Short-term Stock Price Prediction Using Sentiment Analysis and Machine Learning

Portfolio

COMP208 Group 4:
Killian McShane, James Harris, Will Payne, Matthew Blatherwick, James Murphy

1. Report

a. Summary of Team Member Roles

Killian McShane's Role

I formed the project concept and took a relaxed lead. I organised meetings and templates for the group members to select which area of the project they felt most comfortable with.
I then created synthetic sentiment data from normalised stock prices to train the machine learning models on.
Once the models were trained, I developed a python script to generate the sentiment data from Twitter users' comments regarding certain stocks for the test phases of the machine learning models.
I created the Google Sheets database, shared API credentials with the group and wrote Python scripts so that each member could interact with it.

Matthew Blatherwick's Role

My role in this group project was to build and design the website. I did this by using HTML accompanied by another member's JavaScript for graph functionality. I also used bootstrap 5 within my HTML so that the creating process was much less time consuming, and to easily import things such as tables, and certain design elements. Once the design element was finished and all the necessary pages were set up, I worked closely with another student in the group to implement their JavaScript code into the webpage, to ensure that the graphs display correct information and take you to the correct subpage when you click each individual stock. This required good communication between me and the other members of the group.

James Harris' Role

My role in the team has been focused on the machine learning algorithm. It has involved researching and implementing the LSTM machine learning algorithm in Python3. I created a function which trains and saves models for each stock, and I have also created a prediction function which outputs the prediction for the next day and stores this data in Google Sheets so that it can be retrieved by the website.

Will Payne's Role

My role in this project's production was to create the Google Sheets API to store the sentient data and allow the JavaScript program to retrieve this data. However, I came into some issues with Google's OAuth consent, and so Killian came in to help and circumvented the problem by using the Google Sheets page as a webhook.

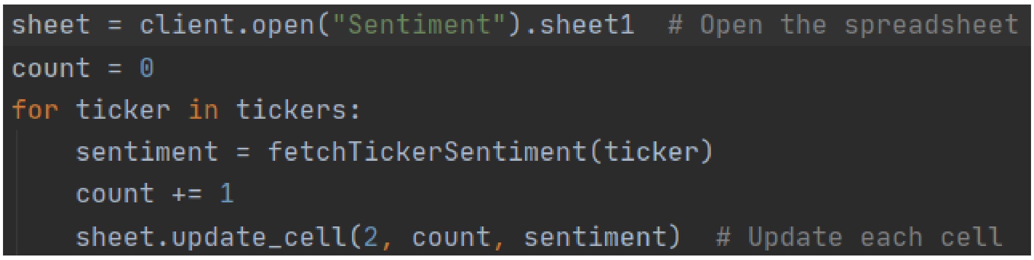
James Murphy's Role

In this project my role was to use JavaScript to add functionality and features to the website. This included creating the graphs of stock prices that are displayed on our website. This role meant that I worked with the Google Sheets that we were using as a database, as I needed to extract the data that had been stored so that it could be presented to the user.

b. Overview of Application

This project involved the production of a web application that enables users to view and interact with short-term stock price predictions by web-scraping Twitter using their developer API and Python's 'textblob' library to collect sentiment scores (a decimal score from 0 to 1, indicating consumer opinion from tweets) for given stock ticker symbols.

These sentiment scores are then uploaded to a Google Sheets database via the Google Sheets API ('gspread') to be accessed by the machine learning models.



```
sheet = client.open("Sentiment").sheet1  # Open the spreadsheet
count = 0
for ticker in tickers:
    sentiment = fetchTickerSentiment(ticker)
    count += 1
    sheet.update_cell(2, count, sentiment)  # Update each cell
```

Predictions are then be made by feeding historical stock data from Yahoo Finance into a recurrent neural network (an LSTM) combined with the daily synthesised sentiment data. Finally, a prediction is then made using both consumer opinion (from sentiment scores that day) and our recurrent neural network’s ability to predict. The system predicts the following day’s highest price.

These predictions are then also uploaded to our Google Sheets database along with the last month’s price data. This data is then pulled and used to create the price graphs on the web application via JavaScript.

Our system currently only has two types of user, the admin (or admins) and those viewing the web application. It is the admin’s choice to change which ticker is being displayed and any other relevant information, viewers are unable to change any of the current settings.

c. Achievements

Throughout the entire production of our system our group achieved many things such as being able to have the team work to create a system containing many different features that can work in parallel with each other. Most of our team had little or no experience with creating their part of the system. We managed to learn quickly how to perform our respective roles. We managed to use machine learning to create our own predictions of stock market values and were able to present that to a user on a website that is accessible by anyone.

Graphs can be displayed to show the user past stock prices for individual companies to see how the prices have changed over time, giving more information to the user to help make their decisions. Furthermore, our website is simple and easy to navigate, giving necessary information to the user without our website seeming too complicated which could overwhelm the user. Our machine learning algorithm is the highlight of our system and will only get better over time, the more values we process the better our results will be.

d. Evaluation: Strengths & Weaknesses

Unfortunately, the team was unable to obtain a Twitter developer account in time (although we did apply at the start of the semester) and so we have been unable to gather actual sentiment data for training. Going forward we would fix this issue and incorporate it into our system given more time.

The team took the decision to only include 5 stocks to begin with as we pilot the tool and to concentrate our efforts on those stocks, this can probably be considered a weakness of our web app, but this can easily be modified going forward. Looking at our system now, it would have been nice to also add on some nice features and visual indicators around the stocks and associated charts on the website. These could give more information to the user and help them understand the predictions more easily and understand how recent forecasts have fared.

Despite these shortcomings the project has finished with the team creating a well-rounded finished product. There are several nice features that we have created, and the team can be proud of. The machine learning algorithm which is the core element of the system has performed well on test data and runs smoothly.
The website itself has an aesthetic, sleek design and incorporates nice visuals to show the previous performance of stocks. Another important and strong feature that is not seen by the end user is the Google Sheets API integration which ties the back-end code to the front end seamlessly. This is a crucial part of the system and allows data to be stored and transferred between the Python code and the web app.
Another strength of the product is that it has been setup, and now left, in a state which makes it easy to train a model on new stocks and add new stocks to the site. Guidance has been provided in the user manual on how this can be quickly and easily completed.

The team functioned well together and supported each other throughout the project. Initially we started out as a team of six, but one member of the team has been absent throughout. This has not deterred the team and we have continued to meet regularly throughout the semester to tackle each deadline and as a collective group, everyone has brought their own ideas and unique skills to the project.

e. Future Developments

Time constraints meant that we were not able to collect 5 years' worth of sentiment data from Twitter. Our application to their Developer API platform was approved at the very last minute, so we were only able to implement real sentiment data for the models' predictions. In the future, the application would benefit from being re-trained on real sentiment data. This would hypothetically allow us to create even more accurate stock price predictions.

Future versions of the application could also be improved by hosting the machine learning models on a cloud service such as Google's Cloud Platform which would allow for much better stability in terms of constant daily price updates. The current system relies on a computer to be constantly running in the admin's home. This comes with many vulnerabilities including power cuts, fires, internet outages, etc.

Another update to the system could involve monetising the platform. This kind of sentiment-based stock price data is invaluable to traders. Offering a monthly subscription for more data on the stocks and their current sentiment could be a great source of income for the developers.

We would also ideally like the website to have the ability to store logins. This would give the website the functionality to be able to (once we increase the number of stocks we predict) offer its user the ability to only show stocks they are interested in. For example, the user could login to the website and favorite certain stocks, so when they navigate over to the stocks page, the only stocks that will show will be the ones they have favorited. This would also come hand in hand with a built-in search engine. Meaning the user could also search for certain stocks as navigating through page after page of all the stocks we have predictions on would be tedious. [Text Wrapping Break] [Text Wrapping Break] With this login function, we would also be able to offer different user views, so that we could have an administrator view, dedicated to things such as security, layout, and other general tools to ensure quality of life on the website. This means the admin would obviously have a different view to the website and would be able to access pages and tools that a normal user (customer) would not be able to.

We could also implement a sort of questionnaire to new users, so that some pages display more complex information. For example, a question such as "Are you familiar/experienced with stocks and shares?" could be asked when creating a new profile. If the user answers with yes, then the website could display more information and tools to the user. However, if they answer with no, the user could instead be displayed with a tutorial showing them how to use the site, or perhaps just a dumbed down version of certain pages, to not overwhelm a new user.

f. Codes of Practice (BCS) (one whole page)

The BCS Code of Conduct comprises four key principles:

1. Public Interest

   This relates to conducting ourselves professionally and fairly and, given that this project simulates completing a project for a client, we believe we have behaved with professionalism throughout the project, and so we have satisfied this principle.

2. Professional Competence and Integrity

   This principle requires us to never take on a task which we do not have the skills and resources to complete. We have adhered to this well as we have produced every feature which we set out to complete, and if anything, we could have been a bit more ambitious. Another section requires that we ensure we have knowledge and understanding of relevant legislation. This does not apply to us necessarily as we are not releasing our program to the public, however given that our software is attempting to predict something as irregular as the stock market, we have included a disclaimer to waive responsibility for anyone who attempts to follow our analysis with their own funds.

3. Duty to Relevant Authority

   This ensures that we are always acting in our clients' best interests and to take responsibility for our actions. We are collectively very proud of the program we have produced, so we have no issues taking responsibility for our actions and believe that in this hypothetical scenario where this has been produced for a client, we are confident that they would also be impressed given what we promised.

4. Duty to the Profession

   The final principle is to support colleagues in the industry to promote it positively worldwide. This one relates to us the least as we are not working in the IT space yet or in an office environment, so most of the points are not something we can realistically achieve. However, the last point on this principle says we should encourage and support fellow members in their professional development, and we have supported each other in making our program and encouraged each other throughout the project.

g. Bibliography

COMP226: Trading and Financial Markets with *Prof. Rahul Savani*, *RS*

Twitter Data Analysis & Social Listening.
Twitter Inc
https://developer.twitter.com/en/use-cases/listen-and-analyze.
Accessed 13 May 2021.

'Python Quickstart | Sheets API'. *Google Developers*
Google LLC
https://developers.google.com/sheets/api/quickstart/python.
Accessed 13 May 2021.

Mark Otto, Jacob Thornton, and Bootstrap Introduction.

https://getbootstrap.com/docs/5.0/getting-started/introduction/.

Accessed from 01 May 2021.

'Stock Closing Price Prediction Using Machine Learning Techniques'. *Procedia Computer Science*, vol. 167, Jan. 2020, pp. 599–606. *www.sciencedirect.com*, doi:10.1016/j.procs.2020.03.326.

Ng, Yibin. 'Machine Learning Techniques Applied to Stock Price Prediction'. *Medium*, 3 Oct. 2019, https://towardsdatascience.com/machine-learning-techniques-applied-to-stock-price-prediction-6c1994da8001.

BCS Code of Conduct | BCS. https://www.bcs.org/membership/become-a-

member/bcs-code-of-conduct/. Accessed 13 May 2021.

Chart.Js | Open Source HTML5 Charts for Your Website.

https://www.chartjs.org/. Accessed 14 Apr. 2021.

2. Design Documentation

a. Background, Aims & Objectives

The proposed stock price prediction system has four key components, each with their own unique challenges. This report aims to quell any confusion regarding the structure of the system and provide a framework for the project's implementation.

The following points outline our major system objectives.

i. **Sentiment Analysis**

- To create web-scraping algorithm using Twitter's Developer API allowing us to collect sentiment scores of Tweets relating to 5 pre-selected stocks.

ii. **Machine Learning**

- To create a machine learning model, trained on historical Yahoo Finance data and sentiment scores to output stock price predictions.

iii. **Database**

- To create a database on Google Sheets, using Google's 'Sheets API' to upload daily stock price predictions to, and allow the web application to pull data from.

iv. **Web Application**

- To create a web application displaying each of our pre-selected stocks' current price, along with our prediction for its closing price and a simple graph.

The following points elaborate on some initial system requirements.

Archive Old Predictions: This is to allow for historic graphing of the price predictions. The prediction line-graph should (hopefully) closely match the actual stock prices. Both the prediction, and real price lines should be graphically displayed on the same chart for an easy, user-friendly accuracy comparison and evaluation.

Responsive Web Design: Users should be able to view the predictions and corresponding graphs on any device they own without display errors. We want our design to respond to the user's behaviour based on screen size, platform, and orientation. To achieve this, we will opt for a mobile first approach. We will handle the resizing of the website while designing in HTML and CSS to make sure that it looks appealing on all devices.

Graphing: We want the graphs to display data using contrasting colours so that users can easily distinguish between actual prices and predicted prices.

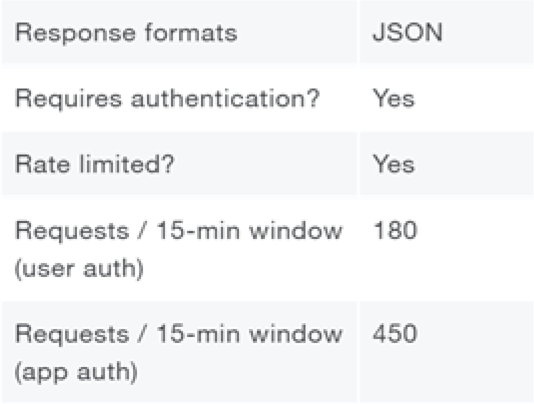
Disclaimer: A disclaimer on the web application's front page should notify users that we are not providing financial advice and that any losses made on their part are their own. We are not liable for any damages/costs incurred.

Desirable Features:

- A login system for users which could potentially be altered to enable a payment plan in the future depending on the success of the project.

b. Changes to Original Proposal

The system is ambitious. It therefore goes without saying that some changes had to be made for the success of the overall project. It was decided that users should not be allowed to select their own stocks to form a price prediction. As the system's userbase scales into the hundreds, the resources required to produce the prediction would be far too demanding for a project of this scale. There is a hard-coded limit on the number of requests one can make to the Twitter API every 15 minutes.



Response formats	JSON
Requires authentication?	Yes
Rate limited?	Yes
Requests / 15-min window (user auth)	180
Requests / 15-min window (app auth)	450

As you can see from the table, our application can produce 1800 tweet-based sentiment scores every hour (450 lots of 4). Initially, to train the machine learning model, each of the 5 stocks will require 1825 sentiment scores (relating to 5 years' worth of daily stock sentiment).

Once the model is trained however, we will only need 1 sentiment score for each stock, each day.

While this seems reasonable, when scaled up to hundreds of users, each requesting new stocks, our system would simply break down. It was therefore concluded that we would clearly need to limit the number of stocks.

c. Summary of Research & Analysis

Research was made into Natural Language Processing and the APIs available to us. We found that the process used a lot of fundamental machine learning knowledge that we already had. However, steps during the pre-processing stage of data were novel to us. Tokenization is a pre-processing step which splits longer strings of text into smaller word pieces, or tokens. It can also be referred to as text segmentation or lexical analysis. These tokens are then converted to numerical data through a process call binarization, so that they can be fed into the machine learning model for training.
We needed to research how data could be obtained from Yahoo Finance and how that data is structured. This was fairly a simple exercise and achieved by exploring the API and through trial and error. We needed to understand the different methods/algorithms that can be used for stock price prediction and how to evaluate them. This information has come from a mix of blog posts, research papers and taught modules at the University, noted at the end of the document.

2. Design

a. **Preface**

Our system is in some ways an empirical investigation into the effect of sentiment data on the predictive ability of a machine learning algorithm. Therefore, *we hypothesise that by training our model on this additional information, its predictive ability should improve.*

b. **Anticipated Components**

- **Sentiment Collection**

Sentiment collection forms the foundation for the system. These scores will be the key attribute for our investigation of their improvements in the predictive analysis of stock prices.

Using Twitter's Developer API, we will be able to collect 5 years-worth of daily sentiment scores for each of the 5 stocks every hour or so. This is due to the hard-coded limitations on requests to their website every 15 minutes.

- **Data Pre-processing**

Historical stock price data from Yahoo Finance will be combined with sentiment scores taken from Twitter in a Panda's data-frame structure. The data will need to be pre-processed for the machine learning model to efficiently learn and predict. Pre-processing is done by a series of the Tokenization's, Binarizations that were discussed in the section above and finally appended to the daily adjusted stock closing prices.
Once this final data-frame is formed for each stock, we should then be able to feed it into the machine learning model.

- **Machine Learning**

Predicting stock price predictions depends on various factors such as the global economy, company financial reports and performance as well as the current political landscape. Typically, there are two approaches that can be used for predicting stock prices for organisations, technical analysis, and fundamental analysis.
We will be focusing on technical analysis which is a method that uses historical prices of stocks to predict future prices. A machine learning algorithm uses large quantities of data to find patterns. It learns with experience and can be fine-tuned to suit the given data. There are multiple algorithms which are

applicable to stock price prediction and these will be explored further in this document.

- **Graphing**

In comparison, graphing is a relatively trivial component of the system however, these images will form the most appealing component of the web application.
Several graphing libraries exist. Seaborn is our current choice for its improved visualisations over MatPlotLib.

- **Google Sheets Upload & Download Function**

We decided to use Google Sheets as it provides a free easy to use API which allows us to upload and download with ease in both Python and JavaScript.
Data uploading will be done using Python, as this is the language the data will be produced in.
Data downloads will be done using JavaScript, as this is the language that the web application will be coded in.

- **Web design**

To design the website, we will need to use a few different languages. HTML will be used to provide the content displayed to the user, while CSS will give structure and style to that content. We will be using JavaScript to give the website interactive functionality and allow the user to interact with the interface. To design the charts, we will be using a free open-source library called chart.js. This will give us a good template to create a clean and informative graph.

- **User accounts**

We want to give users the ability to have their own account when using our website. This will allow each user to choose what they want to see when on the website. A favourites feature will be implemented so that users can keep track of the stocks they are interested in the most.

- **Creating stock graphs**

To create graphs displayed on the website we will be using a JavaScript library called Chart.js. Values for the graph will be taken from google sheets. These graphs will give the most information to the user at just a glance and will give the website more visual appeal. We do not want the website to be cluttered with text, so a more clean and simple presentation of data will be more effective and less intimidating.

- **Ranking system**

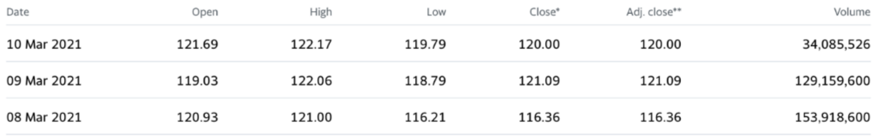
Our website will have a ranking system to display the top stocks at the current time. The user will also be able to change the time frame over which the rankings are decided, such as seeing the best stocks daily, weekly, or monthly. This should give the user a quick look to see which stocks are doing well.

c. **Data Descriptions**

Initially the application will show five stock predictions with the view to expanding on this figure. For each stock prediction we will require a dataset of historical prices from Yahoo finance.
The structure of this data is shown in *Figure 1*.

Each column indicates the following:

- *Date*: The date of the stock prices.
- *Open*: The opening price of the stock on the given date.
- *High:* The maximum price of the stock on the given date.
- *Low:* The minimum price of the stock on the given date.
- *Close:* The close price of the stock on the given date.
- *Adj. close:* The close price of the stock adjusted to reflect corporate actions.
- *Volume:* The total amount traded on the given data



Date	Open	High	Low	Close*	Adj. close**	Volume
10 Mar 2021	121.69	122.17	119.79	120.00	120.00	34,085,526
09 Mar 2021	119.03	122.06	118.79	121.09	121.09	129,159,600
08 Mar 2021	120.93	121.00	116.21	116.36	116.36	153,918,600

Figure 1

The dataset will be split into training (60%), validation (20%) and test (20%) sets. Each partition will correspond to a block of time across the 5 years. For example, the training set will account for oldest data with the test set corresponding the latest 20% of data.

This data will be held in a Panda's data-frame along with an appended column for our Twitter sentiment scores for each day spanning the last 5 years.

d. **Algorithms, Languages & Libraries**

Core Machine Learning Algorithm

- *Last Value*: taking the previous day close price and using it as the stock price We aim to predict the daily adjusted closing prices of several stocks using data from the previous five years. There are several different machine learning algorithms that can be used to predict stock prices. Instead of picking one single algorithm we will be testing multiple prior to releasing the final version of the application. The different algorithms we will be training our data with are:

- e prediction.
- *Moving Average*: the predicted value will be the mean of the previous n values. This n value is arbitrary and is a parameter than can be changed throughout the training process.
- *Linear Regression*: to find a line of best fit, or the regression line.
- *LSTM*: Long short-term memory. Used to predict stocks an arbitrary number of steps into the future. LSTM has five essential components which allows it to model both long-term and short-term data. A diagram of the LSTM model is shown in Figure 2.

All computation not directly involved with the construction of the web application will be done in Python 3. The web application itself will be completed in Java.

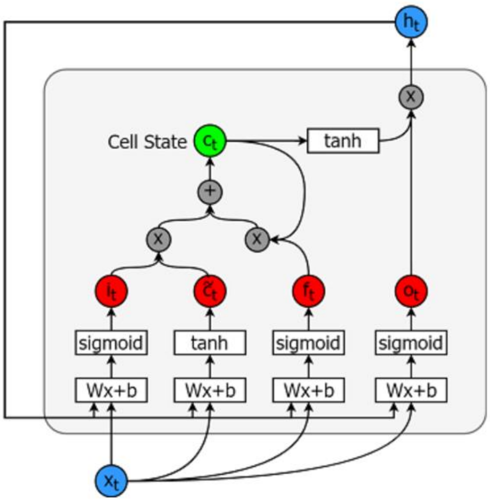


Figure 2
LSTM Architecture (our current favourite).

Current Library Selection:
Numpy, Pandas, TensorFlow, Keras, Seaborn, Google Sheets API, Twitter Developer API, NLTK Library, SKLearn (for evaluation metrics).

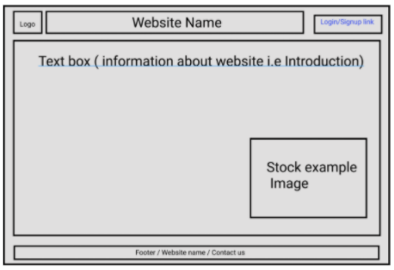
e. **Interface Design**

This is a storyboard of 6 of the pages the user will see. Starting with the welcome page which will mainly display information on what the site is about and what it can do. It will act as an introduction to the website and include buttons to enable the user to sign into an existing account or create a new one. The second page is the login page where the user will enter their account details. It also includes a button to sign up to the website for new users. It will also include buttons such as forgot password for users who have lost their login details.

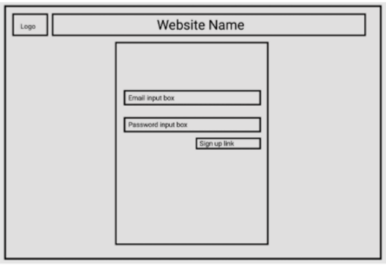
The third page is for new users who need to sign up, it also includes a questionnaire on what type of user they are. The 4th page is a home page which will display some trending stocks the user may be interested in. This will be determined from the information retrieved from twitter. The bookmark page stores stocks that users have specifically saved or "bookmarked". The final page is an example of what a single stock page will show. It also has a button to add or remove the stock to the previous bookmark tab.

The home and bookmark page also include a search bar for the user to navigate the site and search a specific stock that they have in mind which will then take them to the stock page. You will also notice that for most of the pages there is a contact us section at the bottom for users to contact a help team.

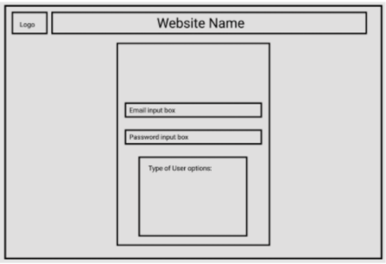
Welcome page


Logo
Website Name
Login/Signup link
Text box (information about website i.e Introduction)
Stock example Image
Footer / Website name / Contact us

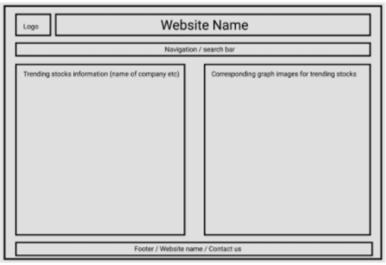
Log in / sign up page



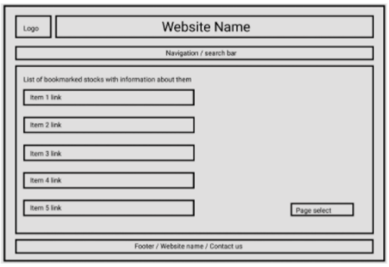
New user sign up page



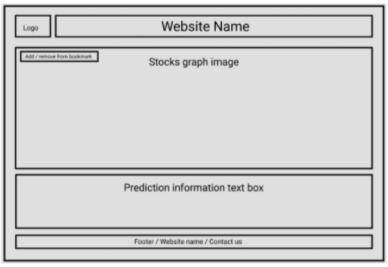
Home page
(navigation bar and trending stocks)

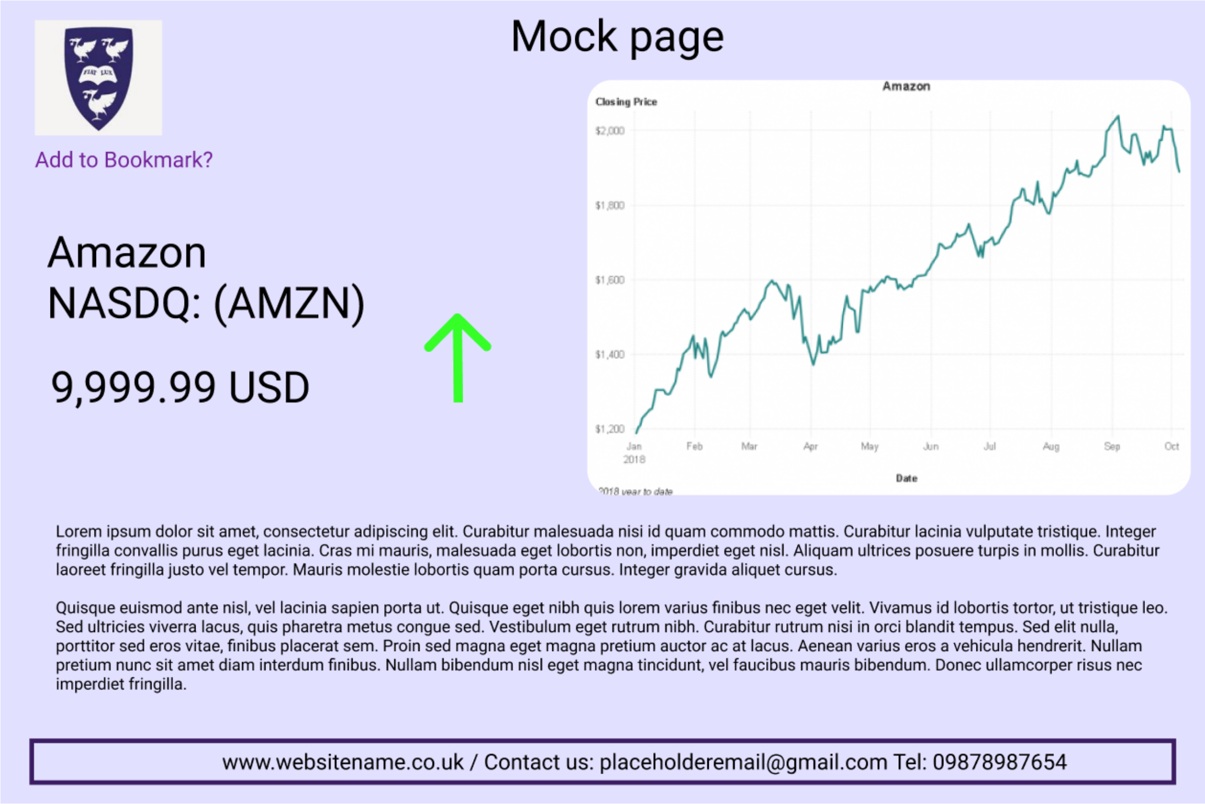

Logo
Website Name
Navigation / search bar
Trending stocks information (name of company etc)
Corresponding graph images for trending stocks
Footer / Website name / Contact us

Bookmark page



Stock page


Logo
Website Name
Add / remove from bookmark
Stocks graph image
Prediction information text box
Footer / Website name / Contact us



Mock page

Add to Bookmark?

Amazon
NASDQ: (AMZN)

9,999.99 USD

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur malesuada nisi id quam commodo mattis. Curabitur lacinia vulputate tristique. Integer fringilla convallis purus eget lacinia. Cras mi mauris, malesuada eget lobortis non, imperdiet eget nisl. Aliquam ultrices posuere turpis in mollis. Curabitur laoreet fringilla justo vel tempor. Mauris molestie lobortis quam porta cursus. Integer gravida aliquet cursus.

Quisque euismod ante nisl, vel lacinia sapien porta ut. Quisque eget nibh quis lorem varius finibus nec eget velit. Vivamus id lobortis tortor, ut tristique leo. Sed ultricies viverra lacus, quis pharetra metus congue sed. Vestibulum eget rutrum nibh. Curabitur rutrum nisi in orci blandit tempus. Sed elit nulla, porttitor sed eros vitae, finibus placerat sem. Proin sed magna eget magna pretium auctor ac at lacus. Aenean varius eros a vehicula hendrerit. Nullam pretium nunc sit amet diam interdum finibus. Nullam bibendum nisl eget magna tincidunt, vel faucibus mauris bibendum. Donec ullamcorper risus nec imperdiet fringilla.

www.websitename.co.uk / Contact us: placeholderemail@gmail.com Tel: 09878987654

Here is a rough mock-up of the general layout of one of the stock pages. The actual website will include a navigation bar to navigate between pages. However, the main layout of the stock pages will hopefully be like this, you can see that the main information on the stock is shown on the left, with a graph to accompany this information to its left. The information below will be information on the company so the user can gain a better understanding about what type of company the page is displaying.

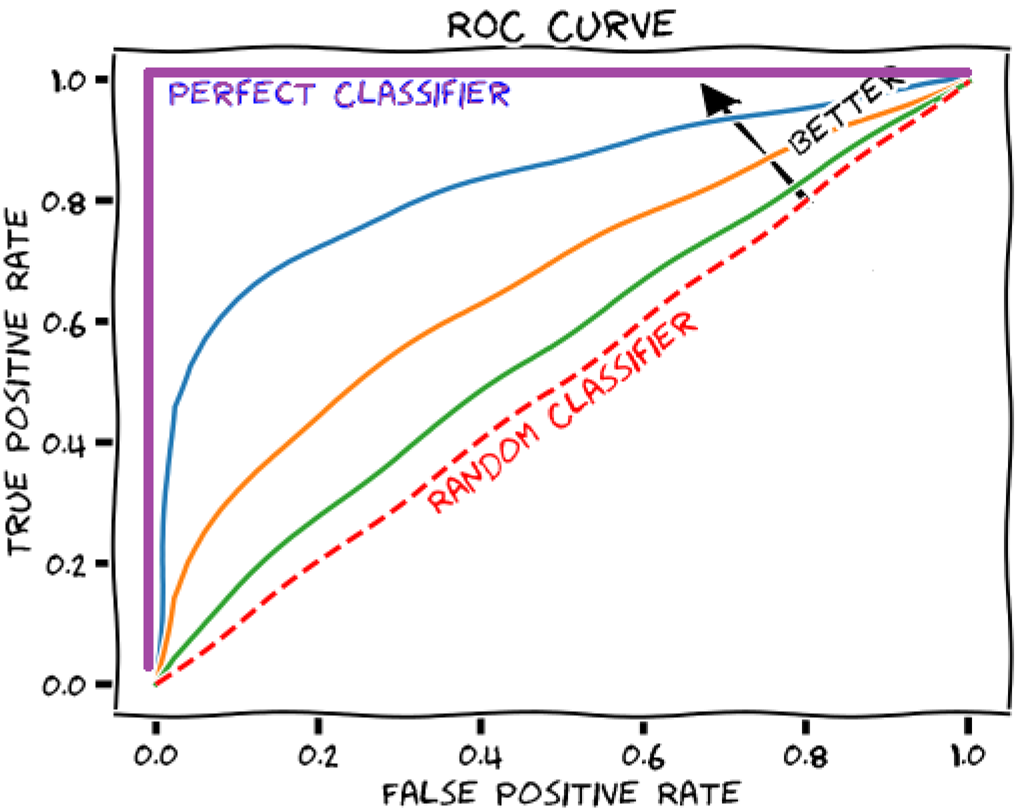
f. Evaluation of Results

To evaluate the effectiveness of the methods we can use the RMSE method, the root mean squared error. The lower the value, the better our prediction. We can then compare our predictions against one another to find the best algorithm for stock price prediction.

The SKLearn library contains a swathe of useful evaluation metrics. Our current favourites include:

- **ROC Curves**

A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.
It enables us to produce line graphs, which allow us to visually evaluate the predictive ability of several models against one another.

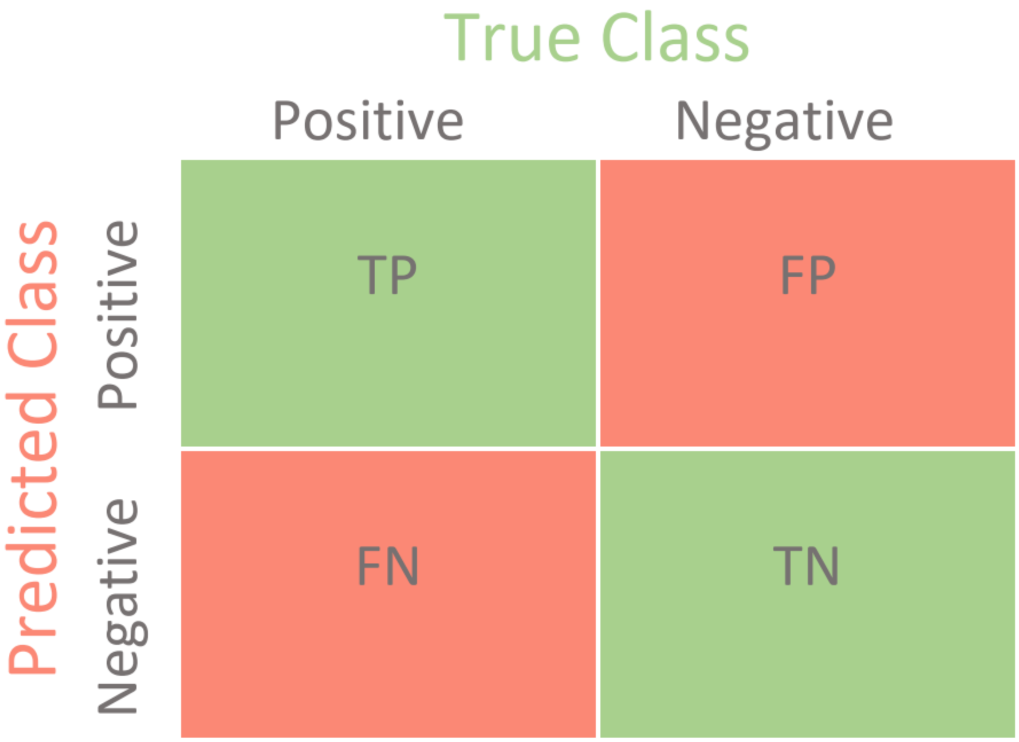


- ***k*-Fold Cross Validation**

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called *k* that refers to the number of groups that a given data sample is to be split into. Of the *k* subsamples, a single subsample is retained as the validation data for testing the model, and the remaining *k* − 1 subsamples are used as training data. The cross-validation process is then repeated *k* times, with each of the *k* subsamples used exactly once as the validation data.
This offers us a holistic evaluation of the model's predictive ability over unseen data.

- **Confusion Matrices.**

Confusion Matrices allow us to visually evaluate how well our model classifies its data and how badly it has mis-classified.
A key advantage is that it enables us to pinpoint which class of data the model is struggling with.

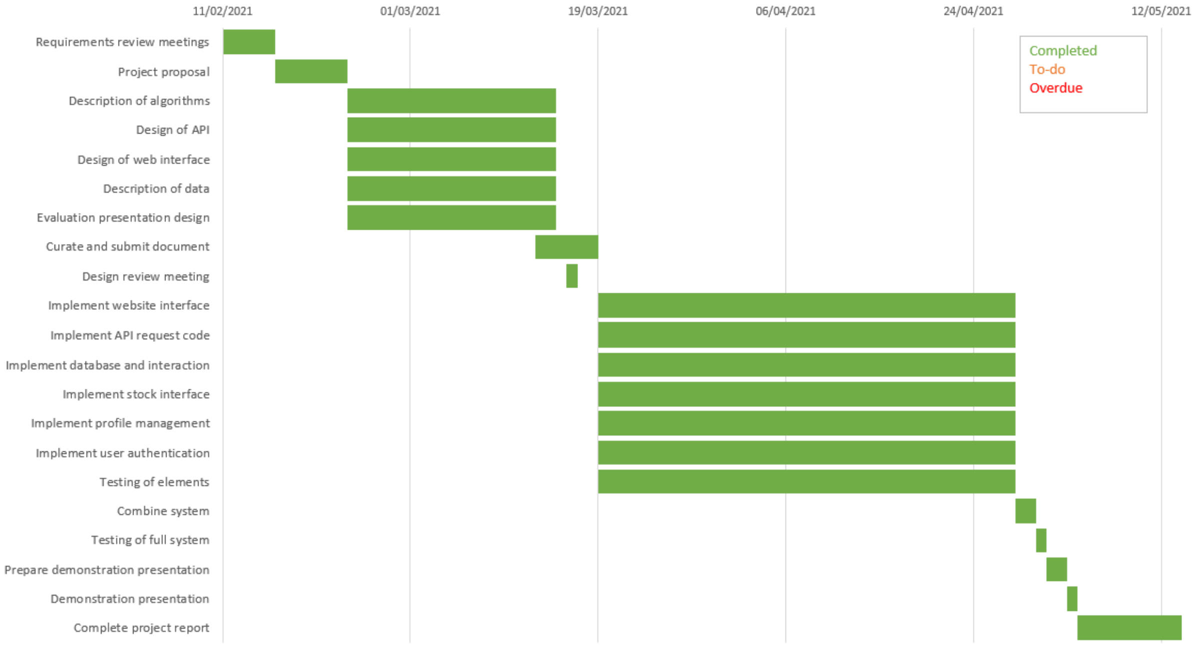


		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

g. **Anticipated Conclusion**

In conclusion, we anticipate that the model's predictive ability will improve with the addition of sentiment data from Twitter proving our hypothesis.

3. **Gannt Chart upon completion**



So far, we are happy with our productivity. We have been holding at least one meeting per week and believe that these have been helpful in assigning roles during the design process and ensuring that this part of the project was done in time. We are confident that if this continues, we will be able to produce our project in the given time frame.

4. References

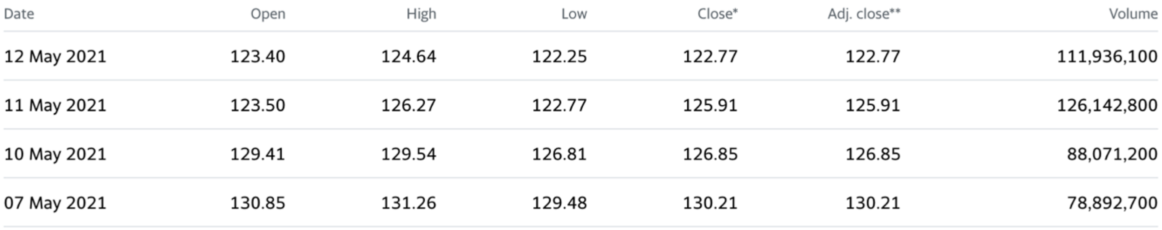
COMP 226: Trading and Financial Markets with *Prof. Rahul Savani*,
Machine Learning techniques applied to stock price prediction,
Stock Closing Price Prediction using Machine Learning Techniques – Research Paper.
https://developers.google.com/sheets/api/quickstart/python

3. Test Documentation

Data Used – historical stock data has been sourced from Yahoo Finance.

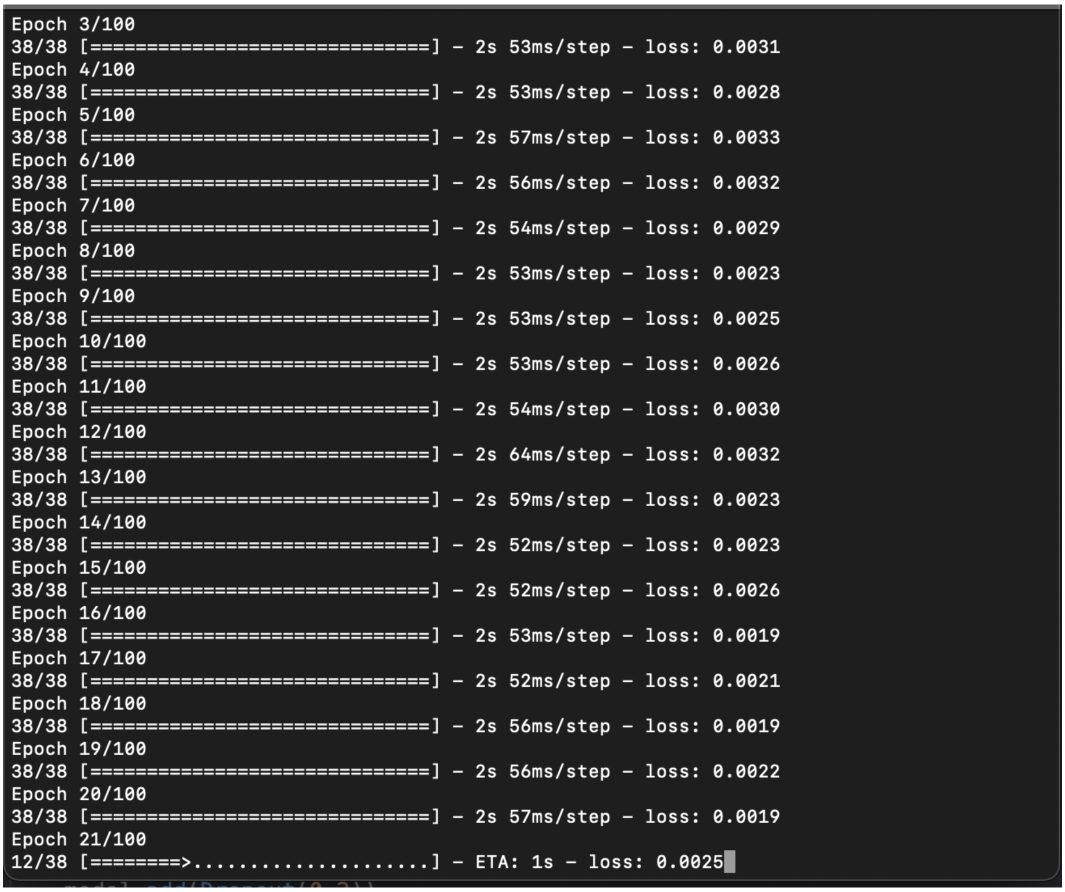
- Apple stock data (APPL): last 5 years data.
- Amazon stock data (AMZN): last 5 years data.
- Coca-Cola stock data (KO): last 5 years data.
- Nike stock data (NKE): last 5 years data.
- Google stock data (GOOGL): last 5 years data.

Example screenshot of Apple (APPL) test/training data.



Date	Open	High	Low	Close*	Adj. close**	Volume
12 May 2021	123.40	124.64	122.25	122.77	122.77	111,936,100
11 May 2021	123.50	126.27	122.77	125.91	125.91	126,142,800
10 May 2021	129.41	129.54	126.81	126.85	126.85	88,071,200
07 May 2021	130.85	131.26	129.48	130.21	130.21	78,892,700

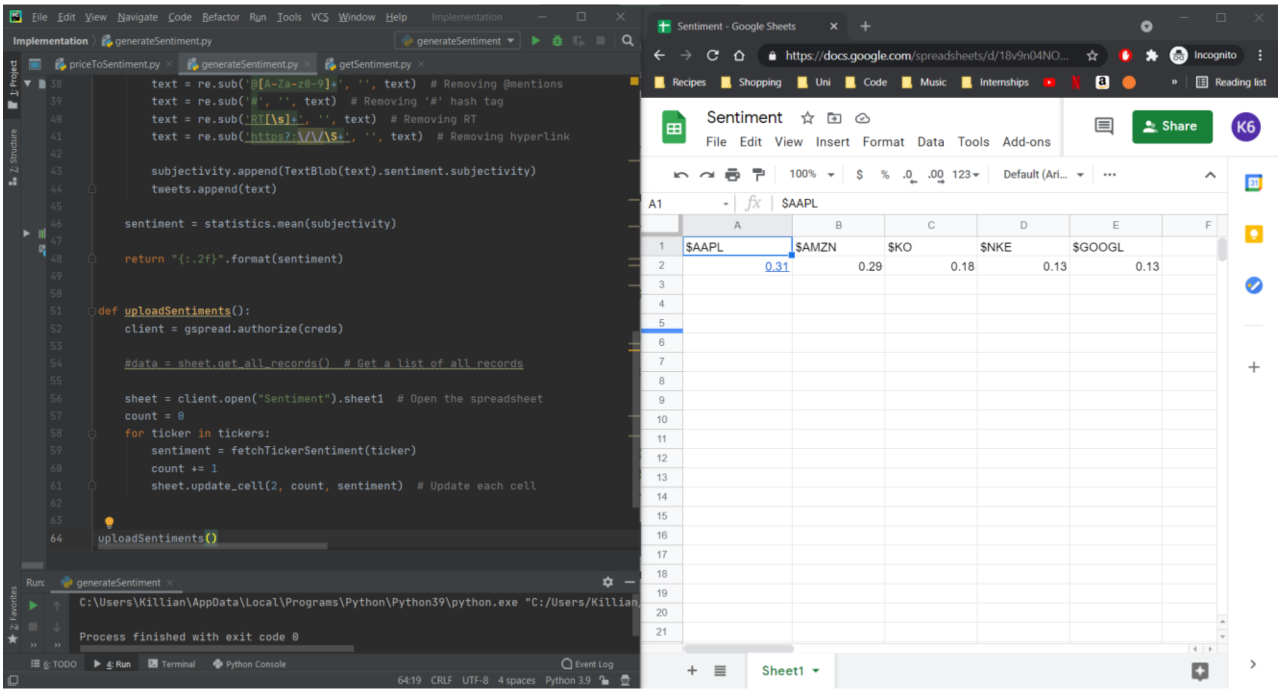
Screenshot showing the Machine Learning algorithm training a model on APPL dataset. You can see the loss function being minimised over the 100 epochs.



```
Epoch 3/100
38/38 [==============================] - 2s 53ms/step - loss: 0.0031
Epoch 4/100
38/38 [==============================] - 2s 53ms/step - loss: 0.0028
Epoch 5/100
38/38 [==============================] - 2s 57ms/step - loss: 0.0033
Epoch 6/100
38/38 [==============================] - 2s 56ms/step - loss: 0.0032
Epoch 7/100
38/38 [==============================] - 2s 54ms/step - loss: 0.0029
Epoch 8/100
38/38 [==============================] - 2s 53ms/step - loss: 0.0023
Epoch 9/100
38/38 [==============================] - 2s 53ms/step - loss: 0.0025
Epoch 10/100
38/38 [==============================] - 2s 53ms/step - loss: 0.0026
Epoch 11/100
38/38 [==============================] - 2s 54ms/step - loss: 0.0030
Epoch 12/100
38/38 [==============================] - 2s 64ms/step - loss: 0.0032
Epoch 13/100
38/38 [==============================] - 2s 59ms/step - loss: 0.0023
Epoch 14/100
38/38 [==============================] - 2s 52ms/step - loss: 0.0023
Epoch 15/100
38/38 [==============================] - 2s 52ms/step - loss: 0.0026
Epoch 16/100
38/38 [==============================] - 2s 53ms/step - loss: 0.0019
Epoch 17/100
38/38 [==============================] - 2s 52ms/step - loss: 0.0021
Epoch 18/100
38/38 [==============================] - 2s 56ms/step - loss: 0.0019
Epoch 19/100
38/38 [==============================] - 2s 56ms/step - loss: 0.0022
Epoch 20/100
38/38 [==============================] - 2s 57ms/step - loss: 0.0019
Epoch 21/100
12/38 [========>.....................] - ETA: 1s - loss: 0.0025
```

4. Sample Screen Shots

Uploading sentiment data from Twitter to Sheet



File Edit View Navigate Code Refactor Run Tools VCS Window Help Implementation
Implementation generateSentiment.py
generateSentiment
priceToSentiment.py generateSentiment.py getSentiment.py
1: Project
Z: Structure
2: Favorites
6: TODO
4: Run
Terminal
Python Console
Event Log
64:19 CRLF UTF-8 4 spaces Python 3.9
Sentiment - Google Sheets
https://docs.google.com/spreadsheets/d/18v9n04NO...
Incognito
Recipes Shopping Uni Code Music Internships
Reading list
Sentiment
File Edit View Insert Format Data Tools Add-ons
Share
K6
100%
Default (Ari...
Sheet1

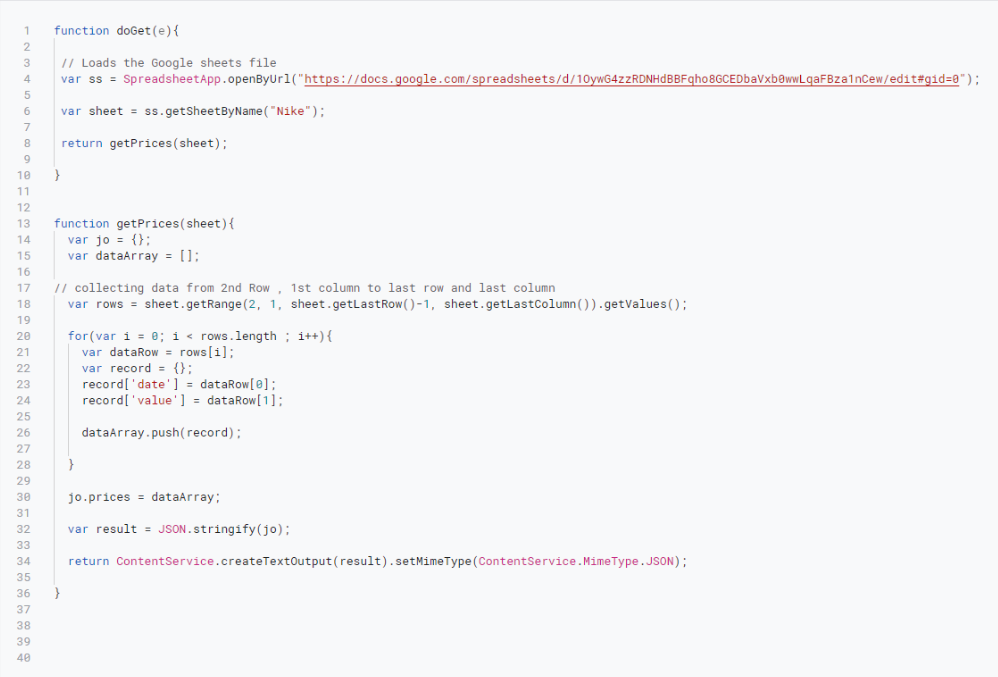
```python
            text = re.sub('@[A-Za-z0-9]+', '', text)  # Removing @mentions
            text = re.sub('#', '', text)  # Removing '#' hash tag
            text = re.sub('RT[\s]+', '', text)  # Removing RT
            text = re.sub('https?:\/\/\S+', '', text)  # Removing hyperlink

            subjectivity.append(TextBlob(text).sentiment.subjectivity)
            tweets.append(text)

    sentiment = statistics.mean(subjectivity)

    return "{:.2f}".format(sentiment)


def uploadSentiments():
    client = gspread.authorize(creds)

    #data = sheet.get_all_records()  # Get a list of all records

    sheet = client.open("Sentiment").sheet1  # Open the spreadsheet
    count = 0
    for ticker in tickers:
        sentiment = fetchTickerSentiment(ticker)
        count += 1
        sheet.update_cell(2, count, sentiment)  # Update each cell


uploadSentiments()
```

```
Run: generateSentiment
C:\Users\Killian\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/Killian

Process finished with exit code 0
```

A1 fx $AAPL

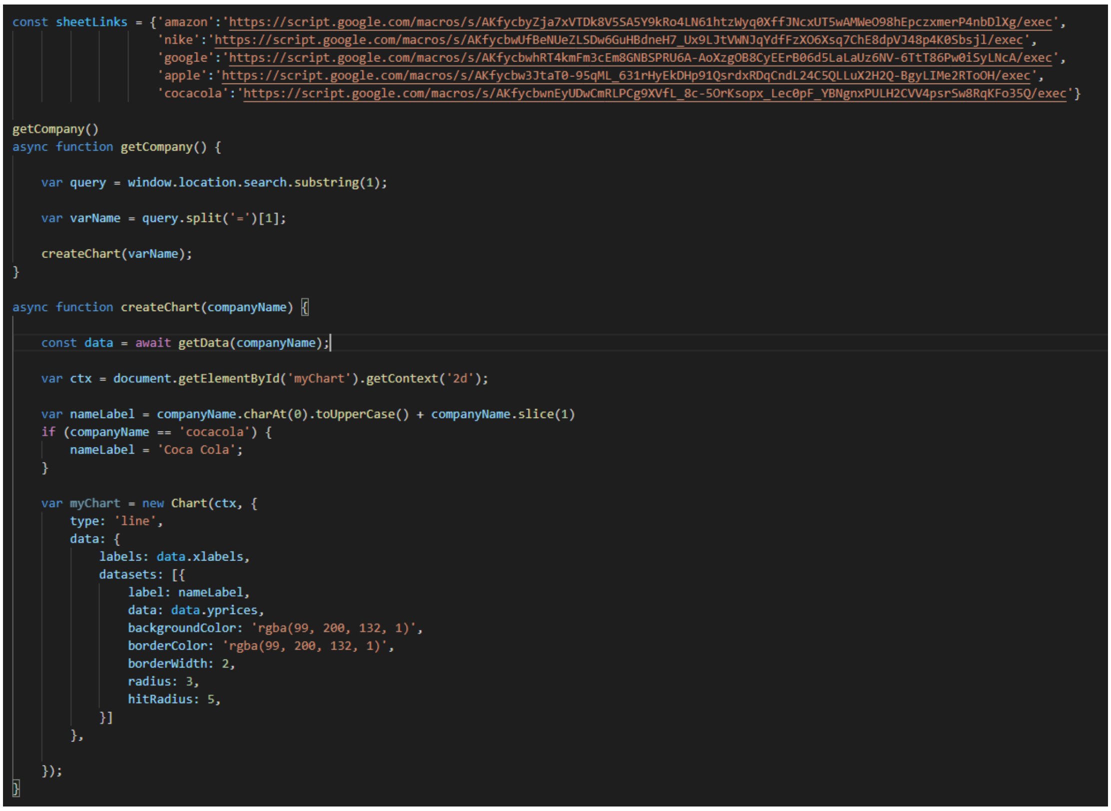
	A	B	C	D	E	F
1	$AAPL	$AMZN	$KO	$NKE	$GOOGL	
2	0.31	0.29	0.18	0.13	0.13	

Google Script used to give data from sheets in Json format –



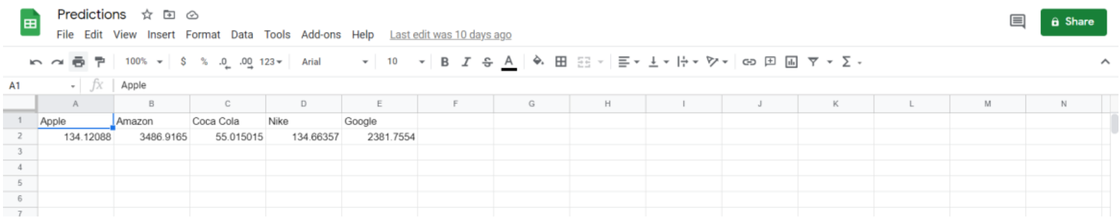
```javascript
function doGet(e){

 // Loads the Google sheets file
 var ss = SpreadsheetApp.openByUrl("https://docs.google.com/spreadsheets/d/1OywG4zzRDNHdBBFqho8GCEDbaVxb0wwLqaFBza1nCew/edit#gid=0");

 var sheet = ss.getSheetByName("Nike");

 return getPrices(sheet);

}


function getPrices(sheet){
  var jo = {};
  var dataArray = [];

// collecting data from 2nd Row , 1st column to last row and last column
  var rows = sheet.getRange(2, 1, sheet.getLastRow()-1, sheet.getLastColumn()).getValues();

  for(var i = 0; i < rows.length ; i++){
    var dataRow = rows[i];
    var record = {};
    record['date'] = dataRow[0];
    record['value'] = dataRow[1];

    dataArray.push(record);

  }

  jo.prices = dataArray;

  var result = JSON.stringify(jo);

  return ContentService.createTextOutput(result).setMimeType(ContentService.MimeType.JSON);

}
```

JavaScript code for graphs –



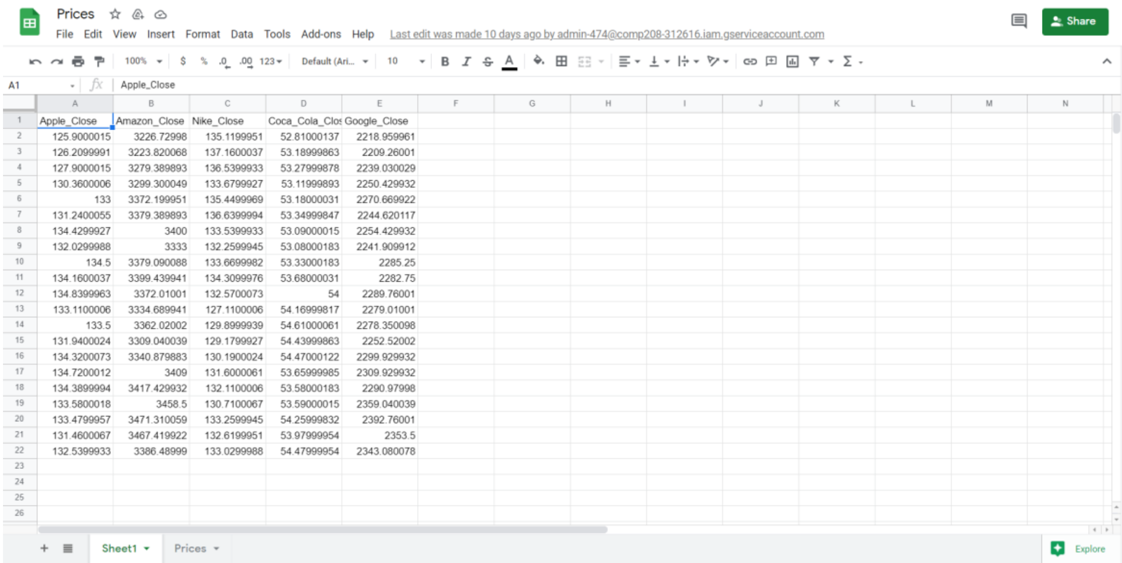
```
const sheetLinks = {'amazon':'https://script.google.com/macros/s/AKfycbyZja7xVTDk8V5SA5Y9kRo4LN61htzWyq0XffJNcxUT5wAMWeO98hEpczxmerP4nbDlXg/exec',
                    'nike':'https://script.google.com/macros/s/AKfycbwUfBeNUeZLSDw6GuHBdneH7_Ux9LJtVWNJqYdfFzXO6Xsq7ChE8dpVJ48p4K0Sbsjl/exec',
                    'google':'https://script.google.com/macros/s/AKfycbwhRT4kmFm3cEm8GNBSPRU6A-AoXzgOB8CyEErB06d5LaLaUz6NV-6TtT86Pw0iSyLNcA/exec',
                    'apple':'https://script.google.com/macros/s/AKfycbw3JtaT0-95qML_631rHyEkDHp91QsrdxRDqCndL24C5QLLuX2H2Q-BgyLIMe2RToOH/exec',
                    'cocacola':'https://script.google.com/macros/s/AKfycbwnEyUDwCmRLPCg9XVfL_8c-5OrKsopx_Lec0pF_YBNgnxPULH2CVV4psrSw8RqKFo35Q/exec'}

getCompany()
async function getCompany() {

    var query = window.location.search.substring(1);

    var varName = query.split('=')[1];

    createChart(varName);
}

async function createChart(companyName) {

    const data = await getData(companyName);

    var ctx = document.getElementById('myChart').getContext('2d');

    var nameLabel = companyName.charAt(0).toUpperCase() + companyName.slice(1)
    if (companyName == 'cocacola') {
        nameLabel = 'Coca Cola';
    }

    var myChart = new Chart(ctx, {
        type: 'line',
        data: {
            labels: data.xlabels,
            datasets: [{
                label: nameLabel,
                data: data.yprices,
                backgroundColor: 'rgba(99, 200, 132, 1)',
                borderColor: 'rgba(99, 200, 132, 1)',
                borderWidth: 2,
                radius: 3,
                hitRadius: 5,
            }]
        },

    });
}
```

Google Sheets database –



Predictions
File Edit View Insert Format Data Tools Add-ons Help
Last edit was 10 days ago
Share
A1
Apple

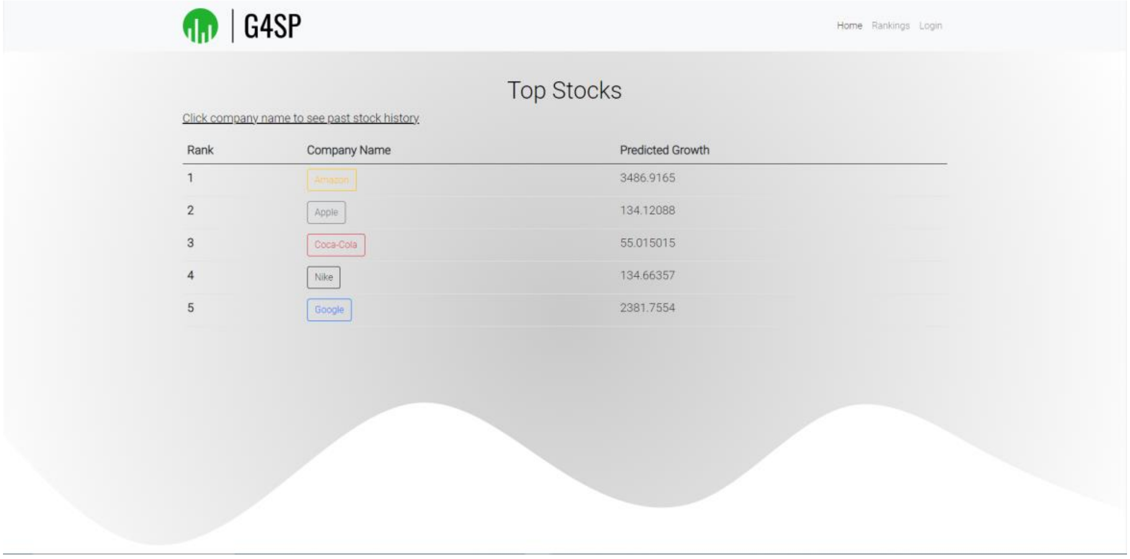
	A	B	C	D	E
1	Apple	Amazon	Coca Cola	Nike	Google
2	134.12088	3486.9165	55.015015	134.66357	2381.7554



Prices
File Edit View Insert Format Data Tools Add-ons Help
Last edit was made 10 days ago by admin-474@comp208-312616.iam.gserviceaccount.com
Share
A1
Apple_Close
Sheet1
Prices
Explore

	A	B	C	D	E
1	Apple_Close	Amazon_Close	Nike_Close	Coca_Cola_Clos	Google_Close
2	125.9000015	3226.72998	135.1199951	52.81000137	2218.959961
3	126.2099991	3223.820068	137.1600037	53.18999863	2209.26001
4	127.9000015	3279.389893	136.5399933	53.27999878	2239.030029
5	130.3600006	3299.300049	133.6799927	53.11999893	2250.429932
6	133	3372.199951	135.4499969	53.18000031	2270.669922
7	131.2400055	3379.389893	136.6399994	53.34999847	2244.620117
8	134.4299927	3400	133.5399933	53.09000015	2254.429932
9	132.0299988	3333	132.2599945	53.08000183	2241.909912
10	134.5	3379.090088	133.6699982	53.33000183	2285.25
11	134.1600037	3399.439941	134.3099976	53.68000031	2282.75
12	134.8399963	3372.01001	132.5700073	54	2289.76001
13	133.1100006	3334.689941	127.1100006	54.16999817	2279.01001
14	133.5	3362.02002	129.8999939	54.61000061	2278.350098
15	131.9400024	3309.040039	129.1799927	54.43999863	2252.52002
16	134.3200073	3340.879883	130.1900024	54.47000122	2299.929932
17	134.7200012	3409	131.6000061	53.65999985	2309.929932
18	134.3899994	3417.429932	132.1100006	53.58000183	2290.97998
19	133.5800018	3458.5	130.7100067	53.59000015	2359.040039
20	133.4799957	3471.310059	133.2599945	54.25999832	2392.76001
21	131.4600067	3467.419922	132.6199951	53.97999954	2353.5
22	132.5399933	3386.48999	133.0299988	54.47999954	2343.080078

Rankings Page –

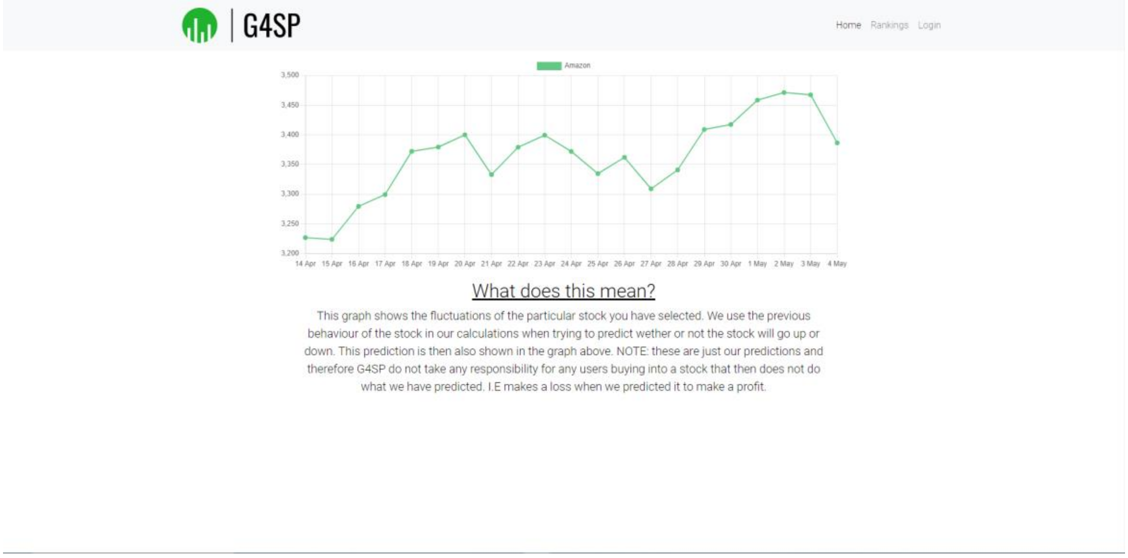


Top Stocks

Click company name to see past stock history

Rank	Company Name	Predicted Growth
1	Amazon	3486.9165
2	Apple	134.12088
3	Coca-Cola	55.015015
4	Nike	134.66357
5	Google	2381.7554

Stock Graph Page –



What does this mean?

This graph shows the fluctuations of the particular stock you have selected. We use the previous behaviour of the stock in our calculations when trying to predict wether or not the stock will go up or down. This prediction is then also shown in the graph above. NOTE: these are just our predictions and therefore G4SP do not take any responsibility for any users buying into a stock that then does not do what we have predicted. I.E makes a loss when we predicted it to make a profit.