Misinformation Stocks

Final Report
CS 4624 Multimedia, Hypertext, and Information Access
Fall 2021
Virginia Tech, Blacksburg VA 24061

Professor: Edward A. Fox
Client: Dr. Reza Barkhi
December 10, 2021

Prepared by:
Justin Briley
Marc Kwiatkowski
Nicholas Sheng
Jordan Stack

Table of Contents

Table of Figures

Abstract

The growing levels of fake news in our media contributes to misinformation campaigns, the impact of which can spread to investment decisions. To analyze the extent of misinformation on investors, we collected financial articles surrounding specific stocks. We leveraged a machine learning framework to automatically determine the sentiment of a given article. A value is manually assigned to each article in reference to its level of misinformation. This information is displayed in a digital library for users to access. From our small case study analysis, our preliminary findings show that the misleading content of an article ultimately has little impact on stock value. Instead, the sentiment of the public towards the news, regardless of its validity, is the driving force behind price fluctuation.

Introduction

Detection of fake information is increasingly important and can have significant impact on human decisions as individuals are likely to make choices based on the information that they have available to them: information that goes into decision models guides human decisions. To focus on a specific example, we look at how misinformation about public companies can influence investor perceptions and subsequent investment decisions. If investors are manipulated to make decisions by relying on fake information, they will make bad investment decisions. The project involves collecting data about selected public companies and the impact that misinformation has on investor perception and investment decisions, taking data from social media and/or news websites involving investors and the stock market. Data was collected via web crawlers and passed through a model that predicts the sentiment of the article. The level of misinformation present in the article was manually labeled.

Requirements

Data Collection

The first thing that we had to do was to collect data about stocks to determine the validity of the news surrounding them. We searched through many stock sites to find a website that both allowed for scraping as well as provided a succinct and easily accessible supply of stock news for different tickers. Once we found a site that fulfilled both of these, we scraped the relevant articles for title, ticker, and date of publication.

Data Processing

Once the data was collected and formatted, we needed to figure out whether the article was misleading and what impact that could cause. To achieve this, we first went through and labeled all of the articles based on their level of misinformation. Then, we passed them through an NLP routine to determine the sentiment of each article to determine what its impact could be on the stock market. Finally, we analyzed the stock market trends for the specific tickers to see if the news about the stock had caused any change in the cost.

Data Visualization

We needed to find a way to visualize the data so that it was easily viewable by users. We created a website that would contain all of the data that we collected from the previous steps. It allows users to download the data and use it for their needs. Finally, it has all the information on how we collected the data and the source code for scraping the data and information about the machine learning model used to analyze the data.

Design

System Design

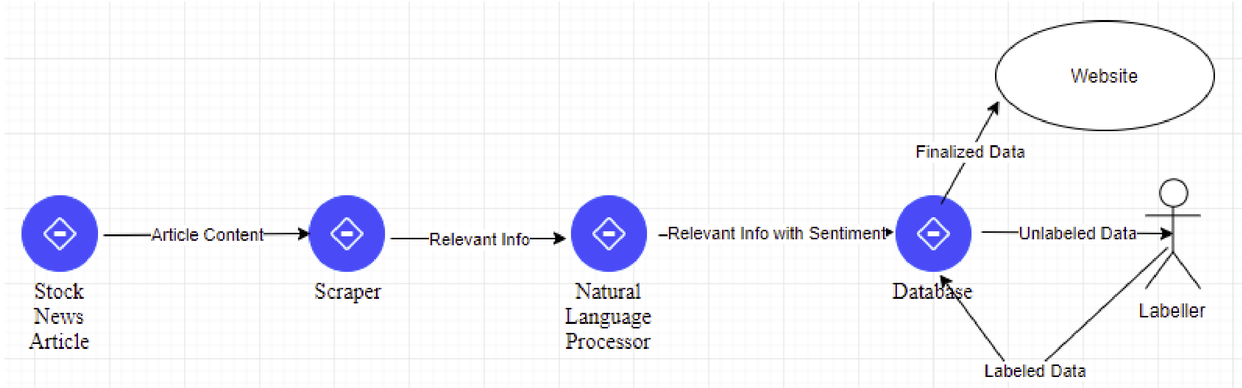

Figure 1. System design diagram

Figure 1 is a depiction of our software system design, created using IBM IT Architect Assistant [1]. Initially, we scraped data from stock news articles. For this, we used MarketWatch [2] as it is a reputable source that allows scraping in their Terms of Service. From our web scraper, we fed that data directly into our natural language processor to predict sentiment. Rather than labeling every entry ourselves, we developed our model to do it for us, saving tons of time. That data was publicly available for download, but we still needed to label misinformation ourselves. Since past stock news doesn't reflect the present, we decided the only way to have accurate data was to label it ourselves. At this point, we completed our backend data flow which started with data gathering and ended with data cleansing and labeling.

Website Design

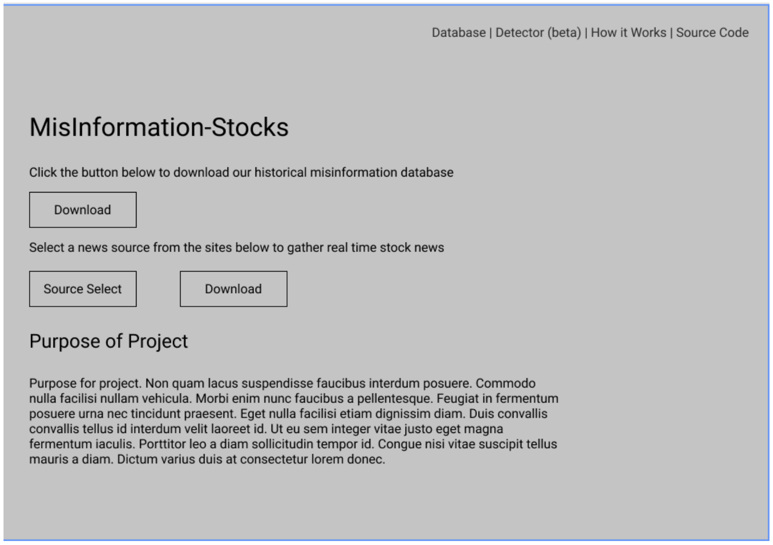

Figure 2. Wireframe of home page

Figure 3. Wireframe of detector

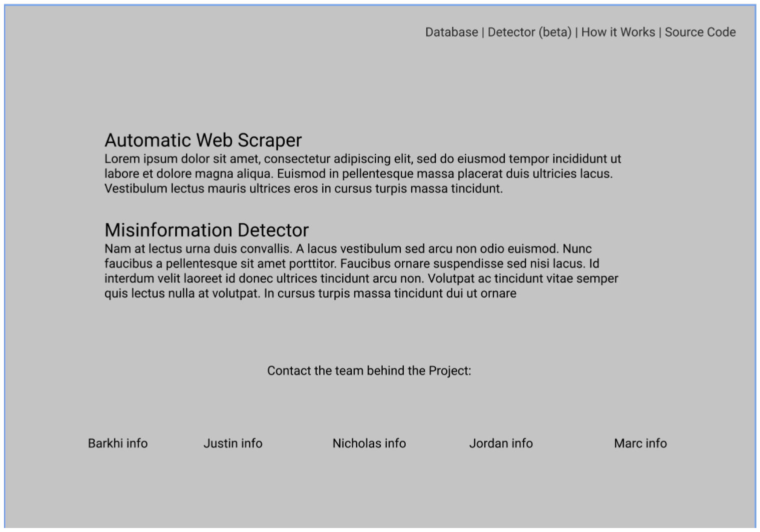

Figure 4. Wireframe of info page

Figures 2 - 4 show our initial wireframe for the website, produced using the wireframing tool Figma [3]. Our plan was to allow users to download data, analyze sentiment in real time, and access our code for personal use. Real time sentiment analysis was seen as a stretch goal, but a fascinating topic to look into. Being able to download our dataset and access our code conveniently was the main goal of the site. As per our client's recommendation, the downloadable datasets are in CSV format to allow for easy parsing.

Implementation

Web Scraper

In deciding which website to scrape, we considered numerous stock market news websites. Of them, we decided to scrape from MarketWatch due to the accessibility of the news articles on the site. We can take a base URL and append any stock ticker that we want in order to look at that stock's news. In Figure 5, you can see what the AMC's news looks like on MarketWatch. The red arrows point to a section on the website that contains a list of recent news articles. This section is what we are scraping, and we are doing this over a list of stock tickers that we have hand selected.

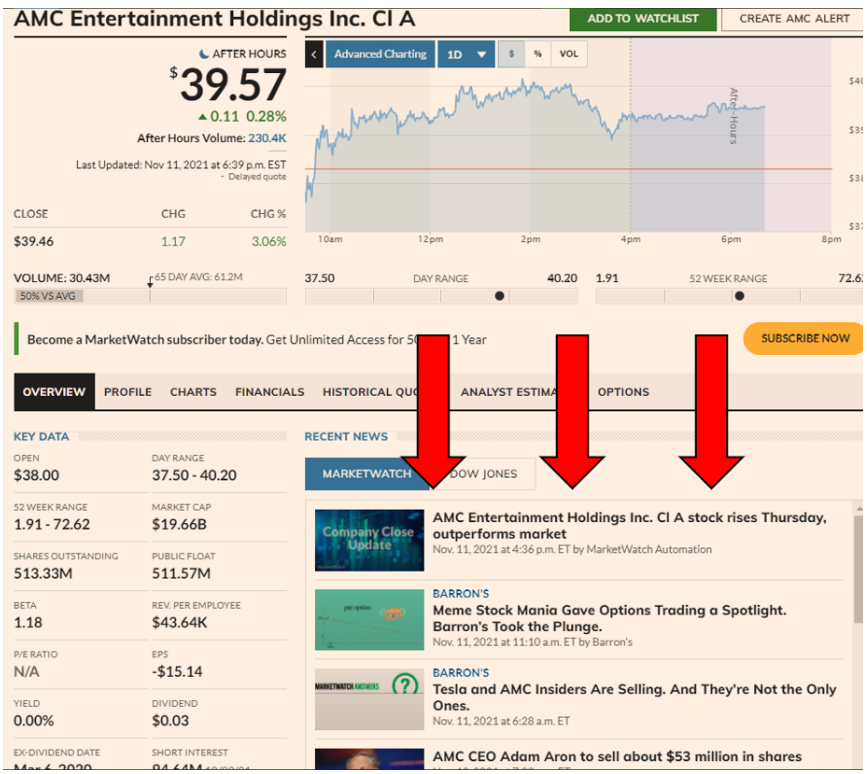

Figure 5. AMC articles in MarketWatch

In order to scrape these webpages, we created a simple Python script that iterates through a list of stock tickers that we are interested in. You can see these stocks on lines 9 - 10 of the code in Figure 6. The code then iterates through the list of tickers, appending each to the base URL as seen in line 13. We next send a request to the altered URL, and pass the response into a BeautifulSoup constructor, which is a Python library that helps to scrape websites [4]. In subsequent lines of code, we select particular parts of HTML from BeautifulSoup that we want to store into our DataFrame. We are using Pandas' DataFrame [5] to store all of this information because there is an easy way to save a DataFrame as a CSV file once we have collected all of the data (line 35 of the code). See Figure 7 for an example of what this looks like.

```
1   import requests
2   from bs4 import BeautifulSoup
3   import pandas as pd
4
5
6   df = pd.DataFrame(columns=["Ticker", "Title", "Time", "Time2"])
7   df['Ticker'] = df.index
8
9   tickerList = ["amc", "amd", "aapl", "tsla", "gme", "bb", "goog", "nvda", "fb", "msft", "jpm", "wmt",
10  "bac", "hd", "dis", "nflx", "crm", "ko", "snap", "sbux", "coin", "baba", "mcd", "t", "vz", "shop"]
11
12  for ticker in tickerList:
13      url = "https://www.marketwatch.com/investing/stock/" + ticker
14
15      headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.61 Safari/537.36'}
16      response = requests.get(url, headers=headers)
17
18      soup = BeautifulSoup(response.text, 'html5lib')
19
20      b = soup.select('div[class="element element--article"]')
21
22      for article in b:
23          title = article.select('a[class="link"]')[0].text.strip()
24
25          if(len(article.select('span[class="article__timestamp"]'))):
26              time = article.select('span[class="article__timestamp"]')[0].text
27              time2 = article.select('span[class="article__timestamp"]')[0].get('data-est')
28
29          ### can edit information before storing it into dataframe here ###
30
31          ##################################################################
32
33          if(title != ""): df.loc[len(df)] = [ticker, title, time, time2] # append to dataframe
34
35  df.to_csv('results.csv', index=False, encoding="utf-8-sig")
```

Figure 6. Web scraper code

aapl	Here's the math for Tesla's stock price if it becomes the Apple of car makers	Oct. 29, 2021 at 9:19 a.m. ET	2021-10-29T09:19:00
aapl	Facebook becomes Meta in rebranding seen as 'an attempt at distraction'	Oct. 28, 2021 at 2:40 p.m. ET	2021-10-28T14:40:00
aapl	Weekend reads: Facebook goes Meta — what's in a name?	Oct. 29, 2021 at 12:04 p.m. ET	2021-10-29T12:04:00
aapl	Apple Stock Is Tumbling After a Rare Sales Miss. Why Its Problems Aren't Over.	Oct. 28, 2021 at 5:04 p.m. ET	2021-10-28T17:04:00
tsla	Dow, S&P 500, Nasdaq end at all-time highs, extending win streak as traders wait for	Nov. 2, 2021 at 5:29 a.m. ET	2021-11-02T05:29:00
tsla	Tesla Inc. stock underperforms Tuesday when compared to competitors	Nov. 2, 2021 at 4:32 p.m. ET	2021-11-02T16:32:00
tsla	These are the four solar stocks to own for the long run	Nov. 2, 2021 at 9:36 a.m. ET	2021-11-02T09:36:00
tsla	Stocks don't get more Meme than Avis	Nov. 2, 2021 at 3:04 p.m. ET	2021-11-02T15:04:00
tsla	Hertz Stock Is Winning as Avis's Gain Outweighs Tweet From Musk	Nov. 2, 2021 at 12:39 p.m. ET	2021-11-02T12:39:00
tsla	Tesla software recall may head off dispute with U.S. safety regulators	Nov. 2, 2021 at 12:01 p.m. ET	2021-11-02T12:01:00
tsla	Tesla Stock Is Sliding. Just Don't Blame Hertz or Recalls.	Nov. 2, 2021 at 6:46 a.m. ET	2021-11-02T06:46:00

Figure 7. Example of scraped data

Data Processing

Once the data was collected from the scraper, we began to process it. This consisted of two main steps:

1. Misinformation detection
2. Sentiment analysis

We found that it was not feasible to use machine learning for misinformation detection. It was simply not reliable enough and the technology isn't quite there for us to automatically detect the level of misinformation contained in a financial news article. Therefore, we approached this part manually. For each article in our dataset, we ranked it on a scale of 1 - 5 based on the amount of misinformation we deemed present in the article, with 1 being the least accurate and 5 indicating the article had no misinformation. By manually labeling, this allowed us to get the most accurate results possible.

There were 417 articles that were manually labeled by two people. We avoided discrepancies in our rankings by discussing how we would handle certain criteria when determining our ratings. The vast majority (~88%) of articles were labeled with a 5 or 4, with the former used to label articles with no lies and the latter indicating the presence of very few lies or slight speculation. A rating of 3 represented a mix of truth, lies, and speculation, and was given to about 10% of our data. About 2% of articles were given a rating of 2. These articles were mostly filled with speculation and misinformation, but had hints of truth within them. A rating of 1 was not given as it was reserved for articles that are entirely inaccurate, something rarely seen in reputable sources such as MarketWatch.

Sentiment analysis was a stretch goal of ours as it involves machine learning, specifically natural language processing. To that end, we use the SimpleTransformers library and the BERT classification model [6] developed by Google as a state-of-the-art machine learning framework used for language processing tasks. This was implemented in a Jupyter Notebook [7] hosted on Google.To help train the model, we supplemented our own data with a dataset found on Kaggle [8], some of which is shown below in Figure 8. This dataset consists of nearly five thousand financial news headlines, each labeled positive, neutral, or negative. There are roughly 600 negative headlines, 1400 positive, and the rest are neutral.

neutral	According to Gran , the company has no plans to move all production to Russia , although that is where the company is growing .
neutral	Technopolis plans to develop in stages an area of no less than 100,000 square meters in order to host companies working in computer technologies and telecommunications , the statement said .
negative	The international electronic industry company Elcoteq has laid off tens of employees from its Tallinn facility ; contrary to earlier layoffs the company contracted the ranks of its office workers , the daily Postimees reported .
positive	With the new production plant the company would increase its capacity to meet the expected increase in demand and would improve the use of raw materials and therefore increase the production profitability .
positive	According to the company 's updated strategy for the years 2009-2012 , Basware targets a long-term net sales growth in the range of 20 % -40 % with an operating profit margin of 10 % -20 % of net sales .
positive	FINANCING OF ASPOCOMP 'S GROWTH Aspocomp is aggressively pursuing its growth strategy by increasingly focusing on technologically more demanding HDI printed circuit boards PCBs .
positive	For the last quarter of 2010 , Componenta 's net sales doubled to EUR131m from EUR76m for the same period a year earlier , while it moved to a zero pre-tax profit from a pre-tax loss of EUR7m .
positive	In the third quarter of 2010 , net sales increased by 5.2 % to EUR 205.5 mn , and operating profit by 34.9 % to EUR 23.5 mn .
positive	Operating profit rose to EUR 13.1 mn from EUR 8.7 mn in the corresponding period in 2007 representing 7.7 % of net sales .
positive	Operating profit totalled EUR 21.1 mn , up from EUR 18.6 mn in 2007 , representing 9.7 % of net sales .
positive	TeliaSonera TLSN said the offer is in line with its strategy to increase its ownership in core business holdings and would strengthen Eesti Telekom 's offering to its customers .
positive	STORA ENSO , NORSKE SKOG , M-REAL , UPM-KYMMENE Credit Suisse First Boston (CFSB) raised the fair value for shares in four of the largest Nordic forestry groups .
positive	A purchase agreement for 7,200 tons of gasoline with delivery at the Hamina terminal , Finland , was signed with Neste Oil OYj at the average Platts index for this September plus eight US dollars per month .
positive	Finnish Talentum reports its operating profit increased to EUR 20.5 mn in 2005 from EUR 9.3 mn in 2004 , and net sales totaled EUR 103.3 mn , up from EUR 96.4 mn .
positive	Clothing retail chain Sepp+Æl+Æ 's sales increased by 8 % to EUR 155.2 mn , and operating profit rose to EUR 31.1 mn from EUR 17.1 mn in 2004 .
positive	Consolidated net sales increased 16 % to reach EUR74 .8 m , while operating profit amounted to EUR0 .9 m compared to a loss of EUR0 .7 m in the prior year period .

Figure 8. Example of labeled data

Before we could train the data, the text itself had to be pre-processed. This involved removing numbers, punctuation, and stop words, which are taken from a list of the most commonly used words in the English language. Once the text was uniform and fit the criteria, we passed it into the model. We set aside some of the data for testing the model's accuracy, which is detailed in the respective section below. After training the model, we passed in our dataset and performed sentiment analysis on each article. Since the dataset is in CSV format, it is trivial to read from that and convert it into a Pandas DataFrame, where we insert another column that contains the sentiment label. This data is then converted back into a CSV file. The Jupyter Notebook containing the code and data files can be found in a zip file named MisinformationStocks.zip.

Website

The goal of our website is to allow for easy access of data to users. They can download our data in the form of a CSV file. From our wireframes earlier, we had an easy time designing our frontend. We chose React [9] for our frontend as it gives us useful industry experience. Our deployed frontend closely follows our wireframe. Using React and MaterializeUI [10] for styling, we were easily able to craft the site. As planned, we have downloadable datasets and an explanation of our project.

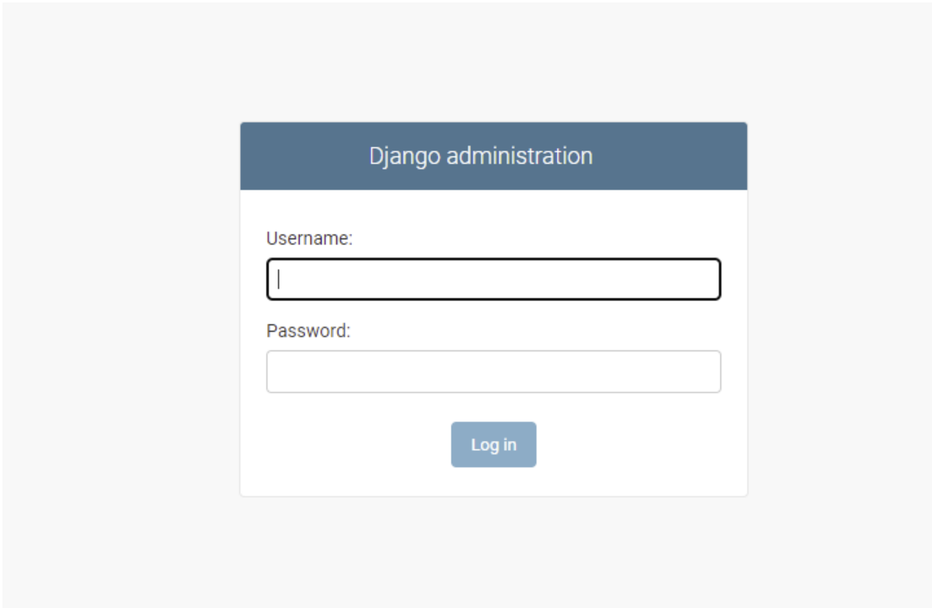

Figure 9. Admin login

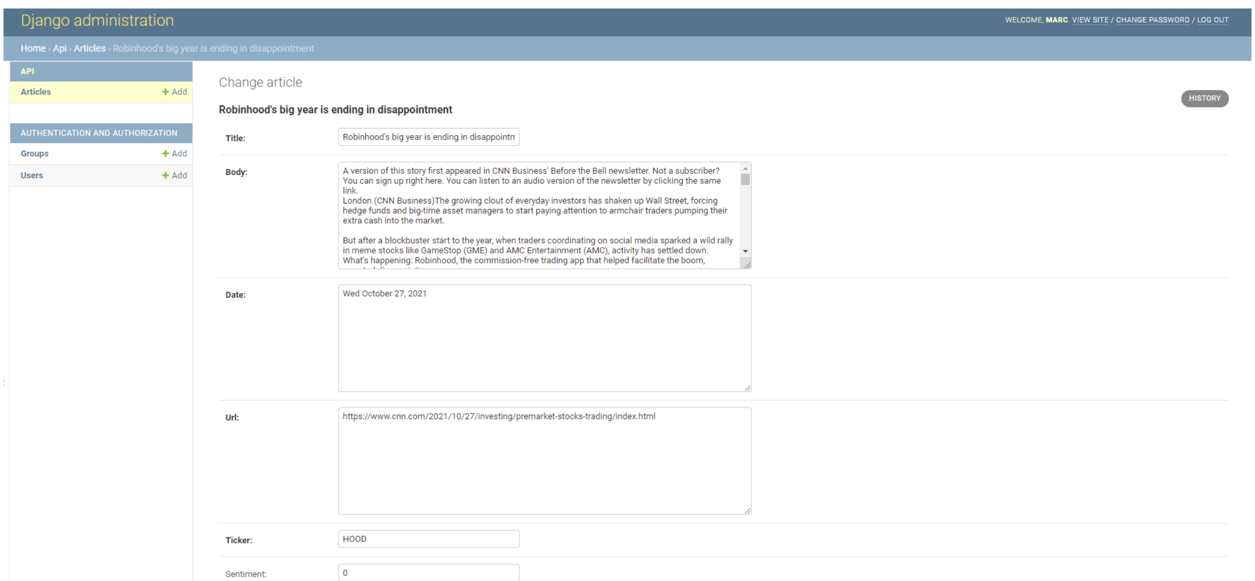

Figure 10. Admin editing interface

Our backend was made using a Django REST API to provide real time functionality potential to the frontend for future iterations of the project. Django [11] is a Python based backend that offers a lot of “out-of-the-box” functionality. The primary example of this is the admin portal shown in Figure 9. Once logged in through the admin portal, you can easily alter any of the article’s information as seen in Figure 10. Since we have to manually label data, this GUI is an invaluable tool to ramp up efficiency.

For hosting services we had to take an alternative approach. Initially we had everything set up through AWS, but decided to go with an alternative because of pricing. Our next attempt was to set up a Docker container [12] to house our website through Virginia Tech hosting services. With VT, their hosting services would be free, but also require an admin for security purposes. Since our team is graduating, and there will be no one to maintain the site we sought a third hosting service. Currently, our frontend is hosted on Heroku [13] and our backend is hosted on PythonAnywhere [14]. These two sites were chosen as they had easy setup and offer free services. Since we don't anticipate a large quantity of network traffic, free tier services will be sufficient. The frontend will no longer be maintained after the semester and will not continue to be hosted.

Testing

Accuracy is an important metric for the evaluation of machine learning models. This is based on the percentage of correct predictions made by the model, comparing each prediction with the corresponding label to determine correctness. One common approach for testing is to partition the labeled dataset into separate datasets for training and testing. We chose to set aside 20% of the data for testing. To determine the accuracy of our model, we passed in the testing set and used the metrics module of the Scikit-learn library in Python [15]. Using this approach, we found that our machine learning model made predictions with an accuracy of 78.5%.

User’s Manual

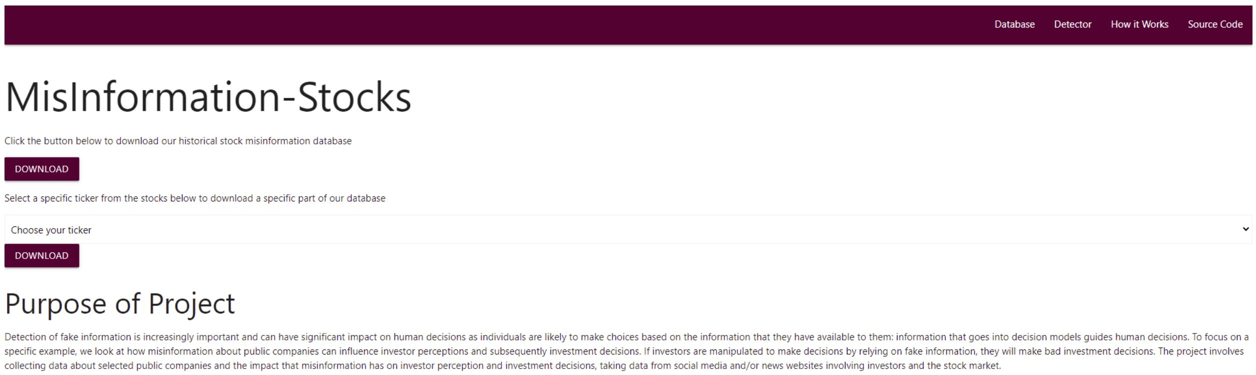

Figure 11. Database page

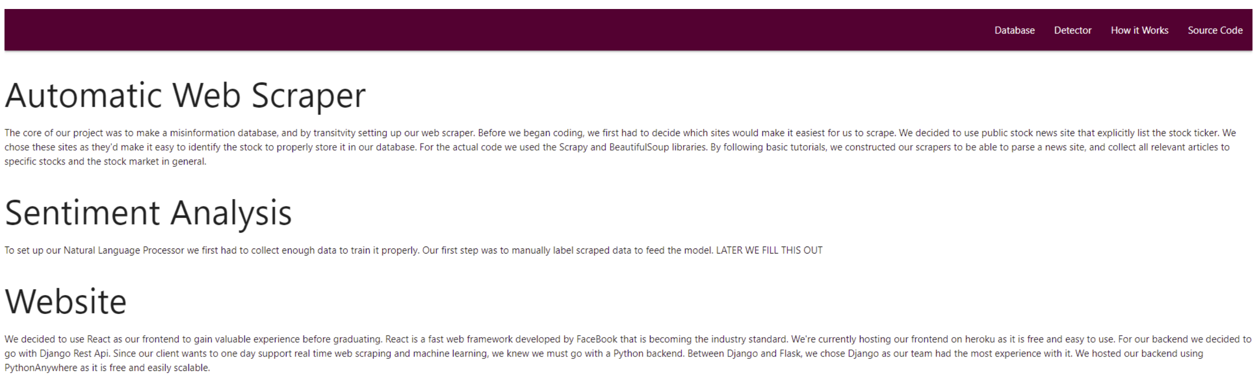

Figure 12. Explanation page

Our site was made to be simple to navigate. The home page is accessed using the database tab and is shown by Figure 11. On this page you can download stock information. Our site also details the purpose of our project. The other page currently accessible is the explanation page displayed in Figure 12. This page goes into further detail about each aspect of our solution.

Developer's Manual

Scraper

There is one Python file that handles the entire scraping process. It uses Python's requests module and Pandas DataFrame. This file can be found in the MisinformationStocks.zip file. The stock tickers that we are interested in are in an array in one of the first lines of code. You can add or remove tickers provided that they are valid. This will let you personalize which news articles are scraped from MarketWatch. The rest of the process is handled for you, and the code generates a CSV file containing the results of the scraping process. If you wish to gather different information from articles, or scrape articles from another source, this scraper will need to be changed appropriately. Currently, it gathers the article titles, dates and times, and links.

Machine Learning

The code for the sentiment analysis machine learning model is in a Jupyter Notebook, which can be identified by the .ipynb file extension. This file is located in MisinformationStocks.zip along with the CSV file all-data.csv, which contains the dataset used to train the model. The code is pretty straightforward to modify. Click on a cell to edit and add your Python code. Be sure to install the necessary packages. Press Ctrl+Enter to run a cell. Any variables or imported modules will persist in all cells below. You can import your own CSV file, convert it to a Pandas DataFrame, and pass it into the model. We encourage you to read through the Jupyter Notebook [7] and Pandas [5] documentation for more information if necessary.

Website

For our website you can find the necessary files in the MisinformationStocks.zip file. The frontend and backend portions of the site are separated into their own directories. Both of them are capable of being deployed for your own services as soon as you fork the directory. The React site is static and made using standard React practices. The CSS library we chose to use is MaterializeUI. We highly recommend referencing React [9] and MaterializeUI [10] documentation if you wish to make your own edits on the site. The backend was made entirely using the Django REST API. Again, we recommend following their documentation [14] for installation.

Lessons Learned

Timeline

Throughout the development process, we had many ideas for what we wanted to do. We weren't given specific direction on where to take it and had to figure out a lot of the details on our own. The broad scope of our project resulted in us continuously changing directions and reworking our plan for the first few weeks. Eventually, we were able to clearly define the scope of our project, removing the ambiguity surrounding the deliverables and how to best achieve our goals. We have learned to properly plan a project and create a timeline after we have a set of realistic goals to work towards. Our final timeline is as follows:

9/16/21: Presentation 1
9/22/21: Complete wireframe and system design
9/30/21: Pick SeekingAlpha as source and start building scraper
10/5/21: Look for alternative sources for scraping due to SeekingAlpha restrictions
10/11/21: Come up with approach to label misinformation manually
10/20/21: Choose MarketWatch as our source to collect articles from
10/23/21: Finish building scraper
10/29/21: Deploy website
11/4/21: Start implementing machine learning model for sentiment analysis
11/9/21: Presentation 2
11/12/21: Finish training model and test for accuracy
11/19/21: Refine code and streamline website design
11/29/21: Begin labeling data
12/2/21: Final presentation
12/9/21: Complete code refinement and final paper, finish labeling data

Problems

During the scraping process, there were problems getting data that we desired from multiple websites. Most of the top websites for market news have policies and scripts in place so that the information is harder or impossible to scrape. For example, Seeking Alpha [16] is a popular market news website, and its layout would have made it extremely easy to scrape thousands of recent articles for our database. However, their Terms of Service disallows the scraping of their website, and there are measures set up so that scraping libraries aren't able to scrape data from their website. Another example is Yahoo Finance [17], whose website is riddled with pop-ups and ads, which makes the scraping process difficult to automate. After considering the aforementioned sites, we then looked at MarketWatch and found that it was the site that best suited our needs and allowed us to scrape.

Data Inferences

Based on the data that we collected from this project, we have come to multiple conclusions. Note that due to time constraints, we were unable to do an in-depth study on this topic. Therefore, these observations only reflect the results of this preliminary exploration we conducted using a case study approach. From our manually labeled data, we looked at articles of different ratings and the corresponding stock prices. We found that in general, misinformation in the news had little impact on a stock, as the sentiment surrounding the article was more indicative of stock price fluctuation. For instance, if misinformation comes out for a stock but the public is unaware of its deceptive nature, then there is no difference in impact, even if the information were true. If the public later realizes that the information is false, the stock will usually level back to where it was before. This is illustrated perfectly when misinformation was spread about LiteCoin partnering with Walmart [18], their stock jumped up about 50%. When the public realized that this was simply untrue, their stock went back down to where it was previously.

We found that there are some stocks that are more likely to have misinformation surrounding them, and are therefore more likely to be affected by it. A lot of these are known as “meme stocks”, which are prone to volatility and are far riskier than most stocks. By using MarketWatch, a mainstream source, most of our articles were reputable since these popular websites have a reputation to uphold. They are more likely to avoid misinformation than sources like Twitter, where individuals are allowed to say whatever they want with little repercussion. On the other hand, mainstream sources are more likely to be used by the majority of users, which increases the potential impact of each article. This is the main factor in our decision to focus on MarketWatch in lieu of social media sites.

Future Work

To move forward with this project, we recommend using significantly more resources. There are many complex aspects to advancing this project, and we believe each one could use a team to perfect them. The three aspects we've identified for improvement are:

1. Real-time data capture
2. Misinformation predictor
3. Website

Currently, the scraper only supports MarketWatch, and this can be improved upon to include more sources along with adding the ability to scrape in real-time. Another useful functionality to add would be to distinguish between repeat articles. This improved implementation would require an efficient way to store data and pass it through to the various parts of the project.

One could also expand upon the machine learning aspect of the project by attempting to train a model to predict misinformation. This might prove to be difficult to accomplish with any reasonable amount of accuracy, but is certainly an interesting concept to explore. It is a natural progression from our current approach of manually labeling misinformation, detailed in the Implementation section of this paper for those wishing to manually label data themselves. When paired with the previous suggestion, this would allow for the compiling of a much larger dataset. This is an advantage due to the potential large amount of data needed to train this hypothetical model.

There are a few quality-of-life features that we believe could be added to the website. The website currently does not support parameterized queries for our data. The implementation of this functionality would allow for users to filter through to find information relevant to their needs. Another feature that would be useful for users is the addition of stock charts to visualize the impact of the article on stock price, if any. The website could also be improved by adding an option for users to submit their own data for review. This would let users add their own contribution and improve the dataset.

Acknowledgements

We would like to thank our client Dr. Reza Barkhi for having us work with him to develop this project. We'd also like to thank our professor Dr. Edward Fox for assisting us throughout our project. Their information can be found below:

Dr. Reza Barkhi
Virginia Tech Professor and Account & Information Systems Department Head
reza@vt.edu

Dr. Edward Fox
Virginia Tech Computer Science Professor
fox@vt.edu

References

[1] IBM, *IBM Architect Assistant*, 2018. [Online]. Available: https://demo.architect-assistant.com/cogarch/pagelogin. [Accessed: 04-Oct-2021].

[2] MarketWatch, *MarketWatch*, 2021. [Online]. Available: https://www.marketwatch.com/. [Accessed: 02-Nov-2021].

[3] Figma, “The Collaborative Interface Design Tool,” Figma, 2021. [Online]. Available: https://www.figma.com/. [Accessed: 14-Sep-2021].

[4] BeautifulSoup, “Beautiful Soup documentation,” *Beautiful Soup Documentation - Beautiful Soup 4.9.0 documentation*, 2020. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Accessed: 28-Sep-2021].

[5] Pandas, “Pandas.dataframe,” *pandas.DataFrame - pandas 1.3.4 documentation*, 2021. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html. [Accessed: 28-Sep-2021].

[6] T. Rajapakse, “Classification models,” *Simple Transformers*, 29-Dec-2020. [Online]. Available: https://simpletransformers.ai/docs/classification-models/. [Accessed: 12-Nov-2021].

[7] Project Jupyter, “The Jupyter Notebook,” *The Jupyter Notebook - Jupyter Notebook 6.4.6 documentation*, 2015. [Online]. Available: https://jupyter-notebook.readthedocs.io/en/stable/. [Accessed: 27-Oct-2021].

[8] A. Sinha, “Sentiment Analysis for Financial News,” *Kaggle*, 27-May-2020. [Online]. Available: https://www.kaggle.com/ankurzing/sentiment-analysis-for-financial-news. [Accessed: 27-Oct-2021].

[9] Facebook, *React*, 2021. [Online]. Available: https://reactjs.org/docs/getting-started.html. [Accessed: 06-Oct-2021].

[10] Google, *Materialize,* 2021. [Online]. Available: https://materializecss.com/. [Accessed: 06-Oct-2021].

[11] Django Software Foundation, “Django Documentation,” *Django,* 2021. [Online]. Available: https://docs.djangoproject.com/en/4.0/. [Accessed: 06-Oct-2021].

[12] Docker, “Docker documentation” *Docker Documentation*, 2021. [Online]. Available: https://docs.docker.com/. [Accessed: 20-Oct-2021].

[13] Salesforce, “Heroku,” *Heroku Dev Center,* 2021. [Online]. Available: https://devcenter.heroku.com/. [Accessed: 08-Nov-2021].

[14] PythonAnywhere LLP, “PythonAnywhere”, *PythonAnywhere,* 2021. [Online]. Available: https://www.pythonanywhere.com/. [Accessed: 08-Nov-2021].

[15] Scikit-learn, “3.3. Metrics and scoring: Quantifying the quality of predictions,” *scikit,* 2021. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html. [Accessed: 12-Nov-2021].

[16] Seeking Alpha, “Seeking Alpha: Stock market analysis & tools for investors,” *SeekingAlpha*, 2021. [Online]. Available: https://seekingalpha.com/. [Accessed: 09-Sep-2021].

[17] Yahoo, “Yahoo Finance - Stock Market Live, quotes, Business & Finance News,” *Yahoo! Finance*, 2021. [Online]. Available: https://finance.yahoo.com/. [Accessed: 20-Oct-2021].

[18] M. Binder, “No, Walmart will not be accepting Litecoin. Here's how people were duped.,” *Mashable*, 29-Oct-2021. [Online]. Available: https://mashable.com/article/litecoin-walmart-fake-news. [Accessed: 14-Sep-2021].