Stock Dashboard

Created by Andris Jekabs Vaivods

Supervisor: Dr Jon Chamberlain

# Acknowledgements

Throughout the creation of this project, I have received a great deal of support and assistance. I would first like to thank my supervisor, Dr Jon Chamberlain, whose expertise was invaluable in helping me choose the right direction to take and successfully complete my project. Jon has been incredibly supportive and helpful throughout the entire process.

I want to acknowledge my university colleagues Jakub, Emilija, Mason, Sean, and Viktoria for their excellent feedback and ideas on the website and poster. Thank you for helping me create the best version of the website and making the whole creation process more exciting and engaging.

In addition, I would like to thank my partner Wiktoria for her excellent feedback and sympathetic ear. You are always there for me and provided happy distractions to rest my mind outside of my research.

# Abstract

The Stock Dashboard is a website where users can search for specific stocks and view live data, such as their article sentiment, analyst target prices, future stock predictions and article topics. The website’s goal is to help the user make future investing decisions. The data analysis is performed in the back-end Python pipeline and stored on the MongoDB cloud database.

The article data is web scraped from various financial news publishers using Requests and Yahoo Finance API. The data is then cleaned using Beautiful Soup and Pandas. After performing more robust data pre-processing like lemmatisation, topics are extracted from the text using Gensim and sentiment is analysed using a lexicon-based approach with the VADER sentiment analyser. Stock price predictions are made using historical prices and features extracted from historical prices like Moving Averages. The data is passed through a Long short-term memory (LSTM) neural network, after which predictions are made for the next ten days. The Python code is tested using unit testing. All the data is stored on MongoDB Atlas and is updated daily.

The website consists of a search page, dashboard and stock screener. The user can search for the desired stock; the dashboard contains all the data visualisations made using ChartJS. A NodeJS server is used in conjunction with ExpressJS to access the data. In the stock screener, the user can filter by various metrics, like price or market cap, to search for stocks, which can help make future investing decisions.

Contents

[1 Acknowledgements 2](#_Toc102118124)

[2 Abstract 3](#_Toc102118125)

[4 Literature Review 6](#_Toc102118126)

[5 Motivation 9](#_Toc102118127)

[5.1 Passion for investing and data analysis 9](#_Toc102118128)

[5.2 Alternative to expensive software 9](#_Toc102118129)

[5.3 The dashboard form is a great fit for the average investor 9](#_Toc102118130)

[5.4 Developing the right skills 10](#_Toc102118131)

[5.5 Inspiration from existing websites 10](#_Toc102118132)

[6 Aims and Objectives 11](#_Toc102118133)

[6.1 Summer Preparation and challenge week 11](#_Toc102118134)

[6.2 Minimum Viable Product 12](#_Toc102118135)

[6.3 Final Product 14](#_Toc102118136)

[6.4 Challenges 15](#_Toc102118137)

[7 Technical Documentation 17](#_Toc102118138)

[7.1 User Interaction and Design 17](#_Toc102118139)

[7.1.1 Dashboard 17](#_Toc102118140)

[7.1.2 Stock Screener 19](#_Toc102118141)

[7.2 Source Code Summary 20](#_Toc102118142)

[7.2.1 Web Scraping and Data Gathering 21](#_Toc102118143)

[7.2.2 Data Cleaning 21](#_Toc102118144)

[7.2.3 Sentiment Analysis 22](#_Toc102118145)

[7.2.4 Topic Extraction 24](#_Toc102118146)

[7.2.5 Predictive Modelling 25](#_Toc102118147)

[7.2.6 Database 26](#_Toc102118148)

[7.3 GDPR and data protection 28](#_Toc102118149)

[8 Quality Assurance 29](#_Toc102118150)

[8.1 Test Strategy 29](#_Toc102118151)

[8.2 Unit Testing 29](#_Toc102118152)

[8.3 User Testing 30](#_Toc102118153)

[9 Project Planning 31](#_Toc102118154)

[9.1 Jira 31](#_Toc102118155)

[9.2 Gitlab 32](#_Toc102118156)

[9.3 Gantt Roadmap 33](#_Toc102118157)

[10 Conclusion 35](#_Toc102118158)

[10.1 Summary 35](#_Toc102118159)

[10.2 Future Work 36](#_Toc102118160)

[11 References 37](#_Toc102118161)

[12 Table Of Figures 41](#_Toc102118162)

# Literature Review

In the Literature Review, I will summarise and synthesise research papers and books related to sentiment analysis, machine learning, as well as dashboard design and investing. The goal of this is to gain a better understating of these fields, discover what technologies are best for my use case, propose recommendations for further research and form conclusions.

**Bonta, V. a. J. N., 2019. A comprehensive study on lexicon-based approaches for sentiment analysis. Asian Journal of Computer Science and Technology, S2(8), pp. 1-6.**

Bonta [1] compares the sentiment analysis accuracy of Python’s three most popular sentiment analysis tools – Natural language processing toolkit (NLTK), Text Blob and VADER. The text analysed is movie reviews from rottentomatoes.com. Bonta and Janardhan (2019) concluded that VADER is the “gold standard” list of lexical features which are especially attuned to finding semantics in microblog texts. The accuracy of VADER was 77%, compared to 74% and 62% for Textblob and NLTK, respectively. Accuracy is the number of correctly predicted data points out of all the data points. Also, VADER follows grammatical and syntactical conventions for expressing and emphasising sentiment intensity, for example, “that movie was good” compared to “that movie was GOOD” or “that movie was GOOD!!!”.

This paper proves that a VADER lexicon-based approach is a good way of identifying sentiment in movie reviews; however, it is unknown whether the same can be said for financial news analysis.

**Robert P. Schumaker, Y. Z. C.-N. H. H. C., 2012. Evaluating sentiment in financial news articles. *Decision Support Systems,* 53(3), pp. 458-464.**

Schumaker (2012) investigated whether the sentiment and tone of financial news articles correlate to measurable stock price movements. The author found that subjective news articles were easier to predict in price direction (59.0% versus 50.0% of chance alone), and using a simple trading engine; subjective articles garnered a 3.30% return. Investigating the correlation further, they found that their system could predict price decreases in articles with a positive sentiment 53.5% of the time and price increases in articles with negative sentiment of 52.4%. This can be an indicator that traders behave contrarily. For example, see good news, sell; see bad news, buy.

From this, we can conclude that sentiment analysis of financial news using a lexicon-based approach can be used to make a return in the stock market.

**A. Porshnev, I. R. a. A. S., 2013. Machine Learning in Prediction of Stock Market Indicators Based on Historical Data and Data from Twitter Sentiment Analysis. 2013 IEEE 13th International Conference on Data Mining Workshops, Volume 13, pp. 440-444.**

Porshnev [2] discussed and tested the possibility of using Twitter users’ moods and psychological states of people to improve stock returns. They used a lexicon-based approach to analyse psychological states and analysed 755 million tweets. The author concluded that the addition of information did not allow for a significant increase in the model’s accuracy. The best average accuracy rate of 64.10% was achieved using a Support Vector Machine algorithm to predict the DJIA indicator, slightly better than a Neural Network.

From Schumaker, we can conclude that negative and positive sentiments can be used to increase returns in the stock market, so this could be a valuable addition to the dashboard. However, there is a gap in the research regarding using the VADER sentiment analyser specifically, which outperforms other lexicon-based approaches [1] in analysing financial news articles. Porshnev [2] stated that Twitter sentiment does not increase stock returns. However, it is known that a simple trading strategy in conjunction with a lexicon-based approach to financial news articles’ sentiment can provide positive returns in the stock market. [3]

**Few, S., 2006. Information Dashboard Design: The Effective Visual Communication of Data. 2nd ed. Sebastopol, CA: O’Reilly.**

Information Dashboard Design: The Effective Visual Communication of Data describes what a dashboard should look like and what makes a dashboard good or bad. For example, a good dashboard uses a mix of graphics and text to help accomplish a specific objective. However, a bad one has irrelevant information, perhaps is too cluttered and might leave the user even more confused than before. The book is a staple in dashboard design and is the basis on which I designed my dashboard.

Stephen Fews’s book is an excellent resource for knowing how to layout a dashboard. However, what information should be included when it comes to financial dashboards? Even though there is not one single metric that an investor can use to make a profit in the long term, there are some metrics that an investor should look at when deciding on investing in a stock.

**Damodaran, A., 2011. The Little Book of Valuation: How to Value a Company, Pick a Stock and Profit. 2nd ed. N.J.: Wiley.**

Damodaran’s (2011) [4] The Little Book of Valuation: How to Value a Company, Pick a Stock and Profit explains the techniques for making better investment decisions when reviewing stock research reports and engaging in independent efforts to value and pick stocks. Damodaran (2011) compares two ways of valuing a stock – Intrinsic and relative value. The intrinsic value is calculated via a discounted cash flow (DCF). A business’s relative value is determined by comparing how the market values the asset in question with similar assets, i.e., one oil company compared to another, based on its P/E, P/B, ROE, margins etc. Relative value metrics like these would also be helpful to the dashboard since they can be easily accessed and do not require future cash flow predictions.

To conclude, a dashboard containing relative value metrics like P/E, ROE, Net margin and others, as well as containing information like future stock price predictions using an LSTM neural network and historical public sentiment analysed using a lexicon approach, would, for most investors, be enough to make a future investing decision and potentially make a profit in the market. However, to accomplish this task, just having data is not enough. As stated by Few (2006), the dashboard must also be data-driven, highly visual and action-oriented. [5]

# Motivation

This section will talk about the motivation behind the project and answer why I decided to create a dashboard for stocks.

## Passion for investing and data analysis

I decided to make a stock dashboard because I am passionate about investing and love analysing and visualising data, especially financial data. I have been following the stock market movements for over four years now, and in the last few years, I have gained enough income that I can start investing myself. I think fundamental stock analysis using historical information is a great way to find inconsistencies in the market. [4]

During my first year studying at the University of Essex, I worked on a team project where we had to predict the house prices in Boston. This project started my journey into data analysis and predictive modelling. Since then, I have worked on various projects requiring data analysis and visualisation and have always found them enjoyable.

So for my Capstone project, I decided to combine my passion for investing and data analysis into one.

## Alternative to expensive software

A lot of stock analysis software is overly expensive or requires a costly monthly subscription; for example, the Bloomberg Terminal can cost 2000$ per month [6] or the FactSet terminal, which costs 12 000$ per year [7]. This is too expensive for the average investor like me and many others. Nevertheless, is it not unfair that wealthy investors can use tools that we cannot?

Creating my stock analysis tool is a great way to save some money and improve my financial knowledge. Of course, my program would never be as advanced as the Bloomberg terminal, but after investing regularly for over two years, I understand that having too much data can also be a burden. A dashboard is an excellent tool since it does not contain too much information, but it can still provide plenty to make an investing decision.

## The dashboard form is a great fit for the average investor

The definition of a dashboard is a graphical summary of various pieces of important information, typically used to give an overview of a business

I chose to have the page in a dashboard form, having all information on a single page that does not require scrolling. In my opinion, it is a useful way of conveying information to both new and experienced investors. The average investor cannot spend hours analysing stocks, so by using a dashboard, the investor can quickly see the most valuable information without the need to browse through multiple pages and websites.

## Developing the right skills

My goal after graduation is to work as a Data analyst or Python Developer. A project requiring data analysis and python programming is a great fit for me. Data visualisation is a big part of being a data analyst, however, if I just used Python for the data visualisation, this would require the user to have Python installed, and in most cases, the visualisations could not be interacted with. Learning another visualisation tool like ChartJS can help me create beautiful graphics that anyone with an internet connection can see [8] and allow me to develop my front-end development skills further.

Natural Language Processing is also a large part of my project regarding Sentiment Analysis and Topic Extraction. As Porshnev (2013) states that sentiment can be used as an indicator to make returns in the stock market, I thought I would put my learned skills during university from modules like Natural Language Processing and Information Retrieval to the test as well as improve those skills further.

Instead of using a Relational Database Management System (RDBMS) like MySQL, I thought it would be a good idea to learn NoSQL since, in the financial industry, many companies use NoSQL instead of SQL. It is excellent for storing financial data like stock price, which in MySQL would require a considerable collection that contains tens of thousands of rows [9].

## Inspiration from existing websites

I have used the online website Simply Wall Street to help me make investment decisions throughout the last two years. Simply wall street is a stock and sector analysis tool that allows simplified stock analysis. It condenses large amounts of data from various stocks to give the user an overview of the current market [10]. I found this tool very helpful but wanted to simplify it by only keeping the essential information and adding additional features like price prediction and sentiment analysis.

# Aims and Objectives

The website’s main aim was to help the user make future investment decisions by providing accurate and helpful information related to the stock and allowing the user to compare different stocks. From this, I conclude that the website must be easy to use, useful for both new investors and experienced investors, and contain important information in deciding whether a stock is a good buy.

On the other hand, the back-end Python code must be detailed and documented, reliable and efficient as well as scalable. Detailed and documented because this can help give context to future readers and allow for the code to be reused in future projects. [11] Well documented code will allow me not to lose momentum when taking a break to work on projects for other modules [11].

The code should be reliable and efficient as I was planning to add hundreds of stocks for the user to view and automate it such that the code runs every morning to update the most relevant information. Any minor improvements to the code would be amplified by the number of stocks the code has to be run for.

## Summer Preparation and challenge week

During the Summer preparation and challenge week, my main objectives were to research what tools I could use for each section of the website and the back-end code and create the layout of the dashboard.

I also researched journal articles on data analysis, sentiment analysis, and price prediction during the summer. I found out that a lexicon-based approach is a viable option in analysing sentiment, as well as that a Long Short Term Memory neural network is a good way of predicting future stock prices [2].

During this time, I also researched multiple Python libraries that I could use to perform the tasks, like Natural Language Toolkit (NLTK) and TensorFlow. Luckily, I already had some experience with NumPy and Pandas, which I used for data cleaning and analysis throughout the project.

My first objective was to find multiple financial news websites for web scraping, which I could use to get the financial data and articles. I found a few that I could use, like FINVIZ, Yahoo Finance and MarketWatch. In the end decided to use FINVIZ, as it contains information about thousands of stocks, articles relevant to the stock (even though later the articles did require further filtering), and many relevant metrics for specific stocks, like Price to Earnings ratios and Dividends. It also contains many metrics that are not as popular but could still be crucial in deciding whether a stock is a worthwhile investment. [12] I decided to also use it together with Yahoo Finance, a free-to-e Application Programming Interface (API), to access information like the stock price more quickly than if I was using FINVIZ [13].

In terms of the front-end, I was unsure what tools I would use except that I would like to use ChartJS because the graphs looked very good and could be interacted with [8]. Since I wanted to keep the front-end quite simple because the back-end was my priority, I decided to use basic HTML, CSS and JavaScript, which would allow me to spend more time on the back-end - improving the code and making better predictions.

However, soon I figured out that I could not access the database from the website with these tools. For this, I needed a back-end server, which I decided to make with NodeJS. Compared to other back-end runtime environments, NodeJS is well documented, has plenty of online learning resources and does not require learning a new language as it can be accessed through JavaScript [14].

Another positive of using NodeJS is that the server can act as an API. Users can send GET requests to access some of the data on the database and use it for their visualisations and predictions.

## Minimum Viable Product

A minimum viable product (MVP) is a version of a product with barely enough features to be usable by early customers who can then provide feedback for future product development. [15] The aims for the Minimal Viable product were as follows:

**Create a sample website that uses the layout made during Challenge Week.** I knew that I would not be able to get all of the back-end functionality done by the MVP, but having at least a dashboard layout would let me know what data I should be focusing on in the back-end.

**Set up the MongoDB Database**. Before getting the data on the website, I first had to store it in the database; I needed to set up the MongoDB database early on. For the database, I decided to use the MongoDB Atlas Cloud because it offers 500MB of free storage and allows for data analysis and querying through their MongoDB Compass tool [16].

**Create the pipeline for newly added stocks.** Since some of the code is different for whether a stock is new to the database or if it is already there, the code requires two parts. The code requires two parts because if a stock is already in the database, there is no point in searching for old articles as those will already be added. If one were already made, there would be no point in creating another stock prediction model. Since I cannot update an existing stock without adding it as a new one, the code for adding a new stock was implemented first.

**Web scrape information about the stock such as articles and fundamental information from the biggest publishers.** My goal was to have the web scraper extract text from the most prominent publishers like Motley fool and Bloomberg. I knew that it would be too time-consuming to have the web scraper already work with all publishers in the MVP, as this would require hours of tedious work of manually finding where the text is stored on the website (The name of the class or ID) and would add only a few more articles to analyse.

**Clean the articles so they are ready for sentiment analysis and topic extraction.** Even though I was not planning on doing Topic Extraction for the MVP, I cleaned the text so it could be used for both. Topic extraction requires more data cleaning than sentiment analysis since VADER considers punctuation and word casing in its results [17]

**Analyse the sentiment of the articles using a lexicon approach.** During the challenge week and summer preparation, I realised that a lexicon-based approach would be the best way to analyse financial articles on a large scale without powerful hardware [1]. Thus, I wrote a sentiment analyser using the VADER library, but further improvements were needed to improve the accuracy of the analyser, for example, tokenisation. Tokenisation is a way of separating a piece of text into smaller units called tokens [18].

**Create the landing page/search page.** Before adding this, the only way to open the dashboard was to enter the specific stock in the URL box. The addition of the search page added the option for me to test the website with more users and allowed for error testing if the user searches for a stock that does not exist.

**Predict the stock price using the historical price in a Neural Network.** During the Summer and Challenge week, I researched what tools I could use to predict future prices, and I decided on an LSTM model. The goal was to use the stock’s historical price as an input to predict the future stock price for the next two weeks. The MVP version of the model used the prices in the last 50 days to predict the price for the next day, and it had not yet been tested for overfitting or underfitting. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the model’s performance on new data. Underfitting is the opposite of overfitting, where a data model cannot accurately capture the relationship between the input and output variables [19].

The plan was to also create the stock screener for the MVP. However, I decided not to create it for the MVP as the dashboard is the central part of the website, as well as it gave me more time to focus on improving the sentiment analyser and web scraper.

## Final Product

In addition to the features in the MVP, the objectives for the final product were to:

**Only analyse the sentiment of important articles - articles that mention the stock multiple times.** In the MVP, the sentiment analyser analyses all the web scraped articles from FINVIZ. However, not all of those articles are actually about the stock; for example, the article “Why Rivian Stock Jumped Today” by Motley Fool was marked as an article for Tesla by FINVIZ, even though Tesla is only mentioned once. To fix this issue, I added code for identifying whether an article is about the stock. For the article to be marked as important, the stock must be mentioned in the title or mentioned in the article at least three times. If there are other stocks mentioned in the article, they must be In the top 3 in the number of times mentioned, thus making sure that an article can be about three stocks at maximum.

**Extract topics from the articles.** The data was already cleaned and ready for topic extraction. I researched what Python tools I could use to extract topics, and I decided to use a Latent Dirichlet Allocation (LDA) model, which is available with the GENSIM library. The frequently mentioned words for all stocks that do not add value are also removed to extract better topics.

**Create the pipeline to update stock already in the database.** The existing code for adding a news stock is modified to update the stocks already in the database. For example, the code for getting the articles for an existing stock only scrapes the articles from today, as articles from before will already be in the database. Also, since the prediction model will already be made, there is no need for creating a new prediction model, thus saving much time.

**Create a stock screener.** The stock screener on the website allows the user to search for and filter stocks by specific metrics. Compared to the dashboard, where the user can view information about one stock, the screener is a way of comparing stocks to each other, which can help find inconsistencies between stocks. For example, if stock A and stock B are both in the energy sector, both have a market cap of fifty million, but stock A has a P/E ratio of 20 while stock B has a P/E of 50, stock A could be undervalued [20].

**Improve stock price predictor.** The stock predictor for the MVP did not use Moving Averages to predict future prices. Adding another feature to predict prices lowered the error in predictions. However, there still was the problem of the model overfitting the data. To fix this, I added Dropout layers in between LSTM layers.

## Challenges

During the early stages of the project, the biggest challenge was learning all the required new tools for making the website, such as ChartJS, which unfortunately does not have very comprehensive documentation or NodeJS and ExpressJS, which I had not used before the project. I took online courses on LinkedIn learning and watched YouTube video tutorials to overcome this problem. However, this was time-consuming and resulted in me spending less time doing the project itself.

As well as being a good way of determining the public opinion of the stock, I also wanted to use the results of sentiment analysis to predict the future stock price. This turned out a lot harder than I expected, and in the end, I decided against it since, for most stocks, I did not have enough sentiment data to make it a valuable metric for prediction. Secondly, the stocks with enough data to be included in the prediction resulted in a worse predictive model than just using the historical price and moving averages. Thus, I decided to focus more on other parts of the code and improve the stock screener.

The main challenge for topic extraction was how to make sure that the topics extracted were valuable to the investor. For example, topics (taken from the actual code results) such as “5G Network” or “TikTok App” help gain a summary of what the company’s articles are about and could be used for further research if necessary. However, topics like “Market Year” and “Fund Inc” do not add anything of note and cannot help with further research or making an investment decision. (all examples taken from Apple topic extraction results). Topics like “Market Year” do not add value since users cannot get any insight into the articles. I tried to fix this problem in two ways.

Firstly, I removed frequently mentioned words, like “stock” and “market”. This fixed the problem most of the time but also removed some critical information and potential topics. For example, during the invasion of Ukraine, there were many articles on Oil stocks that mentioned Russia. With this system in place, the word “Russia” was removed from the texts and not counted because it was mentioned often. This resulted in relevant information being removed. This means I needed a different way of identifying which words to remove besides stop words.

So, for the final product, I decided to use a custom list of words that should not be used for topics. This included words like” market”, “stock", “inc.”, “fund” and others. I made this list using the previously implemented system and manual analysis of the topics generated. Even though this was time-consuming, it has drastically improved the usefulness of the topics.

# Technical Documentation

This section will talk about back-end Python code and the design of the front-end website. Compared to the [README](https://cseegit.essex.ac.uk/ce301_21-22/CE301_vaivods_andris_j/-/blob/master/README.md) on GitLab, which talks about specific functions in the code and describes what they do, this document will discuss the code more broadly, addressing the question of why I used these technologies and the challenges faced in doing so. More pictures and in-depth detail about the features of the website are available in the [README](https://cseegit.essex.ac.uk/ce301_21-22/CE301_vaivods_andris_j/-/blob/master/README.md).

## User Interaction and Design

The user interacts with the website through the three pages: the search page, the dashboard and the stock screener. The user can navigate the website through the navigation bar at the top.

The website's design is modern and straightforward and uses a dark background with more varied and brighter colours used in the charts. It was important for the information on the website to be colour coded in terms of its rating: green for positive, yellow for neutral and red for negative. These colours are used throughout the dashboard to portray better whether a specific metric should be considered good or bad. However, there is some leeway on whether specific metrics are positive or negative, such as volatility, which could be considered suitable for short term trades. This is because a short-term trader wants to make money in as little time as possible. Thus, they are looking for large swings in the stock price, and high volatility indicates this (however, past volatility is not always indicative of future volatility). The definitive version of the website is skewed toward more traditional stock investing, which considers volatility as a negative metric. It also considers high P/E ratios unfavourable; thus, it is more skewed towards value than growth investing.

### Dashboard

Stephen Few [5] defines a dashboard as a visual display of the most information needed to achieve one or more objectives that fit entirely on a single computer screen so it can be monitored at a glance.

This section will talk about how I created the dashboard and what data I decided to include in it.

**The information in a dashboard is presented visually as a combination of text and graphics.** “Dashboards are highly graphical not because it is cute but because graphical representation can often communicate with greater efficiency and richer meaning than text alone.” (Few, 2006, p.12). With this in mind, I gave myself the task of having as little text as possible and trying to convey information through graphical forms instead of text. Python does support dashboard graphical design and visualisation. However, ChartJS, in my opinion, is a better tool as it allows for detailed analysis of graphs and quick rendering speeds for anyone with a modern browser. It would also allow me to create beautiful and eye-catching visualisations, which investing websites like FINVIZ or MarketWatch do not have.

Most of the dashboard information is in graphical form. However, sections like the latest articles and latest analyst ratings have to be presented textually. The summary section and tooltips are also in textual form. In the summary section, the user can view a summary of all the information on the website, which can be especially useful to new investors who are not familiar with what each metric means.

**Dashboards display the information needed to achieve specific objectives.** To create a functional dashboard, it was necessary first to figure out the objective, which in my case was to inform the user about a stock’s health and investing potential. However, now I must decide what information would allow users to improve their understanding of the stock and decide their next step. The financial characteristics of a successful company are still up for debate. However, they are known to be stable earnings. As Payne (2011) stated, the measurement of Value Line Earnings Predictability is the reliability of earnings forecasts and Return on Equity (ROE) and comparing the specific company to the broader market or industry averages. As Porshnev (2013) discovered, sentiment can be used to make market returns. Thus it would be helpful to include the historical sentiment of articles and moving averages of this data to see long-term trends [21].

**Dashboards can be monitored at a glance.** For a dashboard to be useful, it should only take a few minutes to understand its information fully. To ensure this is the case, I included information that would be easy to understand and added short tooltips for users who might not know what a specific metric is. Colour coding information also makes it easier to understand it quickly and efficiently. Having a summary section that condenses the dashboard into a textual form would make monitoring easier.

**Graphical user interface

Description automatically generated**

Figure 1 Dashboard of Amazon (AMZN)

With all of these objectives accomplished, the page (as Seen in Figure 1) can be defined as a dashboard. Through research and a mix of graphics and text, I have made sure that the information in the dashboard is relevant and easy to understand.

### Stock Screener

The stock screener consists of a form where the user can write various values and pick metrics to filter by, for example, the stock price or the P/E ratio. After filtering the stocks, the user can view the dashboards of the filtered stocks by clicking on the tickers. In the filter results, users can view a summary of the stock's metrics and sort by values by clicking on the header.

A stock screener is an excellent tool for more advanced users looking for potential investments in companies with specific metrics. For example, as a value investor, my goal is to invest in cheap companies compared to the market, so I would filter by companies with a low P/E ratio, like under 20.

A screenshot of a computer

Description automatically generated with low confidence

Figure 2 Stock Screener

As seen in Figure 2, in total, there are twelve different metrics to filter by:

P/E, Dividend %, Earnings Per Share (EPS), Volatility, Share Price, Insider Own %, Relative strength index (RSI), Return on Equity (ROE), 7-day Article Sentiment, Analyst Target Price / Share price ratio, P/B ratio and Return on investment (ROI).

The user can pick if they want to filter by below, above, or precisely the value for all metrics. There are also three pre-sets – Value, Growth, or Oversold, which can be used to discover specific types of investments. The value pre-set will filter stock with low P/E and P/B ratios. The growth pre-set, on the other hand, will filter stocks with a high P/E and an analyst expected price increase of more than 20%. The Oversold pre-set will filter companies that have a low RSI, high volatility, and a low 7-day sentiment. However, the pre-sets can be further customised to the user’s goal.

If the user is unfamiliar with any of the metrics, they can highlight it and read a short description , the same as for metrics in the dashboard.

## Source Code Summary

This section will go over each part of the code, explain why I used the technologies I did, how I used them, their efficiency, and the challenges faced.

The code can generally be divided into six sections (not including testing, which I will discuss later in the report): Web scraping, Data cleaning, Sentiment analysis, Topic extraction, Predictive modelling, and database.

### Web Scraping and Data Gathering

In theory, web scraping gathers data through any means other than a program interacting with an API (or, obviously, through a human using a web browser). This is most commonly accomplished by writing an automated program that queries a web server, requests data (usually in HTML and other files that compose web pages), and parsing that data to extract needed information. [22]

The primary source for web scraping information for my project is FINVIZ.com. However, the articles are web scraped from various news publishers like Motley Fool and Bloomberg. I used the Yahoo Finance API to get the historical price data and analyst ratings because it is reliable and has low latency; it is also well documented for Python. The Yahoo Finance API is a range of libraries/APIs/methods to obtain historical and real-time data for various financial markets and products, as shown on Yahoo Finance. [13]

The first step of getting the data was to get the HTML content of the website. I used Requests, an HTTP request library, and BeautifulSoup for parsing the extracted HTML. The fundamental metrics for stocks and HTTP links to the articles are extracted from the FINVIZ page. After this, each link to the article is opened and parsed using Requests and Beautiful Soup. The text and title of each article are extracted. This was the most challenging part of web scrapping since each publisher has a different class name where the article text is stored. Both the metrics and articles are stored in Pandas data frames. The article’s data contains the title, publisher, time of publishing, link and text.

To acquire the analyst ratings used in the latest analyst ratings and historic analyst rating sections of the dashboard and the historical price, which is used in the price prediction and historic price section, I used Yahoo Finance API.

### Data Cleaning

Data quality remains a significant concern in various fields, and dirty data can lead to incorrect decisions and unreliable analysis. Examples of common errors include missing values, typos, mixed formats, replicated entries of the same real-world entity, and violations of business rules. [23]

Data cleaning prepares data for analysis by removing or modifying incorrect, incomplete, irrelevant, duplicated, or improperly formatted data [23].

Thus, the text must first be cleaned to get accurate results from sentiment analysis and topic extraction from the articles.

To clean up the text for sentiment analysis, I removed any unnecessary white space between paragraphs and removed characters that do not affect the sentiment, like the symbols @ and %, parenthesis, and others. Now the text is ready for sentiment analysis.

More robust data pre-processing must be done to prepare the text for topic extraction. Firstly, stop words are removed from the text. Stop words are English words that do not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. For example, the words like the, he, have [1].

Secondly, punctuation, such as dots and exclamation marks, are removed not to generate topics based on punctuation, as those would not be valuable to the user. This is done using the string module.

Thirdly all the words are lemmatised, even though stemming would be the faster option. For grammatical reasons, documents will use different word forms, such as organise, organised, and organising. Additionally, there are derivationally related words with similar meanings, such as democracy, democratic, and democratisation. [24]

Stemming and lemmatisation aim to reduce inflectional and related word forms to a common base form. Stemming refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time and often includes the removal of derivational affixes. [2]

Lemmatization usually refers to doing things properly using a vocabulary and morphological analysis of words, typically aiming to remove inflectional endings only and return the base or dictionary form of a word, known as the lemma. If confronted with the token saw, stemming might return just s. In contrast, lemmatisation attempts to return see or saw depending on whether the use of the token was as a verb or a noun [24].

After lemmatisation on each token (word) and making the text lowercase, significantly improving the expected output’s consistency, the text is ready for topic extraction and sentiment analysis.

### Sentiment Analysis

Sentiment analysis computationally identifies and categorises the opinions expressed in a text to determine whether the writer’s attitude towards a particular topic or product is positive, negative or neutral. [1]

Unlike machine learning algorithms, VADER (Valence Aware Dictionary and sEntiment Reasoner) performs better across various domains. As compared to machine learning techniques, VADER has several advantages. Firstly, it is both quick and computationally economical, which in the case of analysing thousands of articles is highly important. VADER runs directly from a standard modern laptop or computer and does not require powerful hardware; a corpus takes a fraction of a second to analyse with VADER, but it approximately takes hours when using more complex models like Support Vector Machine. The second advantage is that the lexicon and the rules used by the VADER are directly accessible and not hidden. Therefore, VADER is easily understood, extended and modified. [1]

Thus I decided that VADER would be the best fit for my projects, as it is fast, could run on my personal computer and does not require a test and training set, which in the early stages of my project, I could not provide.

VADER relies on a dictionary that maps words and other numerous lexical features common to sentiment expression. For example, “bad” is negative and “happy” is positive. For VADER, the dictionary that the analyser uses must be as fitted for article analysis as it can be.

For the dictionary, I tried using a custom lexicon of negative, neutral and positive words acquired from a GitHub repository for analysing financial news sentiment. Using this custom lexicon resulted in worse results than using the pre-defined lexicon for VADER, which is fitted to be used for social media text analysis. After tinkering, In the end, I decided to use the original dictionary instead of a custom one.

After web scraping and data cleaning, the text is split into sentences because VADER works better on shorter texts [1]. For most publishers, the last three sentences are removed. They are removed because often, at the end of the article, there would be either information about the author, disclosure information, or links to other articles. For example, many Motley Fool articles end with this sentence: “This article represents the opinion of the writer, who may disagree with the “official” recommendation position of a Motley Fool premium advisory service.”

Each sentence is given a weight based on its length, and the title is given the highest weight of 25% of the final rating. After this, the title and each sentence are given a positive, neutral, and negative score, making the compound score from -1 for very negative to 1 for very positive. Together, these create the article compound score used in the dashboard visualisations and the latest article section.

### Topic Extraction

Topic extraction is a machine learning technique that organises and understands extensive collections of text data by assigning “tags” or categories according to each text’s topic or theme. Topic tagging is particularly useful for analysing vast amounts of text data in a fast and cost-effective way [25].

Topic analysis can be applied to different levels of scope: document level, sentence level and subsentence level. The topics are extracted on the document level in the website’s case, specifically from articles in the last 14 days.

The two main topic extraction models are the Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). Latent Semantic Analysis is the ‘traditional’ method for topic modelling. It is based on a principle called the distributional hypothesis: words and expressions that occur in similar pieces of text will have similar meanings. It is based on the word frequencies of the dataset. The general idea is that for every word in each document, you can count the frequency of that word and group together the documents with high frequencies of the same words. [26]

On the other hand, LDA is a bit more complex. In LDA, each document can be described by a distribution of topics, and each topic can be described by a distribution of words. LDA has four main steps:

1. First, each word in each document is randomly assigned to one of the ‘T’ topics.
2. Assume that all topic assignments except for the current one are correct.
3. The proportion of words in the document, ‘d’ that are currently assigned to topic ‘t’ is equal to p(topic t | document d), and the proportion of assignments topic ‘t’ over all documents that belong to word ‘w’ is equal to p(word w | topic t).
4. These two proportions are multiplied and assigned a new topic based on that probability.

The proportion of topics for each document is then determined from these topic assignments. [27]

I decided to use LDA instead of LSA as it is more accurate most of the time as well as it allows for multiple topics to be assigned to one text in the form of percentages, which is traditionally not available for LSA [28].

For topic extraction, I use multiple Python libraries. Firstly, GESIM as it supports an LDA model. Secondly, Pandas for storing the articles and the generated topics.

Firstly, I gathered the articles published in the last 14 days. The next step is to create a term dictionary, where each term is assigned an index. Then the articles are converted into a document term matrix using the term dictionary. Finally, the dictionary term matrix is passed through the Gensim LDA model. The generated topics are saved to a JSON dictionary to be later saved to the database.

### Predictive Modelling

Machine learning algorithms have been used in stock market forecasting for a long time. The most common methods are Neural Networks and Support Vector Machines. Usually, machine learning algorithms are trained on technical data about stock movements, for example, moving averages [2].

In the stock market, the data generated is enormous and is highly non-linear. A model that can analyse the hidden patterns and underlying dynamics is needed to model such dynamic data. Deep learning algorithms can identify and exploit the interactions and patterns in data through a self-learning process.

In this case, I decided to use a non-linear Neural Network over a linear SVM because Neural Networks are generally more accurate but require more training data. However, in this case, this is not a problem since most stocks have plenty of historical price data available [29]. A neural network does require longer to train because it requires more data than an SVM, but this is not problematic since the model has to be trained once for every stock and then it can be reused [29].

The first step is to gather the data. The stock price for the last 2500 days is acquired using the Yahoo Finance API. 2500 might seem like a lot, but 25% of those days will not have price data since there are holidays and weekends. After testing the neural network with 2000, 2500 and 3000 days, 2500 days achieved a good balance between overfitting and underfitting when predicting the future stock price of Amazon. It achieved a Mean Squared error of 0.0028, the lowest of all three models. The Mean Squared Error (MSE) measures how close a fitted line is to data points. [30]

After acquiring the data, 20-day and 50-day moving averages are generated. Now it is split into training and testing sets with a ratio of 80:20. That is, 80% of the dataset goes into the training set, and 20% of the dataset goes into the testing set. When learning a dependence from data, it is important to divide the data into the training and testing sets to avoid overfitting. Gholamy (2018) stated that an 80:20 split is empirically the best division.

After this, the data is scaled or normalised to be between 0 and 1. The price data is scaled because a large spread of values may result in considerable error gradient values causing weight values to change dramatically, making the learning process unstable. [31]

Before the data can be passed through the model, it must be reshaped. I decided to use the last 50 days to predict the future price. The last 50 days are used to predict the next day. The test and training sets are split to accommodate this.

Now that the data pre-processing is done, the data can be passed through the model. I decided to use a Sequential LSMT model since there are many resources on setting up an LSTM model, and Selvin (2017) effectively used LSTM to predict stock price in the short to medium term. The model has four layers with a Dropout layer with a frequency of 0.2 in between each. The Dropout layer randomly sets input units to 0 with a frequency rate at each step during training time, which helps prevent overfitting. In total, predictions are made for the next ten days. For example, when predicting the stock price of Apple stock, the last 50 days are used. After this, the last 49 days plus the one predicted day are used until the price for the next ten days is predicted.

After making the predictions, the data is inversely scaled back to its original form. Now it is saved to a JSON dictionary to be saved to the database. The generated neural network is saved for later use if the stock is new to the database. That is when the stock price is predicted again tomorrow.

### Database

The database is a NoSQL MongoDB database. NoSQL databases (aka “not only SQL”) are non-tabular databases and store data differently than relational tables [9]. NoSQL databases come in various types based on their data model. In this case, they are stored in JavaScript Object Notation (JSON) objects, which are key-value pairs. They provide flexible schemas and scale effortlessly with large amounts of data and high user loads; thus, they are great for storing financial data. [9]

The database consists of 13 collections. The Stocks collection is linked to all other collections as it contains the ticker, for example, “AMZN” for Amazon. As seen in the Entity-Relationship Diagram (Figure 3) from left to right. Each stock has:

* one rating in the Stock\_rating collection
* one text summary in the Text\_summary collection
* many topics in the Topics collection
* many prices and moving averages in the Stock\_price collection
* its fundamental metrics in the Stocks\_info collection
* many latest articles in the Latest\_article collection
* many average sentiments in the Avg\_sentiment collection
* many articles in the Articles, and each article has zero or one in the Latest\_articles collection
* one total analyst rating in the Analyst\_total collection
* many price predictions in the Price\_prediction collection
* many analyst ratings in the Analyst\_ratings collection
* and each analyst rating in the Analyst\_ratings has one or zero ratings in the Latest\_analyst\_ratings

Diagram

Description automatically generated

Figure 3 Entity Relationship Diagram of Database

At the time of writing this (18/04/2022), there were 48 stocks in the database with nine thousand articles split between them.

## GDPR and data protection

The Data Protection Act controls how personal information is used. Everyone responsible for using personal data has to follow strict rules called ‘data protection principles’. This includes making sure that the data is used fairly, used for a specified purpose, and is limited to only the data that is necessary.

The final version of the website complies with the Data Protection act since I am not storing any user data on the website, this is because I’m not storing any session information, there currently is no login system and no way for users to write or publish something to the website.

However, it would be illegal for me to publish the web scraped articles on my own page. In this case that is not a problem since the website does not contain the full article text but only the title and the extracted metadata like sentiment and topics.

# Quality Assurance

This section will talk about the test strategies I used to test the website and back-end, why I used these strategies, and how testing improved the project and its reliability.

## Test Strategy

The main objective of testing the back-end was to make sure that the code was ready to be automated. For example, it would update the database with the newest data every day and that it would run indefinitely without errors (as long as the web scraping sources are still available). The type of testing best suited for this is functional testing and unit testing [32].

The main objective of testing the front-end was to make sure that the user experience was as good as possible. The appropriate testing types are black-box testing and acceptance testing [32].

Black-box testing is a software testing method that examines the functionality of an application without peering into its internal structures or workings. In Acceptance testing, the product is handed over to a few users to test its acceptability [32].

User Testing will be a mix of black-box testing the website and allowing users to test the website and assess whether or not it can be accepted as a tool for helping make investing decisions.

## Unit Testing

Unit testing is a software development process in which minor testable parts of an application, called units, are individually and independently scrutinised for proper operation. [32]

Each unit of code (in this case function) has tests to verify that it is functioning correctly. This includes edge cases and general test cases. Each function is given input, and the output of the function is then verified against the predicted output using the Python unit test library. The advantage of treating each function as a unit is that if one of the units fails, the failure point can be narrowed down to a single function. However, further unit tests could be written to test even smaller parts of the code (as small as even one line of code), making finding the fail point even more straightforward.

The most challenging part of writing unit tests was finding edge cases. An edge case is a situation that occurs only at an extreme (maximum or minimum) operating parameter and often results in unexpected outcomes or even crashes [33]. Finding edge cases takes considerable time and creativity since the goal is to get the application to perform functions it is not intended to do and figure out the results [33]. Despite this, they are worth finding since a 1% chance of crashing can be exaggerated because the code is run for every stock every day. For example, if the chance of an error is only 0.5%, and I run the code 50 times (if I have 50 stocks to update), the actual error rate is 22%[[1]](#footnote-1) , and if the code is run every day, the chance of an error occurring in the space of one week is 82%.

## User Testing

I presented the website to multiple people and gathered their feedback to test the website's functionality. The testing pool had people with various backgrounds; however, most were young adults. Most did not have experience in investing, but since the website's goal was to make sure that beginner investors could also use it, this was fine. In total, I gathered feedback from 10 people. Here are some of the requested features that made it into the final product:

**Moving averages.** The earlier versions of the dashboard did not have moving averages. A moving average is an indicator that shows the average value of a stock’s price over a period [21]. On the website, I added 20-day as well as 50-day moving averages. Moving averages are used for price predictions, but they can also be used to assess the momentum of the stock price.

**Links to articles.** Adding links to the original articles is useful for users looking to do further research into the stock or just looking to read the latest news. The final product’s newest articles section has links to the original articles.

**Search page auto-complete field.** The search page has an autocomplete feature, where the user can see which stocks are available. Before the auto-complete feature, the user could not know what stocks were available on the website.

User Testing has improved the final product by discovering additional features that improve the user experience. Firstly, by giving an unbiased opinion on existing features and secondly, by helping to understand which parts of the website are confusing or what is preventing users from doing what they set out to achieve [32].

# Project Planning

This section will talk about the DevOps tools I used to plan the project. The tools used include Jira for issue tracking, GitLab for version tracking, and Team Gantt to get a holistic view of the project.

Even though this is an individual project, DevOps tools like the ones mentioned are a great way to improve the speed and stability of software development and deployment and allow the project supervisors to see a detailed view of the progress made in the project [34].

## Jira

Jira is a software application used for issue tracking and project management [35]. I used Jira to keep track of tasks that have to be done, bugs that should be fixed, and topics that I should do further research into. For example, tasks like “Add Moving Averages”, “Research how to calculate uncertainty for Price prediction model”, or “Create unit tests for NodeJS server”. For some more extensive tasks, I also had subtasks for improved organisation. For example, the task “Create Stock Screener 1.0” had subtasks like “Give the user the option to filter by specific metrics” and “Display filtered stocks”. I also had different priorities for tasks, such as high, medium, and low. Even though I wanted to do all the tasks, some were more important than others. For example, “Add text to the summary section in Dashboard” is more important than “Add Docstrings to back-end code functions“ because the website would not be fully functional without the summary section, but the back-end is fully functional without the docstrings. However, it would be a good addition. In total, during the project, I did over 80 issues, as seen in Figure 4.

Chart, histogram

Description automatically generated

Figure 4 Cumulative Flow Diagram from Jira

Jira has been extremely helpful in managing the project. Using a Kanban Board with multiple swim-lines made it easier to visualise the workflow. The out-of-the-box reporting tools were also beneficial. They allowed me to see how far I had progressed through the project. Reporting tools like the Cumulative Flow Diagram (Figure 4) and the Average age report were great motivators to keep improving the project and fixing issues or completing backlog tasks that I avoided.

However, there are some improvements to be made in my use of Jira. Firstly, my limited use of User stories. A user story is an informal, general explanation of a software feature written from the end-user’s perspective. Its purpose is to articulate how a software feature will provide value to the customer. In hindsight, I can see how the use of user stories would have improved the final product and made user testing easier. I could have checked if the tester thinks that the user stories are accomplished or if they need future improvements or changes when performing user testing.

To conclude, I have used Jira now in multiple projects, and in each, the software has only improved the outcome, which also applies here. Even though I did not use all the functionality of Jira, like user Stories or Version control, I have used it extensively as a tool for reporting and issue tracking.

## Gitlab

Gitlab is software for tracking changes in any set of files, particularly project files [36]. Usually, Git tools are used in collaborative environments when working on projects. However, they can also be handy when working on a project alone.

Firstly, having an online repository where the project files are saved is a great way to mitigate the risk if something happens to my computer and the files get corrupted. With the project’s latest version being pushed to git, the files are not lost and can be recovered from the git repository.

Secondly, going back to a previous code version can be helpful if the current code version does not work correctly. This came in handy when I modified the dictionary for the VADER sentiment analyser. The dictionary I used at first turned out to be better than the new one, so I reverted the code to a previous version.

However, to revert to the previous version, I must push the current versions often and write comments. This was the case; comments like “Continued coding loop for old tickers, added code to get stock info, added function to delete old data and inserting new data into DB” let me and others (if I was working in a team) know what I have changed to the project files and code.

Issue tracking is another valuable feature of GitLab, though I did not use it since I used Jira to keep track of issues. Issue tracking with Gitlab has the advantage of seeing precisely in which commit the issue was fixed (or the opposite when a new bug was created).

## Gantt Roadmap

The only feature that I think is missing from Jira is a timeline view in the form of a Gantt Chart. A Gantt chart is a bar chart that illustrates a project schedule. I used Team Gantt, which was a great way to see how tasks and subtasks are connected [37]. A timeline view is also helpful when there is a change in the schedule (which can often happen since there are other modules and projects to focus on). If a problem arises, the schedule can be adjusted without worrying about whether the tasks can be finished in time.

Chart, bar chart

Description automatically generated

Figure 5 Gantt Roadmap for January, February and March

As seen in the Gantt Roadmap (Figure 5), I was planning to work on multiple parts of the project simultaneously. The idea is that if I got stuck at some point, I could start working on another part until I researched how the last part could be done.

However, this was not as helpful as I had expected. Trying to do multiple things simultaneously resulted in delays, and I could not follow the Gantt chart Roadmap. Jumping from task to task is slow since it requires some time to get accustomed to the code (mainly because the project is written in multiple languages). Another reason for this delay could be that I had a project for another module in February that I had not accounted for when creating the roadmap. The original roadmap had to be changed multiple times to adjust for this.

To summarise, the roadmap was very helpful in visualising dependencies and approximating how long a task would take. However, the downside is that it is challenging to precisely calculate how long a task would take and that almost in all cases, I am too optimistic in my time approximations. In the future, I would like to continue using Gantt roadmaps like this to visualise the “perfect” scenario when doing a project. Instead of focusing on the deadline and rushing the task, I create a more detailed and reliable product since that is far more important.

# Conclusion

In this section, I will summarise my work and talk about future improvements that could be made to the product.

## Summary

The website’s goal was to have a dashboard that the user could use to make future investing decisions.

I have accomplished this task and added additional features that could further improve the website’s usability, such as the stock screener and tooltips for new investors.

The dashboard contains relevant information for investors, short term and long term, beginners or experienced. The information on the website is colour-coded for ease of use and summarised in textual form to get quick investing insights. The dashboard is visually pleasing and can be quickly monitored, and can be used to discover future research points (such as “Why has the article sentiment dropped in the last seven days?”).

The back-end code is efficient; however, future improvement is needed in the documentation of the code. Each code section does the required job and uses the appropriate tools and libraries. Request and BeautifulSoup for Web scraping stock data. Pandas for data cleaning. Gensim for topic extraction. VADER for sentiment analysis. TensorFlow for price prediction and finally MongoDB for the database. The code uses functional programming techniques for improved productivity and memory efficiency. The code is thoroughly tested through unit testing, including sample use cases and edge test cases.

The generated Sentiment data is accurate over 70% of the time, comparable to research data [1]. The topics extracted are relevant to the stock and can be helpful as points for further research. The stock predictor has a high accuracy for most stocks with medium to low volatility. However, it should not be used as an investing tool on a day-to-day basis but as an indicator of the stock’s momentum. Even though the stock predictor could be used to make money on historical data, past returns do not indicate future returns.

The MongoDB database is fast and efficient and allows the website to have quick loading times. The organisation of the JSON data is memory efficient and easy to understand. Most importantly, the database is scalable, which is crucial as the number of stocks in the database grows, and the number of articles compounds.

I am proud of both parts of the final product, the front-end and the back-end. Even though, in theory, the website is ready for public use and could be launched, there is still room for future improvements in terms of additional features to the website and improvements in the back-end models like the sentiment analyser and stock price predictor.

## Future Work

Even though the final product has come out better than I expected and with additional features, there is still room for future improvements in both the front-end and the back-end.

For the front end, a request form page could be added to the website, which would allow investors to request specific stock to be added to the website. In the current build, I choose a random S&P 500 stock to be added to the database every day, thus steadily increasing the number of available stocks. However, not all stocks would have the same demand. Adding a request form would improve the user experience and allow for a more efficient addition of new stocks.

Another feature that could be added to the dashboard is customisable sections (modularity). For example, suppose a user is more interested in short term trades. In that case, they might be interested in short term indicators, like volatility or the Relative Strength Index. However, these metrics are not as useful for long-term investors, so they could have metrics like the P/E ratio or Price to Book ratio, which are more value-oriented metrics [20].

In the back-end, the future price predictor could be Improved. The current stock predictor uses historical prices and moving averages to predict future prices. However, it could be further improved by adding more relevant features. Sentiment data could predict the future price. Thus it could be a valuable input for the neural network. Data like trading volume, historic analyst ratings and more could be used as inputs. Though too many features could lead to overfitting in the model, a balance must be achieved for more accurate results.

Also, identifying Support/Resistance trend lines using machine learning could improve the price prediction mode and offer more information to the user in the Price graph [38]. In finance, a trend line is a bounding line for the price movement of a security. It is formed when a diagonal line can be drawn between a minimum of three or more price pivot points.

Partitioning is splitting something into smaller parts to make them easier to work with. Partitioning could be implemented to improve the speed of the back-end code. Partitioning could be used in the code by splitting the list of tickers between multiple systems to speed up the time it takes to web scrape and analyse data.

# References

|  |  |
| --- | --- |
| [1] | V. a. J. N. Bonta, “ A comprehensive study on lexicon based approaches for sentiment analysis.,” *Asian Journal of Computer Science and Technology,* vol. S2, no. 8, pp. 1-6, 2019. |
| [2] | I. R. a. A. S. A. Porshnev, “Machine Learning in Prediction of Stock Market Indicators Based on Historical Data and Data from Twitter Sentiment Analysis,” *2013 IEEE 13th International Conference on Data Mining Workshops,* vol. 13, pp. 440-444, 2013. |
| [3] | Y. Z. C.-N. H. H. C. Robert P. Schumaker, “Evaluating sentiment in financial news articles,” *Decision Support Systems,* vol. 53, no. 3, pp. 458-464, 2012. |
| [4] | A. Damodaran, The Little Book of Valuation: How to Value a Company, Pick a Stock and Profit, 2nd ed., N.J.: Wiley, 2011. |
| [5] | S. Few, Information Dashboard Design: The Effective Visual Communication Of, 2nd ed., Sebastopol, CA: O'reilly, 2006. |
| [6] | BLOOMBERG, “The Terminal,” 2022. [Online]. Available: https://www.bloomberg.com/professional/solution/bloomberg-terminal/. [Accessed 18 04 2022]. |
| [7] | FACTSET, “Factset,” 2022. [Online]. Available: https://www.factset.com/. [Accessed 18 04 2022]. |
| [8] | ChartJS, “Chart.js,” 2022. [Online]. Available: https://www.chartjs.org/. [Accessed 18 04 2022]. |
| [9] | O. Komarov, “How to store financial data: a SQL vs No-SQL comparison,” 2020. [Online]. Available: https://towardsdatascience.com/how-to-store-financial-data-a-sql-vs-no-sql-comparison-bbd0d71bfc26. [Accessed 17 04 2022]. |
| [10] | Simply Wall St, “Dashboard,” 2022. [Online]. Available: https://simplywall.st/dashboard. [Accessed 19 04 2022]. |
| [11] | F. Meza, “Olioapps,” 2018. [Online]. Available: https://www.olioapps.com/blog/the-value-of-code-documentation/. [Accessed 04 04 2022]. |
| [12] | FINVIZ, “FINVIZ,” 2022. [Online]. Available: https://finviz.com/. [Accessed 16 04 2022]. |
| [13] | G. Bland, “Yahoo Finance API – A Complete Guide,” 2021. [Online]. Available: https://algotrading101.com/learn/yahoo-finance-api-guide/. [Accessed 05 04 2021]. |
| [14] | U. Pisuwala, “https://www.peerbits.com/blog/nodejs-vs-golang.html,” 2022. [Online]. Available: https://www.peerbits.com/blog/nodejs-vs-golang.html. [Accessed 13 04 2022]. |
| [15] | E. Ries, “Minimum Viable Product: a guide,” 2009. [Online]. Available: http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html. [Accessed 18 04 2022]. |
| [16] | MongoDB, “Compass. The GUI for MongoDB.,” 2022. [Online]. Available: https://www.mongodb.com/products/compass. [Accessed 18 04 2022]. |
| [17] | N. Swarnkar, “VADER Sentiment Analysis: A Complete Guide, Algo Trading and More,” 2022. [Online]. Available: https://blog.quantinsti.com/vader-sentiment/. [Accessed 17 04 2022]. |
| [18] | S. Singh, “How to Get Started with NLP – 6 Unique Methods to Perform Tokenization,” 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/07/how-get-started-nlp-6-unique-ways-perform-tokenization/. [Accessed 16 04 2022]. |
| [19] | J. Brownlee, “Overfitting and Underfitting With Machine Learning Algorithms,” 2016. [Online]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. [Accessed 18 04 2022]. |
| [20] | E. E. S. K. JONAS ELMERRAJI, “5 Must-Have Metrics for Value Investors,” 2022. [Online]. Available: https://www.investopedia.com/articles/fundamental-analysis/09/five-must-have-metrics-value-investors.asp. [Accessed 15 04 2022]. |
| [21] | Corporate Finance Institute, “Moving Average,” 2022. [Online]. Available: https://corporatefinanceinstitute.com/resources/knowledge/other/moving-average/. [Accessed 18 04 2022]. |
| [22] | R. Mitchell, Web scraping with Python: Collecting more data from the modern web, 2nd ed., O'Reilly Media, Inc, 2018. |
| [23] | X. a. I. I. F. a. K. S. a. W. J. Chu, “Data Cleaning: Overview and Emerging Challenges,” *Proceedings of the 2016 International Conference on Management of Data,* vol. 4, pp. 2201-2206, 2016. |
| [24] | P. R. H. S. Christopher D. Manning, Introduction to Information Retrieval, 8th ed., Cambridge: Cambridge University Press, 2008. |
| [25] | A. Y. N. M. I. J. David M. Blei, “Latent Dirichlet Allocation,” *Journal of Machine Learning Research,* vol. 3, pp. 993-1022, 2003. |
| [26] | P. W. F. D. L. Thomas K Landauer, “An Introduction to Latent Semantic Analysis,” *Discourse processes,* vol. 2, no. 25, pp. 259-284, 1998. |
| [27] | S. Kapadia, “Evaluate Topic Models: Latent Dirichlet Allocation (LDA),” 2019. [Online]. Available: https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0. [Accessed 17 04 2022]. |
| [28] | MonkeyLearn, “Topic Analysis: The Ultimate Guide,” 2022. [Online]. Available: https://monkeylearn.com/topic-analysis/. [Accessed 18 04 2022]. |
| [29] | S. a. V. R. a. G. E. A. a. M. V. K. a. S. K. P. Selvin, “Stock price prediction using LSTM, RNN and CNN-sliding window model,” *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI),* pp. 1643-1647, 2017. |
| [30] | Vernier, “What are Mean Squared Error and Root Mean Squared Error?,” 2018. [Online]. Available: https://www.vernier.com/til/1014. [Accessed 18 04 2022]. |
| [31] | C. M. Bishop, Neural Networks for Pattern Recognition, Oxford university press, 1995. |
| [32] | J. O. Paul Ammann, Introduction to software testing, 2nd ed., Cambridge: Cambridge University Press, 2016. |
| [33] | W. Silverstein, “What Are Edge Cases in Software Testing?,” 2021. [Online]. Available: https://www.mindfulqa.com/edge-cases/. [Accessed 16 04 2022]. |
| [34] | Appdynamics, “Why DevOps is Important,” 2021. [Online]. Available: https://www.appdynamics.com/topics/why-devops-is-important. [Accessed 16 04 2022]. |
| [35] | Jira, “Jira,” 2022. [Online]. Available: https://www.atlassian.com/software/jira. [Accessed 19 04 2022]. |
| [36] | Gitlab, “About Gitlab,” 2022. [Online]. Available: https://about.gitlab.com/. [Accessed 17 04 2022]. |
| [37] | TeamGantt, “About TeamGantt,” 2022. [Online]. Available: https://www.teamgantt.com/. [Accessed 19 04 2022]. |
| [38] | G. Morse, “Programmatic Identification of Support/Resistance Trend lines with Python,” 2019. [Online]. Available: https://towardsdatascience.com/programmatic-identification-of-support-resistance-trend-lines-with-python-d797a4a90530. [Accessed 17 04 2022]. |
| [39] | B. Payne, “On The Financial Characteristics Of Firms That Have Achieved The Highest Levels Of Accuracy In Earnings Predictability,” *Journal of Business & Economics Research,* vol. 5, no. 9, 2011. |

# Table Of Figures

[Figure 1 Dashboard of Amazon (AMZN) 19](#_Toc101889436)

[Figure 2 Stock Screener 20](#_Toc101889437)

[Figure 3 Entity Relationship Diagram of Database 27](#_Toc101889438)

[Figure 4 Cumulative Flow Diagram from Jira 30](#_Toc101889439)

[Figure 5 Gantt Roadmap for January, February and March 32](#_Toc101889440)

1. Probability of A occurring = 1 - (1 - 0.005) ^50 = 0.22168744293136 [↑](#footnote-ref-1)