Predicting Stock Market Trends from News Headlines

CS39440 Major Project Report

Author: Nicolas Petras (nip19@aber.ac.uk)
Supervisor: Dr. Neil Mac Parthaláin (ncm@aber.ac.uk)

30th April 2021
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (with Integrated Year in Industry) (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, U.K.

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Nicolas Petras

Date: 29/04/2021

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Nicolas Petras

Date: 29/04/2021

Acknowledgements

I would like to thank my family, friends and the lecturers who have supported me throughout this degree.

Abstract

This project involved using natural language processing and machine learning to predict price movement for the Dow Jones Industrial Average and Tesla. The project goal was to use the news headlines and their sentiment to make these predictions, which was achieved to some degree. The project team went about implementing the majority NLP pipeline from data acquisition to preprocessing and modelling. They explored the various text preprocessing techniques, e.g. stop word removal and punctuation removal, and their impact on prediction

Contents

List of Figures

Chapter 1

Background

This project will address 'Predicting Stock Market Trends from News Headlines' using a machine learning approach. More specifically, it will focus on using the sentiment from news headlines or other data to predict different financial assets' price movements, including stocks. The headline's sentiment will be classified as positive, negative and neutral. Alongside those crisp labels (classes), the degree of each headline's sentiment will also be analysed to provide better predictions. The assets that this project focused on are the Dow Jones Industrial Average (DJIA) stock index and Tesla stock.

The problem will be tackled using intelligent machine learning approaches to complete sentiment analysis and price prediction. The hope is that classifiers built with machine learning will find links between the sentiment around a particular asset and its price. Especially those links or patterns that we as humans struggle to identify. The project will primarily focus on each asset's short-term outlook, attempting to predict the differences in each day's opening and closing prices. However, longer timelines could be considered.

The sentiment analysis will require human input. This input will determine the sentiment and degree of sentiment for each piece of data (headline or other). The hope is that the classifier will learn each asset's inherent characteristics (e.g. volatility) and how its price responds to a particular type of sentiment during the training phase, without the need for manual tuning. Volatility measures how extreme the positive and negative price swings are in a given asset. [1].

The project's end goal is to develop one or more classifiers that will provide insight into financial market trends. The insight may be new ways to predict an asset's worth or provide more robust links between certain news events and an asset's price movement.

1.1 Motivation for the Project

The author of this report was interested in this project because he is interested in the financial markets and machine learning. He has gained a keen interest in the financial markets in the last year and a half through investing. Having experience in investing gave him knowledge of the types of headlines and terminology used in the finance industry and the principal publications active in the industry from which data could be scraped. The author also knew market mechanics

that can impact the price movement of assets, like growth potential, price to earnings, financials, earnings reports, volatility and many other aspects. Stock market prediction is a fascinating idea, and developing AI that can predict the market to a high degree, could give an investor a significant advantage.

The author was also interested in working on a complex problem and domain rather than completing a traditional software engineering project. Since he already had experience in software engineering and development through his degree and industrial year, and wanted to expand his skillset into his other area of interest: artificial intelligence and machine learning. The research-oriented aspect of this project also provided the author with fitting preparation for a master's degree, which he intends to undertake.

1.2 Financial Markets

Most individuals are usually already familiar with the stock market. Investopedia describes the stock market as [2] "the collection of markets and exchanges where regular activities of buying, selling, and issuance of shares of publicly-held companies take place". Other financial markets include the bond, currency (foreign exchange), cryptocurrency and commodity markets [3]. The assets in each of these markets are all candidates for price prediction.

Investors are constantly trying to outperform the market, making this a fascinating problem. However, even professional investors that have success in beating the market are estimated only to be right about 50% of the time. They can usually only outperform the market for limited periods of time. Even Warren Buffet, seen as the most successful investor of all time by many, has struggled to outperform the US's main index S&P 500 [4], in the last 20 years [5]. That is in contrast to Berkshire Hathaway outperforming the S&P 500 by very large margins prior the 20th century [6].

The three main drivers of an asset's price on any financial market are fundamental factors, technical factors and news/market sentiment [7]. Fundamental factors of an asset describe its inherent value, giving its underlying value. These fundamental factors include assets, liabilities, revenues, earnings (profits), expected growth and other factors. Technical factors are external conditions that affect the price of an asset. These technical factors can be speculative in nature, i.e. trends that inflate the value of an asset even though its fundamentals have not improved. However, they can also include factors that will impact the asset's fundamental value indirectly, like inflation or economic conditions. Market sentiment and News can be based on the factors covered by the two categories mentioned above, but they can also be disconnected. Market sentiment covers the psychology of the investors participating in the market.

As mentioned before, this project will explore the market sentiment factors, focusing on news headlines and how they affect price movement. The Analysis Section 2.2.1 will discuss which assets were chosen.

1.3 Natural Language Processing

Natural Language Processing (NLP) is a branch of Artificial Intelligence and Machine Learning that enables computers to understand, comprehend, and process human language data. This sec-

tion provides some background on NLP, and how that background relates to this project.

1.3.1 NLP Tasks

There are several fundamental tasks that are very often tackled [8] in NLP these are:

- Language modelling: Predicting the next word in a sentence
- Text classification: Assigning text into categories
- Information extraction: Extracting useful information
- Information retrieval: Retrieving useful information (e.g. search engines)
- Conversational agent: Building artificial intelligence can converse with humans (e.g. AI assistants Siri and Alexa)
- Text summarisation: Summarising larger text documents into shorter ones
- Question answering: A system than answer questions posed in natural language
- Machine translation: Translate between different natural languages (e.g. Google and Apple Translate)
- Topic modelling: Discovering inherent topics in a series of documents

This project will mainly focus on text classification and information extraction. The news headlines will be classified by their sentiment: positive, neutral and negative. This type of classification is known as sentiment analysis. Other more advanced classification like sentiment orientation may also be used. Sentiment orientation does not only provide the label positive, neutral or negative, but the degree of the sentiment for each class and a compound score indicating the overall sentiment. This degree of sentiment could be used to predict the percentage increase in stocks or other assets, beyond just predicting if they are positive, neutral or negative for the day.

1.3.2 Human Language

NLP is grounded in understanding the rules of human language and teaching those rules to computers so they can complete useful tasks, like the ones mentioned above. There are four main building blocks to language [9]:

- Phonemes
- Morphemes and Lexemes
- Syntax
- Context

Phonemes provide the units of sound and pronunciation in a language. In English, they include all the letters of the alphabet and some combinations of letters like 'sh' and 'th'. Morphemes are a combination of phonemes and provide the smallest units of meaning in a language. They are not always words; they can also be prefixes and suffixes, e.g. pre, ing, and multi. Lexemes are structural variations of morphemes, e.g. run, runs, and running are all part of the same lexeme form. Syntax in both programming and natural language is a set of rules that define how the language is to be used correctly. In natural language, it defines how to construct grammatically correct sentences. Lastly, context is what conveys the meaning of a sentence. A sentence's, word or phrase's meaning can change depending on the context.

This project will not be concerned with phonemes and the syntax of natural language. Phonemes are a core component of text to speech and conversational agents but are not relevant for this project. Whereas the other two will be relevant to this project, context could be important for text classification. At the same time, morphemes and lexemes will be important in several preprocessing techniques used to prepare the data before training a model on it.

1.3.3 NLP Approaches

The main approaches to NLP problems like the one that this project is tackling are:

- Heuristic and rule-based systems (e.g. dictionary or corpus approach)
- Traditional machine learning (e.g. Naive Bayes, Support Vector Machines, Linear Models)
- Deep learning machine learning approaches (i.e. Artificial Neural Networks)

Traditional approaches heuristic or rule-based approaches typically utilise some knowledge base (e.g. dictionary of positive and negative words), which acts as the heuristic, and based on that heuristic establish some rules that enable classification. For example, based on the dictionary of positive and negative words, count the number of words and if there are more positive words than negative, mark the text as positive. More advanced knowledge bases like WordNet, which maps much more complex relationships in language, when compared to a dictionary, is utilised even alongside machine learning approaches. As described in Section 5.4, WordNet was used in this project as part of the preprocessing.

Traditional machine learning approaches are commonly used to obtain better results than a simply heuristic approaches, which have limited effectiveness. Classification and regressions models are the two of the most commonly used machine learning approaches. These models include Naive Bayes, Logistic Regression, Support Vector Machines (SVM), Hidden Markov Model, Decision Trees and others.

Deep learning models has proved to provided very promising results but is not the silver bullet for NLP or sentiment analysis. The artificial neural networks (ANNs) used in deep learning are prone to overfitting on small datasets, costly to train and maintain, resource-intensive, and uninterpretable. ANNs can also be much more complicated than other methods. Due to all these factors, this project will focus on applying traditional machine learning approaches to text classification and sentiment analysis. Some rules or heuristics may also be used to aid the ML approach in the form of a positive/negative dictionary.

1.3.4 NLP Pipeline

An NLP Pipeline describes the steps needed to complete NLP. It includes the data acquisition at the beginning, several processing steps in the middle that process the data acquired, and produce valuable results before the NLP model is deployed as part of a productive system at the end. The key stages can be described as [10]:

1. Data acquisition: Acquire existing public datasets, or collect new personal data for the specific problem being worked on
2. Text extraction and clean-up: Depending on the way the data was acquired, it may need to be cleaned, and the valuable text may need to be extracted
3. Preprocessing: Removing noise and normalising the data to improve feature engineering and modelling
4. Feature Engineering: Involves analysing, extracting and selecting the most valuable and relevant features
5. Modelling: Creating a model from the data that can be used to provide valuable results
6. Evaluation: Evaluating the model's performance – determining its usefulness
7. Deployment: deploying the model into production

Early on in the project, data acquisition was an easy step because the project supervisor knew of a publicly available dataset relevant to this project [11]. Later on in the project, data was acquired using web scraping for headlines. However, there was never any need to use more advanced or niche techniques, many of which use existing data to create new data like back translation and bigram flipping [12]. During the data acquisition stage, the data should also be labelled if supervised learning is being utilised.

When scraping the data from the web, the data may need to be cleaned, extracting only the relevant information without including HTML mark-up, CCS and JavaScript. However, nothing else was required to clean the data and remove noise like spelling or other correction. Using headlines from reputable organisations eliminated the need to correct/clean the data in terms of spelling or other errors. If social media post from Twitter or Reddit were collected, then spelling and other errors would likely be much more of an issue.

Preprocessing can involve various techniques described later in the report like tokenisation, lemmatisation, stemming, stop word removal and others. Preprocessing is designed to remove noise and normalise the data, which should positively affect the results – reducing overfitting and improving accuracy, recall, and other metrics during the evaluation stage. Although this may not always be the case, and preprocessing can remove valuable information.

Feature engineering can be extensive and one of the most important and intensive stages that can be especially important when improving an established system. Its simplest form is converting all the features produced during the previous stages into a format that the model can use. This format is usually a numeric vector that the model can interpret, learn from and then make predictions on. Feature engineering in its more complex form can include engineering or extracting new features from the data, e.g. sentiment, word counts, and others, or selecting the best features for

the modelling stage. During the later stages of the project, sentiment was mined from the news headlines, so it could be used as the feature to predict the stock's price movement.

Modelling does not have to be machine learning-based. It can include a heuristic or rule-based approach that was mentioned in Section 1.3.3, especially when you do not have much data. In this project, a heuristic approach was never employed to classify the data or extract features, which can be seen as unfavourable. They would likely have been useful early on as a simple approach but were not considered until reading the Practical Natural Language Processing book later on in the project.

Evaluation can take two forms intrinsic and extrinsic. Intrinsic evaluation examines the model performance relative to its training and prediction on the data and covers all the traditional metrics like accuracy, recall, precision and F1 Score. In contrast, extrinsic evaluation focus on the value of the model for its users. Intrinsic evaluation focuses on intermediate objectives, while extrinsic evaluation focuses on the model or product's final objective. Intrinsic evaluation is cheap and quick to perform. However, good intrinsic results can still result in an unsuccessful model if it does not satisfy its final objective or provide the intended value. Extrinsic evaluation is not as relevant in an academic project like this because no defined product has been produced. Also, there are no users who can evaluate the results.

Deployment of the model into a productive setting was not really considered or covered as part of this project. The project focused on experimentation and research rather than producing a tool that users can utilise. Although if time permitted, it would be great to convert the code I used for experiments into an actual tool that others can use to run models on different datasets.

1.3.5 Text Representation

During the feature engineering stage of the NLP pipeline, it is common practice to convert the text data into a numeric form. This format is known as vector. The process of vectorisation (reducing the text into a numeric form) reduces the data to a simpler form that can be operated on more effectively. For example it means that the similarity of two documents (pieces of text) can be compared using a mathematical formula called cosine similarity. Advanced vectorisation also allows for context and semantics to be extracted from the text, which can be stored in a effective manner as a vector. The different types of vectorisation approaches provide different text representations, these include [13]:

- One-hot encoding: The simplest approach, each word in the vocabulary is given a unique integer ID between 1 and the size of the vocabulary $|V|$. Each word is represented by a binary vector (0s and 1s) that is length $|V|$. A list of these words is then used to represent a document.
- Bag of words: Another simply and classical text representation. Each word in the vocabulary is again given an unique integer ID between 1 and $|V|$. Although this representation is much more concise because the order and context of the words in removed, instead each word is simply counted. So instead of a list of binary vectors that each correspond to size $|V|$. One binary vector of size $|V|$ can be used to represent each document, with each index corresponding to the word. A value of 0 at an index means that word is not present in the document and any value > 0 represents the number of times that word is present in the document.

- Bag of n-grams: Same as the bag of words approach, but instead of a single word representing a vocabulary item, each item is a grouping of two or more words 'blue car', 'US attacks' and 'north korea' are examples of bi-grams (or 2-grams). This representations attempts to add some context when compared to the bag of words approach, but also greatly expands the number of vocabulary items exponentially.
- TF-IDF (Term Frequency-Inverse Document Frequency): A more advanced approach that ranks each words importance. The ranking is computed by combing the term frequency (TF) and inverse document frequency (IDF) scores. Each document is represent by a vector of the TF-IDF scores for each word present in the document, while words not present in the document have a score of 0.
- Word2Vec: The most advanced representation, falls under a class of representation known as Word Embeddings. It captures the relationship between words e.g. the relationship between the UK, US, Germany and France – all of which are countries. Words with the same meanings are represented by similar vectors, while dissimilar words also have dissimilar vectors

Chapter 2

Analysis

2.1 Technology Choices

This section provides an overview of the technology choices that have been made.

2.1.1 Programming Language

There are many choices in terms of programming language for an AI/Machine Learning project. The main ones being Python, Java, C/C++, R, and JavaScript [14]. Other more niche choices that were also suggested online were Julia, Haskell, and Lisp [15]. Most of these were not considered. The only real considerations were between the Java Virtual Machine ecosystem (Java and Kotlin) and Python.

Java and Kotlin were considered because of the author's familiarity with the languages and the availability of many libraries for NLP and machine learning like JavaML, Apache OpenNLP, LingPipe, NLP4J, and Standard CoreNLP. Using the JVM or Java ecosystem would have allowed the team to utilise Kotlin. Kotlin is a language that boasts many of Python's benefits like conciseness, developer productivity, and functional programming. It brings this in the form of a traditional compiled programming language, which brings its own set of pros and cons.

In comparison, Python was mainly considered because of its popularity in the machine learning, artificial intelligence and data science community. Python also boasts many attractive language features like enhanced readability, development time, conciseness, lots of capabilities for language processing and others. Kotlin can compensate for some of these features, but not all of them, particularly the dynamic programming, language processing and scripting capabilities Python offers. However, the deciding factor between the two languages was the availability of top-class libraries, tutorials and the vast community of developers around NLP and machine learning. Any of the searches I completed online for tutorials regarding sentiment analysis were all Python-based. Python dominance in this online space convinced the team to learn and use it as part of this project. The decision was made easier by Python's relevance in the software industry as both a scripting and programming language, making the learning valuable for any future career in software engineering or machine learning.

2.1.2 Development Tools

In terms of development tools, the team used the Git [16] version control system and hosted the Git repository on GitHub [17]. Git being the go-to solution for version control for its speed and distributed nature. The team used git on both the Z-shell (Zsh) and through a graphical user interface (GUI) with Sublime Merge [18]. Sublime Merge was used for its simple and effective interface and its speed. The prevalent Git client GitKraken was an alternative, but the author has experience performance issues with GitKraken in the past and prefers Sublime Merge's interface. For writing Python code, the team used a combination of Microsoft's Visual Studio Code [19] with Python plugins and Jetbrain's PyCharm [20]. The team was familiar with these development tools, and they are seen as two of the best tools in terms of editor and IDE. The team downloaded the Anaconda Individual Edition [21], which includes Python [22], many popular Python libraries (e.g. NumPy), Jupyter notebooks, and more. Anaconda allowed the team to install Python, Numpy, Pip, Conda, Jupyter notebooks and other libraries all in one go. Any additional packages could then be installed with Pip [23] or Conda [24], two package managers widely used with Python.

Pipenv [25], Pylint [26] and Yapf [27] and the libraries above were all installed with Pip and Conda. Pipenv was used for dependency management of the primary tool, ensuring that all the necessary packages were installed and could easily be installed by anyone that wanted to run the code. Other dependency management tools like Virtualenv or Conda's environments were considered, but Pipenv was the most sophisticated and effortless one. Pylint was used to check for any style issues in the code, and Yapf was used to format the code. These were both the recommended tools to be used with the Google Python Style Guide [28], which was used in this project.

TravisCI [29] was the tool to be used to implement an Continuous Integration for the repository. Jenkins was another framework that was considered for this purpose, but TravisCI was chosen because it is quicker, simpler and easier to implement. It provides a cloud interface, and does not require self-hosting like Jenkins. TravisCI also integrates very well with GitHub, the platform being used to host the project's repository. Jenkins does provide more advanced features and flexibility, but these were not required for this porject, making TravisCI the better choice.

2.2 Requirements, Tasks and Epics

The early requirements, tasks and epics [30] in this section were identified by the second week of the project. Due to the author's inexperience in NLP and Machine Learning, it was challenging to identify the requirements, tasks and epics required for the project. The author had only had two weeks to read up on the basics of NLP, alongside creating the requirements, learning Python and other tasks.

Along with the project supervisor's help, the following list of requirements, tasks and epics was created. The supervisor prompted the team to think of a novel idea for the project. One that has not been explored before or has not been touched on in much depth. The team wanted to use some existing dataset to simplify the early learning process, allowing the team to experiment without the need to start collecting data first. So it was decided to first work with the DJIA dataset [11] before moving on to novel ideas like Tesla, Bitcoin and the WallStreetBets community. The main two novel ideas that the team came up with were listed as requirements 2a and 2b. Ideally, both

ideas could be explored, but the goal was to at least explore one of these ideas.

These ideas were chosen because their novel appeal, and because they are seen as heavily driven by market sentiment and news, which is the study of this project.

2.2.1 Initial Requirements

The focus of the project (requirements):

1. Apply a machine algorithm on an existing dataset [31] focused on the Dow Jones Industrial Average (DJIA). A US blue-chip index [32]. This task will be a learning exercise and provide a baseline performance level for the other classifier focused on novel ideas.
2. After gaining an adequate level of knowledge in sentiment analysis, machine learning, stock prediction and other areas. The project will focus on some novel idea(s):
   (a) An investigation into social media's impact on the financial market, including the impact of the WallStreetBets (WSB) [33] community of retail investors
   (b) An investigation into unpredictable and popular assets that are assumed to fluctuate somewhat 'randomly' by many investors. The assets of focus would be Tesla and Bitcoin. An attempt would be made to find underlying patterns which could be used to predict them

WallStreetBets is a community of retail or individual investors without any corporate backing that had a big influence on a few stocks at the start of 2021 [34]. None of the Wall Street institutions or mainstream media outlets took into account this group's impact, before they started running up the price of several stocks. This made it an interesting idea, to check how much and for how these retail investors were having an impact on the market.

It quickly became evident that apply machine learning algorithms during the modelling stage was not a lengthy process. Most of the work was in the preceding stages of the NLP pipeline. Getting the right data was the most important, and switching out or adding new models was relatively easy.

2.2.2 Initial Tasks

Apart from implementing the requirements themselves directly, the following tasks were identified:

- Research into Natural Language Processing (NLP), specifically Sentiment Analysis and Orientation
- Research into the implementation and evaluation of different machine learning techniques suitable for this problem
- Research and planning in terms of software engineering – testing, version control, continuous integration, coding standards and others
- Gaining more experience with Python

- Learning how to create graphs and diagrams in LaTeX, Python or other specialist software to summarise experiment results
- Preparation for the two demonstrations that are part of the assessment for this project
- Keeping a daily record of the work completed in a project diary
- Weekly meetings with the supervisor
- Converting research, notes, design and others into the final project report
- Creating data sets of news headlines and social media posts

2.2.3 Intial Epics

Epics in order or priority:

- Come up with an initial approach with the DJIA dataset
  - Includes the NLP research and Python learning tasks, from that research and learning price prediction with sentiment analysis should be implemented
- WSB Prediction
- Tesla Prediction
- Bitcoin Prediction

Chapter 3

Methodology and Process

The methodology used to plan and organise the work for the project was the Scrum framework [35]. Since the Scrum framework is lightweight and general, other frameworks and resources are being used to supplement the areas Scrums does not prescribe. Extreme Programming (XP) ideas [36] are being employed in terms of development practices, i.e. continuous integration, simple design, and others. Not all of the practices in XP will be utilised, e.g. planning game is not needed because Scrum provides the sprint planning, and pair programming is not possible outside of a team with multiple members. Other resources have also been utilised to help with other details, such as stories and estimation.

3.1 Why was Scrum chosen?

Firstly, a decision had to be made between an agile and a plan-based approach. Agile was chosen because of its flexibility and its benefits when applied to software development compared to plan-based approaches. The limited personal experience in the subject area meant there was bound to be much uncertainty, and agile approaches are much better at dealing with this uncertainty. Decisions are delayed as long as possible. There is no requirement to create detailed requirements and design specifications up-front, making it a lot easier to approach this project. The team had a focus for the project described in Section 2.2.1, but lacked a concrete plan to implement those ideas. Agile provided the tools to gradually build that plan through an incremental design and experimentation. A plan-based approach would have been detrimental to this project's success because it was impossible to create detailed requirements and design that were concrete enough and unlikely to change significantly.

Scrum was the agile framework chosen because it provides the best structure for planning and organising the work on a project through its events, artefacts and commitments. Scrum achieves these benefits while remaining lightweight and flexible. Extreme Programming was another option, especially since the plan utilises the XP practices for development and short iterations (sprints) like XP. Both frameworks would have been reasonable, but Scrum offered more flexibility and structure through additional events: sprint review and retrospective. As described by Kent Beck [37], Extreme Programming is very business and team-focused (multiple members), which would make it harder to adapt to this project. Practices like sit together negotiated scope contract, pair programming and collective code ownership would be irrelevant for this project. It is also a rel-

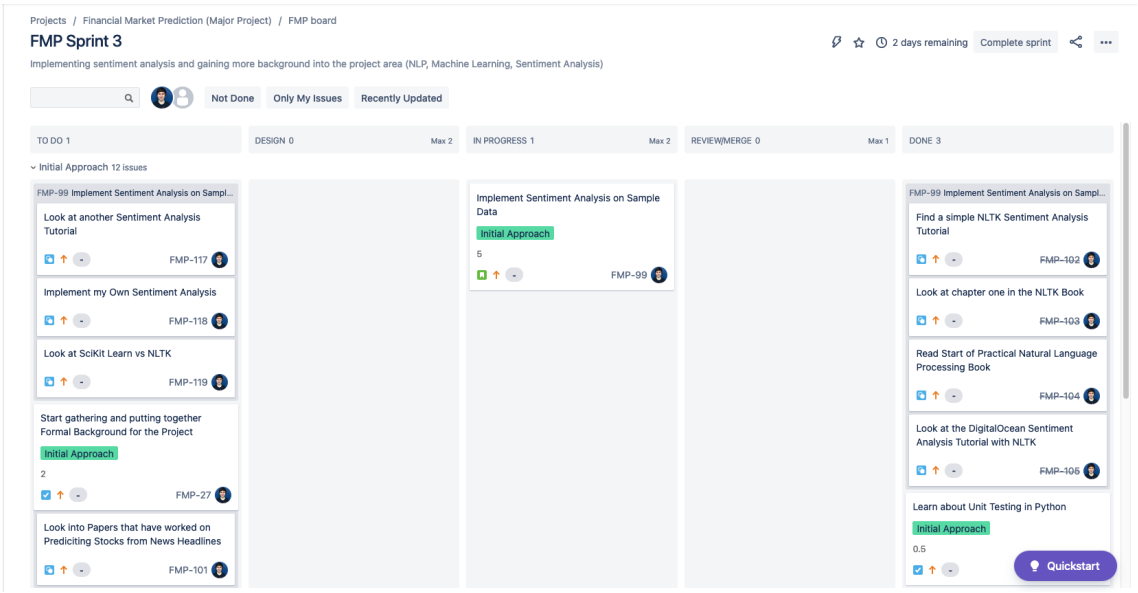


Figure 3.1: A look at the Scrum board used in the project, WIP limits have been applied to the Design, In Progress and Review/Merge columns

atively heavyweight framework when compared to Scrum. Its guide is a book that is a hundred and ninety pages in length. The Scrum Guide is only fourteen pages in comparison. Even the XP inspired development practices that have been utilised in this project are not all explicitly mentioned in Kent Beck's book. Practices like TDD, refactoring, and coding standards may have been common in XP teams, but Beck never explicitly mentions them. These factors combined made Scrum more suitable for this project, while cherry-picking the best and most suitable development practices. See Section 3.3 for the development practices that were to be employed in this project. Many of which were inspired or started with Extreme Programming.

Kanban [38], another popular methodology, was too simple. This simplicity provides little structure, and the project was not exceptionally well suited to the constant flow of work without a time box. Even though the simplicity of Kanban can be beneficial, it was determined that more structure was desired. However, the project did use a Kanban board to visualise the work being completed. A common practice when using Scrum. The project also adopted the work in progress (WIP) limits to improve focus. A limit of two issues was applied to the design and in-progress columns and a one-issue limit on the review/merge column. See Figure 3.1 for an overview of the Scrum board used in the project.

The author's practical experience with Scrum during his industrial year helped cement the choice to use Scrum over Extreme Programming.

3.2 Implementation of Scrum

3.2.1 Roles and Scrum Events

The Scrum roles could not be applied to this project in the traditional sense because there was only one contributor, although they were still employed, because they provided structure to each part of the work, and the role of the Scrum Master reminded me to stick to the process. The author of this report undertook all the roles. He made estimations, created the sprint plan, and completed each sprint's work as a Developer. Managed the Backlog; created the stories and other issues; represented the customer, and attempted to maximise the product's value in the Product Owner role. Lastly, he attempted to enforce the Scrum and Agile principles, values and guidelines as the Scrum Master. His work as a Developer required attention to research, design, development, documentation and quality assurance. The project supervisor acted as an advisor and was also able to provide input from the Customer/Product Owner perspective.

All the sprints were one-week in length and lined up with the weekly supervisor meeting, except the very first sprint. The initial plan was to have two-week sprints, but this was changed after the first retrospective. All Scrum events were retained, including the daily scrum. The daily scrum provided a formal opportunity to inspect progress towards the sprint goal, plan the work for the upcoming day, and consider if any changes needed to be made to the sprint backlog. It was to take place at 10 am and lasted five to ten minutes or less. The daily scrum did not always take place at the scheduled time, but an effort was made to complete it every day before starting that day's work. The sprint review and retrospective were scheduled for a total of thirty minutes. In these thirty minutes, the team reflected on what was completed in the sprint during the review portion. The retrospective portion involved reflecting on what went well and what could be improved in the upcoming sprint. Since there was only one contributor to the project and no other stakeholders, there was no demo or presentation during the review. The sprint planning took place after the retrospective and was timeboxed for a maximum of one hour. The meeting considered all three topics mentioned in the Scrum Guide. See Chapter 5 for an overview of all the Sprints, including notes on the events.

After the sprint planning, the team participated in the weekly supervisor meeting, and all three Scrum events seeped into this meeting. The work completed in the last sprint was presented and discussed in the weekly supervisor meeting, akin to the demo in the sprint review. During this meeting, the supervisor also provided feedback on the progress and process the project was following (retrospective). Furthermore, he also provided input into what he thought should be worked on before next week's meeting (planning). So the supervisor became an indirect participant and contributor to the Scrum events. It is mentioned above that the sprint planning was timeboxed for one-hour. That one hour covered the team's planning before and after the supervisor meeting. The planning was not finalised until after the supervisor meeting because the team wanted to consider the supervisor's feedback and alter the plan based on that feedback.

3.2.2 Product Backlog Management, Stories and Estimation

Grooming of the product backlog, creating stories and other issues, and estimation of those issues was done during the sprint when new ideas arose or the product backlog was deemed to require grooming.

Jira was the software of choice to facilitate the project in the context of Scrum. Jira stored the product backlog, sprint backlog, Scrum board and provided various reports on the sprints, includ-

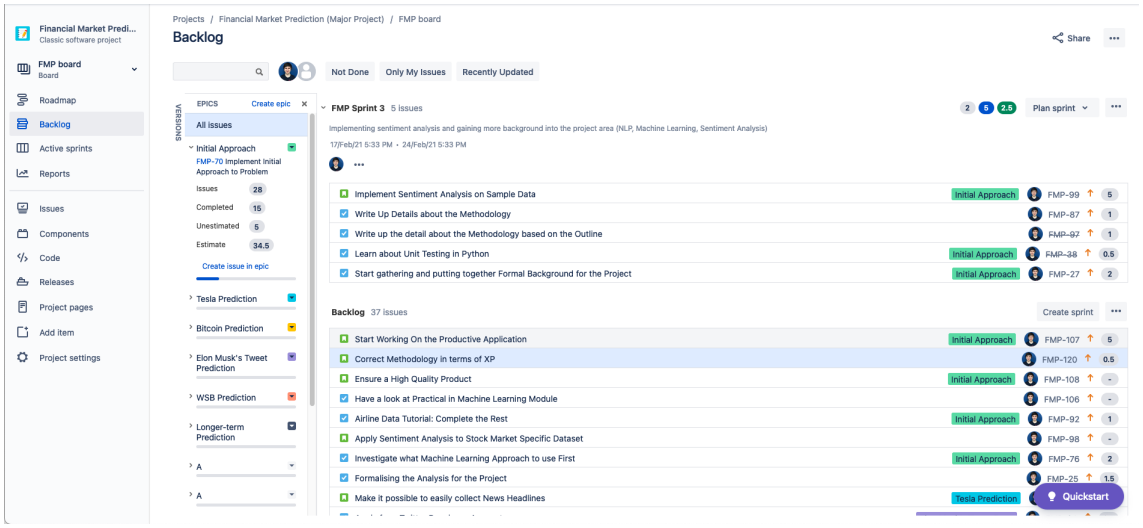


Figure 3.2: A look at the Product Backlog used in the project, Epics are shown on the left and issues like Stories, Tasks and Spikes are shown in the centre of the screen

ing velocity, burn-down and others. Screenshots of the Scrum board and product backlog can be seen in Figures 3.1 and 3.2. The Scrum board had five columns: To-Do, Design, In-Progress, Review/Merge and Done. The Design and Review/Merge columns were not present in the default template and were added by the author. Both these columns were added to provide the developer with an elevated prompt to think about the design before starting to code, make sure their code or other artefacts were reviewed and merged before being marked as done.

Every story and task was given a story point estimation, while spikes were given a time box in terms of hours or days. Within this project, two story points roughly represented one ideal day of uninterrupted work. Two points were decided to represent one ideal day so that one and half a story point could be used to present half and quarter of an ideal day, respectively, without having to resort to 0.25 story points for a quarter of a day. Reducing story points to ideal days is maybe not the right approach, but James Shore recommended it [39]. James Shore's recommendation is why it was adopted in the project. After further research into stories, it was discovered that this somewhat defeats the point of story points [40]. Story points are meant to be a relative measure that provides a universal estimate across the whole team. Whereas ideal hours or days to complete a story will be different across the team, depending on their experience. After discovering that ideal days was not the correct estimation technique for stories, the project tried to move away from it to a more relative and abstract measure. However, the team did not want to make the earlier estimations irrelevant, so the estimations and estimation approach were mostly kept the same. Ideal days estimation made it easier to make estimations, especially in area that the team had little experience. The story point estimation would not bring the benefits it provided in a team of multiple individuals. So even though the team attempted to think of the estimates in a more abstract way. Two story points usually equated to roughly one ideal day of work.

Epics [30] were used to contain larger ideas that spanned multiple sprints. Epics were not estimated, but the stories, tasks and spikes inside them were. An attempt was made to utilise stories as much as possible, but it was often challenging to frame the work from the customer's perspective. Consequently, tasks were often used instead of stories but were categorised under the larger customer goal: epics.

3.2.3 Design and Implementation

Since Scrum, XP or Agile prescribe no specific design process in general like the detailed design specification requirements in plan-based approaches. The project planned to utilise notes, CRC cards, TDD, and other lightweight techniques during design and development. During some work, especially when completing research or spike coding, the XP practices were not expected to be followed if they impeded the exercise's goal.

3.2.4 Product Goal

The product goal, introduced in the 2020 Scrum Guide, was not implemented from the start. It was implemented later because the team was not familiar with the newest Scrum Guide when the project began after reading the 2020 Scrum Guide changes. The team decided to adopt the practice during the third sprint to articulate and keep in mind the project's long-term goal. Jira does not currently provide any way to associate a product goal with the product backlog, but the following goal was kept in mind during the project:

> Implement price prediction for one or more novel assets, which lack investigation: Tesla, Bitcoin and WallStreetBets investments.

3.2.5 Definition of Done

For code to meet the definition of done (DoD), it will have to satisfy the following conditions:

- Automated unit and integration tests exist to cover any functionality, refactoring or other programming completed. Or a strong justification has been made why the tests are not required
- Passes current tests and other quality gates in the continuous integration (CI) build
- Passes linter, style guide followed
- If required, relevant documentation exists to support code

To make sure the DoD is satisfied for each issue that produces code. The code related to that issue will be reviewed manually, and by an automated process, i.e. CI build. Automation will attempt to act as a single source of truth, but human judgement will need to be made to evaluate if the relevant documentation and tests are present. Reviews by humans are prone to human error and are especially problematic when there is only one contributor like in this project. So extra care will be taken when completing code reviews.

The DoD will not be applied to code produced as part of a tutorial, spike or learning, but subsequent 'production' code produced with knowledge gained during the tutorials or spikes will be expected to meet the DoD.

3.3 Development Practices

The project expects to utilise the following practices, if possible:

- Continuous Integration: TravisCI [29] will be used to at least run unit tests and style checks
- Test Driven Development: If it is possible, then TDD will be used to help design and write the code
- Refactoring: Regular refactoring when using TDD, and completing tasks and stories is expected
- Pull Requests, and Branches: Branches should be used for each 'feature' add as part of Jira issue, those bramches will then be merged into the 'master' branch using a pull request process.
- Simple Design: The design should be kept as simple as possible, TDD and Refactoring will aid with this practice
- Coding Standards and Conventional Commits: The project will use the Google Python Style Guide [28], these coding standards will be enforced by running a linter in the CI build. Conventional commits [41] is another guide that has been followed during this project. Alongside the guidelines from Conventional Commits, the Jira issue ID e.g. FMP-1 will also be included at the start of each commit, so that the issues in Jira can be connected to the commits on GitHub
- System Metaphor/Domain Driven Design: The project will attempt to structure the code in an easy to understand manner and relate its structure and the names to concepts in NLP and Machine Learning

Chapter 4

Technical Overview

4.1 Testing

Testing is critical to provide a quality gate to reduce bugs and prove that the code works as intended in the specific scenarios.

4.1.1 Approach

All of the units (functions) outside of the scripts have all have tests, mostly created using test-driven development (TDD). The team attempted to cover all the critical scenarios required.

The testing was completed using the Python unittest framework. These tests were run in every CI build, ensuring the code was working before any changes could be merged.

No integration or acceptance tests have been created. If a more sophisticated tool was created, then these would have likely been added.

4.2 Results

4.2.1 DJIA Bag of Words with Hold-out

Observations of the results in Table 4.1, and the top features output during the experiment:

- Logistic Regression had a lower overall accuracy when compared to Naive Bayes, but probably produced much more generalised and robust classifiers, because it was not overfitting to stop words like ‘the’, ‘to’ and ‘of’. This overfitting shows up in the results, stop word removal and frequency removal generally had negative impact on accuracy. The top features were much more meaningful out of the box for Logistic Regression. Hold-out may be at fault for the Naive Bayes overfitting.
- Stop word removal also affected Logistic Regression negatively, even though it did not seem to be relying on those words based on the top features

Preprocessing Applied	Naive Bayes Accuracy	Logistic Regression Accuracy
CountVectorizer default	49.5%	42.6%
Stopword Removal	50.0%	**40.2%**
Frequency Removal	45.2%	42.9%
Stemming	**51.6%**	**47.4%**
Stemming & Stopwords	50.8%	45.5%
Stemmming, Stopwords & Frequency Removal	48.9%	47.9%
Lemmatisation	48.7%	45.5%
Lemmatisation & Stopwords	48.1%	44.7%
Lemmatisation, Stopwords & Frequency Removal	**46.3%**	46.0%

Table 4.1: Results from the DJIA Bag of Words with Hold-out Experiment

- Interestingly neither classifiers worst accuracy was on the data with with only CountVectorizer's default preprocessing, which includes lowercasing of the data, removing punctuation with spaces and removed words that are two or less letters in length.
- Stemming outperformed lemmatisation for both classifiers, providing the best accuracy for both. For Naive Bayes, lemmatisation, stop word and frequency removal provided the worst accuracy.
- The results are generally inconsistent and showing some strange behaviour. The use of hold-out may be to blame for these inconsistent and strange results.

4.2.2 DJIA Bag of Words with Cross Validation

4.2.2.1 Multinomial Naive Bayes

4.2.2.2 Logistic Regression

Observations of the results from Tables 4.2 and 4.3:

- Improves the general accuracy slighty, over previous best results from 4.2.1.
- Lemmatisation is a lot more on par with stemming accuracy, both performing basically equally in terms of accuracy and F1 Score
- These results make more sense that when hold-out was utilised, which indicates that cross validation has removed the overfitting issue in the last experiment. Likely resulting in classifiers that are much more reliable and robust.
- For Naive Bayes the best test accuracy and F1 score, result in the worst training accuracy and F1 score, which indicates a more general classifier that is not overfitting to the trainning data

Preprocessing Applied	Test Accuracy	Test F1 Score	Train Accuracy	Train F1
Basic Count Vectorizer	50.1%	59.3%	99.9%	99.9%
Stopwords	50.2%	59.2%	99.9%	100%
Frequency Removal	50.3%	54.2%	84.4%	85.5%
Stemming	50.2%	58.2%	100%	99.9%
Stemming & Stopwords	50.4%	58.2%	100%	99.9%
Stemmming, Stopwords & Frequency Removal	**51.8%**	56.6%	83.1%	84.3%
Lemmatisation	50.1%	59.4%	100%	100%
Lemmatisation& Stopwords	**50.0%**	59.3%	100%	100%
Lemmatisation, Stopwords & Frequency Removal	**51.7%**	56.2%	82.7%	83.9%

Table 4.2: Results from the DJIA Bag of Words with Cross Validation Experiment for the Naive Bayes classifier

Preprocessing Applied	Test Accuracy	Test F1 Score	Train Accuracy	Train F1
Count Vectorizer	49.5%	53.9%	100%	100%
Stopwords	49.2%	54.4%	100%	100%
Frequency Removal	49.3%	53.0%	100%	100%
Stemming	51.1%	55.0%	100%	100%
Stemming & Stopwords	51.3%	56.0%	100%	100%
Stemmming, Stopwords & Frequency Removal	50.0%	53.3%	100%	100%
Lemmatisation	51.4%	55.8%	100%	100%
Lemmatisation& Stopwords	51.2%	55.9%	100%	100%
Lemmatisation, Stopwords & Frequency Removal	50.4%	53.5%	100%	100%

Table 4.3: Results from the DJIA Bag of Words with Cross Validation Experiment for the Logistic Regression classifier

Classifier	Test Accuracy	Test F1 Score	Train Accuracy	Train F1
Logistic Regression	53.4%	69.6%	53.4%	69.6%
Bernoulli Naive Bayes F1	53.5%	69.7%	53.5%	69.7%

Table 4.4: Results from the DJIA Sentiment Scores Experiment

Preprocessing Applied	Naive Bayes Accuracy
CountVectorizer	54.4%
Stopword Removal	53.7%
Frequency Removal	51.0%
Stemming	53.7%
Stemming & Stopwords	53.7%
Stemmming, Stopwords & Frequency Removal	55.0 %
Lemmatisation	55.0%
Lemmatisation & Stopwords	54.4%
Lemmatisation, Stopwords & Frequency Removal	54.5%

Table 4.5: Results from the Tesla Bag of Words Experiment

- Logistic Regression has 100% training accuracy and F1 score for all preprocessing, which indicates that Logistic Regression always fits to the training data perfectly. This characteristic may make the classifier more prone to overfitting.

4.2.3 DJIA Sentiment Scores (VADER)

The preprocessing that was applied to this data was lemmatisation, lowercasing, tokenisation, punctuation and stop word removal.

Observations of the results from Table 4.4:

- A significant improvement in accuracy
- F1 Score and Accuracy indicate that the sentiment of the headlines is having an impact on the DJIA stock price

4.2.4 Tesla Bag of Words

Observations of the results from Table 4.5:

- A significant improvement in accuracy when compared to all other results, especially the DJIA Bag of Words experiment. This large improvement in accuracy indicates that the Tesla financial data is more relevant in predicting the stock price, when compared to the world news used for the DJIA dataset.

Chapter 5

Sprint Overview

5.1 Introduction

This section gives an overview of all the work completed during this project in chronological order, based on the work completed in each Scrum sprint. An overview of all of the sprints completed in this project is given below.

No.	Sprint Goal	Length (Weeks)	Dates
1	Complete initial research and learning to be able to begin my first approach	2.5	25/Jan/21 - 10/Feb/21
2	Gain more experience and background in the area of NLP and the problem being studied, to enable work on my own stock market data sets	1	10/Feb/21 - 17/Feb/21
3	Implementing sentiment analysis and gaining more background into the project area (NLP, Machine Learning, Sentiment Analysis)	1	17/Feb/21 - 24/Feb/21
4	Start implementing Sentiment Analysis and Price Prediction using Stock Market Data	1	24/Feb/21 - 10/Mar/21
5	Implement the Productive Application	1	03/Mar/21 - 10/Mar/21
6	Establish Productive Application, and Complete Mid-Project Demo	1	10/Mar/21 - 17/Mar/21
7	Improve DJIA Price Prediction through Sentiment Analysis and Price Prediction, and look to start gathering my own data	1	17/Mar/21 - 24/Mar/21
8	Get the Background Chapter for the Project Completed	1	24/Mar/21 - 31/Mar/21
9	Attempt to collect historical Tesla data and start technical chapters of the Report	1	31/Mar/21 - 07/Apr/21
10	Continue Technical Write-up, and start experimenting with Tesla Data	1	07/Apr/21 - 14/Apr/21

Each section in this chapter will be broken down three subsections: planning and overview, contents and review and retrospective. The planning and overview subsection will include the sprint goal, and description of the sprint planning event. A description of the work completed during the sprint will be included in the contents subsection, and the review and retrospective subsection will include notes from the review and retrospective events held at the end of the sprint.

5.2 Sprint One

5.2.1 Planning and Overview

> "Complete initial research and learning to be able to begin my first approach"

The first sprint concentrated on getting to grips with natural language processing (NLP), Python and machine learning enough so that the project could start working on the DJIA data in the next sprint. One of the sprint's objectives was not mentioned as part of the goal because the team was only made aware of it after the sprint had started. This objective was figuring out the concrete focus of the project through the project outline deliverable.

This sprint's description may lack some detail because the sprint report is not available, and the project diary was also lacking in detail at this early stage.

The initial planning for this sprint was relatively barebones, only the following three Jira issues were included in the initial plan:

- FMP-79: Initial Research/Reading into Sentiment Analysis
- FMP-78: Research Langauge to use for the Project
- FMP-80: Research into the Best Investment Asset to Use in First Approach

The team fully expected to add more Jira issues during the sprint when they became aware of new work items that needed to be tackled. In fact, a further eight issues were added during the course of the sprint. Making such a substantial change to the sprint plan is usually discouraged but was required in this case when the team was uncertain on what issues they would need to complete at the beginning of the sprint. The full list of issues present in the sprint can be seen in Figure 5.1. Most of these issues fell under the 'Initial Approach' epic mentioned in Section 2.2.3.

Two tasks that were completed as part of this sprint: setting up the git repository on GitHub and the Scrum project on Jira, were not included as issues because they were already completed before the sprint was was created in Jira.

The issues in the sprint are all tasks and spikes. The team struggled to frame the work as stories, as mentioned in Section 3.2.2. This framing resulted from there being no apparent customer and the team's unfamiliarity in the subject area.

T	Key	Summary
☑	FMP-40	Think about high-level software design - social media and news analysis
☑	FMP-36	Look into data collection, scraping data from websites
☑	FMP-45	Start experimenting with Sentiment Analysis
☑	FMP-33	First look at MMP Information Pages
☑	FMP-77	Research into Sentiment Orientation and Analysis
☑	FMP-37	Complete the Project Outline
☑	FMP-44	Begin Learning Python - Basics
☑	FMP-46	Look at existing datasets
☑	FMP-80	Research into the Best Investment Asset to Use in First Approach
☑	FMP-78	Research Langauge to use for the Project
☑	FMP-79	Initial Research/Reading Into Sentiment Analysis of Text and NLP

Figure 5.1: An overview of all the issues present in the first sprint

5.2.2 Sprint Contents

During this sprint, the team started doing lots of reading into NLP and looked at various articles and tutorials as part of issues FMP-79, FMP-77, and FMP-45. These articles and tutorials included:

- Text Sentiment Analysis in NLP [42]: Provided overview of the basics of NLP and sentiment analysis process, without getting into the implementation details.
- Top Machine Learning Algorithms Explained [43]: Provided an overview of the main machine learning algorithms.
- How to Perform Text Mining with Sentiment Analysis [44]: Similar to 'Text Sentiment Analysis in NLP', it provided an overview of NLP and sentiment analysis. It was one of the team's first exposures to aspect-based sentiment analysis and opinion unit extraction.
- Alexa Tutorial Part One [45]: Got to apply basic preprocessing to some example data. This was the first practical tutorial the team completed.
- Alexa Tutorial Part Two [46]: The continuation of the above tutorial, got to experiment with topic modelling and sentiment analysis with a pre-trained model known as VADER (Valence Aware Dictionary for Sentiment Reasoning). This tutorial provided the team's first exposure to VADER. A tool used later in the project.

As part of the FMP-78 spike, the team had to choose the primary programming language for the project. The team wanted to make this decision relatively quickly, at the latest, by the first week. So that if Python was chosen, the team could begin learning the language and using it as early as possible. Python was chosen as the language of choice, as described in Section 2.1.1. The team decided to use Python in the project after the first supervisor meeting. They thought it was the appropriate choice because of the reasons mentioned in Section 2.1.1. Since the supervisor had no concerns with the decision, they decided to go ahead with Python as the language of choice.

The team began the 'Complete Python Bootcamp from Zero to Hero' Udemy [47] course after Python was chosen as part of the task FMP-44. The course was bought by the author a few years ago. He had looked through the first parts of the course when he bought it but never used Python outside of the course. During this sprint, the sections on data structures, statements, methods and functions were completed. Completing these sections resulted in the team gaining a solid understanding of all the Python basics. Enabling them better understand tutorials and start creating their own Python code.

During the spike FMP-80 'Research into the Best Investment Asset to Use in First Approach', it was decided that currency would make the most sense, but after discovering the dataset that the supervisor referred the team to was for the Dow Jones Industrial Average. A decision was made to focus on that index as the first asset.

The methodology/process was refined and defined in this sprint. Decisions like the sprint and ceremony timeboxes, the definition of done and how the repository would be managed were some of the questions pondered. This work culminated in the core of the process described in Chapter 3. Very little was changed after the first sprint.

A lot of the effort this sprint was spent on the project outline. At times it felt like too much, but it was critical to get the project's focus figured out. The team spent most of the second week working on the outline. Several revisions were made to make the focus more concrete, refine the report and reduce it to the correct length.

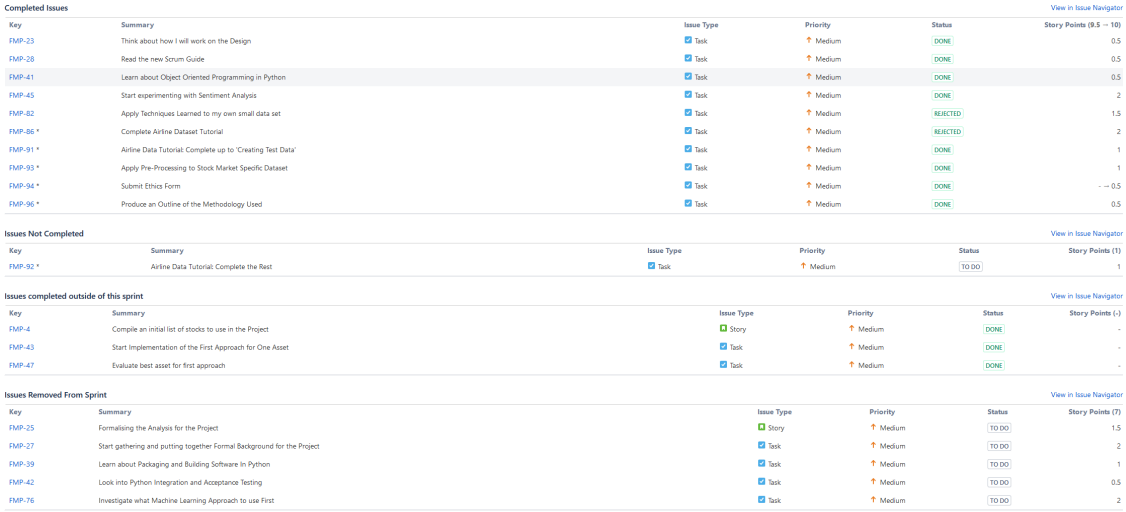
The team investigated how data could be collected in the future as part of FMP-36. The initial plan was to try and collect the data manually as suggested in the 'Text Sentiment Analysis in NLP' [42]. However, the project supervisor said this was not feasible because it would too time-consuming. So, a little investigation was done into what websites could be scraped and how this would be done. The team had a look at a Python HTML parser library called Beautiful Soup [48], which looked like it would be helpful alongside Python the requests module. They looked at several websites Yahoo Finance, Barron's, Wall Street Journal and others. Yahoo Finance was the most promising in terms of data because it collated headlines from a substantial variety of reliable sources. However, the website's HTML mark-up had a complex structure and might be hard to scrape. The team also looked at Reddit and Twitter's APIs for scraping data. Reddit's API looked very easy to use, while Twitter's was more complex but seemed to provide a nice set of features. Reddit and Twitter were investigated because they would likely provide the data if Wall Street Bets or the general social media impact was investigated.

5.3 Sprint Two

5.3.1 Planning and Overview

> "Gain more experience and background in the area of NLP and the problem being studied, to enable work on my own stock market data sets."

The initial plan for sprint two was to complete some formal background and analysis. The plan was to complete a literature review of sorts, formalise this background in a report form, and covert the project outline's tasks into a more formal form suitable for the final report. However, after discussion with the project supervisor, he suggested that the project starts developing some code



Completed Issues — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (9.5 → 10)
FMP-23	Think about how I will work on the Design	Task	Medium	DONE	0.5
FMP-28	Read the new Scrum Guide	Task	Medium	DONE	0.5
FMP-41	Learn about Object Oriented Programming in Python	Task	Medium	DONE	0.5
FMP-45	Start experimenting with Sentiment Analysis	Task	Medium	DONE	2
FMP-82	Apply Techniques Learned to my own small data set	Task	Medium	REJECTED	1.5
FMP-86 *	Complete Airline Dataset Tutorial	Task	Medium	REJECTED	2
FMP-91 *	Airline Data Tutorial: Complete up to 'Creating Test Data'	Task	Medium	DONE	1
FMP-93 *	Apply Pre-Processing to Stock Market Specific Dataset	Task	Medium	DONE	1
FMP-94 *	Submit Ethics Form	Task	Medium	DONE	- → 0.5
FMP-96 *	Produce an Outline of the Methodology Used	Task	Medium	DONE	0.5

Issues Not Completed — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (1)
FMP-92 *	Airline Data Tutorial: Complete the Rest	Task	Medium	TO DO	1

Issues completed outside of this sprint — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FMP-4	Compile an initial list of stocks to use in the Project	Story	Medium	DONE	-
FMP-43	Start Implementation of the First Approach for One Asset	Task	Medium	DONE	-
FMP-47	Evaluate best asset for first approach	Task	Medium	DONE	-

Issues Removed From Sprint — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (7)
FMP-25	Formalising the Analysis for the Project	Story	Medium	TO DO	1.5
FMP-27	Start gathering and putting together Formal Background for the Project	Task	Medium	TO DO	2
FMP-39	Learn about Packaging and Building Software In Python	Task	Medium	TO DO	1
FMP-42	Look into Python Integration and Acceptance Testing	Task	Medium	TO DO	0.5
FMP-76	Investigate what Machine Learning Approach to use First	Task	Medium	TO DO	2

Figure 5.2: The sprint two report

and applying the knowledge acquired in the previous sprint. This approach seemed sensible, especially since the formal analysis was not required.

After the changes to the sprint plan, the plans were no longer concrete and were quite vague, which resulted in a lot of changes over the course of the sprint again. As seen in 5.2, the issues without an asterisk next to their ID were the issues initially planned for the sprint, and a couple issues were Rejected. The issues under 'Issues completed outside of this sprint' were simply marked as done outside of the sprint because they were already completed as part of other issues.

The main two tasks (FMP-45 and FMP-82) are directly oriented around the sprint goal. The other tasks all supported the sprint's goal. The first designs for the program were going to be need as part of this and future sprints, so the team wanted to figure out the design approach. They wanted to figure out a effective approach in an agile project that would also satisfy the examiners. Object-oriented programming was the expected design of the program, so the team wanted to know the specifics of object orientation in Python. The new scrum guide was to be read to check if any changes made to Scrum in the newest guide should be added to the project. The overall sprint report can be seen in Figure 5.2, issues with a asterisk by the issue ID were added after the sprint has begun.

This plan resulted in the team revisiting the Alexa tutorial from last sprint and looking at others, and using the knowledge from those tutorials to create their own code. As well as continuing the Python learning – focusing on object-oriented programming and testing.

5.3.2 Sprint Contents

The first task the team worked on was figuring out the design approach. The team contacted module coordinator Neil Taylor and discussed the issue over email and an online meeting. The team wanted to know how in-depth the examiners expected the design process to be for an agile project. Typically, quick design techniques are needed when utilising agile due to fast iterations and work tasks often being broken down to a day or two of work. Neil acknowledged this point,

and he agreed that the utilisation of quick techniques like CRC (class responsibility collaborator) cards and design notes were enough during the completion of the work. Those notes could then be used to create UML or other diagrams to describe the planned design to the readers of the final report. As mentioned above, this was an important point to confirm because the team was about to start working on the tool's design during this sprint and wanted to know how they should approach that process.

The second topic of discussion that came out of the meeting with Neil Taylor was how to frame the work as user stories. This topic was not initially on the meeting's agenda but had been a lingering question on the team's mind, and Neil was kind enough to offer some advice. His advice was that the project was already following a good process and not to get too bogged down on stories vs tasks vs epics. However, he did advise that stories should be used if possible, instead of only tasks and epics.

The team worked on the airline data tutorial [49], discovered during the first sprint research. This tutorial provided insight into the NLP process, different classifiers, text analysis, performing cross-validation and others. However, it was relatively complex when compared to the Alexa tutorials [45] [46] that were completed in the last sprint, making it harder to follow and understand. During this tutorial, the team acquired exposure to the different libraries available for NLP and machine learning; Python object-oriented programming; how figures and graphs can be created with Python; preprocessing of data with RegEx and NLTK; the differences between stemming and lemmatisation, and others. The learning process was taking significant time and felt slow because all of these concepts were new to the author, including the programming language that was being utilised. The latter stages of the tutorial were not completed because they got too complex to be useful in producing the first code for the project's tool. So the team decided to start using the knowledge from this Airline and the Alexa tutorial to start developing code for the project's program.

The team decided to start with the preprocessing techniques because this was the first step in the tutorials after the data acquired and loaded. The team utilised CRC cards, hand-drawn class diagrams and other notes as design techniques. The team first thought that an object-oriented design might be the ideal strategy, similar to the one used in the Airline tutorial. However, after some thought, there was no need for an object-oriented design because there was no data that had to be stored alongside certain operations. Apart from the headlines, there was nothing that needed to be represented, which could instead be represented by a data frame or a list of strings. So, the decision was made to go with a functional/procedural design. It was later discovered that this was actually the recommended programming approach for many Python programs because it was often quicker than an object-oriented approach [50].

The plan was to create a function for each preprocessing operation, which took the text as input and returned the processed text as output. An overview of the whole design for the preprocessing functions was discussed in Section **??**. The preprocessing operations that were deemed to be required were converting the text to lowercase, removing punctuation, removing stop words and lemmatisation. The removal of stop words involves removing frequent words like 'the', 'me', 'I', and 'them', which are seen as noise because they do not contribute to the sentiment of the text. Lemmatisation is one of two preprocessing techniques used to convert words to their base form, e.g. converting 'running', 'runs', 'ran' to their collective base form run. These techniques allow the dataset to be reduced in size, removing noise due to all these words having the same or similar meaning. The other technique that converts words to their base form is stemming. Stemming

is a much simpler technique when compared to lemmatisation [51]. It produces the stems of words. Stemming does this by looking at the suffix of a word and removing common suffixes that indicate tense (e.g. ‘ed’), plurality (e.g. ‘s’, ‘es’), or other alternative forms of the word (e.g. ‘ing’). This process is very quick but produces words that are not part of the dictionary (e.g. ‘ponies’ would become ‘poni’) and is not as effective as lemmatisation. Stemming is less likely to convert all words to their base form. Lemmatisation is more sophisticated and informative and much slower due to its use of context and part of speech. It will always provide dictionary words known as lemmas, and can reduce more complex words like ‘was’ to its base form ‘be’. The choice between the two is essentially a trade-off between accuracy and speed. There are different types of stemmers available, differentiated by their accuracy and speed.

When the team began creating the necessary code that matched up with the design, they struggled with it at times. It was not behaving as expected. One specific example of the issues the team experienced was their inability to correct the error that would allow them to whitelist certain words from the NLTK stop words list. The team wanted to whitelist words like ‘not’, ‘don’t’, and other variations, which include not, because these words can often have a negative connotation, that would be lost if they removed. However, after the team could not correct this error, they decided to move on and use the unaltered NLTK stop word. During this part of the project, the team decided to gain some experience with the unittest Python module. They did this so that they could create tests for the code that was being written. They attempted to and succeeded in applying TDD to the code created. The testing allowed the team to see what was happening internally with the code and identify use cases and flaws the team would likely have not realised, at least not as easily or early as they did. Over the remaining days of the sprint, the preprocessing functions were refined. Several questions were considered during this process, including:

1. Does the text need to be lowercased before stop words can be removed?
2. Does punctuation need to be removed before the stop words are removed from the text?
3. How should apostrophes be handled? Should they be replaced like the other punctuation?
4. What should punctuation be replaced with a blank or space?

The text should be lowercased before stop word removal, because the stop words lists are all typically lowercased. Punctuation should also be removed before the stop word removal, but words with apostrophes were still included in the list, which brought into question the third point. The third and fourth questions were pondered for some time. The obvious character to replace any punctuation with was simply a blank – removing the punctuation from the text, but when it came to apostrophes, this approach would create non-dictionary words, e.g. ‘don’t’ would become ‘dont’. These non-dictionary words were not present in NLTK’s stop words list, which was printed out and examined. However, the team noticed that words like ‘don’ and ‘t’ were present in this list. These are the two words that would result from replacing the apostrophe in ‘don’t’ with a space. This observation made replacing all punctuation with space the simplest and most effective solution. Any double spaces in the text produced by this processing could be easily removed.

An issue identified in a later sprint, when the processed data was examined, resulted in acronyms with full stops being split up into singular characters due to the punctuation removal approach. For example, ‘U.K.’ would become ‘u k’ after lowercasing and punctuation removal. ‘U.S; would

become just ‘u’, after lowercasing, punctuation and stop word removal, because ‘s’ was registered as a stop word to remove contractions of ‘he is’ (he’s) or others.

Lemmatisation was chosen over stemming because it should provide better results, and the team thought the slower speed was not an issue. Now that the technique was determined, the team needed to pick a lemmatiser to use. There are no universal distinctions between lemmatisers like stemmers, which each have formal algorithms that differentiate them (e.g. Porter, Snowball). However, the corpus used and the implementation of the lemmatiser impact its performance. After looking over this article on the different ways to apply lemmatisation with an assortment of different libraries, [52]. The spaCy lemmatised was chosen because it seemed to provide the best results, based on this article. The spaCy lemmatiser was able to recognise pronouns and convert more words to their base form or lemma than the other options discussed in this article. The other main consideration was NLTK’s WordNet lemmatiser with part of speech (POS) tagging.

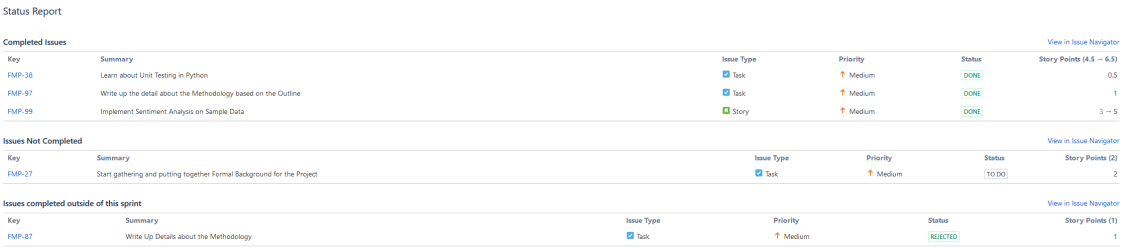
The lemmatisation process identified another problem: when the words were lowercased, the word ‘US’ was confused with the pronoun ‘us’, and spaCy’s lemmatiser replaced instances of lowercased ‘US’ with ‘-PRON-’, as it does with other pronouns. The team could not find a fix to this problem at the time, so they skipped the lemmatisation of the word ‘us’. However, later it was discovered that this was an issue with applying other preprocessing like lowercasing before lemmatisation. Not completing lemmatisation first causes issues because lemmatisation uses the context and part of speech tagging of the text, which starts disappearing when preprocessing is applied.

When these preprocessing functions were applied to the DJIA data, other issues came up, like the b characters that were present in the data, which were removed using another function. The presence of noisy data with spelling mistakes was also found due to the data being user-generated content from Reddit. This noise could possibly have been removed as part of the text cleaning stage of the NLP pipeline but would likely have required significant effort and not offered enough value, so the data was kept the same. Based on manual inspection, most of the data was of good quality, but some spelling and grammar issues were present.

The second half of the sprint included some write-up of the methodology used for the project. The team wanted to formalise this part of the project since the methodology was already established. Putting it all together in writing would allow the team to refer to it, find any gaps in the current approach, and complete a part of the final report. Writing also gave the team rest bite from the much more attention and thought intensive work of learning many new concepts and applying them in code while still making progress. Once this sprint was finished, the basics of the methodology were formed into a draft final report form.

5.3.3 Review and Retrospective

- Seven and a half story points were completed as part of this sprint if the ‘Rejected’ stories in the sprint report are not counted. This number can serve as a guide for future sprint planning events. It should be around how much the team can complete each sprint.
- The sprint plan was once again not very effective because many changes were made during the sprint, with new stories being added and others being rejected. The team planned to improve upon this in the upcoming sprints by extending the planning until after the supervisor meeting, allowing time for the team to digest the feedback provided during that meeting



Status Report

Completed Issues — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (4.5 → 6.5)
FMP-38	Learn about Unit Testing in Python	Task	Medium	DONE	0.5
FMP-97	Write up the detail about the Methodology based on the Outline	Task	Medium	DONE	1
FMP-99	Implement Sentiment Analysis on Sample Data	Story	Medium	DONE	3 → 5

Issues Not Completed — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (2)
FMP-27	Start gathering and putting together Formal Background for the Project	Task	Medium	TO DO	2

Issues completed outside of this sprint — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (1)
FMP-87	Write Up Details about the Methodology	Task	Medium	REJECTED	1

Figure 5.3: Sprint Three Report

and apply any changes before the sprint starts. The team were to try and conceive a more concrete focus for future sprints to reduce the need to make so many changes during the sprint.

- The plan for the upcoming sprints should be on creating more code for the project tool.
- Starting parts of the report worked out well. It allowed the team to work on something else when they came across errors that they needed a break from or ran out of attention/energy to work on the code. The supervisor agreed that getting some parts of the report completed at this stage was a good idea. This approach would be good to continue if it can fit in alongside the main focus.
- Scrum has been well utilised so far in the project, and been very useful in structuring the work

5.4 Sprint Three

5.4.1 Planning and Overview

> "Implementing sentiment analysis and gaining more background into the project area (NLP, Machine Learning, Sentiment Analysis)"

The plan for the sprint was to begin using sentiment analysis on the DJIA dataset, continue gaining NLP background and finish off the write-up of the methodology. The project supervisor recommended that the team looks at some basic NLTK sentiment analysis tutorials. This feedback was incorporated into the sprint's plan as part of implementing sentiment analysis.

5.4.2 Contents

The first part of the research into using NLTK for sentiment analysis led the team to start reading the NLTK book, which gave them insight into some of NTLK's capabilities, definitions and insight into NLP like tokens, collocations, bi-grams, and the tokenisers available in NLTK. This information was helpful but not really what the team needed. They could not find anything directly related to sentiment analysis. Next, the team also looked at another book, 'Practical Natural Language Processing' [53], which has been mentioned extensively in Chapter 1. 'Practical Natural Language Processing' provided insight into the language and the different types of approaches to

NLP: Heuristic, Machine Learning and Deep Learning. This content was again not directly related to the goal of the sprint sentiment analysis. However, it provided some excellent information, so the team decided to read the start of this book.

Then the team started looking at some sentiment analysis tutorials again, like the supervisor recommended. The team completed a Digital Ocean tutorial [54]. This tutorial only utilised NLTK for the whole NLP pipeline 1.3.4. All other tutorials read or completed before this one had all used Scikit Learn or another library for modelling (classification). The tutorial was useful and helpful when working on the project's tool, especially the usage of the WordNet lemmatiser with POS tagging, which was later used in the project. It also helped the team understand how NLTK classifiers can be used, provided examples of how to word density and other analytics can be extracted from the data, helped the team understand the Python properties `__name__` `__main__` and others. The tutorial also helped explain why context and other important parts of the text are removed during preprocessing. Even though context provides meaning, it is hard to process. Other tutorials were also looked at, but they did not prove very helpful, so the team did not spend too much time on them. The team had learned quite a lot from the tutorial and reading they had done so far, but it was hard to see how it would apply to their own dataset since it had no sentiment labels.

The team continued with finishing off the basic write-up of the methodology. There was no Jira issue covering the following topic, but the team spent some time thinking and reading about aspects of productive Python application like the package structure, testing, continuous integration build, and other aspects. The reading was done based on 'The Hitchhiker's Guide to Python', a book available online [55]. This investigation resulted in issues being created for future sprints to create a productive environment based on that research.

The team had a look at other work on the DJIA dataset, they had a look at three Kaggle notebooks. This research gave them insight into how the problem could be approached, with a basic bag of words approach and using another classifier to extract the sentiment from the data.

5.4.3 Review and Retrospective

- Everything was done, apart from part of the background research.
- The velocity was less than last sprint, six and a half story points versus seven and a half points.
- Made good progress, met sprint goal, just not sure what to do in the upcoming sprint.
- We need to figure out the code for stock market prediction in the next sprint.

5.5 Sprint Four

5.5.1 Planning and Overview

> "Start implementing Sentiment Analysis and Price Prediction using Stock Market Data."

Completed Issues — View in issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (6.5)
FMP-27	Start gathering and putting together Formal Background for the Project	Task	Medium	DONE	2
FMP-98	Apply Sentiment Analysis to Stock Market Specific Dataset	Story	Medium	DONE	3
FMP-120	Correct Methodology in terms of XP	Story	Medium	DONE	0.5
FMP-143 *	Read the rest of Chapter 1 of Practical NLP	Task	Medium	DONE	1

Issues Removed From Sprint — View in issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (2)
FMP-126	Set-up Productive Enviroment	Task	Medium	TO DO	2

Figure 5.4: Sprint Four Report

As the sprint goal indicates, the focus was on applying sentiment analysis to stock market specific data. However, the resulting experiment was a bag of words approach. It was not sentiment analysis. It investigated how certain words impacted the stock's price, but not how the sentiment in those headlines affected the same price. The team also wanted to put together some formal background for the project and make a correction in the report's methodology section. They also ended up doing some reading of 'Practical Natural Language Processing' because they had time to spare.

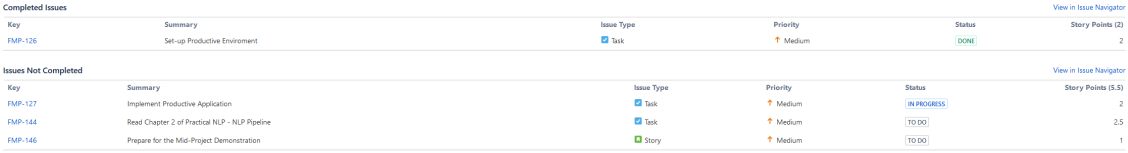
5.5.2 Contents

The team started to try and apply classification to DJIA data, using the other work completed on the DJIA dataset as inspiration. They were torn between whether to use NLTK or Scikit Learn for the modelling stage. They thought NLTK was the better choice because it was simpler to understand and more straightforward. In NLTK, there was a clear separation between the preprocessing, feature extraction and modelling, and the classes and functions, in general, had a more straightforward interface. It would also allow the team to use NLTK for the whole pipeline, making the code simpler. Ultimately the team started with Scikit Learn because it was the library used in most of the other work on the DJIA dataset and many tutorials. They ended up sticking with and using the library for part of the preprocessing alongside NLTK, feature extraction (vectorisation) and modelling. Scikit Learn was more complex and not clearly distinguish between preprocessing and feature extraction. Both were completed as part of the feature extractor named `CountVectorizer`, which applied to preprocess and also converted the text into a bag of words model. However, the `CountVectorizer` could not perform lemmatisation or stemming, so NLTK had to be used for that, and this created a bit of a messy pipeline. Part of the preprocessing was completed before the `CountVectorizer` was run, and another part was completed with the `CountVectorizer`, which led to certain steps being repeated. It made everything more complex and frustrating.

The team spent significant time figuring out how to use the libraries and how to create an effective pipeline. They also researched the specifics of the different classifiers: Multinomial Naive Bayes and Logistic Regression so they could understand how they each worked.

In the end, they applied several preprocessing steps and different combinations of these steps, which provided them with the results shown and discussed in Section 4.2.1.

5.6 Sprint Five



Completed Issues — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (2)
FMP-126	Set-up Productive Enviroment	Task	Medium	DONE	2

Issues Not Completed — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (5.5)
FMP-127	Implement Productive Application	Task	Medium	IN PROGRESS	2
FMP-144	Read Chapter 2 of Practical NLP - NLP Pipeline	Task	Medium	TO DO	2.5
FMP-146	Prepare for the Mid-Project Demonstration	Story	Medium	TO DO	1

Figure 5.5: Sprint Five Report

5.6.1 Planning and Overview

> "Implement the Productive Application"

The defined by the sprint goal above, the plan was to start creating a productive application that includes preprocessing, sentiment analysis, and price prediction. The team also wanted to continue background reading to ensure the program was high quality.

5.6.2 Contents

The team was already stressed from the progress on the project, it felt slow, and after my meeting with the project supervisor. This stress got worse since the supervisor was disappointed in progress and stressed that more 'results' were needed. This stress overwhelmed the team, and they became very demotivated, so most of the work planned did not get completed in this sprint. Unfortunately, after this point, the team really struggled with motivation for the project, probably due to the struggles with the project, exaggerated with the working situation being stuck at home in Northern Ireland, not even being able to be in Aberystwyth, where the team worked much better last semester. It was also becoming evident that the team was struggling to balance the learning and implementation of the project. The learning process was also appearing to be ineffective and too slow. The team felt like they were putting in lots of effort but were still failing to make good progress.

The team did get one task complete in this sprint, 'FMP-126 Set-up Productive Environment', which involved creating the basic folder structure, adding dependency management, and adding a continuous integration (CI) build, which checks the code style and runs the program's tests. The basic source structure included a 'main' package for all the non-test code and a 'test' package for all the tests. The root of the project was to contain a requirements.txt, scripts, and other dependency management files. TravisCI was chosen as the CI tool because of its simplicity and easy set-up, as mentioned in Section 2.1.2. The first configuration for the CI build can be seen below:

```
language: python
python:
  - "3.9"
install:
  - pip install -r source/stockPrediction/requirements.txt
script:
  - python -m unittest discover source/stockPrediction/test
  - pylint source/stockPrediction/main
  - pylint source/stockPrediction/test
```

The configuration specifies the language and version to use for the build. The lines under 'install' are setup commands that are run first before the commands under 'script'. The command `pip install -r source/stockPrediction/requirements.txt` installs all the dependencies required to run the code. The script commands run the tests and a linter that check for style issues in the test and main code.

Pipenv was used for the dependency management, and the requirements.txt mentioned in the install command in the TravisCI configuration was generated from the pipenv 'Pipfile'. The first versions of the Pipfile and requirements.txt that can be seen below, they only include Python and Pylint:

Pipfile:

```
[[source]]

 url = "https://pypi.org/simple"

 verify_ssl = true

 name = "pypi"
 [packages]
 pylint = "*"

 [dev-packages]
 [requires]
python_version = "3.9"
```

requirements.txt:

```
pylint
```

Requirements.txt is simply a text file that includes the needed dependencies, which can then be installed using pip. It is conventional to include a requirements.txt file in a Python project, but it could be excluded because the dependencies can also be installed using pipenv directly because the Pipfile already provides the necessary dependencies.

The CI builds were set up to run on branches and pull requests, and the GitHub settings were changed to prevent the merging of pull requests if the build failed. The screenshot of the TravisCI UI in Figure 5.6. Provides an overview of the latest builds on branches, showing the last six builds on that branch and the current status of the branch build.

When the team started adding code to the project, it encountered some problems with the CI build. The tests were no longer running properly, and NLTK data was missing.

The test were no longer running due to due to the following errors: `ModuleNotFoundError` and `ValueError`. These errors indicated problems with resolving the paths to the packages that needed to be imported. So a file called context.py was added to try and fix the issue. The file was meant to resolve the absolute path to the main package before importing it but did not fix it. Several other things were tried until the test command itself was updated, which fixed the issue. The script command `python -m unittest discover source/stockPrediction/test` was updated to `python -m unittest discover -s source/stockPrediction/test -t source/stockPrediction`, which specified the directories to run the tests from and where to find the main sources. This change fixed the package import issues.

Due to the CI environment being built from scratch each time, the NLTK data present on the team's

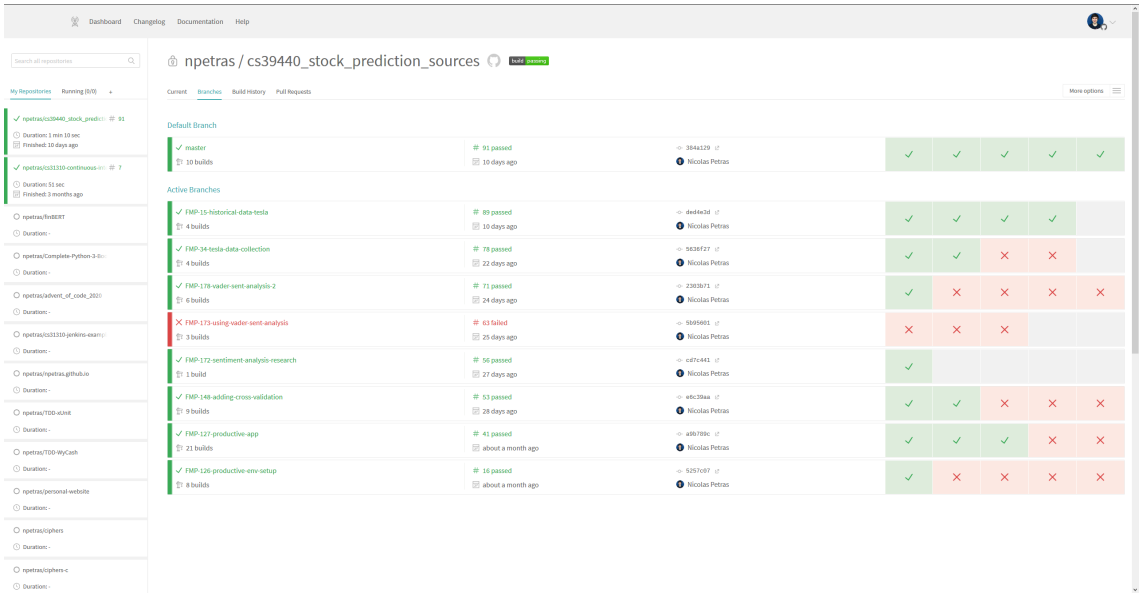


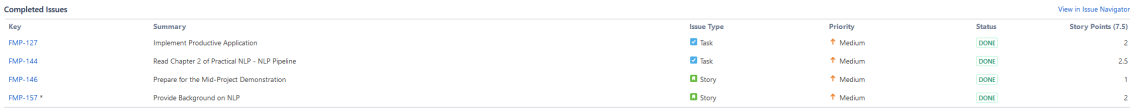
Figure 5.6: A screenshot showing the Travis CI UI

machine were missing from the CI environment. The first solution that was conceived was downloading the data in the Python code itself, although this seemed like a crude solution that felt unnatural. It might cause a performance issue because the download command is run each time the code is run. Since NLTK recommended approach was to install the data using the Python interpreter. The Python command `python -c "import nltk; nltk.download('wordnet'); nltk.download('averaged_perceptron_tagger')"` was added to the TravisCI configuration in the 'install' (set-up) portion. The updated configuration can be seen below. This Python command executes the code in quotes directly. This direct execution is enabled by the `-c` parameter.

```
language: Python
python:
  - "3.9"
install:
  - pip install -r source/stockPrediction/requirements.txt
  - python -c "import nltk; nltk.download('wordnet'); nltk.download('
    averaged_perceptron_tagger')
script:
  - python -m unittest discover -s source/stockPrediction/test -t source/
    stockPrediction
  - pylint source/stockPrediction/main
 - pylint source/stockPrediction/test
```

These issues identified the setup process for the project's markers, so the team made sure to include documentation on these points. Without the CI build, the team was unlikely to have identified these issues, especially the NLTK data one, because they installed the data at the start of the project and never had to think about it again.

The team also began work on the productive application, but the work was not finished as part of this sprint. The team began with creating the design for the productive application. Furthermore, they also started integrating the preprocessing and stock prediction code created in the last sprint. This code was all integrated into the productive application and adding some modules for data



Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (7.5)
FMP-127	Implement Productive Application	Task	Medium	DONE	2
FMP-144	Read Chapter 2 of Practical NLP - NLP Pipeline	Task	Medium	DONE	2.5
FMP-146	Prepare for the Mid-Project Demonstration	Story	Medium	DONE	1
FMP-157 *	Provide Background on NLP	Story	Medium	DONE	2

Figure 5.7: Sprint Six Report

manipulation and loading.

Repository documentation and code documentation was added throughout this sprint, and any style issues identified by pylint were fixed, which required significant effort.

5.6.3 Review and Retrospective

- We need to refocus in the next sprint
- We want to continue with work on the report in the next couple of sprints

5.7 Sprint Six

5.7.1 Planning and Overview

> "Establish Productive Application, and Complete Mid-Project Demo."

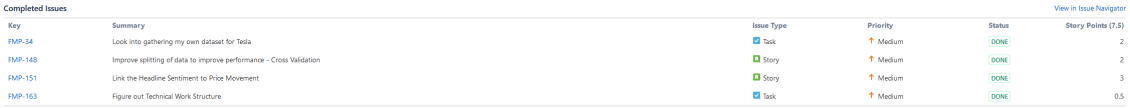
The team planned to continue working on the productive application, preparing for and completing the mid-project demonstration, and continuing reading. The story points were kept low during the planning since they were unsure how much they could get done because they struggled the last sprint. Only five and a half story points were planned for this sprint, even though the team typically completed six and a half to seven a half story points per sprint. The number of story points was, in fact, low, and a story to start writing up the background on NLP was added near the end of the sprint, as indicated by the asterisk in the sprint report in Figure 5.7.

5.7.2 Contents

The team spent the first days of the sprint creating slides in preparation for the mid-project demonstration. The first productive version of the application was completed before the demonstration – all of the spike code created in the previous two sprints was converted to a productive application. The duplication was removed from the spikes, and structure was added to the application, and everything was tied together in one application. The sentiment analysis experiments were run in full from sprint four.

The team is not sure if this is the correct approach, but its the best they could produce, scripts were added in root directory of the project to use the productive module code to run experiments.

The experiments were executed from a script in the root directory of the project. In other compiled languages like C, Java and Kotlin, this code would usually be placed in the main function within an application class/file, but that is discouraged in Python based on the team's research. The scripts



Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (7.5)
FMP-34	Look into gathering my own dataset for Tesla	Task	Medium	DONE	2
FMP-148	Improve splitting of data to improve performance - Cross Validation	Story	Medium	DONE	2
FMP-151	Link the Headline Sentiment to Price Movement	Story	Medium	DONE	3
FMP-163	Figure out Technical Work Structure	Task	Medium	DONE	0.5

Figure 5.8: Sprint Seven Report

were placed at the root of the project, because scripts should not be and are difficult to run from inside a package [56].

The team completed the mid-project demonstration and got feedback to add more advanced classifiers like Support Vector Machines (SVMs) and evaluate the project's work against existing work in the area. Unfortunately the team was not able to implement this feedback in the project.

As mentioned in the planning, the team read chapter two of 'Practical Natural Language Processing', which covers the NLP Pipeline. The team utilised this and previous reading to start writing up Section 1.3 of the report.

5.7.2.1 Review and Retrospective

- The productivity improved this sprint.
- The mid-project demonstration went well – had a good presentation and discussion with the second marker.
- The productive application was finished in this sprint, and the report is starting to take shape.

5.8 Sprint Seven

5.8.1 Planning and Overview

> "Improve DJIA Price Prediction through Sentiment Analysis and Price Prediction, and look to start gathering my own data."

The initial approach epic was closed, and a new epic was created to cover all the issues related to improving the DJIA price prediction. The 'Tesla Prediction' epic was included in the first sprint, and a new epic covering the write-up of the background for the project was created.

In this sprint, these improvements to the DJIA price prediction covered adding sentiment analysis, going beyond a bag of words, and improving the modelling – instead of a hold-out split, the team planned to add cross-validation.

The team also wanted to spend some time figuring out the structure of the technical work in the report because the team was unsure how they wanted to structure it.

5.8.2 Contents

The team started by trying to figure out how to add cross-validation in the most effective and easiest way with the Scikit Learn library. So they looked at some articles to gain some knowledge on the topic [57] [58]. The first method mentioned in the Scikit Learning documentation [57] that used `cross_val_core()` seemed to be the best approach. The approach was implemented, and it was easier to integrate than the team expected, but were no longer able to print the top positive and negative features like before. After implementing the approach, it revealed issues like not being able to apply the necessary preprocessing to the data, so the second more advanced cross-validation approach `cross_validate()` [59] provided by Scikit Learn was used. This approach also gave the team the option to print out the top features, but since the team could not implement it quickly, they decided to exclude it for now. However, this approach had an extra benefit that allowed the team to add various evaluation (or scoring) metrics easily. Apart from test accuracy, F1 score and training accuracy were added, and the standard deviation for all those scores between the different (cross-validation) folds. The results from this experiment are showcased in Section 4.2.2.1.

The use of cross-validation provided much more consistent and expected results. The overall accuracy also improves slightly. The quirks in the previous experiments: a large gap between the data stemming and lemmatisation applied, stop words and frequency removal have an inconsistent effect on accuracy were eliminated. However, there were slightly quirky results for the LogisticRegression classifier. Stemming only and lemmatisation only performed better than stemming or lemmatisation and with stop word and frequency removal. However, stop word removal, and frequency removal did not have a large negative impact on accuracy.

The team researched sentiment mining from the text because the stock market data does not have any sentiment labels. The labels instead correspond to price movement (positive or negative), and the team wanted to utilise the sentiment of the headlines to predict the price movement of stocks. VADER was one option that was previously experimented with, but the team could not get it working last time. The features needed to convert somehow to the format desired by the Scikit Learn estimators. After reading an article on Scikit Learn's website, the team found out that all of the library's estimators require a two-dimensional array as input. This array should be a vectorised form of the features. They had a look at the vectorisation (feature extraction) page [60] on Scikit Learn and found the `DictVectorizer`, which would probably enable them to add sentiment scores as features. Their other research found that there was a dictionary named SentiWordNet and sentiment analyser name FinBERT [61], specialised in financial sentiment. The SentiWordNet dictionary could be used to iterate over text and assign each word a sentiment value: positive, negative or neutral. FinBERT is a pre-trained classifier for sentiment analysis and orientation, it is trained using a transformer-based machine learning technique known as BERT (Bidirectional Encoder Representations).

The team wanted to add logging, but the Python standard library logging system was unlike Java and Kotlin and much more complex than expected. So the idea was mostly abandoned, apart from some warning logs, which were kept.

One observation from the DJIA dataset was that most of the news headlines were either neutral or negative, so the resulting sentiment of most rows would be negative. This observation was made by first inspecting the data manually and the team gauging the sentiment of the headlines. The team also ran VADER on some sample headlines. In both cases, most headlines were negative or

neutral, with few clearly positive headlines. The dominance of negative headlines would result in an unbalanced feature set. This and other issues with the data like the spelling mistakes, lack of relevance to the actual stock market and others, made the team more confident that collecting their own data would lead to better results.

Two headlines that were clearly negative in stock market sentiment: 'EU Stocks plunge!' and 'Stocks collapse on global slowdown fears' were both seen as neutral by VADER. This result was interesting and not entirely surprising since VADER was trained on general social media data. FinBERT, which is specifically targeted at financial sentiment, would most likely not suffer from these issues, which should lead to much better results, particularly with financial headlines that the team planned to start collecting in this sprint.

The team looked at the following tutorial to help with the scraping of headlines [62]. The team attempted to start scraping data with BeautifulSoup and Python requests module, but it was unsuccessful. Firstly, the team could not get the data at all from the HTML. Based on the inspection of the web page using developer tools, the elements that the team was trying to find were not present. So the team started trying to collect headlines manually, but this was too time-consuming, and they started trying to scrape the headlines with a script again. After some investigation, the team tried the urllib module to make the HTTP requests instead of requests and were now able to get the HTML document that they examined in developer tools – the content and elements were now selectable. However, even with the elements being selectable, the team was only able to get the top headlines visible when the page was open. They needed some way to scroll the page, so the historical headlines could also be selected and scraped.

The team used VADER [63] to extract the sentiment for each row (day of headlines). They did this by running VADER on each individual headline, combining the scores for the row of headlines and dividing that score by the number of headlines (25). This approach was taken because VADER is designed for short pieces of text. When the team tried to use VADER for the DJIA data in a sprint four spike, the compound score for each row of headlines was around -0.90. These strange and overwhelming negative sentiment scores were likely a result of the text being too long for VADER to provide a reliable sentiment score. The team expected all of the days to combine into an overall negative sentiment, but there were some positive rows, but primarily negative around -0.3, and none to very few neutral rows, all based on manual inspection.

5.8.2.1 Review and Retrospective

- Productive sprint, even with some frustration with the Tesla data collection and cross-validation.
- We need to figure out how to scroll web pages using code so that we can scrape historical news headlines.

5.9 Sprint Eight

5.9.1 Planning and Overview

"Get the Background Chapter for the Project Completed"

The sprint was focused on getting a chunk of the project report completed: background, analysis and methodology chapters.

5.9.1.1 Contents

Most of the content for background, analysis and methodology chapters of this report were completed during this sprint.

5.9.1.2 Review and Retrospective

- Managed to complete the average seven-point story points, should use this as a guide, and not stretch beyond the seven story points, because the team has not managed to complete more than more than seven and a half story points in any sprint.
- We got a good bit of work done but lacked motivation near the end of the sprint.
- Able to work on technical work again next sprint, alongside the report.

5.10 Sprint Nine

5.10.1 Planning and Overview

> "Attempt to collect historical Tesla data and start technical chapters of the Report."

The team wanted to continue the technical work during this sprint: collecting historical Tesla data, which would allow the team to explore an asset that was part of the initial requirements. It would also allow them to get much higher quality data because it would be related to the financial sector directly and include a much more balanced dataset regarding positive and negative sentiment. A dataset with financial headlines would also allow the team to use FinBERT.

Alongside that technical work, the team still wanted to work on the report because the end of the project was looming, so they decided to start work on the technical section of the report.

Unfortunately, the lack of motivation from the end of the last sprint continued into this sprint, but the team did still get a good bit of work done, around the average seven-point five story points.

5.10.1.1 Contents

After performing some research regarding the Tesla data collection, the team discovered that they would need to use a different library named Selenium [64]. Selenium is a testing framework for webpages but enables scraping websites due to its browser functionality. It opens a real browser and can perform operations like scrolling, unlike BeautifulSoup, an HTML parser for static HTML obtained by other means. The team had to spend some time understanding the framework. They needed to learn how to select the correct elements and then store them in a file. After they were able to scrape the data, the next question was how to structure and group the data. The team needed

to group the headlines by dates so that each day's headlines could be used to predict the price of Tesla.

The data was scrapped from Barron's [65] because their website provided the best interface for scraping from all the websites that were researched. As mentioned previously, the team would have liked to scrape Yahoo News, but it has a complex web page that was not easy to scrape. The Wall Street Journal (WSJ) [66] was the leading candidate for any future scraping because its interface was similar to Barron's and is a highly respected media outlet in the financial industry. The only real difference between the interfaces is the use of a was button to reveal more headlines on the WSJ website versus the scroll wheel on Barron's. The WSJ interface provides headlines from the WSJ and its sister websites: Barron's and MarketWatch, which are all owned by the same company. So the WSJ website provided a wider variety of news sources, one of which is one of the most prominent and most respected in the financial industry and provided an interface very similar to Barron's.

The actual stock market price data also needed to be scraped and converted to labels. This data could easily just be downloaded from Yahoo Finance [67]. The team just specified the data range they required and could download that data as a CSV file. This data was converted to labels by the following algorithm:

```
for row in range(0, len(price_df.index)):
    date = price_df.iloc[row, 0]
    open_price = price_df.iloc[row, 1]
    close_price = price_df.iloc[row, 4]

    if((close_price - open_price) >= 0):
        labels_dict[date] = 1
    else:
        labels_dict[date] = 0
```

The dates in the headlines and labels data sets were used to merge them together into one file. The headline dates needed to be converted to the ISO format so the two sets of dates could be compared. The following algorithm and code was used for this merge:

Alongside this work, the team worked on writing up the first sprint and figuring out the structure for the technical part of the report.

5.10.1.2 Review and Retrospective

- We struggled with motivation again, not sure what can be done to improve upon this situation, but hopefully, moving back to Aberystwyth next week will help
- Even with the lack of motivation, the team still completed most of the planned work, apart from experimentation with FinBERT

5.11 Sprint Ten

5.11.1 Planning and Overview

"Continue Technical Write-up, and start experimenting with Tesla Data."

This was the final sprint of the project, and its initial plan was:

- To apply VADER sentiment analysis and price prediction to the Tesla data
- Write up the first five sprints
- Learn how to use FinBERT so that it could be used next sprint

However, the team only got task two completed during the one-week sprint due to motivation problems, and they also stopped using the Jira project scrum board and product backlog to manage and plan their daily work in the last couple of weeks of the project. They instead decided to focus on writing parts of the report without managing it in Jira anymore.

5.11.2 Contents

The team decided to apply the bag of words approach applied in sprint four (Section 5.5) to the Tesla dataset before using VADER but did not get any further due to a lack of motivation. The team focused on finishing the report instead because that became the top priority due to the limited time left in the project.

Chapter 6

Critical Evaluation

6.1 Project Goal and Requirements

I expected to complete two of the requirements outlined in the analysis section in a reasonable amount of depth. The two that were in focus after a few weeks were working on the DJIA dataset and working on Tesla and Bitcoin data. I worked both of these on, but not really in much depth. I would have liked to explore them in more depth, but I could not execute in terms of the sentiment analysis or feature engineering in much depth, partly because I did not know what to do and felt a bit lost. After a few weeks, I realised that exploring both novel ideas mentioned in Section 2.2.1 was not possible, and decide to focus on the Tesla and Bitcoin idea because the data would be higher quality and would not require sifting the data. I managed to collect Tesla data, but did not get to experiment with the data beyond apply a basic bag of words approach. Investigating if WallStreetBets and social media sentiment, which retail investors dominate, is a good predictor of the stock market would still be an exciting idea to explore.

I am happy with the initial list of requirements. They were great for the experience and ideas I had at that early stage. One change that might have helped was putting more focus on producing a fully-fledged software tool with a GUI. Still, I was unsure if that would be possible and wanted to focus on the machine learning aspect over any software engineering aspect. However, in hindsight would have been a good change, which would have given me a more explicit focus when I started getting lost in the middle of the project. I would have been able to utilise my software engineering experience to create a sophisticated application.

6.2 Process and Methodology

6.2.1 Scrum

Choosing an agile methodology over a plan-based approach was definitely the right choice. It allowed me to refine my plan as the project progressed, which was important because I only had a high-level idea of what I wanted to do but not really how to achieve that goal. I think a plan-based approach would have severely hindered this project. Out of all the agile methodologies available, I also think Scrum was the right choice. It provided the right level of structure and flexibility, as

mentioned in Section 3.1. Scrum helped provide me with a focus for each week, manage all of the work that needed to be completed, provide structure for the day to day work and helped me with planning, review and inspection.

I think I followed the methodology very well, apart from the utilisation of user stories and some early issues with the planning for the sprint. Stories could have been utilised better as part of epics, but I struggled with this part of the methodology because there was no real customer, and I was unfamiliar with the area.

I was able to utilise the methodology effectively through Jira. It provides all the necessary features and more through reports and integration with GitHub. The GitHub integration was advantageous when writing this report. I could easily check what commits, pull requests and branches were associated with a Jira issue.

6.2.2 Development Practices

Not all of the practices outlined in Section 3.3 were actually useful in the end, especially in a project like this, which was focused on experimentation and results. Instead of focusing on producing a polished software application, which would be the case in a software engineering project. Below is an overview of all the practices mentioned and an evaluation of their usefulness during this project:

- **Test-Driven Development** was helpful for the preprocessing, but when it got into the experiments, which utilised external libraries. Testing, in general, stopped being as useful because I would essential be testing the output of those external libraries. This is the reason why I did not add any integration tests. Integration tests would have been more suitable if I had gone beyond the experiments and created an application with a GUI.
- Using the **Conventional Commits** guidelines [41] and making sure to split up the commits into small specific changes made it so much easier to review the changes while working on the technical work and afterwards when this report was being written.
- **Continuous integration** was great for taking care of the reviews in an automatic manner, particularly the code style. However, since the experiments/scripts did not have any direct tests, they did not have the same usefulness they would have in a software engineering project. It was still advantageous and great for catching problems in the set-up process — running the app from scratch on a clean machine. Things like the NLTK data having to be downloaded was something I would have never noticed without the CI build and something the project's markers might encounter. They probably would not have much trouble with it, especially if they had Python experience. Still, it is good to ensure the installation and set-up process is complete for any software.
- I did not utilise **refactoring** as much as I would have liked because I just wanted to get results at the end of the project and did not prioritise clean code. However, the code should be refactored in the future.
- The **pull request** process was definitely valuable. It ensured the code that did not meet the definition of done never or rarely made it into the master branch. It also provided a formal event to inspect the changes made as part of a Jira issue, allowing me to correct any mistakes found.

- Alongside the yapf formatter [27], the Google **coding standards** [28] allowed me to use an effective style for the code without thinking about it too much. This practice helped me keep the code readable.
- I tried to design the code using the **system metaphor** practice (in a domain-driven design manner). I did this by basing the package structure of the application on the NLP pipeline stages described in Section 1.3.4. Furthermore, I tried to make the file, variable, and functions names simple and easy to understand. Without this practice, I do not think I would have taken the time to base the design on the generic NLP pipeline. The design driven by this practice makes it easier to understand for anyone familiar with NLP and provides a good separation of the different modules. Without this practice, I assume the package structure would be a lot more arbitrary and harder to understand.
- The simple design practice helped remind me to focus on simplicity. However, the application might require some refactoring to make it more simplified.

6.3 Choice of Technologies

6.3.1 Python

Python was the best language for the task in a technical sense. It was a great scripting language for the experiments and has a fantastic community and first-class libraries for machine learning and NLP. The language also had other advantages like its readability, conciseness, and programming was effortless and quick. It is also widely used in the industry and is very useful to know for any programming and software engineering work.

However, I had to get familiar with and learn the language during this project, increasing the overall learning load of the project. In hindsight, it may have been better to use a worse language for this project, but one that I was familiar like Java or Kotlin. I say this because I did not end up having enough time to get the results I wanted to achieve, and using a language that I was already familiar with, would have eliminated saved time in terms of learning. I am also familiar with using a Java/Kotlin based stack to create a micro-service architecture with backend and frontend services, which I could have utilised to bring the ideas mentioned in Section 6.7 to reality. It would have allowed me to harness my industry experience with Angular, Spring Boot, Kotlin/Java and Docker, which I gained during my industrial year. However, I do not know how a smaller community and less prestigious libraries would have affected the project.

6.4 Project Choice

This project may not have been as suitable for me as others because of my lack of prior knowledge in the practical application of machine learning and NLP. I thought I would be able to understand and learn about these two topics, but instead, I often just felt a bit lost. Maybe I should have taken a more software engineering approach to the project, utilising my experience in that area to alleviate my knowledge gaps in machine learning and NLP. Instead of just focusing on developing the project's machine learning, which I struggled to make progress with, and got me demotivated. I could have focused on creating a proper software tool, like the one described in Section 6.7,

which would have provided a better balance, and would have alleviated a lot of the frustration of 'hitting a wall' working on the machine learning and NLP aspects. If this happened, I could work on the user interface, architecture, and other elements. I did not take this approach because I wanted to focus on the machine learning and NLP aspect. After all, it is why I chose this project for those aspects. However, in hindsight, more focus on software engineering likely would have been a good idea.

I always knew I was taking a risk choosing this project over others, but I wanted to work on a research-based project that tackled a complex problem like this one.

6.5 Technical Work

6.5.1 General

During the technical work, I could have focused more on the feature engineering and sentiment analysis part of the project and took less time on the preprocessing stage. This change would have gotten me 'real' results earlier. The learning process should have likely been minimised or accelerate, and work on the actual experiments should have started earlier. For example, I should have taken the initiative in the first sprint to begin experimenting with my own preprocessing functions, and I likely should not have spent so much time in sprint three looking at NLTK tutorials.

I often did not know what to be working on and felt lost, as previously mentioned. I do not think there was much more that I could have done on that point. I tried to do as much research as possible to figure out how to approach the problem. Still, I could not find any publically available examples or tutorials on how to mine sentiment from data really hindered my progress. One thing that might have helped with the issue of being lost was if I started reading the Practical Natural Language Processing book earlier on. However, even that would not have helped me with mining sentiment but would have provided me with a more solid and complete background in NLP. I just lacked the NLP knowledge and did not manage to obtain it during the project.

The central part I struggled with was sentiment mining. I did not know how to approach this problem and could not find any guidance on how to approach it in my research. Apart from discovering some libraries that I could use, but no tutorials on sentiment mining, except for VADER.

By the time the mid-project demonstration came around, I had was really struggling to progress with the project. I was confused by my supervisor help. He tried to offer me some guidance but it was very general. I was attempting to meet my supervisor expectations but often just felt lost because what I thought was correct did not meet my supervisor's expectations.

6.6 Motivation

I had a hard time staying motivated, after a negative supervisor meeting around two to three weeks before the mid-project demonstration, just between sprint four and five. I was putting in lots of hours and effort but seemed to be making little progress. This situation impacted my motivation, and I started feeling overwhelmed and unsure of what to do next to make progress.

The COVID lockdown exacerbated the motivation issues and reduced my productivity, even compared to semester one, when I could be in Aberystwyth. Rather than home in Northern Ireland, where my productivity was reduced.

I tried my best to deal with these motivation issues by taking a break several times, fighting through and coming back to Aberystwyth, but none of these was very effective, apart from the move back to Aberystwyth. However, that move was late in the project, so did not have a large impact.

Even though it was a challenging situation, I need to improve in this area and become much more mentally strong.

6.7 Future Work and Ideas

Some ideas that I did not implement were already mentioned in Section 6.1. Still, if I had a few more weeks to work on this project, my focus would be making the code created during the experiments into a fully-fledged application with a GUI. The plan would be to create an application with a frontend (GUI), probably built with Angular or a Python GUI framework like Flask. Angular because it a technology with which I familiar. If time allowed for it, it would also be great to deploy the application with a microservice architecture utilising Docker. I already have substantial experience with Docker, Angular and this type of microservice architecture. So this would likely be possible within a few weeks, and I likely should have focused in this as part of this project, instead of purely focusing on the experiments. I create a rough sketch near the end of the project, illustrating the design for an initial prototype for the UI. See Figure 6.1 for the sketch.

I would also like to continue the Tesla experiments, improve the feature engineering, explore Bitcoin's price movement, and applying information extraction on the data. The preprocessing could also be improved. A custom preprocessing solution would offer some benefits, like stopping negative stop words and an abbreviation like the 'U.S' being removed. The team would also like to add more advanced classifiers like support vector machines (SVMs), as suggested in the mid-project demonstration.

6.8 Conclusion

Even though this evaluation has had a primarily negative tone, I am still happy with the project's outcome, even if the results were not what I hoped. I learned a lot about NLP, machine learning and the Python language, and I am happy with the results and outcomes under the circumstances. The methodology was very well utilised during the project and was very helpful. Design, testing and implementation were good in general terms. I also had an excellent development process and managed to utilise the variety of development practices mentioned in Section 6.2.2 to improve the quality of the code.

However, as mentioned above, I could have made several improvements like adding more software engineering, accelerating the learning, focusing more on the results rather than learning, and focusing on sentiment analysis and feature engineering earlier than I did. I also could have saved some time using a language with which I was already familiar.

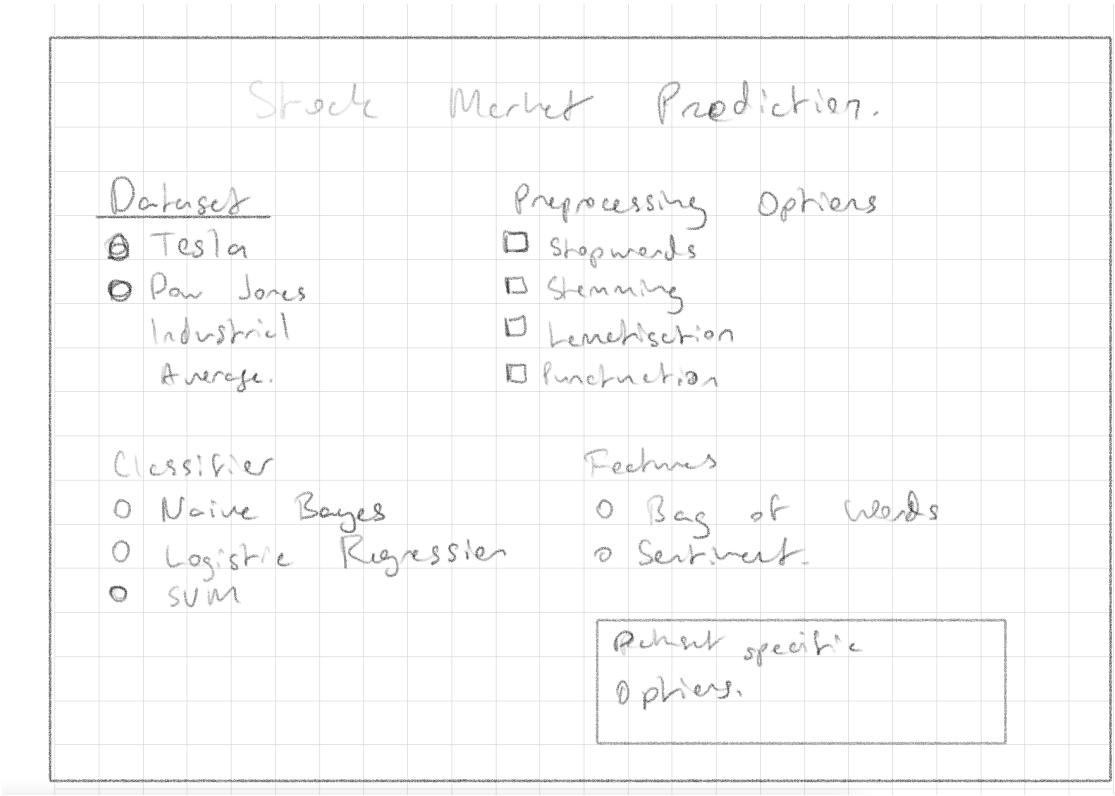


Figure 6.1: A sketch of what a basic UI could look like for the application

Annotated Bibliography

[1] J. Kuepper, "Volatility Definition," https://www.investopedia.com/terms/v/volatility.asp, March 2020, accessed 01/02/2020.

Provides a formal definition of volatility, as well as in-depth discussion and explanation of the concept in the financial markets.

[2] J. Chen, "Stock Market," https://www.investopedia.com/terms/s/stockmarket.asp, March 2021, accessed 25/03/2020.

A definition and description of the stock market from Investopedia. Part of this article is quoted in the report.

[3] A. Hayes, "Financial Markets," https://www.investopedia.com/terms/s/stockmarket.asp, February 2021, accessed 25/03/2020.

A definition and description of the financial markets from Investopedia.

[4] W. Kenton, "S&P 500 Index – Standard & Poor's 500 Index," https://www.investopedia.com/terms/s/sp500.asp, March 2021, accessed 25/03/2020.

A definition and description of the S&P 500 US index from Investopedia.

[5] E. Rosenbaum, "What Warren Buffett's losing battle against S&P 500 says about this market," https://www.cnbc.com/2021/01/08/how-warren-buffetts-uphill-battle-against-the-sp-500-is-changing.html, January 2021, accessed 25/03/2020.

This articles describes how Berkshire Hathaway, Warren Buffet's investment company, has performed against the US's premier index the S&P 500 in recent times, since 1999.

[6] A. Kiersz, "Here's how badly Warren Buffett has crushed the market," https://www.businessinsider.com/warren-buffett-vs-sp-500-2017-5?r=US&IR=T, May 2017, accessed 25/03/2020.

This articles describes how Berkshire Hathaway, Warren Buffet's investment company, has performed against the US's premier index the S&P 500 over Berkshire Hathway's lifetime, dating back to 1964.

[7] D. Harper, "Forces That Move Stock Prices," https://www.investopedia.com/articles/basics/04/100804.asp, November 2019, accessed 25/03/2020.

This articles describes the forces that move the stock market, but this can generally be applied to any financial market, even if it differs in the details.

[8] V. Sowmya, M. Bodhisattwa, G. Anuj, and H. Surana, *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP systems*. O'Reilly Media, 2020, pp. 29–31, NLP Tasks.

This section 'NLP Tasks' of the Practical Natural Language Processing book, provides an overview of the main fundamental NLP tasks that occur in most if not all NLP projects. Read pages 29 to 31 of the book for more insight into the NLP tasks than my report provides. This section was read by the author to gain background on NLP Tasks.

[9] ——, *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP systems*. O'Reilly Media, 2020, pp. 32–38, What is Language?

This section 'What is Language?' of the Practical Natural Language Processing book, provides an overview of the natural language, focusing on the building blocks that are important in NLP. Read pages 32 to 38 of the book for more insight into natural language than my report provided. This section was read by the author to gain background on Language structure.

[10] ——, *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP systems*. O'Reilly Media, 2020, pp. 71–126, NLP Pipeline.

Chapter 2: NLP Pipeline of the Practical Natural Language Processing book, provides an overview of the the typical NLP pipeline and the different stages of that pipeline. Read the book's second Chapter (pages 71 to 126) for more insight into the NLP Pipeline than my report provided. This section was read by the author to gain background on the NLP Pipeline.

[11] J. Sun, "Daily News for Stock Market Prediction," https://www.kaggle.com/aaron7sun/stocknews, August 2016, accessed 24/03/2020.

The first dataset that was used in the project. The dataset includes nearly eight years of world news headlines with a label that indicates whether the stock index went up, or decrease or did not change. This makes any classification binary and the index decrease or staying the same are merged into one label. The dataset was focused on the Dow Jones Industrial Average (DJIA), a US blue-chip stock market index. It covered the period from late 2008 to late 2016.

[12] V. Sowmya, M. Bodhisattwa, G. Anuj, and H. Surana, *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP systems*. O'Reilly Media, 2020, pp. 73–78, Data Acquistion.

This section 'Data Acquisition' of the Practical Natural Language Processing book, provides an overview of the different techniques that can be used to acquired data. Read bpages 73 to 78 for descriptions of the techniques back translation and bigram flipping that I mention in my report along with others,

[13] ——, *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP systems*. O'Reilly Media, 2020, pp. 130–177, Text Representation.

This Chapter 'Text Representation' of the Practical Natural Language Processing book, provides an overview of the different different methods of representing text in an efficient manner. Read pages 130 to 177 for descriptions of the techniques back translation and bigram flipping that I mention in my report along with others,

[14] D. Economics, "What is the best programming language for Machine Learning?" https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7, May 2017, accessed 25/03/2020.

This article describes the top languages used in the artificial intelligence and machine learning industry. The article goes into specific categories like sentiment, game artificial intelligence and more. Python is the dominant in most areas, specifically sentiment analysis – a core focus of this project.

[15] A. Zola, "The 5 Best Programming Languages for AI," https://www.springboard.com/blog/best-programming-language-for-ai/, November 2018, accessed 28/03/2020.

This articles provides a top five list of the best programming languages for artificial intelligence. Python and Java being the top two.

[16] "Git –fast-version-control," https://git-scm.com/, accessed 28/03/2020.

The online documentation and website for Git, the tool used for source control management.

[17] "GitHub," https://github.com/, accessed 28/03/2020.

The hosting service used to host the project's Git repository.

[18] "Sublime Merge," https://www.sublimemerge.com/, accessed 28/03/2020.

The Git GUI client used in this project.

[19] "Visual Studio Code," https://code.visualstudio.com/, accessed 28/03/2020.

The code editor used in this project.

[20] "PyCharm," https://www.jetbrains.com/pycharm/, accessed 28/03/2020.

The IDE used in this project.

[21] "Installation Anaconda," https://docs.anaconda.com/anaconda/install/, accessed 28/03/2020.

Installation of the Anaconda Individual Edition, used to download Python, Pip, Jupyer notebooks and other libraries.

[22] "Python," https://www.python.org/, accessed 28/03/2020.

The website and documentation for Python, this was the programming language used during this project.

[23] "Pip," https://pip.pypa.io/en/stable/, accessed 28/03/2020.

One of the package managers used for Python.

[24] "Conda," https://docs.conda.io/en/latest/, accessed 28/03/2020.

One of the package managers used for Python.

[25] "Pipenv," https://pypi.org/project/pipenv/, accessed 28/03/2020.

The dependency management tool used in the project to manage Python dependencies.

[26] "Pylint," https://pypi.org/project/pylint/, accessed 28/03/2020.

Pylint was used to check for style issues.

[27] "Google YAPF," https://github.com/google/yapf, accessed 28/03/2020.

The formatter used to automatically format Python code.

[28] "Google Python Style Guide," https://google.github.io/styleguide/pyguide.html, accessed 28/03/2020.

The style guide used in this project for Python.

[29] "TravisCI," https://travis-ci.com/, accessed 29/03/2020.

[30] M. Rehkopf, "Agile epics: definition, examples, and templates," https://www.atlassian.com/agile/project-management/epics, November 2019, accessed 08/02/2020.

The existing dataset that will be used to predict the DJIA.

[31] J. Sun, "Daily News for Stock Market Prediction," https://www.kaggle.com/aaron7sun/stocknews, November 2019, accessed 08/02/2020.

The existing dataset that will be used to predict the DJIA.

[32] A. Ganti, "Dow Jones Industrial Average (DJIA)," https://www.investopedia.com/terms/d/djia.asp, November 2020, accessed 04/02/2020.

Includes a summary as well as an in-depth explanation of the Dow Jones Industrial Average (Dow 30)

[33] "WallStreetBets," https://www.reddit.com/r/wallstreetbets/, accessed 19/03/2020.

The WallStreetBets sub-Reddit, the community's main platform. Investigating this community's impact on stocks was one idea that was considered at the start of the project.

[34] M. Rehkopf, "What is Wall Street Bets, the upstart Reddit group that is pummeling Wall Street?" https://uk.finance.yahoo.com/news/what-is-wall-street-bets-the-upstart-reddit-group-that-is-pummeling-wall-street-162734056.html, January 2021, accessed 29/03/2020.

An article that gives insight into the WallStreetBets community and their impact on the market.

[35] K. Schwaber and J. Sutherland, "The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game," https://www.scrumguides.org/index.html, November 2020.

This guide will be used as a basis for my project methodology, since this is not a team project the methodology described in this guide will need to be altered. The adjustments that are made will be mentioned in project report.

[36] Altexsoft, "Extreme Programming: Values, Principles, and Practices," https://www.altexsoft.com/blog/business/extreme-programming-values-principles-and-practices, January 2021, accessed 29/01/2020.

This online post describes the Extreme Programming methodology. I plan on utilise most of the practices mentioned in this article to add structure to the development of my work, and not just the planning and organisation of my work, provided by Scrum. Some practices that are target at teams like Pair Programming, Collective Ownership or are already present in Scrum i.e. The Planning Game will not be utilised.

[37] K. Beck, C. Andres, and E. Gamma, *Extreme Programming Explained: Embrace Change*, ser. XP series. Addison-Wesley, 2004, 9780321278654.

Overview of Extreme Programming

[38] Atlassian, "What is kanban?" https://www.atlassian.com/agile/kanban, accessed 16/03/2020.

Provides an overview of Kanban, including comparisons to the Scrum Framework.

[39] J. Shore, "The Art of Agile Development: Estimating," http://www.jamesshore.com/v2/books/aoad1/estimating, April 2010, accessed 10/02/2020.

Pre-release book excerpt about Estimation from James Shore's Art of Agile Development, Second Edition. This excerpt provided insight into estimation, and acted as a guide.

[40] M. Cohn, "Don't Equate Story Points to Hours," https://www.mountaingoatsoftware.com/blog/dont-equate-story-points-to-hours, accessed 16/03/2020.

[41] "Conventional Commits," https://www.conventionalcommits.org/en/v1.0.0/, accessed 04/04/2020.

Conventions used for commits in the project.

[42] A. Jagota, "Text Sentiment Analysis in NLP," https://towardsdatascience.com/text-sentiment-analysis-in-nlp-ce6baba6d466, July 2020, accessed 29/01/2020.

This article was read to gain background in Sentiment Analysis and NLP, I can use this article to help me implement my sentiment analysis.

[43] M. Learn, "Top Machine Learning Algorithms Explained: How Do They Work?" https://monkeylearn.com/blog/machine-learning-algorithms, accessed 27/04/2020.

Provided an overview of the different machine learning algorithms, and how they work. Some of which like Logistic Regression with which I was not already familiar.

[44] A. Jagota, "How to Perform Text Mining with Sentiment Analysis," https://monkeylearn.com/blog/text-mining-sentiment-analysis/, accessed 27/04/2020.

Provided a level overview on how to perform sentiment analysis.

[45] M. Kosaka, "Cleaning & Preprocessing Text Data for Sentiment Analysis," https://towardsdatascience.com/cleaning-preprocessing-text-data-for-sentiment-analysis-382a41f150d6, November 2020, accessed 09/02/2020.

The first part of a tutorial into NLP and Sentiment Analysis that focuses on pre-processing the data, getting it ready for sentiment analysis. The tutorial focuses on a dataset of Alexa Dot reviews. I have complete this whole tutorial as part of my research and spike work. The other parts of this tutorial and citation of the data set used are mentioned below.

[46] ——, "Topic Modeling and Sentiment Analysis on Amazon Alexa Reviews," https://towardsdatascience.com/topic-modeling-and-sentiment-analysis-on-amazon-alexa-reviews-81e5017294b1, December 2020, accessed 09/02/2020.

Second part of the Alexa NLP and Sentiment Analysis tutorial, this part focuses on topic modelling and sentiment analysis, specifically using VADER. I completed this tutorial as part of my research and spike work.

[47] J. Portilla, "2021 Complete Python Bootcamp From Zero to Hero in Python," https://www.udemy.com/course/complete-python-bootcamp/, accessed 27/04/2020.

The main resource used throughout the project to learn Python.

[48] "Beautiful Soup Documentation," https://www.crummy.com/software/BeautifulSoup/bs4/doc/, accessed 28/03/2020.

The online documentation and website for Beautiful Soup, the library was used for text cleaning stage of the NLP pipeline. This documentation was referenced when using the library.

[49] B. Carremans, "Sentiment Analysis with Text Mining," https://towardsdatascience.com/sentiment-analysis-with-text-mining-13dd2b33de27, August 2018, accessed 29/01/2020.

Part of this tutorial was completed during the project, it provide lots of insight.

[50] K. Reitz, "Structuring Your Project - Object-Oriented Design," https://python-docs.readthedocs.io/en/latest/writing/structure.html#object-oriented-programming, 2016, accessed 27/04/2020.

A page from the online version of the The Hitchhiker's Guide to Python! Book, discussing object-oriented design in Python, and how a functional design is often more efficient.

[51] A. Beri, "Stemming vs Lemmatization," https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221, May 2020, accessed 11/02/2020.

An article that explains stemming and lemmatisation in more depth.

[52] B. Carremans, "Python – Lemmatization Approaches with Examples," https://www.geeksforgeeks.org/python-lemmatization-approaches-with-examples/, September 2020, accessed 27/04/2020.

Comparison of different lemmatisers in Python.

[53] V. Sowmya, M. Bodhisattwa, G. Anuj, and H. Surana, *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP systems*. O'Reilly Media, 2020.

This is the main book I read during this project to gain an understanding of Natural Language Processing (NLP), and how to apply it.

[54] S. Daityari, "How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK)," https://www.digitalocean.com/community/tutorials/how-to-perform-sentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltk, September 2019, accessed 27/04/2020.

A tutorial completed as part of this project.

[55] K. Reitz, "The Hitchhiker's Guide to Python," https://docs.python-guide.org, 2016, accessed 27/04/2020.

[56] "Python – Lemmatization Approaches with Examples," https://stackoverflow.com/questions/24048687/running-python-scripts-within-a-subpackage-of-my-package, 2014, accessed 27/04/2020.

[57] "3.1. Cross-validation: evaluating estimator performance," https://scikit-learn.org/stable/modules/cross_validation.html, accessed 27/04/2020.

Guide on using Cross-Validation with Scikit Learn library.

[58] V. Dekanovsky, "Complete guide to Python's cross-validation with examples," https://towardsdatascience.com/complete-guide-to-pythons-cross-validation-with-examples-a9676b5cac12, May 2020, accessed 27/04/2020.

Describes Scikit Learn's cross validation process in more detail, and from a different angle than that of the official guide on Scikit Learn's website.

[59] "sklearn.model_selection.cross_validate," https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html#sklearn.model_selection.cross_validate, accessed 27/04/2020.

Scikit Learn's documentation on the cross_validate() function used for cross validation within the project.

[60] "6.2. Feature extraction," https://scikit-learn.org/stable/modules/feature_extraction.html, accessed 27/04/2020.

Guide to feature extraction (vectorisation) with Scikit Learn's library.

[61] C. Hutto and E. Gilbert, "Finbert: Financial sentiment analysis with pre-trained language models," 06 2019.

The paper on the FinBERT sentiment analysis classifier.

[62] M. F. Zafra, "Web Scraping news articles in Python," https://towardsdatascience.com/complete-guide-to-pythons-cross-validation-with-examples-a9676b5cac12, July 2019, accessed 27/04/2020.

[63] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," 01 2015.

The paper on the VADER sentiment analysis classifier.

[64] "How to load all entries in an infinite scroll at once to parse the HTML in python," https://stackoverflow.com/questions/21006940/how-to-load-all-entries-in-an-infinite-scroll-at-once-to-parse-the-html-in-pytho, accessed 28/04/2020.

StackOverflow post that pointed me to use Selenium to scrape web pages that require scrolling.

[65] "Tesla Inc." https://www.barrons.com/quote/stock/tsla, accessed 28/04/2020.

Webpage for Tesla on the Barron's website, the page used to scrape the Tesla data.

[66] "Tesla Inc." https://www.wsj.com/market-data/quotes/TSLA?mod=searchresults_companyquotes, accessed 28/04/2020.

Wall Street Journal web page on Tesla, that the teamF planned to scrape in the future to get headlines from WSJ, Barron's and MarketWatch.

[67] "Tesla Inc. Historical Data," https://finance.yahoo.com/quote/TSLA/history?p=TSLA, accessed 28/04/2020.

Yahoo Finance web page on Tesla that shows and allows user to download the historical stock price data for Tesla.

[68] A. Zola, "Natural Language Toolkit," https://www.nltk.org/, accessed 28/03/2020.

The NLTK library documentation. This library and it's documentation was used throughout the project.

[69] NLTK, "NLTK Source," https://github.com/nltk/nltk, accessed 28/03/2020.

The sources for the NLTK library on GitHub.

[70] B. Steven, K. Ewan, and L. Edward, "Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit," https://www.nltk.org/book/, accessed 28/03/2020.

The online book for NLTK, updated for NLTK 3 and Python 3. Part of this books was used for research, and it is being cited due to the use of the NLTK library.

[71] ——, *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP systems*. O'Reilly Media, 2009.

This book was not used during the work on this project, but was requested as a citation by the NLTK developers, if the library is used. It is the first book published around NLTK, but is now outdated.

[72] "Scikit-Learn Machine Learning in Python," https://scikit-learn.org/stable/index.html, accessed 28/03/2020.

The online docs and website for ScikitLearn. These documentation was used as part of my work.

[73] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

An article from the Journal of Machine Learning Research regarding ScikitLearn, that was requested as a citation for using the ScikitLearn library.

[74] "Pandas," https://pandas.pydata.org/, accessed 28/03/2020.

The online documentation and website for Pandas, the libary was used for reading in and manipulation the project's data. This documentation was referenced, when using the library.

[75] "Industrial Strength Natural Language Processing in Python," https://spacy.io/, accessed 28/03/2020.

The online documentation and website for spaCy, the library was used for some experimentation. This documentation was referenced when using the library.

[76] "Rule-based Matching," https://spacy.io/usage/rule-based-matching, accessed 28/03/2020.

The online documentation for spaCy rule-based matching, could have been used to some heuristic/rule-based modelling or features.

[77] "MMP Student," https://gitlab.dcs.aber.ac.uk/mmp/mmp-student/, accessed 28/03/2020.

The Latex template used as a basis for this report.

[78] J. Chen, "Blue Chip Definition," https://www.investopedia.com/terms/b/bluechip.asp, December 2020, accessed 01/02/2020.

Provides a formal and in-depth definition of a blue chip stock and company.

[79] R. Markets, "Which is Harder to Trade Forex or Stocks?" https://rockfortmarkets.com/en/which-is-harder-to-trade-forex-or-stocks/, April 2020, accessed 02/02/2020.

Makes the case that FOREX (currency) trading is easier than stock trading due to higher liquidity, longer market hours, and higher leverage

[80] "Time, Not Timing, Is What Matters," https://www.capitalgroup.com/individual/planning/investing-fundamentals/time-not-timing-is-what-matters.html, accessed 05/02/2020.

This article shows that the likelihood that an investment is positive over time gets more and more likely. Showing that over the last 100 years the market has been very predictable in one sense, it will go up over time, even if there are negative periods.

[81] J. Shore, "AoAD2 Practice: Stories," https://www.jamesshore.com/v2/books/aoad2/stories, January 2020, accessed 10/02/2021.

Pre-release book excerpt about Stories from James Shore's Art of Agile Development, Second Edition. I used this excerpt to guide me on how to create and amange stories. I attempted to make them customer centric, but did also use Tasks to specific the technical details, and workd needed to complete a story or customer requirement. Also, I do not conform to all the ideas in this excerpt (e.g. I consider bugs as separate from stories, and do not size them).

[82] "," https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html, 0, accessed 12/02/2020.

Provides an overview of all the POS tags.

[83] "Tesla Inc. Historical Data," https://www.selenium.dev/documentation/en/, accessed 28/04/2020.

Yahoo Finance web page on Tesla that shows and allows user to download the historical stock price data for Tesla.

Appendices

Appendix A

Third-Party Code and Libraries

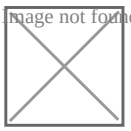
All third-party libraries were used without modification, and have open-source licenses:

- NLTK (Natural Language Toolkit) [68]
- ScikitLearn (sklearn) [72]
- Pandas [74]
- BeautifulSoup [48]
- Selenium [83]

Appendix B

Ethics Submission

Assessment ID (reference): 18784



18/02/2021

For your information, please find below a copy of your recently completed online ethics assessment

Next steps

Please refer to the email accompanying this attachment for details on the correct ethical approval route for this project. You should also review the content below for any ethical issues which have been flagged for your attention

Staff research - if you have completed this assessment for a grant application, you are not required to obtain approval until you have received confirmation that the grant has been awarded.

Please remember that collection must not commence until approval has been confirmed.

In case of any further queries, please visit www.aber.ac.uk/ethics or contact ethics@aber.ac.uk quoting reference number **18784**.

Assesment Details

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
nip19@aber.ac.uk

Full Name
Nicolas Petras

Please enter the name of the person responsible for reviewing your assessment.
Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department

cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
Predicting Stock Market Trends from News Headlines

Proposed Start Date
25 Janauary 2021

Proposed Completion Date
1 June 2021

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Does your research involve human participants?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
I am going to try and predict the price movement of various assets in the financial markets, by using sentiment analysis on news headlines and maybe also social media posts. The sentiment analysis will be done using machine learning.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?
Yes

Does the research pose more than minimal and predictable risk to the researcher?
Not applicable

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

Please include any further relevant information for this section here:

Is your research study related to COVID-19?

No

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

Please include any further relevant information for this section here: