An overview of the function of the code

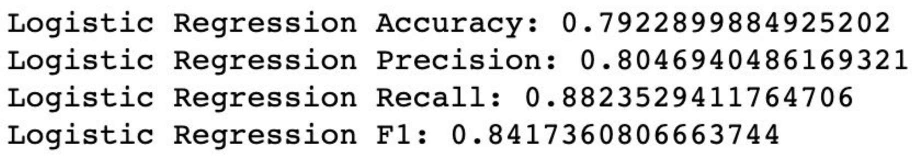
Our project aims to analyze stock sentiment from Twitter data to help our users understand whether or not to buy or sell stocks. Understanding sentiment from tweets is very helpful as it provides textual context to the market outlook of different stocks.

We started by running a pre-existing sentiment analyzer, Vader, to have a baseline performance. Then, we did research into different methods of sentiment analysis to build our own sentiment analyzer, which we trained on a kaggle dataset, and later tested on the Twitter API using recent tweets. We wanted our user to easily understand the sentiment of a stock's set of tweets, so we added a pie graph as a visualization for user's to make the stock purchasing decisions.

How software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.

We can consider the code to be in two different portions: creating and training the sentiment analyzer and actually running it with the tweets pulled from the Twitter API.

For our sentiment analyzer, we used knowledge from class to create a sentiment analyzer that utilizes TF-IDF and a Bag-Of-Words model. From MP1, we realized that tokenizing and lemmatizing the words and then using n-gram on words would be effective. We tried many tutorials to understand how to use SpaCy, but utilized https://www.dataquest.io/blog/tutorial-text-classification-in-python-using-spacy/ the most as it incorporated TF-IDF and the Bag-Of-Words model. While testing the model, we also played around with the n-gram_range that was fed into the BOW-vector and found that unigrams gave the best results.

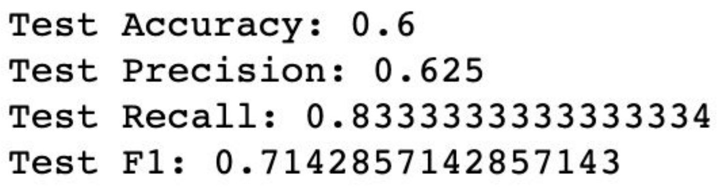


```
Logistic Regression Accuracy: 0.7922899884925202
Logistic Regression Precision: 0.8046940486169321
Logistic Regression Recall: 0.8823529411764706
Logistic Regression F1: 0.8417360806663744
```

Figure 1: Model Results

We used a kaggle dataset to train our model, which performed above the Vader benchmark. We ended up using Logistic Regression because it gave the best result compared to the other models on Scikit-Learn. We then used the Twitter API to pull the most recent tweets from Twitter pertaining to a certain stock ticker (input as '$tickerName'), run it through the model, and then output a pie chart that shows the user the percentage of tweets that had a positive and negative connotation. Based on this chart, the user can then make a judgement as to whether or not they want to short or long the stock.

In order to test the overall precision/recall/F1 of the entire program, we established the ground truths of the tweets that were pulled by ourselves and then calculated the metrics.



```
Test Accuracy: 0.6
Test Precision: 0.625
Test Recall: 0.8333333333333334
Test F1: 0.7142857142857143
```

Figure 2: Performance metrics of entire program

Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run a software, whichever is applicable.

Firstly, the user must have access to Jupyter notebook. There is a browser version, which the user can find with a google search for "Jupyter notebook browser."

Our application also uses the Twitter API. To be able to receive real time information, the user must create a developer portal on Twitter and gain Access Tokens and Keys. Additionally, the user must save their credentials in a json format to a file called: twitter_credentials.json so that the application can pull them. (e.g. {"CONSUMER_KEY": "", "CONSUMER_SECRET": "", "ACCESS_TOKEN": "", "ACCESS_SECRET": ""})

Once these two steps are complete, the user can proceed to the last cell of the Jupyter notebook to change the ticker name. The ticker name must be in a string format with a preceding '$'. To run our application, the user can click on Cell -> Run All, and they will see a pie graph with the sentiment distribution corresponding to that stock (an example can be seen below).

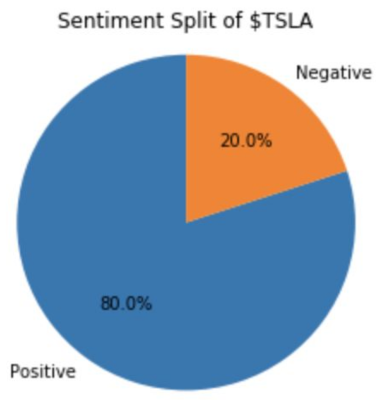


Figure 3: Example of Sentiment Distribution of Tesla

Brief description of contribution of each team member in case of a multi-person team.

Both members of the team had a very nice and collaborative experience with this project. In the first few weeks, both Asha and Sita researched different methods of sentiment analysis and tried to find tutorials that would be most helpful for them while creating it. Once the research phase was over, Asha mainly worked with coding the first part of the project as well as finding the benchmark metrics with Vader while Sita worked with the Twitter and graphing portion as she had a Twitter account. The full breakdown of tasks can be found below.

Task	**Asha**	**Sita**
Researching about sentiment analysis/analyzer tutorials	yes	yes
Vader benchmark with Kaggle dataset	yes	
Coding sentiment analyzer	yes	yes
Fine-tuning sentiment analyzer	yes	
Research about pulling tweets	yes	yes
Coding Real-Time Twitter Pull		yes
Coding Data Visualization		yes
Creating ground truths for testing/accuracy purposes with the Twitter results	yes	yes