Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

An intelligent sentiment proxy analysis system for estimating price changes in financial markets

Michael McGuinness

April 27, 2021

Supervisor: Prof. Khurshid Ahmad

A Final Year Project submitted in partial fulfilment
of the requirements for the degree of
B.A. (Mod.) Integrated Computer Science

Declaration

I hereby declare that this Final Year Project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: _____ Date: _____

Abstract

TODO - 400 words

Acknowledgements

TODO - thank who you gotta thank

Contents

List of Figures

List of Tables

Listings

Nomenclature and Definitions

TODO

1 Introduction

This chapter outlines the motivations behind the project and presents the objectives for the research made in this project.

1.1 Objective

The title of this project is An intelligent sentiment proxy analysis system for estimating price changes in financial markets. The first part of this title is 'An intelligent sentiment analysis system'. This indicates that some sort of system must be created which analyses sentiment proxy intelligently. This means that there must be an emphasis put within this project on the understanding sentiment proxy, and use this understanding to analyse sets of data which involve sentiment. This is the followed by 'for estimating price changes in financial markets'. This indicates that the previous understanding of sentiment proxies should be used to understand price changes in the market. This requires an understanding of the relationship between sentiment proxy and price changes. This must be heavily emphasised within the project. This understanding of the relationship may then be used to develop a model for estimating these changes once causation between these segments is understood. This assumes there is causation however. These items laid out form one overarching question: What is the relationship between sentiment and prices changes? This question can be broken down into two segments.

1. How do we start understand the relationship between sentiment and price changes? Sentiment is an inherently qualitative form of data and price changes are an inherently quantitative form of data. Therefore how is qualitative data used in a quantitative setting?
2. Once we can use sentiment in the same system as price changes, what is the relationship between them? Is there a causal link between the two items or are they even correlated?

1.2 Report Structure

Chapter 2 outlines the background required to understand some of the decisions made in order to answer the posed questions. As well as this it contains some prerequisite knowledge required to understand some of the concepts within the report itself.

Chapter 3 outlines the design choices made within the creation of the system that was built in order to explore the questions that are put forward in this project.

Chapter 4 outlines the details of implementing the system laid out in chapter 4 in a manner that is useful.

Chapter 5 outlines the results provided by the implemented system in reference to the questions being explored, as well as exploring what these results might mean.

Chapter 6 reflects upon the work done within the project as well as how it may be enhanced.

2 Background

TODO what? why? how? should consist of a description of the background to the project, e.g. motivation, state-of-the-art. In some cases it may be necessary to split this review into two chapters. This information is absolutely essential since the project must be viewed in context and the relevance of the work must be clearly stated. It is not a technical exercise done in isolation.

2.1 more subsections and subsubsections

TODO what? why? how?

2.2 Background Summary

TODO what? why? how?

3 Design

This chapter discusses the overall design of the project, its architecture and the individual components.

3.1 Brief

The project is 'An intelligent sentiment proxy analysis system for estimating price changes in financial markets'. It aims to understand the relationship between sentiment proxy and the market through creating a tool which aids in the analysis of sentiment proxy and market price data. Secondarily, this information is used to create estimations of the next day's returns for a given stock. The data-sets required and organising of said data-sets was a very important part of the project, which had to be executed with a high regard for precision and speed. The project revolved around the sentiment proxy and market price data-sets.

3.2 Requirements Gathering for System Design

Gathering the correct requirements for the system was an essential part of development. It is important to understand the limits of such a project, one important one being time, as well as understand how similar projects have attempted solving the problems that may come up. For this reason the reading of various research papers which attempted to perform a similar task was essential. These helped give a guideline of what the project structure was going to look like. The design was considered in relation to the requirements of this project, however, these papers helped expedite some design choices. The final design required understanding what information is required as an input for the system, and how it is structured. Then it is important to understand what the outputs of the system are. This then allows the conceptualisation of a pipeline which will achieve the required results. Finally, the pipeline must be refined and broken down into it's various parts. This allows a comprehensive and understandable design for the project.

3.2.1 Obtaining & Understanding Data-sets

The data-set requirements for the project are stock prices and sentiment proxies, both over time, and a dictionary containing words and their corresponding sentiment attributes. In order to fulfil these requirements three different source were required.

Price Source

One source is one which allows the gathering of stock prices. The requirements for this source are that it:

- allows the extraction of prices for a specific company
- allows the extraction of daily prices over a large period of time
- gives pieces of information beyond closing prices such as the volume of trades on that day

The source chosen is IEX Cloud.

IEX Cloud IEX Cloud is a financial data infrastructure platform that connects developers and financial data creators. It provides an API which allows access to a large amount of data centred around stock prices, including minute by minute prices for a given stock. The particular endpoint that was required for this project returns historical prices. This endpoint was perfect as it allowed the return of up to 5 years of data-points within the free tier. If there were expansion to be made it would even allow the return of the entire lifetime of a stock if the tier were upgraded. A few more points in favour of the use of IEX Cloud are that it is available at any time of day any day of the week, and it it is simple to use with good customer support in case of issues.

Sentiment Proxy Source

Another source is one which allows the gathering of sentiment proxy data. The requirements for this source are that:

- allows the extraction of sentiment proxy for a specific company
- allows the extraction of sentiment proxy over a large period of time
- allows extraction of sentiment proxies in a batch manner
- gives differing kinds sources, for example newspapers, academic articles

The sources explored are LexisNexis and Proquest, with the final decision having been LexisNexis.

Proquest Proquest provides access to many different databases containing licensed scholarly journals, newspapers, wire feeds, reports, etc. Using a trinity account, access is allowed 30 of these databases. Some of the databases included are:

- European Newsstream
- ProQuest Historical Newspapers: The New York Times with Index
- ABI/INFORM Global

For a comprehensive and up to date list of databases, it can be looked up on the website itself with trinity credentials. Proquest even allows the selection of specific databases to be searched. Allowing for more specificity in data-sources. Depending on the database articles can go back a very long

time, especially with the Historical Newspapers sources. Proquest can search for a specific company then allows the download of up to 50 articles at a time.

LexisNexis LexisNexis is a research tool for news, companies and markets insights, multiple legal practice areas, and business and science biographies. For the purposes of this project, it has an extensive, reliable and quite importantly licensed library with hundreds of different sources, containing many kinds of articles including newspapers, magazines and journals. An interesting point to make is that it is recommended by the Californian Supreme Court and published Court of Appeal opinions in the US as an accurate, authentic, up-to-date, and reliable source for citing and quoting. It allows for a very detailed search of it's sources, as in it allows to search by type of source as well as allowing the use of Boolean expressions within the search parameters. For a full list of sources it can be found under the sources tab once logged in with trinity credentials. For our purposes it allows the search of these articles for a given company. It then allows the batch download of up to 500 articles at a time.

Source Choice There are a few reasons for having chosen LexisNexis over Proquest as the final:

- the quantity of news sources
- the reliability of news sources
- the ability to download much larger batch sizes

All of these factors allow for a more reliable, and much larger set of articles available for the project, allowing for more accurate final results.

Dictionary Source

The final source is one which supplies words and their corresponding sentiment attributes. This is essential for being able to understand what sentiment proxies are indicating. The requirements for this source are that:

- it has an comprehensive set of words
- it has been built in relation to sentiment proxy analysis
- it has generic positive and negative sentiment attributes

The dictionary chosen is one provided by Rocksteady. It is a generic dictionary in relation to economic terms. As far as I have understood, the information gathered for it was based on the Inquirer newspaper. It has 11,788 word entries with over 183 attributes that may be assigned to each word, including the generic positive and negative. This makes it quite an extensive and deep dictionary, on top of it fulfilling the requirements for the project.

A potential expansion and focus to the dictionary may improve it. This would mean adding more words, and changing the attribute values to be more in line with the problem being looked at and even the company being examined. A more specific dictionary may be found, or even a mix of both of these suggestions may cause great improvement in the result accuracy.

3.2.2 Users Interacting with the System

It is important that the project has the ability to produce certain outputs. Many of these being various statistics and graphing elements. The focus was on making sure all the required pieces of information were available. Since the user base for this program is mainly computer scientists, the interface could be left in the command line. It is formed by a set of menus which allows many different all the required kinds of of operations. However, they are all executed in the command line.

3.3 System Architecture Design

The design of the system is very important when it comes to understanding it's functionality, and purpose. The general purpose as discussed is taking sentiment proxy data and price data, then analysing it and making estimations with it. The reasoning for having a daily separation between endpoints that the newspapers and articles are being used for the analysis. These come out on a daily basis, this would lead to a large amount of inaccuracies if broken down in to smaller chunks. It may be useful to consider breaking the data down on a weekly or monthly basic if there were to be future expansion, as this may give a longer term relationship view, however, this is unexplored within this project.

It can be broken down into 3 main components, with the arrangement seen in figure **??**. These being:

- `The Price Gatherer` – Gathers of all of the required price data in a date sorted array
- `The Sentiment Gatherer` – Gathers of all of the required sentiment data in a date sorted array
- `The Analyser` – Takes the data from the other two components and analyses it in various ways as well as run it through an estimator

Each of component has a very specific role within the system, and will be broken down further.

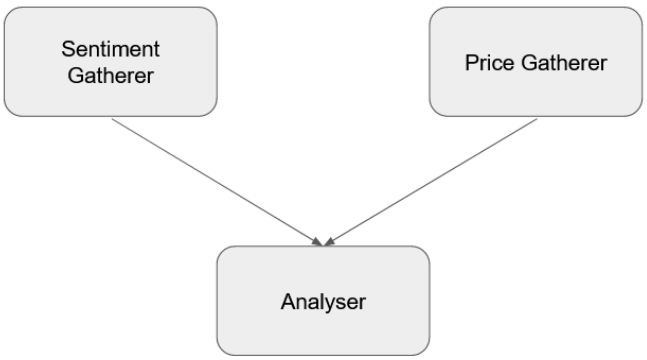


Figure 3.1: Overall Structure

3.3.1 The Price Gatherer

The Price Gatherer's purpose is to build a timeline of prices for a given company. It takes in data-points in reference to a given company's prices ordered by date and filters them in order to adapt them to the system. It can be broken down into 3 stages, arranged as seen in figure **??**:

- `The Price Source` – Handles the gathering of price values
- `Key Filtering` – Filters the raw data and extracts the desired keys for each data-point
- `The Return Adder` – Adds returns to the data-points as extra keys

The overall output of this section is an array of JSON objects, ordered by the date key. Each of these objects containing the keys, the exception being the return keys as will be discussed:

- `date` – the date of the data-point
- `close` – the closing price for the date of the data-point
- `symbol` – the symbol representing the company this data is about
- `volume` – the volume of trades for the date of the data-point
- `return1Day` – the return in relation to 1 data-point previous
- `return7Day` – the return in relation to 7 data-points previous
- `return14Day` – the return in relation to 14 data-points previous
- `return21Day` – the return in relation to 21 data-points previous

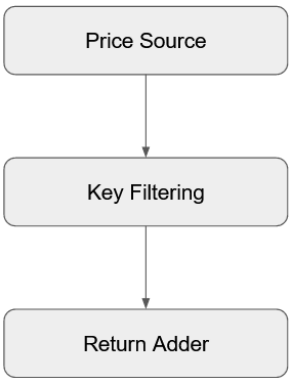


Figure 3.2: Price Gatherer Structure

The Price Source The purpose of the price source component is to contact the IEX Cloud API and collect daily price points for a given period. Due to the limitations of the tier chosen with the platform, the maximum period allowed is the past 5 years.

Key Filtering The IEX Cloud API returns an array of JSON objects, ordered by date, with each object containing the date, close, symbol and volume keys, as well as many others. This component filters out only the desired keys, removing the ones that are not required within the system. This allows the data to managed more easily, as well as any processes being executed faster, and

anything that is stored requires less memory. This thus simplifies the process. However, it does create room for future expansion and potential improvement in understanding, as more specificity would be added.

The Return Adder The keys missing after the key filtering are those that determine the return values. This component adds them to each entry. The return for a given n determines the difference in closing prices between the current entry and the entry n days before. The returns require previous entries, therefore the first n entries will not have a return given n, the number of previous entries required. The reasoning for adding returns is due to the fact that it allows to examine the change between days rather than the full days themselves, since this project is studying the change in market values.

3.3.2 The Sentiment Gatherer

The Sentiment Gatherer's purpose is to build a timeline of sentiment for a given company. It takes in articles and a dictionary and processes them together in order to create an array of sentiment values ordered by date. It can be broken down into 6 stages, arranged as seen in figure **??**:

- `The Article Source` – Handles the gathering of articles
- `The Article Parser` – Parses the raw articles into a format usable by the system
- `The Dictionary` – Handles the extraction of the dictionary from it's source
- `Key Filtering` – Filters the dictionary entries and extracts only the desired keys
- `The Sentiment Extractor` – Uses the dictionary entries in order to extract frequencies of sentiment from the articles
- `Z-Scores` – Modifies the sentiment extracted from absolute values to relative values using z-scores

The overall output of this section is an array of JSON objects, ordered by the date key. Each of these objects containing the keys:

- `date` – date of the data-point
- `articles` – z-score of number of articles
- `totalWords` – z-score of number of total words
- `positiveSentiment` – z-score of number of positive sentiment words
- `negativeSentiment` – z-score of number of negative sentiment words

The reasoning for using z-scores is due to the fact that it allows to examine the change between days rather than the full days themselves, since this project is studying the change in market values.

The Article Source The purpose of the article source is to provide the articles that will be used in the system.

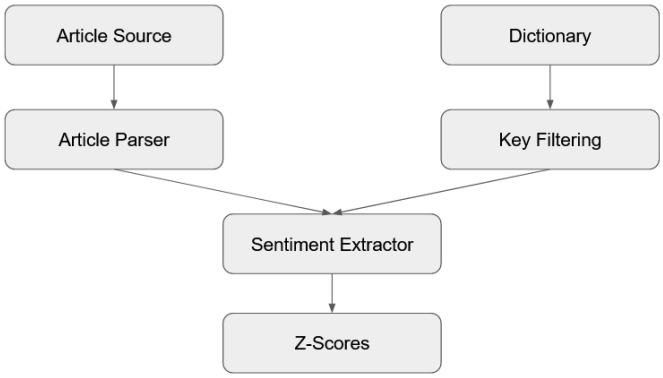


Figure 3.3: Sentiment Gatherer Structure

The Article Parser The purpose of this component is to import the files and parse them into a format usable by the system, this being JSON. This is required for LexisNexis as the articles downloaded have the rich text format. The output of this component returns an array of JSON objects, this allows metadata to be stored alongside the body of the article itself.

The Dictionary The purpose of this component is to handle the extraction of the dictionary into a JSON object array from the excel sheet it is stored in.

Key Filtering The purpose of this component is to filter only the desired keys from the dictionary, in order to decrease processing times and memory requirements. This quite importantly simplifies the analysis significantly. However, it does create room for future expansion and potential improvement in understanding, as more specificity would be added.

The Sentiment Extractor The purpose of the component is to extract sentiment frequencies from the articles, using the dictionary as a reference. What this means is that it tallies the number of words belonging to each desired attribute for each article. The articles are then joined by date, tallied appropriately, and ordered by date. The output is an array of JSON objects which represent the absolute values of the different attributes required.

Z-Scores The purpose of the component is to take the JSON object produced from the previous component and get the z-scores of the number of articles, of the total number of words in relation to the number of articles, and of each of the sentiment attributes in relation to the total number of words.

3.3.3 The Analyser

There are three components that are predecessors of the main analyser components. These are:

- `The User Interface` – A command line menu was created in order to organise and utilise all of the various components in a fast and easy way
- `The Joiner` – It joins the sentiment and prices data-sets where they overlap. Creating two new data-sets.

- `The Period Selector` – This allows for different periods to be explored within the dataset

The Analyser's purpose is to take the timelines produced by the Price Gatherer and the Sentiment Gatherer and analyse them in various different ways. It is a set of unrelated components which allow for the full analysis required in order to understand the data in the way desired for this project. The sub-components build for this component are the following:

- `Return Vs Sentiment Grapher` – Allows the graphing of sentiment keys and price keys in a time-series
- `Single Point Estimator` – Uses machine learning model to estimate next day returns
- `Autocorrelator` – Calculates correlation of key with itself with given lag
- `Return Vs Sentiment Correlator` – Calculates correlation of given keys with predefined lags
- `Descriptive Statistics` – Calculates and displays descriptive statistics of given key
- `Vector Autoregressor` – Calculate Vector Autoregression elements

The User Interface This is an essential part for the usability of the system create. It allows a user to navigate through the system, and analyse the data in any desired way, given the constraints of the system. It is a set of menu's which allow the selection of any given component and the selected that is desired in the analysis.

The Joiner For certain parts of the analysis the data being analysed should be overlapping appropriately. This is an important feature as the price and sentiment data-set have different start and end dates. As well as this not all dates that are covered by one are covered by the other, therefore data must be inserted in these cases, with the assumption that there was no activity for the data that did not exist previously. This has two main cases the first being a day with not sentiment, the assumption can be made that the day has zero sentiment, allowing an appropriate object to be inserted. As for days with no price the assumption can be made that the prices did not vary and the return was 0. However, this case is rare, and no examples of such a situation have been seen.

The Period Selector The purpose of this component is quite straight forward. It allows for the narrowing of scope. This is done by allowing for the selection of subsets of the main datasets by selecting a shorted period to examine.

Return Vs Sentiment Grapher The purpose of this component is to allow the creation of graphs which compare various columns between the prices data-set and the sentiment data-set. It however also allows the graphing of multiple columns in the one data-set, as well as graphing columns from only one data-set. This allows the creation of any graph desired. The graphs are graphed against the date of the data-point.

Single Point Estimator The purpose of this component is to explore the applications of machine learning models in this area. The models take in data from a given date, this being sentiment and

price data and estimate whether the nest day would have a positive or negative return. This allows for the comparison between machine learning methods and mathematical methods. The output of this component is two fold. The first item returned is the accuracy of the most accurate model. The second item returned is said model, thus allowing for potential future use and/or storage. This area could be explored in a lot more depth through the use of a more substantial amount of models, hyperparameters and the like.

Autocorrelator The autocorrelator allows for the calculation of the correlation of a given column in a given data-set with itself, and it displays the information with n days of lag, where n is selected by the user. This is to help understand how closely related the data is with itself in nearby days, allowing for insight into whether there is a possibility of this data affecting itself. It is important to note, however, that high values of correlation only indicate that there is a potential relationship between days, and does not indicate necessarily causation.

Return Vs Sentiment Correlator The return vs sentiment correlator allows for the calculation between two datapoints. It calculates the correlation for every day up to a given lag, it does this in both directions. Similarly to the autocorrelator it is important to note, however, that high values of correlation only indicate that there is a potential relationship between elements, and does not indicate necessarily causation.

Descriptive Statistics The purpose of the component is to gather a column from the data-set and explore it's descriptive statistics, in order to understand said column better. On top of the descriptive statistics, this component prints out a graph which allows for a visual element. This can be of great aid when trying to understand the significance of the descriptive statistics.

Vector Autoregressor The purpose of this component is to explore causation between return and negative sentiment column using vector auto-regression. This adds a layer of rigorousness to the previously explored correlations, as it allows for the understanding of not just what is correlated but which columns cause data in other columns to change. This removes elements such as coincidence.

3.4 Scope of Project

The scope of the project was mainly limited by the data being explored. Many columns were removed from the various data-sources in order to simplify this. The data-set being explored is therefore limited it a general analysis. This can be seen through the use of the generic sentiment columns as well as the basic price columns. On top of this only a certain amount of depth was explored with the machine learning models, correlation and causation. For these reasons I believe there is very large amount of potential expansion to the project, and therefore the understand of the data and its relationships.

3.5 Design Summary

This chapter provides a brief overview of the project, the system design, the handling of the required data and its sources, user interaction with the system, it's design architecture, an explanation of the system components and the scope of the project.

4 Implementation

This chapter outlines and covers the process of implementation of the project. This includes the technologies used for the project, as well as the breakdown of the structure and implementation of the different parts of the project.

4.1 Technology Used

There are various technologies used within this project. These are Python 3.9, Visual Studio Code, Git and Github, Command-Line Interface, AntConc 3.5.9, Rocksteady 0.4 and Gretl. Each of these had a different function within the project.

4.1.1 Python 3.9

Python is an interpreted, high-level and general-purpose programming language. The version used was version 3.9.

Python was used in this project to develop and execute the procedures outlined in the implementation. This is due to the fact that python is a very useful language for handling projects which require a large amount of various different features as it is general purpose. For that reasons it has many publicly available libraries which help for many of the scenarios come across throughout development.

The libraries used had three main purposes: file handling, data tidying and mathematical operations.

File Handling

In order to handle the files downloaded from *lexisnexis* and *proquest*, various libraries and modules had to be used. The libraries and modules being `sys`, `os` and `striprtf`.

The `sys` module provides functions and variables used to manipulate different parts of the Python runtime environment. It is used to create a global variable which allowed the setting of a source for files, and allowed this to be used throughout the various files in the program. The source being the choice between using *lexisnexis* and *proquest* files.

The `os` module provides functions and variables used to perform operating system tasks. It is used to access environment variables, as well as to help parse through files in a given directory.

The `striprtf` library is used to translate rtf to a python string. When files are downloaded from *lexinexis* they are in rtf format. This library is used to help parsed the information in these files into a usable format.

Data Tidying

In order to tidy up the data extracted from articles and make it usable in the context required the use of some libraries is needed. These libraries and modules are `pandas`, `json`, `copy`, `datetime`, `operator`, `matplotlib`, `seaborn` and `warnings`.

The `pandas` is an open source data analysis and manipulation tool. This library is used to aid in the tabling of data. This was useful in order to use this data within graphs and to create an excel spreadsheet. For graphing timeseries, this library was especially useful with it's `to_datetime()` function which allowed the dates to be appropriately used as indices. This is significant especially when comparing two separate time series, as it spaced the values according to date and not which datapoint it is along the sequence.

The `json` library is used to dump json data from files and the extract it back from the file in order to cache the information extracted for future use.

The `copy` library is used to create deepcopies of json objects. This is necessary as the copies had to be completely separate from the original version, and due to how python works this cannot simply be done through a shallow copy. Therefore the library was used to facilitate this.

The `datetime` module supplies classes for manipulating dates and times. As they are usually stored in strings they can be complex to perform operations with. The library makes this process a lot more straightforward.

The `operator` module exports a set of efficient functions corresponding to the intrinsic operators of Python. It is used sort arrays of json objects that contain a given key. This was very useful when ordering data entries by date.

The `matplotlib` library provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. It therefore aids in the creation and display of graphs within the system.

The `seaborn` library is a data visualization library based on matplotlib. It helps the graphs look more aesthetically pleasing as well helping them become cleaere, therefore easier to read.

The `warnings` module is used in order to supress warnings. This helps make the program be easier to read for an end user.

Mathematical Operations

In order to perform mathematical operations appropriately multiple libraries are used. This is to avoid the recreation of tested and efficient functions, and avoid any potential errors when recreating them. These libraries and modules are `numpy`, `statistics`, `scipy`, `math`, `sklearn` and `time`.

The `numpy` is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

The `statistics` is a built-in Python library for descriptive statistics.

The `scipy` is a collection of mathematical algorithms and convenience functions built on the `numpy` extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. Specifically the `scipy.stats` module was used. Similarly, to the `statistics` library it was used to gather various types of statistical information.

The `math` module is a built-in module that you can use for mathematical tasks. It is used specifically for the `log()` function contained within.

The `sklearn` library is an incredibly useful machine learning library. The `sklearn` library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. It is used for the machine learning functionality required for the `singlePointEstimator`. These being various trainable models, appropriately metric measuring functions, and some utility functions.

The `time` module provides various time-related functions. Most of the functions defined in this module call platform C library functions with the same name. This important because it is used to time some elements of the system, and these timings must be precise.

4.1.2 Visual Studio Code

Visual Studio Code is a freeware source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. It was used to ease the development of the codebase.

4.1.3 Git and Github

Git is a version control system. Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. Git allows changes by multiple people to all be merged into one source. This can be used to create features then once they are complete merge them into a final version of the system.

GitHub is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management functionality of Git, plus its own features.

4.1.4 Command-Line Interface

A command-line interface processes commands to a computer program in the form of lines of text. It is used to execute the programs developed, git commands and any other required commands.

4.1.5 AntConc 3.5.9

AntConc is a freeware corpus analysis toolkit for concordancing and text analysis. It is used for preliminary text analysis of the articles downloaded from *lexisnexis* and *proquest*. It can be used to create words lists, n-grams, concordance plots, amongst other useful tools that may be used to examine word choices in texts.

4.1.6 Rocksteady 0.4

Rocksteady is a sentiment analysis tool. It creates a timeline of sentiment, and allows the filtering as well as visualisation of this data. It is used within this project to understand how the sentiment proxy extraction process works.

4.1.7 Gretl

Gretl is an open-source statistical package, mainly for econometrics. The name is an acronym for Gnu Regression, Econometrics and Time-series Library. It has both a graphical user interface and a command-line interface. It was mainly used for vector autoregression within this project.

4.2 Setting up the System

The technologies that have been used to implement the system have been covered. This section therefore covers the details of the implementation of the indivdual components laid out in the design, using these technologies.

The system is run as a script using the terminal. The commmand for initiating the program being `python IntelligentAnalysis.py`. This is assuming that the default version of python running is version 3.9, otherwise the command may have to be modified slightly to accomodate for this. The full codebase can be found at the following url: `https://github.com/DaVinciTachyon/FinalYearProject`. This command is run from the root directory of this git repository.

4.2.1 The Price Gatherer

The Price Gatherer has a straightforward flow. Thus making the understanding of it's implementation quite simple.

The Price Source

The Price Source component has one main function. That is to contact the IEX Cloud API and return the past 5 years of price points for a given stock.

This component does require a bit of set up in order to have access to the api:

1. Register for IEX Cloud
2. Gain access to and retrieve an API key
3. Insert API key into environment variables

Once these steps are done the component itself can be used. The component can be divided into a few different steps:

1. Check if the cache exists
    - If cache exists, load items from cache
    - otherwise, continue
2. Load API key from environment
3. Create API request, with certain important information:
    - URL – This contains the API Key and Stock Symbol
    - API Key
    - Stock Symbol
    - Period desired – This is set to 5 years given the payment tier chosen on the website
4. Gather API response into JSON String
5. Sort the JSON array by date
6. Cache the data

An important item to highlight within this procedure are the use of a caching system. This allows for a large speed up in the process of gathering prices, when the program is run for a second time. As well as this it save a lot of money in API fees and network access fees. The caching system in this case is the creation of a json file, however, it may be optimised to use a database.

In order to get the cache, the program check whether the file containing th edata exists, if it does it opens the file and loads the data as a json string. It can be seen below:

Listing 4.1: Loading Prices Cache

```
if os.path.isfile(pricesFilename):
    with open(pricesFilename) as json_file:
        data = json.load(json_file)
```

The cache is the created by opening a file and simple dumping the json string within, as can be seen here:

Listing 4.2: Creation of Prices Cache

```
with open(pricesFilename, 'w') as json_file:
    json.dump(data, json_file)
```

Another item to not is the sorting of the JSON array. This is usually a more complex process, however, the `operator` module eases this process significantly.

Listing 4.3: Sort JSON Array

```
sorted(data, key = operator.itemgetter('date'))
```

Key Filtering

The key filtering process takes in the full data-set and the keys desired for each element of the data-set. It then creates a new data-set with only the desired keys extracted. This is currently a `for` loop which goes through the entire data-set adding each entry individually.

Listing 4.4: Filtering Desired Keys

```
for entry in originalDataset:
    filteredEntry = {}
    for key in keys:
        filteredEntry[key] = entry[key]
    newDataset.append(filteredEntry)
```

The current solution has a time complexity of `O(N)`, where N is the length of the data-set. Some research has been done in order to find a more efficient solution, however, so far this has been unsuccessful.

The Return Adder

This component requires an array of numbers for the returns to create. What this means the that it will reveceive a 1 in the array if it wants to calculate 21 days returns. It then loops through the data-set adding the appropriate returns to the given entry.

The return cannot be added to the first n days, this is simply done using an `if` statement to make sure to not attempt this before it is possible.

The formula used for return is $log(\frac{r_t}{r_{t-l}})$ where r is the return, t is the current day, and l is the number of days being used.

The final code put this all togther in the following way:

Listing 4.5: Adding Returns

```
for t in range(len(prices)):
    for l in returnLengths:
        if(t >= l):
            prices[t][f"return{l}Day"] =
                math.log(prices[t]['close']/prices[t-l]['close'])
```

4.2.2 The Sentiment Gatherer

The Sentiment Gatherer can be divided into three main subsections, each one of those being divided into two components. It is important to not that the first two main subsections can be run in parallel, and must be run before the last. The Article Source and Article PArser compose this first subsection. This can be run in parallel to the Dictionary and Key Filtering components. However,

they must all be run before the Sentiment Extractor and consequently the Z-Scores components.

The Article Source

The Article Source component has one main function. That is to gather the dowloaded articles and import them into the system.

This component requires a bit of set up:

1. Log into LexisNexis with Trinity credentials
2. Search for a given company
3. Download articles, this part has a few steps:
   (a) Select rtf format
   (b) Remove formatting from files
   (c) Download 500 files by giving an appropriate range – this is the maximum available for downloading at once
   (d) Repeat from step (a) 5 times
4. Wait 2 hours – Once 5 downloads have been executed, LexisNexis does not allow any more dowloads for 2 hours
5. Repeat from step 1 as many times as necessary

Using this process 7000 articles were dowloaded for this iteration of the system, for the given company, which will be discussed later on. The reason for the choice of 7000 article is due to the fact that they cover the 5 year range covered by the Price Source quite thoroughly. They are a significant amount of articles, however more could be dowloaded. That said they covered the requirements for this project.

There was though of automating this procedure in order to improve up this process. This would expedite the process and lead to the ability to download a lot more articles. There were a few reasons this was not done:

- The 2 hour wait is still required, therefore the process would not be expedited significantly
- The amount of articles gathered manually were sufficent for this project
- There was a focus on creating other parts of this project, as the creation of this automation would require a significant amount of time

Once these step are executed and the downloaded files are placed in the correct directory the component functionality is complete.

The Article Parser

The Article Parser Component gathers the articles and turns them into a format usable by the system. The component can be broken down similarly to the Price Source component with some differences:

1. Check if the cache exists
    - If cache exists, load items from cache
    - otherwise, continue
2. Find all files with articles
3. Iterate through all the files, doing the following:
    (a) Read the contents of the file and assign them to an rich text format (rtf) string
    (b) Parse the rtf string into a legible string using the `rtf_to_text()` function
    (c) Split the string into the articles contained
    (d) Parse the article strings into a JSON object containing the required content and metadata
    (e) Append the articles to a JSON array
4. Cache the article JSON array

The caching system is identical to the Price Source, the JSON array is dumped to a file and extracted from the file when required.

The extraction of the articles from the file into an rtf string has a point that should be highlighted. The way this works requires each line to be extracted separately, then concatenated to the previous, until the full file is extracted. This part of the program was given in order to aid in the project, however it was quite inefficient. The original code looked like this:

Listing 4.6: Slow Text Extraction

```
fileContent = ''
for line in file:
    if line.strip() != '':
        fileContent += line + "\n"
    else:
        fileContent += line
```

The improvement made to the code then looked like this:

Listing 4.7: Optimised Text Extraction

```
fileContent = "".join([f"{line}\n" if line.strip() != '' else f"{line}" for line in
    file])
```

Beyond reducing the number of lines in the program and looking cleaner, this change creates a radical change in the speed of the program. The program was run on the 7000 articles, and originally was let run for 12 hours before being terminated. Then once the change was made the program consistently finishes within 5 minutes. The reason for this being such a big deal is that if there were more articles added, the entire program would have be run again. The reason for the for the speed difference explains why there is such a large timing difference.

The original way uses the += operator, what this means is that an array is created for the original string, as a string is a character array, then a new character array is created with the new appropriate length in order to allow for the change. This is quite a costly process both timewise and spacewise, and it is repeated for every line in every file. It is important to remember that each time this process is executed the array becomes bigger, making it a slower process each time.

The optimised method used the the `"".join()` function. The way works is that it creates a set of strings one for each line, then does the joining only once. Therefore the large array array has to be created only once and does not have to have its size change.

The final point to discuss is the parsing of the python string into a JSON array. In order to utilise the information in an easier manner, some metadata is extracted alongside the content of the article itself. Each article is represented by a JSON object with the following keys:

- `title` – The article title
- `source` – The name of the publication the article was published in
- `date` – The date the article was published
- `copyright` – The type of copyright of the article
- `length` – The length of the body of the article
- `section` – If the article is divided into parts, this will indicate which part this is
- `language` – The language the article is in, for this project, the articles are in English
- `pubtype` – The type of publication the article was published in - newspaper, magazine, etc.
- `subject` – Key words which describe the article contents
- `geographic` – The locaiton of publication
- `loaddate` – The date the article was uploaded to the the Article Source
- `byline` – The author(s) of the article
- `body` – The contents of the article

These metadata items are extracted into the array from the original string by using certain indicators withing the text. An example of such an indicator would be that the line containing the source thatsrts with `"Source:"`. This is executed in the code the following way:

Listing 4.8: Source Extraction

```
if tempLine.startswith('Source:') and extractValue(line) != '':
```

```
    document['source'] = extractValue(line)
```

The extractValue function simple removes the indicator, allowing for the extraction of the value only.

The special case is the body. This still has indicators. The indicator of the start of the body is simply a line only containing "Body" and the end of it is simple represent by "End of Document". The lines within these two indicatos are simply concatenated with each other and assigned to the correct key at the end. The += operator is used for this. There is room for creation of efficiency, however, due to the relatively short nature of these articles, this is efficient enough for use within this process currently. In the future this would be an area to explore the creation of such efficiencies.

The Dictionary

The dictionary is extracted from the Rocksteady files. If there were future development to this project it may be worth spending time adding more entries, as well as making the sentiment choice more specific to the context of the articles that are chosen. An example of such a thing would be that for a company such as Gamestop the word game may have positive sentiment, even though in general it may have no sentiment attached.

The extraction of the entries from the excel file is very simply achieved with the use of the pandas library. The following function extracts all the entries and associates each item to it's correct row and column.

Listing 4.9: Dictionary Extraction

```
data = pd.read_excel(r"./dictionaries/inquirerbasic.xls")
```

Key Filtering

This process of extracting only the desired sentiment attribute columns is achieved through the use of a dataframe. It takes the data-set from the previous component and an array containing the titles of the desired columns as inputs, and returns only said columns.

Listing 4.10: Key Filtering Dataframe

```
df = pd.DataFrame(data, columns= ['Entry', 'Positiv', 'Negativ']).to_numpy()
```

The dataframe is then turned into a dictionary object. What this does is it assigns the values of the desired sentiment attribute columns to the word itself. The object will have the following format: `{ word: [ sentimentColumnValue ] }`. This is achieve with the following segment of code:

Listing 4.11: Dictionary Creation

```
dictionary = {}
for index, item in enumerate(df):
    dictionary[item[0]] = item[1:]
```

It is important to note that `item[0]` is the word itself and the rest of the items in the array are the sentiment attribute column values.

The Sentiment Extractor

This component can be broken down into two steps:

1. Extraction
2. Date Joining

Extraction This step takes in the articles from the Article Parser and the Dictionary from the Key Filtering. It then creates a new array of JSON Objects. Each object represent an article. Each object has the following keys:

- `date` – The date key from the article
- `totalWords` – The length key from the article
- `positiveSentiment` – The number of words that have the `Positiv` attribute determined by the dictionary
- `negativeSentiment` – The number of words that have the `Negativ` attribute determined by the dictionary

If more sentiment attributes were to be included they would have an extra key assigned to them. The sentiment attribute keys are found by iterating through all the words in the body of the article and tallying the words according to which columns are labeled true in the dictionary.

Date Joining This step takes the array created in the previous step and merges all of the elements on any given day together. This means the final JSON object will be an array ordered by date with the following keys:

- `date` – The date of the articles
- `articles` – The total number of articles on said day
- `totalWords` – The total number of words in all the articles
- `positiveSentiment` – The number of words that have the `Positiv` attribute determined by the dictionary in all the articles
- `negativeSentiment` – The number of words that have the `Negativ` attribute determined by the dictionary in all the articles

Similarly to the Price Gatherer, the array is ordered by date in the following way:

Listing 4.12: JSON Array Ordered by Date

```
sentimentByDate = sorted(sentimentByDate, key = operator.itemgetter('date'))
```

Z-Scores

This component can be divided into four steps:

1. Column Separation
2. Percentage Calculation
3. Z-Score Calculation
4. Object Assignment

Column Separation The array created in the Sentiment Extractor is iterated through and an array is created for each key within the objects. This means that each key will have its own array.

Percentage Calculation In this project it is important to undesrtand relative amounts. Therefore a few different percentages must be calculated. The sentiment columns are replaced with the percentage of sentiment words in relation to the total number of words for that article. The totalWords column is replaced with the percentage of words in relation to the number of articles.

Z-Score Calculation The `scipy.stats` module is then used to calculate the z-score for each in element in each of the arrays, excluding the `date` array. An example of this is:

Listing 4.13: Z-Score Calculation

```
articles = stats.zscore(articles)
```

Object Assignment The final step in the component is create an array from the separated array with the following keys:

- `date` – date of the data-point
- `articles` – z-score of number of articles
- `totalWords` – z-score of number of total words
- `positiveSentiment` – z-score of number of positive sentiment words
- `negativeSentiment` – z-score of number of negative sentiment words

This will still be ordered by date, as the order was never changed.

4.2.3 The Analyser

The Analyser is composed of various unrelated components, which are preceded by three components which allow the other to achieve their tasks efficently, and in an easy to execute manner. The inital three components are the following:

- The User Interface
- The Joiner

- The Period Selector

The User Interface

The User Interface is essential towards creating a navigatable environment for users of the system. In a very basic sense it is a set of menus which allows the selection of which tool to use within the Analyser, and selected the desired parameters for these tools.

These menus work by showing a user a set of options, and allowing the user to put in a number in order to select one of these. This methods allows for full usage of the program, and allows limitations to be imposed so that it may not be abused. The steps to achievinng one of the menus is the following:

1. Print out the options so that the user may know what they are
2. Request the input
3. Ensure the input is an integer, and convert the string input to one
4. Select the appropriate the appropriate function to run given the option selected, or repeat the question if the input was erroneous

Step 1 The options can be laid out with the following code:

Listing 4.14: Menu Options

```
print("1: 1 year", "2: 2 year", "3: 3 year", "4: maximum available", sep="\t")
```

Step 2 Python has a function to simplify the process of gathering the input. The input has the `strip()` function run on it in order to remove beginning and ending whitespace, just incase this is done by mistake.

Listing 4.15: Get Input

```
sampleSize = input("Please select the length of your sample size: ").strip()
```

Step 3 In order to ensure the input is a valid integer the follwing statement is run. It simply checks if the input is a digit, and if it is it converts it to an integer.

Listing 4.16: Check if Integer

```
if sampleSize.isdigit():
        sampleSize = int(sampleSize)
```

Step 4 The correct function is selected in one of two methods.

1. A set of `if/elif` statements, since a `switch-case` statement does not exist in python

Listing 4.17: If/Elif Statements

```
if section == 1:
    dataset = prices
elif section == 2:
    dataset = sentiment
```

2. Check if the input is an integer and in the desired range, and can be given into the function

Listing 4.18: Integer Range Choice

```
if isinstance(column, int) and column > 0 and column <= len(keys):
    column = keys[column - 1]
```

The `isinstance` function allows the system to determine if a variable is an instance of a given type. In this case if it is an `int`.

Both of the allow for the situation where an input is entered incorrectly. In these cases the `else` option is chosen which asks you to try again, and the boolean which determines whether the item has been chosen is set kept as false, since it is set to true in other cases.

The Joiner

The Joiner wants the prices and sentiment data-sets to overlap. This has a requires a few step to be excuted in ored to be achieved:

1. Select start date – this will be the start date of the data-set that starts later between the two
2. Select end date – this will be the end date of the data-set that ends earlier between the two
3. Find the element indexes for start and end dates for both data-sets
4. Create data-sets with new start and end indexes
5. Add empty elements to days where the datset does not have a day the other does, in order to have fully overlapping data-sets

For step 4, this can be achieved using the following notation within Python:
`prices[priceStart:priceEnd]`, `sentiment[sentStart:sentEnd]`, where the indexes have been previously discovered.

For step 5, the empty objects are the following:

- For a day with no sentiment

Listing 4.19: No Sentiment Day

```
{'date': prices[i]['date'], 'articles': 0, 'totalWords': 0,
    'positiveSentiment': 0, 'negativeSentiment': 0}
```

- For a day with no price activity

Listing 4.20: No Price Day

```
{'date': sentiment[i]['date'], 'close': prices[i - 1]['close'], 'symbol':
    prices[i]['symbol'], 'volume': 0}
```

It is important to note that this step is done on the the full values of these arrays, then the z-scores are calculated afterwards.

The Period Selector

This item has it's own menu that runs after the tool to use has been selected. It allows the choice between 4 different periods. These being:

- 1 year – The data-sets returned will start 1 year before the end date of the data-sets, and have all the following values
- 2 years – The data-sets returned will start 2 years before the end date of the data-sets, and have all the following values
- 3 years – The data-sets returned will start 3 years before the end date of the data-sets, and have all the following values
- maximum available – The full data-sets will be used

This is done by getting the date of the last element, and subtracting the appropriate time from it. The date is an instance of `datetime`, not a string.

Listing 4.21: Subtract years

```
date - relativedelta(years=years)
```

All of the elements with a date including and past the start date are then return as the data-set to be used.

Return Vs Sentiment Grapher

The menu for this component allows the user to select the prices columns and sentiment columns they which to visualise.

The first step towards creating these graphs is inserting the objects into dataframes, where the indexes are the sets of dates as `datetime` instances. This enables them to be used as timeseries.

The next step is to use the `matplotlib` library in order to create the graphs themselves. The figure on which the graph will be displayed needs to be split into two subplots. These will be overlapping each other, with the x-axis being the date, and the y-axis being on one side representative of the price and the other the sentiment.

Listing 4.22: Split Window into Multiple Subplots

```
fig, ax = plt.subplots()
```

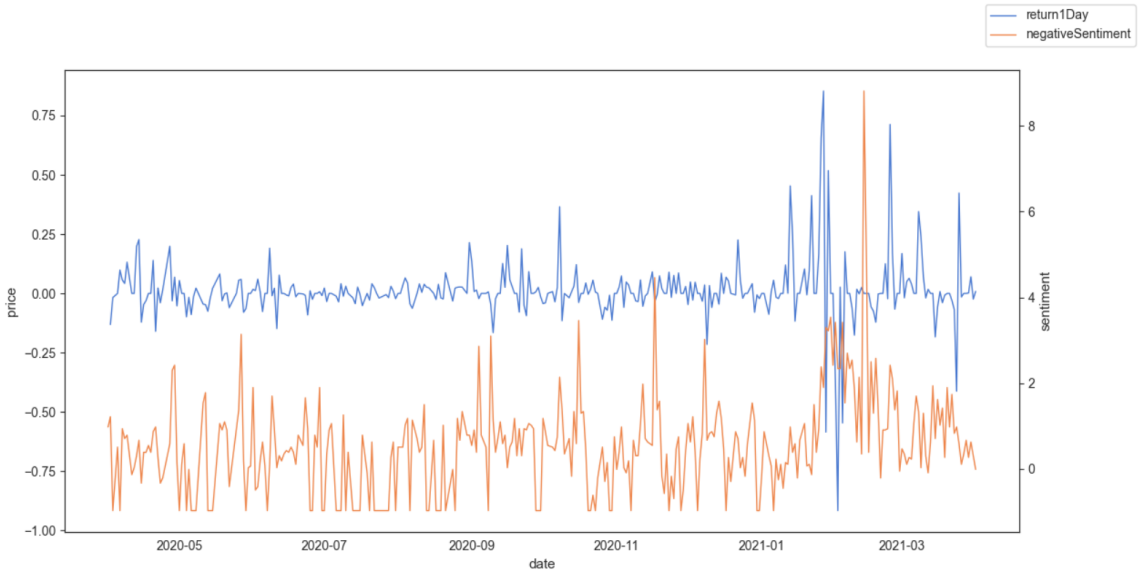


Figure 4.1: Sample Graph

Listing 4.23: Create Overlapping Subplot

```
ax2 = ax.twinx()
```

Listing 4.24: Plot Dataset

```
ax.plot(dataset, color=colour, marker=marker, linewidth=linewidth)
```

Listing 4.25: Plot Legend

```
fig.legend(legend)
```

Listing 4.26: Save picture of plot

```
fig.savefig('priceVsSentiment.jpg', format='jpeg', dpi=100, bbox_inches='tight')
```

Single Point Estimator

The Single Point Estimator can be broken down into 4 main steps:

1. Create input and output series
2. Input/Output Series Shuffling
3. Model Testing
4. Return Highest Accuracy Model and its accuracy

Create input and output series The first step changes the prices and sentiment JSON object arrays into an array containing all of the desired columns, excluding date, as an array for each entry. As well as creating a second array with Boolean values for whether the next day's return is positive

or negative. An example of this is if the close and volume columns are desired from the prices data-set and the negative sentiment column is desired from the sentiment data-set, the input series array elements would look like this `[ closeValue, volumeValue, negativeSentimentValue ]`. If the next day's return for a given entry is positive, the output series element would be `True`.

Input/Output Series Shuffling The shuffling simply randomises the order of the entries, so that there is no bias based on date. This is done using the function given by the `sklearn.utils` module in the following way:

Listing 4.27: Shuffle series

```
X, y = shuffle(X, y)
```

This function shuffles the order of both series while mainting the correspondece between the two sets.

Model Testing The model testing itself can be broken down into a discrete set of steps common accross all models. The difference being that some models require hyperparameter training, meanwhile others do not. These steps are:

- `Hyperparameter Training` – A hyperparameter is a parameter whose value is used to control the learning process, it is given as an input to the model. An example for a hyperparameter is the number of neighbors to examine for the `KNeighborsClassifier`. Ranges of values are given and iterated through, the highest accuracy value is stored and used to create a final model.
- `K-Folds Cross-Validator` – It provides train/test indices to split data in train/test sets. Split dataset into k consecutive folds (without shuffling by default). Each fold is then used once as a validation while the k - 1 remaining folds form the training set. This implementation uses 2 folds. This is an arbitrary value, further exploration may lead to finding a more suitable value. The `kf.split(inputDataset)` is the function which excutes the split on the dataset.
    1. `model.fit()` – Train the model on the training dataset
    2. `model.predict()` – Create a predicted output set given the test input dataset
    3. `accuracy_score()` – Calculate the accuracy of the values predicted from the model against the test output dataset
- `mean(kFoldAccuracies)` – The mean accuracy of the accuracies calculated for each of the folds is set as the accuracy for the given hyperparameter
- `max(hyperparameterAccuracies)` – The maximum accuracy of the accuracies calculated for the various hyperparameters is set as the defined accuracy for the model
- `timer` – The `perf_counter()` function from the `time` module is used to time the training and discovery of hyperparameter for the model. The start time is set to before the iteration through the hyperparameters, and the end time is just after.

- `max(modelAccuracies)` – The accuracy of the highest accuracy model as well as the trained model itself are returned

An example of the steps being executed on the KNeighborsClassifier is following:

Listing 4.28: KNeighborsClassifier Example

```
kf = KFold(n_splits=2)

startTime = perf_counter()
K = range(1, 10)
highestAccuracyK = 0
highestKNNAccuracy = 0
for k in K:
    accuracies = []
    for train, test in kf.split(X):
        model = KNeighborsClassifier(n_neighbors=k).fit(X[train], y[train])
        pY = model.predict(X[test])
        accuracies.append(accuracy_score(y[test], pY))
    accuracies = np.array(accuracies)
    accuracy = np.mean(accuracies)
    if(accuracy > highestKNNAccuracy):
        highestKNNAccuracy = accuracy
        highestAccuracyK = k
print("Processed: ", model.__class__.__name__)
print("Time: ", round((perf_counter() - startTime) * 1000), "ms")
```

Accuracy Score vs Mean Squared Error An important point to discuss is the choice of function used in order to determine the accuracy of a given prediction set. The two options to consider are `accuracy score` and `mean squared error`. The `accuracy score` option computes subset accuracy, the set of labels predicted for a sample must exactly match the corresponding set of correct labels. The `mean squared error` of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. For this reason the choice when using classifier models is the `accuracy score` option since classifier results are either labelled correctly or not.

Models Comparison For this component a number of models are used in order to find the most accurate model for a given dataset. The following models were all chosen because they are various different different models which are excellent at classifications. They each have different strength and weaknesses, and for that reason one the models may work better or worse depending on the dataset.

- `KNeighborsClassifier (KNN)` – KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. What this means is that the decision for the output will be based on the k nearest neightbors. K is

determined through hyperparameter training as described above.

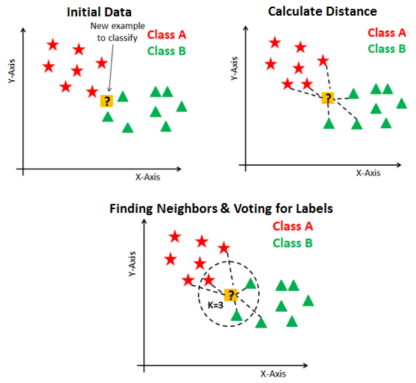


Figure 4.2: KNeighborsClassifier

- `DecisionTreeClassifier` – A decision tree is created from the training data in order to achieve the classification. A decision tree is a flowchart-like tree structure where an internal node represents a feature, the branch represents a decision rule, and each leaf node represents the outcome. This can be seen in the figure below. Within the system the

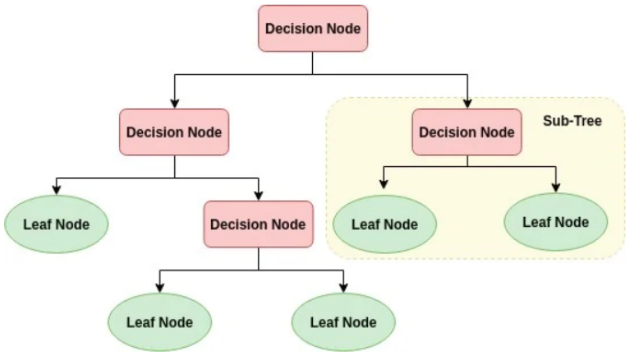


Figure 4.3: Decision Tree

`DecisionTreeClassifier()` class is created. However to quickly cover the steps in creating a decision tree, they are the following:

1. Select the best feature using Attribute Selection Measures to split the dataset. Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner. Attribute Selection Measure provides a rank to each feature by explaining the given dataset. There are multiple ways of assigning each feature a value. Some of these being:
   - `Information Gain` – It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values
   - `Gain Ratio` – It extends Information Gain by handling the issue of bias by normalizing the information gain using Split Info
2. Make that feature a decision node and breaks the dataset into smaller subsets.

3. Starts tree building by repeating this process recursively for each child until one of the condition will match:
   - All the tuples belong to the same feature value.
   - There are no more remaining features.
   - There are no more instances.

- `GaussianProcessClassifier` – It is a classifier which uses Gaussian Processes. Gaussian Processes are a generalization of the Gaussian probability distribution. They are a type of kernel model capable of predicting highly calibrated class membership probabilities, although the choice and configuration of the kernel used at the heart of the method can be challenging. For the purposes of this project an arbitrarily chosen kernel configuration was chosen in order to simplify the process. The choice was the basic confugration shown in the documentation for the class. A radial basis function was used to create this. A radial basis function being a real-valued function whose value depends only on the distance between the input and some fixed point. For further explanation, the documentation for this can be referenced.
- `AdaBoostClassifier` – AdaBoost or Adaptive Boosting is a type of ensemble boosting classifier. Boosting algorithms are a set of weak classifiers which create a strong classifier. These algorithms help decrease model bias. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. The general steps for this classifier are:
  1. Initially, all observations are given equal weights
  2. A model is built on a subset of data
  3. Using this model, predictions are made on the whole dataset
  4. Errors are calculated by comparing the predictions and actual values
  5. While creating the next model, higher weights are given to the data points which were predicted incorrectly
  6. Weights can be determined using the error value. For instance, the higher the error the more is the weight assigned to the observation
  7. This process is repeated until the error function does not change, or the maximum limit of the number of estimators is reached
- `RandomForestClassifier` – This algorithm creates decision trees on randomly selected data samples, gets a prediction from each tree and selects the best solution by means of voting. The n hyper parameter must be trained, this determines the number of forests to use, i.e. the number of random samples. The general steps are:
  1. Select random samples from a given dataset
  2. Construct a decision tree for each sample and get a prediction result from each decision tree

3. Perform a vote for each predicted result
4. Select the prediction result with the most votes as the final prediction

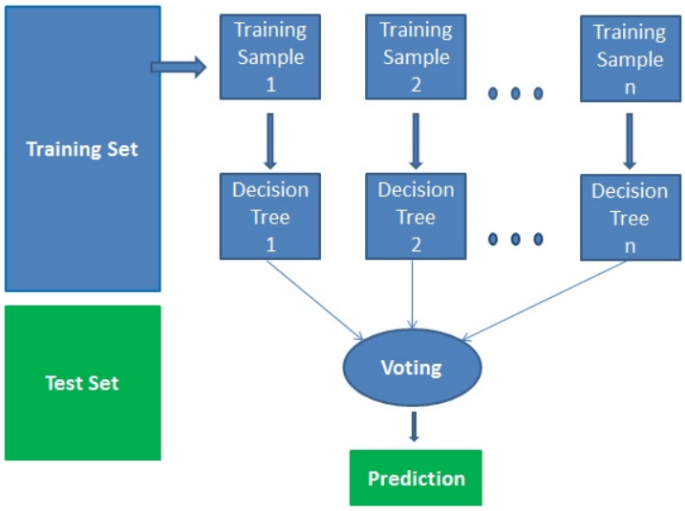


Figure 4.4: RandomForestClassifier

Return Highest Accuracy Model and its accuracy The accuracy of the different models is compared and the highest accuracy model is returned to the user.

Autocorrelator

The menu for the autocorrelator allows the selection of a column and lag to be examined. This component calculate the correlation between a column and itself with an offset of n days. N starts at 1 and calculates the autocorrelation for every integer until the lag amount is reached. All of these values are returned.

Listing 4.29: Example AutoCorrelator Output when lag

```
return1Day Auto Correlation
1 day lag 0.010044065231783677
2 day lag 0.13420635746867923
3 day lag 0.17706335839486953
4 day lag -0.23281554097728582
5 day lag 0.0305722042951903
```

The way correlation is calculated is using pearson correlation. The `scipy.stats` module provides a function, `pearsonr()`, which allows this. It outputs a tuple where the first value is the correlation coefficient and the second is the p-value. The correlation coefficient is the proportion of the variance in the dependent variable that is predictable from the independent variable, i.e. correlation.

Return Vs Sentiment Correlator

This component calculates the correlation between a column in the prices dataset and one in the sentiment dataset, at a lag chosen by the user. Correlation is calculated in the same way as within

the AutoCorrelator, i.e. using pearson correlation.

Listing 4.30: Example Return Vs Sentiment Output

```
return1Day/negativeSentiment Correlation
same day -0.009080953348214611
1 day lag return1Day to negativeSentiment 0.060205743120825606
1 day lag negativeSentiment to return1Day -0.052446888115240176
```

Descriptive Statistics

The menu for this component allows for the selection of the desired column. It the calculates the descriptive statistics for the column. It also displays a graph in order to allow for a better understanding of the statistics. The descriptive statistics shown are:

- `Mean` – The average of all the data-points
- `Standard Error` – Measures the accuracy with which a sample distribution represents a population by using standard deviation
- `Median` – The value separating the higher half from the lower half of a data-set
- `Mode` – The most common value in the data-set
- `Standard Deviation` – Measures the amount of variation or dispersion of the data-points
- `Sample Variance` – The average of the squared differences from the mean, helps measure variance within the data-points
- `Kurtosis` – Defines how heavily the tails of a distribution differ from the tails of a normal distribution, i.e. it measures the number of extreme values in the data-set
- `Skewness` – Measures the asymmetry of the distribution about its mean
- `Range` – The difference between the maxim and minimum values
- `Minimum` – The minimum value found in the data-points
- `Maximum` – The maximum value found in the data-points
- `Sum` – The sum of the data-points
- `Count` – The number of data-points analysed

Vector Autoregressor

For this section, the Gretl software was used to gather the desired results. The steps to using Gretl for Vector Autoregression analysis are the following:

1. Use an excel sheet created by the python program
2. Import excel sheet into gretl
3. Change the data to have a time-series interpretation

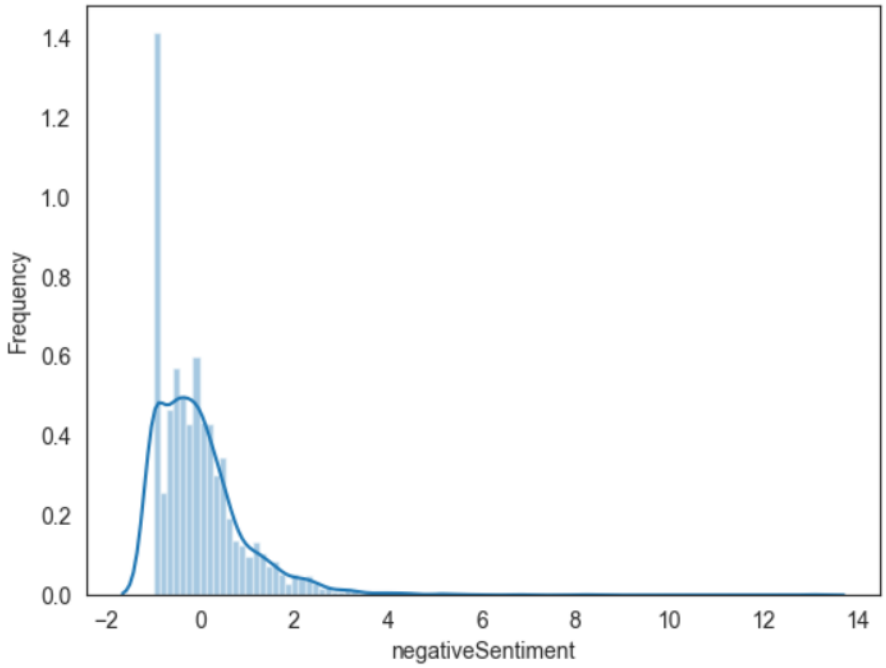


Figure 4.5: descriptiveGraph

4. Select a lag length, by using VAR lag selection model. For the purposes of simplicity in this project the maximum allowable lag is 10, and the lag length is chosen based on the most optimised AIC, Akaike criterion, value
5. Execute Unit Root Test in order to determine whether data is dependent on time, in this project the Augmented Dickey-Fuller Test was used. If the asymptotic p-value is less than desired significance level, in this case 0.1, then it is not dpendent on time.
6. Perform the Vector Autoregression Analysis, by inserting all the desired columns as endogeneous values. For each column causation is confirmed if the p-value of the f-value is less than the significance level, in this case 0.1.

4.3 Implementation Summary

In this chapter, the technologies used and how they were used to implement the design decided upon is discussed. It breaks down specifics of implementation as well as discussing why certain choices were made within.

5 Results Evaluation

This chapter discusses results provided by the developed system and how they answer the questions posed at the beginning of this project. The full set of data looked at is stored in the appendix ??.

5.1 Examined Entity

The entity that will examined is a company called Gamestop. Gamestop is an American video game, consumer electronics, and gaming merchandise retailer. It is the largest video game retailer worldwide.

Figure 5.1: Gamestop Logo

The main reason for choosing this company is due to the fact that it has historically been a relatively stable company. However, an interesting sentiment fuelled phenomenon happened recently. In January of 2021, the company stock prices increased by 1,500% over the course of two weeks due to a short squeeze mainly attributed to the members of a subreddit, r/wallstreetbets. Reddit is a social news aggregation, web content rating, and discussion website, and a subreddit is a channel within this website which specialises in a given topic. In this case a subreddit dedicated to stocks with high market risk. This seems to indicate that the stock prices of Gamestop were heavily driven by sentiment. Making this an exceptional case-study for this project.

5.2 Descriptive Statistics

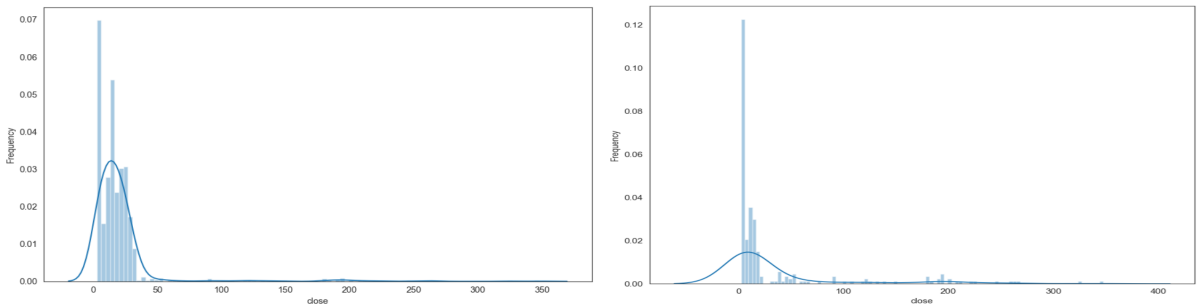
The first step towards understanding the data-set is to understand each of the columns by themselves. This first step that may be used is to create an outline of the data-set using descriptive statistics, alongside a graph. This lays out the data in such a way were time is not an element and simply the general trends of the data can be understood. A part of this analysis is to attempt to

understand whether these general trends have remained consistent. Therefore, various time spans are examined alongside the data-set as a whole. The time-spans examined here are the entire data-set and the past year worth of data.

5.2.1 Price Columns

The columns which give the most amount of information are:

- Closing Prices
- 1 Day Returns



(a) (b)

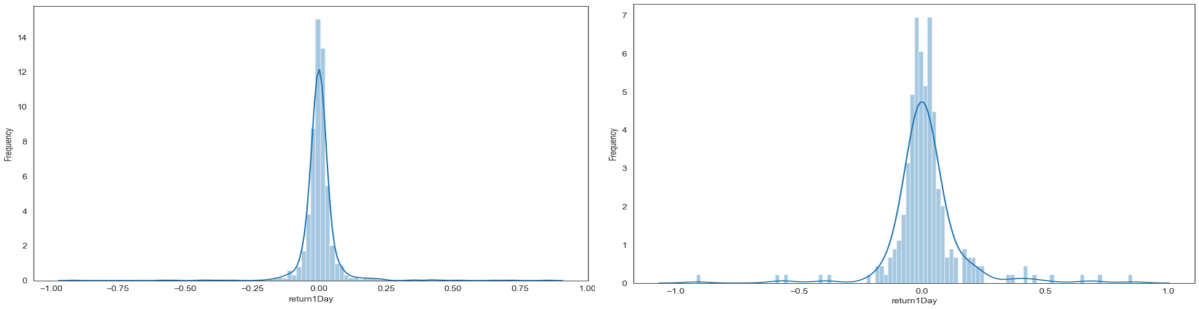
Figure 5.2: Closing Prices (a) entire data-set (b) last year of data

The first column examined is the closing price column. The initial observations from the overall dataset are:

- There is a heavy left skew, confirmed with a skewness value of 6.0809, evidenced with a mean of 20.2554 when the standard deviation is 30.5661 and the minimum value is 2.8 while the maximum value is 347.51
- There are long tails, confirmed with a kurtosis value of 43.0012 and a range of 344.71 when the standard deviation is 30.5661
- There is a large concentration of points in the one area, making it unimodal, evidenced by a standard error of 0.8618

The data-set which includes only the last year of data-points compares in the following ways. The overall value of the closing prices has increased to having a mean of 35.3204 from 20.2554. However the most common values do lie in the same are as the mode went from 4.14 to 4.44, indicating that in general the price stayed the same. The median went from 15.22 to 10.22, actually decreasing a little bit, indicating that the much higher highs were balanced by much lower lows. The heavy left skew is maintained, it is however diminished with a new skewness value of 2.5837. This can partially be explained by the maximum and minimum values being the same, and the large amount of curve flattening seen by the standard error increasing to 4.0341 from 0.8618, the standard variance increasing to 4117.2946 from 935.0304. Similarly, the tails are still long given the kurtosis value of 6.1112, and the maintenance of the same range. However, the tails have more values within. All of this combined indicates that in the last year the stock's closing price has been far more volatile that

usual, with the highest highs and lowest lows in the range examined. It seem that it has generally increased in value given the new mean. However, it is still hovering around the same price given the mode and median.



(a) (b)

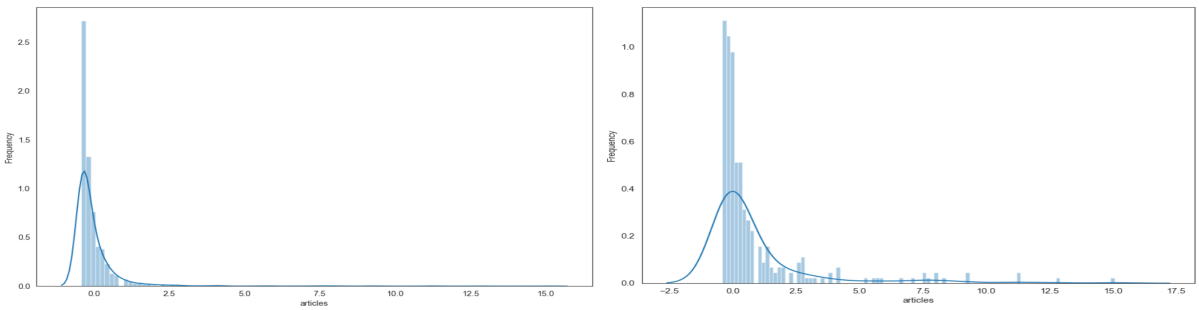
Figure 5.3: 1 Day Returns (a) entire data-set (b) last year of data

The 1 day returns column is also examined. The initial observations are:

- There is little to no skew, confirmed with a skewness value of 0.7638
- There are long tails, given the kurtosis value of 51.5650 and evidenced with the range of 1.7700 with a standard deviation of 0.0755
- There is a large concentration of points in the one area, making it unimodal, and evidenced with a standard error of 0.0021

The examination of the full data-set confirms the points made for closing prices. The mean, mode and median are all increased to 0.0162, 0.0053 and 0.0223 from 0.0014, 0.0 and 0.0. Indicating higher returns than over the entire data-set, however, this is marginal. Similarly as with closing process the skewness remains quite similar, to 0.7638 from 0.3239, where the decrease may be attributed to the increased flatness of the curve, seen by the standard error increase to 0.0096 from 0.0021. As well as this the tails are kept long, seen in the kurtosis value of 12.8680, and the same range being maintained. This information confirms the points laid out for closing prices.

5.2.2 Sentiment Columns



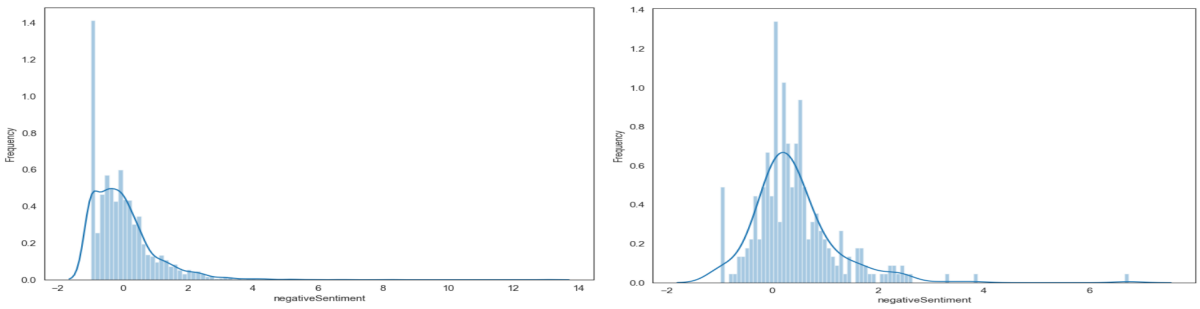
(a) (b)

Figure 5.4: Article Volume (a) entire data-set (b) last year of data

The first column examined from the sentiment columns is the article volume column. The initial observations from the full data-set are:

- There is a heavy left skew, confirmed by the skewness value of 7.3609
- There are long tails, given by the kurtosis value of 73.7692 and the range of 15.4788 and standard deviation of 1.0
- There is a large concentration of points in one area, making it unimodal, and evidenced with a standard error of 0.0220

The data-set which only takes into account one year is then considered. The mean and median are raised to 0.8623 and 0.1096 from 0.0 and -0.2422, meanwhile the mode is maintained constant at -0.4181. This indicates that there are more higher values, however, most of them are still concentrated in the same area. The other difference to point out is that flattening of the curve, indicated in the change of standard error to 0.1325 from 0.0220. The tailness maintained but slightly decreased due to the flattening of the curve, seen in the change of kurtosis to 12.2162 from 73.7692. The changes indicate that in the last year there was was more variation in the number of article towards the high-end, however, the majority of points remain the same. This aligns with what is seen in the prices and returns, which indicates some correlation between the two aspects.



(a) (b)

Figure 5.5: Negative Sentiment (a) entire data-set (b) last year of data

The next column examined is the negative sentiment column. The positive and negative sentiment column behave very similarly. So only one of these is examined. The initial observations are:

- There is a left skew, confirmed by the skewness value of 2.7791
- There are long tails, given by the kurtosis value of 19.5138 and the range of 14.05 and standard deviation of 0.9998
- The data is concentrated around one area, making it unimodal, and evidenced with a standard error of 0.0220

The 1 year data-set compare in the following ways. The mean, median and mode all increase to 0.4129, 0.27 and 0.08 from 0.0001, -0.17 and -0.99. This indicates a general increase in negative sentiment in the last year. The curve flattened a small amount in the year given the change of standard error to 0.0489 from 0.0220. Unlike the rest of the examined columns however, the range of data is significantly decreased to 7.72 from 14.05, where the minimum remained the same at

-0.99, and the maximum decreased from 13.06 to 6.73. These items seem to indicate that the general sentiment was more negative more regularly, however, it never reached the extremes that it previously had. It is interesting to note that a similar pattern occur when positive sentiment is examined. This seems to indicate that when there is more volatility in the market, and a larger amount of articles, the sentiment is generally higher, negative and positive.

It is important to note that preceding information indicates that the examination is conditional on the time-span examined for this data-set. This is due to the fact that the descriptive statistics change significantly depending on the time-span chosen.

5.3 Auto Correlation

This step in understanding the data-set continues the understanding of columns by themselves. However, it examines the relationship of the columns with themselves. It does this by finding the correlation of a column entry with a previous entry given a lag time. For example if the lag is 5, it will find the correlation with 5 entries prior, i.e. 5 days before. This is important to understand whether previous days have any correlation, and potentially an effect on the current day. For the purposes of this examination, the lag looked at is 5 for all items. In this section a p-value of 0.1 or smaller is going to be considered of statistical significance. The p-value indicates the probability of a non-correlated system producing data-sets that have a correlation at least as extreme as the one computed from these data-sets. Therefore if 0.1 is chosen as a p-value to match or surpass for statistical significance, it means that the coefficient calculated or higher will be observed up to 10% of the time. It is important to remember that correlation is a measure of linear relationship, and given that many of the relationships may be non-linear, the following may indicate a relationship, it may not describe it appropriately however. There has been some research into exploring non-linear correlation values for this paper, however, so far it is inconclusive.

Closing Price Auto-Correlation with entire data-set		
Lag	Correlation	P-Value
1	0.9412	0.0
2	0.9109	0.0
3	0.8589	0.0
4	0.7893	0.0
5	0.7581	0.0

Closing Price is the first column examined. All of the values examined in this case are statistically significant. It is interesting to note how the correlation starts at over 90%, it then quickly decreases down to 76% over the five days. This indicates that the prices vary frequently between one day and the next, i.e. it is quite volatile. As found in paper (**?**), the general correlation of a non-volatile stock is over 95%, also known as a well behaved stock.

1 Day Return Auto-Correlation with entire data-set		
Lag	Correlation	P-Value
1	0.01	0.722
2	0.1342	0.0
3	0.1771	0.0
4	-0.2328	0.0
5	0.0306	0.2795

1 Day Returns is the next column examined. The statistically significant values are those when the lag is 2, 3 and 4. There are two interesting things to note within this data. The first is that the correlation increases daily up to day 3, then flips and is greatly negative. This indicates a behaviour known as mean reversal. Mean reversal is a term for a stock price's tendency to move to the average price over time. It is however, interesting to note that the summation of the significant correlations is 0.0785, which is greater than 0, indicating that there is a general growth direction for the stock price. This may also be reflected in the closing prices' decreasing correlations.

Article Volume Auto-Correlation with entire data-set		
Lag	Correlation	P-Value
1	0.7087	0.0
2	0.5052	0.0
3	0.3913	0.0
4	0.3588	0.0
5	0.4346	0.0

For article volume, all of the values are statistically significant. The notable item is that there is a very rapid decrease in correlation as lag is increased. It starts at 0.7087, this indicates that the article volume may be clumped in values, however, this does not last very long.

Negative Sentiment Auto-Correlation with entire data-set		
Lag	Correlation	P-Value
1	0.1936	0.0
2	0.1473	0.0
3	0.1388	0.0
4	0.1002	0.0
5	0.1903	0.0

For negative sentiment, all of the values are statistically significant. The correlation starts relatively low, it then decreases for the first 4 days of lag, then returning to a value similar to the 1 day lag value on the 5^{th} day. This indicates that there may be some sort of delayed feedback in sentiment, however, this is quite speculative. Interestingly enough quite a similar pattern is seen in positive sentiment, shown in appendix.

5.4 Return to Sentiment Correlation

This step is when the relationship between return and sentiment is explored. This is done by exploring the correlation between given columns in the different areas. The value of 0.9 for the

p-value is again chosen for statistical significance. The full tables can be found in appendix ??. The tables in this section will only show statistically significant values.

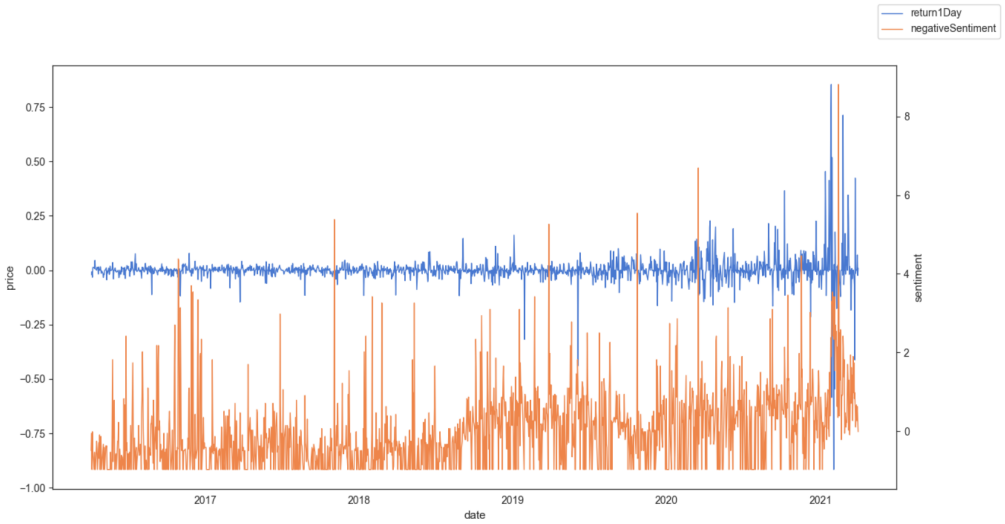


Figure 5.6: 1 Day Returns Vs Negative Sentiment with entire data-set

This graph shows that as the volatility of the returns increases, the negative sentiment increases. Indicating that there may be a potential relationship. It is also interesting to note that as of the end of 2018 the general negative sentiment level increased. It may be interesting to explore why this is the case.

1 Day Return Vs Negative Sentiment Correlation with entire data-set		
Lag	1 Day Return/Negative Sentiment	Negative Sentiment/1 Day Return
0		
1	0.0477	
2		
3	0.0474	
4		-0.0442
5	0.0431	-0.0444

The first thing that is interesting to note is the relationship between the two directions of lag. It seems that 1 Day Return to Negative Sentiment has generally positive correlations, and for the opposing directions the opposite is true. This indicates the potential role of mean reversal, as if return increases negative sentiment will increase the next day, then the return will decrease in turn. It is also interesting to note that all of the values are very similar. The negative values even mirror the positive ones. This shows that the current return is related to what is said in the past few days as well as what will be said for a few days. It should also be noted that return positively correlates to negative sentiment and negative sentiment negatively correlates to returns. What these mean is that an increase in returns will correlate to an increase in negative sentiment, and an increase in negative sentiment correlates to a decreased level of returns.

This graph indicates that the large amount of return volatility recently works in tandem to a larger amount of articles that usual.

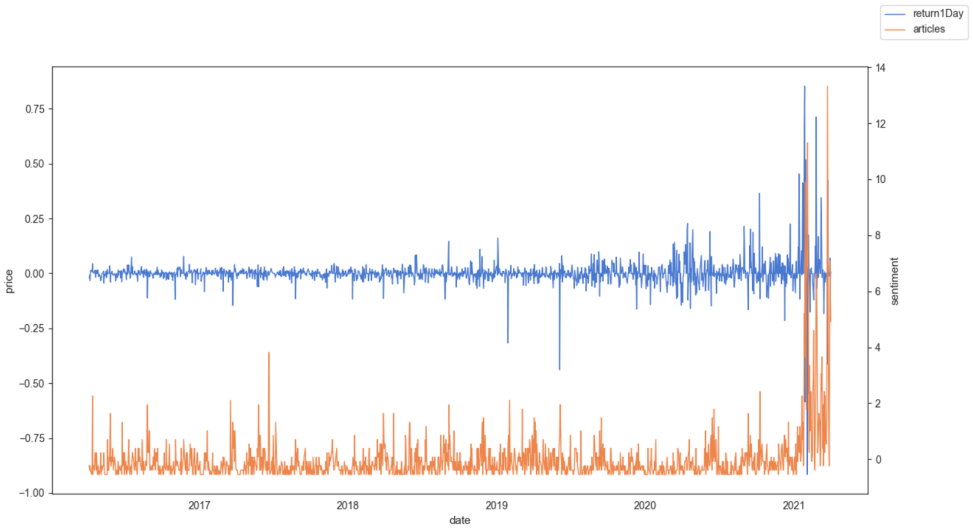


Figure 5.7: 1 Day Returns Vs Article Volume with entire data-set

1 Day Return Vs Article Volume Correlation with entire data-set		
Lag	1 Day Return/Article Volume	Article Volume/1 Day Return
0		
1		
2	0.0926	
3	0.0749	
4		-0.07
5	0.1267	-0.0679

It should be noted that returns positively correlates to article volume and article volume negatively correlates to returns. This indicates that if returns increase, article volume will too, however, if article volume increases returns will decrease. This is quite similar to the correlations between returns and negative sentiment.

1 Day Return Vs Negative to Positive Sentiment Ratio Correlation with entire data-set		
Lag	1 Day Return/Negative to Positive	Negative to Positive/1 Day Return
0		
1		
2	-0.0476	
3	0.0721	
4	-0.0615	
5		

The positive to negative sentiment ratio correlation to 1 day returns was explored. This gave a few insights. It seems that this ratio does not correlate to future returns in a statistically significant way. However, the reverse is not true. The interesting part is that for the three days with statistically significant results the direction of influence is inconsistent, while maintaining a similar absolute value.

5.5 Vector Auto-regression

Finally all of this will be put together to explore causation. The full data-sets are stored in appendix ??. The first three vector auto-regression use only one more variable on top of 1 day returns. The variables are then added on starting with the one which will have the greatest effect. This enable all of the variable explored to be added, and allows to see how much of an effect these have.

VAR Comparisons						
	Lag Selection		Unit root Test	VAR		
Name	AIC	Lag	asymptotic p-value	f-value	p-value	augmented R^2
1 Day Return	0.113197	6	5.185e-043	13.75066	1.68e-27	0.086788
Positive Sentiment	0.113197	6	5.969e-026	11.78705	3.92e-23	0.074417
1 Day Return	-0.014553	9	3.842e-034	10.82126	8.07e-27	0.089024
Negative Sentiment	-0.014553	9	9.545e-014	24.22991	2.17e-64	0.187747
1 Day Return	-0.719845	9	2.87e-033	11.11346	1.33e-30	0.101754
Article Volume	-0.719845	9	8.907e-009	130.6266	3.0e-297	0.592161
1 Day Return	1.863201	9	2.87e-033	7.870976	1.64e-28	0.103495
Article Volume	1.863201	9	8.907e-009	16.08440	2.60e-65	0.202196
Negative Sentiment	1.863201	9	9.545e-014	88.52368	1.5e-292	0.595228
1 Day Return	4.279104	10	2.049e-028	5.850417	7.63e-27	0.107786
Article Volume	4.279104	10	8.052e-007	11.44909	3.48e-62	0.206507
Negative Sentiment	4.279104	10	1.107e-009	60.29136	7.8e-284	0.596244
Positive Sentiment	4.279104	10	5.969e-026	4.332794	2.42e-17	0.076646

From the examined models, the best fitting model is Negative Sentiment estimation from 1 Day Return, Article Volume, Negative Sentiment and Positive Sentiment, with an Adjusted R^2 of 0.596244. Meanwhile the most accurate model which estimates 1 Day Returns has an Adjusted R^2 of 0.107786 and is estimated from 1 Day Return, Article Volume, Negative Sentiment and Positive Sentiment.

5.6 Machine Learning Exploration

The various machine learning models are run against each other. The models are compared in figure ??.

The model selector returns the GaussianProcessClassifier, with a 61.34% accuracy. However, looking at figure ??, it may have been wise to select the AdaBoostClassifier due to it's similar accuracy, 58.71%, but significant training speed advantage that is 18974 ms against 420 ms training times.

5.7 Results Evaluation Summary

This chapter uses the results provided by the system to explore the questions posed for this project.

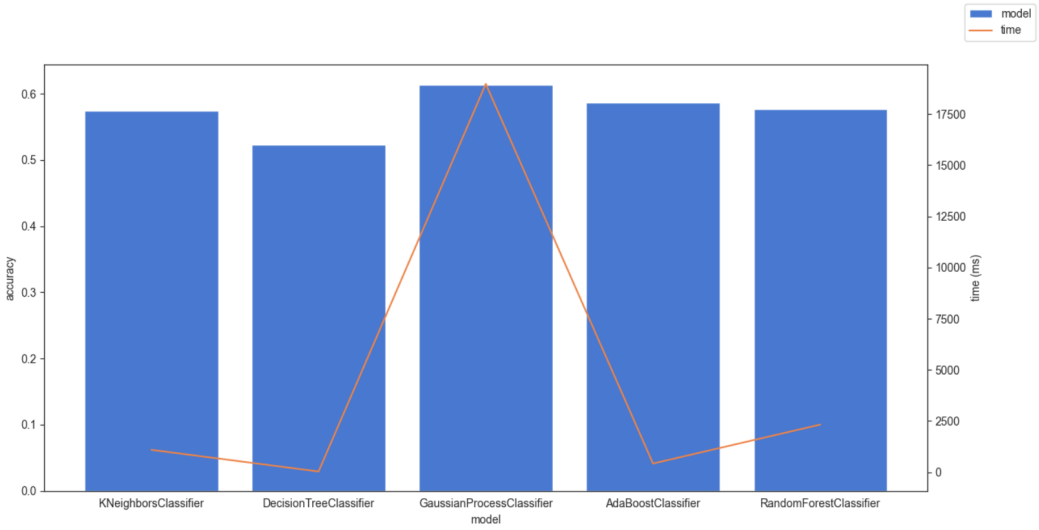


Figure 5.8: Model Comparison

6 Discussion & Conclusion

In this chapter, the questions explored and the way they are answered is explored, the success to which they have been answered, and how the challenges put in the way were overcome. There will also be an outline of potential continued work in the area.

6.1 Project Achievements and Challenges

The overall goal of the project is to explore the relationship between price changes and sentiment. That is to understand how to use qualitative data in a quantitative system and following this exploring and potentially identifying causal links between these sets of data. Both of these areas are explored thoroughly within the project.

The first step in answering these questions was developing an understanding for the area, and designing a system which would allow the exploration of these questions. There was a significant amount of ground to cover when it came to understanding what was required in order to develop such a system, then put this understanding together into an appropriate design. In order to achieve the learning required to do this, a few different things were done. Papers relevant to the subject area had to be read and understood, as well as similar systems having to be broken down and understood. Both of these were requirements in order to develop the desired system, and both of these thing required putting myself in situations I had little to no experience in. For reading papers, this was the first time I had read and understand many different papers in a subject area, in order to build on that understanding. I found this procedure to be very different to any previous experience I had in the area. There was a lot more depth that had to be covered to get a thorough enough understanding. The other part of this was the breaking down of systems that do parts of what was required for this. This was quite new, as there was a development in not only the understanding of how to break down a system in order to understand it, but then also how to relate that to the more concrete information learnt in papers that have been read in order to produce a desirable design which achieves what is required. This amount of learning took a great effort, and it is quite interesting how some time was spent just learning how to learn from these sources.

The next main area was the exploration of the data-sets required and how to use these data-sets. This was one of the areas which surprised me the most. It required a large amount of consistent effort to develop. This was done three times in three different ways in order to gather all of the required datasets. A few general important steps were identified, however, each data-set displayed it's own particular set of challenges. It is important to note that I had little to no experience in this

area previously, so it required a large amount of learning to achieve the most basic tasks within this, as the procedures were all novel to me. Firstly, it is important to identify a reliable data-source by understanding the content it returns. Within this datasource it is then important to discover the information to be extracted, as depeding on the goal of the project the data-source may deliver different results. It is then important to learn how to gather the data-set, parse it into a usable format, and extract the desired information from the data-set, in the case of the sentiment data-set the correct meta-data had to be extracted from different articles. Once the only the desired information is fully extracted from the data-source, the creation of the dataset used for analysis may not be done. As in the case of sentiment, the meta-data from the articles had to be used in order to create a timeline of sentiment. To do this required an understand of what sentiment is as well as the creation of a dictionary, which is a data-set in and of itself. Once all of this is done in order to work with the price and sentiment data-sets together they had to be joined in such a way that they were covering the same days. As may be understood all of this required a large amount of learning in various different areas at each step, especially given my inexperience in the areas beforehand. However, this process helped develop and understanding for the question which explored using qualitative data in a quantitative system, as well as creating data-set that could then answer the other question.

The final step in this project was the examination of the data-sets as well as the relationship in and between them. This part required the most amount of learning by far, the data-set had to be understood in order to determine the analysis procedures that will be run on them in order to achieve the knowledge the andswer the question of causation, as in what data-points influence what other data-points. Once these procedures are run, it is then important to understand what conclusions can be drawn from the results, and how those results influence the further analysis that will be done on the data.

All of this must then be put together into developing and understanding which allows the implementation of a system which allows all of the discovered procedures to be executed upon, so that we may actually learn from the data. It was important to do this in such a way such where the desired procedures are corretly implemented, but also in such a way that allows for utmost efficency. This is very important considereing the sizes of the data-sets.

I find that the project answers the original questions posed in full, as well as laying out the ability to answer these questions with different data-sets. It also achieves the aim of teaching me about the area in question as thre was a very large amount of learning required to complete this project.

As for explicitly answering the two questions laid out.

1. How to examine qualitative data in a quantitative system?
2. Is there a causal relationship between sentiment and pirice changes? Can this relationship be used to to estimate price changes?

All of these questions have been answered within this project. Briefly, question 1 is answered through the use of dictionaries, and percentages of sentiement words in relation to the whole. Question two can simply be answered with two yeses. The details of these responses are withing the paper. However, I am quite happy with the conclusion.

Despite the success of the project, there are thing that may have been done better. Many of them are discussed within the relevant sections. However, one I would like to bring up is the use of correlation as a statistic. It is used since it gives and indicator of the linearity between two sets of points. However, my issue lies in the fact that these datasets are non-linear, meaning there may be a better approach to this process. I have done some research but was not currently able to find anything, but I propose that there may be a better way to analyse the relationship between two sets of non-linear data. The current way is useful, as it does simplify the procedure. However it may be executed upon in a more precise way.

In regards to the distilling the main challenge of this project, it can be simply stated as volume of learning. It was a very unfamiliar area an almost every item approach required something to be learnt, if not the entire thing. It was an enjoyable experience.

As for a personal statement, in my opinion, the project could be so much better and cleaner given more time, as there was a fair amount of fumbling and going down wrong paths. However, once things started to click there was a great satisfaction in understanding how and why things are done.

6.2 Future Work

The goals for this project were achieved. However, there is still work to be done.

It may be summarised as improvement on the current system. There are a few main areas that may be improved, these main be summarised as depth and breadth.

Simply, depth refers to the further exploration of current attributes explored. As well, as making methods and data-sets more specialised for the task at hand. A few examples of this are:

- the specialisation of the dictionary used for sentiment in regards to financial terms, word sentiment dependent on the company being analysed
- the choice in mathematical functions such as correlation

In general, I feel there is far greater exploration that could be done with the current available data-sets.

As for breadth, this refers to the expansion of the tools and items explored. This means, exploring more sentiment columns within the dictionary, as well as increasing the sample size of articles chosen. It may also mean analysing different companies in order to find patterns between them, and potentially patterns for companies as a whole.

There are always a lot of interesting directions a project could go, and this is no exception.

6.3 Discussion & Conclusion Summary

In this chapter, the achievements and challenges on the way to completing the project are summarised, as well as giving a brief overview of what may still be done.

A1 Appendix

The appendix contains extensions upon the information in the rest of the text. However, this information was not necessary to reach the desired conclusions in the text.

A1.1 Descriptive Statistics

A1.1.1 Closing Prices

Entire Dataset

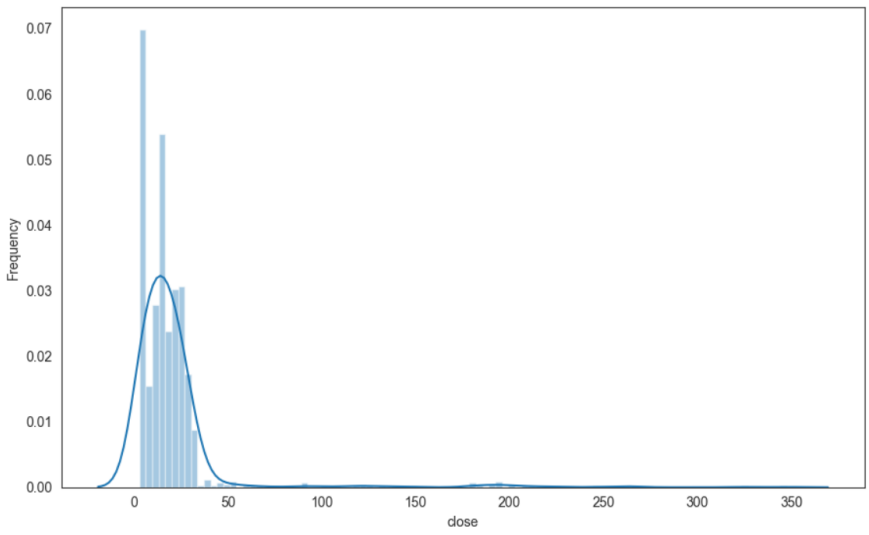


Figure A1.1: Closing Prices with entire dataset

Closing Price Descriptive Statistics with entire dataset	
Mean	20.25539316918189
Standard Error	0.8617871146224114
Median	15.22
Mode	4.14
Standard Deviation	30.56612021354625
Sample Variance	935.0303819398896
Kurtosis	43.00117239290764
Skewness	6.080884821536175
Range	344.71
Minimum	2.8
Maximum	347.51
Sum	25501.540000000005
Count	1259

Last Year

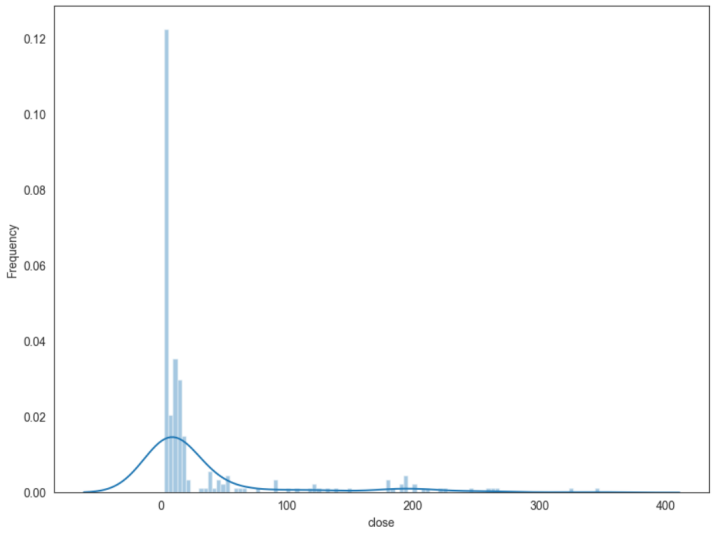


Figure A1.2: Closing Prices in the last year of data

Closing Price Descriptive Statistics in the last year of data	
Mean	35.32043478260869
Standard Error	4.034091200037088
Median	10.02
Mode	4.44
Standard Deviation	64.03921248871352
Sample Variance	4117.294627984817
Kurtosis	6.11122201031286
Skewness	2.5837008559997834
Range	344.71
Minimum	2.8
Maximum	347.51
Sum	8936.07
Count	253

A1.1.2 Trading Volume

Entire Dataset

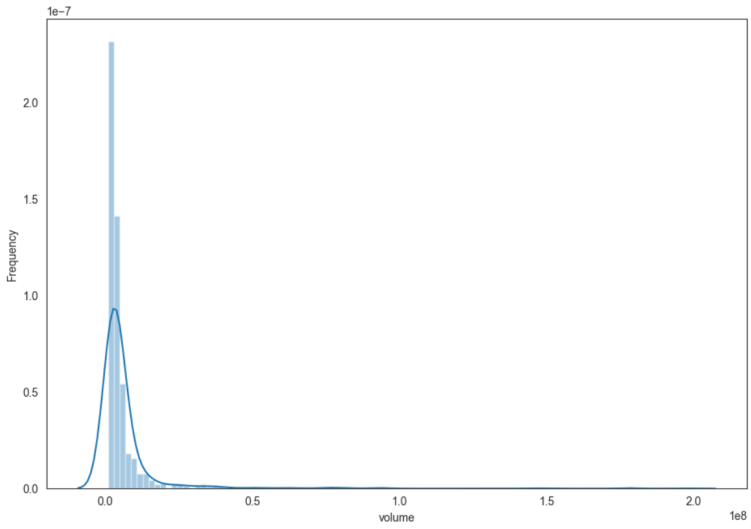


Figure A1.3: Trading Volume with entire dataset

Trading Volume Descriptive Statistics with entire dataset	
Mean	6435867.801429706
Standard Error	399372.9943423072
Median	3130713
Mode	1491760
Standard Deviation	14165079.458700724
Sample Variance	200808974859915.2
Kurtosis	81.30752829995787
Skewness	8.021781388564962
Range	196185042
Minimum	972904
Maximum	197157946
Sum	8102757562
Count	1259

Last Year

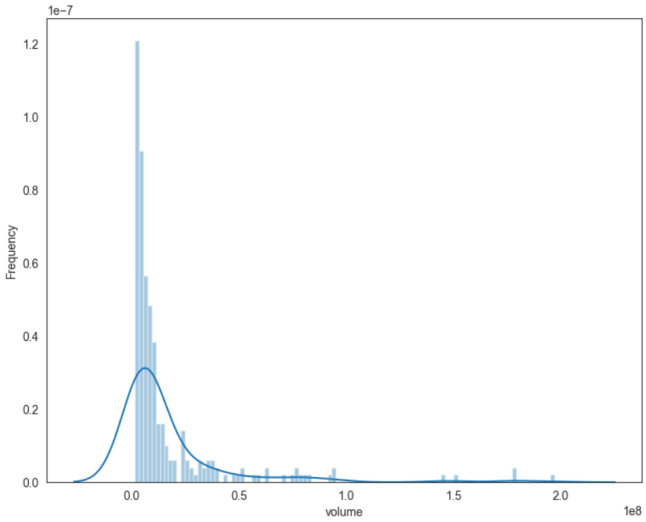


Figure A1.4: Trading Volume in the last year of data

Trading Volume Descriptive Statistics in the last year of data	
Mean	16731690.841897232
Standard Error	1798626.81876273
Median	6603951
Mode	4568695
Standard Deviation	28552315.58314456
Sample Variance	818469783592652.2
Kurtosis	16.215823114980463
Skewness	3.730320402302758
Range	195827485
Minimum	1330461
Maximum	197157946
Sum	4233117783
Count	253

A1.1.3 1 Day Returns

Entire Dataset

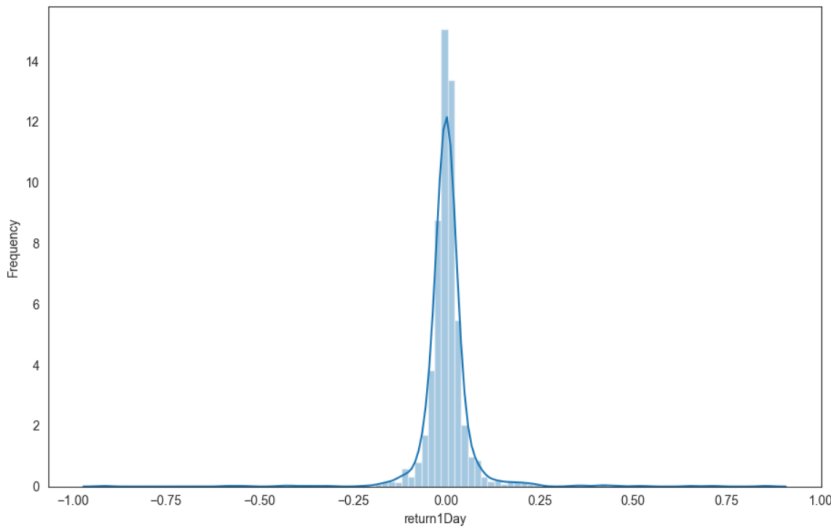


Figure A1.5: 1 Day Returns with entire dataset

1 Day Return Descriptive Statistics with entire dataset	
Mean	0.0014490461774454091
Standard Error	0.0021291609999713954
Median	0.0
Mode	0.0
Standard Deviation	0.07548769098797223
Sample Variance	0.005702924817259385
Kurtosis	51.56501043384626
Skewness	0.7637560467880287
Range	1.770007043945965
Minimum	-0.916290731874155
Maximum	0.85371631207181
Sum	1.8229000912263227
Count	1258

Last Year

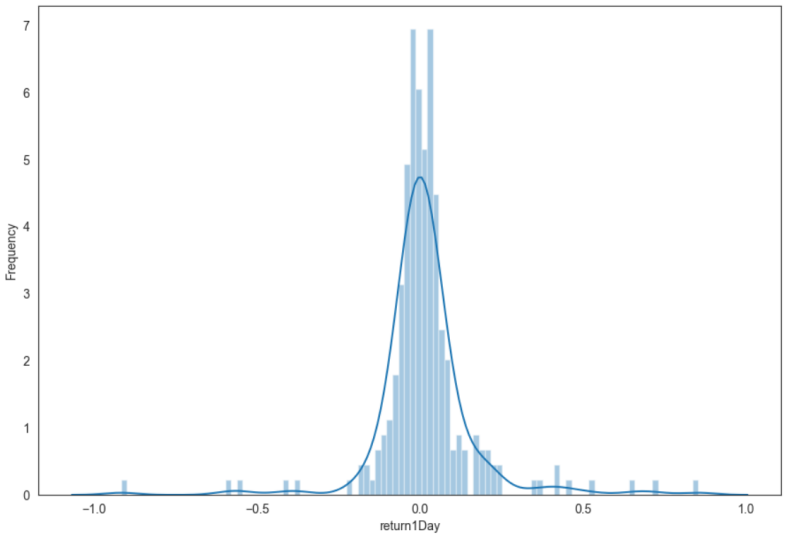


Figure A1.6: 1 Day Returns in the last year of data

1 Day Return Descriptive Statistics in the last year of data	
Mean	0.01617449079992721
Standard Error	0.009609836418644279
Median	0.005313187705878563
Mode	0.022335953942063298
Standard Deviation	0.15224844154955602
Sample Variance	0.02327193691026168
Kurtosis	12.867982373920272
Skewness	0.32394336153811276
Range	1.770007043945965
Minimum	-0.916290731874155
Maximum	0.85371631207181
Sum	4.075971681581655
Count	252

A1.1.4 Article Volume

Entire Dataset

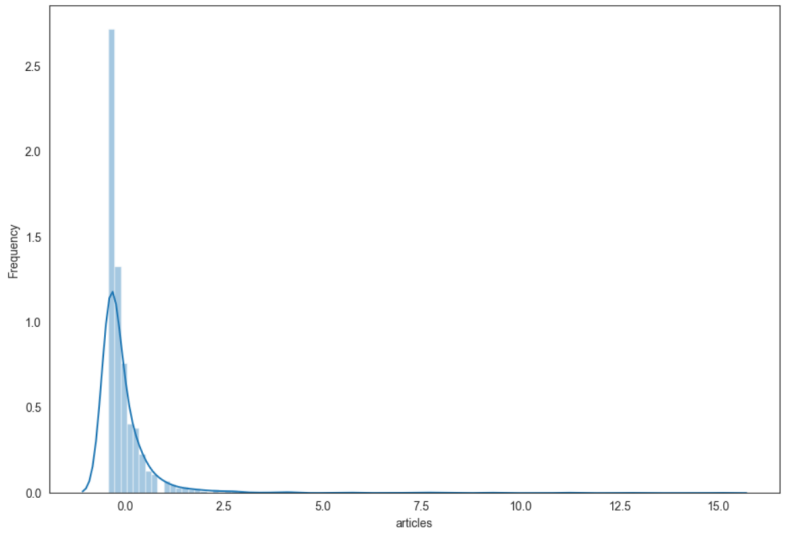


Figure A1.7: Article Volume with entire dataset

Article Volume Descriptive Statistics with entire dataset	
Mean	-2.2065213996409285e-17
Standard Error	0.02196873875818734
Median	-0.2421631034547878
Mode	-0.41805802758295
Standard Deviation	1.0
Sample Variance	1.0004826254826256
Kurtosis	73.76924857053827
Skewness	7.360900854018193
Range	15.478753323278276
Minimum	-0.41805802758295
Maximum	15.060695295695327
Sum	-4.642022877199281e-12
Count	2073

Last Year

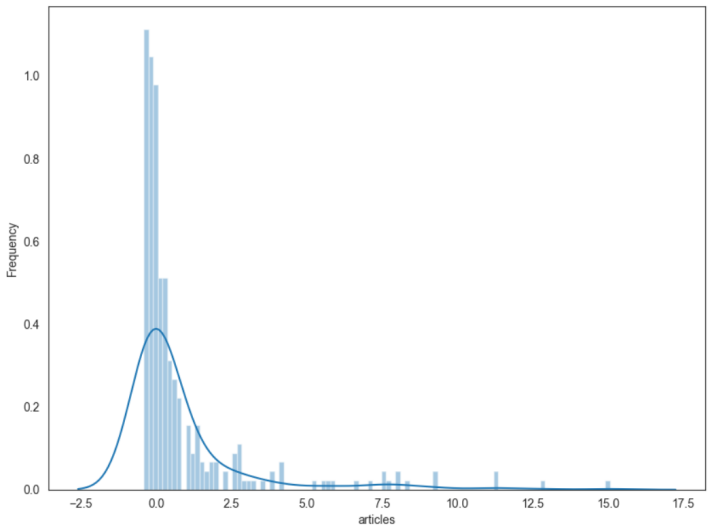


Figure A1.8: Article Volume in the last year of data

Article Volume Descriptive Statistics in the last year of data	
Mean	0.8623357132258448
Standard Error	0.1325148497318309
Median	0.1096267448015367
Mode	-0.41805802758295
Standard Deviation	2.2527524454411254
Sample Variance	5.092453765840419
Kurtosis	12.21615870052046
Skewness	3.296079615884022
Range	15.478753323278276
Minimum	-0.41805802758295
Maximum	15.060695295695327
Sum	250.07735683549507
Count	290

A1.1.5 Words Volume

Entire Dataset

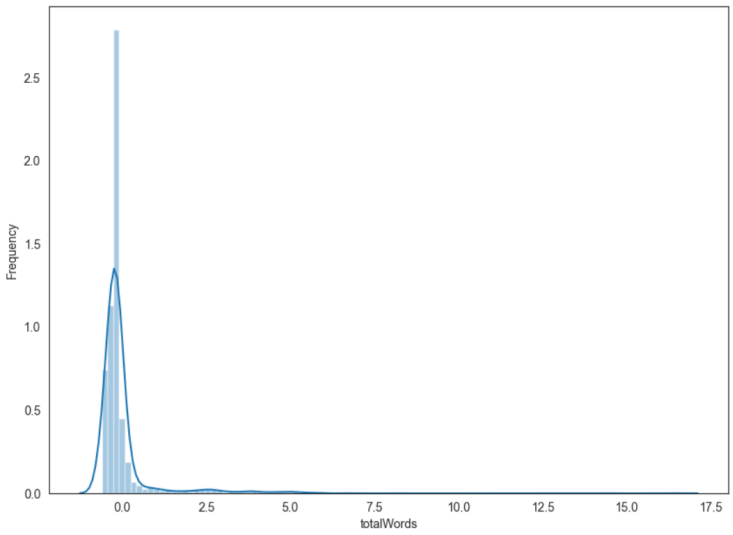


Figure A1.9: Words Volume with entire dataset

Words Volume Descriptive Statistics with entire dataset	
Mean	0.00013989387361311993
Standard Error	0.021968180574674083
Median	-0.2
Mode	-0.19
Standard Deviation	0.999974591918116
Sample Variance	1.0004317854395641
Kurtosis	66.70204164146547
Skewness	6.45609886055835
Range	17.07
Minimum	-0.61
Maximum	16.46
Sum	0.28999999999995796
Count	2073

Last Year

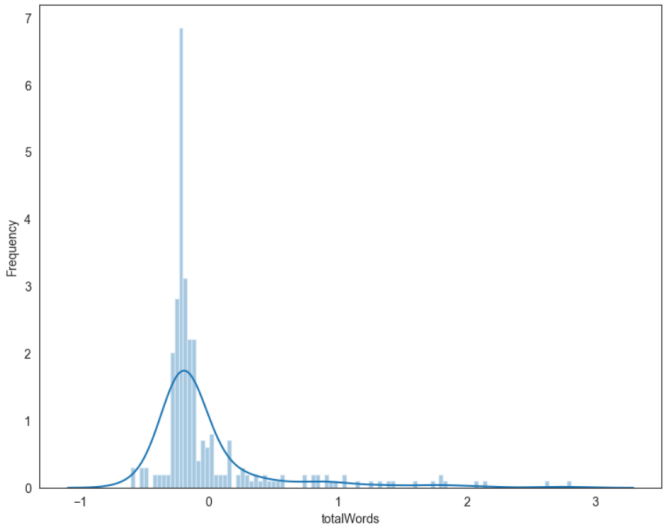


Figure A1.10: Words Volume in the last year of data

Words Volume Descriptive Statistics in the last year of data	
Mean	-0.010758620689655175
Standard Error	0.029379055463608056
Median	-0.19
Mode	-0.22
Standard Deviation	0.4994439428813369
Sample Variance	0.2503073809807899
Kurtosis	9.61184972410166
Skewness	2.9411586965134755
Range	3.42
Minimum	-0.61
Maximum	2.81
Sum	-3.1200000000000068
Count	290

A1.1.6 Positive Sentiment

Entire Dataset

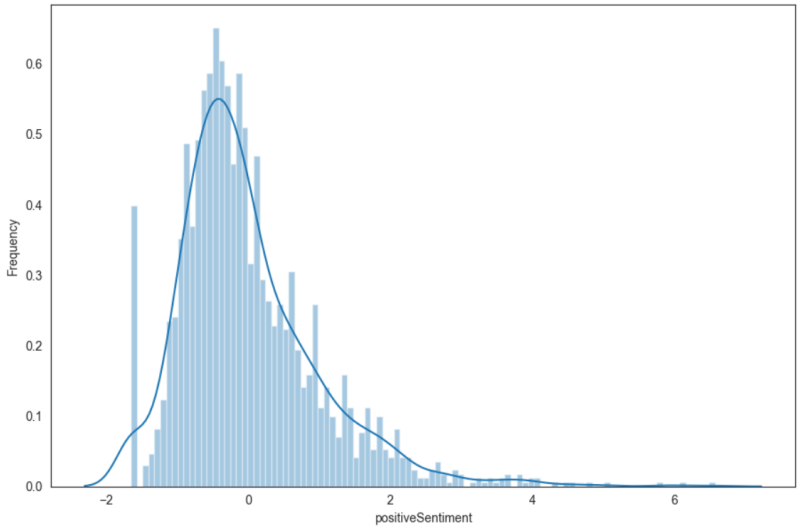


Figure A1.11: Positive Sentiment with entire dataset

Positive Sentiment Descriptive Statistics with entire dataset	
Mean	-0.00011577424023154774
Standard Error	0.021969944580020426
Median	-0.21
Mode	-1.65
Standard Deviation	1.0000548880773885
Sample Variance	1.0005924576323273
Kurtosis	4.061676439612937
Skewness	1.4873401549309522
Range	8.22
Minimum	-1.65
Maximum	6.57
Sum	-0.23999999999965826
Count	2073

Last Year

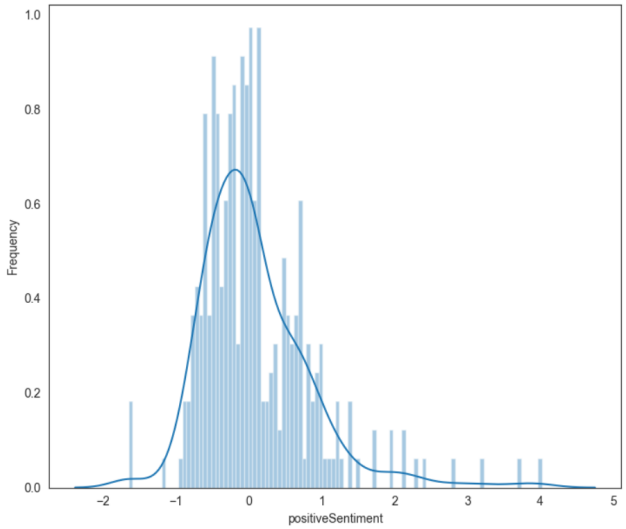


Figure A1.12: Positive Sentiment in the last year of data

Positive Sentiment Descriptive Statistics in the last year of data	
Mean	0.0863103448275862
Standard Error	0.04453127962901483
Median	-0.055
Mode	-0.1
Standard Deviation	0.7570317536932523
Sample Variance	0.5750801109652786
Kurtosis	5.009115638344465
Skewness	1.636685164397725
Range	5.67
Minimum	-1.65
Maximum	4.02
Sum	25.029999999999973
Count	290

A1.1.7 Negative Sentiment

Entire Dataset

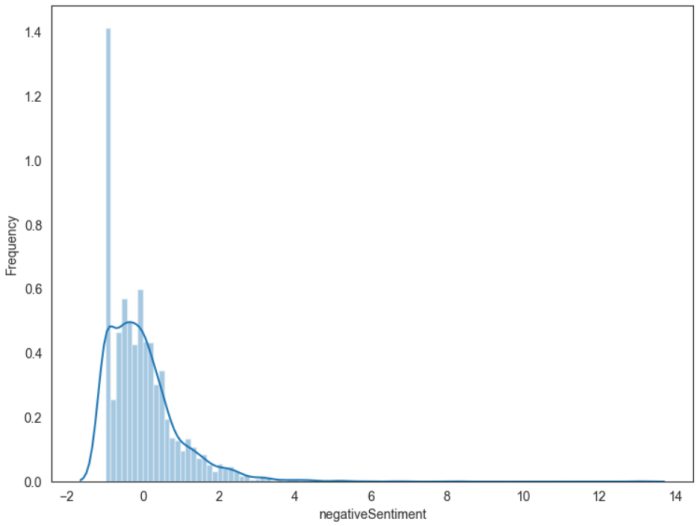


Figure A1.13: Negative Sentiment with entire dataset

Negative Sentiment Descriptive Statistics with entire dataset	
Mean	0.00014471780028943782
Standard Error	0.021964634770156394
Median	-0.17
Mode	-0.99
Standard Deviation	0.9998131896384166
Sample Variance	1.000108859355531
Kurtosis	19.513842113440127
Skewness	2.77912574439437
Range	14.05
Minimum	-0.99
Maximum	13.06
Sum	0.30000000000174065
Count	2073

Last Year

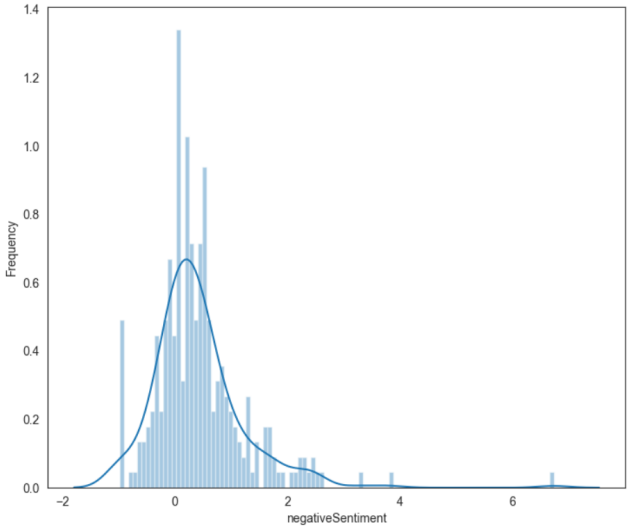


Figure A1.14: Negative Sentiment in the last year of data

Negative Sentiment Descriptive Statistics in the last year of data	
Mean	0.41293103448275864
Standard Error	0.04888104930864328
Median	0.27
Mode	0.08
Standard Deviation	0.8309778382469358
Sample Variance	0.6929135246390645
Kurtosis	11.895721323794636
Skewness	2.2637110757731076
Range	7.720000000000001
Minimum	-0.99
Maximum	6.73
Sum	119.74999999999997
Count	290

A1.2 Auto-Correlation

A1.2.1 Closing Prices

Closing Price AutoCorrelation with entire dataset		
Lag	Correlation	P-Value
1	0.9412	0.0
2	0.9109	0.0
3	0.8589	0.0
4	0.7893	0.0
5	0.7581	0.0

A1.2.2 Trading Volume

Trading Volume AutoCorrelation with entire dataset		
Lag	Correlation	P-Value
1	0.7671	0.0
2	0.6016	0.0
3	0.5088	0.0
4	0.4437	0.0
5	0.4851	0.0

A1.2.3 1 Day Returns

1 Day Return AutoCorrelation with entire dataset		
Lag	Correlation	P-Value
1	0.01	0.722
2	0.1342	0.0
3	0.1771	0.0
4	-0.2328	0.0
5	0.0306	0.2795

A1.2.4 Article Volume

Article Volume AutoCorrelation with entire dataset		
Lag	Correlation	P-Value
1	0.7087	0.0
2	0.5052	0.0
3	0.3913	0.0
4	0.3588	0.0
5	0.4346	0.0

A1.2.5 Word Volume

Word Volume AutoCorrelation with entire dataset		
Lag	Correlation	P-Value
1	0.0512	0.0198
2	0.0317	0.1497
3	0.037	0.0927
4	0.0106	0.6302
5	0.0285	0.1947

A1.2.6 Positive Sentiment

Positive Sentiment AutoCorrelation with entire dataset		
Lag	Correlation	P-Value
1	0.2189	0.0
2	0.167	0.0
3	0.1394	0.0
4	0.1571	0.0
5	0.1268	0.0

A1.2.7 Negative Sentiment

Negative Sentiment AutoCorrelation with entire dataset		
Lag	Correlation	P-Value
1	0.1936	0.0
2	0.1473	0.0
3	0.1388	0.0
4	0.1002	0.0
5	0.1903	0.0

A1.3 Return Vs Sentiment Correlation

A1.3.1 1 Day Return Vs Negative Sentiment

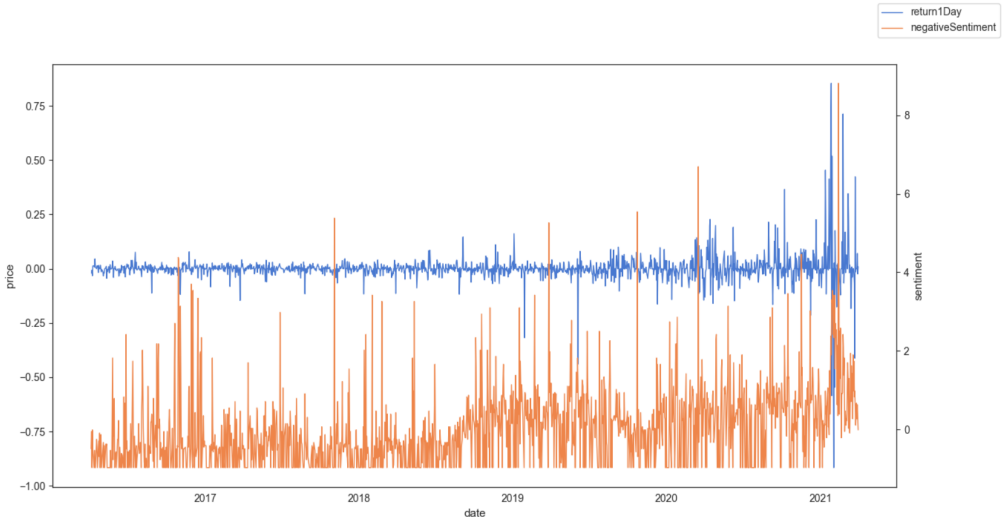


Figure A1.15: 1 Day Returns Vs Negative Sentiment with entire dataset

1 Day Return Vs Negative Sentiment Correlation with entire dataset				
	1 Day Return/Negative Sentiment		Negative Sentiment/1 Day Return	
Lag	Correlation	P-Value	Correlation	P-Value
0	-0.0057	0.8173	-0.0057	0.8173
1	0.0477	0.0554	-0.0199	0.4249
2	0.0318	0.2013	-0.0186	0.4542
3	0.0474	0.0571	-0.0319	0.2002
4	-0.007	0.7782	-0.0442	0.0757
5	0.0431	0.0834	-0.0444	0.0748

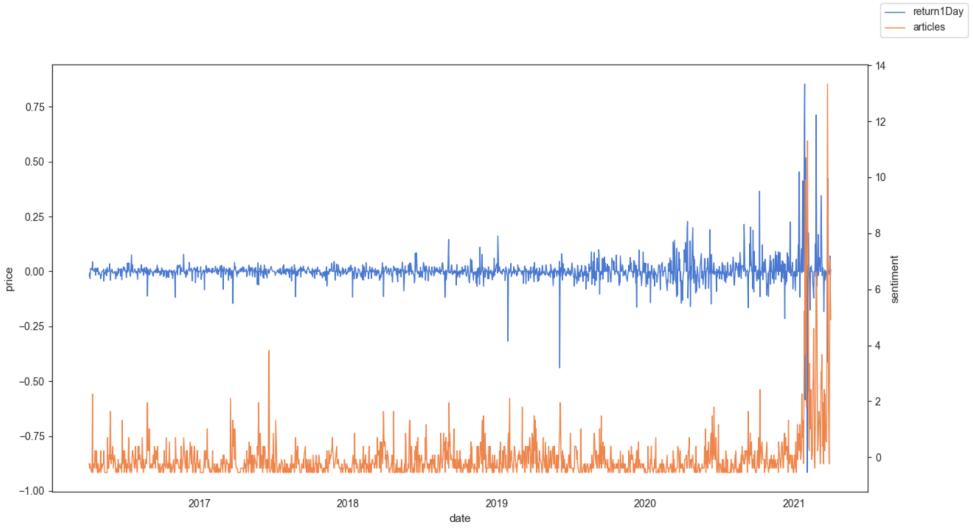


Figure A1.16: 1 Day Returns Vs Article Volume with entire dataset

A1.3.2 1 Day Return Vs Article Volume

1 Day Return Vs Article Volume Correlation with entire dataset				
	1 Day Return/Article Volume		Article Volume/1 Day Return	
Lag	Correlation	P-Value	Correlation	P-Value
0	-0.0316	0.2037	-0.0316	0.2037
1	-0.0344	0.1674	0.019	0.4449
2	0.0926	0.0002	0.0098	0.6924
3	0.0749	0.0026	-0.0354	0.1555
4	0.0402	0.1066	-0.07	0.0049
5	0.1267	0.0	-0.0679	0.0064

A1.3.3 1 Day Return Vs Negative to Positive Sentiment Ratio

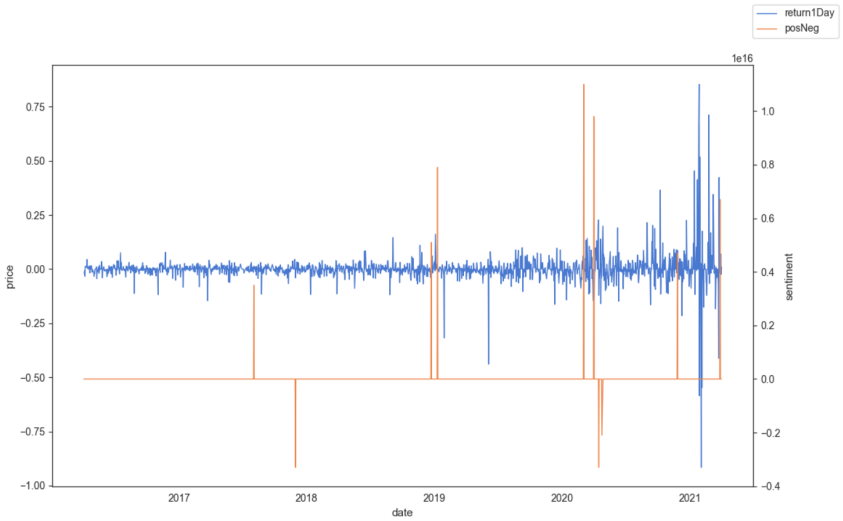


Figure A1.17: 1 Day Returns Vs Negative to Positive Sentiment Ratio with entire dataset

1 Day Return Vs Negative to Positive Sentiment Ratio Correlation with entire dataset				
	1 Day Return/Negative to Positive		Negative to Positive/1 Day Return	
Lag	Correlation	P-Value	Correlation	P-Value
0	-0.0152	0.5416	-0.0152	0.5416
1	-0.0015	0.9513	-0.0167	0.5031
2	-0.0476	0.0557	0.0177	0.4784
3	0.0721	0.0038	-0.0042	0.8674
4	-0.0615	0.0134	0.0187	0.4525
5	0.0071	0.7761	-0.0168	0.4997

A1.4 Vector Autoregression

A1.4.1 1 Day Return, Positive Sentiment

Lag Selection

lags	loglik	p(LR)	AIC	BIC	HQC
1	-167.03549		0.215352	0.235447	0.222813
2	-145.69666	0.00000	0.193773	0.227265	0.206207
3	-144.93663	0.82308	0.197805	0.244694	0.215213
4	-142.65017	0.33399	0.199938	0.260223	0.222319
5	-123.75024	0.00000	0.181394	0.255076	0.208749
6	-64.95396	0.00000	0.113197	0.200276	0.145525
7	-63.78198	0.67278	0.116717	0.217192	0.154018
8	-63.68900	0.99594	0.121579	0.235451	0.163855
9	-58.31330	0.02950	0.119867	0.247136	0.167116
10	-56.14951	0.36349	0.122152	0.262818	0.174375

Unit Root Test

1 Day Return Augmented Dickey-Fuller test for return1Day testing down from 6 lags, criterion AIC sample size 1611 unit-root null hypothesis: a = 1

test with constant including 5 lags of (1-L)return1Day model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.917446 test statistic: tau_c(1) = -17.9067 asymptotic p-value 5.185e-043 1st-order autocorrelation coeff. for e: 0.007 lagged differences: F(5, 1604) = 32.364 [0.0000]

Positive Sentiment Augmented Dickey-Fuller test for positiveSentiment testing down from 6 lags, criterion AIC sample size 1612 unit-root null hypothesis: a = 1

test with constant including 5 lags of (1-L)positiveSentiment model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.535408 test statistic: tau_c(1) = -12.0745 asymptotic p-value 5.969e-026 1st-order autocorrelation coeff. for e: -0.002 lagged differences: F(5, 1605) = 9.252 [0.0000]

Vector Autoregression

VAR system, lag order 6

OLS estimates, observations 2016-04-13–2022-06-15 ($T = 1611$)

Log-likelihood = −62.7906

Determinant of covariance matrix = 0.00370600

AIC = 0.1102

BIC = 0.1971

HQC = 0.1425

Portmanteau test: LB(48) = 340.694, df = 168 [0.0000]

Equation 1: return1Day

	Coefficient	Std. Error	t-ratio	p-value
const	0.00104449	0.00158960	0.6571	0.5112
return1Day$_{t-1}$	0.0532996	0.0242004	2.202	0.0278
return1Day$_{t-2}$	0.120869	0.0240410	5.028	0.0000
return1Day$_{t-3}$	0.00536865	0.0242304	0.2216	0.8247
return1Day$_{t-4}$	0.0291020	0.0242268	1.201	0.2298
return1Day$_{t-5}$	0.124562	0.0240438	5.181	0.0000
return1Day$_{t-6}$	−0.253442	0.0242196	−10.46	0.0000
positiveSentiment_1	0.000327211	0.00165166	0.1981	0.8430
positiveSentiment_2	0.00166040	0.00167586	0.9908	0.3219
positiveSentiment_3	−0.000924404	0.00168180	−0.5497	0.5826
positiveSentiment_4	0.000692590	0.00168187	0.4118	0.6805
positiveSentiment_5	0.00110231	0.00167603	0.6577	0.5108
positiveSentiment_6	−0.000182041	0.00165063	−0.1103	0.9122

Mean dependent var	0.001149	S.D. dependent var	0.066721
Sum squared resid	6.496350	S.E. of regression	0.063760
R^2	0.093595	Adjusted R^2	0.086788
$F(12, 1598)$	13.75066	P-value(F)	1.68e–27
$\hat{\rho}$	0.006756	Durbin–Watson	1.986226

F-tests of zero restrictions

All lags of return1Day	$F(6, 1598) = 27.2175$	[0.0000]
All lags of positiveSentiment	$F(6, 1598) = 0.342766$	[0.9143]
All vars, lag 6	$F(2, 1598) = 54.7514$	[0.0000]

Equation 2: positiveSentiment

	Coefficient	Std. Error	t-ratio	p-value
const	0.00140979	0.0239976	0.05875	0.9532

Continued on next page

Table A1.2 – *Continued from previous page*

	Coefficient	Std. Error	t-ratio	p-value
return1Day$_{t-1}$	0.286432	0.365343	0.7840	0.4332
return1Day$_{t-2}$	0.231605	0.362937	0.6381	0.5235
return1Day$_{t-3}$	−0.00497079	0.365796	−0.01359	0.9892
return1Day$_{t-4}$	0.0475274	0.365742	0.1299	0.8966
return1Day$_{t-5}$	0.428067	0.362979	1.179	0.2384
return1Day$_{t-6}$	0.219204	0.365634	0.5995	0.5489
positiveSentiment_1	0.190227	0.0249345	7.629	0.0000
positiveSentiment_2	0.0860239	0.0252998	3.400	0.0007
positiveSentiment_3	0.00283178	0.0253894	0.1115	0.9112
positiveSentiment_4	0.0254839	0.0253905	1.004	0.3157
positiveSentiment_5	0.0740993	0.0253024	2.929	0.0035
positiveSentiment_6	0.0819091	0.0249189	3.287	0.0010

Mean dependent var	0.003290	S.D. dependent var	1.000501
Sum squared resid	1480.565	S.E. of regression	0.962555
R^2	0.081316	Adjusted R^2	0.074417
$F(12, 1598)$	11.78705	P-value(F)	3.92e−23
$\hat{\rho}$	−0.002010	Durbin–Watson	2.003006

F-tests of zero restrictions

All lags of return1Day	$F(6, 1598) = 0.529305$	[0.7864]
All lags of positiveSentiment	$F(6, 1598) = 22.6225$	[0.0000]
All vars, lag 6	$F(2, 1598) = 5.56674$	[0.0039]

For the system as a whole —

Null hypothesis: the longest lag is 5

Alternative hypothesis: the longest lag is 6

Likelihood ratio test: $\chi^2_4 = 117.929$ [0.0000]

A1.4.2 1 Day Return and Negative Sentiment

Lag Selection

lags	loglik	p(LR)	AIC	BIC	HQC
1	-117.24868		0.153390	0.173485	0.160850
2	-83.29973	0.00000	0.116117	0.149608	0.128551
3	-73.30921	0.00050	0.108661	0.155550	0.126069
4	-52.32948	0.00000	0.087529	0.147814	0.109910
5	-23.21221	0.00000	0.056269	0.129951	0.083624
6	33.11379	0.00000	-0.008854	0.078225	0.023475
7	35.69730	0.27059	-0.007091	0.093385	0.030211
8	45.00797	0.00093	-0.013700	0.100172	0.028575
9	49.69306	0.05248	-0.014553	0.112716	0.032696
10	51.88213	0.35724	-0.012299	0.128367	0.039923

Unit Root Test

1 Day Return Augmented Dickey-Fuller test for return1Day testing down from 8 lags, criterion AIC sample size 1608 unit-root null hypothesis: a = 1

test with constant including 8 lags of (1-L)return1Day model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.944131 test statistic: tau_c(1) = -14.6599 asymptotic p-value 3.842e-034 1st-order autocorrelation coeff. for e: -0.003 lagged differences: F(8, 1598) = 21.197 [0.0000]

Negative Sentiment Augmented Dickey-Fuller test for negativeSentiment testing down from 8 lags, criterion AIC sample size 1610 unit-root null hypothesis: a = 1

test with constant including 7 lags of (1-L)negativeSentiment model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.313169 test statistic: tau_c(1) = -8.24341 asymptotic p-value 9.545e-014 1st-order autocorrelation coeff. for e: -0.002 lagged differences: F(7, 1601) = 20.863 [0.0000]

Vector Autoregression

VAR system, lag order 8

OLS estimates, observations 2016-04-15–2022-06-15 ($T = 1609$)

Log-likelihood = 46.9491

Determinant of covariance matrix = 0.00323375

AIC = −0.0161

BIC = 0.0977

HQC = 0.0261

Portmanteau test: LB(48) = 399.605, df = 160 [0.0000]

Equation 1: return1Day

	Coefficient	Std. Error	t-ratio	p-value
const	0.000987822	0.00158973	0.6214	0.5344
return1Day$_{t-1}$	0.0588697	0.0250364	2.351	0.0188
return1Day$_{t-2}$	0.117217	0.0251214	4.666	0.0000
return1Day$_{t-3}$	0.00727102	0.0244870	0.2969	0.7666
return1Day$_{t-4}$	0.0321519	0.0243106	1.323	0.1862
return1Day$_{t-5}$	0.125891	0.0242941	5.182	0.0000
return1Day$_{t-6}$	−0.250912	0.0244974	−10.24	0.0000
return1Day$_{t-7}$	0.0300316	0.0251173	1.196	0.2320
return1Day$_{t-8}$	0.00446900	0.0253934	0.1760	0.8603
negativeSentiment_1	−0.000570795	0.00176169	−0.3240	0.7460
negativeSentiment_2	−0.00146513	0.00179837	−0.8147	0.4154
negativeSentiment_3	−0.000745000	0.00180272	−0.4133	0.6795
negativeSentiment_4	−0.00261853	0.00179040	−1.463	0.1438
negativeSentiment_5	−0.00140985	0.00178944	−0.7879	0.4309
negativeSentiment_6	0.00214476	0.00180030	1.191	0.2337
negativeSentiment_7	−0.000698839	0.00179663	−0.3890	0.6973
negativeSentiment_8	0.00326075	0.00175921	1.854	0.0640

Mean dependent var	0.001119	S.D. dependent var	0.066753
Sum squared resid	6.462398	S.E. of regression	0.063713
R^2	0.098089	Adjusted R^2	0.089024
$F(16, 1592)$	10.82126	P-value(F)	8.07e–27
$\hat{\rho}$	0.000578	Durbin–Watson	1.998764

F-tests of zero restrictions

All lags of return1Day	$F(8, 1592) = 20.2874$	[0.0000]
All lags of negativeSentiment	$F(8, 1592) = 1.0975$	[0.3619]
All vars, lag 8	$F(2, 1592) = 1.72858$	[0.1779]

Equation 2: negativeSentiment

	Coefficient	Std. Error	t-ratio	p-value
const	7.90245e–005	0.0225083	0.003511	0.9972
return1Day$_{t-1}$	0.799396	0.354478	2.255	0.0243
return1Day$_{t-2}$	0.732989	0.355682	2.061	0.0395
return1Day$_{t-3}$	0.404381	0.346699	1.166	0.2436
return1Day$_{t-4}$	−0.374238	0.344203	−1.087	0.2771
return1Day$_{t-5}$	0.482715	0.343969	1.403	0.1607
return1Day$_{t-6}$	0.0975990	0.346847	0.2814	0.7784
return1Day$_{t-7}$	−0.0575599	0.355624	−0.1619	0.8714

Continued on next page

Table A1.4 – *Continued from previous page*

Coefficient	Std. Error	*t*-ratio	p-value	
return1Day$_{t-8}$	1.08066	0.359532	3.006	0.0027
negativeSentiment_1	0.208236	0.0249430	8.348	0.0000
negativeSentiment_2	0.0828018	0.0254623	3.252	0.0012
negativeSentiment_3	0.0307012	0.0255238	1.203	0.2292
negativeSentiment_4	0.105098	0.0253494	4.146	0.0000
negativeSentiment_5	0.117337	0.0253358	4.631	0.0000
negativeSentiment_6	0.0467817	0.0254896	1.835	0.0666
negativeSentiment_7	0.0332576	0.0254376	1.307	0.1913
negativeSentiment_8	0.0623372	0.0249078	2.503	0.0124

Mean dependent var	0.005382	S.D. dependent var	1.000916
Sum squared resid	1295.477	S.E. of regression	0.902077
R^2	0.195829	Adjusted R^2	0.187747
$F(16, 1592)$	24.22991	P-value(F)	$2.17\mathrm{e}{-64}$
$\hat{\rho}$	0.000219	Durbin–Watson	1.999505

F-tests of zero restrictions

All lags of return1Day	$F(8, 1592) = 3.09983$	[0.0018]
All lags of negativeSentiment	$F(8, 1592) = 44.948$	[0.0000]
All vars, lag 8	$F(2, 1592) = 7.53276$	[0.0006]

For the system as a whole —

Null hypothesis: the longest lag is 7

Alternative hypothesis: the longest lag is 8

Likelihood ratio test: $\chi^2_4 = 18.596$ [0.0009]

A1.4.3 1 Day Return and Article Volume

Lag Selection

lags	loglik	p(LR)	AIC	BIC	HQC
1	355.55334		-0.435038	-0.414943	-0.427578
2	393.91940	0.00000	-0.477809	-0.444317	-0.465375
3	399.65306	0.02178	-0.479966	-0.433078	-0.462559
4	416.64785	0.00000	-0.496139	-0.435854	-0.473758
5	498.71888	0.00000	-0.593303	-0.519621	-0.565948
6	591.91582	0.00000	-0.704313	-0.617235	-0.671985
7	603.23522	0.00015	-0.713423	-0.612948	-0.676121
8	603.57482	0.95387	-0.708867	-0.594995	-0.666592
9	616.39529	0.00004	-0.719845	-0.592576	-0.672596
10	618.08708	0.49580	-0.716972	-0.576307	-0.664750

Unit Root Test

1 Day Return Augmented Dickey-Fuller test for return1Day testing down from 9 lags, criterion AIC sample size 1607 unit-root null hypothesis: a = 1

test with constant including 9 lags of (1-L)return1Day model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.985703 test statistic: tau_c(1) = -14.3735 asymptotic p-value 2.87e-033 1st-order autocorrelation coeff. for e: 0.002 lagged differences: F(9, 1596) = 19.198 [0.0000]

Article Volume Augmented Dickey-Fuller test for articles testing down from 9 lags, criterion AIC sample size 1609 unit-root null hypothesis: a = 1

test with constant including 8 lags of (1-L)articles model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.153152 test statistic: tau_c(1) = -6.45545 asymptotic p-value 8.907e-009 1st-order autocorrelation coeff. for e: 0.001 lagged differences: F(8, 1599) = 26.622 [0.0000]

Vector Autoregression

VAR system, lag order 9
OLS estimates, observations 2016-04-18–2022-06-15 ($T = 1608$)

Log-likelihood = 617.755

Determinant of covariance matrix = 0.00158986

AIC = −0.7211

BIC = −0.5939

HQC = −0.6739

Portmanteau test: LB(48) = 505.855, df = 156 [0.0000]

Equation 1: return1Day

	Coefficient	Std. Error	*t*-ratio	p-value	
const	0.00108782		0.00158045	0.6883	0.4914
return1Day_{t-1}	0.0591637		0.0250827	2.359	0.0185
return1Day_{t-2}	0.122909		0.0251453	4.888	0.0000
return1Day_{t-3}	−0.0245893		0.0257675	−0.9543	0.3401
return1Day_{t-4}	0.0342852		0.0249822	1.372	0.1701
return1Day_{t-5}	0.129546		0.0248566	5.212	0.0000
return1Day_{t-6}	−0.267219		0.0251912	−10.61	0.0000
return1Day_{t-7}	0.0303273		0.0260085	1.166	0.2438
return1Day_{t-8}	0.0115134		0.0263921	0.4362	0.6627
return1Day_{t-9}	−0.0685212		0.0265911	−2.577	0.0101
articles_{t-1}	0.00766581		0.00247897	3.092	0.0020

Continued on next page

Table A1.5 – *Continued from previous page*

	Coefficient	Std. Error	t-ratio	p-value
articles_{t-2}	−0.000435273	0.00289421	−0.1504	0.8805
articles_{t-3}	−0.00383832	0.00286352	−1.340	0.1803
articles_{t-4}	0.00189756	0.00284012	0.6681	0.5042
articles_{t-5}	−0.00608534	0.00283780	−2.144	0.0322
articles_{t-6}	−0.00164996	0.00282611	−0.5838	0.5594
articles_{t-7}	−0.000697022	0.00286670	−0.2431	0.8079
articles_{t-8}	−0.00234650	0.00284151	−0.8258	0.4090
articles_{t-9}	0.00512786	0.00259083	1.979	0.0480

Mean dependent var	0.001117	S.D. dependent var	0.066774
Sum squared resid	6.364034	S.E. of regression	0.063286
R^2	0.111815	Adjusted R^2	0.101754
$F(18, 1589)$	11.11346	P-value(F)	$1.33\text{e}{-30}$
$\hat{\rho}$	−0.003284	Durbin–Watson	2.006407

F-tests of zero restrictions

All lags of return1Day	$F(9, 1589) = 19.5768$	[0.0000]
All lags of articles	$F(9, 1589) = 3.00166$	[0.0015]
All vars, lag 9	$F(2, 1589) = 5.60717$	[0.0037]

Equation 2: articles

	Coefficient	Std. Error	t-ratio	p-value
const	−0.000666807	0.0159719	−0.04175	0.9667
return1Day_{t-1}	0.216461	0.253484	0.8539	0.3933
return1Day_{t-2}	1.87414	0.254116	7.375	0.0000
return1Day_{t-3}	0.0851509	0.260405	0.3270	0.7437
return1Day_{t-4}	−0.528434	0.252468	−2.093	0.0365
return1Day_{t-5}	1.25754	0.251199	5.006	0.0000
return1Day_{t-6}	0.567181	0.254581	2.228	0.0260
return1Day_{t-7}	1.17170	0.262840	4.458	0.0000
return1Day_{t-8}	0.0307484	0.266717	0.1153	0.9082
return1Day_{t-9}	−0.340330	0.268727	−1.266	0.2055
articles_{t-1}	0.594105	0.0250523	23.71	0.0000
articles_{t-2}	0.00941176	0.0292487	0.3218	0.7477
articles_{t-3}	−0.0330034	0.0289385	−1.140	0.2543
articles_{t-4}	−0.0151086	0.0287021	−0.5264	0.5987
articles_{t-5}	0.0971494	0.0286786	3.388	0.0007
articles_{t-6}	0.203121	0.0285605	7.112	0.0000
articles_{t-7}	0.0639955	0.0289706	2.209	0.0273
articles_{t-8}	0.0280164	0.0287161	0.9756	0.3294
articles_{t-9}	−0.0970051	0.0261828	−3.705	0.0002

Mean dependent var	0.000485	S.D. dependent var	1.001466
Sum squared resid	649.9577	S.E. of regression	0.639559
R^2	0.596729	Adjusted R^2	0.592161
$F(18, 1589)$	130.6266	P-value(F)	3.0e–297
$\hat{\rho}$	0.001956	Durbin–Watson	1.995999

F-tests of zero restrictions

All lags of return1Day	$F(9, 1589) = 14.8106$	[0.0000]
All lags of articles	$F(9, 1589) = 232.346$	[0.0000]
All vars, lag 9	$F(2, 1589) = 7.40886$	[0.0006]

For the system as a whole —

Null hypothesis: the longest lag is 8

Alternative hypothesis: the longest lag is 9

Likelihood ratio test: $\chi^2_4 = 25.673$ [0.0000]

A1.4.4 1 Day Return, Article Volume and Negative Sentiment

Lag Selection

lags	loglik	p(LR)	AIC	BIC	HQC
1	-1762.40319		2.208342	2.248533	2.223263
2	-1702.40096	0.00000	2.144867	2.215200	2.170979
3	-1689.51977	0.00223	2.140037	2.240512	2.177339
4	-1657.76963	0.00000	2.111723	2.242341	2.160215
5	-1566.07760	0.00000	2.008808	2.169569	2.068491
6	-1469.58983	0.00000	1.899925	2.090828	1.970798
7	-1454.42045	0.00038	1.892247	2.113293	1.974311
8	-1442.12537	0.00346	1.888146	2.139334	1.981400
9	-1413.08214	0.00000	1.863201	2.144532	1.967646
10	-1407.79587	0.30615	1.867823	2.179296	1.983458

Unit Root Test

1 Day Return Augmented Dickey-Fuller test for return1Day testing down from 9 lags, criterion AIC sample size 1607 unit-root null hypothesis: a = 1

test with constant including 9 lags of (1-L)return1Day model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.985703 test statistic: tau_c(1) = -14.3735 asymptotic p-value 2.87e-033 1st-order autocorrelation coeff. for e: 0.002 lagged differences: F(9, 1596) = 19.198 [0.0000]

Article Volume Augmented Dickey-Fuller test for articles testing down from 9 lags, criterion AIC sample size 1609 unit-root null hypothesis: a = 1

test with constant including 8 lags of (1-L)articles model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.153152 test statistic: tau_c(1) = -6.45545 asymptotic p-value 8.907e-009 1st-order autocorrelation coeff. for e: 0.001 lagged differences: F(8, 1599) = 26.622 [0.0000]

Negative Sentiment Augmented Dickey-Fuller test for negativeSentiment testing down from 9 lags, criterion AIC sample size 1610 unit-root null hypothesis: a = 1

test with constant including 7 lags of (1-L)negativeSentiment model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.313169 test statistic: tau_c(1) = -8.24341 asymptotic p-value 9.545e-014 1st-order autocorrelation coeff. for e: -0.002 lagged differences: F(7, 1601) = 20.863 [0.0000]

Vector Autoregression

VAR system, lag order 9
OLS estimates, observations 2016-04-18–2022-06-15 ($T = 1608$)

Log-likelihood = −1412.57

Determinant of covariance matrix = 0.00116306

AIC = 1.8614

BIC = 2.1426

HQC = 1.9658

Portmanteau test: LB(48) = 748.282, df = 351 [0.0000]

Equation 1: return1Day

	Coefficient	Std. Error	*t*-ratio	p-value
const	0.00110917	0.00157925	0.7023	0.4826
return1Day$_{t-1}$	0.0588840	0.0251373	2.342	0.0193
return1Day$_{t-2}$	0.122661	0.0252066	4.866	0.0000
return1Day$_{t-3}$	−0.0233277	0.0258477	−0.9025	0.3669
return1Day$_{t-4}$	0.0372937	0.0250639	1.488	0.1370
return1Day$_{t-5}$	0.134711	0.0249507	5.399	0.0000
return1Day$_{t-6}$	−0.265540	0.0252720	−10.51	0.0000
return1Day$_{t-7}$	0.0307187	0.0260704	1.178	0.2389
return1Day$_{t-8}$	0.0103033	0.0264720	0.3892	0.6972
return1Day$_{t-9}$	−0.0723152	0.0267091	−2.708	0.0069
negativeSentiment_1	−0.00160754	0.00180978	−0.8883	0.3745
negativeSentiment_2	−0.00159062	0.00183897	−0.8650	0.3872
negativeSentiment_3	−0.000386668	0.00184061	−0.2101	0.8336
negativeSentiment_4	−0.00300356	0.00184138	−1.631	0.1031
negativeSentiment_5	−0.000796881	0.00183811	−0.4335	0.6647

Continued on next page

Table A1.7 – *Continued from previous page*

	Coefficient	Std. Error	*t*-ratio	p-value
negativeSentiment_6	0.00271652	0.00183884	1.477	0.1398
negativeSentiment_7	−0.000558257	0.00184224	−0.3030	0.7619
negativeSentiment_8	0.00424485	0.00183928	2.308	0.0211
negativeSentiment_9	−0.000988880	0.00181023	−0.5463	0.5850
$articles_{t-1}$	0.00845481	0.00254737	3.319	0.0009
$articles_{t-2}$	−0.000121030	0.00297328	−0.04071	0.9675
$articles_{t-3}$	−0.00404796	0.00293513	−1.379	0.1680
$articles_{t-4}$	0.00243713	0.00291361	0.8365	0.4030
$articles_{t-5}$	−0.00571170	0.00291214	−1.961	0.0500
$articles_{t-6}$	−0.00257950	0.00290198	−0.8889	0.3742
$articles_{t-7}$	6.63167e–005	0.00294610	0.02251	0.9820
$articles_{t-8}$	−0.00393303	0.00291957	−1.347	0.1781
$articles_{t-9}$	0.00601433	0.00267566	2.248	0.0247

Mean dependent var	0.001117	S.D. dependent var	0.066774
Sum squared resid	6.315724	S.E. of regression	0.063224
R^2	0.118558	Adjusted R^2	0.103495
$F(27, 1580)$	7.870976	P-value(F)	1.64e–28
$\hat{\rho}$	−0.003191	Durbin–Watson	2.006268

F-tests of zero restrictions

All lags of return1Day	$F(9, 1580) = 19.6809$	[0.0000]
All lags of negativeSentiment	$F(9, 1580) = 1.34285$	[0.2096]
All lags of articles	$F(9, 1580) = 3.3141$	[0.0005]
All vars, lag 9	$F(3, 1580) = 4.38368$	[0.0044]

Equation 2: negativeSentiment

	Coefficient	Std. Error	*t*-ratio	p-value
const	0.00489564	0.0223345	0.2192	0.8265
$return1Day_{t-1}$	0.807408	0.355504	2.271	0.0233
$return1Day_{t-2}$	0.845346	0.356484	2.371	0.0178
$return1Day_{t-3}$	0.238618	0.365551	0.6528	0.5140
$return1Day_{t-4}$	−0.355834	0.354465	−1.004	0.3156
$return1Day_{t-5}$	0.801151	0.352864	2.270	0.0233
$return1Day_{t-6}$	0.225387	0.357409	0.6306	0.5284
$return1Day_{t-7}$	−0.289744	0.368699	−0.7859	0.4321
$return1Day_{t-8}$	1.06455	0.374380	2.844	0.0045
$return1Day_{t-9}$	−0.218110	0.377732	−0.5774	0.5637
negativeSentiment_1	0.188032	0.0255947	7.346	0.0000
negativeSentiment_2	0.0782485	0.0260076	3.009	0.0027
negativeSentiment_3	0.0254832	0.0260307	0.9790	0.3277

Continued on next page

Table A1.8 – *Continued from previous page*

	Coefficient	Std. Error	*t*-ratio	p-value
negativeSentiment_4	0.0868601	0.0260416	3.335	0.0009
negativeSentiment_5	0.0951837	0.0259954	3.662	0.0003
negativeSentiment_6	0.0380987	0.0260057	1.465	0.1431
negativeSentiment_7	0.0281676	0.0260539	1.081	0.2798
negativeSentiment_8	0.0442766	0.0260120	1.702	0.0889
negativeSentiment_9	−0.00355533	0.0256011	−0.1389	0.8896
articles_{t-1}	0.0527335	0.0360261	1.464	0.1435
articles_{t-2}	0.00323561	0.0420496	0.07695	0.9387
articles_{t-3}	−0.0588831	0.0415099	−1.419	0.1562
articles_{t-4}	0.00965100	0.0412056	0.2342	0.8148
articles_{t-5}	0.0776235	0.0411849	1.885	0.0596
articles_{t-6}	−0.00285362	0.0410411	−0.06953	0.9446
articles_{t-7}	−0.0730910	0.0416650	−1.754	0.0796
articles_{t-8}	0.0119000	0.0412899	0.2882	0.7732
articles_{t-9}	0.157644	0.0378404	4.166	0.0000

Mean dependent var	0.005840	S.D. dependent var	1.001059
Sum squared resid	1263.203	S.E. of regression	0.894145
R^2	0.215600	Adjusted R^2	0.202196
$F(27, 1580)$	16.08440	P-value(F)	2.60e–65
$\hat{\rho}$	−0.005317	Durbin–Watson	2.010178

F-tests of zero restrictions

All lags of return1Day	$F(9, 1580) = 3.16674$	[0.0008]
All lags of negativeSentiment	$F(9, 1580) = 21.2197$	[0.0000]
All lags of articles	$F(9, 1580) = 4.18189$	[0.0000]
All vars, lag 9	$F(3, 1580) = 6.22988$	[0.0003]

Equation 3: articles

	Coefficient	Std. Error	*t*-ratio	p-value
const	−0.00159515	0.0159151	−0.1002	0.9202
return1Day_{t-1}	0.238048	0.253325	0.9397	0.3475
return1Day_{t-2}	1.89397	0.254023	7.456	0.0000
return1Day_{t-3}	0.0534964	0.260484	0.2054	0.8373
return1Day_{t-4}	−0.546437	0.252584	−2.163	0.0307
return1Day_{t-5}	1.19810	0.251444	4.765	0.0000
return1Day_{t-6}	0.572174	0.254682	2.247	0.0248
return1Day_{t-7}	1.19301	0.262728	4.541	0.0000
return1Day_{t-8}	0.0457508	0.266775	0.1715	0.8639
return1Day_{t-9}	−0.294299	0.269164	−1.093	0.2744
negativeSentiment_1	−0.0251158	0.0182383	−1.377	0.1687

Continued on next page

Table A1.9 – *Continued from previous page*

	Coefficient	Std. Error	*t*-ratio	p-value
negativeSentiment_2	0.0419922	0.0185325	2.266	0.0236
negativeSentiment_3	0.0234041	0.0185490	1.262	0.2072
negativeSentiment_4	0.0334652	0.0185567	1.803	0.0715
negativeSentiment_5	0.00653985	0.0185238	0.3531	0.7241
negativeSentiment_6	−0.00671100	0.0185311	−0.3621	0.7173
negativeSentiment_7	−0.0186727	0.0185654	−1.006	0.3147
negativeSentiment_8	−0.00132041	0.0185356	−0.07124	0.9432
negativeSentiment_9	0.0337236	0.0182428	1.849	0.0647
articles_{t-1}	0.596795	0.0256714	23.25	0.0000
articles_{t-2}	−0.00362183	0.0299636	−0.1209	0.9038
articles_{t-3}	−0.0346455	0.0295791	−1.171	0.2417
articles_{t-4}	−0.0269349	0.0293623	−0.9173	0.3591
articles_{t-5}	0.0941144	0.0293475	3.207	0.0014
articles_{t-6}	0.203247	0.0292450	6.950	0.0000
articles_{t-7}	0.0626147	0.0296896	2.109	0.0351
articles_{t-8}	0.0254083	0.0294223	0.8636	0.3880
articles_{t-9}	−0.108868	0.0269643	−4.037	0.0001

Mean dependent var	0.000485	S.D. dependent var	1.001466
Sum squared resid	641.4155	S.E. of regression	0.637149
R^2	0.602029	Adjusted R^2	0.595228
$F(27, 1580)$	88.52368	P-value(F)	$1.5\text{e}{-292}$
$\hat{\rho}$	0.000441	Durbin–Watson	1.999066

F-tests of zero restrictions

All lags of return1Day	$F(9, 1580) = 14.7182$	[0.0000]
All lags of negativeSentiment	$F(9, 1580) = 2.33798$	[0.0129]
All lags of articles	$F(9, 1580) = 171.892$	[0.0000]
All vars, lag 9	$F(3, 1580) = 6.01536$	[0.0005]

For the system as a whole —

Null hypothesis: the longest lag is 8

Alternative hypothesis: the longest lag is 9

Likelihood ratio test: $\chi^2_9 = 58.113$ [0.0000]

A1.4.5 1 Day Return, Article Volume, Negative Sentiment and Positive Sentiment

Lag Selection

lags	loglik	p(LR)	AIC	BIC	HQC
1	-3683.56116		4.609286	4.676269	4.634154
2	-3616.05558	0.00000	4.545184	4.665755	4.589946
3	-3599.05625	0.00544	4.543941	4.718098	4.608597
4	-3563.43020	0.00000	4.519515	4.747259	4.604065
5	-3464.20650	0.00000	4.415938	4.697269	4.520383
6	-3363.43830	0.00000	4.310440	4.645357	4.434778
7	-3344.44683	0.00152	4.306717	4.695221	4.450950
8	-3325.07122	0.00118	4.302516	4.744607	4.466643
9	-3291.12808	0.00000	4.280184	4.775862	4.464206
10	-3274.26040	0.00590	4.279104	4.828369	4.483020

Unit Root Test

1 Day Return Augmented Dickey-Fuller test for return1Day testing down from 10 lags, criterion AIC sample size 1606 unit-root null hypothesis: a = 1

test with constant including 10 lags of (1-L)return1Day model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.934753 test statistic: tau_c(1) = -12.8346 asymptotic p-value 2.049e-028 1st-order autocorrelation coeff. for e: -0.003 lagged differences: F(10, 1594) = 17.730 [0.0000]

Article Volume Augmented Dickey-Fuller test for volume testing down from 10 lags, criterion AIC sample size 1607 unit-root null hypothesis: a = 1

test with constant including 10 lags of (1-L)volume model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.161774 test statistic: tau_c(1) = -5.64743 asymptotic p-value 8.052e-007 1st-order autocorrelation coeff. for e: -0.001 lagged differences: F(10, 1595) = 44.269 [0.0000]

Negative Sentiment Augmented Dickey-Fuller test for negativeSentiment testing down from 10 lags, criterion AIC sample size 1607 unit-root null hypothesis: a = 1

test with constant including 10 lags of (1-L)negativeSentiment model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.273223 test statistic: tau_c(1) = -6.80401 asymptotic p-value 1.107e-009 1st-order autocorrelation coeff. for e: -0.008 lagged differences: F(10, 1595) = 16.081 [0.0000]

Positive Sentiment Augmented Dickey-Fuller test for positiveSentiment testing down from 10 lags, criterion AIC sample size 1612 unit-root null hypothesis: a = 1

test with constant including 5 lags of (1-L)positiveSentiment model: (1-L)y = b0 + (a-1)*y(-1) + ... + e estimated value of (a - 1): -0.535408 test statistic: tau_c(1) = -12.0745 asymptotic p-value 5.969e-026 1st-order autocorrelation coeff. for e: -0.002 lagged differences: F(5, 1605) = 9.252 [0.0000]

Vector Autoregression

VAR system, lag order 10
OLS estimates, observations 2016-04-19–2022-06-15 ($T = 1607$)

Log-likelihood = −3274.26

Determinant of covariance matrix = 0.000691594

AIC = 4.2791

BIC = 4.8284

HQC = 4.4830

Portmanteau test: LB(48) = 1028.53, df = 608 [0.0000]

Equation 1: return1Day

	Coefficient	Std. Error	*t*-ratio	p-value
const	0.00118395	0.00157730	0.7506	0.4530
return1Day$_{t-1}$	0.0562900	0.0253065	2.224	0.0263
return1Day$_{t-2}$	0.116351	0.0252815	4.602	0.0000
return1Day$_{t-3}$	−0.0238123	0.0259221	−0.9186	0.3584
return1Day$_{t-4}$	0.0301352	0.0259380	1.162	0.2455
return1Day$_{t-5}$	0.136387	0.0251458	5.424	0.0000
return1Day$_{t-6}$	−0.261666	0.0254372	−10.29	0.0000
return1Day$_{t-7}$	0.0340623	0.0261742	1.301	0.1933
return1Day$_{t-8}$	0.00990771	0.0267374	0.3706	0.7110
return1Day$_{t-9}$	−0.0726555	0.0268730	−2.704	0.0069
return1Day$_{t-10}$	−0.0218758	0.0268376	−0.8151	0.4151
negativeSentiment_1	−0.00179594	0.00214768	−0.8362	0.4032
negativeSentiment_2	−0.00371331	0.00217510	−1.707	0.0880
negativeSentiment_3	6.61488e–005	0.00216911	0.03050	0.9757
negativeSentiment_4	−0.00513745	0.00216463	−2.373	0.0177
negativeSentiment_5	−0.00297463	0.00217806	−1.366	0.1722
negativeSentiment_6	0.00317858	0.00217013	1.465	0.1432
negativeSentiment_7	−0.00160495	0.00216564	−0.7411	0.4587
negativeSentiment_8	0.00592625	0.00216774	2.734	0.0063
negativeSentiment_9	−0.00324044	0.00216991	−1.493	0.1355
negativeSentiment_10	0.00186277	0.00213059	0.8743	0.3821
articles$_{t-1}$	0.00924184	0.00259425	3.562	0.0004

Continued on next page

Table A1.10 – *Continued from previous page*

	Coefficient	Std. Error	t-ratio	p-value
articles$_{t-2}$	−0.000409975	0.00301193	−0.1361	0.8917
articles$_{t-3}$	−0.00353701	0.00300983	−1.175	0.2401
articles$_{t-4}$	0.00152645	0.00297488	0.5131	0.6079
articles$_{t-5}$	−0.00612817	0.00296394	−2.068	0.0388
articles$_{t-6}$	−0.00229855	0.00296309	−0.7757	0.4380
articles$_{t-7}$	−0.000198006	0.00298212	−0.06640	0.9471
articles$_{t-8}$	−0.00311112	0.00298890	−1.041	0.2981
articles$_{t-9}$	0.00270204	0.00306538	0.8815	0.3782
articles$_{t-10}$	0.00402751	0.00275266	1.463	0.1436
positiveSentiment_1	0.000143441	0.00199387	0.07194	0.9427
positiveSentiment_2	0.00308837	0.00203294	1.519	0.1289
positiveSentiment_3	−0.000618748	0.00203465	−0.3041	0.7611
positiveSentiment_4	0.00358024	0.00203282	1.761	0.0784
positiveSentiment_5	0.00380140	0.00203166	1.871	0.0615
positiveSentiment_6	−0.00111559	0.00203695	−0.5477	0.5840
positiveSentiment_7	0.00169063	0.00203787	0.8296	0.4069
positiveSentiment_8	−0.00318486	0.00203692	−1.564	0.1181
positiveSentiment_9	0.00399601	0.00203890	1.960	0.0502
positiveSentiment_10	−0.00159573	0.00200646	−0.7953	0.4266

Mean dependent var	0.001117	S.D. dependent var	0.066795
Sum squared resid	6.233678	S.E. of regression	0.063092
R^2	0.130008	Adjusted R^2	0.107786
$F(40, 1566)$	5.850417	P-value(F)	7.63e–27
$\hat{\rho}$	−0.001473	Durbin–Watson	2.002799

F-tests of zero restrictions

All lags of return1Day	$F(10, 1566) = 17.1961$	[0.0000]
All lags of negativeSentiment	$F(10, 1566) = 2.47678$	[0.0061]
All lags of articles	$F(10, 1566) = 3.02778$	[0.0008]
All lags of positiveSentiment	$F(10, 1566) = 1.76999$	[0.0612]
All vars, lag 10	$F(4, 1566) = 1.02613$	[0.3924]

Equation 2: negativeSentiment

	Coefficient	Std. Error	t-ratio	p-value
const	0.00523439	0.0222995	0.2347	0.8144
return1Day$_{t-1}$	0.786887	0.357778	2.199	0.0280
return1Day$_{t-2}$	0.847649	0.357425	2.372	0.0178
return1Day$_{t-3}$	0.205693	0.366481	0.5613	0.5747
return1Day$_{t-4}$	−0.237628	0.366706	−0.6480	0.5171
return1Day$_{t-5}$	0.861627	0.355506	2.424	0.0155

Continued on next page

Table A1.11 – *Continued from previous page*

	Coefficient	Std. Error	*t*-ratio	p-value
return1Day$_{t-6}$	0.299377	0.359625	0.8325	0.4053
return1Day$_{t-7}$	−0.214102	0.370045	−0.5786	0.5630
return1Day$_{t-8}$	1.08794	0.378007	2.878	0.0041
return1Day$_{t-9}$	−0.243090	0.379925	−0.6398	0.5224
return1Day$_{t-10}$	0.358822	0.379424	0.9457	0.3444
negativeSentiment_1	0.195229	0.0303635	6.430	0.0000
negativeSentiment_2	0.0460839	0.0307510	1.499	0.1342
negativeSentiment_3	0.0261475	0.0306663	0.8526	0.3940
negativeSentiment_4	0.119977	0.0306030	3.920	0.0001
negativeSentiment_5	0.0621072	0.0307930	2.017	0.0439
negativeSentiment_6	0.0405198	0.0306808	1.321	0.1868
negativeSentiment_7	0.0266039	0.0306174	0.8689	0.3850
negativeSentiment_8	0.0533139	0.0306470	1.740	0.0821
negativeSentiment_9	0.0162801	0.0306778	0.5307	0.5957
negativeSentiment_10	0.0217559	0.0301218	0.7223	0.4702
articles$_{t-1}$	0.0671244	0.0366770	1.830	0.0674
articles$_{t-2}$	−0.00504745	0.0425820	−0.1185	0.9057
articles$_{t-3}$	−0.0702482	0.0425523	−1.651	0.0990
articles$_{t-4}$	0.00979854	0.0420582	0.2330	0.8158
articles$_{t-5}$	0.0552098	0.0419036	1.318	0.1878
articles$_{t-6}$	0.000551421	0.0418915	0.01316	0.9895
articles$_{t-7}$	−0.0829506	0.0421605	−1.967	0.0493
articles$_{t-8}$	0.0128108	0.0422564	0.3032	0.7618
articles$_{t-9}$	0.114493	0.0433377	2.642	0.0083
articles$_{t-10}$	0.0959632	0.0389166	2.466	0.0138
positiveSentiment_1	−0.0240134	0.0281890	−0.8519	0.3944
positiveSentiment_2	0.0488851	0.0287413	1.701	0.0892
positiveSentiment_3	0.00134272	0.0287655	0.04668	0.9628
positiveSentiment_4	−0.0549748	0.0287396	−1.913	0.0559
positiveSentiment_5	0.0586290	0.0287232	2.041	0.0414
positiveSentiment_6	−0.00724479	0.0287979	−0.2516	0.8014
positiveSentiment_7	0.00511095	0.0288109	0.1774	0.8592
positiveSentiment_8	−0.0147288	0.0287976	−0.5115	0.6091
positiveSentiment_9	−0.0303368	0.0288255	−1.052	0.2928
positiveSentiment_10	−0.0523216	0.0283669	−1.844	0.0653

Mean dependent var	0.005999	S.D. dependent var	1.001351
Sum squared resid	1245.968	S.E. of regression	0.891985
R^2	0.226271	Adjusted R^2	0.206507
$F(40, 1566)$	11.44909	P-value(F)	3.48e–62
$\hat{\rho}$	0.000030	Durbin–Watson	1.999705

F-tests of zero restrictions

All lags of return1Day	$F(10, 1566) = 3.10478$	[0.0006]
All lags of negativeSentiment	$F(10, 1566) = 12.7499$	[0.0000]
All lags of articles	$F(10, 1566) = 4.21157$	[0.0000]
All lags of positiveSentiment	$F(10, 1566) = 1.61363$	[0.0970]
All vars, lag 10	$F(4, 1566) = 2.2552$	[0.0611]

Equation 3: articles

	Coefficient	Std. Error	t-ratio	p-value
const	−0.00169222	0.0159133	−0.1063	0.9153
return1Day$_{t-1}$	0.281539	0.255317	1.103	0.2703
return1Day$_{t-2}$	1.89445	0.255064	7.427	0.0000
return1Day$_{t-3}$	0.0544912	0.261527	0.2084	0.8350
return1Day$_{t-4}$	−0.566226	0.261688	−2.164	0.0306
return1Day$_{t-5}$	1.22429	0.253695	4.826	0.0000
return1Day$_{t-6}$	0.593986	0.256635	2.315	0.0208
return1Day$_{t-7}$	1.24026	0.264071	4.697	0.0000
return1Day$_{t-8}$	0.0637254	0.269752	0.2362	0.8133
return1Day$_{t-9}$	−0.288372	0.271121	−1.064	0.2877
return1Day$_{t-10}$	−0.0190537	0.270764	−0.07037	0.9439
negativeSentiment_1	−0.00109565	0.0216679	−0.05057	0.9597
negativeSentiment_2	0.0448732	0.0219445	2.045	0.0410
negativeSentiment_3	0.0439229	0.0218840	2.007	0.0449
negativeSentiment_4	0.0394396	0.0218388	1.806	0.0711
negativeSentiment_5	0.0105943	0.0219744	0.4821	0.6298
negativeSentiment_6	−0.00269474	0.0218944	−0.1231	0.9021
negativeSentiment_7	−0.0212422	0.0218491	−0.9722	0.3311
negativeSentiment_8	−0.0104733	0.0218703	−0.4789	0.6321
negativeSentiment_9	0.0337346	0.0218922	1.541	0.1235
negativeSentiment_10	0.0270232	0.0214954	1.257	0.2089
articles$_{t-1}$	0.598591	0.0261733	22.87	0.0000
articles$_{t-2}$	−0.00523384	0.0303873	−0.1722	0.8633
articles$_{t-3}$	−0.0299992	0.0303661	−0.9879	0.3233
articles$_{t-4}$	−0.0319469	0.0300135	−1.064	0.2873
articles$_{t-5}$	0.0964800	0.0299031	3.226	0.0013
articles$_{t-6}$	0.200202	0.0298945	6.697	0.0000
articles$_{t-7}$	0.0599664	0.0300865	1.993	0.0464

Continued on next page

Table A1.12 – *Continued from previous page*

	Coefficient	Std. Error	t-ratio	p-value
articles_{t-8}	0.0235524	0.0301549	0.7810	0.4349
articles_{t-9}	−0.120027	0.0309265	−3.881	0.0001
articles_{t-10}	0.0131162	0.0277715	0.4723	0.6368
positiveSentiment_1	−0.0434596	0.0201161	−2.160	0.0309
positiveSentiment_2	−0.0107416	0.0205103	−0.5237	0.6006
positiveSentiment_3	−0.0385023	0.0205275	−1.876	0.0609
positiveSentiment_4	−0.0119470	0.0205091	−0.5825	0.5603
positiveSentiment_5	−0.0140884	0.0204974	−0.6873	0.4920
positiveSentiment_6	−0.0123161	0.0205507	−0.5993	0.5491
positiveSentiment_7	0.00722059	0.0205600	0.3512	0.7255
positiveSentiment_8	0.0130336	0.0205504	0.6342	0.5260
positiveSentiment_9	−0.00537420	0.0205704	−0.2613	0.7939
positiveSentiment_10	−0.00866925	0.0202431	−0.4283	0.6685

Mean dependent var	0.000631	S.D. dependent var	1.001760
Sum squared resid	634.5093	S.E. of regression	0.636536
R^2	0.606301	Adjusted R^2	0.596244
$F(40, 1566)$	60.29136	P-value(F)	$7.8\text{e}{-284}$
$\hat{\rho}$	−0.000583	Durbin–Watson	2.001165

F-tests of zero restrictions

All lags of return1Day	$F(10, 1566) = 13.504$	[0.0000]
All lags of negativeSentiment	$F(10, 1566) = 2.73512$	[0.0024]
All lags of articles	$F(10, 1566) = 149.159$	[0.0000]
All lags of positiveSentiment	$F(10, 1566) = 1.4833$	[0.1395]
All vars, lag 10	$F(4, 1566) = 0.541814$	[0.7050]

Equation 4: positiveSentiment

	Coefficient	Std. Error	t-ratio	p-value
const	0.00132996	0.0240386	0.05533	0.9559
return1Day_{t-1}	0.496963	0.385681	1.289	0.1978
return1Day_{t-2}	0.317541	0.385299	0.8241	0.4100
return1Day_{t-3}	−0.0945202	0.395062	−0.2393	0.8109
return1Day_{t-4}	0.101235	0.395304	0.2561	0.7979
return1Day_{t-5}	0.428240	0.383231	1.117	0.2640
return1Day_{t-6}	−0.00843046	0.387672	−0.02175	0.9827
return1Day_{t-7}	0.104297	0.398904	0.2615	0.7938
return1Day_{t-8}	0.0101039	0.407487	0.02480	0.9802
return1Day_{t-9}	−0.357616	0.409555	−0.8732	0.3827
return1Day_{t-10}	−0.0719904	0.409015	−0.1760	0.8603
negativeSentiment_1	0.00482949	0.0327315	0.1475	0.8827

Continued on next page

Table A1.13 – *Continued from previous page*

Coefficient		Std. Error	*t*-ratio	p-value
negativeSentiment_2	−0.0129106	0.0331492	−0.3895	0.6970
negativeSentiment_3	−0.00391991	0.0330580	−0.1186	0.9056
negativeSentiment_4	0.0769950	0.0329897	2.334	0.0197
negativeSentiment_5	0.0516165	0.0331945	1.555	0.1202
negativeSentiment_6	0.0164526	0.0330736	0.4975	0.6189
negativeSentiment_7	−0.0350162	0.0330052	−1.061	0.2889
negativeSentiment_8	0.0328842	0.0330372	0.9954	0.3197
negativeSentiment_9	−0.00178332	0.0330703	−0.05393	0.9570
negativeSentiment_10	0.0129688	0.0324709	0.3994	0.6897
articles_{t-1}	0.0319561	0.0395374	0.8082	0.4191
articles_{t-2}	0.00802012	0.0459029	0.1747	0.8613
articles_{t-3}	−0.0397942	0.0458709	−0.8675	0.3858
articles_{t-4}	0.0585206	0.0453382	1.291	0.1970
articles_{t-5}	−0.0172290	0.0451715	−0.3814	0.7029
articles_{t-6}	−0.0109665	0.0451585	−0.2428	0.8082
articles_{t-7}	0.0591901	0.0454485	1.302	0.1930
articles_{t-8}	−0.0315024	0.0455519	−0.6916	0.4893
articles_{t-9}	0.0533770	0.0467175	1.143	0.2534
articles_{t-10}	−0.0762418	0.0419516	−1.817	0.0694
positiveSentiment_1	0.172450	0.0303874	5.675	0.0000
positiveSentiment_2	0.0838972	0.0309828	2.708	0.0068
positiveSentiment_3	0.00353683	0.0310088	0.1141	0.9092
positiveSentiment_4	−0.0291545	0.0309809	−0.9410	0.3468
positiveSentiment_5	0.0402818	0.0309633	1.301	0.1935
positiveSentiment_6	0.0626920	0.0310438	2.019	0.0436
positiveSentiment_7	0.0248225	0.0310578	0.7992	0.4243
positiveSentiment_8	−0.0221040	0.0310434	−0.7120	0.4765
positiveSentiment_9	0.0128304	0.0310735	0.4129	0.6797
positiveSentiment_10	0.0223202	0.0305792	0.7299	0.4656

Mean dependent var	0.004474	S.D. dependent var	1.000662
Sum squared resid	1447.888	S.E. of regression	0.961549
R^2	0.099644	Adjusted R^2	0.076646
$F(40, 1566)$	4.332794	P-value(F)	2.42e−17
$\hat{\rho}$	−0.000758	Durbin–Watson	2.001384

F-tests of zero restrictions

All lags of return1Day	$F(10, 1566) = 0.497016$	[0.8928]
All lags of negativeSentiment	$F(10, 1566) = 1.38747$	[0.1800]
All lags of articles	$F(10, 1566) = 0.940172$	[0.4948]
All lags of positiveSentiment	$F(10, 1566) = 6.2247$	[0.0000]
All vars, lag 10	$F(4, 1566) = 0.960273$	[0.4283]

For the system as a whole —

Null hypothesis: the longest lag is 9

Alternative hypothesis: the longest lag is 10

Likelihood ratio test: $\chi^2_{16} = 33.735$ [0.0059]