Stock Dashboard

# Acknowledgements

# Abstract

The Stock Dashboard is a website where the user can search for specific stocks and view live data, such as their article sentiment, analyst target prices, future stock predictions, article topics and more. The website's goal is to help the user make future investing decisions. The data analysis is performed in the backend 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 vigorous data pre-processing like lemmatization, topics are extracted from the text using Gensim and sentiment is analyzed using a lexicon-based approach with the VADER sentiment analyzer. 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. On the search page, the user can search for the desired stock; the dashboard contains all the data visualizations 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

[Acknowledgements 1](#_Toc100484747)

[Abstract 1](#_Toc100484748)

[Literature Review 2](#_Toc100484749)

[Motivation 4](#_Toc100484750)

[Passion for investing and data analysis 4](#_Toc100484751)

[Alternative to expensive software 4](#_Toc100484752)

[The dashboard form is a great fit for the average investor 5](#_Toc100484753)

[Developing the right skills 5](#_Toc100484754)

[Inspiration from existing websites 5](#_Toc100484755)

[Personal use 6](#_Toc100484756)

[Aims and Objectives 6](#_Toc100484757)

[Summer Preparation and challenge week 6](#_Toc100484758)

[Minimum Viable Product 7](#_Toc100484759)

[Final Product 7](#_Toc100484760)

[Challenges 8](#_Toc100484761)

[Technical Documentation 8](#_Toc100484762)

[User Interaction and Design 8](#_Toc100484763)

[Dashboard 9](#_Toc100484764)

[Stock Screener 9](#_Toc100484765)

[Source Code Summary 9](#_Toc100484766)

[Web Scraping and data gathering 10](#_Toc100484767)

[Data Cleaning 10](#_Toc100484768)

[Sentiment Analysis 11](#_Toc100484769)

[Topic Extraction 12](#_Toc100484770)

[Predictive Modelling 12](#_Toc100484771)

[Database 12](#_Toc100484772)

[Quality Assurance 12](#_Toc100484773)

[Test strategy 12](#_Toc100484774)

[User Testing 12](#_Toc100484775)

[Unit Testing 12](#_Toc100484776)

[Project Planning 12](#_Toc100484777)

[Jira 13](#_Toc100484778)

[Gitlab 13](#_Toc100484779)

[Gantt Roadmap 13](#_Toc100484780)

[Conclusion 13](#_Toc100484781)

[Summary 13](#_Toc100484782)

[Future Work 13](#_Toc100484783)

[References 13](#_Toc100484784)

# Literature Review

**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, 2019) compares the sentiment analysis accuracy of the 3 most popular sentiment analysis tools in Python – Natural language processing toolkit (NLTK), Text Blob and VADER. The text analysed is movie reviews from rottentomatos.com[[1]](#footnote-1). Bonta, V. and Janardhan, N.K.N. concluded that VADER is the “gold standard” list of lexical features which are especially attuned to find semantics in microblog texts. The accuracy of VADER was 77.0%, 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 emphasizing sentiment intensity, for example “that movie was good” compared to “that movie was GOOD” or “that movie was GOOD!!!”.

**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.**

(Robert P. Schumaker, 2012) investigated whether sentiment and tone of finical news articles correlate to measurable stock price movements. (Robert P. Schumaker, 2012) found out 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. Investigate the correlation further they found that our system was able to predict price decreases in articles of a positive sentiment 53.5% of the time, and price increases in articles of a negative sentiment 52.4% of the time. This can be an indicator that traders behave in a contrarian manner, e.g., see good news, sell; see bad news, buy.

**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.**

(A. Porshnev, 2013) discussed and tested the possibility of using Twitter users’ moods and psychological states of people to improve stock returns. For the analysis of psychological states, they used a lexicon-based approach and analysed 755 million tweets. (A. Porshnev, 2013) 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 by using a Support Vector Machine algorithm to predict the DJIA indicator, which was only slightly better than a Neural Network.

From (Robert P. Schumaker, 2012) we can conclude that negative and positive sentiment can be used as a tool for increasing returns in the stock market, this could be a useful addition to the dashboard. However, there is a gap in the research In terms of the use of the VADER sentiment analyser specifically, which outperforms other lexicon-based approaches (Bonta, 2019), in analysing finical news articles. (A. Porshnev, 2013) stated that Twitter sentiment does not increase stock returns, but it is known that a simple trading strategy in conjunction with a lexicon-based approach of financial news article sentiment analysis can provide positive returns in the stock market. (Robert P. Schumaker, 2012)

# Motivation

## Passion for investing and data analysis

I decided to make a stock dashboard because I am passionate about investing and love to analyse and visualize data, especially financial data. I have been following the stock market movements for over 4 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. ADD REFERENCE

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

So for my Capstone project, I decided to combine both, 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 or the FactSet terminal, which costs 12 000$ per year. For the average investor like me and many others, this is way too expansive. But isn’t it unfair that the rich investors can use tools that we can’t?

Creating my own stock analysis tool is a great way to save some money and improve my financial knowledge along the way. Of course, my program would never be as advanced as the Bloomberg terminal, but after investing regularly for over 2 years I have understood that having too much data can also be a burden. A dashboard is a great tool for this since it doesn’t 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, that is, having a lot of information on a single page that does not require any scrolling, because, in my opinion, it is a useful way of conveying information, to both new and experienced investors. The average investor cannot spend hours analyzing 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 that requires both data analysis and python programming is a great fit for me. Data visualization is a big part of being a data analyst, however, if I just used Python for the data visualization, this would require the user to have python installed, and in most cases, the visualizations could not be interacted with. Learning another visualization tool like ChartJS can be useful in creating beautiful graphics that anyone with an internet connection can see, as well as allowed me to further develop my front-end development skills.

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

Instead of using a Relational Database Management System (RDBMS) database like MySQL. I thought it would be a good idea to learn NoSQL since i the financial industry a lot of companies use NoSQL instead of SQL and it is great for sstoring financialdata like stock price, which in MySQL would require a huge collection that contain tens of thousands of rows. [REFERENCE]

## Inspiration from existing websites

Throughout the last two years, I have used the online website Simply Wall Street to help me make investing decisions. Simply wall street is a stock and sector analysis tool with the goal of making stock analysis simpler. It condenses large amounts of data from various stocks to give the user an overview

## Personal use

# Aims and Objectives

The main aim of the website was to help the user make future investment decisions, by providing accurate and useful information related to the stock, as well as allowing the user to compare different stocks to each other. From this, I conclude that the website must be easy to use, useful for both new investors and experienced investors, as well as contain information that is important in deciding whether a stock is a good buy or not.

The backend Python code on the other hand 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. (Meza, 2018). Well documented code will allow me not to lose momentum when coding or taking a break to work on projects for other modules.

The code should be reliable and efficient as I was planning to add hundreds of stocks for the user to view and automate such that the code runs every morning to update the most relevant information. Since most of the code would run for very stock any minor speed and efficiency improvements would be amplified by the number of stocks to be updated.

## Summer Preparation and challenge week

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

During the summer I also researched research papers, in the field of data analysis sentiment analysis and price prediction. During this time I also did research into multiple Python libraries that I could use to perform the tasks, like NLTK and TensorFlow. Luckily already had some experience with NumPy and Pandas, which I used throughout the project for data cleaning and data analysis.

For web scraping, my first objective was to find multiple financial news websites, 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 on to use FINVIZ, as it contains information about thousands of stocks, contains articles relevant to the stock (even though later the articles did require further filtering) and contains many relevant metrics for specific stocks, like Price to Earnings[[2]](#footnote-2) ratios and Dividends[[3]](#footnote-3), as well many metrics that are not as popular but could still be important in deciding whether a stock is a good investment. Together with Yahoo France, as it has a free to use API to access information like the stock price more quickly than if I was using FINVIZ.

In terms of the frontend I was quite unsure what tools I would use to except that I would lietoo use Chart JS, because the graphs looked very good and could be interacted with. Since I winted to keep the forntend quite simple because the backend was my priority. I decided to use basic HTML, CSS and JavaScript, which would allow me to spend more time on the bakend - improveong the code and making better predictions. Howver with these tools I oculd not accesss the databse from the wesbite. For thgis I needed a JavaSciuprt abkcend server, whoch I made using NodeJS. THwe ncie thing about this is that this servber allows me to create an API, where other users can use my data, to permod their own vizuilations or predictions.

## Minimum Viable Product

The aims for the Minimal Viable product were as follows:

**Create a samle website, that uses the layout makde during Challnage Week.** i knew that I wouldntr be able to get all of the backend functionality done by the MVP, but having atleast a layout of the daashbord let me know what data I should be focuising on i n the backend.

Setup the MongoDb Database

**Create the pipeline for newly added stocks.** Since some of the code is different for whether a stock is new to the database or not (for example ifd a stock is early in the database than there is no point in searchin for old articles as those will already been added, as well as there would be no point in creating another stock prediction model, if one was already pickled[[4]](#footnote-4).

**Web scrape information about the stock such as articles and fundematal information from the biggest publishers.** My goal was to have the web scarapet to extreact text from the largest 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 a lot of tedius work of amnaully finding where the text is stored in 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 sentiemnt 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 takes into account puncation and word casing. [REFERENCE]

Analyse the sentiment of the articles using a lexicon approach. During the challenge week and summer preparation I realized that a lexicon based approach would be the best way to analyse financial articles on a large scale, without the need for powerful hardware. Thus I wrote a sentiment analyser using the VADER library, but future improvement was needed to improve the accuracy of the analyser, for example tokenization[[5]](#footnote-5).

**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 further test the website with more users, as well as allowed for error testing if the user searches for a stock that does not exist.**

**Predict the stock price using the historic price in a Neural Network. During the Summer and Challnage week I did research into what toops I could use to predict future price and I decided on an LSTM model, using the**

All of the above objectives were accomplished except for the the stock screener.

## Final Product

In addition to the features in the MVP, the objective for the final Product was to:

- [x] Only analyze the setimnint of imporatnat artilces, that is artrilces taht mention the stock multimple times

- [x] Extract topics from the articles

- [x] Create the pipeline to update stock already in the database

- [x] Stock Screener

- [x]

<p align="right">(<a href="#top">back to top</a>)</p>

## 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 sadly does not have very comprehensive documentation or NodeJS and ExpressJS, which I had not used before the project. To overcome this problem, I took online courses on LinkedIn learning and watched YouTube video tutorials. However, this was time-consuming and resulted in me spending less time on making 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 firstly for most stocks I did not have enough sentiment data to make it a valuable metric for prediction and secondly, the stocks that did have enough data to be included in the prediction resulted in a worse predictive model than just using the historic price and moving averages[[6]](#footnote-6). Thus I decided to focus more on other parts of the code as well as improving the stock screener.

The main challenge for topic extraction was how to make sure that the topics extracted are actually valuable to the investor. For example topics (taken from the actual code results) such as “5G Network” or “TikTok App” are useful in gaining a summary of what the companies articles are about and could be sued for further research if necessary, however topics like “Market Year” and “Fund Inc” do not add anything of note to and cannot help with further research or making an investing decision. (all examples taken from Apple topic extraction results). I tried to fix this problem in two ways.

Firstly, I removed words that were mentioned in most or all articles many times. This fixed the problem most of the time but also removed some important information and potential topics. For example during th invastion of Ukraine, there were a lot of articles for Oil stocks that mentioned Russia. With this system in place Russia was removed from the texts and not counted, as it was mentioned too many times. This resulted in relevant infoamtion being removed.

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 by 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.

As for the final product I decided to use a lexicon-based approach using VADER, it is very important that the “bag of words” that the analyser uses is as fitted for article analysis as it can be. For this I tried using a custom lexicon of negative, neutral and positive words, which resulted in worse results than using the pre-defined lexicon for VADER, which is fitted to be sued for social media text analysis.

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

**Information Dashboard Design: The Effective Visual Communication is a book**

# Technical Documentation

## User Interaction and Design

The user interacts with the website through the 3 pages: search page. The dashboard and the stock screener.

The design of the website is simple and uses a dark colourway with more varied colours used in the charts. It was important for the information in 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 better portray 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 good for short term trades.

### Dashboard

Stephen Few (2006) defines a dashboard as a visual display of the most information needed to achieve one or more objectives that fits entirely on a single computer screen so it can be monitored at a glance (Few, 2006)

In this section I 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 usually as a combination of next and graphics.** Dashboards are highly graphical not because it is cute but because graphical representation can often communicate with greater efficiency and richer meaning that text alone. (Few, 2006). With this in mind, I gave myself the task of having as little of text as possible and trying to convey information almost exclusively through graphical forms instead of text. Even though Python in itself does support dashboard graphical design and visualization, ChartJS in my opinion is an even better tool as it allows for detailed consumption of the graphs, quick rendering speeds for anyone with a modern browser and would allow me to create beautiful and eye-catching visualizations, which most investing websites like FINVIZ or MarketWatch do not have.

**Dashboards display the information needed to achieve specific objectives.** To create a useful dashboard it was important to first figure out the objective, which in my case was to inform the user about a stock’s health and investing potential. However, now it was crucial to understand what information would allow the user to improve their understanding of the stock and decide their next step. The financial characteristics of a successful company are still up for debate, but they are known to be stable earning or as stated by (Payne, 2011) the messurment of Value Line Earnings Predictabilit, which is the reliability of earnings forecasts. Return on Equity (ROE). As well as comarping the sepcif company to the broeader market or industry averages. As !!! dicsovred setiment can be used to make market returns, thus it would be useful to include the historical sentiment of articles as well as mobbing averages of this data to see long term trends.

**Dashboards can be monitored at a glance.** For a dashboard to be useful, it should only take a maximum of a few minutes to fully understand the information in it. To make sure this is the case I included information that would be easy to understand but also added short tooltips for users, who might not know what a specific metric is. Colour coding information also makes it easier to understand the information quickly and efficiently, as well as having a summary section that summarized the dashboard in a textual form could make it easier to monitor.

With all of these objectives accomplished the page can be defined as a dashboard, and I have made sure that the information in the dashboard is relevant and easy to understand.

### Stock Screener

## Source Code Summary

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

In general, the code can be divided up into 6 sections (not including testing which I will talk about 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 is the practice of gathering 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 the form of HTML and other files that compose web pages), and then parses that data to extract needed information. (Mitchell, 2018)

The main source for web scraping information for my project is FINVIZ.com, however, the articles are web scraped from a variety of news publishers like Motley Fool and Bloomberg. For getting the historic price data and analyst ratings I used the Yahoo Finance API because it’s reliable and has low latency, also its well documented for Python. The Yahoo Finance API is a range of libraries/APIs/methods to obtain historical and real-time data for a variety of financial markets and products, as shown on Yahoo Finance. (Bland, 2021)

The first step of getting the data was to get the HTML content of the website. For this, I used Requests, which is an HTTP request library as well as BeautifulSoup for parsing the extracted HTML. The fundamental metrics for the stock are extracted from the FINVIZ page, together with the HTTP links for the articles. After this, each link to the article is opened and parsed using Requests and Beautiful Soup and 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, which are used in the latest analyst ratings and historic analyst ratings sections of the dashboard and the historic price, which is used in the price predcion and historic price section I used Yahoo Finance API

### Data Cleaning

To get the sentiment and extract the topics from the articles the text must be cleaned, so as to improve the accuracy of the analysis.

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

To prepare the text for topic extraction more vigorous data pre-processing must be done. Firstly stop words are removed from the text. Beside the stop words I also add some additional words to remove that are often mentioned in articles but do not add any value, for example, company, stock, inc and others.

Secondly punctuation is removed, so as to not generate topics based on punctuation, as those would not be valuable to the user, using the string module.

Thirdly all the words are lemmatized, even though stemming would be the faster option. For grammatical reasons, documents are going to use different forms of a word, such as organize, organize, and organizing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. (Christopher D. Manning, 2008)

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word 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.

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. If confronted with the token saw, stemming might return just s, whereas lemmatization would attempt to return either see or saw depending on whether the use of the token was as a verb or a noun. (Christopher D. Manning, 2008)

Finally, after making all the text lowercase, it is ready for topic extraction.

### Sentiment Analysis

Sentiment analysis is the process of computationally identifying and categorizing the opinions expressed in a piece of text, especially in order to determine the writer’s attitude towards a particular topic, product, etc. is positive, negative or neutral. (Bonta, 2019)

Unlike machine learning algorithms VADER performs better across various kinds of domains. As compared to machine learning techniques, VADER has several advantages. Firstly, it is both quick and computationally economic, which in the case of analysing thousand od articles is highly important. VADER runs directly from 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. 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. (Bonta, 2019)

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.

After performing web scraping and data cleaning, the text is split into sentences, because VADER works better on shorter texts [REFERENCE] and for most publishers the last 3 sentences are removed. The last 3 sentences are removed because often at the end of the article there would be either information about the author or disclosure information, like “This article represents the opinion of the writer, who may disagree with the “official” recommendation position of a Motley Fool premium advisory service.” or information about other articles or the publisher.

After this the title and each sentence is given a positive, neutral and negative score, which together make the compound score from -1 for very negative to 1 for very positive. Each sentence is given a wight based on its length and the tile is given the highest weight of 25% of the final rating, together these create the article compound score which is used is the dashboard visualizations and latest article section.

### Topic Extraction

### Predictive Modelling

Machine learning algorithms are used in the stock market forecasting for a long time [1], [2]. The most common methods are Neural Networks and Support Vector Machine [1], [3]. Usually machine learning algorithms trained on technical data about stock movements, for example moving averages. (A. Porshnev, 2013)

### Database

## Quality Assurance

In this section I will talk about the test strategies I used, to test the website and backend, why I used these strategies and how testing improved the project and its reliability.

### Test strategy

sadasd

### User Testing

### Unit Testing

# Project Planning

## Jira

## Gitlab

## Gantt Roadmap

# Conclusion

In this section I will summarize my work, talk about future improvemtns that could imoprve the product.

## Summary

The goal of the website was to have a dashboard that the user could use to make future investing decisions. I think I have accomplished this task, as well as added additional features that could further improve the usability of the website, such as the stock screener and tooltips for new investors.

The dashboard contains relevant information for all kinds of investors, short term and long term, beginner or experienced. The dashboard is visually pleasing and can be quickly monitored, as well can be used as a tool to discover future research points (such as “Why is the has the article sentiment dropped in the last 7 days? ”. The infaomtion in the website is color coded for ease of use and summarized in textual form to get quick investing insights.

The backend code is efficient , however future improvement is needed in temrns of the documentation of the code. Each section of the code does the required job and uses the appropriate tools and libraries to do so. Request sand BeautifulSoup for Web scraping stock data. ….. The codes uses functional programming techniques for impotoved profuctivyti and memory efficiency. Each dfuntion of the code is thoroughly tested through unit testing, including sample use cases and edge test cases.

The geneared Sentiment data is accurate over 70% of time which is comparable to research data. The topics extracted are relevtn to the stock and can be useful 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 a indicator for the momoentim of the strock. Evn though the stoick predicate could be sued to male money on historic data, past returs nare not indicatev of future returns.

The MongoDB database is fast and effienct and allows for the wbsite to have quick loading times. The orginziont of the JSON data is memory efficient and easy to understand.

## Future Work

Request form.

Customizable Website sections (modular)

Improved Stock predictor.

# References

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.

Bland, G., 2021. *Yahoo Finance API – A Complete Guide.* [Online]   
Available at: https://algotrading101.com/learn/yahoo-finance-api-guide/  
[Accessed 05 04 2021].

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.

Christopher D. Manning, P. R. H. S., 2008. *Introduction to Information Retrieval.* 8th ed. Cambridge: Cambridge University Press.

Few, S., 2006. *Information Dashboard Design: The Effective Visual Communication Of.* 2nd ed. Sebastopol, CA: O'reilly.

Meza, F., 2018. *Olioapps.* [Online]   
Available at: https://www.olioapps.com/blog/the-value-of-code-documentation/  
[Accessed 04 04 2022].

Mitchell, R., 2018. *Web scraping with Python: Collecting more data from the modern web.* 2nd ed. s.l.:O'Reilly Media, Inc.

Payne, B., 2011. On The Financial Characteristics Of Firms That Have Achieved The Highest Levels Of Accuracy In Earnings Predictability. *Journal of Business & Economics Research,* 5(9).

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.

1. <https://www.rottentomatoes.com/> is an American review-aggregation website for film and television. Each film or show is given a score of 0% to 100% by critics and users. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)
3. [↑](#footnote-ref-3)
4. [↑](#footnote-ref-4)
5. [↑](#footnote-ref-5)
6. [↑](#footnote-ref-6)