes our objective

$$\begin{aligned}\text{obj}^{(t)} &= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{i=1}^{t} \omega(f_i) \\ &= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \omega(f_t) + \text{constant}.\end{aligned} \tag{2.43}$$

If we consider using logistic loss, it is not so easy to get a nice form. In the general case, we then take the *Taylor expansion of the loss function up to the second order*:

$$\text{obj}^{(t)} = \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \omega(f_t) + \text{constant}, \tag{2.44}$$

where the g_i and h_i are defined as

$$\begin{aligned}g_i &= \partial_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right) \\ h_i &= \partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right).\end{aligned} \tag{2.45}$$

After we remove all the constants, the specific objective at step t becomes

$$\sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \omega(f_t). \tag{2.46}$$

This is then our goal of optimization for the new tree. One important advantage of this definition is that the value of the objective function only depends on g_i and h_i.

Model Complexity After the training step we introduce the regularization term. We define the complexity of the tree $\omega(f)$. In order to do so, we first refine the definition of the tree $f(x)$ as

$$f_t(x) = w_{q(x)}, w \in R^T, q : R^d \to \{1, 2, \cdots, T\}. \tag{2.47}$$

Here w is the vector of scores on leaves, q is a function assigning each data point to the corresponding leaf, and T is the number of leaves. We then define the complexity as

$$\omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2. \tag{2.48}$$

The structure score After re-formulating the tree model, we can write the objective value with the t-th tree as

$$\begin{aligned}\text{obj}^{(t)} &\approx \sum_{i=1}^{n}\left[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T,\end{aligned} \tag{2.49}$$

where $I_j = \{i|q(x_i) = j\}$ is the set of indices of data points assigned to the j-th leaf. We note that in the second line we have changed the index to the summation because all the data points on the same leaf get the same score. We could further compress the expression by defining $G_j = \sum_{i\in I_j} g_i$ and $H_j = \sum_{i\in I_j} h_i$:

$$\text{obj}^{(t)} = \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2\right] + \gamma T. \tag{2.50}$$

In this equation, w_j are independent with respect to each other, the form $G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2$ is quadratic and the best w_j for a given structure $q(x)$ and the best objective reduction we can get is

$$\begin{aligned} w_j^* &= -\frac{G_j}{H_j + \lambda} \\ \text{obj}^* &= -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T.\end{aligned} \tag{2.51}$$

The last equation measures how good a tree structure $q(x)$ is.

Learn the tree structure. When we have a way to measure how good a tree is, we would like to enumerate all possible trees and pick the best one. This is intractable in practice, so we try to optimize one level of the tree at a time. Specifically, we try to split a leaf into two leaves, and the score it gains is:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma. \tag{2.52}$$

This formula can be decomposed as 1) the score on the new left leaf 2) the score on the new right leaf 3) the score on the original leaf 4) regularization on the additional leaf. We notice that if the gain is smaller than γ, we would do better to not add that branch. This is *pruning* the tree.

3 Previous Work

This chapter introduces relevant previous work for this thesis. As stated in Section 2.2.1, NLP has theoretical value with respect to the predictions of the price of the stock. There are also multiple research articles investigating the relationship between financial news articles and stock price responses. We differentiate between *interday* and *intraday* data, where interday is day-to-day data and intraday is data within one day.

It was difficult to find relevant studies on intraday directional stock price predictions. Considering interday studies with applications of NLP, several studies are trying to predict volatility and not directional returns on shares. This chapter introduces previous work on predicting interday returns with and without the application of NLP. Studies on intraday stock price predictions with and without NLP are introduced later. At the end of the chapter, we present some of our thoughts considering previous work on predicting intraday returns.

3.1 Predicting Interday Returns

There is a lot of work done when it comes to predicting interday returns using deep learning. A search on Google Scholar for "*deep learning for stock market forecasting*" between the years 2000 and 2022 returns 16,400 results. This may be because interday data on stocks are widely available. In recent times, modern machine learning algorithms have been applied to predict stock prices. This section introduces studies that have approaches similar to ours on an interday basis. It must be considered that several studies are using NLP to model volatility instead of stock returns. These are interesting because the correlation between financial news and the direction of stock price movements may be similar to the correlation between financial news and stock volatility. It seems that volatility is more important on an interday basis.

Patel et al. (2015) addresses the problem of predicting directional movements of stock prices on the Indian stock market by comparing an artificial neural network (ANN), support vector machine (SVM), random forest and naive Bayes using two different input approaches. They study how different data preprocessing can influence predictions. The dataset used consists of 10 years of historical data from 2003 to 2012. Both the ANN and random forest yield good results when predicting the stock market. The ANN achieves an accuracy of 86.69% and the random forest achieves an accuracy of 89.98%. These results are from the best-performing data preprocessing. This suggests that the use of both the ANN and the random forest gives good results when forecasting the stock market.

Moghaddam et al. (2016) studies the ability of ANNs to forecast the daily NASDAQ stock exchange rate. They tried several FNNs and used short-term historical prices alongside a feature representing the day of the week. The data used is the daily exchange rates of NASDAQ from January 28, 2015, to June 18, 2015. Results suggest that the use of FNNs gives good results when forecasting the NASDAQ daily exchange rates with an R^2 of over 0.94.

Song et al. (2018) has published an article in which they compared a backpropagation NN to several other machine learning models and regression models on stocks in China. Here, they compared the backpropagation NN to models such as a radial basis function NN, a general regression NN, a support vector regressor, and a least squares support vector regressor. We note that they create a separate classifier for three different stocks in the Chinese market. The results show that backpropagation NN outperformed the other methods consistently and robustly, with an MSE of 0.009 on the Bank of China stock.

3.1.1 Applying NLP to Interday Returns.

Liu et al. (2017) uses sentiment analysis to improve the prediction of stock volatility using a recurrent neural network (RNN). It is important to note that this article defines volatility as the percentage change in the return from the previous day, in contrast to volatility, which typically is the standard deviation of the returns. It uses a dictionary-based weighted approach to sentiment analysis. The paper proposes an emotion model (EMM) following the assumptions that: (1) Increased bullishness of stock posts is associated with higher stock prices. (2) An increase in post-volume suggests more substantial volatility. The pure RNN model achieves an accuracy of 0.616, whereas the RNN+EMM model achieves an accuracy of 0.655. On the basis of this, the paper concludes that the sentiment analysis, combined with RNN gives a significantly better result. The model presented in the paper uses a relatively small sample size. For day traders, a shorter volatility time span can be more useful. This paper does however not address other performance metrics than accuracy, so it is hard to assess whether these results are, in fact, relevant.

Seng and H.-F. Yang (2017) analyzes the association between stock price volatility and financial news. They use a dictionary-based approach where they build their own dictionaries by reading and labeling data. The results show that financial reports, financial news, and added information influenced the volatility of the stock market. In the end, they find a statistically significant association between financial news and stock market volatility.

Deveikyte et al. (2020) focuses on predicting the return and volatility of the next day market. The article finds statistically significant correlations between sentiment and market volatility. They only investigate one single index. Furthermore, they use latent Dirichlet allocation and topic modeling, which further enhances the predictive power. In addition, this paper uses a relatively small sample size.

In a long short-term memory (LSTM)-based sentiment analysis for the forecast of the stock price in the Chinese stock market, Ko and Chang (2021) used a combination of a transformer model BERT to extract sentiment from news related to stocks and a PTT online stock forum in Taiwan. Sentiment was accompanied by the open, high, low, close, and volume for the related stock used as input variables in an LSTM to forecast the stock price at close the following day. They used several input vectors that included and excluded sentiment variables. Their results show an improvement in accuracy in the RMSE model by 12.05% when sentiment variables were included.

Chandola et al. (2022) investigates if a hybrid model can forecast the directional movement of stocks. The input of the hybrid model is news headlines from the Reuters website alongside a financial time series with closing prices from the related stocks. The chosen stocks are PepsiCo, Apple, APEI, NRG and AT&T. The hybrid model consists of an LSTM network where they use a word2vec embedder to vectorize the news headlines and then run the news data through the LSTM to extract the relevant textual information. The stock data are run through an ANN. The two outputs are then combined and run through a final artificial NN to make the final predictions. Overall, they average an accuracy of 52 % on validation data. They propose using a convolutional neural network to extract semantic information instead of LSTM for future work.

3.2 Predicting Intraday Returns

As witnessed in the previous subsection, much has been done regarding interday returns and applying NLP. When it comes to intraday predictions, it is hard to find much relevant literature. This could be because it is more difficult or expensive to retrieve these data.

Predicting intraday returns seems to be a difficult task. Some of the earlier work, for example, Ghosh et al. (2022) predicts intraday directional movements. It uses LSTM and random forests. They report that a daily return of LSTM, before transaction costs, is 0.64%. They use the opening and closing price for a given day as variables to construct HPR.

Ferreira and Medeiros (2021) conducts some experiments to model and forecast intraday market returns in SPY,[1] with some positive results. They investigated how the VIX[2] index can be used. An LSTM network appears to be the strongest candidate for prediction in this article, and random forests also give promising results. The paper focuses on the return, not the direction.

3.2.1 Applying NLP to Intraday Returns.

One of the first to examine the correlation between stock price movements and news data was Gidófalvi (2001). This article examines the prices of 127 stocks using 10-

[1]SPDR S&P 500 ETF. An exchange-traded trust fund designed to track the S&P 500 stock market index.

[2]CBOE Volatility Index. A measure of the stock market's expectation of volatility based on S&P 500 index options.

minute intervals and incorporates news articles with timestamps. The data cover the time period from November 14, 1999, to February 11, 2000. It establishes a prediction window of 20 minutes before and after the publication of news articles. His results show that news articles can predict the direction of a stock up to 20 minutes after the article has been published. The work also states that there is a strong correlation between news articles and the behavior of stock prices from 20 minutes before and up to 20 minutes after a news article has been published.

Cheng (2010) utilizes text mining of financial news to forecast the change in intraday stock prices. By extracting textual features from Money DJ stocks' financial news and stock price changes from the Taiwan Stock Exchange Corporation from January 1, 2007, to October 31, 2009, Cheng (2010) concludes that news events will definitely affect the short-term stock price. The empirical results show that the proposed model can accurately predict the directional movements of stocks in 15-minute intervals from 15 and up to 60 minutes after publication. It should be mentioned that the results of this study are almost too good to be true, and the most important aspect we consider is the short time frame studied.

Alostad and Davulcu (2015) investigates how the direction of stock prices can be predicted based on breaking news on Twitter. Here, an *up* movement is defined as a movement larger than 0, which can be problematic when considering transaction costs. This issue is not addressed. The article mostly uses n-grams to extract the textual features and construct a separate classifier for each stock ticker. They report a high accuracy of around 70%, but they do not mention other performance metrics or how balanced the dataset is.

Khadjeh Nassirtoussi et al. (2015) attempts a related task, where they attempt to predict currency instead of stocks. They do display other performance metrics, with the best models achieving precision in the domain of 70–85%. They use a 2 hour prediction window and use term frequency - inverse document frequency (TF-IDF) to extract textual features.

3.2.2 Thoughts Regarding the Previous Work

A particular point to note is that most of these articles only use accuracy as their performance metric. A model can have good accuracy, but it can be unusable if precision and recall are close to 0. There is no point in having a model with 70% accuracy if it only predicts *not-up*. It must also be considered that several of these studies never describe the distribution of their dataset. If the dataset is unbalanced, measuring performance using only accuracy may not be the best alternative. Therefore, even the most promising results must be taken with a grain of salt, since the results presented do not show the entire picture.

We find it curious that there is so little previous work in the field to be found and we have thought of two possible reasons for this. First, it might not be possible to predict intraday directional returns, and researchers have not wanted to publish negative results. Second, if someone were to find an algorithm that accurately predicts *up* movements, they would probably not want it to become available to the public.

They have monetary incentives to keep the algorithm for themselves.

We note that some of the previous work mentioned in this chapter is not peer-reviewed. This is a potential weak point in the literature, as articles are not necessarily checked by someone with expertise in the field. Specifically, the articles we are uncertain of are Deveikyte et al. (2020), Ferreira and Medeiros (2021), Gidófalvi (2001), and Song et al. (2018). We also note here that we are a bit critical of the articles that are peer-reviewed, as it seems strange that more performance metrics are not mentioned.

This page intentionally left blank.

4 Data and Research Design

In this chapter, we describe the data used in the analysis and present our research design. First, we give a description of the raw data sets used before preprocessing. Then we give an overview of what kind of preprocessing was required on the two different data sets. The second part of this chapter provides an exploratory data analysis concerning our target variable and our feature variables. Lastly, in this chapter, we present our research design.

4.1 Description of Data Sets

This section contains a description of both data sets used in our thesis. This description is a short introduction to what the data looked like before preprocessing.

NASDAQ news data set[1] The first data set contains approximately 1.4 million news articles. The data set contains the headlines of articles, the content, and the publication date. Most of the data is also labeled with stock tickers. We have chosen to filter out and only consider the 20 most mentioned stocks. These stocks make up 10% of the total data set. This might not seem much, but if we consider the 1,000 most mentioned stocks, that number only increases to 40%. Figure 4.1 displays the number of observations for the 1,000 most mentioned stocks, and how much of the total data set they make up. The number of companies contained in the data set is 14,740.

Refinitiv stock data set[2] This data set provides minute-level intraday stock prices. We have decided to gather data only for the 20 most mentioned stocks in the NASDAQ news dataset (NND). The data is retrieved from Refinitiv DataScope Select, a service providing this kind of stock data. When trying to retrieve the data for META[3], the database did not return anything. As we had some trouble accessing the stock database, another stock was not retrieved, and we only consider 19 stocks in this thesis.

To collect the stocks from the database, a RIC[4] must be provided. This code contains information about what financial instrument it is and on what exchange

[1] `https://blockchain-research-center.com/blockchain-explorer/`
[2] `https://select.datascope.refinitiv.com/DataScope/`
[3] Meta Platforms, Inc., previously Facebook, Inc. We also attempted to retrieve the FB ticker, but with no luck
[4] Refinitiv Identification Code

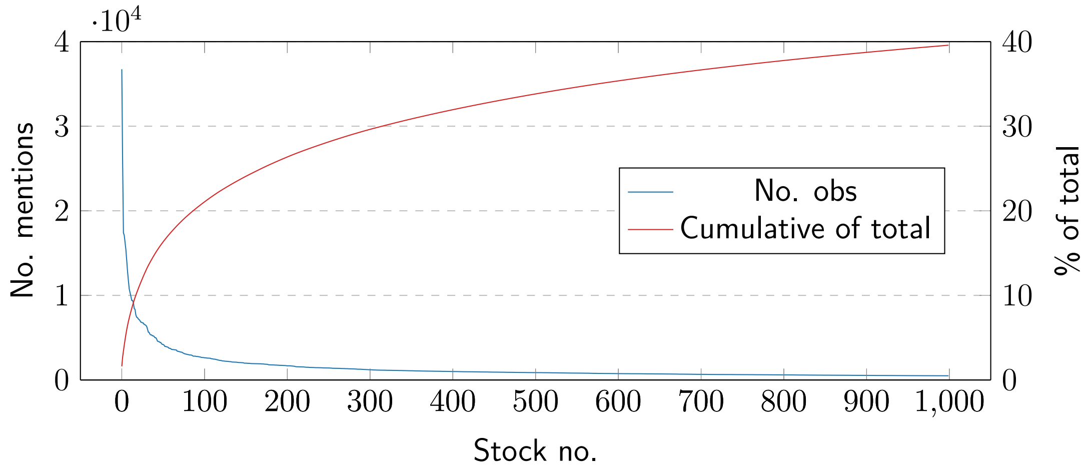


Figure 4.1: Distribution of mentioned stocks in NND, with the total number of mentions and the percentage of the total. The stocks are ordered based on the number of observations.

the instrument is traded on. We have decided to first collect the prices from the NASDAQ Stock Exchange. If the stock is missing from the NASDAQ Stock Exchange, we gather the price from the New York Stock Exchange. This is because the two exchanges are located in the same city, and thus we assume that they have approximately the same prices.

4.2 Preprocessing

This section describes and argues what preprocessing has been done on the data sets. Our raw data sets required various amounts of preprocessing to be able to perform any meaningful analysis on them. A more detailed description of the preprocessing is found in Appendix B.

NASDAQ news data set To be able to perform a textual analysis on the NND, preprocessing was necessary before the data could be run through a model. The preprocessing involves exploding the data so that each article only has one ticker associated with it. By doing this, the same article is represented in several rows, but only with one ticker in each row. A more technical description of this preprocessing is given in Appendix B.

Refinitiv stock data set The Refinitiv stock dataset (RSD) did not need much preprocessing. Since we have the stock price for each minute, it is required that a trade was made at every minute during the day. This is not the case in the data set, and missing timestamps have been added. We decided that the missing timestamps should take the value of the previous valid time. This is called forward-filling. For example, if AAPL had an observation for timestamp `2015-09-29 19:09:00`, but not for the next two minutes, we assume that the actual price for those two minutes was equal to the last trade.

The final data set Finally the NND and the RSD is merged on date. To be able to get the price for the time frames of [−30, −25, −20, −15, −10, −5, 20] minutes before/after the article was published, we simply added/subtracted the minutes from the actual time in the data set to get the time before/after the article was published. These times represent the lagged stock prices and the price 20 minutes after a news article was published. With this data we constructed a variable for the HPR, as defined in Equation (2.1). We then have data for the returns of all the time frames. The close price for minute 0 that is the reference when calculating HPR before and after the publication time of a news article. We note that perhaps the most relevant reference should have been close at minute −1. After exploring the descriptive statistics presented later in this chapter, we see that this single minute should not make a difference.

The data set now consists of 169,769 observations due to the removal of duplicates and N/A values. Since the stock exchanges we retrieved data from are only open for a specified time frame, we choose to filter out the times when the exchange was closed. This implies that we kept observations between 14:30 and 21:00 UTC during UTC−5 and between 13:30 and 20:00 UTC during UTC−4. Furthermore, since we use data 30 minutes before and 30 minutes after an article, we remove another 30 minutes at the start and end of the usual opening hours. In addition, we remove observations for Saturday and Sunday. The resulting data set consists of 55,952 observations.

Train-Test Split

When developing the data set we have decided to split it into training, validation, and test sets. The data set used for training consists of the first 70% of observations, the validation set consists of the next 20% and the test set consists of the last 10%. Additional descriptive statistics for each of the data sets are shown in Table 4.1.

Table 4.1: Descriptives of the different data sets

Metric	Training	Validation	Test
First observation	2009-10-16	2017-09-28	2018-12-04
Last observation	2017-09-28	2018-12-04	2019-08-27
No. observations	39,166	11,190	5,596

4.3 Exploratory Data Analysis and Description of Variables

To further describe the data set, we have conducted some exploratory data analysis. This analysis is based on the final data set after preprocessing, and only on the training and validation sets, to prevent data leakage.

4.3.1 Target Variable

The target variable chosen in this thesis is the HPR 20 minutes after an article was published. Since we predict stock price movements in intraday stock data, HPR is a way to standardize the price movements. This is important because different stocks have different values. By using HPR we take into account this difference in value. When trying to forecast the HPR of a stock 20 minutes after the publication of an article, we transform the problem into a binary classification task. The objective is to determine whether the stock's HPR experienced a *significant* increase or not. In this context, we define significant as the stock rising by more than 0.001 to account for transaction costs. Although this approach limits the potential for gains through short selling, it is more appropriate considering not all investors have access to short selling and the extremely short time frame involved. The resulting distribution of the target variable in the total data set is shown in Table 4.2. As the table displays, *not-up* movements represent 71.8% of the training set and 67.3% of the validation set. A model that only predicts *not-up* would then achieve high accuracy, and we know that metrics such as F_1 , precision, and recall are going to be better metrics.

Table 4.2: Distribution of the target variable across training and validation data sets.

Target class	Training	Validation	Sum
0	28,127 (71.8%)	7,532 (67.3%)	35,659
1	11,039 (28.2%)	3 658 (32.7%)	14,697
Sum	39,166	11,190	50,356

In Figure 4.2 we display how the returns are distributed for all the observations throughout the data set. We have also created a line at the cutoff point at 0.001. The mean (μ) for the variable is 0.000019 and the standard deviation (σ) is 0.002833. From Figure 4.2 we also observe that the amount of *up* movements seems to be spaced about the same throughout time.

4.3.2 Features

The features used for our final model are HPR in five-minute intervals starting from 30 minutes before an article was published to the time it was published. Textual

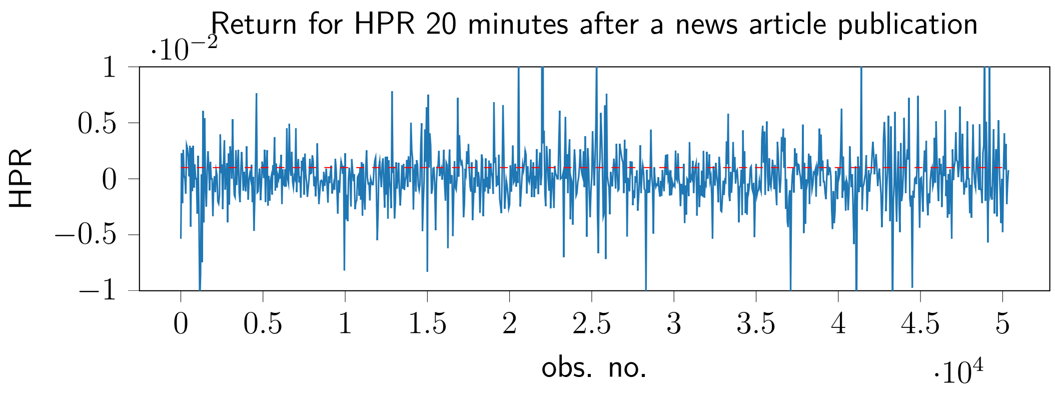


Figure 4.2: Return for the target variable. The red line is 0.001.

features in the form of a classification output from different BERT models are also part of the features.

We also included a variable for volume. Since different stocks have different volumes associated with them, we normalize the effect. To do this, we use minmax normalization defined as

$$x' = \frac{\min(x)}{\max(x) - \min(x)}. \tag{4.1}$$

4.3.3 Descriptive Statistics

To further describe the data set, we have conducted some exploratory data analysis. This analysis is based on the training and validation data sets, after preprocessing.

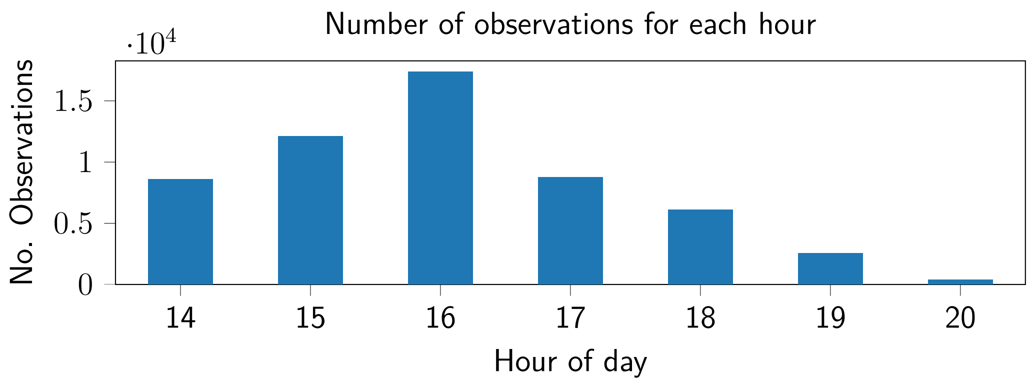


Figure 4.3: Number of observations per hour of the day. Note that the observations in each column are the amount of observations from the time stated to the next timestamp. The time is UTC.

Figures 4.3 and 4.4 display how the observations are distributed throughout the hours of the day and days of the week. We observe a peak around 16, with fewer

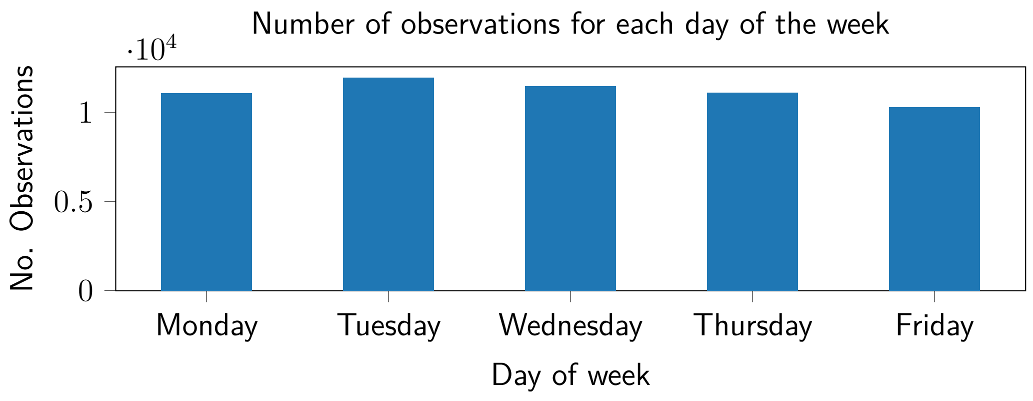


Figure 4.4: Number of observations per day of the week.

observations in the opening and closing parts of the exchange's opening hours. The observations seem to be equally spaced throughout the week.

Table 4.3 displays how many observations we have for the different stocks. We notice that AAPL is clearly the most mentioned stock. This might give our models a bias toward predicting the movements of AAPL. We also observe that Google is represented by two different tickers, both GOOG and GOOGL. We could have combined the two into one ticker since it is the same company, but considering that the prices for the two tickers are different, we did not do that.

Table 4.3: Number of observations per stock ticker.

Stock Ticker	# Observations
AAPL	9,782
AMZN	4,821
MSFT	4,740
BAC	3,307
INTC	2,976
JPM	2,750
NFLX	2,593
T	2,500
WMT	2,294
DIS	2,259
TSLA	2,238
GOOG	2,157
F	2,024
BA	1,996
XOM	1,977
VZ	1,948
GM	1,931
GOOGL	1,867
WFC	1,792

In addition, in Figure 4.5 we display how the HPR from -30 minutes up to the publication time is distributed, with Figure 4.6 showing the mean and standard deviation. We see a clear trend that the returns get smaller as the time frames become shorter.

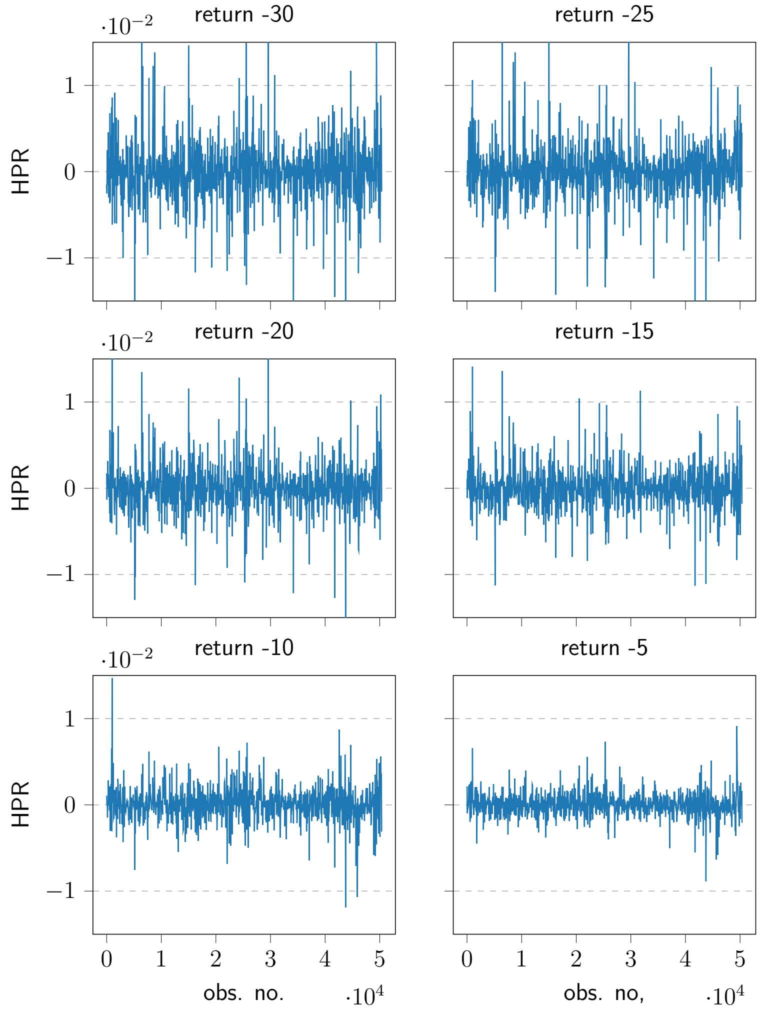


Figure 4.5: Distribution of returns for each feature variable. The graphs display the HPR from 30 minutes before a news article was published to 5 minutes before.

As displayed in Figure 4.6, it is evident that the mean returns exhibit a decline as the publication of the article approaches. The same applies to the standard deviation. This is natural as stocks tend to fluctuate less in a shorter time frame. A decrease in the standard deviation as we approach time 0 explains why a single minute's deviation when calculating the HPR for each time step is unproblematic.

4.4 Research Design

Our research design consists of using both the NND and the RSD. First, we preprocess both data sets, as discussed in Section 4.2. In this section, we outline how the data sets are combined to obtain our predictors.

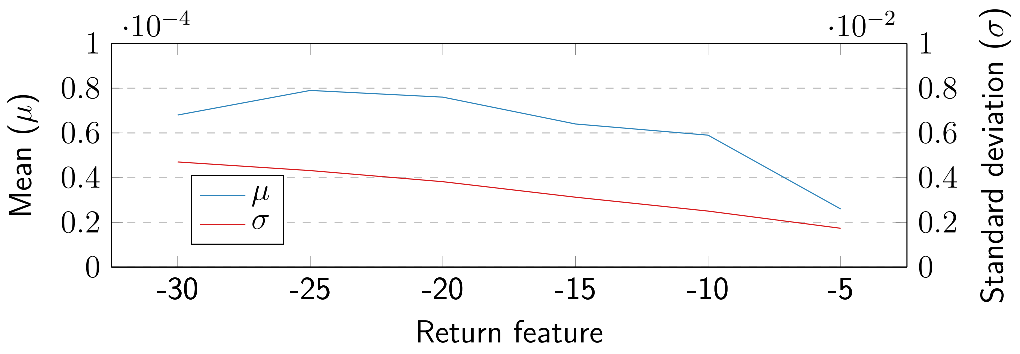


Figure 4.6: Mean (μ) and standard deviation (σ) for each return feature. Note that the scales is different for each measure.

NASDAQ News data set A flow chart showing how textual features are retrieved is displayed in Figure 4.7. We deploy a FinBERT sentiment classification and a DistilBERT directional classification on the data set. To be able to classify the direction using DistilBERT, we need to fine-tune it. This fine-tuning uses a 30% random sample from the training data set. After fine-tuning, we run the tuned classification model. We then extracted three sentiments,[5] and two directional features[6] from the textual data. The same procedure is applied to both news headlines and news content.

Additional methodology on FinBert and DistilBERT and FinBERT descriptive statistics are shown in Chapter 5, Table 5.2 (p. 49).

Refinitiv Stock data set A flow chart showing the features from the RSD is displayed in Figure 4.8. We calculate different HPR for different lags and calculate the sum of volumes in the previous half hour. The HPR for 20 minutes after a timestamp is fed into the textual data to fine-tune DistilBERT.

4.4.1 Benchmark Model

In this thesis, the research questions refer to how the use of textual data will help predict stock returns on intraday data. To see how well the model that includes textual features performs, we must also set a benchmark model. We create benchmark models for each model used. The benchmark model is an application of the same model, but without textual features.

By using a benchmark, we are able to explore if the textual data bring value to our models. We can compare the models including and excluding text using McNemar's test, and assess whether our models are significantly better than the benchmarks.

[5]positive (1), neutral (0) and negative (-1)
[6]*up* (1) and *not-up* (-1)

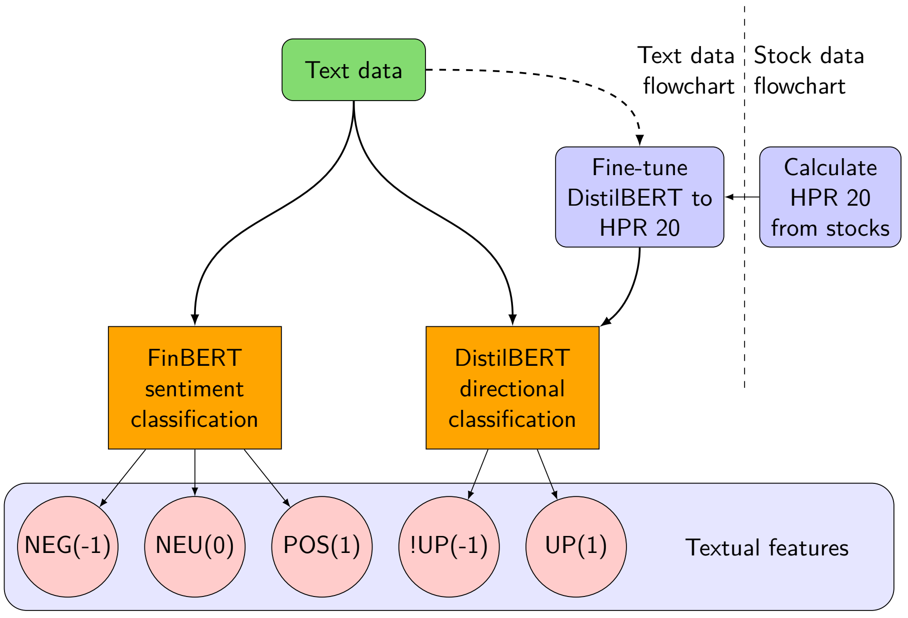

Text data
Text data flowchart
Stock data flowchart
Fine-tune DistilBERT to HPR 20
Calculate HPR 20 from stocks
FinBERT sentiment classification
DistilBERT directional classification
NEG(-1)
NEU(0)
POS(1)
!UP(-1)
UP(1)
Textual features

Figure 4.7: Research design for textual data. This procedure is done for both the article headlines and content. The calculation of HPR comes from the stock data set. The possible values for textual features is the bottom five nodes.

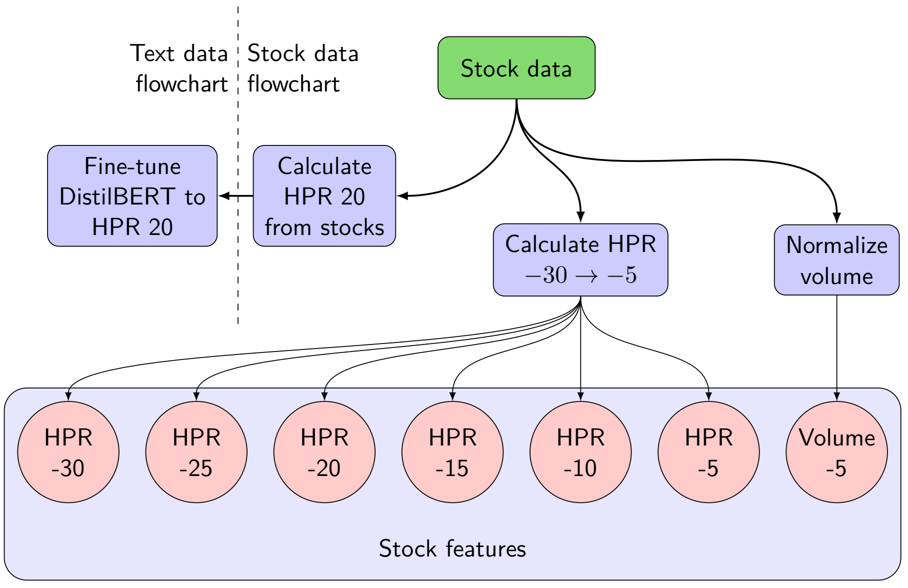


Figure 4.8: Research design for stock data. The stock features are the bottom nodes. The volume is not included in all models.

4.4.2 Performance Metrics

After the models are trained and the predictions on unseen test data have been completed, the results must be evaluated. Our model is set to solve a classification problem and we use the relevant classification metrics as mentioned in Section 2.3.1 when measuring the performance of the model. We are mostly going to focus on precision as our primary target for performance measurement. The reason behind this choice is that we need our model to accurately predict *up* movements. If the model is going to be implemented in an investment strategy, we need it to be correct when it tells us to buy. This is mostly relevant if the model has a performance that is good enough for a real-world application. To evaluate performance based on how well textual features help explain the stock market, several other metrics are also relevant.

To further investigate the connection between news articles and the stock market, we use the ROC curve and AUC to assess model performance. This is used alongside precision to see how well the model performs compared to a random model. These metrics together will give us enough information to state how well our model performs.

5 Feature Extraction from Text

In this chapter, we present how we extracted features from text. Feature extraction refers to the way in which we convert text into numerical representations. We did this by using a selected FinBERT and fine-tuning a DistilBERT model on the NND. The output of these models becomes the textual features. These textual features, combined with stock data, create the feature variables for our ML classifiers presented in Chapters 6 and 7.

We begin by motivating which BERT models to use for sentiment and text classification tasks and describing their limitations. Following, we present the features extracted from FinBERT and how we fine-tuned DistilBERT. Since DistilBERT will perform a text classification based on HPR 20 minutes after the publication of an article, we also evaluate its results on the validation set.

We end the chapter by evaluating the results of DistilBERT on a single stock. This is performed to see if DistilBERT performs better when fine-tuned on one stock compared to all stocks in the NND.

5.1 Methodology for Feature Extraction

This section tackles the methodology used to extract features from the text. Firstly, we discuss what BERT model to use and the limitations of BERT-based models. Then, we describe how FinBERT was used and how we fine-tuned a DistilBERT model.

5.1.1 Deciding Which BERT Models to Use

Since we have decided to use the BERT-based transformers model, we must also choose which BERT models to use. We introduce FinBERT and DistilBERT in Chapter 2, and those are the two models that we use to extract sentiment and classify text.

Model for extracting sentiment There are several ways to extract sentiment from text. Everything from simpler approaches, such as using dictionary-based models, to transformer neural networks utilizing attention. For this thesis, we need the sentiment feature to be as precise as possible. Therefore, we should use a model that we know is highly performing on similar data. Since the NND does not contain sentiment labels and the size is too large for any manual labeling to be carried out, we concluded that the best model that can be applied is a pre-trained model. At the Huggingface hub[1] there are a vast amount of pre-trained BERT models available.

We decided to use FinBERT-tone by Huang et al. (2020). This model is a variant of BERT that is fine-tuned on 10,000 manually annotated sentences from analyst reports and achieves superior performance in financial tone analysis tasks. It excels at identifying positive or negative sentiment in sentences that other algorithms mislabel as neutral, probably because it uses contextual information in financial text (Huang et al., 2020). This suggests that we utilize a model that has been proven to yield good performance, which is precisely what we are looking for.

Model for text classification For text classification, we are using the DistilBERT model. The model we are using is the `distilbert-base-uncased` which is a distilled BERT model that is not case sensitive. As explained in Section 2.4.6, it is smaller and yields almost the same performance as the basic BERT. The reason why we chose to use DistilBERT instead of BERT is due to hardware limitations, as it made it possible to train longer.

5.1.2 Limitations of BERT-Based Models

Since BERT is the base model from which both FinBERT and DistilBERT are derived, they share the same limitations as BERT. The primary limitation is that BERT has a maximum length of 512 tokens, where one token is one word. This is not an issue when using the model on article headlines, but article content can be longer. The average length of the content of the article is 606.23 tokens (words). In this thesis, we decided to truncate the inputs to 512 tokens. This signifies that any words appearing after 512 will be excluded. Truncation is not the optimal solution, but we hope that 512 tokens will be sufficient for the model. Extending this limit would require the input to be split into sections of 512 tokens, and then the result would be an average of classifications from the different sections. The models already are quite computationally intensive, and this expansion would make it so that running them would take a lot more time.

The time it took to run the models on the hardware described in Appendix A.1 is shown in Table 5.1. The reason why DistilBERT runs faster is because the model is smaller. Keep in mind that this is only the time it took to classify all the observations, not the training. The DistilBERT content model took about 10 hours to train on our hardware.

5.1.3 Application of FinBERT

FinBERT requires that the text in the news articles is tokenized and embedded. This process is simplified by using a pre-trained tokenizer. FinBERT uses a pre-

[1]`https://huggingface.co/docs/hub/index`

Table 5.1: Timings from running BERT-models. The time is displayed in MM.

FinBERT		DistilBERT	
Headlines	Content	Headlines	Content
11	49	06	19

trained `BertTokenizer`. The `BertTokenizer` is used as it is from HuggingFace.[2] Table 5.2 shows the distribution of the sentiment features extracted by FinBERT in the training and validation set.

Table 5.2: Descriptive statistics from FinBERT sentiment classification. We display the statistics for training and validation sets on both headlines (H) and content (C).

	Training		Validation	
Predicted class	H	C	H	C
Negative (−1)	5 193	4 458	1 700	1 298
Neutral (0)	26 294	21 680	6 496	5 175
Positive (1)	7 679	13 028	2 994	4 717

After performing a sentiment analysis with FinBERT, and analyzing how sentiment matches price movements, we considered the possibility that sentiment in the title and content may not be sufficient. We wanted to use sentiments based on the assumption that positive sentiment will indicate an *up* movement in HPR and the opposite will indicate *not-up* movement. On the basis of the investigation of our sentiment analysis, we concluded that our hypothesis did not hold. Therefore, we decided to fine-tune a DistilBERT model to predict the HPR values using only text data. In this way, we obtain a model tuned for our specific task of predicting stock movement using HPR. Adding another dimension of textual features may also help to understand how textual data influence HPR movements in the short term.

5.1.4 Fine-Tuning of DistilBERT

When fine-tuning DistilBERT we create a subset of the training data we use in our prediction models. To construct this subset, we draw 30% of the training data as a random sample. As fine-tuning takes a lot of time, we use a random sample to give the model a representative sample across the whole training set. Furthermore, 20% of the random sample is reserved for validation. The hope is that this random sample can be generalized across the entire data set. When the subset is created, we tokenize the text and encode it using `DistilBertTokenizerFast`[3], a faster tokenizer built

[2] `https://huggingface.co/docs/transformers/model_doc/bert`

[3] `https://huggingface.co/docs/transformers/model_doc/distilbert`

for the DistilBERT model. When the data is tokenized we can train the model. During training, the DistilBERT model will then fine-tune its parameters to predict the target variable, which is HPR 20 minutes after a news article has been published. Considering that we are using both article headlines and article content, we must fine-tune a model for both of these features.

A detailed description of the fine-tuned models and training loss is found in Appendix C.1. Evaluation metrics for fine-tuned models during training are shown in Table C.1. Although these results are not particularly impressive, the goal is that a model can make use of them when predicting HPR. Our hope is that these features will give better predictions when used alongside stock data. In the next section, we evaluate the DistilBERT model.

We note that a 30% sample might not be enough data to fine-tune the best possible model. Due to our limitations in computing power, it would not be feasible to use the whole training set. In this way, we will not get problems with overfitting.

5.2 Evaluation of DistilBERT

To understand how well DistilBERT performs the text classification task, we evaluate its performance on the validation set. The evaluation gives us insight into how well textual data can explain stock price movements in isolation from stock-related data. The metrics presented are based on the training and validation sets. We use the validation set to evaluate the DistilBERT model, to prevent data leakage.

Table 5.3: Evaluation metrics from fine-tuned DistilBERT on training and validation sets. These values are the evaluation metrics for class 1, *up* movements.

	Training		Validation		Both	
Metric	Headlines	Content	Headlines	Content	Headlines	Content
Accuracy	0.7235	0.7268	0.6049	0.6047	0.6971	0.6997
Precision	0.4146	0.4299	0.2239	0.1706	0.3671	0.3654
Recall	0.5116	0.5185	0.3411	0.3100	0.4755	0.4809
F_1	0.4580	0.4701	0.2703	0.2201	0.4144	0.4153

As shown in Table 5.3, the performance achieved on the training data can be considered satisfactory. Of all the *up* predictions our model generates, it is correct above 40% of the time (precision), and it hits around 50% of the total number of *up* movements (recall). We note that DistilBERT performs worse when evaluating the results on the validation set. The reason why these predictions are interesting is because a ML classifier or a NN could learn which of the predictions to use in different scenarios, or a combination of them, combined with stock data.

A metric that could be more useful to look at is the number of times the model trained on headlines *or* the model trained on content hits *up* predictions. These metrics are shown in Table 5.4.

Table 5.4: Number of correct *up* predictions (precision) from DistilBERT.

Data set	Headlines	Content
Training	0.5116	0.5185
Validation	0.3411	0.3100
Both	0.4755	0.4809

Considering that this only applies to the correct *up* predictions (precision), we can look at the accuracy of DistilBERT predictions, where we want to know how many correct predictions the model makes as a whole. These metrics are shown in Table 5.5.

Table 5.5: Number of correct predictions (accuracy) from DistilBERT.

Data set	Headlines	Content	Either
Training	0.7235	0.7268	0.8791
Validation	0.6049	0.6047	0.7623
Both	0.6971	0.6997	0.8531

One of our fears when conducting these experiments was that both models would predict the same directional movement. As shown in Tables 5.4 and 5.5, this is not the case. The *either* column shows how the models perform if a correct *up* prediction from one of the models is considered an *up* prediction from both. The accuracy is higher than for the individual models, and from this, we see that the models make different predictions. Considering the target variable distribution in Table 4.2, the accuracy in the *either* column is better than just prediction *not-up*.

5.2.1 Considerations Regarding Threshold Values

Prediction thresholds were not considered when performing the feature extraction. Models based on BERT output a label paired with a *confidence score* for each class when making predictions. Compared to a softmax probability distribution with class probabilities that add up to 1, the confidence score works differently. It is only a score that specifies how confident the BERT model is that the text input matches the predicted label. By default, it returns the label with the highest confidence score.

The fact that we did not assess different thresholds for our DistilBERT model is a potential shortcoming with respect to our predictions. This might have consequences for the later models that use the DistilBERT predictions.

5.3 A Single-Stock DistilBERT Model

In this section, we attempt to train a model on only the AAPL stock. This data set consists of 9,782 observations. The results of the fine-tuning are shown in Table 5.6. From this initial test, it seems like a model trained on only one stock has roughly the same performance as the one trained on the whole data set. We note that these metrics are not from the test set.

Table 5.6: Evaluation metrics from fine-tuned DistilBERT on AAPL during training.

Metric	Headlines	Content
Loss	3.006	3.038
Accuracy	0.6715	0.5912
F_1	0.2197	0.3171
Precision	0.3393	0.3023
Recall	0.1624	0.3333

Based on these results, we see that the text affects a stock in the same way, regardless of what stock it is. Building on this, we assume that building a separate classifier for each stock would not bring relevant gains to our predictions.

6 Integrating Text and Stock Data in a Neural Network

This chapter describes the methodology and experimental results of NNs. We begin by motivating what kind of network architecture we chose for our experiments. Then we run our experiments with FNN using multiple different topographies, including and excluding textual features, and present the results. The results of the best-performing typography are then evaluated using the ROC curve. Finally, we discuss our results before giving a brief summary of our findings.

6.1 Methodology for Neural Networks

In this section, we describe the methodology used for our neural networks. The methodology includes how we chose our network architecture and how we use the model to explore our research questions.

Deciding on NN Architecture NNs comes in different forms and the different forms are suitable for different tasks. Since our goal is to explore how news articles can help explain the stock market, we have processed our data set to have features of HPR at several time steps before the publication of an article. This data structure works well when using FNNs. Since the FNN architecture is one of the least computationally heavy architectures to train, we decided to use an FNN as our NN.

A hybrid model where textual features are extracted using LSTM, and used alongside stock data in an FNN has been done by Chandola et al. (2022). Although they did not achieve an accuracy higher than 52%, they also suggested using a different model to extract semantic features. In this case, we are using two BERT models to extract the semantic data. Having textual features extracted from the BERT models alongside stock data in an FNN is an approach that builds on the work of Chandola et al. (2022).

Building the FNN Since there are no simple, or algorithmic, ways to find the optimal NN, we conduct experiments. In these experiments, we test different network sizes to see which ones give the best fit to our data. We also test different activation functions to see if they give different results. The results of the experiments are presented in Section 6.2.

Excluding textual features To see if textual data add value when predicting HPR 20 minutes after the publication of an article, we train two separate FNNs. An FNN including text and one excluding textual features. We can then compare the results, including and excluding textual features, to see if there are any differences. Hopefully, the textual data adds value to the predictions. If the predictions are better when including text, the results will indicate that the data contained in the news articles are explaining the stock market.

Evaluation metrics To evaluate the results of the experiments, we use the evaluation metrics introduced in Section 2.3.1. The ROC curve is used to investigate whether models that include and exclude textual features are better than random guessing. We also present precision, recall, F_1 and accuracy for both models. These metrics will give us a full picture of the performance of our NNs.

6.2 Experiments Running FNN

In this section, we present some of our attempts to train an FNN. We present the architectures used and the results we got.

The goal of these experiments is for a network to be able to understand which features to use in its predictions. We attempted to run a network with four hidden layers and [128-64-32-4] nodes in each layer. We let the model run for 5 hours (340 epochs) to obtain the results shown in the first column of Table 6.1.[1] Graphs of the evaluation metrics obtained during training are displayed in Figure C.3. It should be mentioned that the metrics presented stayed the same for most of the training period. Although an accuracy of 66% might seem good, we notice that the recall for *up* predictions is very low. We only hit about 4% of the total number of *up* predictions. The precision of our network is not particularly impressive either, of all our *up* predictions, only 30% are actually true.

Table 6.1: Results of running multiple different topographies on the network. The first network ran for 340 epochs and ReLU activation, the second for 50 and leaky ReLU, and the third for 395 epochs and tanh activation.

	Typography		
Metric	[128-64-32-4]	[540-1080-720-64]	[5000-3000-1500-150-5]
Accuracy	0.6663	0.6702	0.6702
Recall	0.0371	0.0383	0.0383
Precision	0.2956	0.2975	0.2975
F_1	0.0659	0.0679	0.0679

[1]Note that the first model was run on a slightly different data set, that did not properly filter out the times when the exchange was closed.

Since the performance of the first network shown in Table 6.1 is poor, we decided to add more complexity to the network. The reason behind this is that the current FNN is underfitting. It is unable to find any patterns in our data. One way to improve an underfitting network is to add complexity. We decided to increase the number of nodes in the layers of the network. The more complex network also consisted of four hidden layers with a configuration of [540-1080-720-64] nodes in the layers. In this network, we also tried using leaky ReLU activation functions instead of regular ReLU since we thought we might have encountered the dying ReLU problem explained in Section 2.4. This network ran for 50 epochs with a learning rate of 0.01. It gave the results of the second column shown in Table 6.1. We observe that the performance metrics are approximately the same as the metrics for the smaller network.

Finally, we tried an even larger network, with [5000-3000-1500-150-5] nodes in the hidden layers. This model ran for 16 hours with a learning rate of 0.001 and tanh activation functions. The results of this network are shown in the last column of Table 6.1.

Looking at the performance metrics, adding complexity did not improve the network. The increase in complexity from [540-1080-720-64] to [5000-3000-1500-150-5] did not produce better performance. This can indicate that there is no substantial pattern to find in the data.

6.2.1 Adding Volume

We attempted to add normalized volume to the largest network, and the results improved slightly. The results are shown in Table 6.2. On observing the results, it is apparent that the precision of this network is marginally higher than that of the network without volume. Due to limitations in computing time, we were unable to run this network as long as the largest one did.

Table 6.2: results of adding volume to the largest network, running for 2h (54 epochs)

Metric	With volume	Without volume
Accuracy	0.6639	0.6702
Recall	0.0389	0.0383
Precision	0.3139	0.2975
F_1	0.0693	0.0679

6.2.2 Dropping Textual Data

We ran a test on the [5000-3000-1500-150-5] network, where we removed all textual features. This resulted in an accuracy of 0.6865, but it did not predict any *up* movements. Although the accuracy is better than any of the models shown in Table 6.1, it would not make a good trading model because it would never tell you to buy, even though you would lose less money by using that model. The model was trained for 2.5 hours (62 epochs).

Table 6.3: Comparison table between the best performing FNN [5000-3000-1500-150-5] with and without textual data.

Metric	Text	No Text
Accuracy	0.6702	0.6787
Recall	0.0383	0.0000
Precision	0.2975	0.0000
F_1	0.0679	0.0000

6.2.3 Assessing the ROC curve

We show the ROC curves for our FNN with and without textual features displayed in Figure 6.1. The analysis reveals that the model with text does not present a noticeable advantage over random guessing. This is disappointing, but not unsurprising given our performance metrics. We see an interesting characteristic with the model without text. It is slightly better than random guessing at high threshold values and slightly worse than random guessing at low threshold values, as observed when we move up to the right in the figures. This is interesting considering that the model achieves a precision and recall of 0.

It is probably due to the low recall for both models that the AUC is close to 50%. Since AUC is a metric that plots TPR and FPR, the overall low recall indicates that both models are unable to find patterns for *up* predictions. This is true for the model that includes textual features at all prediction thresholds.

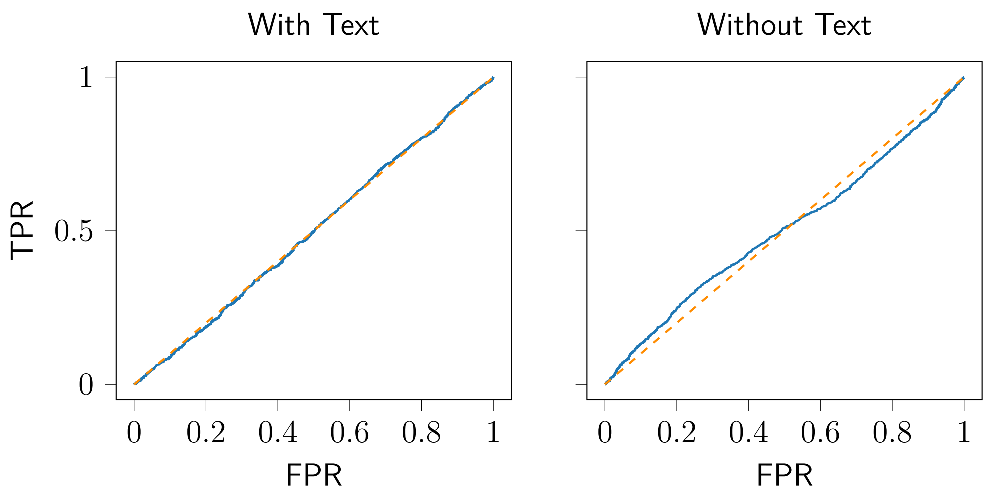


Figure 6.1: ROC curves of FNN with and without text. The AUC is 0.50 in both cases.

6.3 Discussion

This section explores the results of our FNN and problematizes them. Model training is also problematized, since increasing the model size did not add much value. After problematizing the FNNs training, we discuss the results that include and exclude textual features.

Exploring the experimental results The results of testing different network typographies are shown in Table 6.1. Although there is no difference between the deepest [5000-3000-1500-150-5] and the second deepest [540-1080-720-64] network, we proceeded with the deepest network. Since the FNN did not train, we also added trading volume as a feature to increase the number of stock-related features. The addition of volume to the deepest network gave a small increase in precision of 1.5%, shown in Table 6.2, but the performance of the FNN is still poor. It seems that the FNN was unable to find patterns in the data. If the model found a pattern in the data, we should have observed loss values closer to 0. Figure C.4 displays the development of the best-performing model during training.

Out-of-sample predictions are shown Table 6.3 and they are very interesting. The network without textual features has a precision of 0 and a recall of 0. This implies that the model does not predict any *up* movements in HPR at all.

Comparing models with and without textual features Since the model without textual features did not predict any *up* movements in HPR at all, we must discuss this fact. Table 6.3 shows a recall, precision, and F_1 score of 0 on the network excluding text. When including textual features, it is able to make a few *up* predictions. Considering research question 1, the performance difference with and without text shows that textual data helps to predict the direction of stock returns on intraday data when using an FNN. This will indicate that the textual data contained in the news articles are explaining the stock market. What we find somewhat concerning is that the performance of the model without textual data is very poor.

Even though the model that excludes textual features has poor performance, it should not be overlooked that the model that includes textual features has better performance when evaluating precision. However, when adding more intricate performance metrics, we see that there is little to no difference between the models that include and exclude textual features. The ROC curve in Figure 6.1 displays an AUC of 0.50 for both models. This is considered to be the same performance as a model that performs random guesses when performing classifications.

The random walk theory states that stock price movements are random and that there are no patterns in their movements. Taking into account this theory, our results indicate that it is true. An FNN is capable of finding patterns in noisy data, but it seems that intraday stock data in the time frame used in this thesis is too noisy. However, we still problematize the performance of the FNN to see if we could have done things differently.

Assessing model performance Since the FNN in general performed poorly, both including and excluding text, need to understand why. This is difficult because there is not much previous work published on the subject of intraday HPR predictions. Previous work on the subject of interday HPR prediction shows better results, but most of these articles only mention the accuracy of their models. For example, Liu et al. (2017) measures an accuracy of 61.6% on an RNN model excluding text features. Comparing the results of Liu et al. (2017) to our FNN which achieved an accuracy of 67.87% excluding textual features and zero *up* predictions made. The comparison of results shows how only using accuracy as a metric can be misleading as well.

The scarcity of previous work on the subject of predicting intraday stock price movements using an FNN with applications of NLP, makes it difficult to assess the performance of our FNN. In the future, researchers using FNNs to perform a similar task may use our model as a benchmark to beat and then explore if other variables may add value to the predictions.

Assessing model training It is a fact that our FNNs did not train well. The chart in Figure C.3 displays the training process, and it looked like this when training every model. We must then try to understand why the model parameters did not converge toward their optimal value. The first place to start is the data set. Our data set uses the HPR in five-minute intervals from $t - 30$ minutes to $t - 5$ minutes. Volume is minmax-normalized to shrink the effect of outliers. The structure of our data should be in a way that an FNN should understand. There may be some issues with all of our lagged HPR variables being very close to zero. This is perhaps the biggest obstacle for the model. Having most features close to zero may make it difficult for the model to find the optimal weights and biases for our data.

Another possibility is the activation functions applied in the network. In the first network, we used the ReLU activation. This implies that if a lot of the features are negative numbers, we encounter the dying ReLU problem. This happens because the ReLU returns 0 if the input is 0 or less. To adjust for this, we used leaky ReLU functions with a negative slope of 0.1. The leaky ReLU will return small negative numbers when a negative number is an input in the function. The use of leaky ReLU made no difference.

Since our features are already very close to zero, returning a number even closer to zero may not make any difference at all. Since the leaky ReLU did not help, we tried tanh in the final, deepest, network. However, using tanh did not help our network train better. We also dropped the learning rate when training the different models and we also applied learning rate schedulers in our experiments. Adding learning rate schedulers did not improve the training process.

Since the loss values during training do not improve and are stuck around 0.5, it is not a very good model. This will in the end suggest that the network is either not complex enough or that the stock data is too noisy for the model to find any pattern. If there is no pattern to be found, then the model will not train no matter how complex it is.

Investigating other threshold values We have observed from the ROC curves that our FNN seems to perform equally well regardless of the threshold value for the model with text, while the model without text shows tendencies to be susceptible to changes in threshold values. This implies that our results of the FNN must be taken with a grain of salt.

6.4 Summary Results of FNN

The results of the FNN indicate that the inclusion of text data improves the predictive capacity of the model, even though the overall results are suboptimal. Of particular interest is the difference between models with and without text. In particular, the model with text achieves a precision of 30%, while the model without textual features has a precision of 0%. Although the accuracy evaluations may seem similar, the precision metric reveals that the model with text is superior in our context.

The results demonstrate that textual data help predict the direction of stock returns on intraday data using an FNN with a threshold value of 0.5. The ROC curves and the AUC indicate that the model with text performs consistently across different threshold values, whereas the model without text performs better at lower threshold values, and subsequently worse at higher values. These findings suggest that the inclusion of text data could add value to the prediction and should not be overlooked.

Given that the AUC of both ROC curves displayed in Figure 6.1 is 0.5, we conclude that it is best to discontinue the use of FNN and explore alternative classifiers in Chapter 7. This chapter will introduce and evaluate alternative classifiers that could potentially yield better results. It will also discuss the advantages and disadvantages of each classifier and suggest which might be the best fit for the data.

This page intentionally left blank.

7 Integrating Text and Stock Data using Other Classifiers

The results of FNNs described in Chapter 6 indicate that textual features do not help explain the short-term movements of HPR. The FNN which includes textual characteristics made some correct *up* predictions. This resulted in higher precision and a very small increase in recall. Since the FNN was unable to find patterns in the data, we decided to explore traditional ML methods. We investigate how the data from the news articles can help further understand the stock market by using the following models: XGBoost (XGB) and random forests (RF) and logistic regression (LR).

This chapter presents how we tuned the hyperparameters for the different ML models and the results they achieved. We also explore different prediction thresholds to see if the results change significantly when reducing the threshold. The results are presented and compared with and without text. The significance between models including and excluding text, as well as models using different prediction thresholds, are measured using McNemar's test statistic.

The effect of news articles on the stock market is further explored by looking at the LR coefficients. An interpretation of the coefficients can help to understand how the characteristics of the different features affect the log odds of an *up* prediction of HPR.

7.1 Hyperparameter Setup

To perform hyperparameter tuning, we are using three-fold cross-validation, as introduced in Section 2.3.2. The scoring is precision, as that is the metric we are most interested in. We use cross-validation on the training set and use the validation set to set a threshold.

General methodology for selecting hyperparameters To get the optimal hyperparameters we use a general algorithm as described in detail in Appendix D. The basics are that we try with some initial values and then try with more values that are close to the best from the previous step. We use precision as an initial scoring parameter but also monitor recall when we choose parameters. This happens because a model naturally gets higher precision with lower recall.

The hyperparameters for the different models are shown in Tables D.1 to D.3 in Appendix D (p. 101). Where nothing is defined, default values from `scikit-learn`

(Pedregosa et al., 2011) are used. We note that a more extensive hyperparameter search could have been conducted, but the results of our testing indicate that there are limited gains to be had by doing this.

7.1.1 Evaluating Prediction Thresholds

With the findings from the ROC curves presented in Section 6.2.3 in mind, we evaluate our other classifiers using different prediction thresholds. In this subsection, we investigate what thresholds are best on the validation set. We also try to run some different thresholds for our classification models.

ROC curve analysis and AUC scores We start by analyzing the ROC curves and AUC scores for our models with and without text. These curves provide valuable insights into the overall performance of the classifiers. The ROC curves are displayed in Figure 7.1. We notice from the plots that the model without textual features has a higher AUC. This is strange, as results from Chapter 6 suggest that a model with textual features performs better than a model without it, because it actually made *up* predictions. This discrepancy suggests that the threshold for our FNN may not be optimal.

The dashed orange line, displayed in Figure 7.1, indicates that, with the exception of logistic regression, all models are anticipated to outperform random guessing. We notice that the tree-based models with text exhibit two “bumps”: the first in the FPR domain 0.1–0.4, and the second in the FPR domain 0.4–1.0. These results indicate that the models perform better at lower prediction thresholds and worse at higher prediction thresholds. Where the blue line intersects the orange dashed line, the models are no better than random guessing. This is another indication that our models are susceptible to significant variation based on threshold values.

Experiments with different thresholds Experiments using thresholds of 0.3, 0.4 and 0.5 are shown in Table 7.1. The table shows a clear trend that having lower prediction thresholds improves the models. It also makes the text variables more and more redundant as the threshold descends.

The difference in models including and excluding text features is significant when looking at McNemar’s test. The only exception from this is XGB at a threshold value of 0.3 where there are no significant differences between the models.

Overall, the models appear to have the best performance when using the threshold of 0.3, considering all evaluation metrics. If precision is considered, there is no significant change in the metric when using different thresholds. The significant change in performance comes from the recall. It is much higher when there is a lower prediction threshold. This implies that our model is uncertain of the predictions and that we must introduce the threshold as a bias to improve the model performance. The added bias helps the model gain a higher recall.

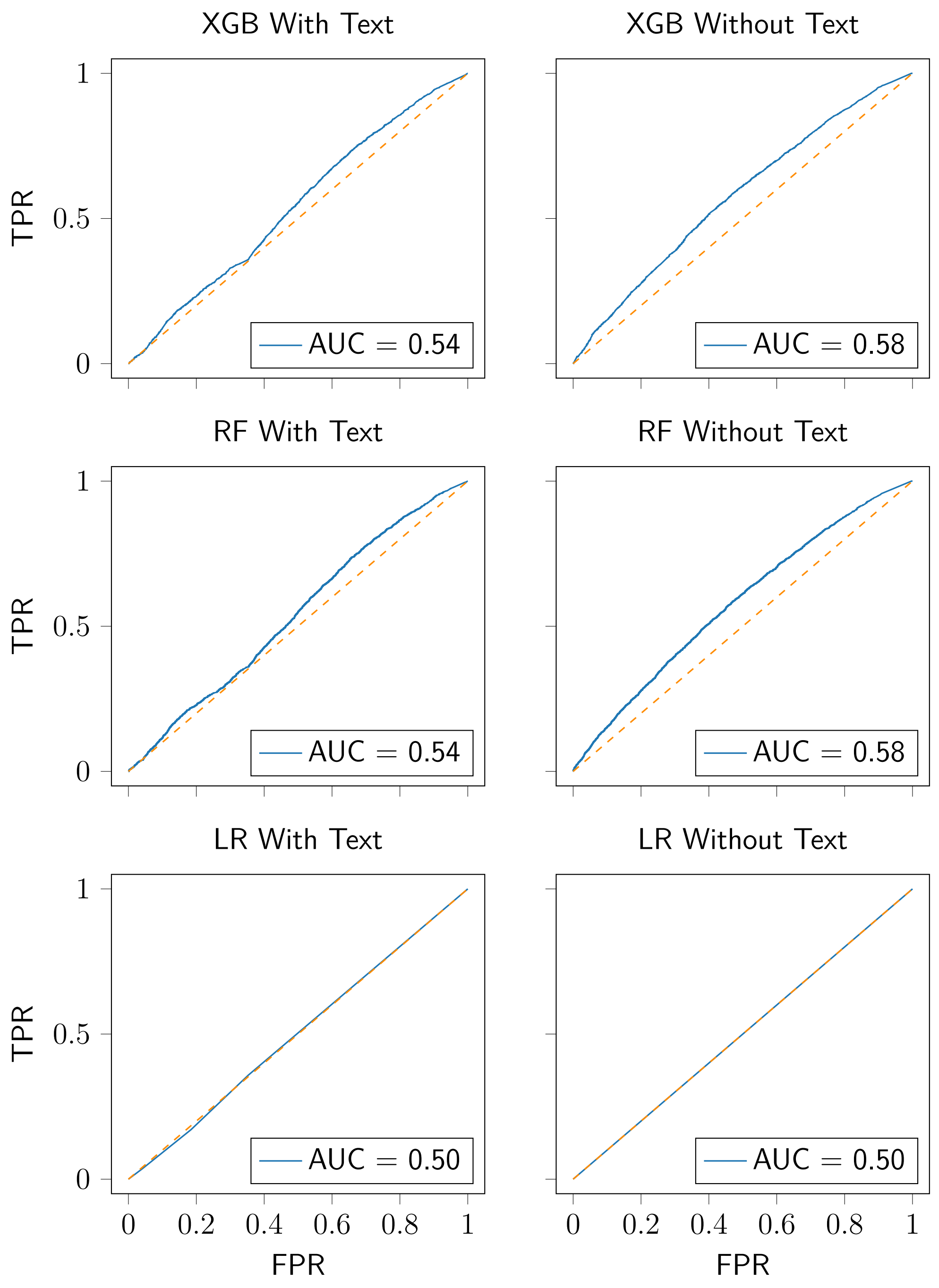


Figure 7.1: Combined ROC curve for all classifiers. The plots in the right column is the model without text. The orange dashed line represents an AUC = 0.50, a random guess.

An interesting result is that the logistic regression never predicts *up* movements, no matter how much bias is introduced. This suggests a better performing model including text, but this is not the case when the AUC is taken into consideration.

Table 7.1: Results of testing thresholds with other classifiers. McNemar's test statistics are displayed at the bottom of the table.

	Threshold 0.3			Threshold 0.4			Threshold 0.5		
Metric	**XGB**	**RF**	**LR**	**XGB**	**RF**	**LR**	**XGB**	**RF**	**LR**
Accuracy	0.5571	0.5553	0.5569	0.6127	0.6468	0.6661	0.6559	0.6658	0.6661
Recall	0.3470	0.3677	0.3464	0.2479	0.1080	0.0341	0.0895	0.0414	0.0341
Precision	0.3211	0.3261	0.3207	0.3499	0.3357	0.3005	0.3493	0.3203	0.3005
F_1	0.3335	0.3456	0.3331	0.2902	0.1634	0.0613	0.1425	0.0733	0.0613
REMOVING THE TEXTUAL COMPONENT FROM THE MODELS									
Accuracy	0.5666	0.5930	0.6806	0.6735	0.6767	0.6806	0.6806	0.6806	0.6801
Recall	0.5339	0.4281	0.0000	0.0655	0.0576	0.0000	0.0000	0.0000	0.0000
Precision	0.3747	0.3787	0.0000	0.4270	0.4518	0.0000	0.0000	0.0000	0.0000
F_1	0.4403	0.4019	0.0000	0.1135	0.1022	0.0000	0.0000	0.0000	0.0000
MCNEMAR'S TEST STATISTIC FOR MODEL WITH AND WITHOUT TEXT									
χ^2	1.0169	18.2608	247.3994	86.6674	44.2311	31.5270	40.9803	29.1082	31.5270
p-value	0.31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In addition, we investigate whether there is a significant difference between the models that include and exclude text at the different thresholds. Table 7.2 shows that there is a significant difference at a 5% level between the models at different thresholds. These results indicate that the best models are at the prediction threshold of 0.3. The models are also becoming significantly better when the prediction threshold decreases.

Table 7.2: Comparison of models with and without text at different thresholds using McNemar's test statistic and corresponding p-value

	0.3 vs 0.4		0.3 vs 0.5		0.4 vs 0.5	
Model	χ^2	p-value	χ^2	p-value	χ^2	p-value
XGB with text	261.9170	0.0	384.3971	0.0	141.4794	0.0
XGB without text	309.5726	0.0	315.0687	0.0	11.2044	0.0
RF with text	387.9018	0.0	404.0136	0.0	28.8373	0.0
RF without text	236.3153	0.0	249.1540	0.0	14.6574	0.0
LR with text	395.8082	0.0	395.8082	0.0	∞	0.0
LR without text	1325.8782	0.0	1325.8782	0.0	∞	0.0

7.2 Results of Other Classifiers

In this section, we present the performance metrics for each of the different models. Models without textual features are based on the same hyperparameter tuning as models with textual features. In this way, we see how the textual information affects the model with those hyperparameters exactly, not necessarily how it affects the method as a whole.

Table 7.3: Results of out-of-sample predictions using the best threshold 0f 0.3 from validation. The McNemar's test statistics is displayed at the bottom of the table.

Metric	**XGB**	**RF**	**LR**
Accuracy	0.5507	0.5483	0.5507
Recall	0.3551	0.3707	0.3551
Precision	0.3286	0.3312	0.3286
F_1	0.3414	0.3498	0.3414
REMOVING TEXT			
Accuracy	0.5571	0.5787	0.3279
Recall	0.5399	0.4388	1.0000
Precision	0.3773	0.3774	0.3279
F_1	0.4442	0.4058	0.4938
MCNEMAR'S TEST STATISTIC			
χ^2	0.9129	23.0672	860.0953
p-value	0.33	0.0	0.0

The results shown in Table 7.3 show how well the models perform out-of-sample. Models seem to perform better without textual features than with textual features. It is interesting to notice the shift in the predictions for logistic regression without textual features from the validation set to the test set. With the test set, logistic regression hit all *up* predictions with a precision of 33%. Considering the drop in accuracy, it implies that the model sacrifices correct *not-up* predictions to hit more *up* predictions.

Based on the findings shown in Table 7.3 we observe that XGBoost without text appears to be the best performing method, with a slightly higher F_1 score than the RF model. However, XGB does not have significant differences between the model with and without text. XGB and logistic regression seem to yield the exact same performance including textual features using a prediction threshold of 0.3.

7.3 Evaluation of the Different Models

In this section, we evaluate the different models with each other using McNemar's test. As mentioned, McNemar's test is a test to identify if the performance between two models is significantly different. If the p-value is lower than our significance level α, we discard our null hypothesis that two models have equal performance. Considering the values shown in Table 7.4, we note that there are significant differences at a 5% level of significance between all models. The models compared here are the models *with* textual data.

Table 7.4: Comparison of the different models with textual features with a threshold of 0.3 using McNemar's test using the p-value.

	XGB	RF	LR
XGB	-	0.02	0.00
RF	0.02	-	0.02
LR	0.00	0.02	-

7.3.1 Logistic Regression Coefficients

Since we have included logistic regression, we have the opportunity to explore the feature coefficients. These coefficients are shown in Table 7.5.

Table 7.5: Coefficients from the logistic regression including and excluding text. The numbers are rounded to four decimals. The coefficients are the changes in log odds for an *up* movement

Feature	With Text	Without Text
`return_-30`	0.0	0.0
`return_-25`	0.0	0.0
`return_-20`	0.0	0.0
`return_-15`	0.0	0.0
`return_-10`	0.0	0.0
`return_-5`	0.0	0.0
`distilbert_class_content`	0.4975	n/a
`distilbert_class_title`	0.4658	n/a
`finbert_sentiment_label_title`	0.0	n/a
`finbert_sentiment_label_content`	0.0	n/a
`Volume_minmax`	0.0	0.0

The coefficients show that the only variables that the logistic regression model relies on are the values of the fine-tuned DistilBERT model. From this, we see that when included, the logistic regression only values the textual features. The most important feature is the `distilbert_class_content` which gives an increase in log odds of 0.4975 for an *up* movement and the same decrease in log odds for a *not-up* movement. Both features drive a small increase in log odds, but they are also the only features that matter. The features extracted from DistilBERT are only the values -1 and 1, which conveys that the log odds either increase or decrease with regard to the results of DistilBERT. This signifies that the log odds for *up* movement in HPR can only increase by a maximum of 0.9633 if DistilBERT predicted both headlines and content as 1. Overall, this is a very small increase in log odds.

Compared with LR excluding textual features, we see that it does not find a pattern in the data. All the coefficients are 0.0 when excluding textual features. This suggests that the stock data do not explain the changes in stock prices. Therefore, the model only performs random guesses. The coefficient values indicate that the movement in HPR intraday is impossible to predict without including more information than previous stock price movements.

It would have been interesting to view the significance levels of the coefficients however, `scikit-learn` has no great integrations for viewing significance levels. The performance of the logistic regression is poor and it does not add additional value to explore the significance levels. The results of the logistic regression ROC curve show that it is not better than a random model. Based on the ROC curve, it is highly plausible that the text feature coefficients are not significant at all. Therefore, the logistic regression coefficients will not be discussed further.

7.4 Discussion

When evaluating our other classifiers, we have investigated different threshold values. This investigation shows that a threshold of 0.3 is the best for predicting *up* movements. The fact that the threshold must be set so low is interesting. We observe that recall increases when the threshold is decreased, but precision remains constant. These results indicate that the added bias with regard to the prediction threshold increases the number of false positives. The ROC curves in Figure 7.1 displays that FPR increases when recall increases. When considering the ROC curves, the FPR increases at a higher rate than TPR when the slope of the curve is lower than 1. As the slope becomes less steep when the prediction threshold decreases, it suggests a higher increase in FPR compared to TPR. This could imply that a more conservative model, with lower precision and recall, is better, but it is hard to quantify when we do not know by how much stock price falls if the model predicts *not-up*.

Testing significance By using McNemar's test, we have investigated where there are significant differences between models. We have observed that there are significant differences between the different thresholds for all models with and without text. This tells us that the thresholds are different from each other. We have also concluded that there are significant differences between the models with and without text across all thresholds, except for XGB. Finally, we found that there are significant differences between the three different models at threshold 0.3.

Comparing models with and without text We notice some strange behavior for the logistic regression in the test set. It went from a recall and precision of 0 to a recall of 1 and a precision of 0.33. This could imply that the model is not suited to the task at hand and struggles to be generalized. The two other classifiers, XGBoost and RF, perform as expected on the test set considering the metrics on validation data. Looking at the coefficients for logistic regression, we find that the only thing that matters is the results of Distilbert. The model sets all other coefficients to 0.

Since the models excluding textual features became significantly better than the models including them at a lower prediction threshold, it indicates that the data contained in the news articles are irrelevant in our time frame. By decreasing the prediction threshold, we add a bias telling the models what it should consider an *up* prediction. Since the predictions improve significantly when the prediction threshold is decreased, the probabilities of *up* movements are mostly quite low. Therefore, it can be argued that the models become more certain of some of their predictions when including textual features. This is because the models predict higher probabilities for some of the observations, including textual features. However, the predictions are best at lower prediction thresholds. Perhaps textual data do not help us understand the stock market when using a 20-minute time frame. It should be considered that a 20-minute time frame is too short for investors to respond properly to a news article.

7.5 Summary results of Other Classifiers

This chapter has presented the results of other ML classifiers. Unlike the FNN we have investigated using different threshold values for *up* predictions. The findings of the prediction threshold experiments have shown that a lower threshold seems to be better at predicting *up* movements in HPR. Although the models improved at lower prediction thresholds, the effect of textual features decreased. This indicates that the news articles are irrelevant when we decrease the prediction threshold. The results of traditional ML models indicate that the data contained in the news articles do not explain the movements of the stock price in a time frame of 20 minutes.

8 Overall Discussion

In this chapter, we discuss the results of our experiments conducted in Chapters 6 and 7. Furthermore, we discuss the results of our feature extraction in Chapter 5. We explore the sentiment extraction of FinBERT and compare the results of DistilBERT on the test set with other models.

In addition, we review the comparative effectiveness of various classifiers. Finally, we explore the practical applications and broader implications of our findings, incorporating ethical considerations to round off this chapter.

8.1 Results of BERT-Based Models

We have used two different BERT-based models to extract textual features to use in future predictions. Chapter 5 describes how the models were used and fine-tuned. This section discusses how they performed.

The textual features were the key aspects that allowed us to study the importance of text when making predictions on stock data. We have used FinBERT to extract the sentiment and a fine-tuned DistilBERT to perform a text classification. The text classification was fine-tuned to predict HPR 20 minutes after a news article was published. Given that these variables serve as representations of news articles, it is crucial to address their significance as well. Their significance is addressed in light of the EMH.

8.1.1 FinBERT

By themselves, the results of FinBERT are not that interesting. This is because the sentiment is only used as an input feature in later models. Our first thoughts were that sentiment alone could be a good enough representation of the news articles to combine with stock data for a classifier to make predictions.

When performing the sentiment analysis, we expected that a positive sentiment would drive an *up* movement of the related stock. For us, it makes sense that an article that has positive sentiment contains positive news about the stock and therefore investors will respond by buying the mentioned stock. This will increase the demand for the stock and, therefore, the corresponding price will increase, making HPR positive. Overall, the FinBERT results did not meet our expectations.

Although our expectations differed from our results, the results of FinBERT may still be sensible. Stock prices are inherently noisy, and the expectation that a positive sentiment of an article will make the price of the related stock increase is incorrect.

This is because a stock price reflects much more information than a news article's sentiment. Considering the EMH, using sentiment as a singular textual feature is not enough when EMH states that stock prices reflect all available information. The EMH is also the reason why the sentiment should still be included. A sentiment from a news article referencing a stock is information about the stock, and it should therefore add value to a model trying to predict stock price movements.

A compelling aspect of NLP in general is that human written text could be related to the author's opinion. Therefore, the sentiment of an article may be the authors' view on the related stock. It should be considered that the author's view of a stock may differ from how the market views and values the stock.

It should also be considered that FinBERT has predicted sentiments that are not true. Although Huang et al. (2020) proves that this FinBERT model has promising results, we have not had any true labels to test the performance of FinBERT on our data. It is used as is, and our only means of evaluation is by looking at the performance on labeled data. Huang et al. (2020) proves that it has a good performance, but we have to understand that these results may not be generalized to our data. We note that FinBERT is fine-tuned on 10,000 manually annotated sentences from analyst reports and that these sentiments might not be the same as sentiments in financial news. It is therefore up to later prediction models to see whether the sentiment from FinBERT matters for the predictions or not.

8.1.2 Fine-Tuned DistilBERT

To extract more textual features, we also fine-tuned a DistilBERT model. This model was tuned to predict HPR 20 minutes after the news article was published. The results of DistilBERT were interesting. To assess the performance of the models, we measured the accuracy, precision, recall, and F_1 score. On the validation set, it measured an accuracy of 60% on both headlines and content, which is somewhat good considering that it is only fine-tuned on a smaller random sample of the training data. The precision is only slightly higher than 30%. However, this model is only supposed to extract more textual features for later models to use.

Keeping our results in mind, DistilBERT does not perform well enough to be usable on its own. However, the goal of this thesis is not to beat the market, but rather explore whether textual data can explain the stock market. As shown in Table 5.5, DistilBERT is able to find different patterns in headlines and content. By considering an *up* prediction to be true if the model made a correct *up* prediction on either headlines or content, we achieve an accuracy of 76.23% on the validation set. In this context, DistilBERT is able to capture some patterns in the textual data, and it captures different patterns when considering headlines and content in isolation. Keeping the EMH in mind, this information should help capture the fundamental value of a stock.

DistilBERT could perhaps add value if used for fundamental analysis. Since financial news could contain information concerning the fundamental value of a stock, having a model with the ability to extract the effects of these values on individual stocks will be of great value to investors. It should also be considered that financial news may contain technical indicators. DistilBERT should be able to find patterns from technical indicators contained in the articles as well. It is a possibility that DistilBERT is able to capture some of this information, but the time frame is too small. This topic will be explored in greater detail in Section 8.3.

In our data set, we have 19 different stocks. Considering that there may be a difference in how articles regarding different stocks are written, we constructed a model only considering the AAPL stock. This was to assess whether the difference in writing regarding different stocks has confused our model. The performance during training is, however, almost identical to the model considering all the stocks in our data set. A possible reason for this is the fact that AAPL is the clearly most mentioned stock in our data set.

A source of error when using news articles to predict HPR is that the author's opinion of the stock may influence the article's view of the stock. This is due to the same reason as why the author's view may impact the sentiment of the article. Considering that our data set is news articles and not stock exchange announcements, the author's view is more important when applying NLP methods on news articles. This is because the author's view may be a source of noise that can confuse the model.

8.2 Comparing the Different Results

In this thesis, we have used many different models to understand the effect of text data on predicting intraday returns. Table 8.1 shows the results of all the methods we have used. The results of the fine-tuned DistilBERT on the test set are also included. This is because DistilBERT gave interesting results on validation data. By testing the DistilBERT using the test set, we can compare it to the other models using stock data and textual features.

Table 8.1: Combined results of all methods for models including textual features. This also includes the metrics for our two fine-tunes DistilBERT models on the test set, previously only metrics for the validation set were presented.

Metric	Title	Content	FNN	XGB	RF	LR
Accuracy	0.5972	0.6199	0.6639	0.5507	0.5421	0.5507
Recall	0.2108	0.1685	0.0389	0.3551	0.3750	0.3551
Precision	0.3122	0.3241	0.3139	0.3286	0.3319	0.3286
F_1	0.2517	0.2217	0.0693	0.3414	0.3521	0.3414

Comparing DistilBERT with the FNN, we see that DistilBERT converges the loss function during training. This is displayed in Figures C.1 and C.2, and indicates that the DistilBERT is able to find patterns in the text data.

When precision is considered in isolation, the best-performing model is a random forest classifier. This is somewhat interesting since we expected the FNN to perform the best due to its ability to perform well on noisy data. We also see that the rest of the non-deep learning methods are outperforming the FNN. There is the possibility that we have not managed to build a well functioning NN, but we must consider the fact that non-deep learning methods are still robust and high performing as well.

Considerations concerning threshold values When experimenting with XGBoost, random forests, and logistic regression, we tried different prediction thresholds. The results of the threshold experimentation indicate that the models perform better when we add some bias toward predicting *up* movements in HPR. The ROC curves confirm this indication with the exception of logistic regression. The best overall models are found when using the threshold of 0.3, but in a financial context, it can be argued that a threshold of 0.4 may give a better model considering that precision is a more important metric. The model with the highest precision is the random forest without text at threshold 0.4.

8.3 Textual Data for Market Comprehension

As textual data did not help to explain the stock market when used in conjunction with stock data, it should be considered how well text-based models perform in isolation from stock data. When comparing DistilBERT with other models that include stock data, we see that it outperforms our FNN. This could indicate that the BERT-based models are finding patterns that become clouded when used alongside stock data.

As financial news is a great source of information for investors, it must contain relevant information. We used FinBERT to extract the sentiment. A sentiment value should be a relevant feature since, in theory, it should influence how the reader values the stock. In our project, the time frame is either too small to see an effect of the sentiment, or the effect of the sentiment is too small. Our hypothesis that a positive sentiment should drive an increase in HPR was wrong, at least for the 20-minute time frame.

An issue may arise if news articles contain information on many different factors that drive stock prices. DistilBERT was made to give one classification based on the entire content of the article. This implies that DistilBERT must compress all the information contained in a news article into one single value. It could be argued that DistilBERT also captures the sentiment when performing a text classification on HPR, but not as well as FinBERT.

Considering these arguments, perhaps different models could be fine-tuned to extract different aspects from the news article. Splitting the DistilBERT model into several smaller models with more specific tasks could be a viable strategy. This suggests that several other specific features will be extracted from the text. DistilBERT could have been confused by the various different information contained in the news articles and therefore could not make more precise predictions. Extracting more specific textual features may help to further understand the stock market.

For deep learning models, the more information, the better. As the EMH states that a stock price reflects all available information on the stock, it suggests that we need more information. Having more information should, in theory, help explain how the stock market works. The questions are only about how the information can be extracted and what information should be extracted.

8.4 Real-World Applications and Implications

This section considers the criteria our model would have to match if it should be used for trading. The real-world application is more hypothetical, but it adds some consideration regarding relevant performance metrics in a model that only predicts *up* and *not-up* movements in HPR. The section dives further into the implications regarding our models and project in general. In the end, we discuss some ethical considerations if a similar model was successful.

8.4.1 Considerations for Real-World Application

Our models are not accurate enough to be applied in real-world scenarios. There are certain factors that must be considered if a model like ours is to be applied. For starters, we must consider the precision of the model. This is because we care more about how accurate the model is in predicting *up* movements. We would in theory need a precision of a minimum of 50% for the model to be usable.

If the precision was above 50% for *up* predictions, we would have a statistically higher chance of a true *up* prediction and a positive return than not. However, the model does not quantify the possible loss when the model is wrong. This must be taken into account if we are going to use the model in an investment strategy.

Alongside the possible loss, we must also consider the investors' appetite for risk. Risk-averse investors may need an even higher precision before they feel that their utility score[1] starts to increase, unless the expected return is large enough relative to the risk. When intraday trading we see from the charts regarding HPR in Chapter 4 that HPR after 20 minutes mostly have small movements. This is explained by a standard deviation of 0.002833. The expected return is 0.000019, which is the average HPR on investments placed right after the publication of an article and sold 20 minutes after publishing. If we then adjust the HPR to take into account the transaction costs of 0.001, we can calculate the utility score for investors.

Consider three investors, one risk-neutral ($A = 0$), one risk-seeking ($A = -4$), and one risk-averse ($A = 4$). A is the value referencing the investors' appetite for risk. If we input the expected return and standard deviation in the utility function, we can calculate the utility score for these investors.

[1]See Equation (2.2), p. 7.

Table 8.2: Calculation of utility scores using HPR 20 minutes after the publication of an article, adjusted for transaction costs, with regard to different investors' appetites for risk.

Investor	Utility Score
Risk-averse	−0.0066
Risk-neutral	−0.0098
Risk-seeking	0.0047

Considering the utility scores of the various investors, shown in Table 8.2, we see a positive utility score for the risk-seeking investor. This suggests that only risk-seeking investors should invest in intraday stocks.

Discussion considering the selected time frame It is possible that the time frame of 20 minutes after publishing is a bit narrow and that more investors may have had time to react if we considered a longer time frame. We saw that with some prototyping with a larger time frame, we managed to have a better performing model. We have also observed some indications that having a wider time frame after the publishing time may yield better results. However, to use a model like ours in a wider time frame, we must also be stricter regarding model performance. When the time frame expands, the potential loss is greater. Taking this into account, the model must yield even greater performance if risk-averse investors were to rely on its performance alone.

Following up on the results of Gidófalvi (2001) we see that news articles can predict the direction of a stock up to 20 minutes after the article has been published. He also states a strong correlation between news articles and the behavior of stock prices from 20 minutes prior to 20 minutes after the publication of a news article.

If we consider our results compared to theirs, the answer might not be as clear. Even if he found a correlation in 2001, a lot has changed since then. The biggest change, by far, is the Internet. It must be considered that news articles could have a stronger correlation with stock price behavior in 1999–2000 compared to 2009–2020. This might be related to the faster flow of information in 2009–2020 and the much larger number of news articles published. Today, investors have to consider more information than investors did around the year 2000. Considering that some news may also be fake news, investors must also be more critical of their news sources. It is important to take these arguments into consideration when determining the optimal prediction time frame.

8.4.2 Psychological and Ethical Considerations

In this thesis, we have attempted to create a model that is able to predict intraday stock returns. This is theoretically a very interesting topic and would be a very interesting model. However, we consider this quote from Dr. Ian Malcomn in the 1993 movie Jurassic Park: “Your scientists were so preoccupied with whether they could, they didn’t stop to think if they should”. Only because something is possible does not insinuate that it should be done. One must really think about the consequences of what such an achievement would imply.

If our model were successful and made publicly available, it could disrupt the market. The question will then consider how long the model would work. If everyone uses the model, it could be that the model stops working since the world it was trained to model no longer works in the same way. In this way, the model creates its downfall.

Another important consideration is that such a model could create an economic bubble. If everyone follows the model, it could lead to an increase in herd mentality. The model could make it so that investors believe that the market is in a better state than it actually is.

The performative characteristics of economical models Callon (2007) states that economic models do not model reality but rather form it. The article states that the black-scholes formula for option valuation performed rather poorly when it was introduced, but got better with time. Based on this, it could be the case that a successful prediction model could become the most commonly used model for valuation, and itself form reality.

Although it is important to consider the potential ethical implications of developing a model to predict intraday stock returns, the scope of this thesis is limited to the technical aspects of developing the model. The ethical and psychological implications of such a model are vast and complex and require thorough investigation beyond the scope of this thesis. For example, the use of such a model could potentially lead to unfair advantages for those who have access to it, widening the gap between the wealthy and the less fortunate. It could also encourage a short-term perspective on investments, which could have negative consequences for the long-term stability of the stock market.

Furthermore, the psychological effects of relying on a model to make investment decisions could lead to emotional detachment from the stock market. A detachment that can result in a lack of accountability and responsibility. These are all important considerations that warrant further investigation and analysis beyond the technical development of the model itself.

8.4.3 Summary of Real-World Applications and Implications

Investors have different appetites for risk. The calculation of utility scores suggests that intraday investing should be avoided in general unless the investor is risk-seeking. The expected return is negative when adjusting for transaction costs, and this strengthens the suggestion. In this regard, we can suggest that a well-functioning model should help investors achieve a higher return on their investments. The model should locate investments with the highest probability of achieving returns above the transaction cost. It must be noted that this model would have to quantify both the potential return and loss adjusted for the transaction costs.

Regarding the time frame, we discussed the results of Gidófalvi (2001). The time frame of 20 minutes may be a bit narrow since information flows faster in the years 2009–2020 compared to the years 1999–2000. Investors may need more time to fully process the amount of information, and 20 minutes is not a sufficient amount of time to complete this processing and placing an investment.

As discussed in this section, this kind of market modeling and prediction has some potential drawbacks. Most important is the discussion of whether a model could model the market accurately. We note that unlike a *valuation* formula, such as the black-scholes formula, the model we outline in this thesis is a *prediction* model. It is a prediction model with the possibility of changing the market. We argue that building a model that changes the market is a hard, if not impossible, task. A prediction model that changes the market it predicts cannot continue to accurately predict the market.

9 Conclusion

In this thesis, we have investigated the effect of news articles on the stock market on an intraday basis. We used a data set with news articles about stocks on the NASDAQ Stock Exchange. From the thousands of stocks in NND we considered the 19 most mentioned stocks and built a data set with intraday price movements in 5-minute intervals from -30 minutes to -5 minutes in relation to the time of publication of the news article. From the news articles, we performed a sentiment analysis and a text classification to extract textual features. These data were used to predict HPR 20 minutes after a news article was published.

Research Question 1

Our first research question to answer was: "Does textual data help predict the direction of stock returns on intraday data?". The answer to this question can help us understand whether textual data have an effect and how much textual data can affect movements in intraday stock prices. We observe that textual data in some way can help predict directional movements on an intraday basis. This is mainly due to the results of our DistilBERT models. On headlines, DistilBERT achieved a precision of 31.22% and a recall of 32.41%. On content, it achieved a precision of 21.08% and a recall of 16.85%. This is the precision and recall for class 1, *up* movements, on the test set. If the models are combined and we consider an *up* prediction to be true if either headlines or content is predicted to be up *up*, DistilBERT achieves an accuracy of 76.23%. Note that this result is from the validation set, but it is still an out-of-sample result for the DistilBERT model. This suggests that textual data help predict the direction of stock returns on intraday data and thus that textual data help explain the stock market.

Research Question 2

To measure how accurately a model using textual features can predict movements, we answer the second research question: "Can a machine learning model including textual features predict the direction of a stock price movement 20 minutes after the news has been published?". This helps to understand if our model performs well enough to be implemented in a trading strategy. On the basis of our findings, it is hard to answer yes to this question. We still believe that textual data could be part of the puzzle to predict intraday returns, but there are still many missing variables.

From our findings, we observe that tree-based models seem to yield the best results in predicting intraday stock returns using NLP, with a precision of approximately

33% and a recall of approximately 36%. If we compare our results with previous work in the field, the accuracy is comparable, but we go further and also present how accurately we can predict a *significant up* movement.

Research Question 3

Lastly, we ask "Does textual data clear up some of the noise in stock data?". The last research question is to facilitate a more philosophical discussion. As mentioned in the answer to research question 1, we believe that it does clear up some noise. From a theoretical point of view, textual data should clear up some noise, and based on our findings, we believe this to be true.

Concerning this final research question, we also point out some of the concerns presented in Section 8.4.2. Although the text does clear up some of the noise, it does not indicate that it could or should do it forever. It could be the case of a self-fulfilling prophecy. Further research in this field should consider whether the field of intraday stock prediction is a feasible field of study.

A Final Consideration

Before continuing to suggestions for future work, we state one final point: If one's goal is to earn money, we would recommend not buying intraday stocks, unless you are risk-seeking. On the basis of our findings, textual data do not seem to clean up enough noise in intraday data. If you wish to earn money, either you must know something that the others do not, or you should look at longer time frames for investing.

9.1 Future Work

This thesis has started work in the field of predicting the directional movements of stocks on an intraday basis. In our thesis, we have discovered more aspects of the field, and we are now going to make some suggestions for further work in the field. As mentioned in Section 8.4.2, future researchers should consider how smart it is to develop long-term prediction models for the market.

Increasing the number of stocks We noticed that many news articles were published outside the opening hours of the NASDAQ Stock Exchange and the New York Stock Exchange. One possibility to gain more data would be to include data from multiple different stock exchanges, such that one could capture as many news articles as possible.

Investigating the selected time frame We have observed some promising results in terms of a longer time frame. This is likely due to the fact that a stock will fluctuate more in a larger time frame. Another project we have noticed, which has not yet[1] been published, investigates in what time frames the correlation between news articles and stock price movements is strongest. They find that this correlation is strongest in larger time frames of up towards two hours.

Another possibility in terms of time frame is to investigate whether a stock price has increased significantly *during* a larger time period. Our models only look at a snapshot *exactly* 20 minutes after an article was published. This works for a small time frame but might not be as suited when the time frame is expanded. If we look at a time frame of 60 minutes, it might be that the stock price had increased significantly at minute 42, but went down in the last minutes, resulting in the model not predicting *up*.

Increasing the number of textual features DistilBERT has to extract a lot of information from each article and only return a single value to represent all this information. Perhaps, using similar models to extract specific information from news articles is an option. The models should be better when having a more specific task rather than truncating an entire article into one feature. We propose using techniques such as topic modeling or latent Dirilecht allocation to capture more information from the texts.

Use other transformers models or neural networks It could be possible to construct the data set in another way so that sequence-based neural networks could work and perform better than our FNN. One could also investigate other transformers models such as other BERT models or GPT models.

Inclusion of other financial features We have not investigated the inclusion of other financial features than the HPR and volume. These features could be better combined with textual features to create superior models.

[1]At the time of writing

This page intentionally left blank.

References

Aggarwal, Charu C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing. ISBN: 978-3-319-94463-0. DOI: 10.1007/978-3-319-94463-0. URL: http://link.springer.com/10.1007/978-3-319-94463-0 (visited on May 9, 2023).

Alostad, Hana and Hasan Davulcu (Dec. 6, 2015). "Directional Prediction of Stock Prices Using Breaking News on Twitter". In: *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). Vol. 1. Journal Abbreviation: 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), pp. 523–530. DOI: 10.1109/WI-IAT.2015.82.

Basilone, Ryan (2021). "The Impact of News on Stock Market Investors". In: *SSRN electronic journal*. DOI: https://dx.doi.org/10.2139/ssrn.3908662.

Bengfort, Benjamin (2018). *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning*. Beijing: O'Reilly. ISBN: 978-1-4919-6304-3.

Bodie, Zvi, Alex Kane, and Alan J Marcus (2021). *Investments*. Twelfth edition, International student edition. The McGraw-Hill Education series in finance, insurance, and real estate. New York: McGraw-Hill Education. ISBN: 1-260-57115-7.

Bridle, John (1989). "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters". In: *Advances in neural information processing systems* 2.

Callon, Michel (2007). "What does it mean to say that economics is performative?" In: *Do economists make markets? On the performativity of economics*, pp. 311–357.

Chandola, Deeksha et al. (Aug. 1, 2022). "Forecasting Directional Movement of Stock Prices using Deep Learning". In: *Annals of Data Science*. ISSN: 2198-5812. DOI: 10.1007/s40745-022-00432-6. URL: https://doi.org/10.1007/s40745-022-00432-6.

Chen, Tianqi and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. event-place: San Francisco, California, USA. New York, NY, USA: ACM, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

Cheng, Shou-Hsiung (2010). "Forecasting the Change of Intraday Stock Price by Using Text Mining News of Stock". In: *2010 International Conference on Machine*

Learning and Cybernetics. Vol. 5, pp. 2605–2609. DOI: 10.1109/ICMLC.2010.5580879.

Deveikyte, Justina et al. (2020). *A Sentiment Analysis Approach to the Prediction of Market Volatility*. DOI: 10.48550/ARXIV.2012.05906. URL: https://arxiv.org/abs/2012.05906.

Devlin, Jacob et al. (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

Dietterich, Thomas Glen (1995). "Overfitting and Undercomputing in Machine Learning". In: *ACM computing surveys (CSUR)* 27.3, pp. 326–327.

– (1998). "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". In: *Neural computation* 10.7, pp. 1895–1923.

Eschenbach, Warren J. von (Dec. 1, 2021). "Transparency and the Black Box Problem: Why We Do Not Trust AI". In: *Philosophy & Technology* 34.4, pp. 1607–1622. ISSN: 2210-5441. DOI: 10.1007/s13347-021-00477-0. URL: https://doi.org/10.1007/s13347-021-00477-0.

Fama, Eugene F (1965). "The Behavior of Stock Market Prices". In: *The journal of Business* 38.1. Publisher: JSTOR, pp. 34–105.

Ferreira, Iuri H. and Marcelo C. Medeiros (2021). *Modeling and Forecasting Intraday Market Returns: a Machine Learning Approach*. _eprint: 2112.15108. URL: https://doi.org/10.48550/arXiv.2112.15108.

Fong, Bryan (July 5, 2021). "Analysing the Behavioural Finance Impact of 'Fake News' Phenomena on Financial Markets: A Representative Agent Model and Empirical Validation". In: *Financial Innovation* 7.1, p. 53. ISSN: 2199-4730. DOI: 10.1186/s40854-021-00271-z. URL: https://doi.org/10.1186/s40854-021-00271-z.

Ghosh, Pushpendu, Ariel Neufeld, and Jajati Keshari Sahoo (May 1, 2022). "Forecasting Directional Movements of Stock Prices for Intraday Trading Using LSTM and Random Forests". In: *Finance Research Letters* 46, p. 102280. ISSN: 1544-6123. DOI: 10.1016/j.frl.2021.102280. URL: https://www.sciencedirect.com/science/article/pii/S1544612321003202.

Gidófalvi, Gyözö (2001). "Using news articles to predict stock price movements". In.

Hastie, Trevor J, Robert J Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer series in statistics. New York: Springer Science + Business Media. ISBN: 978-0-387-84857-0.

Hovy, Eduard H. (2015). "What are Sentiment, Affect, and Emotion? Applying the Methodology of Michael Zock to Sentiment Analysis". In: *Language Production, Cognition, and the Lexicon*. Ed. by Núria Gala, Reinhard Rapp, and Gemma Bel-Enguix. Springer International Publishing, pp. 13–24. ISBN: 978-3-319-08043-7.

DOI: 10.1007/978-3-319-08043-7_2. URL: https://doi.org/10.1007/978-3-319-08043-7_2.

Huang, Allen, Hui Wang, and Yi Yang (2020). "FinBERT—A Deep Learning Approach to Extracting Textual Information". In: *SSRN Electronic Journal*. ISSN: 1556-5068. DOI: 10.2139/ssrn.3910214. URL: https://www.ssrn.com/abstract=3910214 (visited on Feb. 13, 2023).

Jabbar, H and Rafiqul Zaman Khan (2015). "Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)". In: *Computer Science, Communication and Instrumentation Devices* 70, pp. 163–172.

James, Gareth et al. (2021). *An Introduction to Statistical Learning : With Applications in R*. Second edition. Springer texts in statistics. New York: Springer. ISBN: 978-1-07-161417-4.

Janocha, Katarzyna and Wojciech Marian Czarnecki (2017). *On Loss Functions for Deep Neural Networks in Classification*. _eprint: 1702.05659.

Kavzoglu, Taskin (Jan. 1, 2017). "Chapter 33 — Object-Oriented Random Forest for High Resolution Land Cover Mapping Using Quickbird-2 Imagery". In: *Handbook of Neural Computation*. Ed. by Pijush Samui, Sanjiban Sekhar, and Valentina E. Balas. Academic Press, pp. 607–619. ISBN: 978-0-12-811318-9. DOI: 10.1016/B978-0-12-811318-9.00033-8. URL: https://www.sciencedirect.com/science/article/pii/B9780128113189000338.

Kavzoglu, Taskin and Ismail Colkesen (June 1, 2013). "An Assessment of the Effectiveness of a Rotation Forest Ensemble for Land-Use and Land-Cover Mapping". In: *International Journal of Remote Sensing* 34, pp. 4224–4241. DOI: 10.1080/01431161.2013.774099.

Khadjeh Nassirtoussi, Arman et al. (Jan. 1, 2015). "Text Mining of News Headlines for FOREX Market Prediction: A Multi-layer Dimension Reduction Algorithm with Semantics and Sentiment". In: *Expert Systems with Applications* 42.1, pp. 306–324. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2014.08.004. URL: https://www.sciencedirect.com/science/article/pii/S0957417414004801.

Ko, Ching-Ru and Hsien-Tsung Chang (Mar. 11, 2021). "LSTM-Based Sentiment Analysis for Stock Price Forecast". In: *PeerJ Computer Science* 7, e408. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.408. URL: https://peerj.com/articles/cs-408.

LeCun, Yann A. et al. (2012). "Efficient BackProp". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3. URL: https://doi.org/10.1007/978-3-642-35289-8_3.

Li, Yang and Tao Yang (2018). "Word Embedding for Understanding Natural Language: A Survey". In: *Guide to Big Data Applications*. Ed. by S. Srinivasan. Springer International Publishing, pp. 83–104. ISBN: 978-3-319-53817-4. DOI: 10.1007/978-3-319-53817-4_4. URL: https://doi.org/10.1007/978-3-319-53817-4_4.

Liu, Yifan et al. (2017). “Stock Volatility Prediction Using Recurrent Neural Networks with Sentiment Analysis”. In: *Advances in Artificial Intelligence: From Theory to Practice*. Ed. by Salem Benferhat, Karim Tabia, and Moonis Ali. Cham: Springer International Publishing, pp. 192–201. ISBN: 978-3-319-60042-0. DOI: `https://doi.org/10.1007/978-3-319-60042-0_22`.

McNemar, Quinn (June 1, 1947). “Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages”. In: *Psychometrika* 12.2, pp. 153–157. ISSN: 1860-0980. DOI: `10.1007/BF02295996`. URL: `https://doi.org/10.1007/BF02295996`.

Moghaddam, Amin Hedayati, Moein Hedayati Moghaddam, and Morteza Esfandyari (2016). “Stock Market Index Prediction Using Artificial Neural Network”. In: *Journal of Economics, Finance and Administrative Science* 21.41, pp. 89–93. ISSN: 2077-1886. DOI: `https://doi.org/10.1016/j.jefas.2016.07.002`. URL: `https://www.sciencedirect.com/science/article/pii/S2077188616300245`.

Montesinos López, Osval Antonio, Abelardo Montesinos López, and Jose Crossa (2022). “Fundamentals of Artificial Neural Networks and Deep Learning”. In: *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer International Publishing, pp. 379–425. ISBN: 978-3-030-89010-0. DOI: `10.1007/978-3-030-89010-0_10`. URL: `https://doi.org/10.1007/978-3-030-89010-0_10`.

Patel, Jigar et al. (Jan. 1, 2015). “Predicting Stock and Stock Price Index Movement Using Trend Deterministic Data Preparation and Machine Learning Techniques”. In: *Expert Systems with Applications* 42.1, pp. 259–268. ISSN: 0957-4174. DOI: `10.1016/j.eswa.2014.07.040`. URL: `https://www.sciencedirect.com/science/article/pii/S0957417414004473`.

Pearson, Karl (1904). *Mathematical Contributions to the Theory of Evolution. XIII. On the Theory of Contingency and Its Relation to Association and Normal Correlation*. Biometric series 1. London: Dulau and Co.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Sanh, Victor et al. (2019). “DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter”. In: *arXiv preprint arXiv:1910.01108*.

Seng, Jia-Lang and Hsiao-Fang Yang (Jan. 1, 2017). “The Association between Stock Price Volatility and Financial News – A Sentiment Analysis Approach”. In: *Kybernetes* 46.8. Publisher: Emerald Publishing Limited, pp. 1341–1365. ISSN: 0368-492X. DOI: `10.1108/K-11-2016-0307`. URL: `https://doi.org/10.1108/K-11-2016-0307`.

Shmueli, Galit et al. (2020). *Data Mining for Business Analytics : Concepts, Techniques, and Applications in Python*. Hoboken, New Jersey: Wiley. ISBN: 1-119-54984-1.

Sokolova, Marina, Nathalie Japkowicz, and Stan Szpakowicz (2006). “Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation”. In: *AI 2006: Advances in Artificial Intelligence*. Ed. by Abdul Sattar and Byeong-ho Kang. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1015–1021. ISBN: 978-3-540-49788-2.

Song, Yue-Gang, Yu-Long Zhou, and Ren-Jie Han (2018). *Neural networks for stock price prediction*. arXiv: 1805.11317 [q-fin.ST].

Tharwat, Alaa (2020). "Classification Assessment Methods". In: *Applied Computing and Informatics*. Publisher: Emerald Publishing Limited.

UFLDL Tutorial (2023). *Optimization: Stochastic Gradient Descent*. Optimization: Stochastic Gradient Descent. URL: http://deeplearning.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/ (visited on Mar. 24, 2023).

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Willding, D. et al. (2018). *The Impact of Digital Platforms on News and Journalistic Content*. Tech. rep.

This page intentionally left blank.

A Tools

This appendix describes what hard- and software was used to achieve our results. We describe the specifications of CPU, GPU and RAM, as well as version number of related software.

A.1 Hardware

The following table displays what hardware was used to achieve our main results:

Component	Type
CPU	AMD Ryzen 3600 6-Core 3.6GHz
GPU	ASUS NVIDIA GeForce GTX 1660 SUPER 6GB GDDR6
RAM	16GB DDR4 2666Mhz

We note that we did run one model on a slightly different setup, where we used a NVIDIA Quadro GPU. That GPU ran a slightly different CUDA version. After some testing, we assume that this did not make a difference in predicitons.

A.2 Software

The software used to make the predictions is as follows:

Software	Version
`python`	3.10.9
`pandas`[1]	1.5.3
`numpy`	1.24.2[2]
`PyTorch`	1.13.1
`PyTorch Lightning`	1.9.4[3]
`torchmetrics`	0.11.3[4]
`scikit-learn`[5]	1.2.1
`xgboost`[6]	1.7.4
`transformers`	4.26.1[7]

[1] `https://pandas.pydata.org/pandas-docs/version/1.5/index.html`
[2] `https://numpy.org/doc/1.24/index.html`
[3] `https://lightning.ai/docs/pytorch/1.9.4/`
[4] `https://torchmetrics.readthedocs.io/en/stable/`
[5] `https://scikit-learn.org/stable/`
[6] `https://xgboost.readthedocs.io/en/stable/python/python_api.html`
[7] `https://huggingface.co/docs/transformers/v4.26.1/en/index`

B Preprocessing Details

This appendix describes how the preprocessing was done in greater detail. It also explains some of the challanges we faced when constructing our final dataset. Parts of this appendix is repeating what is mentioned in Sections 4.1 and 4.2.

In Appendix B.1 we illustrate how the text from a news article is represented in the NND.

NASDAQ News Dataset

The NND was a 6GB `.json`-file containing approximatly 1.4 million news articles from various sources regarding stocks on the NASDAQ stock exchange. The size of the dataset meant that it did not fit into our RAM, so we used chunks. For each chunk we removed columns `article_link`, `author_link`, `author_name`, `related_articles` and `appears_in`. This was to reduce the size of the dataset.

After dropping the columns we had to tackle the different symbols (tickers) in the dataset. This is the part where we filter out the tickers we don't want. Some articles had multiple tickers associated with it, and our solution here was to explode the dataset, meaning that we only have one ticker on each row. This meant that the same news article can appear in multiple rows. Since we knew that Facebook changed ticker from FB to META we changed all occurrences of FB to META in the dataset, this did not matter as we were unable to acquire the stock data for META.

Another nuisance with the NND was the fact that some of the columns contained dictionaries. This meant that we had to extract the information from the dictionary, and then rename the new columns. The columns [`$oid`, `$date`] became [`id`, `date`].

Finally after this process we merged the different chunks into a new dataframe before formatting the dates to have a `datetime` format to be able to merge with the RSD. We also set each publishing time to the closest whole minute so we can match as many publishing times as possible with the intraday stock data in the RSD.

We note that the column for symbols (ticker) were missing in the last chunks of the dataset. The dataset had no detailed explanation as to why this was the case, so the meaning of the columns had to be interpreted from content and heading.

Refinitiv Stock Dataset

The RSD came more complete than the NND. It was retrieved from Refinitiv Datascope Select. Since we are looking at minutely data, we assume that the last price from $t-1$ is equal to the opening price of t. The preprocessing of the RSD is mainly finding out how to handle missing values.

Our solution was to brute-force it by forward-filling all missing values to be the same as the previous valid value. This meant that we also forward-filled when the exchange was closed. It works for 19 stocks, but if the stock number increases, the dataset increases with almost triple.

Merging the Datasets

After cleaning the NND and RSD we merge the datasets on common time. We set a list of desired lags, and then merge on those lags as well. In our thesis, the selected timestamps were $[-30, -25, -20, -15, -10, -5, 20]$. The merging was done by removing/adding the specified number of minutes to the whole RSD and then merging. During the merging, we also remove duplicates and N/A values. Table B.1 displays how many observations remain at the different steps in the merging process.

Table B.1: Removed observations at each point in the merging process.

Merging step	Remaining observations	Change
Cleaning news data	490,140	-
Merging with stock data	490,140	-
Dropping duplicates	489,466	-674
Dropping N/A	169,769	$-319,697$
Removing closed times	55,952	$-263,745$

After merging, we generated the returns for all timesteps with regards to time zero. Finally we had to filter out the times where the exhanges was not open, as well as 30min at the start and 30min at the end of opening hours. This was done to take into account the fact that we use those timestamps in our lags.

Troubles regarding timezones. One key aspect to consider when working with daily data is time and timezones. We have to make sure that the data from NND and RSD is in fact merged on the exact same time. When we got the data we assume that it was on UTC, since the time column in both dataset had a trailing `Z`. This made it so that we could merge on time without too much hazzle. The trouble began when we had to consider daylight savings time. We do not want to have data for when the stock exchange were not open, so we had to figure out when than was. Luckily the data provided by refinitiv had a column named `GMT Offset`. This column was either -4 or -5. This is where another problem arises. Does that offset take into consideration daylight savings time? After some research it seems like GMT does in fact not have summertime, and is the same as UTC. It is nevertheless a weakness of the refinitiv database, and it should have given an UTC offset rather than GMT offset.

The processing of removing the time where the exchange was closed then had the extra step of filtering out different UTC timestamps depending on the value of the column `GMT Offset`. This seems like a straightforward task, but requires some nontrivial logic where we first create one dataframe for GMT -4 and another for GMT -5, where we then filter based on three OR statements to not only filter on whole hours, but also minutes.

B.1 Example News Article

Below, we present one news article without changing the formatting. As we can see, the text seems to be scraped from the website, as there are some strange line breaks. The length of this article is 804 words (tokens). This is slightly above average.

Date

`2014-10-15 14:30:00+00:00`

Ticker

`MSFT, AAPL`. Note that this is the tickers associated with the article in our *final data set*. In the raw NND, it might have more tickers associated with it.

Article Title

`Should Microsoft Investors Worry About The Surface Pro 3?`

Article Content

Note that some of the lines is too long to display on the page. We have not made any corrections to this.

```
Reportedly, the Surface Pro 3 is struggling, but its success
isn’t essential to Microsoft. Source: MicrosoftA recent

Digitimes

article

raised eyebrows in regards to
Microsoft

’s beleaguered Surface tablet. According to upstream supply chain
sources, the Surface tablet line has created US losses of about
$1.7 billion. However, the biggest takeaway was related to its
heavily marketed Surface Pro 3:With the factors above, the sources believe sales of the
Surface Pro 3 are unlikely to surpass one million units, adding
that Microsoft is also not very aggressive about development of
a next-generation Surface and is likely to terminate the
product line.Microsoft quickly pushed back on the rumor with a
blog post

from Brian Hall, the General Manager of Surface. Under the ending
```

```
subhead, "Businesses can buy with confidence. We are here to
stay," Mr. Hall quotes Microsoft CEO Satya Nadella's strong
support for the device: "Microsoft is putting its full and
sustained support behind the ongoing Surface program as one of a
number of great hardware choices for businesses large and small."
It is apparent that Microsoft is committed to the Surface line,
but should they be?Microsoft's Surface struggles



After two years - and three iterations --  with no
meaningful results from the product, the question for investors
is should Microsoft abandon the product. And on the surface (pun
shamelessly intended), it seems like the Surface abandoning crowd
has a point. If the numbers provided by Digitimes are correct
(Microsoft doesn't release specific product results, so
third-party results are the only source of estimates), then
Microsoft can enrich shareholders simply by exiting the business
(after the costs of winding down the business are expensed).Recently, CEO Satya Nadella appeared to distance himself f
the Surface line by calling Microsoft's hardware a "supporting"
business. Analysts and observers have closely followed Nadella's
comments about the Surface, partly because the tablet line was
former-CEO Steve Ballmer's idea and mostly because poor sales
results for the Surface RT were widely considered the
"final straw," leading to Ballmer's exit.

Nadella hedged on Ballmer's characterization of a "devices and
services" company instead choosing to refer to Microsoft as a
"mobile first, cloud first" company.Devices are flashy, but not essential to Microsoft



Lost in the discussion of whether or not Microsoft should keep
the Surface tablet lies a rather important fact: Microsoft isn't
dependent upon devices for revenue or earnings. Unlike fellow
device-makers
Samsung

and
Apple

, Microsoft isn't dependent upon gadgets. Last fiscal year,
Microsoft's "Devices and Consumer Hardware" segment only provided
13.4% of total revenue. For perspective, Apple's iPhone along
provides over 50% of its revenue.So while the news covers the struggles its devices face -
Windows Phone's distant third place in worldwide smartphone
operating systems, Xbox One's losing battle against
Sony

's PlayStation 4, and the aforementioned Surface's woes -
shareholders continue to benefit from Microsoft's strong overall
performance.As a matter of fact, since the beginning of 2012, Microsoft's
stock holds up extremely well against Apple although their
devices since then have had two widely differing paths:MSFT

data by
YChartsAs you can see, Apple's provided a negligibly better return
since then by capital appreciation of nearly 68% versus 65% for
Microsoft. However, when compared to the greater market (as
indicated by the S&P 500), both are doing well for
shareholders. So as you can see, although devices are flashy and
interesting to discuss, Microsoft doesn't need to be a device
leader to enrich shareholders.Final thoughts
```

```
Microsoft continues to struggle with its devices, but that's not
what investors should focus on. In its "Commercial Other"
division, the company reported $7.5 billion in sales last fiscal
year -- that's where Microsoft books its cloud-based revenue. And
although currently that's only 9% of revenue, it grew 33.3% on a
year-over-year basis. Satya Nadella wisely described Microsoft as
a "cloud first" company, investors would be wise to pay more
attention to Microsoft's moves in the cloud than in devices.Apple Watch revealed: The real winner is
inside



Apple recently revealed the product of its secret-development
"dream team" -- Apple Watch. The secret is out, and some early
viewers are claiming its everyday impact could trump the iPod,
iPhone,
and

the iPad. In fact, ABI Research predicts 485 million
of this type of device will be sold per year. But one small
company makes Apple's gadget possible. And its stock
price has nearly unlimited room to run for early in-the-know
investors. To be one of them, and see where the real money is
to be made, just
click here

!The article
Should Microsoft Investors Worry About The
Surface Pro 3?

originally appeared on Fool.com.Copyright Â© 1995 - 2014 The Motley Fool, LLC. All rights
reserved. The Motley Fool has a

disclosure policy


.The views and opinions expressed herein are the views and opinions of the author and do not necessarily reflect those of Na
```

This page intentionally left blank.

C Evaluation Graphs From Training Networks

This appendix shows how different networks were tuned. We first describe the fine-tuning of DistilBERT before we describe the varios NNs.

C.1 Fine-Tuning of DistilBERT

The tuning in Figure C.1 is trained on 30 epochs on article headlines. The loss seems to be converging to about 0.05. The tuning on content is trained on 50 epochs. Figure C.2 displays the metrics from tuning. As we can see, it takes quite a few more steps for the train loss to converge downward. We observe that the graphs for training on content and headlines have approximately the same shape, but the model trained on content takes more steps to achieve the results. This is natural, as it has more data to fit to. The evaluation metrics from training is displayed in Table C.1.

Both figures also show the development of the features from the validation set.

Table C.1: Evaluation metrics from fine-tuned DistilBERT during training. Note that these metrics are micro-average values across both classes.

Metric	Headlines	Content
Loss	3.1525	2.8256
Accuracy	0.6620	0.6672
F1	0.3016	0.3017
Precision	0.3301	0.3474
Recall	0.2776	0.2666

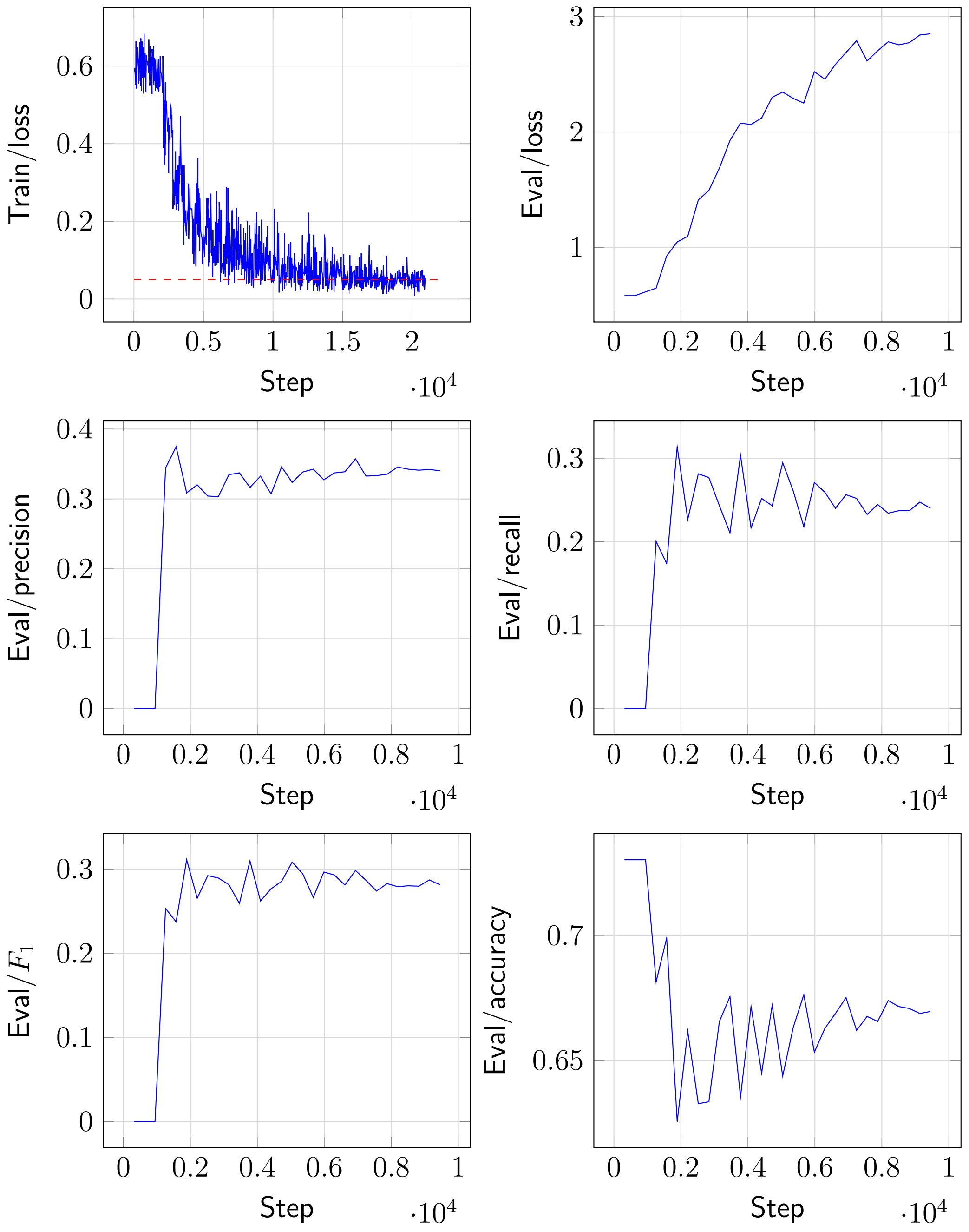


Figure C.1: Evaluation metric graphs from finetuning DistilBERT on headlines. The red line in loss graph is $y = 0.05$.

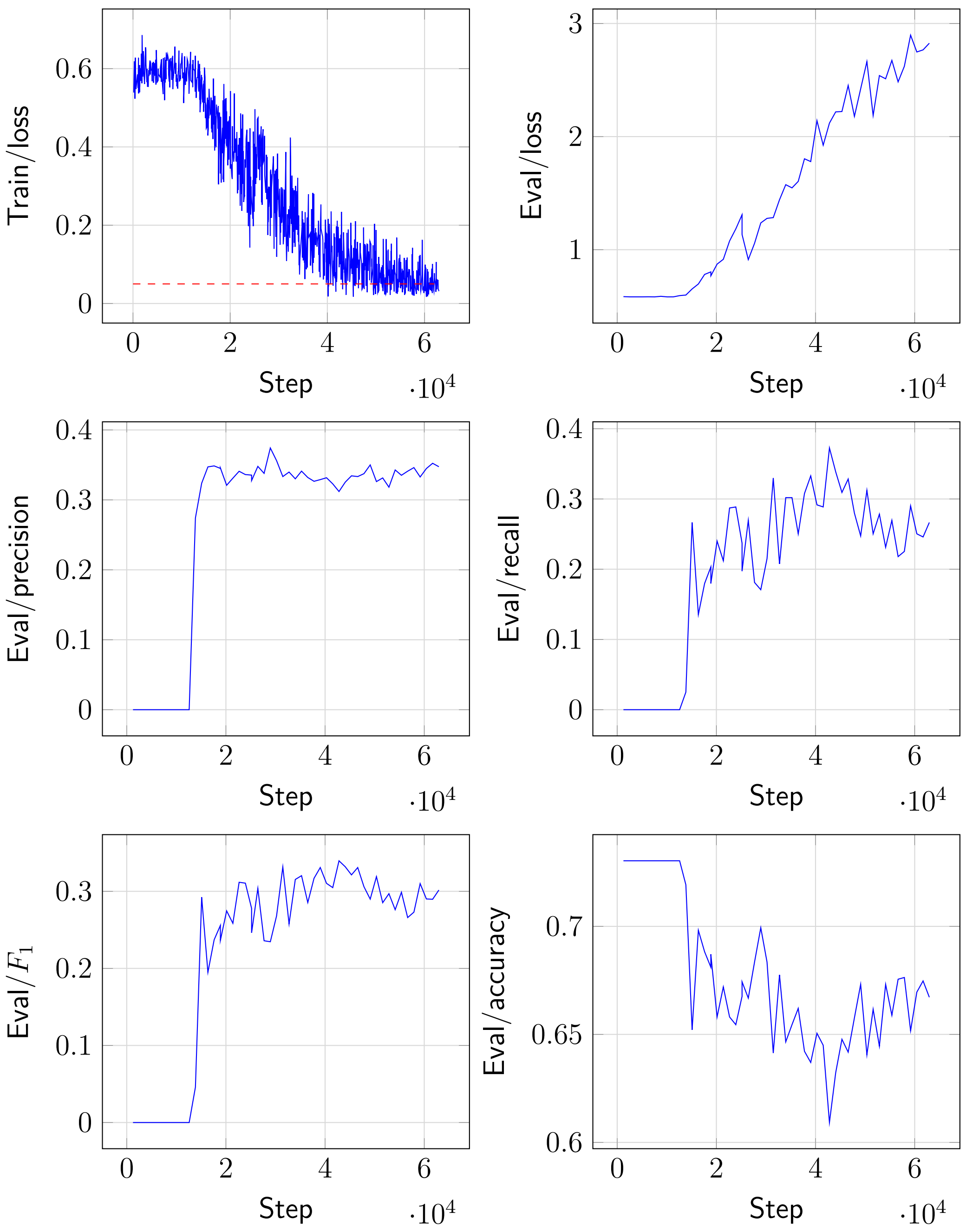


Figure C.2: Evaluation metric graphs from finetuning DistilBERT on content. The red line in loss graph is $y = 0.05$

C.2 Graphs and Results of Testing Multiple NNs

Here we present some of the other typologies we have tried for the FNN models. As we can see, almost all of them performed the same. The data was retrieved from tensorboard logs.

Firstly, in Figure C.3, we see that the training loss does not converge. The model seems to be unable to detect a pattern. This loss graph is consistent across most of our networks. Figure C.4 displays the evaluation graphs from training a network with a small learning rate of 1×10^{-6}. We observe an interesting pattern where the training and evaluation loss have a fan shape. This is a sign that the model is not training properly. Lastly, in Figure C.5, we see the evaluation graphs of a network trained using an adaptive learning rate. As with the model shown in Figure C.3, this model seems to be unable to detect a pattern.

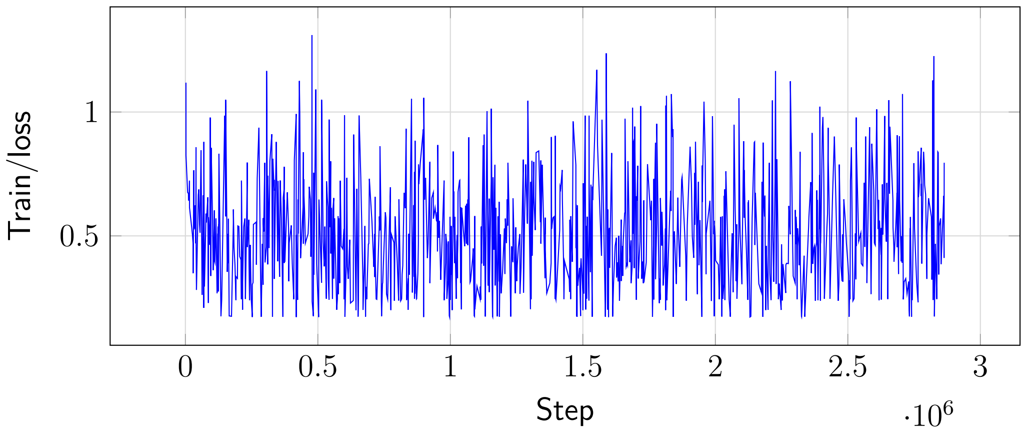


Figure C.3: Results of running a FNN for 340 epochs.

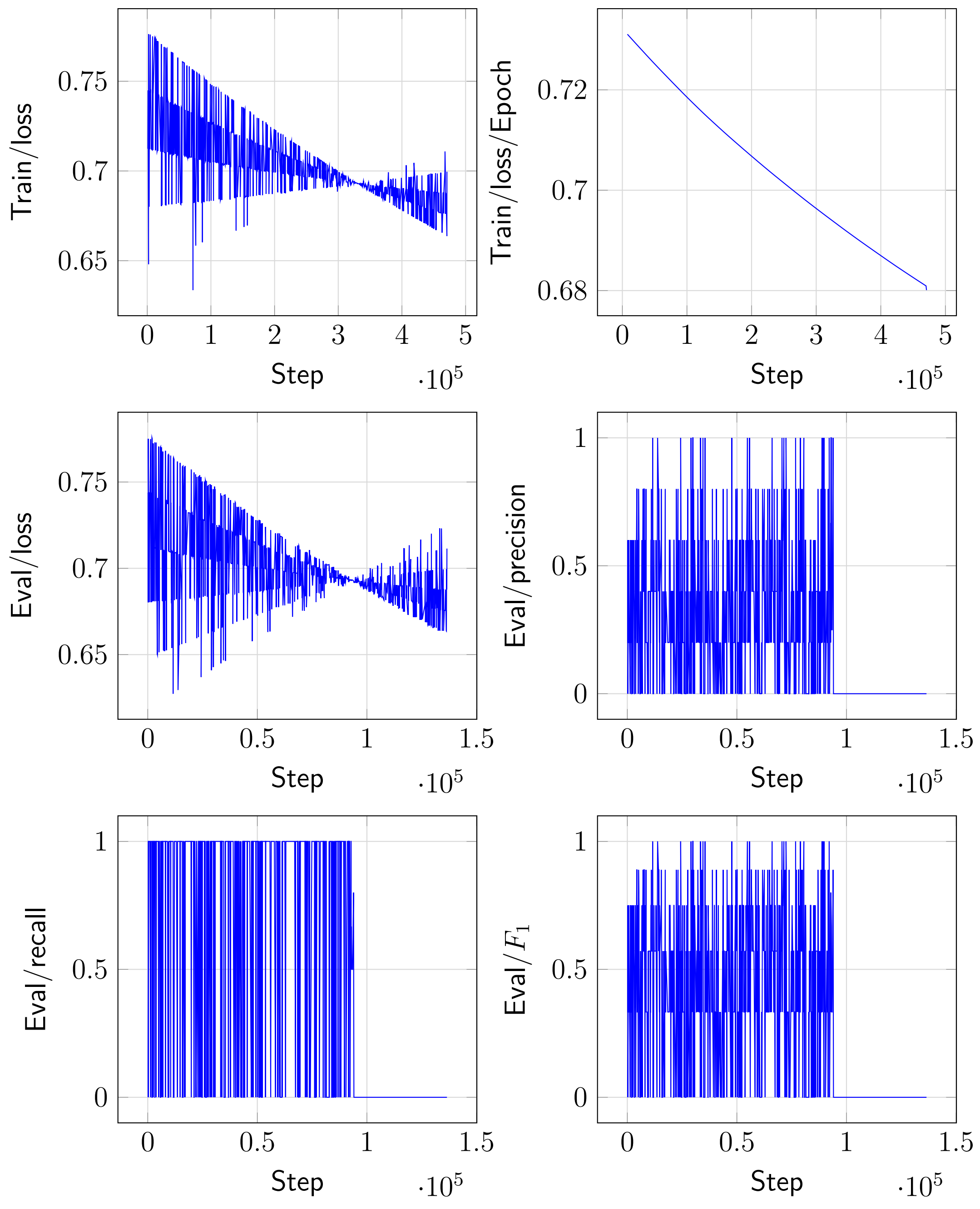


Figure C.4: Evaluation metric graphs of running a [5000-3000-1500-150-5] NN with 1×10^{-6} learning rate.

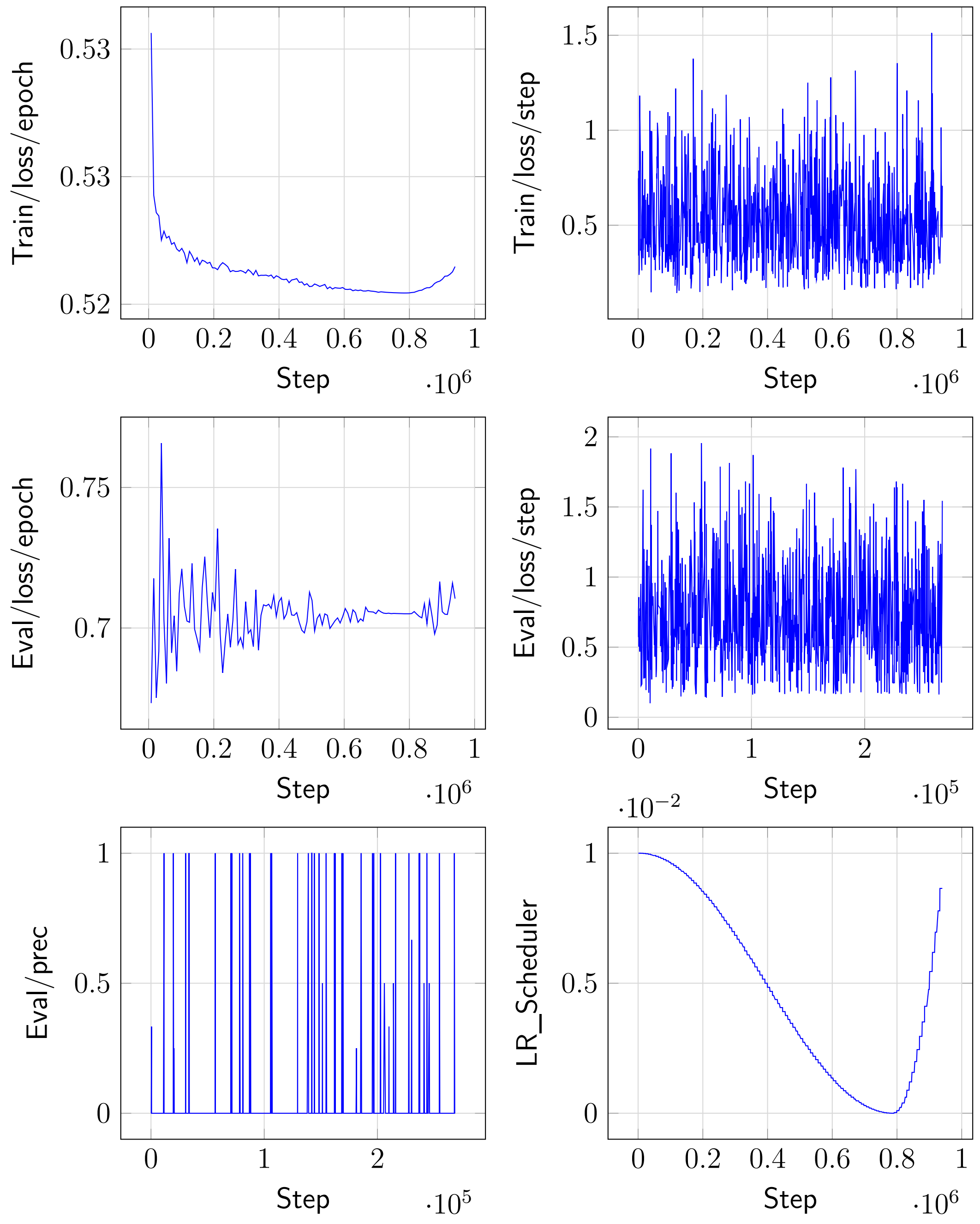


Figure C.5: Evaluation metric graphs of running a [5000-3000-1500-150-5] NN with `CosineAnnealingLR` lr_scheduler

D Hyperparameters

In this appendix we show the different hyperparameters used for training. The precision is the precision score that variation of parameters achieved in our grid search cross validataion. Default values are used when nothing is specified. We also display the general algorithm for achieving these parameters.

Table D.1: XGBoost hyperparameters.

Parameter	Value
`max_depth`	3
`n_estimators`	200
`gamma`	10
`lambda`	4
`objective`	logistic
Precision	0.6016

Table D.2: Random forests hyperparameters.

Parameter	Value
`max_depth`	3
`n_estimators`	200
`criterion`	entropy
Precision	0.6472

Table D.3: Logistic regression hyperparameters.

Parameter	Value
`penalty`	l1
`solver`	saga
`C`	0.001
Precision	0.6983

General algorithm for choosing hyperparameters

When choosing the hyperparameters we used the same general approach.

1. Set some initial values
2. Record the precision
3. Investigate values close to the ones selected
4. Investigate the recall on validation dataset to see if the recall are reasonably high. This is where we exercise discretion to make sure that the model is usable.