HOCHSCHULE DARMSTADT, UNIVERSITY OF APPLIED SCIENCES

INFORMATION SCIENCE

BACHELOR THESIS

Predicting Stock-Values using Sentiment Analysis on Aggregated Social Media Data

presented by:	Jan BURGER
	Grafenstraße 24 64283 Darmstadt
Matriculation number:	765391
Supervisors:	Prof. Dr. Margot MIESKES
	Prof. Dr. Melanie SIEGEL
Date of submission:	July 26, 2022

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbmd
FACHBEREICH MEDIA

Abstract

Due to the financial potential of the stock market, researchers have been trying to predict its behavior for a long time now. Different techniques for stock market forecasting include a quantitative approach, a macro economical approach or a sentiment analysis approach. In this thesis, I focus on the sentiment analysis technique to predict stock returns. Specifically, I use social media data from a financial forum called `r/wallstreetbets` to analyze the sentiment of these posts. For the sentiment analysis, I finetune five different pretrained Neural Network transformer models and compare their results with the models before the finetuning process. I use the sentiment data to do a correlation analysis between the sentiment scores and the stock returns of the two companies, GameStop (GME) and AMC Entertainment Holdings Inc. (AMC) and furthermore the index SPDR SP 500 ETF (SPY). I further use the sentiment data to predict the stock returns of the upcoming day, based on the sentiment data of the previous day. I also extract different sentiment and financial features, which are included in the models. The performance of the different machine learning models is evaluated and compared against each other. The final conclusion gives an overview whether sentiment data from `r/wallstreetbets` is a suitable source for stock market prediction.

The results of the sentiment analysis show a mean performance difference between none-finetuned and finetuned pretrained Neural Network transformer models of 31 percentage points. The mean performance before finetuning is 44.8% and after finetuning 71.9%. The t-test for the difference of two dependent means shows a p-value of $8.7e05$. Given an α value of 0.05, the results are statistically significant. The correlation results show a significant correlation between the sentiment features and the stock returns of the same day, with the highest correlation measured for the GME stock of 0.4. There is no statistically significant correlation for any sentiment feature with the stock return of the next day, based on sentiment data of the current day. The best model for the stock return predictions of the next day, based on the sentiment data of the previous day was the XGBoost model with an accuracy of 58.7%. This was achieved for the GME stock and the most important features for the model are the raw sentiment features, which consists of the probabilities for the classes, Positive, Negative and Neutral. These results lead to the conclusion that the sentiment information of the stock discussions on `r/wallstreetbets` does not drive the stock returns of the upcoming day, but rather reflects the current sentiment situation of the market.

Contents

Acronyms

AMC AMC Entertainment Holdings Inc.. 3, 13, 17, 18, 41, 43, 45–49, 57, 58, 60

ANN Artificial Neural Network. 3

API Application Programming Interface. 3, 11, 12, 24–26

ARIMA AutoRegressive Integrated Moving Average. 3, 10

B&H Buy & Hold. 3, 12

BERT Bidirectional Encoder Representations from Transformers. 3, 10, 12, 58

CNN Convolutional Neural Network. 3

DNN Deep Neural Network. 3

DT Decision Tree. 3, 10

ERD Entity Relationship Diagram. 3, 25

GAN Generative Adversarial Network. 3, 10

GME GameStop. 3, 41, 43–51, 53, 54, 56–58, 60

GRU Gated Recurrent Unit. 3

KNN K-Nearest-Neighbour. 3, 12

LSTM Long Short Term Memory. 3, 10, 12, 58

MLP Multi Layer Perceptron. 3, 12

NB Naive Bayes. 3, 10

NLP Natural Language Processing. 3, 9

NLTK Natural Language Toolkit. 3, 10

PCA Principal Component Analysis. 3, 47, 48, 50–52, 55–57, 60

RF Random Forest. 3, 10

RMSE Root Mean Squared Error. 3, 10, 47

RNN Recurrent Neural Network. 3, 9, 10, 21, 22, 60

SPY SPDR SP 500 ETF. 3, 41, 43–48, 57, 58, 60

SVM Support Vector Machine. 3, 10, 12

TF-IDF Frequency-inverse document frequency. 3, 8, 10, 60

UTC Coordinated Universal Time. 3, 25

1 Introduction

The complexity of stock prediction attracts researchers from various fields like finance, economics and information technology all over the world. Because of the large variety of researchers working on this topic and due to its complex nature, there are a lot of different approaches when it comes to stock price prediction. Researchers and investors from the financial sector, like Mehtab et al. [2020], have a bigger focus on the quantitative side of stock prediction and might use techniques like time series analysis to develop predictive models of the stock market. Economists like Konchitchki and Patatoukas [2013] often try to model the macroeconomic situation by evaluating the political and economic situation of the world. One thing all researchers have in common is that they use data and try to extract relevant information from it in order to gain knowledge and predict future stock prices and returns.

Another method to predict stocks is to use textual data. From text, a variety of information can be extracted and used for predictive modelling of the stock market. Common extraction techniques include Topic Modelling, Named Entity Recognition, Keyword Extraction and Sentiment Analysis. Whereas the first three methods focus more on the content level, sentiment analysis is all about the mood of the text. Commonly used source documents for sentiment analysis in the context of stock prediction are news articles. Researchers try to identify whether a stock is positively or negatively discussed in the news and build models based on this information.

Since the modern internet appeared and with it the rise of social media platforms, information transfer and discussions about stock news are not only restricted to news articles or magazines anymore. Nowadays, topics about the stock market are heavily discussed on social media sites and in forums. This also means that sentiment analysis on social media sites will cover a wider range of opinions about the stock market compared to news articles or financial magazines for example.

In this thesis, I use sentiment analysis to extract the sentiment from social media posts and use this information to predict the returns of specific stocks. In order to keep the research focus specific, I do a case study on the social media platform Reddit and specifically on the subreddit `r/wallstreetbets`.

1.1 Reddit and the subreddit r/wallstreetbets

”Reddit is an online forum that works in some ways like other social media so that users have free access to view and create posts, replies, votes, and comments in Reddit. What makes Reddit different from other social media is that the discussion in Reddit is classified into sub-boards (called ‘subreddit’) focused on specific topics” [Gui, 2019]. Based on this information, the subreddit I focus on in my research is called `r/wallstreetbets`. In this forum, users mostly discuss volatile stocks and companies which are new to the stock market. The discussion leans more towards single shares rather than indices such as the S&P 500 for example. There are two major reasons why this subreddit is an interesting use case for research.

1.1.1 GameStop and the short squeeze

The subreddit first raised awareness to a broader audience during the so-called ”short squeeze” of the GameStop stock in January 2021. As seen in Figure 1, short selling describes the process of an investor who borrows a share of a company from a broker (arrow one in Figure 1) and then sells it on the market for a certain amount of money (arrow two in Figure 1). The investor then expects the price of the share to drop in value (arrow three in Figure 1). After the price dropped, the investors buys the share back from the market (arrow four in Figure 1) and returns it to the broker (arrow five in Figure 1). The difference in share price between sold and bought will then be his profit (arrow six in Figure 1).

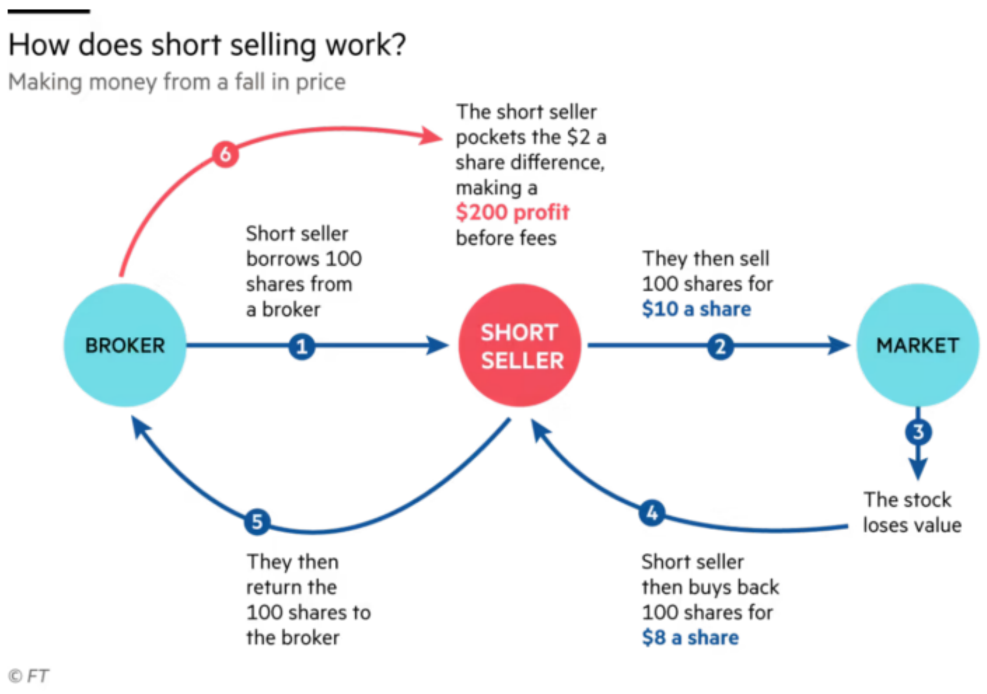

How does short selling work?
Making money from a fall in price
The short seller pockets the $2 a share difference, making a $200 profit before fees
6
Short seller borrows 100 shares from a broker
They then sell 100 shares for $10 a share
BROKER
1
SHORT SELLER
2
MARKET
3
The stock loses value
5
4
They then return the 100 shares to the broker
Short seller then buys back 100 shares for $8 a share
© FT

Figure 1: The process of short selling visualized [FT, 2022]

Lots of hedge funds and investors had short positions on the GameStop stock. They knew that GameStop had an expiring business model and wanted to exploit this situation by benefiting from the constantly falling stock price via short selling. However, in January 2021 day traders on `r/wallstreetbets` realized ”...that if they could actually push the retailer’s shares higher, that would leave its short-sellers facing big losses, forcing them to buy back the shares to ’cover’ their shorts, but then worsen the losses for other shorts and triggering a spiral higher for the stock. This so-called ’short squeeze’ proved successful, with Melvin[1] and another short-seller, Citron Research, closing out their positions at a significant loss” [FT, 2022]. Because the discussion on `r/wallstreetbets` triggered the ”short squeeze” of the GameStop stock, this example shows how powerful social media discussions about stocks can potentially be, and why Reddit is an interesting use case for this kind of research.

[1]Melvin Capital, Hedge Fund that shorted GameStop

1.1.2 The language used in r/wallstreetbets

Over time, the people on `r/wallstreetbets` developed their own kind of language. They use special expressions to make statements about certain things. For example, consider the post ”More bullish news for TSLA[2] and TWTR[3] Ready to penetrate the MOON!!” from the user ”hi-imBen” published in April 2022. The expression ”penetrate the MOON!!” is a euphoric statement, which means that the person expects the stocks of Tesla and Twitter to rise in value. Other examples of specialized language in `r/wallstreetbets` are ”bullish” and ”bearish”. A bullish stock market means that the values of the stocks go up, meanwhile a bearish stock market indicates the exact opposite, meaning that the values of the stocks go down. The highly specific language used on `r/wallstreetbets` makes it a very interesting case for sentiment analysis, because simple rule based strategies might have difficulties classifying posts with such specific language. Therefore, the approach I choose in this research is to build a model which is trained on the specific language used in `r/wallstreetbets`.

1.2 Goal of the thesis

In this thesis, I investigate the influence of the sentiment of Reddit posts from `r/wallstreetbets` on stocks discussed in the forum. Specifically, I divide my research in three different parts.

1.2.1 Sentiment analysis using Transformer Neural Networks

I use pretrained Transformer Neural Networks to analyze and predict the sentiment of Reddit posts and comments. I then compare the performance between pretrained models only and pretrained models which are fine-tuned on the specific language on r/wallstreetbets. My specific research question in this subsection is:

”Is there a performance difference between none fine-tuned and fine-tuned pretrained Neural Network transformer models on the task of sentiment classification with posts from r/wallstreetbets? And if so, how large is the performance difference and is it statistically significant?”

To evaluate the performance of the models I use accuracy as well as precision, recall and the F1-Score.

1.2.2 Correlation Analysis

The second part of my research includes a correlation analysis between the sentiment and the stock returns of a given stock discussed on `r/wallstreetbets`. I use linear correlation tests to determine for example whether an increase in the sentiment score goes along with an increase in stock returns. I also investigate whether the sentiment score is really the underlying factor, or if for example simply the total number of posts about a given stock on Reddit is the better predictor. The research questions for this subsection are:

[2]Stock symbol for Tesla
[3]Stock symbol for Twitter

"Is there a significant correlation between the sentiment of a given stock discussed on r/wallstreetbets and its returns?"

"What influence does the total number of posts about a given stock has on its returns?"

1.2.3 Predictive modeling

The last part of this thesis covers the predictive modeling of stock returns given its sentiment score. I train and evaluate different machine learning models and compare their results. The research question for this section reads:

"To which extent can the sentiment of a given stock discussed on r/wallstreetbets predict its returns?"

My hypothesis for this question is that in some specific timeframes where a stock is heavily discussed in the forum, it is possible to predict its returns purely by the sentiment or the number of posts and comments that were written on `r/wallstreetbets`. But the majority of the time, it is rather difficult to predict a stock purely by using the sentiment score of Reddit posts.

1.3 Methodical approach

In order to answer my research questions, I use a quantitative approach. The data I use consists of Reddit posts and comments from `r/wallstreetbets` as well as the stock prices and returns from Yahoo Finance. I then conduct different experiments in predictive modeling as well as statistical analysis and compare and evaluate the results.

1.4 Thesis Outline

First, I discuss the state of the art results and related work to the topic of stock value prediction using sentiment analysis. I specifically provide an overview of the latest research regarding sentiment analysis techniques themselves. After that, I differentiate between state of the art publications of stock prediction using sentiment analysis of social media data and stock prediction using sentiment analysis of news articles. In the next section, I introduce some theoretical foundations about financial engineering and Neural Networks. These foundations define and present the most important techniques I use throughout the thesis. After a short note about the data extraction, the sentiment analysis part of my research is introduced. It starts with the annotation process of the reddit posts, explains the experimental setup and after that presents the results of the experiments. At the end of the chapter, I discuss the results and conclude the chapter. The second part of my research focuses on the stock value prediction, using the results of the sentiment analysis. First, I do a correlation analysis with the sentiment values and the stock returns. After that, I use several machine learning algorithms to model the stock returns, given the sentiment scores from research part one. Both, the correlation analysis as well as the predictive modeling chapter start with the experiment setup. After that, the results are being presented and at the end discussed and concluded. At the end of all three research parts, I draw a final conclusion and explain some possible future work topics, which result from my research.

2 State of the art and related work

In this section, I give a brief overview of the most relevant literature in the field of stock prediction via sentiment analysis. This part is divided into three sections. Section 2.1 talks about generic techniques for sentiment analysis. Section 2.2 discusses the literature regarding stock prediction using sentiment analysis of news articles. Section 2.3 focuses on the literature of stock prediction using sentiment analysis of social media data.

2.1 Techniques for sentiment analysis

In recent years, it has been demonstrated that deep learning models are a promising solution to the challenges of NLP [Dang et al., 2020]. According to current NLP leaderboards, most state of the art (SOTA) sentiment analysis models are based on some form of supervised learning [Chun, 2021]. Dang et al. [2020] did a comparative study about the performance of different models and processing techniques for sentiment analysis, such as Frequency-inverse document frequency (TF-IDF) and word embedding. They collected the datasets, models, processing techniques and performance from 32 different papers and aggregated their results. Figure 2 shows the results of different datasets in combination with different models and processing techniques.

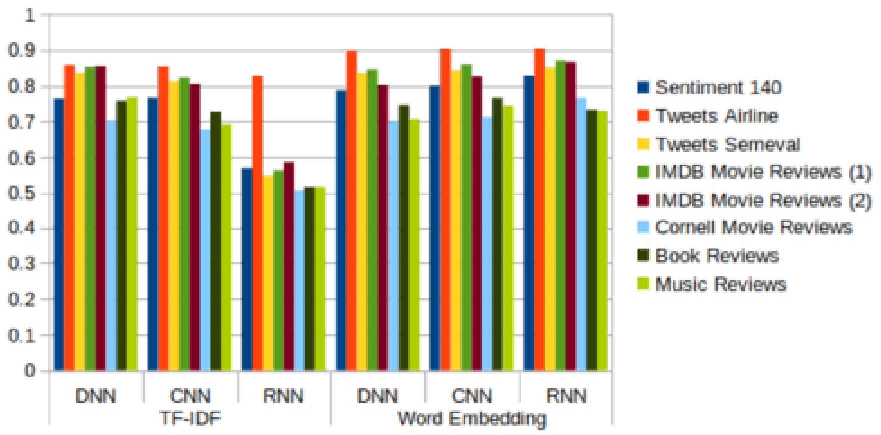


Figure 2: Accuracy values of deep-learning models with TF-IDF and word embedding for different datasets [Dang et al., 2020].

The X-Axis of Figure 2 shows the text processing technique and the type of Neural Network which was used. The accuracy values range from 0 to 1 and can be seen on the Y-Axis. Each bar in the graph describes the accuracy for one specific model with one specific text processing technique. In total eight different datasets were tested for each model. ”In general, the best behavior is shown by the combination of RNN and word embedding, although there are some exceptions. These are produced in the “book reviews” and “music review” datasets, where the values of all the metrics are slightly higher for DNN + TF-IDF than for RNN + word embedding” [Dang et al., 2020]. We can also see that in general the accuracy is higher when word embeddings are used.

Transformer models are also a widely spread technique these days. There are two main approaches to sentiment analysis of texts [Birjali et al., 2021] based on [Razova et al., 2021]: lexicon-based (or rule-based) and machine learning, in particular, using deep neural network models based on the Transformer architecture [Vaswani et al., 2017], including BERT [Devlin et al., 2019]. Transformer build on top of Recurrent Neural Networks (RNNs) and pretrained versions of them are often used for Natural Language Processing (NLP) tasks. One library which offers pretrained transformer-based models for free usage is Hugging Face [Wolf et al., 2020].

2.2 Stock prediction using sentiment analysis of news articles

The first approach for stock prediction via sentiment analysis is to use news articles and in particular news headlines as a basis for the sentiment analysis. Despite the rise of social media platforms this approach is still a common and ongoing area of research. Mostly because the algorithms and text processing techniques are constantly changing and therefore offer further possibilities. Table 1 shows literature in the area of stock prediction via sentiment analysis of news articles. For each reference, I document the text processing technique, the algorithms used, the performance achieved as well as the year in which the literature was published.

Reference	**Text processing techniques**	**Algorithms used**	**Results**
Stock trend prediction using news sentiment analysis [Kalyani Joshi, 2016]	Dictionary based approach which uses Bag-of-Word technique	SVM, RF, NB	80% accuracy
Stock price prediction using BERT and GAN [Sonkiya et al., 2021]	Word embeddings	BERT, GAN	Outperformance of algorithms like GRU, LSTM and ARIMA
Combining news sentiment and technical analysis to predict stock trend reversal [Attanasio et al., 2019]	-	SVM, NB, MLP, DT	News information is worth considering in trend prediction
Explainable stock prices prediction from financial news articles using sentiment analysis [Gite S, 2021]	-	LSTM, LSTM-CNN	88% accuracy
Prediction of stock values changes using sentiment analysis of stock news headlines [Nemes and Kiss, 2021]	Word embeddings	BERT, RNN, VADER sentiment package, textblob	Correlation analysis between compound values and sentiment score

Table 1: Literature overview - Stock prediction using sentiment analysis of news articles

The most commonly used text processing techniques are dictionary and rule-based approaches as well as word embeddings, which are used by [Kalyani Joshi, 2016], [Sonkiya et al., 2021] and [Nemes and Kiss, 2021]. Machine learning algorithms like Support Vector Machine (SVM), Decision Tree (DT), Naive Bayes (NB) or Random Forest (RF) are often used for stock predictions, for example by [Kalyani Joshi, 2016] and [Attanasio et al., 2019]. Likewise, deep learning algorithms like Long Short Term Memory (LSTMs) or RNNs are used.

Furthermore, Sonkiya et al. [2021] and Nemes and Kiss [2021] used transformer models like Bidirectional Encoder Representations from Transformers (BERT). These transformer models were used for the sentiment analysis part and not for the stock predictions itself. The performance of the models was measured in different ways. Some authors like Gite S [2021] and Kalyani Joshi [2016] report the accuracy as evaluation. Others like Sonkiya et al. [2021] compare their results against a baseline and report according to the performance against the baseline model.

In their paper, Kalyani Joshi [2016] collected three years worth of data from news articles of the Apple stock. They primarily used `https://news.google.com/`, `https://www.reuters.com/business/finance/` and `https://finance.yahoo.com/` for their collection process. For text processing, they used the bag-of-words method and tokenized each word. They also added a stop word list for filtering. The document was represented using the TF-IDF scheme. All models tested had a similar performance with an accuracy around 80%.

Sonkiya et al. [2021] used BERT for sentiment analysis and a Generative Adversarial Network (GAN) for the stock prediction. They used a specific version of BERT called FinBERT which was "trained on TRC2-financial data and on Financial PhraseBank" [Sonkiya et al., 2021]. The news articles were scraped from the investment platform Seeking Alpha (`https://seekingalpha.com/`). They collected a total amount of ten years worth of data for the Apple stock. As an evaluation metric for the stock prediction, they used the Root Mean Squared Error (RMSE).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x}_i)^2}{N}} \tag{1}$$

Unlike the other authors who used accuracy as their evaluation metric for their classification tasks, Sonkiya et al. [2021] used RMSE to evaluate their regression tasks. A regression task in the field of stock prediction means, that the authors goal is to predict the exact price or return of a stock. Stock classification only means the author wants to predict whether a stock rises or falls in value. Sonkiya et al. [2021] achieved a significant difference compared to time series algorithms like AutoRegressive Integrated Moving Average (ARIMA) or LSTMs. For example, the testing set of the ARIMA model had a RMSE of 18,25 meanwhile the proposed GAN model by the authors had a RMSE of 1,82 in the testing set.

In their article, Nemes and Kiss [2021] compared different sentiment analysis techniques to a baseline BERT model. Specifically they used RNNs, the Natural Language Toolkit (NLTK) tool VADER with its SentimentIntensityAnalyzer and the library textblob which is also a rule-based system.

As a data source, they used the finance-platform `https://finviz.com/` and scraped the news articles of several stocks such as Amazon, Facebook and Google. As an evaluation technique, the correlation between the sentiment score and the stock price was calculated. Their results show that the highest correlation between the sentiment of news articles and the stock open-price was achieved by the textblob python library with a correlation coefficient of 0.5.

2.3 Stock prediction using sentiment analysis of social media data

A modern approach of stock price prediction is to use social media data for the sentiment analysis part. Platforms like Twitter, Facebook or Reddit often provide Application Programming Interfaces (APIs) to collect large amounts of data. The general methods and algorithms used for sentiment analysis and stock prediction based on social media data are very similar to the ones used for news articles. Table 2 shows literature in the area of stock prediction via sentiment analysis of social media data. For each reference, I document the text processing technique, the algorithms used, the performance achieved as well as the year in which the paper was published.

Reference	**Text processing techniques**	**Algorithms used**	**Results**
Stock prediction based on social media data via sentiment analysis [Gui, 2019]	VADER, Sentiwordnet	KNN, DT, SVM	Compared with a baseline B&H model better performance
Predicting $GME Stock Price Movement Using Sentiment from Reddit r/wallstreetbets [Wang and Luo, 2021]	VADER for sentiment analysis combined with word2vec + BERT	SVM, RF, DT, MLP	DT + word2vec + VADER had an accuracy of 99%
Stock Prediction Using Twitter Sentiment Analysis [Mittal and Goel]	Dictionary based method based on Profile of Mood States (POMS) questionnaire	Linear Regression, Logistic Regression, SVM, NN	75% accuracy
WallStreetBets on Wall Street An Empirical Analysis of the Market Power of WallStreetBets [Jacobsen and Pedersen, 2021]	Dictionary based method	Individual Regression, Aggregated Regression	Forum activity was a significant driving force for volume of the affected stocks
HISA-SMFM: Historical and sentiment analysis based stock market forecasting model [Gupta et al., 2022]	Textblob library	LSTM	95% accuracy

Table 2: Literature overview - Stock prediction using sentiment analysis of social media data

The most commonly used text processing techniques are word embeddings and rule-based systems such as the VADER sentiment package or the Textblob python library. Machine Learning algorithms like Decision Tree, K-Nearest-Neighbour (KNN) and SVM as well as Deep learning techniques such as LSTMs or BERT were used for the stock prediction and sentiment analysis. The results were measured in different ways. Some authors like Mittal and Goel and Gupta et al. [2022] provided accuracy metrics for their evaluation, while others compared their models against a baseline model or against a Buy & Hold (B&H) strategy like Gui [2019] did.

Wang and Luo [2021] tried to classify the GameStop price movement by using sentiment analysis of `r/wallstreetbets` posts. They picked a timeframe of half a year and used an already existing dataset from Kaggle. The dataset consists of approximately 36.000 posts which were written over the course of six months. Wang and Luo [2021] then used the VADER sentiment package for sentiment analysis. They combined this approach with word embeddings from word2vec and pretrained BERT wordvectors. Furthermore, they extracted features like wordcount, stopwordcount and emojicount. For the stock prediction part, they used algorithms like Decision Tree, Random Forrest as well as a basic Multi Layer Perceptron (MLP). The best performing model was the decision tree in combination with the additional features mentioned above. The authors state that it achieved an accuracy of 99%.

Jacobsen and Pedersen [2021] also use `r/wallstreetbets` as a basis for their sentiment analysis. Their choosen timeframe for the collection of posts and comments was January 1, 2020, to March 15, 2021. They focused on a number of stocks and in particular tech-stocks like Apple, Google and Facebook. For the sentiment analysis, they purely relied on the VADER sentiment package in R. For each day, they calculated a positive sentiment percentage rate and used this as a basis for the stock prediction. For stock prediction they used individual regression to see the impact of the sentiment score on the stock returns. They also added additional features and used multiple regression in order to have the maximum potential. The results are very different for each stock. The sentiment score for Nokia for example was statistically significant, while the sentiment score for Game Stop and Palantir did not show a statistically significant influence on the stock returns.

In their paper, Gupta et al. [2022] combined sentiment analysis with time series methods to predict the future of stocks. They utilized the Twitter API to collect the necessary data. The sentiment analysis was done using the rule-based Textblob python library. They then merged the sentiment score and historical stock data and used a LSTM to predict future stock prices and returns. Their model achieved an accuracy of 95%.

3 Theoretical Foundations

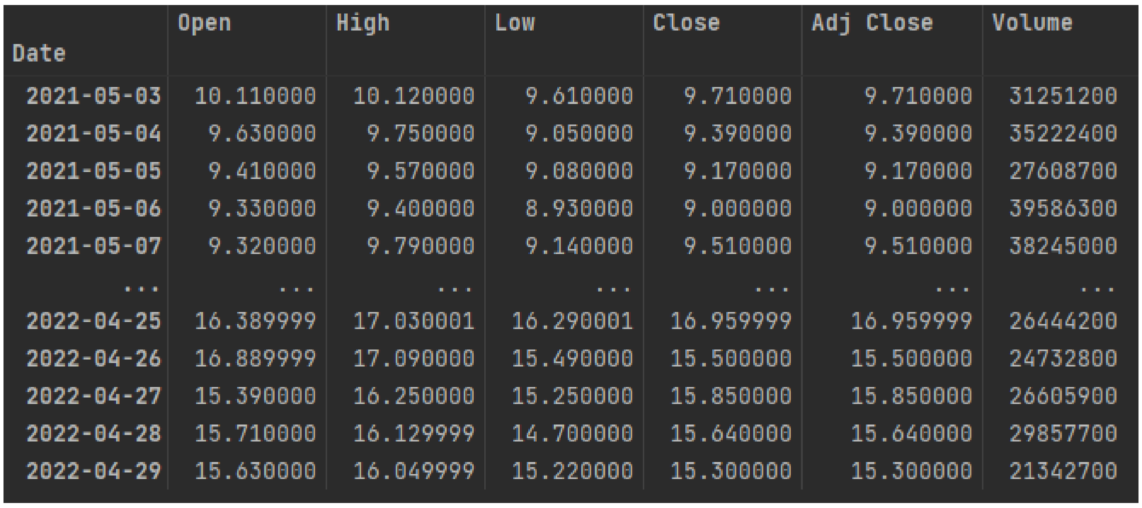
This section introduces some theoretical concepts in financial engineering and the usage of Neural Networks. These concepts are fundamental and important to understand in order to follow the decisions I do and explanations I give throughout the whole thesis.

3.1 Financial Engineering

Financial engineering is the use of mathematical techniques to solve financial problems. Financial engineering uses tools and knowledge from the fields of computer science, statistics, economics, and applied mathematics to address current financial issues as well as to devise new and innovative financial products [inv].

3.1.1 Financial data

Financial data is usually expressed as a time series with a specific starting time and ending time. The units in which the data is expressed can differ. Common units are for example, days, hours or minutes. For every period in the time series, there are a number of datapoints which characterize this specific period in the time series. Figure 3 shows an excerpt of financial time series data. It presents the stock price data for the AMC Entertainment Holdings Inc. (AMC) stock in the period from 03 May 2021 to 29 April 2022. The periods are expressed in days, meaning that each row in the table represents one specific day.



```
            Open       High       Low        Close      Adj Close  Volume
Date
2021-05-03  10.110000  10.120000   9.610000   9.710000   9.710000  31251200
2021-05-04   9.630000   9.750000   9.050000   9.390000   9.390000  35222400
2021-05-05   9.410000   9.570000   9.080000   9.170000   9.170000  27608700
2021-05-06   9.330000   9.400000   8.930000   9.000000   9.000000  39586300
2021-05-07   9.320000   9.790000   9.140000   9.510000   9.510000  38245000
       ...        ...        ...        ...        ...        ...       ...
2022-04-25  16.389999  17.030001  16.290001  16.959999  16.959999  26444200
2022-04-26  16.889999  17.090000  15.490000  15.500000  15.500000  24732800
2022-04-27  15.390000  16.250000  15.250000  15.850000  15.850000  26605900
2022-04-28  15.710000  16.129999  14.700000  15.640000  15.640000  29857700
2022-04-29  15.630000  16.049999  15.220000  15.300000  15.300000  21342700
```

Figure 3: Financial time series data - AMC

The index column with the name of ”Date”, represents the chosen period, which in this case is on day for each row. The open price, which is shown in the second column, is the stock price at the beginning of the period. The close price, shown in column five, is the stock price at the end of the period. The high price is the maximum price achieved during one day and the low price is the minimum price the stock has fallen down to during one day. The volume, shown in the last column, is the total number of shares that were traded during that time period. Traded means, that the shares were sold by someone and bought by someone else.

The difference between column five, the close price, and column six, the adjusted close price, is that the adjusted close price accounts for stock splits and dividend payments. A stock split can happen, if for example the price for one share is very high, and it becomes difficult for people to buy a whole share. Then, the company splits the stock and for example, reduces its price from 1000$ to 500$. Everyone who is an owner of shares for this company then receives two shares for each share they already possess. This way, the stock becomes more affordable for the majority of the people. In order to adjust the historical prices, which were documented before the stock split, all historical prices need to be divided by two in this example. This is the reason why there is an additional column with the adjusted close price. This adjusted close price is usually the primary datasource used for financial engineering. I also use this datasource as a basis for the stock return predictions.

3.1.2 Missing data

Stock exchanges, where stocks are traded usually, are only open during working days. This means that there are no stock trades possible during the weekend and on holidays. Therefore, on these days there is no stock data available. Other common reasons for missing data might be technical errors in the datasource or simply the fact that there were no trades for a specific stock on a day. This might happen with very small and insignificant companies.

One intuitive, but incorrect way to deal with missing data would be to, visually speaking, draw a straight line from the stock price on Friday to the stock price on Monday, in order to fill the data for the weekends. This technique is called linear interpolation. However, for financial time series data, this is an incorrect approach, because prices of the future, which are not known on Fridays, are used to interpolate prices in the past. This is called backward filling and is the incorrect way of filling missing financial time series data. The correct way is to use forward filling, which is simply using the last known price and filling all missing values in the future of that last known price with it. In the example of the missing data for weekends, this means that the stock price of Friday will be projected onto Saturday and Sunday.

Since weekends will continuously have missing data, it is also a common practice to delete all rows in the dataset, which have missing values due to being part of the weekend. This also makes sense, because the difference between Friday and Monday is one day in ”trading” time, but not in ”real” time. This method is also the preferred choice I use later on in my stock return predictions.

3.1.3 Stock returns

Stock returns are always defined over some time period. In the example shown in Figure 3, the time period would measure one day. Therefore, given some time period, assuming no dividends, the net return is calculated as follows:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \tag{2}$$

where P_t is the final price of the given time period and P_{t-1} is the initial price. For an initial price of 50$ and a final price of 100$, the net return would be 1.

Usually, further mathematical operations would use this exact value, but it can also expressed explicitly in a percentage value, which in this case would be a net return of 100%.

Another option is to express the net return R_t as the gross return, which is a multiplication factor of the original investment. It can be expressed by adding +1 to the equation of the net return R_t:

$$1 + R_t = \frac{P_t}{P_{t-1}} \tag{3}$$

If for example, the investment in the stock was 100$ and the stock was sold for 120$, the gross return of this investment is 1.2, using the formula above.

From the gross return or the net return, the log return can be calculated. In section 3.1.4, I explain why the log return is important in financial time series forecasting. The log return, also known as the continuously compounded return, is calculated by using the natural logarithm of the gross return:

$$r_t = \log(1 + R_t) = \log \frac{P_t}{P_{t-1}} = \log P_t - \log P_{t-1} \tag{4}$$

The log return is usually indicated as r_t, as shown in the formula above. The formula shows different ways to calculate the log return from the gross return, but the easiest way to understand the log return, is to simply calculate the logarithm of the gross return $1+R_t$.

3.1.4 Stock prices vs. stock returns

In financial time series forecasting, stock returns, rather than stock prices, are predicted. This has several reasons. The main reason has to do with a phenomenon called stationary. A time series can be described as stationary if the mean μ and the standard deviation σ do not change over time. Figure 4 shows an example of a stationary and none-stationary time series.

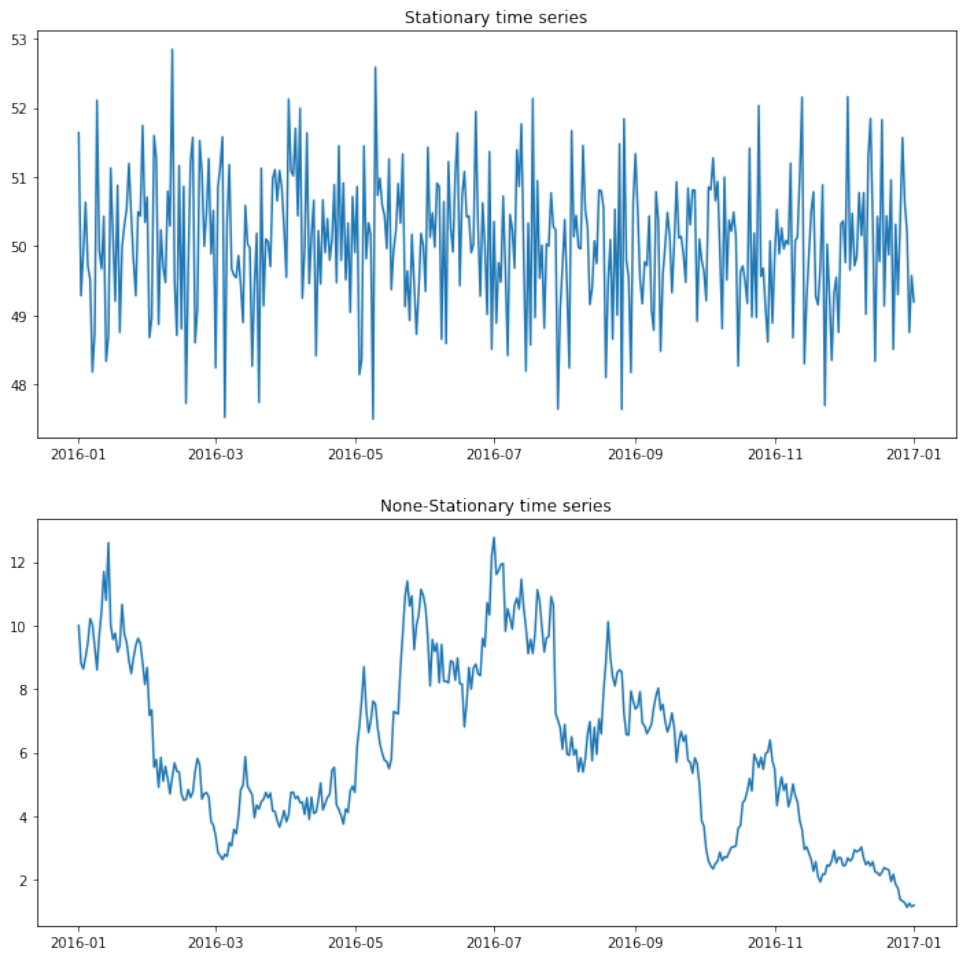


Figure 4: Example of stationarity

It can clearly be seen that the stationary time series at the top has a constant mean and variance. The none-stationary time series at the bottom of Figure 4 has a constantly changing mean and variance over time.

Stock prices are usually none-stationary, because for example when a company is new, their stock price will be very low, usually starting around one dollar. After a few years, the stock price, depending on the performance of the company, can reach up to a few thousand dollars, as for example, it was the case with many tech companies. Since machine learning models are very bad at extrapolating and predicting values which are outside the range they have already seen, stock price data is not the primary source for time series forecasting.

A better option for time series forecasting is to predict the returns of a stock for a given period. Stock returns, and especially the log return r_t of a stock is often times normally distributed and the mean and variance do not change over time. Applying the logarithm to large return values, which are not very common, transforms them into smaller return values, which are closer to the mean of the return distribution. Applying the logarithm to already small return values, the value does not change much. Mathematically, this means that for small values of x, $x \approx \log{(x+1)}$.

Figure 5 below shows the log return r_t of the AMC stock at the top and the stock price at the bottom.

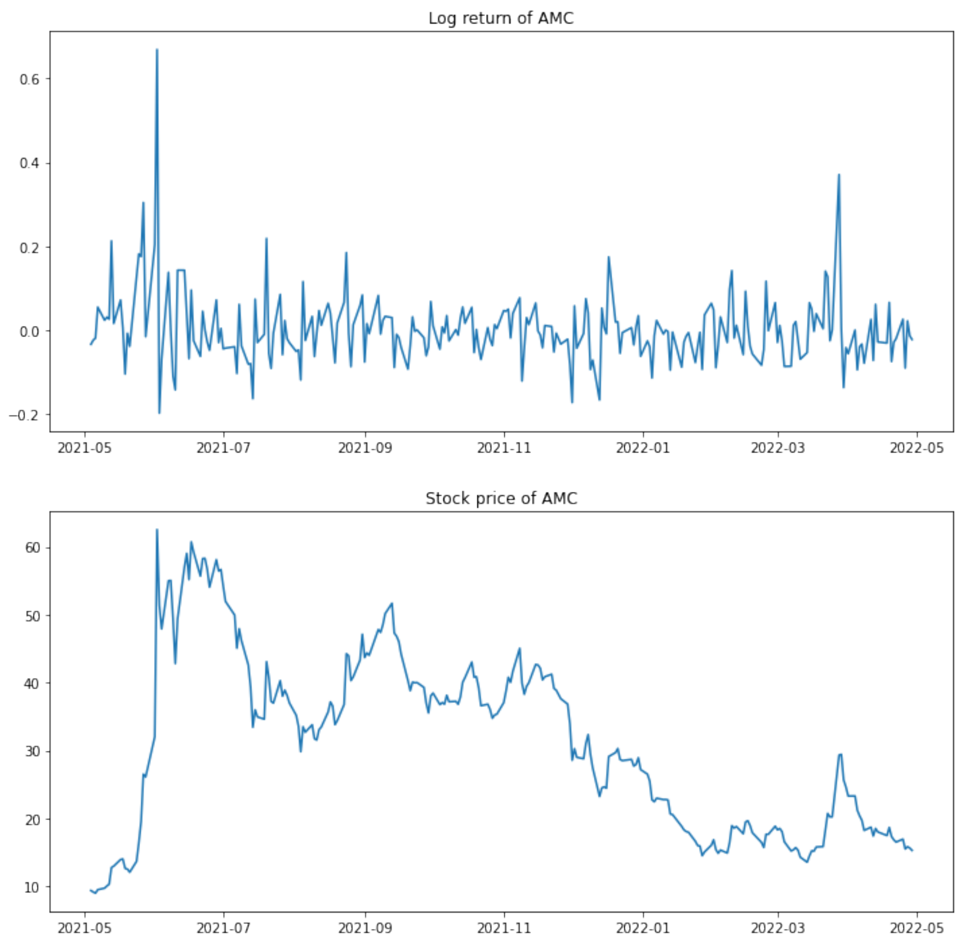


Figure 5: Log return and stock price comparison of AMC

Visually, it can be seen that the log returns of AMC are very close to being stationary, whereas the stock price of AMC is clearly not stationary. Moreover, the log returns of AMC are roughly normally distributed, which is shown in Figure 6.

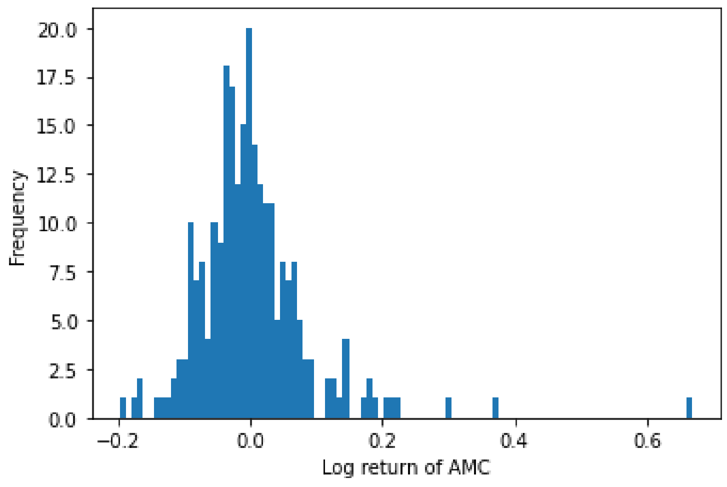


Figure 6: Log return distribution of AMC

The x-axis of Figure 6 shows the log return of AMC and the y-axis shows the frequency. To further confirm, that the log returns of AMC follow a normal distribution, a QQ-Plot could be used, but this would be outside the scope of this introduction.

3.1.5 Efficient market hypothesis

A capital market is said to be efficient if it fully and correctly reflects all relevant information in determining security prices. Formally, the market is said to be efficient with respect to some information set, ϕ, if security prices would be unaffected by revealing that information to all participants. Moreover, efficiency with respect to an information set, ϕ, implies that it is impossible to make economic profits by trading on the basis of ϕ [Malkiel, 1989].

If the efficient market hypothesis is true, this would lead to the fact, that with sentiment information from any kind of social media site, it is close to impossible to gain an advantage over the market and predict stock returns accurately. The hypothesis itself consists of three different forms, which are outside the scope of this introduction, but are described further in the book of Malkiel [1989].

3.1.6 Random walk hypothesis

The random walk hypothesis builds upon the efficient market hypothesis. The basic idea behind the random walk hypothesis is that in a free competitive market the price currently quoted for a particular good or service should reflect all of the information available to participants in the market that influence its present price. To the extent that future conditions of the demand or supply are currently known, their effect on the current price should be properly taken into account [Smidt, 1968].

The random walk hypothesis leads to one specific method, which is often used in financial time series prediction to have a comparison of the achieved results. This method is called the naive forecast. Based on the random walk hypothesis, the best prediction for the stock price at timepoint$_{t+1}$ is to use the stock price at timepoint$_t$. In other words, a return of zero and a constant stock price over time would be predicted using the naive forecast. This is the best model, if the random walk hypothesis together with the efficient market hypothesis are both true. If a time series model could predict future returns better than the naive forecast for a long period of time and with a high consistency, this is a big indicator for an efficient model. However, the naive forecast can only be used as a comparison when doing stock return regression, where the exact return is predicted. In stock return classification, this comparison is not possible, because usually the only two classes that are being predicted are postive and negative stock returns.

3.1.7 Classification and regression in time series prediction

There are two different supervised techniques which are used for financial time series prediction. In regression, the goal is to predict the exact stock returns. In classification, usually, the only distinction that's being made is whether a stock return is positive or negative. Since classification is oftentimes the first step and is more intuitive to understand in its evaluation, this is the preferred method I use for my research about stock return prediction.

3.2 Neural Networks

An artificial neural network (or simply neural network) consists of an input layer of neurons (or nodes, units), one or two (or even three) hidden layers of neurons, and a final layer of output neurons [Wang, 2003]. The simplest form of a Neural Network is called the perceptron.

3.2.1 The perceptron

A perceptron, which is the basic unit of every Neural Network, implements the following calculation:

$$f(x) = \sum_{i=1} w_i \times x_i + b \tag{5}$$

where $f(x)$ is the result of the calculation and the value the perceptron tries to predict, w is the weight, x is the given value, which is the only true input into the function and b is the bias. Since this is a linear equation, in order to solve none-linear problems, the result of this equation needs to be put into a none-linear function, also called the activation function.

3.2.2 Activation functions

There are several activation functions, like the ReLU, Sigmoid, Tanh and many more. The most commonly used activation function today is the ReLU. It is defined as:

$$f(x) = \max(0, x) \tag{6}$$

where x is the output of the perceptron and the input of the activation function and $f(x)$ is the result of the activation function. Let σ be any type of activation function of a Neural Network, then the full forward pass of a Neural Network to predict a value can be expressed as:

$$f(x) = \sigma\left(\sum_{i=1} w_i \times x_i + b\right) \tag{7}$$

3.2.3 Loss functions

After the input values went through the perceptron calculation and the activation function, the result is the prediction of the Neural Network. This result needs to evaluated in order to know how accurate the Neural Network did predict the values. This is the point where loss functions are used. In Regression, usually the Mean Squared Error function is used as the loss function. It calculates the squared differences between the predicted value and the actual value. It is defined as:

$$L_{MSE} = \frac{1}{N} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{8}$$

where y_i is the true value, $\hat{y_i}$ is the predicted value by the Neural Network and N is the sample size of all predicted and true values.

For Classification, the most basic and commonly used loss function is the Cross-Entropy loss. It is defined as follows:

$$L_{CE} = -\left[\frac{1}{N}\sum_{i=1}^{n} y_i \cdot \log \hat{y_i} + (1 - y_i) \cdot \log\left(1 - \hat{y_i}\right)\right] \tag{9}$$

where y_i is the true value, $\hat{y_i}$ is the predicted value by the Neural Network and N is the sample size of all predicted and true values. L_{CE} stands for Cross Entropy Loss.

When combining the concepts of the perceptron, the activation function and the loss function, the final equation to calculate a full forward pass and the loss of that particular forward pass could look like this for a Regression task:

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{n}\left(y_i - \sigma\left(\sum_{i=1} w_i \times x_i + b\right)\right)^2 \tag{10}$$

where N is the sample size, y_i is the true value, σ is some activation function, w_i is the weight for variable x_i and b is the bias.

3.2.4 The structure of a deep Neural Network

A deep Neural Network consists of multiple perceptron units. It also has multiple inputs and multiple layers. Figure 7 shows an example architecture of a deep Neural Network.

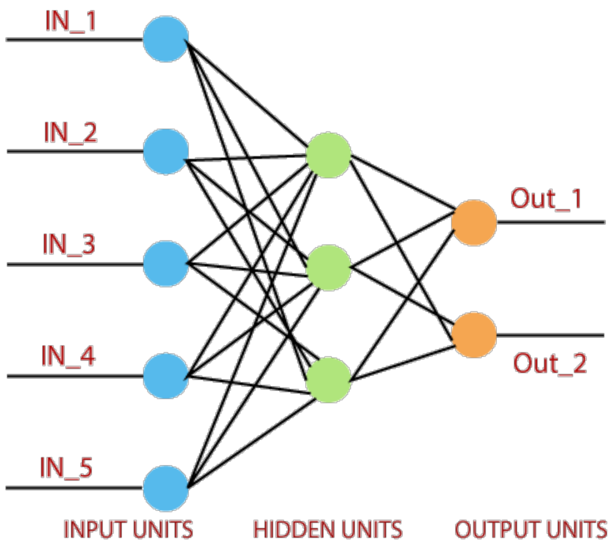


Figure 7: Example architecture of a deep Neural Network [neu]

In the example above, The Neural Network has five input units, which means that it takes into account five values for its prediction. It has one hidden layer, which consists of three Neurons. The last layer, which is marked in orange, has two output Neurons. Therefore, this architecture could be used for a binary classification problem.

3.2.5 Backpropagation

To update the weights in a Neural Network, which in the end determine the predictions, they need to be adjusted according to the loss function. When the weights are being adjusted the output of the loss function should decrease with each epoch. To achieve this, an optimization technique called gradient descent is used. The most basic form of gradient descent is defined as:

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t} \tag{11}$$

where w_{t+1} is the updated weight after the gradient descent calculation, w_t is the current weight, α is the learning rate and $\frac{\partial L}{\partial w_t}$ is the partial derivative of the loss function with respect to the weight w_t. This process is repetitively performed for all weights in the Neural Network until convergence is achieved.

3.2.6 Recurrent Neural Networks

A recurrent neural network (RNN) is a special type of an artificial neural network adapted to work for time series data or data that involves sequences. RNNs have the concept of 'memory' that helps them store the states or information of previous inputs to generate the next output of the sequence [An]. These states are often called hidden states and refer to different points in the sequence. Figure 8 shows the architecture of an unrolled RNN.

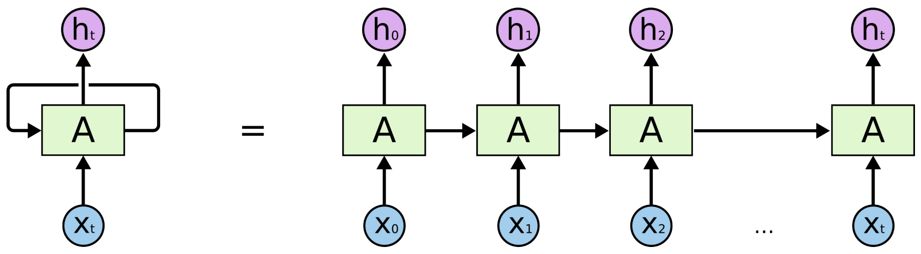


Figure 8: Recurrent Neural Network architecture [Rec]

The blue circles at the bottom of Figure 8 are the inputs of the sequence. The green squares in the middle represent the hidden layer of the RNN and the purple circles at the top are the hidden states, which are the outputs of the RNN. At each step in the sequence, the previous calculated hidden states are taken into account. Mathematically, the forward pass of a Recurrent Neural Network for some hidden state h_t can be expresses as:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t) \tag{12}$$

where h_t is the new calculated hidden state, σ represents any suitable activation function, h_{t-1} is the previous hidden state and x_t is the current input. W_{xh} is the weight at the current input Neuron and W_{hh} is the weight at the recurrent Neuron. This calculation is done for every input in the sequence. In addition, at the end of the Recurrent layer, there is often a generic feed forward layer implemented in the Neural Network.

Another visual and intuitive understandable example of a RNN is shown in Figure 9. It presents the sentence ”What time is it?” as an input sequence.

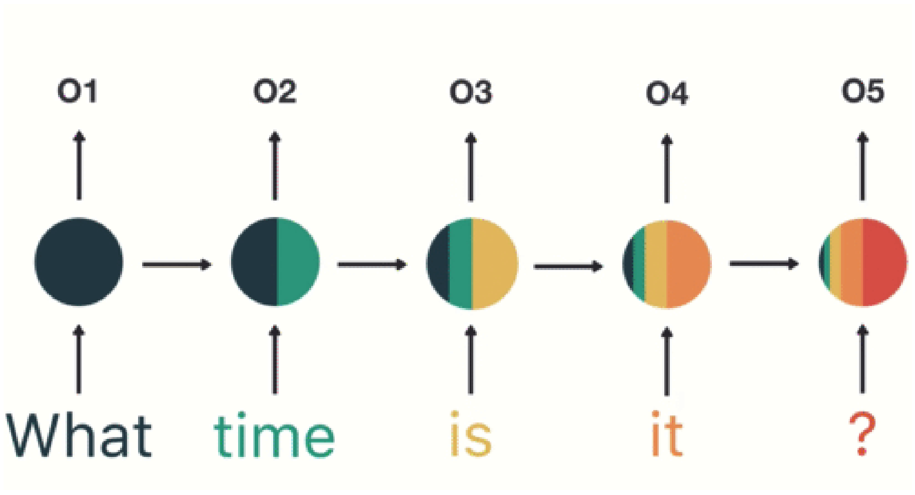


Figure 9: Example application of a RNN [Ill]

The words at the bottom of Figure 9 represent the input sequence. The circles in the middle represent the recurrent calculation of the Neural Network and the numbers at the top represent the output of the hidden state. The colors in the circles represent the information, which is passed through each step of the sequence. For example, in the last Neuron of the sequence, half of the information comes from the newest input, which is the question mark in this case, the other half of the information is provided by the calculation of previous hidden states.

3.2.7 Transformers

In their paper ”Attention is all you need”, Vaswani et al. [2017] published a new Neural Network architecture called ”transformer”. This architecture functions as a basis in advanced language models until today. Transformers were introduced to replace Recurrent Neural Networks, because they have several disadvantages like for example the slow training process and the long term memory loss. Figure 10 shows the transformer model architecture. The transformer consists of an encoder part (represented on the left side of Figure 10) and a decoder part (represented on the right side of Figure 10). First, the input embeddings of the words are being passed into the decoder. These are raw word vectors from algorithms like word2vec or GloVe, which do not have context awareness. The first step of the encoder is to add positional embeddings to the raw word embeddings. The word vectors are being passed simultaneously into the encoder. The positional embeddings are additional vectors, which represent the order of the words. Therefore, the raw word embeddings together with the positional embeddings represent the final vectors, which are being passed onto the next step in the encoder. The next step in the encoder is an attention layer. In these layers (shown in the orange boxes of Figure 10), each word in a sequence is related to all other words in it. With this technique, the model gets even more context awareness and learns, which words in the sequence are in relation to each other. After the attention unit, the encoder follows with a batch normalization (represented in the yellow boxes of Figure 10), followed by a Feed Forward layer, which then again applies a batch normalization.

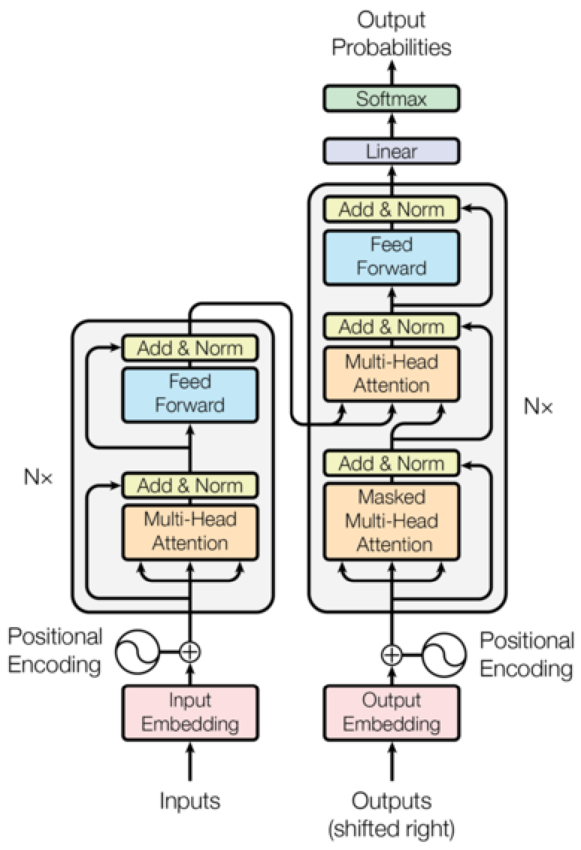


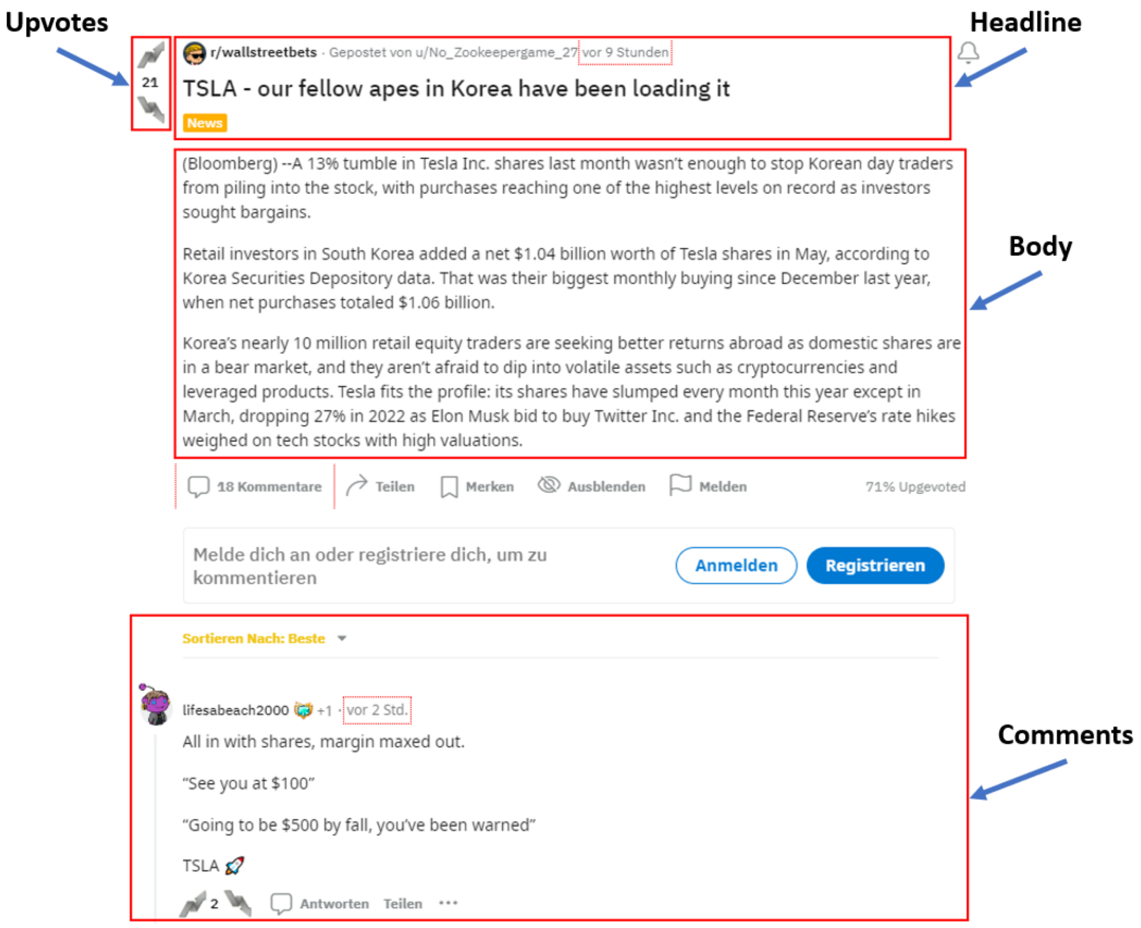
Figure 10: The transformer - model architecture [Vaswani et al., 2017]

After all the steps mentioned above, the encoder outputs the final vector representation of the input sequence, which is now being passed into the decoder. In the decoder, at first the targets, which the model should predict, are being passed into. Just as with the encoder, the inputs of the decoder are word embeddings which are complemented with the positional embeddings. After the first attention unit and a batch normalization, the output of the encoder is combined with the targets of the decoder in another attention unit. After this process, the output is passed through a Feed Forward layer as well as a linear layer. At the end of the transformer, a softmax function calculates the final output probabilities.

The described process of a transformer Neural Network above is strongly simplified. For more details, I suggest reading the original paper ”Attention is all you need” by Vaswani et al. [2017].

4 Data extraction and data flow

Data on Reddit can be divided into posts, which are also called submissions, and comments. A post consists of a headline and body. The headline is limited in its length and can only include text. Besides text, the post body can include images or videos. For each post, an arbitrary number of comments can be created by users below the post. On the left side of each post, the upvotes for this particular post are displayed. Each user has the opportunity to upvote or downvote a post. As a result, the total number of upvotes are displayed. This result can also be negative if more people downvoted than upvoted the post. In most cases, the result will be positive. Figure 11 below shows the structure of a Reddit post and its comments.



r/wallstreetbets · Gepostet von u/No_Zookeepergame_27 vor 9 Stunden

TSLA - our fellow apes in Korea have been loading it

News

(Bloomberg) --A 13% tumble in Tesla Inc. shares last month wasn't enough to stop Korean day traders from piling into the stock, with purchases reaching one of the highest levels on record as investors sought bargains.

Retail investors in South Korea added a net $1.04 billion worth of Tesla shares in May, according to Korea Securities Depository data. That was their biggest monthly buying since December last year, when net purchases totaled $1.06 billion.

Korea's nearly 10 million retail equity traders are seeking better returns abroad as domestic shares are in a bear market, and they aren't afraid to dip into volatile assets such as cryptocurrencies and leveraged products. Tesla fits the profile: its shares have slumped every month this year except in March, dropping 27% in 2022 as Elon Musk bid to buy Twitter Inc. and the Federal Reserve's rate hikes weighed on tech stocks with high valuations.

18 Kommentare 71% Upgevoted

Melde dich an oder registriere dich, um zu kommentieren

lifesabeach2000 +1 · vor 2 Std.

All in with shares, margin maxed out.

"See you at $100"

"Going to be $500 by fall, you've been warned"

TSLA 🚀

2

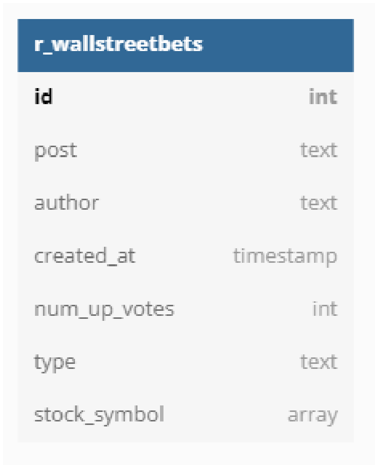
Figure 11: Structure of a Reddit post

For the purpose of my research, I extract all posts and comments of the subreddit `r/wallstreetbets`. Reddit offers a native API to extract posts and comments. However, working with the native Reddit API can be cumbersome and timeconsuming. Therefore, I chose to work with the Pushshift API [pus]. The Pushshift API is an open source project which automatically extracts all posts and comments of all subreddits and stores them in a database. They also provide a separate Python package to easily make requests to their database.

The subreddit is active since January 2012 and has approximately 12.2 million users as of June 2022. Therefore, collecting data from over 10 years would simply take too much time. Therefore, I pick a timeframe of exactly one year. In order to have the newest stocks discussed on `r/wallstreetbets`, I select the period from 01 May 2021 to 30 April 2022. From the API, I extract the author, the time the post or comment was created, the title, the body and the upvotes. As a first preprocessing step, I combine the title and the body of each post. Second, I set the timestamp from the API to Coordinated Universal Time (UTC).

In r/wallstreetbets not all posts and comments are necessarily about a specific stock. Therefore, I filter all posts and comments for potential mentions of a stock. This way I can assign each post or comment to a specific stock it is talking about. In the subreddit in most cases people use the ticker symbol of a stock to mention it. In order to get the most stock mentions possible, I use a list of the top 1000 mentioned stocks on `r/wallstreetbets` in the last 24 hours provided by the Apewisdom project [ape]. For everyday worth of data I receive from the API, I check whether each stock symbol from the list is mentioned in the post. If it is mentioned, I assign the corresponding stock symbol to it. For the screening of the stock symbols in the posts and comments, I use the normal stock symbol, which for Apple would be AAPL for example, as well as the commonly used version with a dollar sign in front of the stock symbol, like $AAPL. In the final step, I add a column that distinguishes, whether an entry is a post or a comment.

After the processing steps, I store the data in a relational Postgres Database. Figure 12 shows the final Entity Relationship Diagram (ERD) with the name of the table on top and the names of all columns below.



r_wallstreetbets	
id	**int**
post	text
author	text
created_at	timestamp
num_up_votes	int
type	text
stock_symbol	array

Figure 12: Entity Relationship Diagram of all collected posts and comments

From 01 May 2021 to 30 April 2022, I collect a total amount of 365.741 Reddit posts and 14.217.444 Reddit comments. Out of these entries, 1.616.928 have an actual stock symbol assigned and build the basis for the sentiment analysis as well as the stock prediction process. Due to rate limitations of the Pushshift API, the whole data extraction process took around 48 hours to complete.

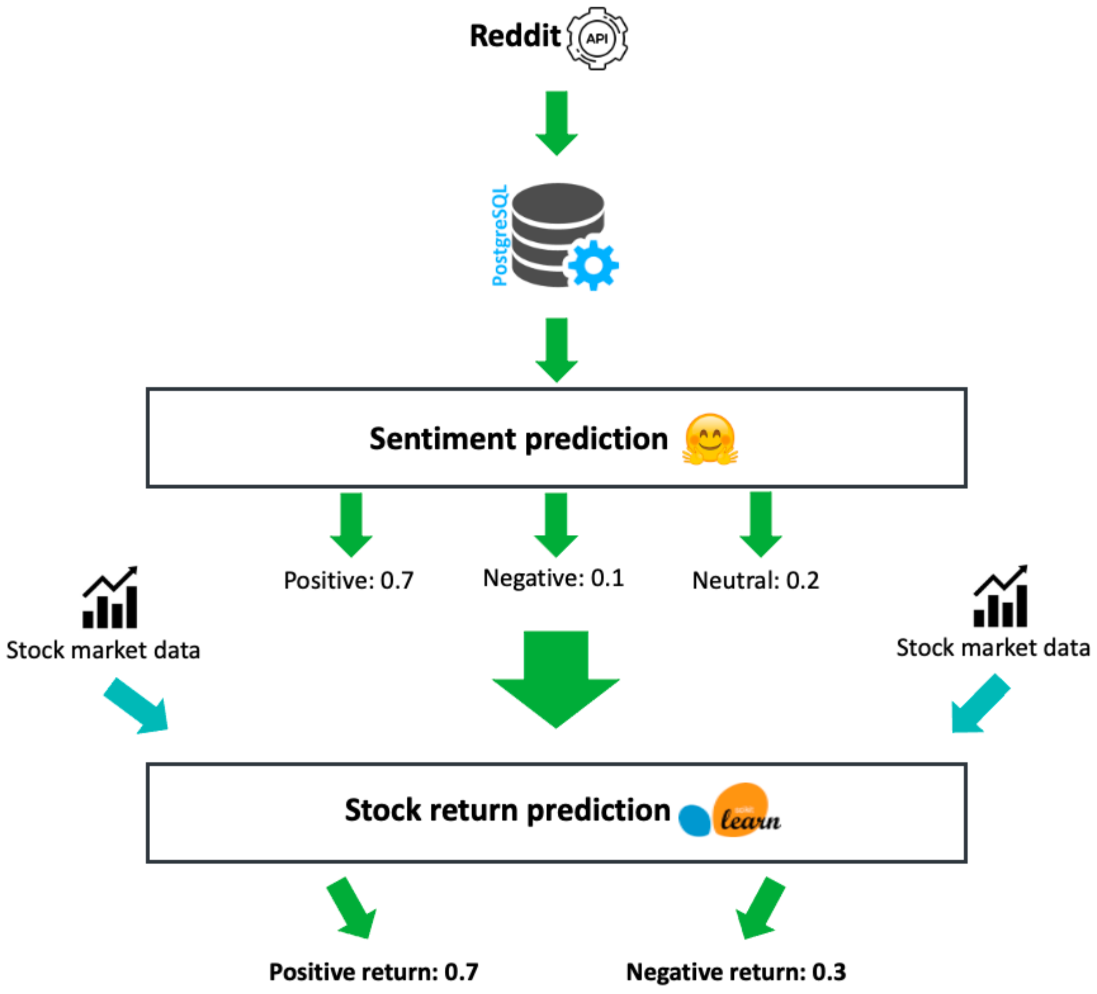


Figure 13: Data flow overview

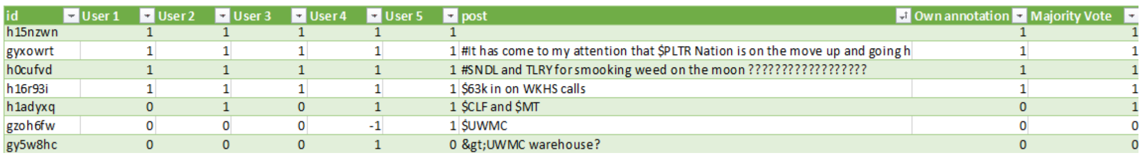
Figure 13 shows an overview of the data flow of the project. The reddit posts and comments are extracted via the reddit API and stored in a postgresql database. Based on this database, the sentiment predictions are being made. The results are probability values for each sentiment class. These probability values are aggregated by day and then together with additional financial information feed into the stock return prediction models. The final output of the pipeline are two probability values for the positive and negative stock return.

5 Sentiment Analysis

In this section, I present the sentiment analysis of my research. I explain the annotation process, the experiment setup as well as the results. I also discuss the results and answer my research question for the sentiment analysis part.

5.1 Annotation process

Currently, as of June 2022, there is no publicly annotated dataset of reddit posts available which offers enough data for my research purpose. Therefore, I annotate 1000 reddit posts manually. However, an annotation with only one person is not reliable. Hence, I chose to evaluate the inter-rater reliability between my own annotation and an already existing annotation of roughly 200 reddit posts provided by `https://bit.io/`. In their annotation, five authors annotated the reddit posts with one of the three labels positive, neutral or negative. I annotate the same posts as the five authors did and evaluate the inter-rater reliability. Based on this evaluation, I determine whether my own annotation is significant. Figure 14 shows an excerpt of the annotation process.



id	User 1	User2	User 3	User4	User 5	post	Own annotation	Majority Vote
h15nzwn	1	1	1	1	1		1	1
gyxowrt	1	1	1	1	1	#It has come to my attention that $PLTR Nation is on the move up and going h	1	1
h0cufvd	1	1	1	1	1	#SNDL and TLRY for smooking weed on the moon ?????????????????	1	1
h16r93i	1	1	1	1	1	$63k in on WKHS calls	1	1
h1adyxq	0	1	0	1	1	$CLF and $MT	0	1
gzoh6fw	0	0	0	-1	1	$UWMC	0	0
gy5w8hc	0	0	0	1	0	&gt;UWMC warehouse?	0	0

Figure 14: Excerpt of the annotation process

In Figure 14, column one represents the unique ID of the post. Columns two to six represent the annotation of the five authors. Label -1 stands for a negative annotation, label 0 stands for neutral and label 1 stands for a positive annotation. In the column ”post”, the actual reddit post is displayed. The last two columns represent my own annotation and the majority vote of the five authors. My evaluation of the inter-rater reliability is based on the majority vote from the five authors and my own annotation.

5.1.1 Label definitions

In most cases of sentiment analysis, the labels positive, negative and neutral are self explanatory. In the case of sentiment analysis for stock prediction, the labels have a specific meaning. The label ”positive” means that the reddit post or comment is associated with a rising share price of the stock it is talking about. For example, a user could talk about how he bought Apple shares and gained 10% in the last 24 hours. This post would be labeled as ”positive”. In contrast, the label ”negative” means that the reddit post or comment is associated with a falling share price of the stock it is talking about. A negative example would be a user reporting his or her losses about a given stock. The label ”neutral” is used when it is not clear whether the post is associated with a rising or falling share price of a stock, or when the post talks about a completely different topic than stocks.

There are much more expressions for either a positive label, which is associated with a rising share price, or a negative label, which is associated with a falling share price. Especially in the forum `r/wallstreetbets`, where additional new expressions are being created constantly. In Table 3 I give an overview of the most common expression associated with either a rising share price (positive label) or a falling share price (negative label) in `r/wallstreetbets`.

Positive expressions	**Negative expressions**
Call	Put
Long	Short
Buy	Sell
Bullish	Bearish
Diamond hands	Paper hands
DD (Double Down)	TDM (Threatening downward movement)
Hold	
To the moon	

Table 3: Common expressions on `r/wallstreetbets` with the associated label

Calls and puts are stock options. A call option is the equivalent of a long position and a put option is the equivalent of a short position. Bullish and bearish are expressions regarding the current market situation. Bullish means that the market has an upward trend, while bearish means that the market has a downward trend. The expressions diamond hands and paper hands refer to the buying or selling motivation of a user. Diamond hands means that the user wants to hold the stock no matter what happens. Paper hands means that the user wants to sell the stock as soon as it drops in value or negative news is spread about the stock. DD, which stands for Double Down means that a user wants to tell other users to invest money in a specific stock because from his or his subjective opinion it is a promising investment.

5.1.2 Inter-rater reliability

To measure the inter-rater reliability, I calculate the Cohen's Kappa score and take a look at the inter-rater matrix to evaluate the results based on each class.

5.1.2.1 Cohen's Kappa

Cohen's kappa, symbolized by the lower case Greek letter, κ [Marston, 2010] is a robust statistic useful for either interrater or intrarater reliability testing. Similar to correlation coefficients, it can range from -1 to +1, where 0 represents the amount of agreement that can be expected from random chance, and 1 represents perfect agreement between the raters. While kappa values below 0 are possible, Cohen notes they are unlikely in practice [Marston, 2010]. As with all correlation statistics, the kappa is a standardized value and thus is interpreted the same across multiple studies [McHugh, 2012]. Cohen's kappa is defined as:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{13}$$

Where p_0 represents the actual observed agreement, and p_e represents chance agreement [McHugh, 2012]. p_0 is also known as the accuracy and is calculated using the following formula:

$$p_0 = \frac{\sum TP + \sum TN}{\sum TP + \sum FP + \sum TN + \sum FN} \tag{14}$$

where TP = True positive, FP = False positive, TN = True negative and FN = False negative. p_e is obtained through the following formula:

$$p_e = \frac{1}{N^2} \sum_k n_{k1} n_{k2} \tag{15}$$

where k is the number of categories, N is the number of observations and n_{ki} is the number of times rater i predicted category k. For the three categories positive, neutral and negative and a number of observations of 183, I calculate the following Cohen's Kappa score:

$$\kappa = \frac{0.83 - 0.39}{1 - 0.39} = 0.72 \tag{16}$$

Cohen suggested the Kappa result be interpreted as follows: values ≤ 0 as indicating no agreement and 0.01–0.20 as none to slight, 0.21–0.40 as fair, 0.41– 0.60 as moderate, 0.61–0.80 as substantial, and 0.81–1.00 as almost perfect agreement [McHugh, 2012]. Therefore, the resulting Cohen's kappa score with 0.72 can be considered as substantial agreement.

5.1.2.2 Inter-rater matrix

Figure 15 shows the inter-rater matrix. On the x-axis, my own annotations are listed with the three labels Negative, Neutral and Positive. On the y-axis, The labels of the majority vote of five annotators are listed. The numbers on each square indicates how many posts were labeled with each category.

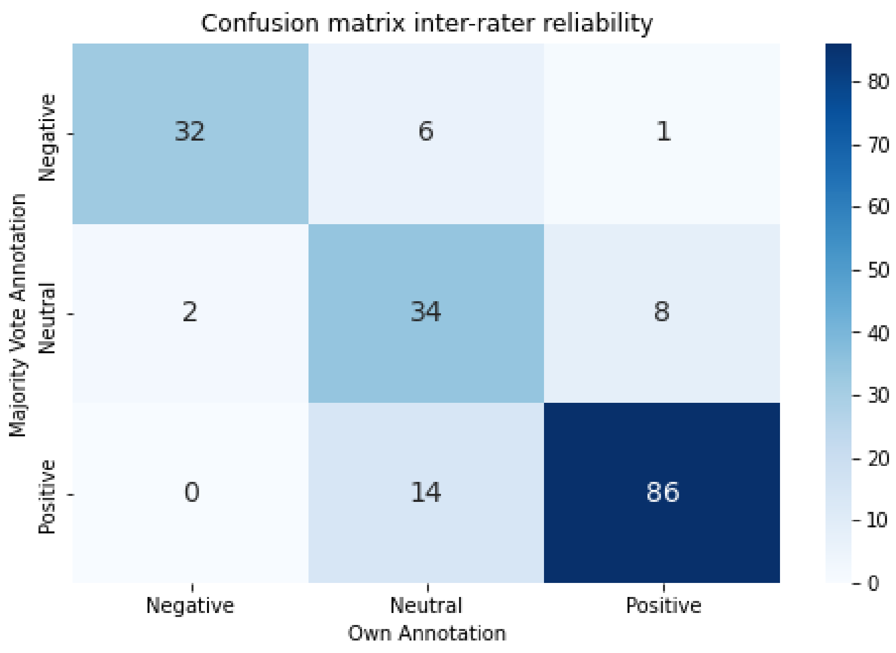


Figure 15: Inter-rater reliability confusion matrix

In total, I annotated 152 out of 183 posts with the same label as the majority vote of the five authors. Almost all posts who do not have matching labels, differ only between positive and neutral or negative and neutral. Only one case shows a complete opposite label, which I label positive and the majority vote of the five authors label it negative. This case is shown in the top right square of Figure 15.

5.1.3 Data selection and annotation results

The total dataset consists of roughly 1.6 million posts. To get a representative sample of 1000 annotated posts, I select 1000 posts from the dataset randomly and label them. If the classes are not equally distributed, I select another 1000 samples randomly from the dataset and continue the labeling process but focus on the minority class. This way I get the most representative sample while having roughly equally distributed classes. Table 4 shows the result of the annotation.

Positive	**Neutral**	**Negative**
386	316	298

Table 4: Annotation results - Distribution of the classes

The majority class is the positive label with 38.6% of all posts. The neutral class has 31.6% of all posts. The class with the least posts was the negative class with 29.8% of all posts in the dataset. Although there are slight differences in the distribution, the overall class distribution is roughly equally distributed and therefore suitable for model training. The majority class is positive because people on `r/wallstreetbets` tend to spread more positive news about a stock rather than talking negative about it or selling the stock.

5.2 Experimental setup

In this section, I explain the experimental setup for the sentiment analysis. I describe the dataset used for training, the model selection process and the training and evaluation process. I determine the performance difference between none fine-tuned and fine-tuned pretrained Neural Network transformer models on the task of Sentiment classification by calculating the accuracy and the loss of all selected models before the finetuning and after the finetuning. To determine the statistical significance, I use a dependent sample t-test. This is in order to answer the first research question:

"Is there a performance difference between none fine-tuned and fine-tuned pretrained Neural Network transformer models on the task of Sentiment classification with posts from r/wallstreetbets? And if so, how large is the performance difference and is it statistically significant?"

5.2.1 Dataset

The training dataset consists of 1000 manually annotated reddit posts within the period from May 2021 to May 2022. Each post is assigned one of the three classes, positive, negative or neutral.

5.2.2 Model selection criteria

To select pretrained transformer models for sentiment analysis, I use the open source platform `https://huggingface.co/`. One prerequisite all models should have refers to the output behavior. All models should have three output for the labels negative, neutral and positive. Furthermore, all models should be trained on the English language, since all my reddit posts are in English. If all these conditions are met, I try to pick the five most downloaded and best performing models from Huggingface and use them as a basis for my experiments.

5.2.3 Model overview

Table 5 shows an overview of the selected models and their attributes. For each model I list the name of the model, the base model and the training dataset, which was used to train the model.

Model	Base model	Training dataset
finiteautomata/bertweet-base-sentiment-analysis [Pérez et al., 2021]	BERTweet, a RoBERTa model trained on English tweets	SemEval 2017 corpus (around 40k tweets)
ProsusAI/finbert [Araci, 2019]	BERT	Financial corpus with news headlines
avichr/heBERT_sentiment-analysis [Chriqui and Yahav, 2021]	BERT	A Hebrew version of OSCAR (Ortiz, 2019)
cardiffnlp/twitter-roberta-base-sentiment-latest [Barbieri et al., 2020]	roBERTa-base	Trained on 124M tweets from January 2018 to December 2021
cardiffnlp/twitter-xlm-roberta-base-sentiment [Barbieri et al., 2021]	XLM-roBERTa-base	Trained on 198M tweets and finetuned for sentiment analysis

Table 5: Overview of the selected models and their attributes

Column two of Table 5 shows which base model was used by the authors. Pérez et al. [2021] and Barbieri et al. [2020] used a RoBERTa-base model. Barbieri et al. [2021] used a XML-roBERTa-base model. Two authors used a BERT as the base model, namely Araci [2019] and Chriqui and Yahav [2021]. Column three of Table 5 shows the training dataset, which the authors used to train their model. Three models were trained on a twitter dataset. The FinBERT model was trained on financial news headlines and the model by Chriqui and Yahav [2021] was trained on a general word corpus called OSCAR.

5.2.4 Training and evaluation process

For the training and evaluation process, I split the dataset in 80% training data and 20% testing data. I do a stratified split, meaning that the distribution of the classes are equal in both the train set and the test set. This ensures that there is no bias in either of the datasets, which can often occur, when splitting the data randomly using a small dataset, as in my case. To have the maximum training data possible, I do not split the data further into a third validation set. I also do not do any hyperparameter tuning and therefore do not need a third and final validation set.

The hyperparameters I use for each model are based on the suggestions the authors make in their paper. Except for the learning rate they are mainly the same across all models. For the optimizer, I use Adam and retrain the whole model. The loss function I use is called BCEWithLogitsLoss. This loss combines a Sigmoid layer and the BCELoss in one single class [Paszke et al., 2019]. The loss function is described as:

$$l_n = -[y_n \cdot log(\sigma(x_n)) + (1 - y_n) \cdot log(1 - \sigma(x_n))] \quad (17)$$

where n is the number of samples in one batch, x is the prediction of the model, y is the ground truth and σ represents the logistic sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{18}$$

In most cases, the authors suggest 3 - 5 training epochs. To make sure the model fully maxes out on its training process, I train each model with 10 epochs. To avoid overfitting, I save the model each time it reaches a new minimum in its validation loss. This way, at the end of the training process, the final model will be the model with the lowest validation loss throughout the training process.

5.3 Results

For all pretrained none finetuned models, I report the accuracy and the micro averaged precision, recall and f1-Score. For all pretrained finetuned models, I report the training accuracy, training loss, validation accuracy, validation loss for each training epoch. At the end of the training process, I report the micro averaged precision, recall and f1-score. In addition to that, I also report the maximum validation accuracy and the minimum validation loss the model achieved in the 10 training epochs. In this section, I describe the most important results for the sentiment analysis. After the model performance report, I go into detail about the training process and the best performing model.

5.3.1 Model performance before finetuning

Figure 16 shows an overview of the validation accuracy of all models before finetuning. The x-axis represents the accuracy and each bar represents one model with a specific accuracy value.

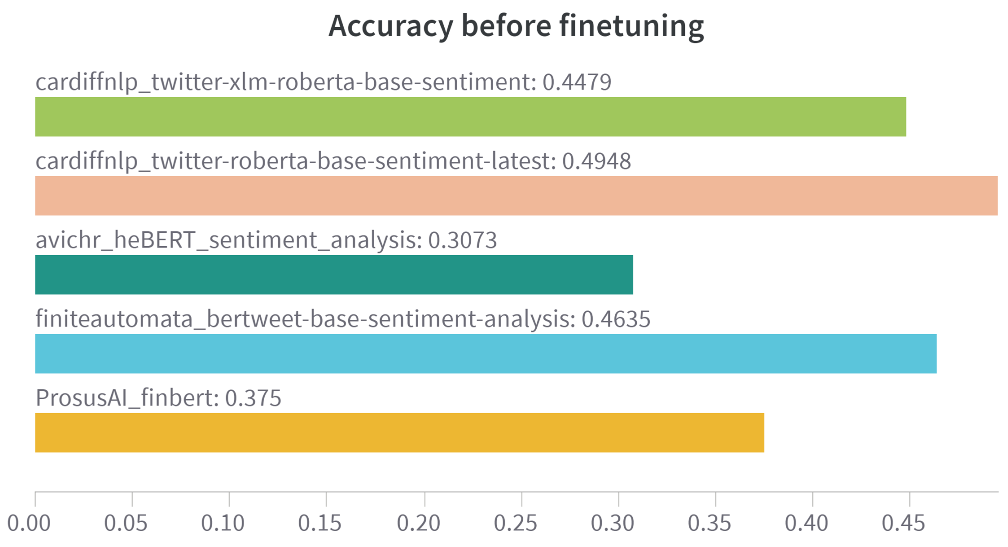


Figure 16: Model performance before finetuning

The model cardiffnlp_twitter-roberta-base-sentiment-latest achieved the highest validation accuracy with 49.5%. The lowest validation accuracy was achieved by the model avichr-heBERT_sentiment_analysis which has a validation accuracy of 30.7%. The mean validation acuracy for all models before finetuning is 41.8%.

5.3.2 Model performance after finetuning

Figure 17 shows an overview of the maximum validation accuracy of all models during the training process. The x-axis represents the accuracy and each bar represents one model with a specific accuracy value.

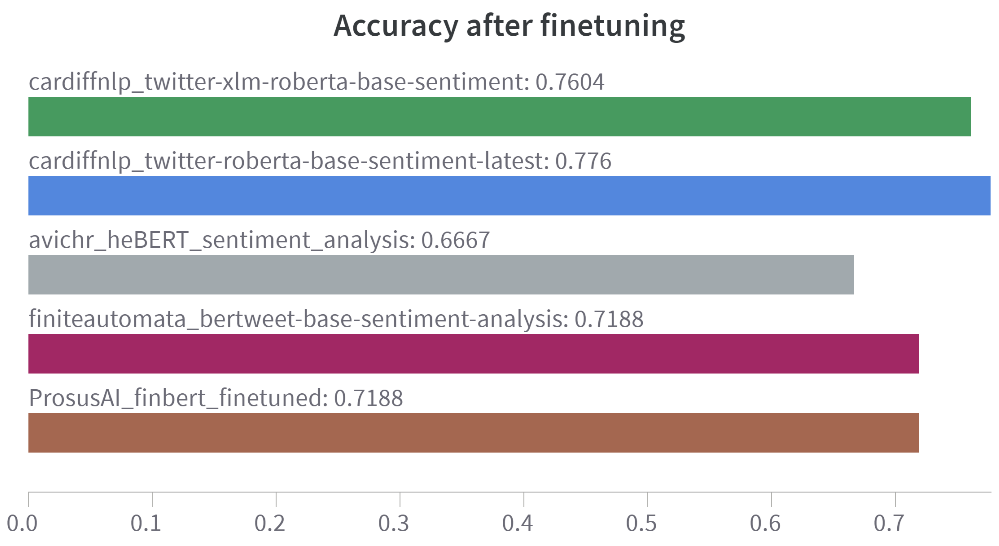


Figure 17: Model performance after finetuning

The model cardiffnlp_twitter-roberta-base-sentiment-latest achieved the highest validation accuracy with 77.6%. The lowest validation accuracy was achieved by the model avichr-heBERT_sentiment_analysis which has a validation accuracy of 66.7%. The mean validation acuracy for all models after the finetuning is 72.8%.

5.3.3 Model training progress during finetuning

Throughout the training process of 10 epochs, the training loss of all models went constantly down and the training accuracy went constantly up. Figure 18 shows the progress of the training loss while Figure 19 shows the progress of the training accuracy. The x-axis represents the training epochs and the y-axis of Figure 18 represents the training loss while the y-axis of Figure 19 represents the training accuracy.

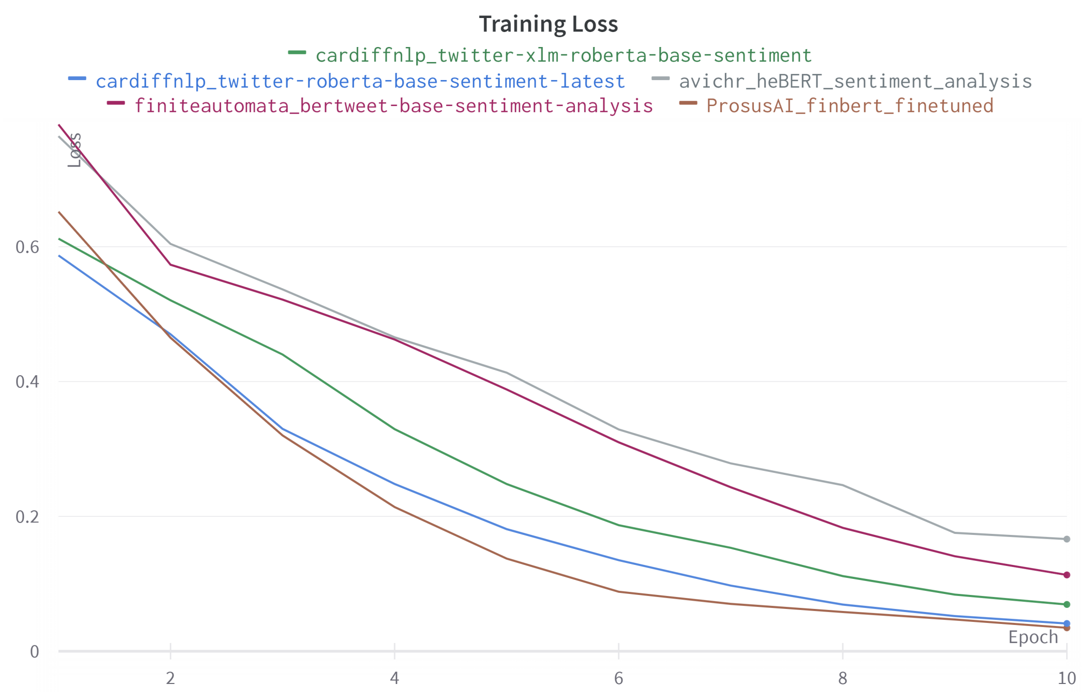


Figure 18: Progress of the training loss throughout the training process

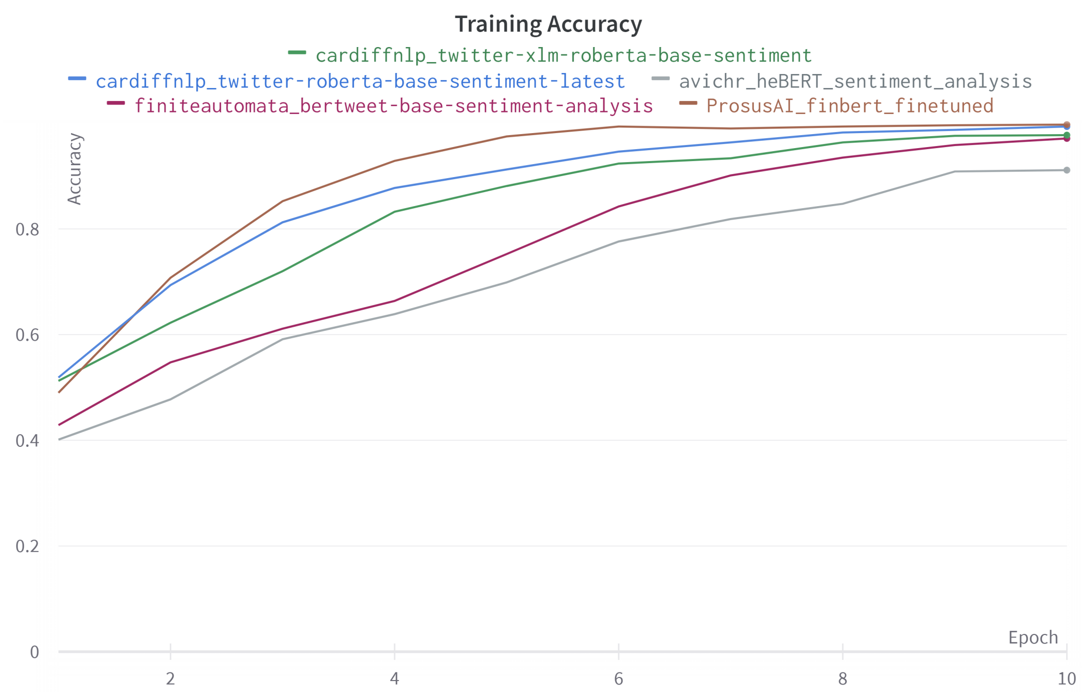


Figure 19: Progress of the training accuracy throughout the training process

The validation loss and the validation accuracy are shown in Figure 20 and Figure 21. The x-axis of both Figures represents the training epochs. The y-axis of Figure 20 represents the validation loss, while the y-axis of Figure 21 represents the validation accuracy. For all models the validation loss decreases until epoch three and starts to stagnate around epoch five. After epoch five, the validation loss starts to increase for all models. On the other side, the validation accuracy constantly increases until epoch five. After epoch five it starts to stagnate and at the end of the training process it starts to slightly decrease for most models.

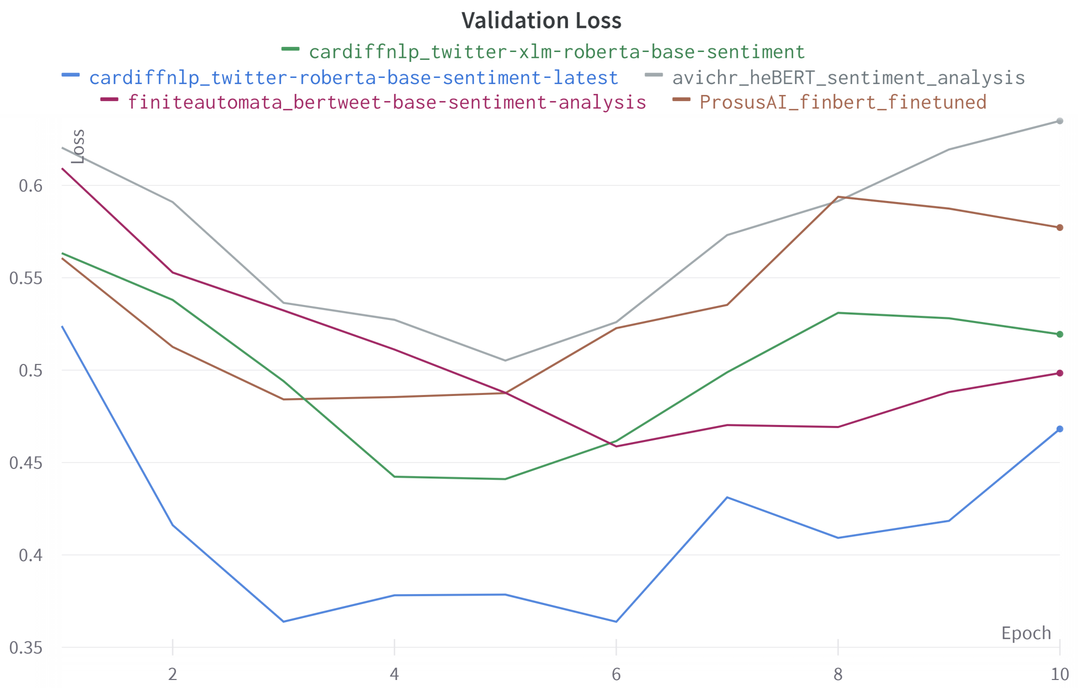


Figure 20: Progress of the validation loss throughout the training process

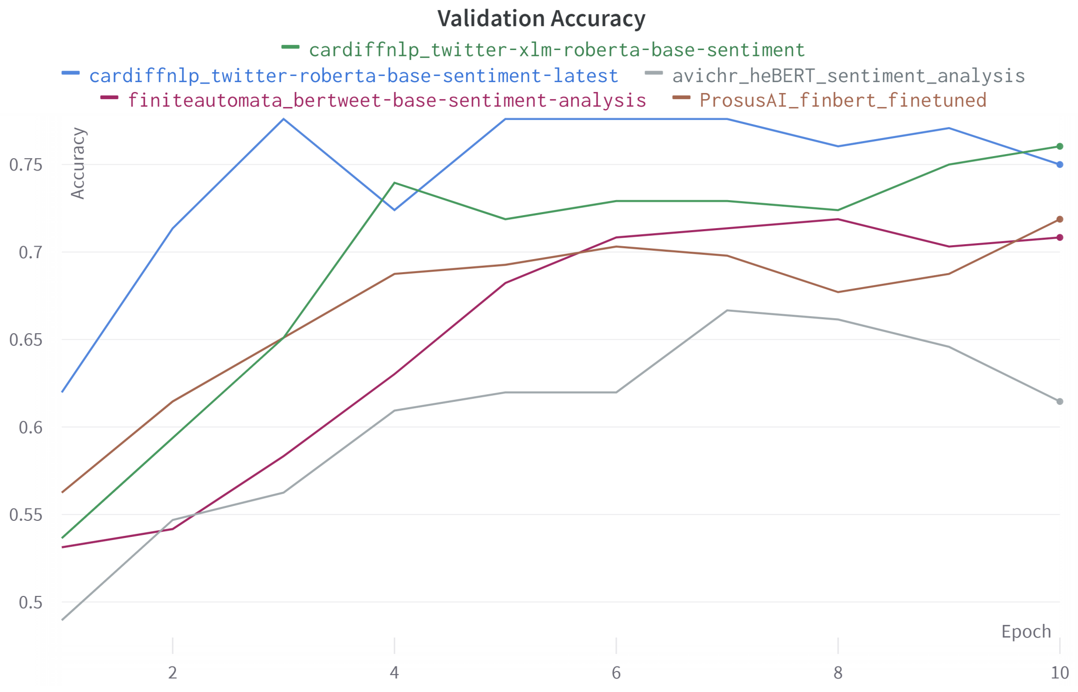


Figure 21: Progress of the validation accuracy throughout the training process

5.3.4 Model performance comparison before and after finetuning

To get an overview of the overall performance difference of all five models before and after the finetuning, I display the performance comparison in a box plot shown in Figure 22. The x-axis shows the model type, which either can be before or after finetuning. On the y-axis, the validation accuracy is displayed. Each box shows the interquartile range of the Accuracy of the five models, either before or after finetuning. Inside each box, the median is displayed with a horizontal line.

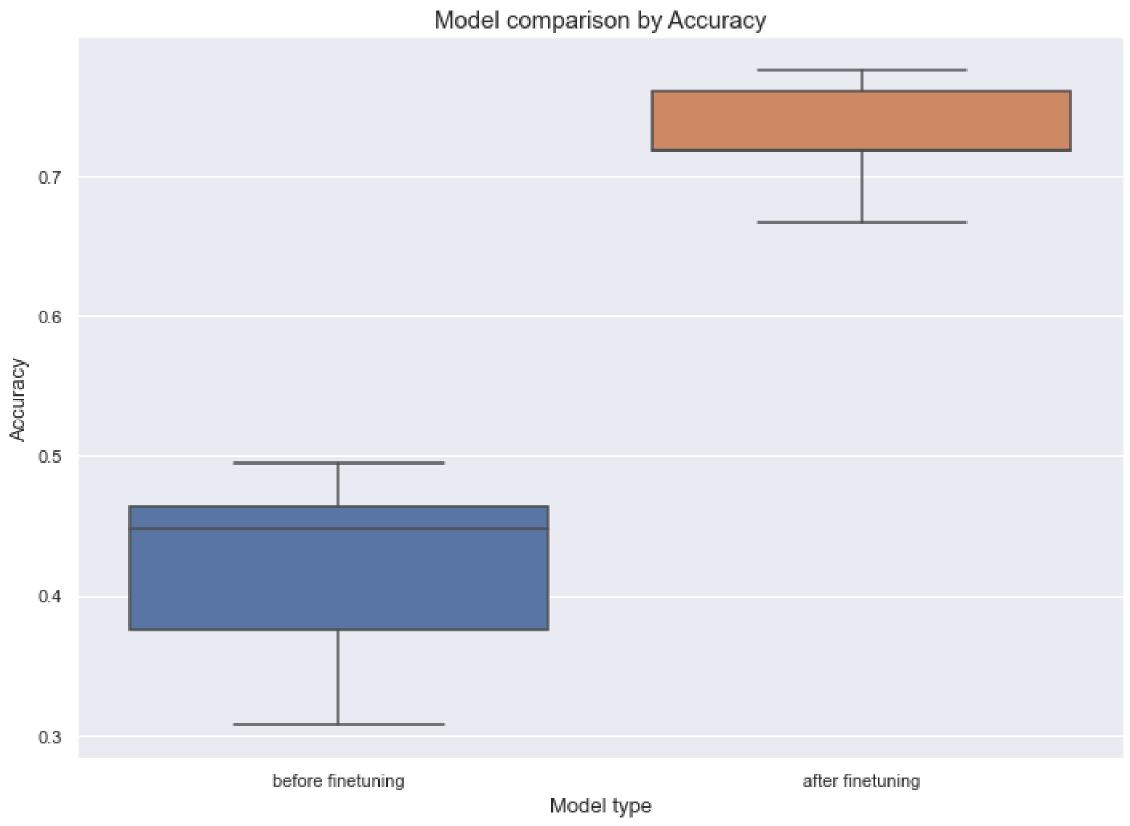


Figure 22: Model performance comparison by accuracy

The median accuracy for all models before finetuning is 44.8% and after finetuning is 71.9%. The interquartile range for all models before finetuning is 0.09 and after finetuning is 0.04.

5.3.5 Best model results

The best model with the lowest validation loss of 0.36 and the highest validation accuracy of 77.6% is the cardiffnlp_twitter-roberta-base-sentiment-latest model. The lowest validation was achieved in epoch six. Therefore, this is the final checkpoint of the model I save and later on use for the sentiment predictions of the whole dataset.

Figure 23 shows the confusion matrix for this particular model. The x-axis shows the predicted label by the model and the y-axis shows the true label.

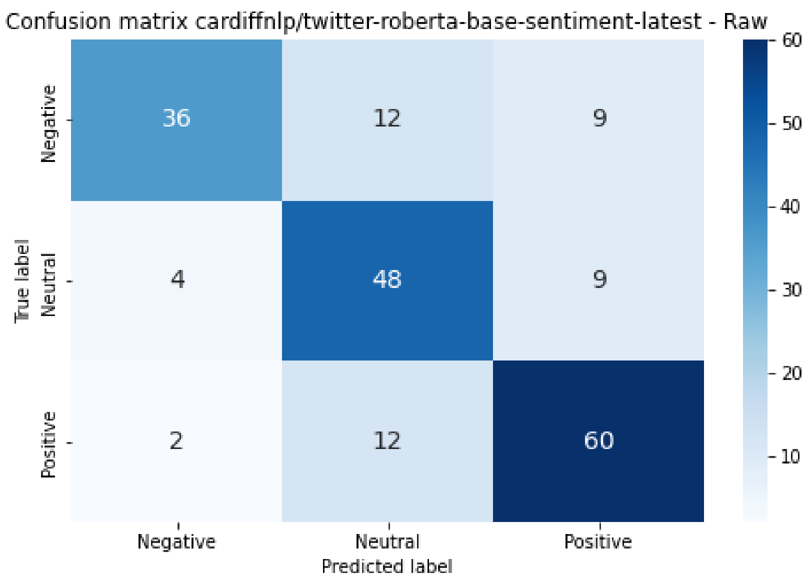


Figure 23: Confusion matrix for the model cardiffnlp_twitter-roberta-base-sentiment-latest

For the positive class, 60 samples are correctly classified as positive, 12 samples are incorrectly classified as neutral and 2 samples are incorrectly classified as negative. For the neutral class, 48 samples are correctly classified as neutral, 9 samples are incorrectly classified as positive and four samples are incorrectly classified as negative. The negative class has 36 correctly classified samples, 12 samples are incorrectly classified as neutral and 9 samples are incorrectly classified as positive.

5.3.6 Statistical significance

To determine the statistical significance for the performance between finetuned and none finetuned pretrained transformer Neural Networks, I use a dependent t-test, which is described as:

$$t = \frac{\bar{d}}{\sigma/\sqrt{n}} \tag{19}$$

where $\bar{d}$ is the sample mean of differences, σ is the sample standard deviation of the differences and n is the sample size. I do a two-tailed test where the null hypothesis is:

$$H_0 : \mu_{before} = \mu_{after} \tag{20}$$

and the alternative hypothesis is:

$$H_1 : \mu_{before} \neq \mu_{after} \tag{21}$$

The degrees of freedom are defined as $df = N-1$ which in this case is $5-1=4$. I set the significance level α to 0.05. The resulting t-statistic is -16.13, which results in a p-value of

$8.7e{-}05$. Since the p-value is less than the significance level α, I reject the null hypothesis and accept the alternative hypothesis. Therefore, the performance between finetuned and none finetuned pretrained transformer Neural Networks is statistically significant for the significance level α of 0.05.

5.4 Discussion and limitations

In this section, I discuss the results presented in section 5.3. I interpret, explain and point out the most important insights of the results.

Figure 16 shows that the three best models are all based on a roBERTa model. The two models with the worst performance, namely ProsusAI/finbert and avichr/heBERT_sentiment_analysis, both have a BERT base model. The second difference between the top 3 models and the last two models is that the top 3 models were all trained on tweets. This is interesting because of two things. First, tweets are very similar to the language used in `r/wallstreetbets`. Second, on reddit, emojis are used very often. All three model, which are roBERTa based and trained on tweets can in their word embeddings take into account these emojis. This is something ProsusAI/finbert and avichr/heBERT_sentiment-analysis are not capable of and where a lot of information is lost. All of these factors lead to the performance difference between the top 3 models and ProsusAI/finbert and avichr/heBERT_sentiment_analysis.

The model performance after finetuning is quite similar across all models as seen in Figure 17. But the three best performing models are still the same as before finetuning, namely, the two cardiffnlp models and the finiteautomata model. Therefore, if a model has a good performance before training, it tends to also have a good performance after training. This behavior also consolidates that training on tweets and taking emojis into account are two very important factors when it comes to sentiment analysis on `r/wallstreetbets`.

As the validation loss in Figure 20 and the validation accuracy in Figure 21 show, all models have a very fast learning curve and reach their maximum performance around epoch 3 - 5. After that, all models are overfitted and the validation loss starts to go up again. This behavior is often described in the original papers of the models. However, to make sure that the models reach their full training potential, I train each model 10 epochs. As mentioned in section 5.2.4, the overfitting is not reflected in the final model, since I save a model checkpoint each time a new low of the validation loss is reached.

As shown in Figure 22 the median accuracy for all models before training is 44.8% and after training is 71.9%. That is a median accuracy difference of 27.1 percentage points. What is important to consider in this case is, that the sample size with $n = 5$ is very small. Therefore, testing more models would result in a significant result. The median of the accuracy of the models after finetuning is equal to the lower bound of the interquartile range. That's why the horizontal line for the median is not really visible in this case in Figure 22.

The confusion matrix in Figure 23 shows that for the positive class, the model predicted 12 samples as neutral and only two samples as negative. This is an overall good trend because if the model's prediction is wrong for the positive class, it tends to predict neutral rather than the exact opposite negative. The same trend can be seen with the negative class. 12 samples were predicted as neutral but were actually negative and only 9 samples were predicted as positive and were actually negative. In a research which focuses exclusively on sentiment analysis, it would make sense to take a look at exactly these cases where the model predicts for example positive, but the actual class is negative.

In section 5.3.6, I calculated the test statistic and the p-value to determine the statistical significance of the difference in the means of the accuracy of all models before and after finetuning. Due to the very high difference in the means of 31% accuracy, the result is statistically significant with a p-value of $8.7e - 05$ regardless of the low sample size with $n = 5$. As already suggested above, it would still be better to have a larger sample size and test more models.

5.5 Sentiment analysis conclusion

The research question for the sentiment analysis part can be answered as follows: There is a mean performance difference between none fine-tuned and fine-tuned pretrained Neural Network transformer models on the task of Sentiment classification with posts from r/wallstreetbets of 31% accuracy. Models before training have on average 31% accuracy less than models which are finetuned on the `r/wallstreetbets` posts. Regardless of the small sample size, the result is statistically significant with a p-value of $8.7e - 05$.

6 Correlation analysis

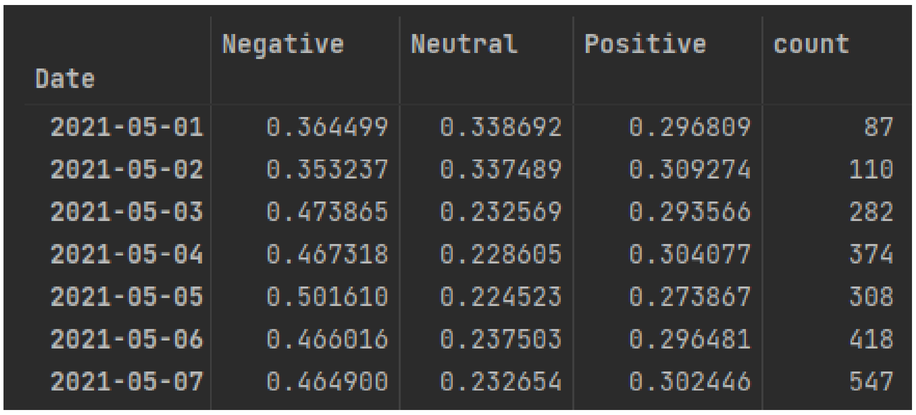
One method to measure the relationship between the sentiment of a stock and its returns is a correlation analysis. In this section, I create different sentiment features from the previous sentiment analysis and measure its correlation with the stock returns for the three most discussed stocks on `r/wallstreetbets`, namely GameStop (GME), AMC and SPDR SP 500 ETF (SPY). The analysis is restricted to the timeframe from May 2021 to April 2022.

6.1 Experimental setup

In this section, I describe the experimental setup for the correlation analysis. I explain how I aggregate the sentiment data, how I deal with missing values and which features I create from the raw sentiment values.

6.1.1 Data aggregation

After the sentiment analysis, each post is assigned a probability value for the positive, negative and neutral class. I continue to work with these probability values because they reflect the sentiment of a given stock more accurately than the absolute predictions like positive, negative or neutral. Because I want the sentiment scores for each day and not for each post, I group the posts by day and use the average of the positive, negative and neutral sentiment probabilities for each day. The resulting dataframe looks as follows:



Date	Negative	Neutral	Positive	count
2021-05-01	0.364499	0.338692	0.296809	87
2021-05-02	0.353237	0.337489	0.309274	110
2021-05-03	0.473865	0.232569	0.293566	282
2021-05-04	0.467318	0.228605	0.304077	374
2021-05-05	0.501610	0.224523	0.273867	308
2021-05-06	0.466016	0.237503	0.296481	418
2021-05-07	0.464900	0.232654	0.302446	547

Figure 24: Sentiment aggregation by day

In Figure 24, the index-column named ”Date” represents the day. The columns Negative, Neutral and Positive are the average sentiment probabilities for a given day. The column ”count” represents the amount of posts that exist for a particular stock on a given day.

6.1.2 Missing values

The sentiment probabilities are consistent throughout the dataset and each day has a probability as well as a score for the number of posts per day assigned. Since stock exchanges in general are closed on weekends and holidays, there is no stock price data available for these days. Therefore, I delete the sentiment scores on each day, where there is no stock price data available. For weekends, the result is that the sentiment probabilities from Friday predict the stock returns for Monday of the next week. The same result happens if there is a holiday during the week. The sentiment probabilities predict for the next day on which financial data is available.

6.1.3 Sentiment feature extraction

Out of the three main sentiment features, I calculate a number of different features to test their correlation with the stock returns. All features with its calculations are shown in Table 6 below:

Feature	**Calculation**
Positive, Negative	average sentiment probabilities for each day
count	Number of posts per day for a particular stock
Pos_min_neg	$Positive \times 1 + Neutral \times 0 + Negative \times -1$
Pos_min_neg_times_count	$Pos\_min\_neg \times count$
Pos_min_neg_diff	$Pos\_min\_neg_t - Pos\_min\_neg_{t-1}$
count_diff	$count\_diff_t - count\_diff_{t-1}$

Table 6: Sentiment features with calculation

As described in section 6.1.1, the positive and negative sentiment features are the average probabilities for each day and the count represents the number of posts per day for a particular stock. The feature "pos_min_neg" is a weighted sum between the base sentiment features positive, neutral and negative. It is calculated by multiplying the positive class with 1, the neutral class with 0 and the negative class with -1. As a result, the final sum of all the weights are taken. The feature "Pos_min_neg_times_count" is created by multiplying the "Pos_min_neg" feature with the count feature. The two features "Pos_min-neg_diff" and "count_diff" are calculated by subtracting their base-feature at time-point$_{t-1}$ from the base-feature at time-point$_t$. For example, if the number of posts on the 01.05.2021 was 100 and the number of posts on the 02.05.2021 was 110, the final value for the feature "count_diff" on the 02.05.2021 would be calculated as $110 - 100 = 10$.

6.1.4 Pearson correlation coefficient

To determine the correlation between each feature and the stock returns, I calculate the Pearson correlation coefficient, which can be described with the following formula:

$$r = \frac{\sum (x - m_x)(y - m_y)}{\sqrt{\sum (x - m_x)^2 \sum (y - m_y)^2}} \tag{22}$$

where m_x is the mean of the vector x and m_y is the mean of the vector y. The Pearson correlation is bound between -1 and +1, where -1 is the maximum negative correlation and +1 is the maximum positive correlation.

6.2 Results

In this section, I present the results of the correlation analysis. For the three stocks, SPY, AMC and GME, I calculate the correlation of each feature and the stock returns. After that, I present the p-values for the correlation and determine the statistical significance.

6.2.1 Correlation

Figure 25, Figure 26 and Figure 27 show the Pearson correlation between the sentiment features and the log returns of the three stocks AMC, GME and SPY. The blue bars represent the correlation between the sentiment features and the log returns at the same day. The green bars represent the correlation between the sentiment features and the log returns of the next day. The log returns are described as log_return$_t$ for the same day correlation and log_return$_{t+1}$ for the next day.

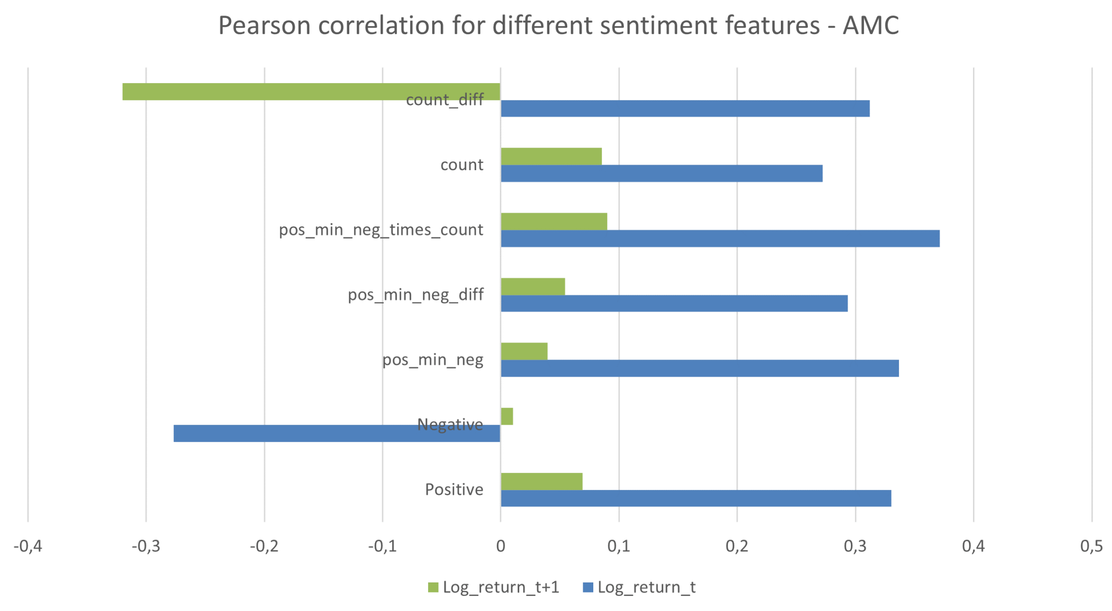


Figure 25: Pearson correlation for different sentiment features - AMC

Figure 25 shows the correlation results for the stock AMC. The highest absolute correlation for the log_return$_t$ was achieved by the feature "pos_min_neg_times_count" with a correlation of 0.37. The highest absolute correlation for the log_return$_{t+1}$ was achieved by the feature "count_diff" with a negative correlation of 0.32. In general, the absolute correlation is higher for the log_return$_t$ compared to the log_return$_{t+1}$.

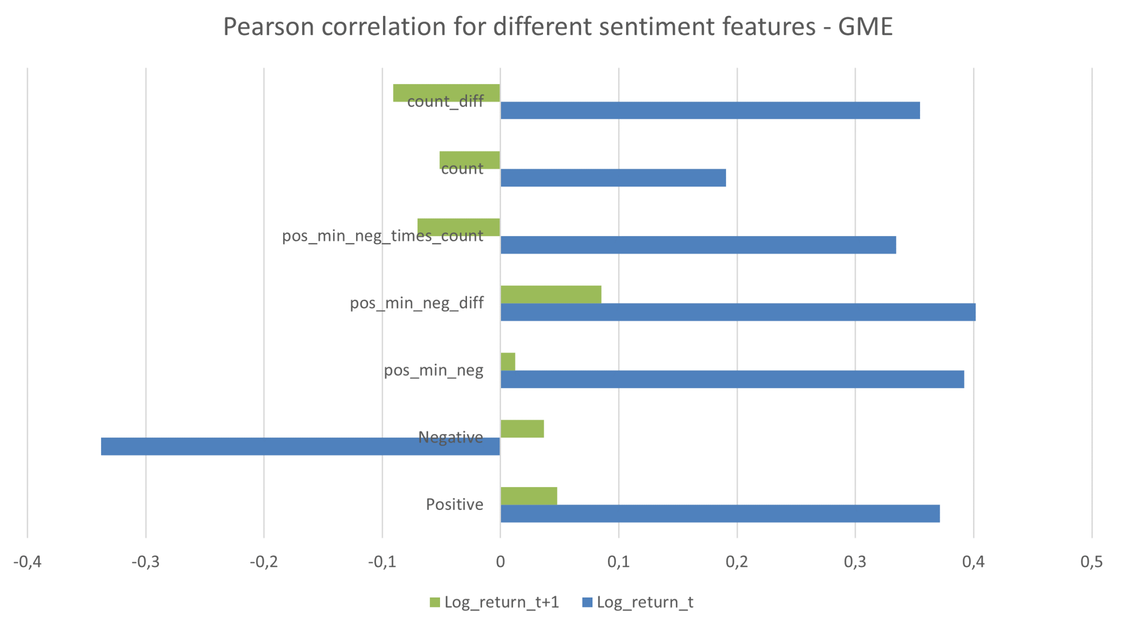


Figure 26: Pearson correlation for different sentiment features - GME

Figure 26 shows the correlation results for the stock GME. The highest absolute correlation for the log_return$_t$ was achieved by the feature ”pos_min_neg_diff” with a correlation of 0.4. The highest absolute correlation for the log_return$_{t+1}$ was achieved by the feature ”count_diff” with a negative correlation of 0.09. In general, the absolute correlation is higher for the log_return$_t$ compared to the log_return$_{t+1}$.

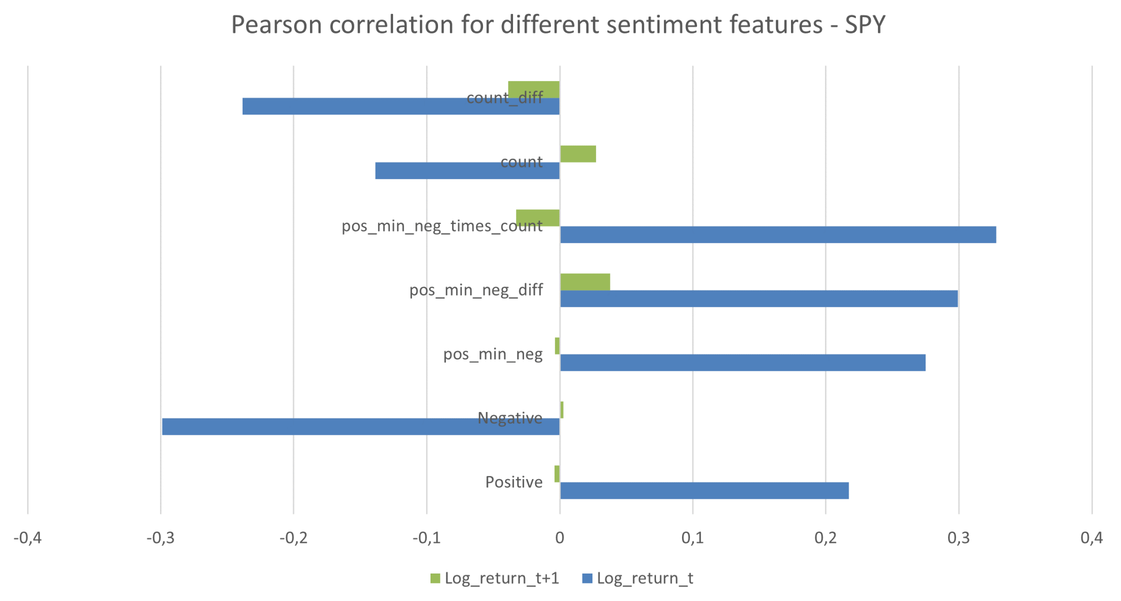


Figure 27: Pearson correlation for different sentiment features - SPY

Figure 27 shows the correlation results for the stock SPY. The highest absolute correlation for the log_return$_t$ was achieved by the feature ”pos_min_neg_times_count” with a correlation of 0.33. The highest absolute correlation for the log_return$_{t+1}$ was achieved by the feature ”count_diff” with a negative correlation of 0.04. In general, the absolute correlation is significantly higher for the log_return$_t$ compared to the log_return$_{t+1}$. Compared to the two stocks AMC and GME, SPY shows an overall lower correlation throughout all features.

6.2.2 P-values and statistical significance

Table 7 shows the p-values for all correlation measurements. The left column represents the feature for which the p-value is calculated. The three main columns each represent the results for one stock. Each stock is split up into two columns where the column t represents the correlation with the log_return$_t$ and the column $t+1$ represents the correlation with the log_return$_{t+1}$.

	AMC		**GME**		**SPY**	
Feature	t	$t+1$	t	$t+1$	t	$t+1$
Positive	8.82e-08	0.28	1.34e-09	0.45	0.0005	0.95
Negative	9.06e-06	0.87	4.36e-08	0.57	1.49e-06	0.97
Count	1.27e-05	0.18	0.002	0.42	0.03	0.67
Pos_min_neg	4.82e-08	0.53	1.31e-10	0.85	1.03e-05	0.96
Pos_min_neg_times_count	1.36e-09	0.16	6.00e-08	0.27	1.10e-07	0.60
Pos_min_neg_diff	2.35e-06	0.39	4.04e-11	0.18	1.46e-06	0.55
Count_diff	4.70e-07	2.31e-07	7.93e-09	0.15	0.0001	0.54

Table 7: P-value for correlation measurements

Table 7 shows, that for all three stocks the p-value is very low for the correlation with the log_return$_t$ and very high for the log_return$_{t+1}$. If a significance level $\alpha = 0.05$ is assumed, the correlation results for the log_return$_t$ are all statistically significant. For the log_return$_{t+1}$ all correlation results are not statistically significant, except for the stock AMC in combination with the feature "Count_diff", which shows a p-value of $2.31e-07$.

6.3 Discussion and limitations

For all three stocks, the correlation between the sentiment features and the log_return$_t$ is significantly higher than the correlation between the sentiment features and the log-return$_{t+1}$. This concludes that each feature itself is not a good predictor for the stock return for the next day. Furthermore, the p-values for the correlation between all sentiment features and the log_return$_{t+1}$ are very high and therefore the correlation is not even statistically significant. The sample size, in this case, is sufficient enough, but the correlation is very small, which impacts the p-value and increases its value.

The correlation between the sentiment features and the log_return$_t$ for all stocks lies in the range of 0.2 - 0.4 and all p-values are close to zero, which means the correlation measurements are statistically significant. Therefore, the interpretations is that the sentiment on `r/wallstreetbets` follows the current stock price and returns rather than being a predictor for the future. Because a correlation around 0.4 is not that high, it is clear that there must be more relevant sentiment data being produced about these stocks outside `r/wallstreetbets`. This could for example happen in other social media platforms such as twitter or in financial news articles. Another reason why the correlation even for the same day returns is low could be the performance of the sentiment classification model. The model used for the classification task has an accuracy of 78% and therefore an error of 22% must always be calculated into the final sentiment classification results.

One interesting correlation case I want to point out here, is the correlation of the feature "count_diff" with the log_return$_{t+1}$ from the AMC stock shown in Figure 25. The correlation of this feature and the log_return$_{t+1}$ has a value of -0.32. This means that the less often a stock is mentioned in the forum, the higher are the stock returns for the upcoming day. This logically does not make any sense and probably happened due to several limitations such as the chosen timeframe. Another reason for this behavior could be that the discussion on `r/wallstreetbets` about the AMC stock is very small compared to the overall discussion about this stock on the internet and in general. Therefore, the count of this particular forum does not play a significant role in the overall discussion volume of the AMC stock.

When comparing the average correlation between all sentiment features and the log-return$_t$, it appears that the average correlation for SPY is significantly lower than for AMC and GME. This might happen because the two stocks AMC and GME are actual stocks from one specific company. SPY however is an index fond, which means that it is an investment in multiple stocks. Therefore, SPY is more known in general than AMC and GME and has a higher discussion volume overall. This leads to the fact that `r/wallstreetbets` has less of an impact for SPY comapared to AMC and GME.

6.4 Correlation analysis conclusion

To conclude the correlation analysis, there is a significant correlation between the sentiment features and the log_return$_t$ with the highest correlation measured for the feature "pos_min_neg_diff" and the GME stock with 0.4. There is no statistically significant correlation for any sentiment feature with the log_return$_{t+1}$. Furthermore, the total number of posts on a given day did not show a higher correlation than the feature "pos_min_neg-diff", which is the feature with the highest correlation.

7 Predictive modeling

Predictive modeling in the field of stock value prediction, as discussed in section 3.1.7, can be done with regression and classification. In this section, I do stock value classification for the three stocks AMC, GME and SPY. I use the same machine learning algorithms for each stock and measure their performance individually.

7.1 Experiment setup

The timeframe for the stock return predictions is limited to the data which was extracted earlier and explained in section 4. It ranges from 01 May 2021 to 30 April 2022. Since stock exchanges are not open on weekends, there are no stock prices available for these days. Thus, the total number of datapoints in each dataset decreases to 250. I predict the log_return_{t+1} since the final goal of a stock value prediction is to predict the return of the next day. In the case of missing stock price data for the weekends, this means that the Friday of each week will predict the returns for Monday of the next week.

To split the data into train and test set, I use a stratified split with a testsize of 25% and a trainsize of 75%. Compared to the sentiment classification, the testsize is a little larger, because there is fewer data in this dataset and the goal is to make the testset as representative of the trainset as possible.

7.1.1 Stock return classification

Stock return classification uses accuracy as an evaluation method. This is more intuitive compared to stock return regression, which uses RMSE as an evaluation method. Furthermore, the simplification of using classes instead of the exact stock return oftentimes increases the model's performance. To prepare the classification, I group the stock returns into two classes. Positive stock returns are labeled with +1 and negative stock returns are labeled with -1. The distribution of the classes is roughly equal for all three stocks.

7.1.2 Features

For the stock return prediction, I use the same features explained in section 6.1.3, namely Positive, Negative, Neutral, Count, Pos_min_neg, Pos_min_neg_times_count, Pos_min-neg_diff and count_diff. In addition to that, I also add the trading volume, which is a measure for how many shares of a stock were traded on a particular day.

7.1.3 Algorithms

The algorithms used for the stock return prediction are: Logistic Regression, Decision Tree, Random Forest, XGBoost, K-Nearest Neighbour and a Neural Network. Since I do not do any hyperparamter tuning and further optimizations, I use all algorithms with their standard hyperparameters defined by sklearn, which is the python package I use to implement these algorithms. In addition to the quantitative analysis via the supervised algorithms, I also use an unsupervised algorithm for dimensionality reduction called Principal Component Analysis (PCA). This algorithm reduces the high dimensional feature space into 3 dimensions and allows a visual analysis of the data.

7.1.4 Data normalization

To bring all features into the same scale, a best practice before predictive modeling is to normalize the data. To do this, I use the Z-Score method, which can be defined as follows:

$$Z = \frac{x - \mu}{\sigma} \tag{23}$$

where x represents one datapoint in a feature, μ is the mean of the feature and σ is the standard deviation of the feature.

7.2 Results

First, I give an overview of the model performance for each stock. Second, I present the results of the PCA and at the end of this results-section, I investigate the performance of the best model in detail.

7.2.1 Model performance overview

Figure 28, Figure 29 and Figure 30 show the train and test accuracy for all models explained in section 7.1.3 for the three stocks SPY, AMC and GME. The x-axis represents the accuracy and the y-axis shows the respective algorithm.

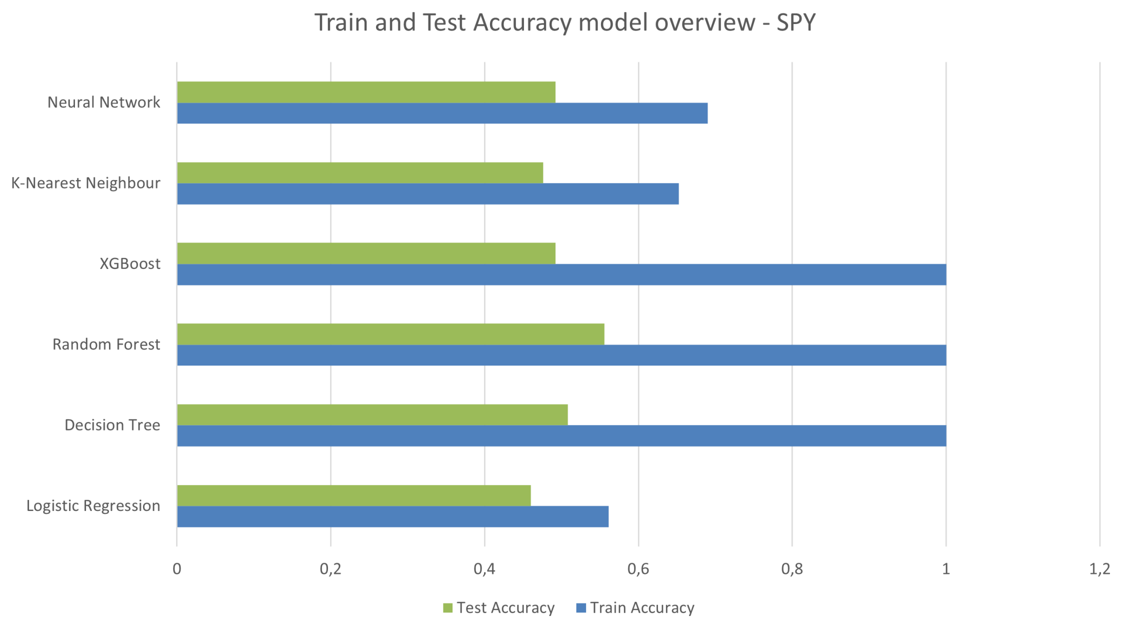


Figure 28: Train and test accuracy for the different models - SPY

The best model for the stock return prediction of SPY is the Random Forest with a test accuracy of 55.5%. In general the three models Decision Tree, Random Forest and XGBoost, which are all part of the same model family, have the highest test performance and all reach a train score of 100% accuracy.

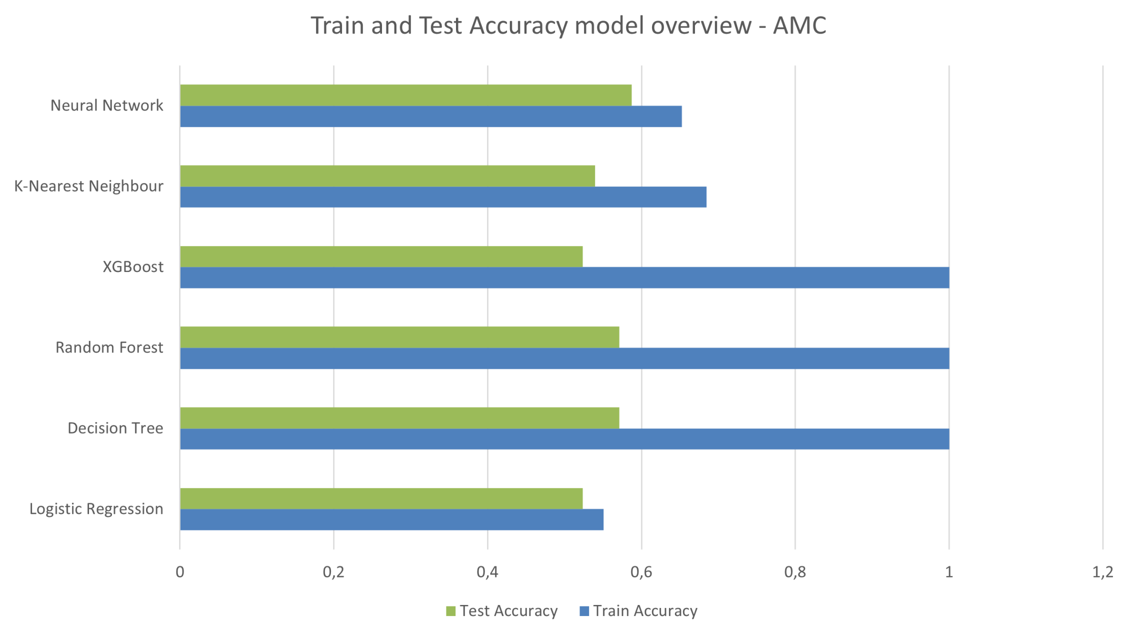


Figure 29: Train and test accuracy for the different models - AMC

The best model for the stock return prediction of AMC is the Neural Network with a test accuracy of 58.7%. The second and third-best models are the Decision Tree and the Random Forest with a test accuracy of 57% each. The Logistic Regression has the worst performance with a test accuracy of 52%.

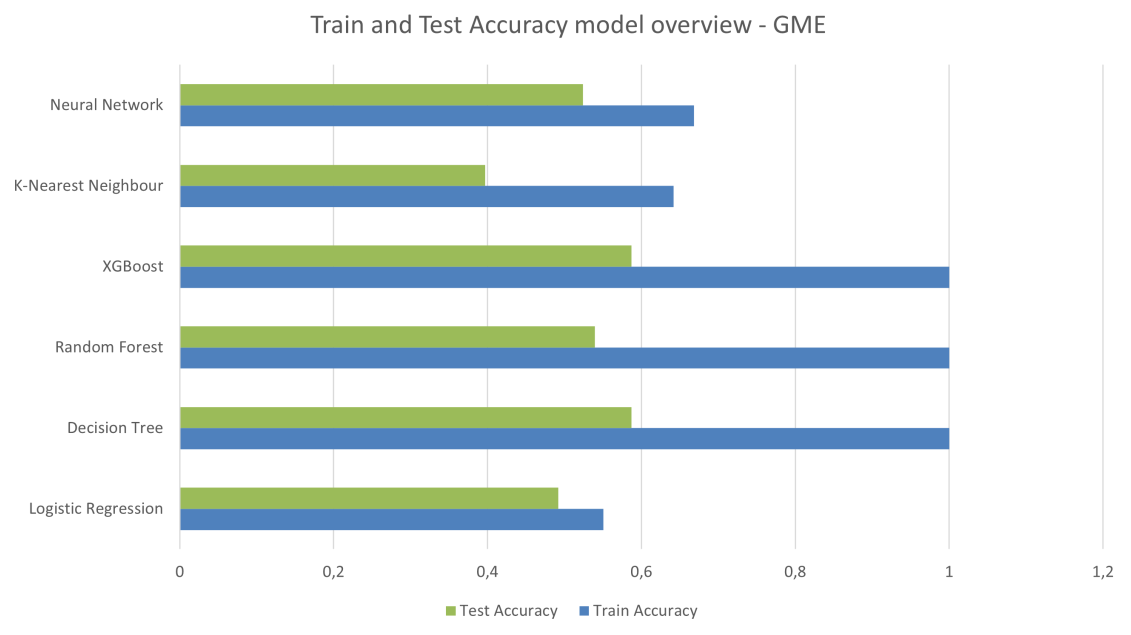


Figure 30: Train and test accuracy for the different models - GME

The best model for the stock return prediction of GME is XGBoost with a test accuracy of 58.7%. The worst model is K-Nearest Neighbours with a test accuracy of 39.6% which is even worse than flipping a coin, considering this is a binary classification problem. All models across all stocks in general show a similar behavior with a very low test accuracy below 60%. The model family of the Decision Tree, namely the Decision Tree itself, the Random Forest and XGBoost all show a train accuracy of 100%. The K-Nearest Neighbour Model is in general amongst the worst models and in the case of GME even shows a worse performance than flipping a coin.

7.2.2 Principal Component Analysis

The original dataset consists of nine features. I use the PCA to reduce the nine dimensional feature space into three dimensions. I visualize the three features produced by the PCA in two different plots. Figure 31 shows the three principal components with labels for the log_return$_t$, which are the returns for the same day. Figure 32 shows the three principal components with labels for the log_return$_{t+1}$, which are the returns for the next day. Positive returns are colored in yellow and negative returns are colored in blue. The color bar on the right of each Figure also emphasizes this behavior. Due to the reoccurring similarity between the three stocks, Figure 31 and Figure 32 show the PCA results for the GME stock only.

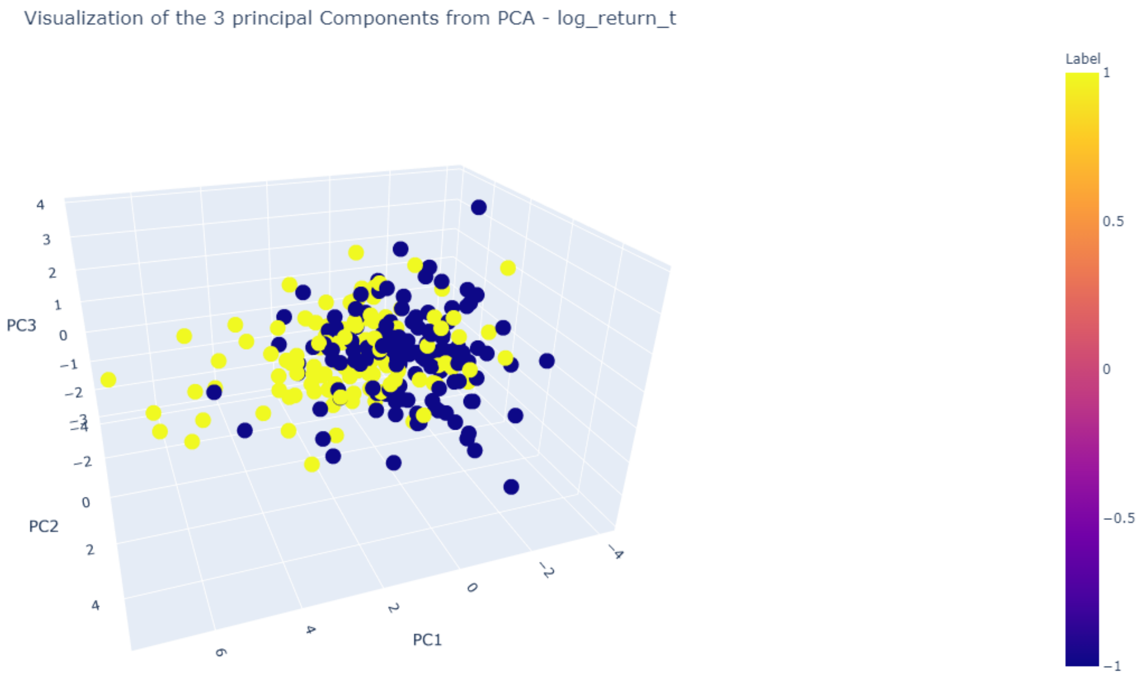


Figure 31: Visualization of the three principal components for the log_return$_t$ - GME

Figure 31 shows a slight separation of positive and negative returns on the horizontal axis, which represents PC1. The higher the value of PC1, the more positive returns can be seen, which are marked in yellow. The lower the value of PC1, the more negative returns can be seen, which are marked in blue.

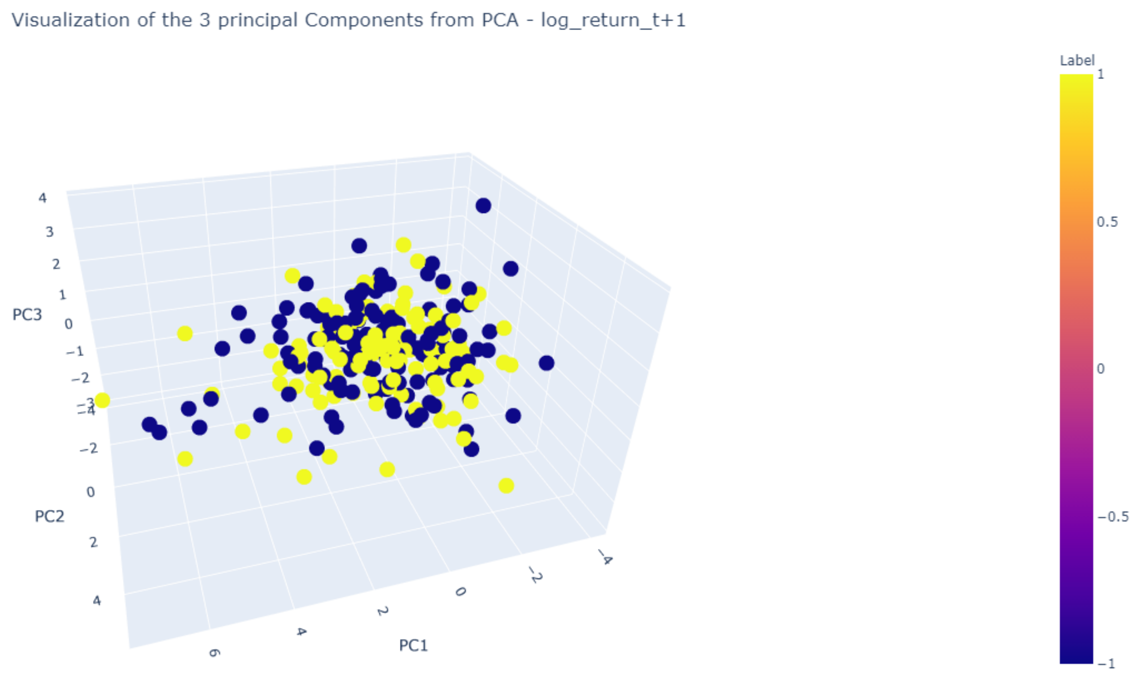


Figure 32: Visualization of the three principal components for the log_return$_{t+1}$ - GME

Figure 32, where the colored labels represent the log_return$_{t+1}$, does not show any possible separation between positive and negative returns. No principal component tends to separate the datapoints. There are also no clusters of data visually recognizable.

The principal components of the PCA are for themselves not interpretable anymore. To get an idea which principal components represents which feature, I calculate the correlation between each feature and each principal components. The result is shown in Figure 33. On the x-axis, each principal component in combination with all features is shown and on the y-axis, the respective absolute Pearson correlation is presented.

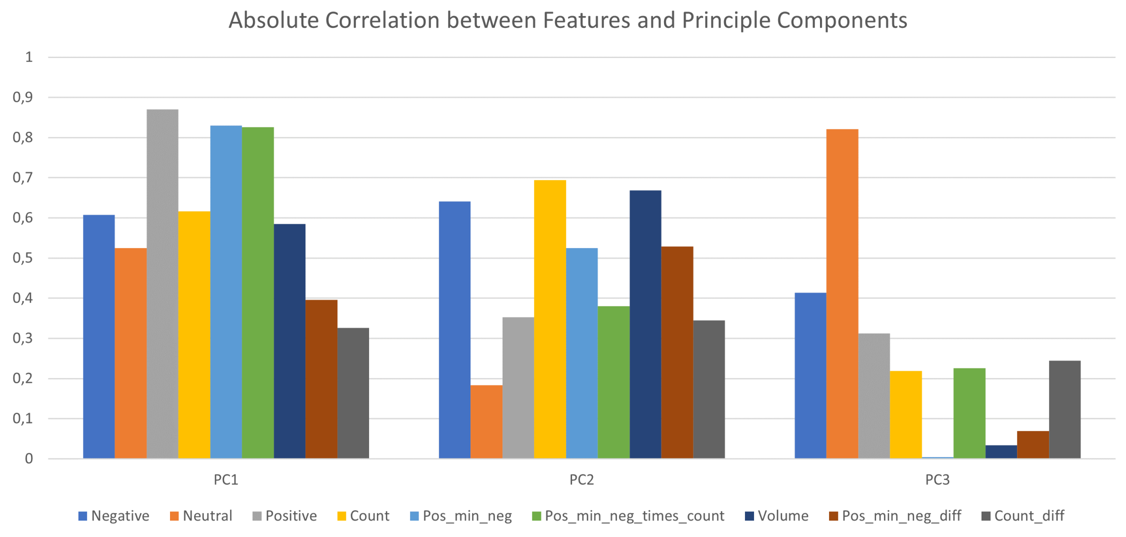


Figure 33: Absolute correlation between the features and the principal components

Figure 33 shows that PC1 mostly represents the three features Pos_min_neg, Pos_min-neg_times_count and Positive. The three features Count, Negative and Volume have the highest absolute correlation with PC2. PC3 has only one Feature with a high absolute correlation, which is the Neutral feature. The three main sentiment features, Positive, Negative and Neutral, which the correlation analysis in section 6 showed as the three most important features, are all split between the three principal components. The sentiment feature Positive has a high correlation with PC1, the feature Negative has a high correlation with PC2 and the feature Neutral has a high correlation with PC3.

The cumulative explained variance can give an idea of how much variance in the data was lost by the reduction of the feature space. Figure 34 shows the cumulative explained variance by the PCA. The x-axis shows the number of components used and the y-axis shows how much variance of the original data is left in percent. The blue vertical line shows that in this case, three principal components were chosen. The red horizontal helps to identify how much variance the black line refers to.

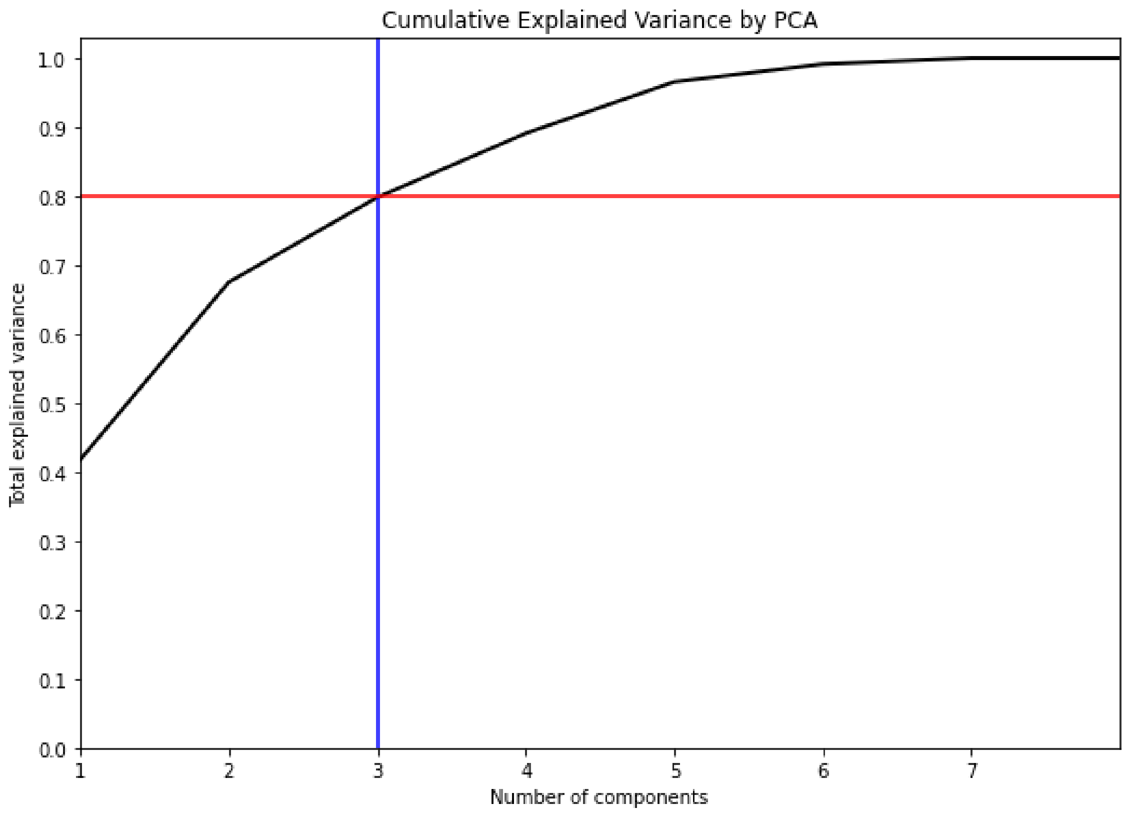


Figure 34: Cumulative explained variance by PCA

Figure 34 shows that with three components, 80% of the variance of the original dataset are obtained. In other words, 20% of the variance was lost during the reduction of the feature space from nine dimensions to three.

7.2.3 Best model investigation

The highest test accuracy was achieved by the XGBoost model for the GME stock, which has a test accuracy of 58.7%. For this particular model, I create a raw and normalized confusion matrix and compare the classified stock returns to its original exact returns. Figure 35 shows the raw confusion matrix for the XGBoost model. The x-axis represents the predicted label, and the y-axis shows the true label.

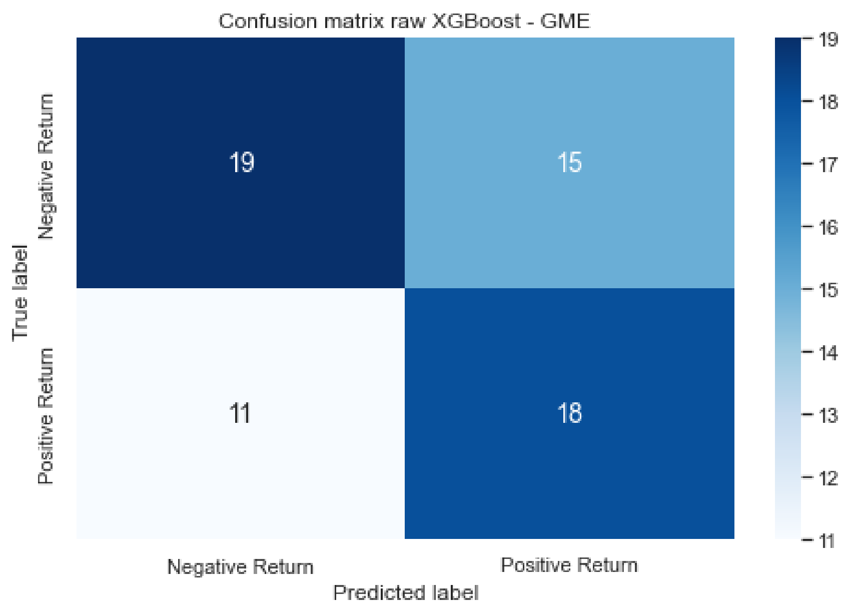


Figure 35: Raw confusion matrix of the XGBoost model - GME

In total, the model classified 19 datapoints correctly as a negative return and 18 datapoints correctly as a positive return. In 11 cases the model predicted a negative return, but the actual return for the next day was positive. In 15 cases, the model predicted a positive return, but the actual return for the next day was negative. Therefore, the resulting precision for the model is 54.5%, the recall is 62.0% and the f1-score is 58.0%.

Figure 36 shows the normalized confusion matrix for the XGBoost model. The axis have the same labeling as Figure 35. The data is normalized for the true labels, meaning that the percentages of each row, which represents one true category, sum up to 100%.

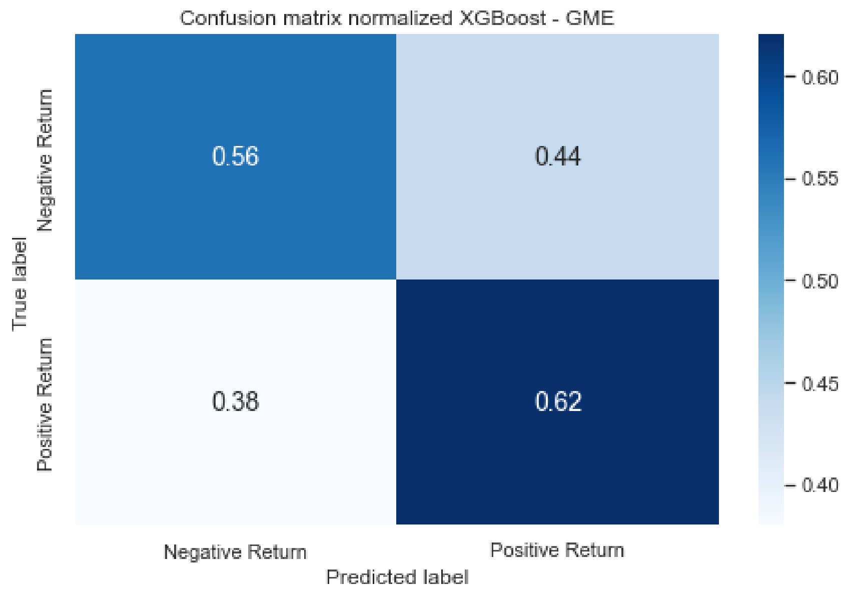


Figure 36: Normalized confusion matrix of the XGBoost model - GME

Figure 36 shows that out of all negative returns, 56% were classified correctly and out of all positive returns, 62% were classified correctly by the model. Therefore, the true positive rate is higher than the true negative rate.

One last visual investigation I present is the relation between the classified log returns and the original exact returns. One logical assumption would be that the model makes more mistakes when the original exact return is close to zero and makes fewer mistakes when the original return has either a very high positive or negative value. Figure 37 shows this exact scenario. The x-axis represents the original exact return. The y-axis shows the true label for the classification, which can be either be -1 or +1. The color shows the predicted label by the model. Dark points represent a positive return prediction and bright points represent a negative return prediction.

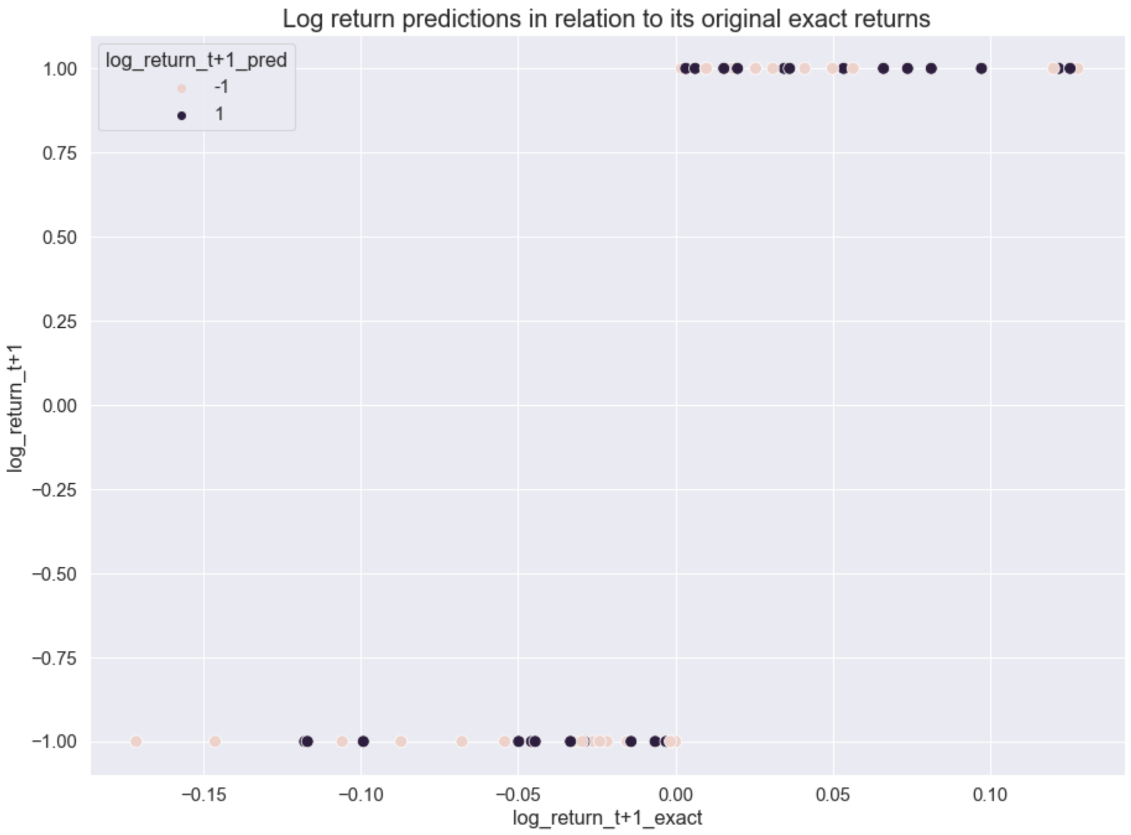


Figure 37: Log return predictions in relation to its exact returns

Figure 37 shows that datapoints to the far left as well as to the far right, which represent either high positive or negative returns, are in general classified correctly. The majority of the mistakes by the model is done when the stock returns are close to zero.

7.3 Discussion and limitations

The model performance overview in section 7.2.1 shows that the Logistic Regression together with the K-Nearest Neighbour algorithm have the lowest test accuracy. This is plausible, because the Logistic Regression is a linear model and the target variable, as shown in section 7.2.2, is clearly not linearly distributed. The K-Nearest Neighbour algorithms may have not performed well due to a phenomenon called ”the course of dimensionality”, which states that in a higher dimensional space, the distance based procedure of the algorithm does not work well anymore, because the distances between all datapoints do not differ anymore.

In general, it can be said, that all models have a very poor performance with less than 60% test accuracy. This behavior was expected for a number of reasons. First, the overall dataset, with only 250 samples, is very small. Second, as shown in the correlation analysis in section 6 all three main sentiment features did not show a significant correlation with the log_return$_{t+1}$. And third, even the visualization of the PCA did not show any possible separation of the data. All of this leads to the poor performance of the models.

The PCA for the log_return$_t$ did show some separation potential for the two classes, mainly on the scale of PC1, as shown in Figure 31. However, there is no separation potential for the two classes identifiable if the datapoints are labeled with the log_return$_{t+1}$, which is shown in Figure 32. As shown in Figure 33, the three main sentiment features Positive, Negative and Neutral all have a very high correlation with a specific principal component. This again emphasizes, that these features are the main predictors for the machine learning models. The cumulative explained variance by the PCA for three principal components is still 80%, which is very high. Therefore, the visualization of the PCA does give a very accurate impression of the original distribution of the data.

The best model, mentioned in section 7.2.3, has a higher recall than precision, which indicates that the model is better in finding all possible positive returns, but therefore also predicts too many samples as positive which leads to a higher false positive count, which then again leads to a lower precision.

Figure 37 shows the log return predictions in relation to its original exact stock returns. The assumption, that the model makes more mistakes when the original exact return is close to zero and makes fewer mistakes when the original return has either a very high positive or negative value, can at least visually be confirmed. However, since there are only 62 datapoints in the testset, this result has to be viewed with caution.

In general, the biggest limitation for the predictive modeling is the small dataset. It consists of only 250 samples, which represent one year of time series data of reddit posts and comments.

7.4 Predictive modeling conclusion

The research question for the predictive modeling section can be answered as follows: Given a dataset with time series data of reddit posts and comments from one year, containing a total of 250 entries, the best performance was achieved by the XGBoost model for the GME stock with a test accuracy of 58.7%. The three most important features are the sentiment features Positive, Negative and Neutral. The model's performance is mainly limited by the size of the dataset and the quality of the predicted sentiment features Positive, Negative and Neutral.

8 Conclusion

In this thesis, I ran several experiments to answer the final research question to which extent the sentiment of a given stock discussed on `r/wallstreetbets` can predict its returns. Leading up to this question, I did a sentiment analysis using Transformer Neural Networks. The best model with the lowest validation loss of 0.36 and the highest validation accuracy of 77.6% was the cardiffnlp twitter-roberta-base-sentiment-latest model. The biggest limitation for the sentiment classification models was the size of the dataset with 1000 samples and the fact that the labeling of the reddit posts and comments was oftentimes very subjective. Also in some posts, positive and negative signals were mixed. Despite these limitations, a mean performance difference between none fine-tuned and fine-tuned pretrained Neural Network transformer models on the task of Sentiment classification with posts from r/wallstreetbets of 31 percentage points of accuracy was achieved. Regardless of the small sample size, the result is statistically significant with a p-value of $8.7e - 05$.

Out of the three main sentiment features, Positive, Negative and Neutral, I developed a number of other features and tested their correlation with the log_return$_t$ for the same day and the log_return$_{t+1}$ for the next day. This procedure was done for the three most discussed stocks on `r/wallstreetbets`, namely SPY, AMC and GME. I found that there is a significant correlation between the sentiment features and the log_return$_t$ with the highest correlation measured for the feature "pos_min_neg_diff" and the GME stock with 0.4. There is no statistically significant correlation for any sentiment feature with the log_return$_{t+1}$. Furthermore, the total number of posts on a given day did not show a higher correlation than the feature "pos_min_negdiff", which is the feature with the highest correlation.

In the last part of my research, I ran several experiments and tested the performance of different machine learning models for the task of stock return classification. The overall performance of the models for all three stocks strengthen the results of correlation analysis. The best performance was achieved by the XGBoost model for the GME stock with a test accuracy of 58.7%. The results of the PCA also gave a visual impression that a separation of the data with the log_return$_{t+1}$ label might not be possible, at least with the current size of the dataset.

All of these results lead to the final conclusion, that the sentiment of the stock discussions on `r/wallstreetbets` does not drive the stock returns on the upcoming day. Due to the statistically significant correlation between the sentiment features and the stock returns of the same day, it can be concluded that the discussion on `r/wallstreetbets` reflects the current stock price movements and stock returns, but does not give any indication of how future stock prices and returns might develop.

9 Future work

Due to the complexity of this topic and the limitations mentioned above, a number of issues arise for future work. Regarding the sentiment analysis, to get a more accurate labeled dataset for training, a higher number of annotators are necessary. Oftentimes, the labeling process is very subjective. With clear annotation guidelines and for example five annotators, where the final labels are chosen by majority vote, a higher accuracy for all models could be achieved. Furthermore, the model accuracy could be improved by a larger dataset with more training data. During my research, I only tested the performance of five finetuned transformer models. Even though the performance difference between finetuned and none-finetuned transformer models was already statistically significant, a larger model selection would lead to an even more statistically significant result. In addition to that, it would also be interesting to see the performance of a transformer model, which was not trained on a sentiment classification task before. For example, a plain BERT model would be an interesting case. The reason for that is, that the pretrained sentiment classification models, without finetuning, did not achieve a high accuracy. Maybe the language on `r/wallstreetbets` is too specific and a pretrained model on a generic sentiment classification task like twitter posts might not be the best starting point for the finetuning process. The last interesting part for the sentiment analysis would be an in depth error analysis with specific examples in which cases the model makes mistakes. Complementary to that, using a hyperparameter tuning process could lead to an even higher accuracy of the models. All these mentioned aspects above lead to a better quality in the training data as well as the training process and eventually to a higher accuracy of the models.

In my research, I tested the correlation of the three most discussed stocks on `r/wall-streetbets`, namely SPY, AMC and GME. These are all very prominent stocks and the general discussion volume of reddit posts might be very small compared to the total amount on the internet and in general. It would be interesting to see a comparison with not so well-known stocks, which have a lower discussion volume in general and where `r/wallstreetbets` might have a larger proportion of the total discussion volume. A correlation analysis and predictive modeling approaches would be an interesting topic for future work in this case. The machine learning models in my research predict the log return of the next day based on sentiment information of the current day. Other possible timeframes for predictions would be a two-day window or even a weekly prediction. This means that the sentiment information of the last two days or the last week could predict the stock returns of the upcoming day or even the average return of the upcoming week.

Because my research did mainly focus on the sentiment features for stock return prediction, the time series aspect, which comes with stock data was neglected. This time series aspect could be combined with the sentiment features. For example, a typical time series algorithm such as a LSTM could be used to predict the return$_{t+1}$ based on the stock price or the stock returns of return$_t$, return$_{t-1}$, return$_{t-2}$ etc., depending on the defined sequence length of the LSTM. This algorithm could be combined with the sentiment features, Positive, Negative and Neutral. The result would be an algorithm which does consider the sentiment information of reddit as well as the performance of the stock during the previous days, which in financial time series forecasting is a very important factor.

The last future work regarding the predictive modeling part refers to the supervised method used for the stock return predictions. In my research, I focused on stock return classification. But it is also possible to do a regression and try to predict the exact stock returns. In fact, in combination with a time series algorithm, this would even make more sense than using a classification.

Since the topic of stock return prediction is so complex, considering other data sources such as financial information about a company or other sources of stock discussions such as twitter data could be a next step for further research.

List of Figures

List of Tables

References

An introduction to recurrent neural networks and the math that powers them. https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/.

Illustrated guide to recurrent neural networks. https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9.

Recurrent neural network (rnn). https://kobiso.github.io/research/research-rnn/.

Apewisdom project. https://apewisdom.io/wallstreetbets/.

Financial engineering. https://www.investopedia.com/terms/f/financialengineering.asp.

Deep neural network example architecture. https://www.javatpoint.com/single-layer-perceptron-in-tensorflow.

Pushshift api. https://pushshift.io/api-parameters/.

D. Araci. Finbert: Financial sentiment analysis with pre-trained language models, 2019. URL https://arxiv.org/abs/1908.10063.

G. Attanasio, L. Cagliero, P. Garza, and E. Baralis. Combining news sentiment and technical analysis to predict stock trend reversal. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 514–521, 2019. doi: 10.1109/ICDMW.2019.00079.

F. Barbieri, J. Camacho-Collados, L. Espinosa-Anke, and L. Neves. TweetEval:Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Proceedings of Findings of EMNLP*, 2020.

F. Barbieri, L. E. Anke, and J. Camacho-Collados. Xlm-t: Multilingual language models in twitter for sentiment analysis and beyond, 2021. URL https://arxiv.org/abs/2104.12250.

M. Birjali, M. Kasri, and A. beni hssane. A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems*, 226:107134, 05 2021. doi: 10.1016/j.knosys.2021.107134.

A. Chriqui and I. Yahav. Hebert amp; hebemo: a hebrew bert model and a tool for polarity analysis and emotion recognition, 2021. URL https://arxiv.org/abs/2102.01909.

J. Chun. Sentimentarcs: A novel method for self-supervised sentiment analysis of time series shows sota transformers can struggle finding narrative arcs, 2021. URL https://arxiv.org/abs/2110.09454.

N. C. Dang, M. N. Moreno-García, and F. De la Prieta. Sentiment analysis based on deep learning: A comparative study. *Electronics*, 9(3), 2020. ISSN 2079-9292. doi: 10.3390/electronics9030483. URL https://www.mdpi.com/2079-9292/9/3/483.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

FT. Gamestop's wild ride: how reddit traders sparked a 'short squeeze'. 2022. URL https://www.ft.com/content/47e3eaad-e087-4250-97fd-e428bac4b5e9.

K. K. S. S. M. P. P. N. Gite S, Khatavkar H. Explainable stock prices prediction from financial news articles using sentiment analysis. peerj computer science 7:e340, 2021. URL https://doi.org/10.7717/peerj-cs.340.

H. Gui. Stock prediction based on social media data via sentiment analysis, 2019.

I. Gupta, T. K. Madan, S. Singh, and A. K. Singh. Hisa-smfm: Historical and sentiment analysis based stock market forecasting model, 2022. URL https://arxiv.org/abs/2203.08143.

T. Jacobsen and T. F. Pedersen. Wallstreetbets on wall street - an empirical analysis of the market power of wallstreetbets, 2021.

P. J. R. Kalyani Joshi, Prof. Bharathi H. N. Stock trend prediction using news sentiment analysis, 2016. URL https://arxiv.org/ftp/arxiv/papers/1607/1607.01958.pdf.

Y. Konchitchki and P. N. Patatoukas. Taking the Pulse of the Real Economy Using Financial Statement Analysis: Implications for Macro Forecasting and Stock Valuation. *The Accounting Review*, 89(2):669–694, 10 2013. ISSN 0001-4826. doi: 10.2308/accr-50632. URL https://doi.org/10.2308/accr-50632.

B. G. Malkiel. *Efficient Market Hypothesis*, pages 127–134. Palgrave Macmillan UK, London, 1989. ISBN 978-1-349-20213-3. doi: 10.1007/978-1-349-20213-3_13. URL https://doi.org/10.1007/978-1-349-20213-3_13.

L. Marston. *Introductory statistics for health and nursing using SPSS*. Sage Publications, 2010.

M. L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282, 2012.

S. Mehtab, J. Sen, and S. Dasgupta. Robust analysis of stock price time series using CNN and LSTM-based deep learning models. In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, nov 2020. doi: 10.1109/iceca49313.2020.9297652. URL https://doi.org/10.1109%2Ficeca49313.2020.9297652.

A. Mittal and A. Goel. Stock prediction using twitter sentiment analysis.

L. Nemes and A. Kiss. Prediction of stock values changes using sentiment analysis of stock news headlines. *Journal of Information and Telecommunication*, 5(3):375–394, 2021. doi: 10.1080/24751839.2021.1874252. URL `https://doi.org/10.1080/24751839.2021.1874252`.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

J. M. Pérez, J. C. Giudici, and F. Luque. pysentimiento: A python toolkit for sentiment analysis and socialnlp tasks, 2021. URL `https://arxiv.org/abs/2106.09462`.

E. Razova, S. Vychegzhanin, and E. Kotelnikov. Does bert look at sentiment lexicon?, 2021. URL `https://arxiv.org/abs/2111.10100`.

S. Smidt. A new look at the random walk hypothesis. *Journal of Financial and Quantitative Analysis*, 3(3):235–261, 1968. doi: 10.2307/2329812.

P. Sonkiya, V. Bajpai, and A. Bansal. Stock price prediction using bert and gan, 2021. URL `https://arxiv.org/abs/2107.09055`.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

C. Wang and B. Luo. Predicting $GME stock price movement using sentiment from Reddit r/wallstreetbets. In *Proceedings of the Third Workshop on Financial Technology and Natural Language Processing*, pages 22–30, Online, 19 Aug. 2021. -. URL `https://aclanthology.org/2021.finnlp-1.4`.

S.-C. Wang. *Artificial Neural Network*, pages 81–100. Springer US, Boston, MA, 2003. ISBN 978-1-4615-0377-4. doi: 10.1007/978-1-4615-0377-4_5. URL `https://doi.org/10.1007/978-1-4615-0377-4_5`.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-Art Natural Language Processing, Oct. 2020. URL `https://doi.org/10.5281/zenodo.6554885`. If you use this software, please cite it using these metadata.

Statutory declaration

I hereby certify that I have written this thesis on my own and have not used any sources other than those indicated in the list of sources. All passages taken verbatim or in spirit from published or unpublished sources are identified as such. This also applies to sources which I myself have prepared for other purposes. The drawings or illustrations in this work have been prepared by myself or have been provided with an appropriate source reference. This work has not been submitted in the same or similar form to any other examination or examination authority. I am aware that if the above declaration proves to be incorrect, this constitutes an attempted de-registration which may result in expulsion.

Darmstadt, August 01, 2022

[illegible]

Archiving declaration

I agree to the archiving of a printed version of the thesis in the library. It may be viewed there by third parties.

Darmstadt, August 01, 2022

[illegible]