Predicting Option Prices Through Sentiment Analysis of Volatility

By: Alex Roesler, Andrew Weiner, Brian Kwon

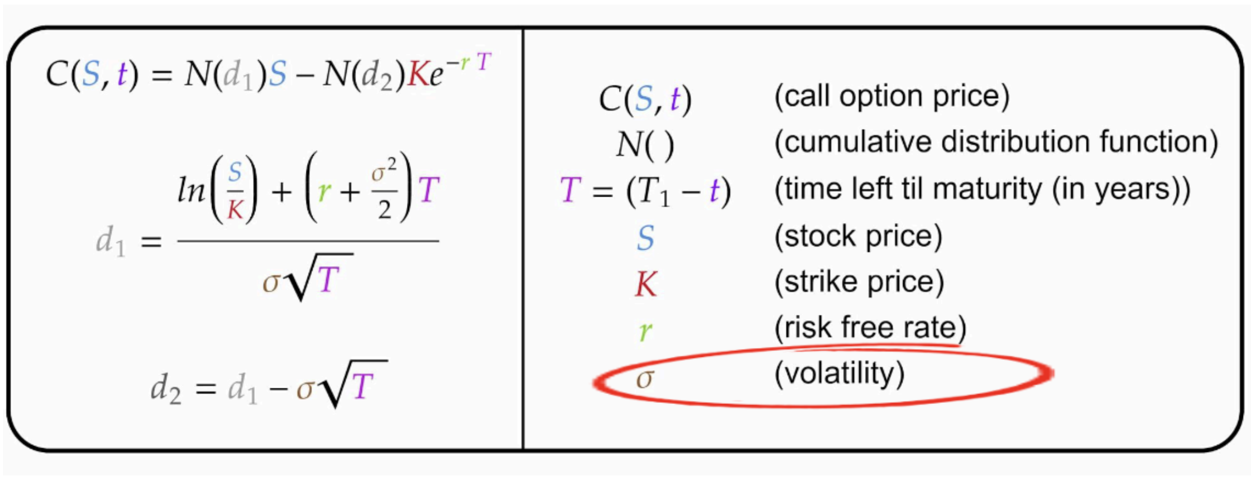
1. Introduction

Everyone wants to predict the market. This desire spans from financial companies with billions of dollars to casual hobbyists who trade their own pocket money. But very few people actually succeed because this task essentially boils down to trying to predict the unpredictable. George Box famously said “All models are wrong, some are useful.” So how can we create such a model ourselves? Perhaps we can take an existing “useful” model and put our own twist on it.

2. Background

There are a few key terms that are important to understand. First, an **option** is a financial derivative that gives someone the right (but not the obligation) to buy or sell an underlying asset at a predetermined price by a certain expiration date. There are two main types of options: American-style options can be exercised at any point before expiration, while European-style options can only be exercised at expiration.

The foundation of our project is to build on the **Black-Scholes Model** - a popular and classic model used for European-style option pricing. It calculates option prices using several numerical inputs, but it is grounded in several assumptions, including constant volatility. **Volatility** is the measure of how much the price of an asset fluctuates over time. In other words, it quantifies the uncertainty or risk associated with an asset.



$$C(S,t) = N(d_1)S - N(d_2)Ke^{-rT}$$

$$d_1 = \frac{ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

$C(S,t)$	(call option price)
$N(\,)$	(cumulative distribution function)
$T = (T_1 - t)$	(time left til maturity (in years))
S	(stock price)
K	(strike price)
r	(risk free rate)
σ	(volatility)

Figure 1: Black-Scholes Model - on the left are the equations to calculate the option price and the right shows inputs including volatility,

3. Problem

While widely adopted, the Black-Scholes model's reliance on its assumptions often leads to limitations in accurately capturing real-world scenarios, particularly in the presence of changing market conditions.Volatility can be high or low for many different reasons, but when it is used in the Black-Scholes model, the value for the input comes from a mathematical formula that annualizes historical data. We question whether this is truly the best way to determine volatility though considering how many qualitative factors such as geopolitics, social media, and human behavior can influence uncertainty. Thus, we decided to explore **sentiment analysis** as an alternative way to calculate volatility for the Black-Scholes Model.

4. Project Goal

The goal of our project is to develop a model in Python that gathers sentiment data, analyzes it, and then generates a custom value for the volatility of a particular stock. This volatility value is then inputted into a Black-Scholes model to price options based on sentiment analysis. Of course, our aim is for this model to make accurate predictions about the volatility of a stock so that the resulting option prices are reflective of the reality of the market. Regardless of the accuracy however, another goal of ours is to visualize our idea through an interactive interface that people can use to test different numerical inputs and stocks.

In the end, we did not achieve a consistently reliable level of accuracy, but we did discover some insightful correlations between volatility and different aspects of sentiment. Moreover, we also achieved our goal of building an interactive Black-Scholes option pricer that includes a sentiment-based calculator through a Python UI called Streamlit.

5. Techniques/Tools/Resources

5.1. Data Sources

A vast majority of the data in our project came from three sources - Yahoo Finance, Reddit, and Google News. Python offers a library called 'yfinance' that allows us to easily extract necessary data such as stock/option prices and volatilities. For Reddit, Python has a tool called 'PRAW' that gives users easy access to Reddit's API to extract content from a limited number of posts. And for Google News data, we employed Python's 'feedparser' and 'urllib.parse' modules to obtain RSS feed articles and parse through them.

5.2. Modeling Techniques

Since financial data often tracks changes over time, we knew we wanted to create a time series model to reflect this. Ultimately, we decided on a Long Short Term Memory (**LSTM**) Network, a type of Recurrent Neural Network (**RNN**). While traditional RNNs do update based on previous timesteps, key information often gets lost from earlier inputs that we would likely need to make our predictions since financial metrics like price and volatility from weeks prior can influence today. This is where LSTM comes in with its ability to retain memory that is essential for long-term dependencies, which can be valuable for identifying and analyzing complex patterns based on historical trends.

Next, we tried **Linear Regression,** which is a common statistical technique that aims to fit a linear correlation between an independent and dependent variable based on a set of data points. For us, the

To expand the scope of our project in hopes of achieving better results, we also tried a **Feed Forward Neural Network** - the simplest type of neural network since data only moves in direction without any feedback or loops. It is often used for predicting continuous values like house or stock prices, so we were curious if this contrast in simplicity from the LSTM model could result in anything insightful.

Finally, we combined the three previous models into an **Ensemble/Mixed Model** that performed a linear regression on the respective outputs for LSTM, Linear Regression, and Feed Forward Neural Network.

5.3. Tools

For sentiment analysis, we employed **TextBlob** - an open source Python library that provides a simple API for common natural language processing tasks. It served as our main tool for conducting sentiment analysis on relevant posts and articles about certain stocks. TextBlob works by analyzing an article's text and assigning a polarity score based on the content (-1: negative, 0: neutral, 1: positive). It then aggregates these polarity scores for a series of articles and generates an overall sentiment score for a specific stock over a certain period of time.

To organize our data, we used the **Pandas** library to generate and manipulate data frames. And to visualize this data, we utilized **matplotlib.pyplo**t to produce multiple graphs comparing a variety of correlations such as sentiment vs. volatility and article volume vs. volatility.

Finally, for our interactive interface, we discovered **Streamlit** - an open-source Python framework that can transform data scripts into web apps within minutes. Its user-friendly API enabled us to turn our model into an interactive dashboard where users can customize inputs for an option pricing calculator and compare our sentiment-based predictions against the Black-Scholes model's outputs.

6. Method

To execute this project in a streamlined and efficient manner, we designed a Machine Learning Pipeline to structure our workflow that can be summarized in the figure below.

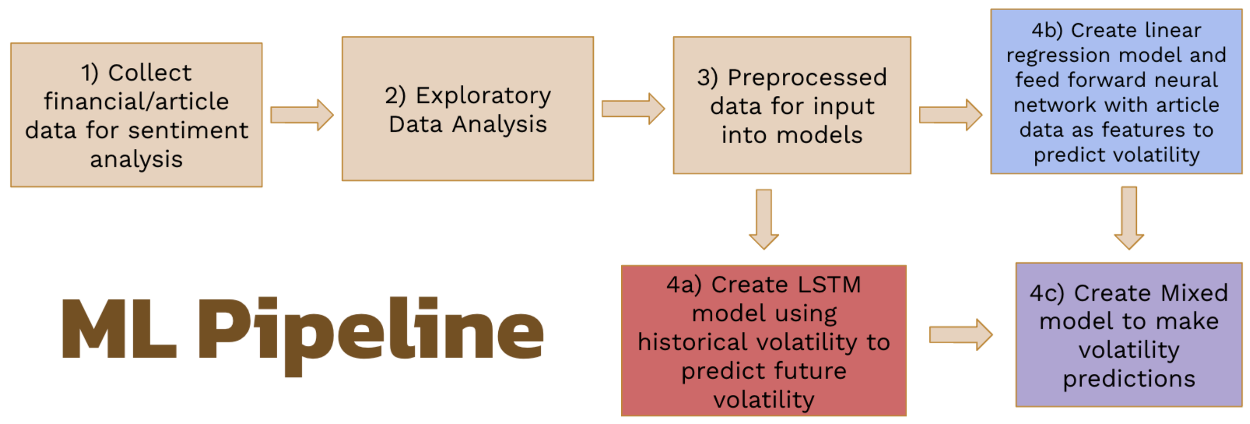

1) Collect financial/article data for sentiment analysis
2) Exploratory Data Analysis
3) Preprocessed data for input into models
4b) Create linear regression model and feed forward neural network with article data as features to predict volatility
ML Pipeline
4a) Create LSTM model using historical volatility to predict future volatility
4c) Create Mixed model to make volatility predictions

Figure 2: Flow chart of Machine Learning Pipeline that serves as the methodology for developing our mode

As you can see, this pipeline is ultimately able to produce four distinct types of models - an LSTM, Linear Regression, Feed Forward Neural Network, and a mixed ensemble model (combining LSTM, Linear Regression, and Feed Forward Neural Network).

7. Data Workflow

7.1. Data Extraction & Pre-Processing: Reddit Data

For the Reddit portion of the model, we discovered that PRAW's date filtering feature did not work as intended. To address this, we used PRAW's search() function to retrieve posts related to $SPY and manually filtered them by date using the submission.created_utc attribute. We then calculated sentiment scores for each post by analyzing its title, body, and all associated comments. To weight sentiment scores, subjectivity values were scaled from [0, 1] to [1, 10] and multiplied by polarity values (ranging from [-1, 1]). This method gave higher subjectivity posts more influence in the overall sentiment score. The combined sentiment score for each post was calculated by averaging the sentiment of the title + body with the average sentiment of the comments. For each post, we compiled the date, title, body, query used, title + body weighted polarity, title + body polarity, title + body subjectivity, comments weighted polarity, and overall combined sentiment score into a DataFrame. We then aggregated these metrics by day, averaging the sentiment scores and counting the number of posts per day, and saved the resulting daily data to a JSON file for use in volatility prediction.

Beyond sentiment analysis, we calculated $SPY's historical volatility using a rolling 30-day window on log returns and annualized the results. We also retrieved $VIX index values, which represent the annualized implied volatility of a hypothetical $SPY option with 30 days to expiration, and converted these values from percentages to decimal form for comparison with our model's predictions. For data preprocessing, sentiment scores and post counts were loaded from the JSON file into a pandas DataFrame, where duplicate entries were removed. To align sentiment scores and post counts with trading days, scores from non-trading days were averaged and assigned to the next trading day, while post counts were summed and similarly assigned. This ensured that all data, including sentiment scores, post counts, historical volatilities, and

$VIX values, was correctly aligned with trading days, enabling accurate model training and evaluation. We then used MinMax scaling on the sentiment scores, post counts, and historical volatilities before using them in our LSTM model.

Between MinMax scaling and feeding data into the model, several preprocessing steps are performed. First, the scaled data (containing sentiment scores, post counts, and volatility metrics) is transformed into sequences using a rolling window approach with a lookback period of 14 days. Each sequence consists of 14 consecutive days of input features, and the target variable is the volatility on the following day. These sequences are used to create the training dataset (X for inputs and y for targets). The resulting arrays are reshaped to match the input format expected by the LSTM model, which requires 3D data with dimensions representing the number of samples, the time steps (lookback period), and the number of features.

7.2. Data Extraction: Google News Data

For our Google News articles, we collected weekly sentiment data (average sentiment and article volume) along with weekly stock data for 5 different stocks, BBIO, RCKT, RNA, META, and TSM. Three of these are pharmaceutical companies which we chose to investigate, because their stock prices are most subject to change after big news. We only gathered data for 5 different stocks for the past year (November 2023 - December 2024), because sentiment analysis was computationally expensive. However, when we attempted to create models to predict volatility using our weekly data, the models were very inaccurate, and we attributed this to a sparsity/lack of data:

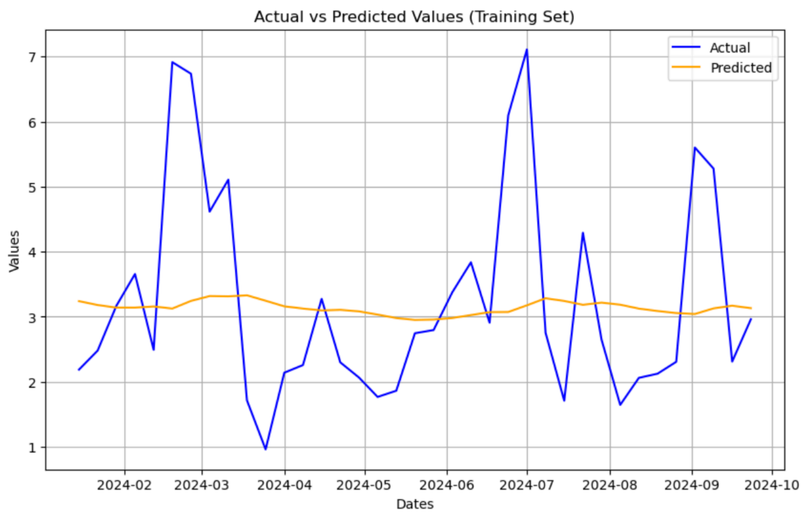


Figure 3: Initial attempt at LSTM on weekly data for BBIO

Therefore, we instead chose to focus on one stock, TSM, and collect daily stock and sentiment data for model training. Our results show that the switch to daily data was effective.

7.3. Exploratory Data Analysis

For exploratory data analysis, we visualized the trends in sentiment scores and post counts over the range of dates used for our model. This range was relatively low because we mainly wanted to focus on short term trends in volatility, as we were using predicted volatilities to predict the price of options with short term expiration dates (30 days into the future). Additionally, it took our Reddit data extraction code about 30 minutes to run, so we decided to stick with the data we originally collected from 10/31 - 11/29. It took a while because in each search, we pulled 1000 posts for the given query, and then had to filter these 1000 posts to see if they were posted on the correct date. The visualizations for our sentiment scores and post counts are shown below. We did notice a significant dip in sentiment scores on the day of the US presidential election which is noteworthy.

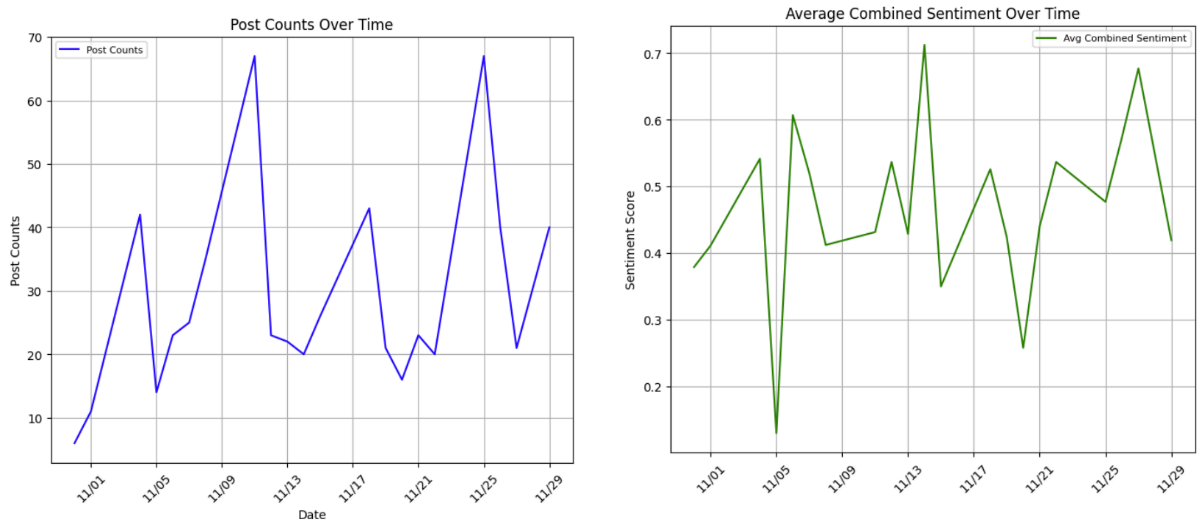


Figure 4: Daily Sentiment Scores and Daily Post Counts from 10/31 to 11/29

Moreover, using the Google News Feed data, we were able to notice some interesting correlations between a stock and the sentiment surrounding it. Although the correlation between a stock and its sentiment seemed low overall, some things stood out. For example, in the graph below, we see that there was a surge in sentiment surrounding the pharmaceutical stock, RCKT (Rocket Pharmaceutical), in early June, 2024, immediately followed by a substantial increase in ROC (Rate of Change) of the stock:

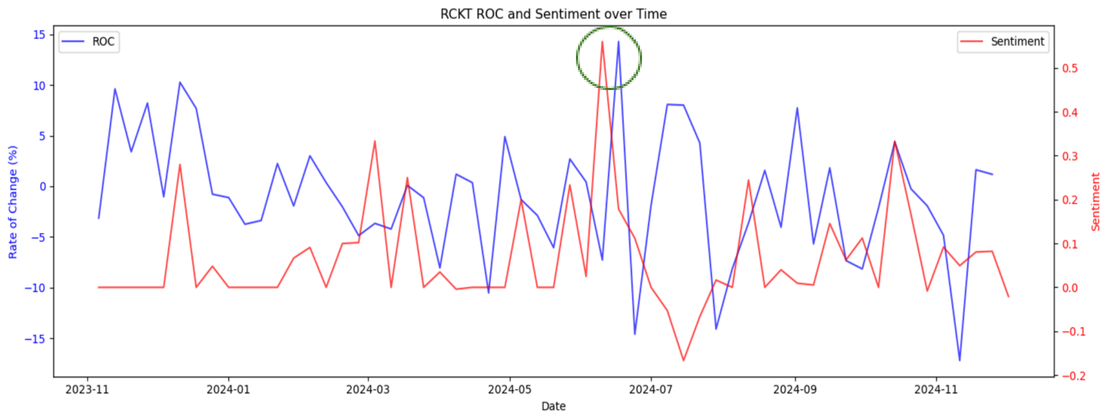
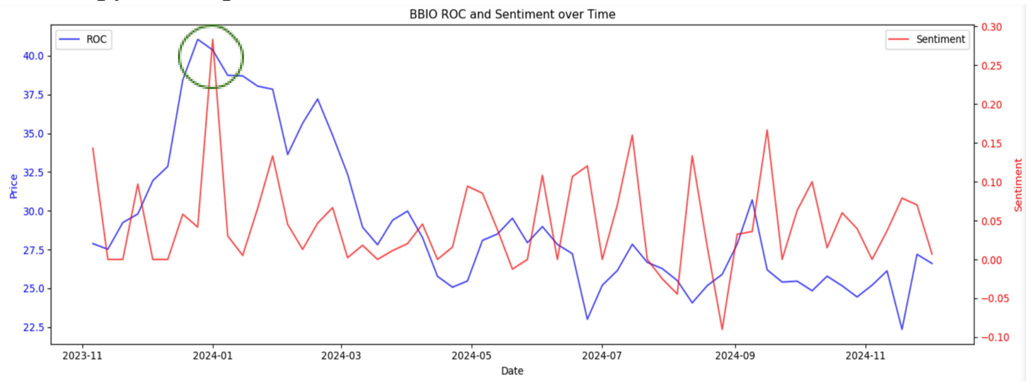
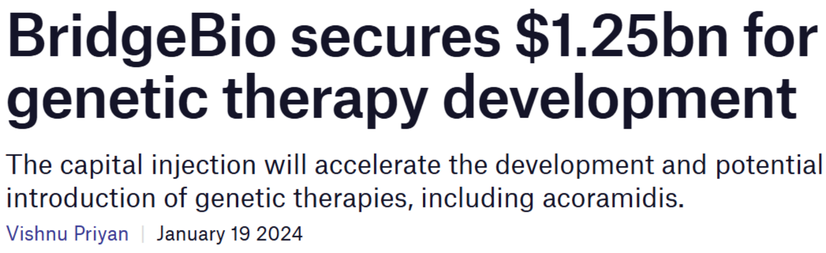


Figure 5: RCKT sentiment vs ROC of stock overtime with coinciding peaks

Also, to make sure that our sentiment analysis was meaningful we looked at some notable spikes in sentiment across different stocks and investigated them case by case. One such case is shown below, where the pharmaceutical stock, BBIO (BridgeBio Pharmaceutical), had a sharp increase in sentiment in early January, 2024. After some brief investigation, we found that the cause for this jump in sentiment was because, at the time, BBIO had acquired $1.25billion in investment for genetic therapy development.



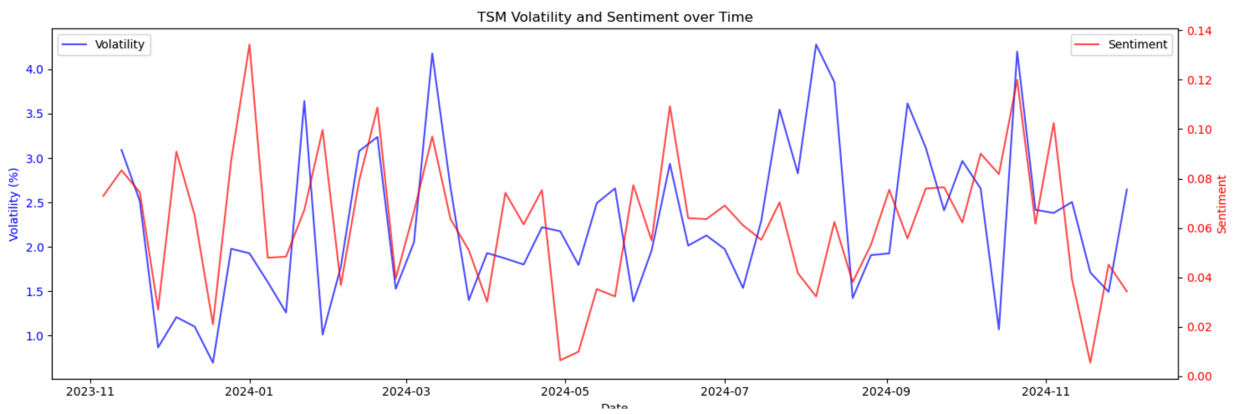

BridgeBio secures $1.25bn for genetic therapy development

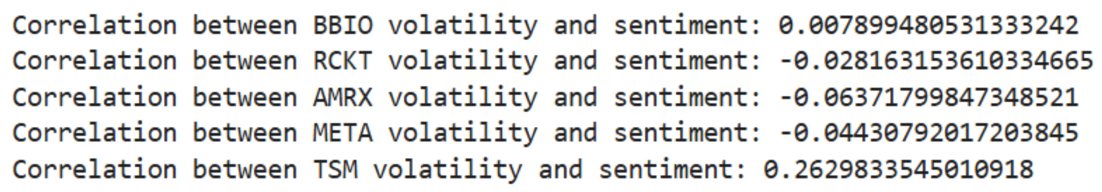
The capital injection will accelerate the development and potential introduction of genetic therapies, including acoramidis.

Vishnu Priyan | January 19 2024

Figure 6: A spike in BBIO's sentiment and a news article that details the reasons behind it

We eventually settled on one stock, TSM (Taiwanese Semiconductor Company) to create predictive models for, because it had the strongest correlation between sentiment and volatility over time, which is the relationship we are trying to use:





```
Correlation between BBIO volatility and sentiment: 0.007899480531333242
Correlation between RCKT volatility and sentiment: -0.028163153610334665
Correlation between AMRX volatility and sentiment: -0.06371799847348521
Correlation between META volatility and sentiment: -0.04430792017203845
Correlation between TSM volatility and sentiment: 0.2629833545010918
```

Figure 7: TSM's sentiment aligns well with its volatility by inspection and the numbers back it up

8. Models & Results

8.1. Initial Model

By the end of our project, we had two main working models - the second one being a pivot after experiencing several setbacks from the first. The LSTM model used on the Reddit data consists of two LSTM layers with 50 units each. The first LSTM layer is configured to return sequences, allowing for multi-layered processing of the input data. Dropout layers with a rate of 20% follow each LSTM layer to prevent overfitting. A fully connected Dense layer with a single neuron is used at the end to output the predicted volatility value. The model is compiled with the Adam optimizer and a mean squared error (MSE) loss function, making it well-suited for regression tasks. For our training process, our model iterates over 50 epochs with a batch size of 32. This ensures that the model learns the patterns in the data effectively while balancing computational efficiency.

The results from the Reddit portion of our project were disappointing, leading us to shift focus to the Google News portion. The graphs below illustrate that our model struggled to predict future volatility accurately. While it did capture the general downtrend in implied volatility toward the end of the test date range—something not reflected in realized historical volatility—the overall performance was unsatisfactory. Metrics further highlight the limitations of the Reddit-based features: the correlations between sentiment scores, post counts, and historical volatilities were all below 0.3 in magnitude, indicating minimal overlapping contribution from these features to the predictions.

Additionally, we calculated predicted call prices for $SPY options expiring in 30 days using the Black-Scholes model and compared them to the actual values for calls with the same strike prices. Unfortunately, our model incorrectly forecasted a decline in call option prices over time, while the actual prices increased. Although there was a high correlation of 0.95 between our predicted and actual option prices, this metric overstates the model's accuracy. The key issue lies in the model's failure to correctly predict the future direction of option price movements, rendering its predictions ineffective for practical use.

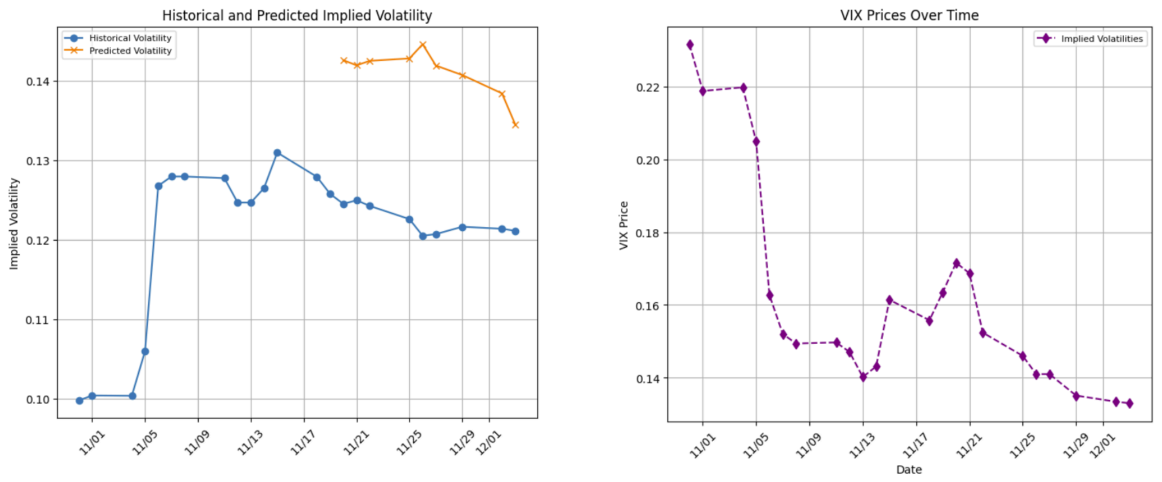
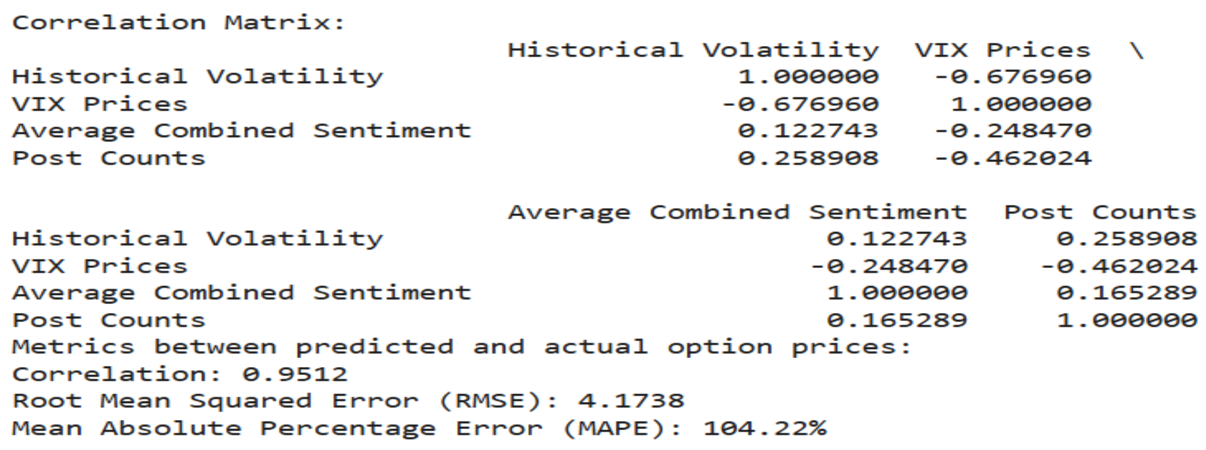


Figure 8: Historical vs. Predicted Volatility vs. Actual Implied Volatility Trends



```
Correlation Matrix:
                            Historical Volatility  VIX Prices  \
Historical Volatility                   1.000000   -0.676960
VIX Prices                             -0.676960    1.000000
Average Combined Sentiment              0.122743   -0.248470
Post Counts                             0.258908   -0.462024

                            Average Combined Sentiment  Post Counts
Historical Volatility                         0.122743     0.258908
VIX Prices                                   -0.248470    -0.462024
Average Combined Sentiment                    1.000000     0.165289
Post Counts                                   0.165289     1.000000
Metrics between predicted and actual option prices:
Correlation: 0.9512
Root Mean Squared Error (RMSE): 4.1738
Mean Absolute Percentage Error (MAPE): 104.22%
```

Figure 9: Correlations Between Model Features, Predicted Option Prices, Actual Option Prices, and Error Metrics

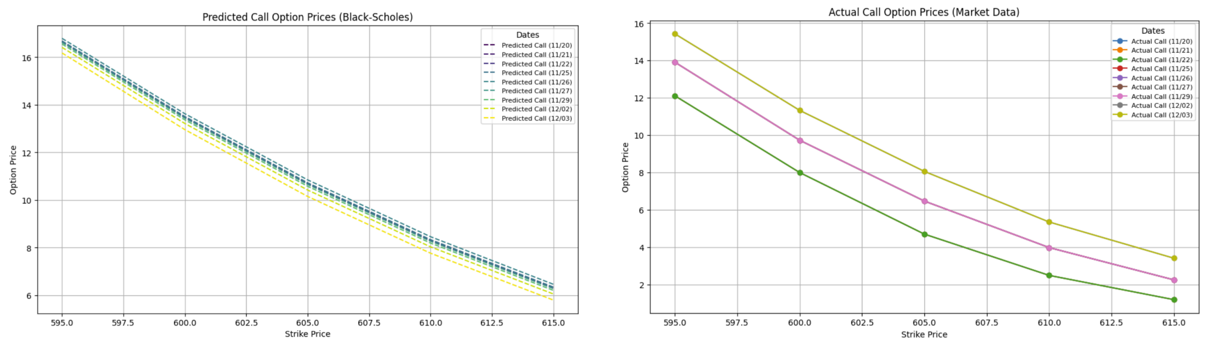


Figure 10: Predicted vs. Actual $SPY Call Option Prices (30-Day Expiration)

8.2. New LSTM Model

After pivoting, we developed a new LSTM model using the time-series element of volatility to predict future volatility based on prior volatilities.

We preprocessed the data by using a MinMax scaler on the volatility data and then created the input and target variables, where the target was simply the volatility, and the input was a lookback window of volatility 10 days prior. We opted to use a lookback window of 10 days, because 10 days was long enough to capture short term trends in the data, and the long-term memory aspect of the LSTM did not make it necessary to extend this window.

We used Adam as our optimizer and used Mean Squared Error as our error function, because we felt that this would force the model to learn the often drastic changes that volatility makes in short periods of time.

We created a train set using the first 80% of the time series data, a validation set using the next 10% and a test set using the final 10%. The loss of models calculated on the validation data was used to find the 'best' model in the gridsearch detailed below.

To optimize the hyperparameters during training, we created a **class**, LSTMGridSearch, that took in a grid of parameters as input and outputted the best model that was found. Our parameter grid consisted of: lstm_units = [64, 128], dense_units: [32, 64], batch_size: [16, 32], learning_rate: [0.001, 0.0001], epochs: [10, 20]. Eventually after training all 32 models, the optimal parameters were: {'batch_size': 16, 'dense_units': 64, 'epochs': 20, 'learning_rate': 0.001, 'lstm_units': 128}

Our best model had a MSE on the test set of 0.01

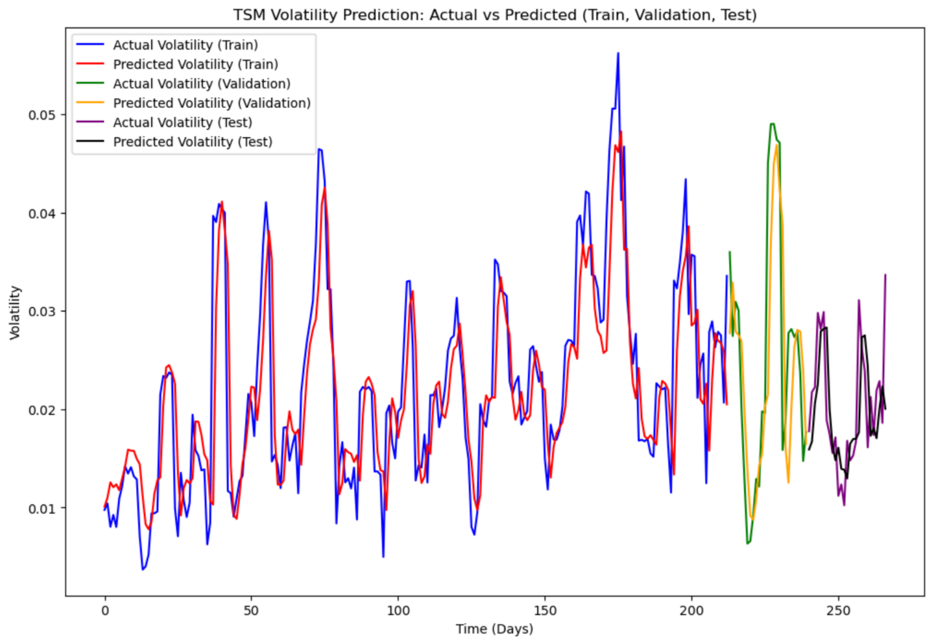


Figure 11: The LSTM model's predictions against the actual values on the train, validation, and test sets

8.3. Linear Regression Model

In our linear regression model we did not use the time-series aspect of the data. Instead we created a model that predicted the volatility at a certain time, given the sentiment data at a certain time, irrespective of the changes happening before or after the time.

For our preprocessing, we decided to create new features in the data based on the sentiment features we currently had, namely: Average sentiment squared, article volume squared, and average sentiment multiplied by article volume. We did this in hopes that the linear regression model would find/capture non-linear relationships between the sentiment data and volatility.

We did not scale the features beforehand, because it did not seem to have an effect on the model results or convergence, possibly due to the scale of the inputs.

Again, we created a train-val-test split of 80-10-10 randomly throughout the data (so not dependent on time unlike for the LSTM split). Then we created our input with our features, namely: [average sentiment, article volume, average sentiment^2, article volume^2, sentiment*volume]

We then trained a multiple linear regression model using sklearn's Ridge Regression Model and tuned the regularization parameter, alpha, as a hyperparameter. We ended up trying Alpha = 0.01, 0.1, 1, 10, and 100. Ultimately, alpha = 100 was the most successful with a MSE on the test set of 0.01

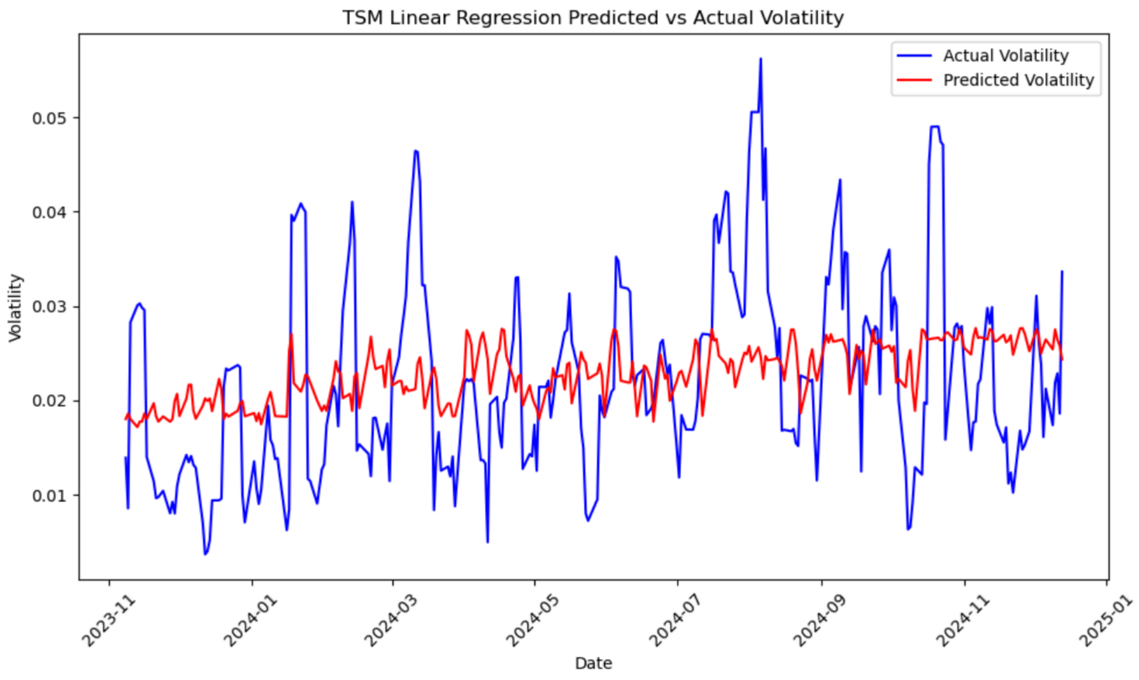


Figure 12: The linear regression model predictions vs actual volatilities

8.4. Feed Forward Neural Network

We performed a similar process for training our neural network. For our features we simply used article volume and average sentiment, because a neural network is good at finding non-linear relationships within the data.

We created a neural network using Tensorflow composed of two hidden Dense layers, both using reLu as an activation function. We tested a range of different numbers of units in each layer and different learning rates as described below.

Like with the LSTM, to tune our hyperparameters we performed grid search using our parameter grid of: units_1 = [32, 64, 128], units_2 = [16, 32, 64], learning_rate = [1e-4, 1e-3, 1e-2]. Ultimately, the best parameters were: {'units_1': 128, 'units_2': 32, 'learning_rate': 0.001}

Again, we used mean squared error as our loss function and ended with a test loss of around 0.0001

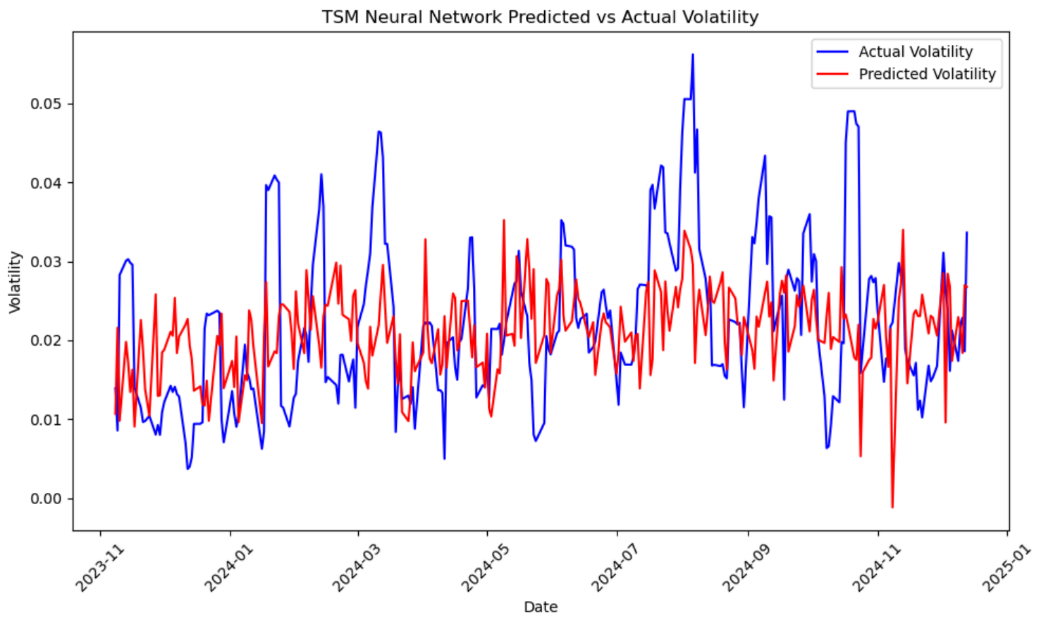


Figure 13: Predictions using Neural Network vs actual volatilities

8.5. Mixed Model

Finally, we created a mixed model that combined the results of our previous models into one to predict volatilities based on our previous models’ predictions.

The mixed model had a final Test MSE of 0.00003, the best out of all the previous models

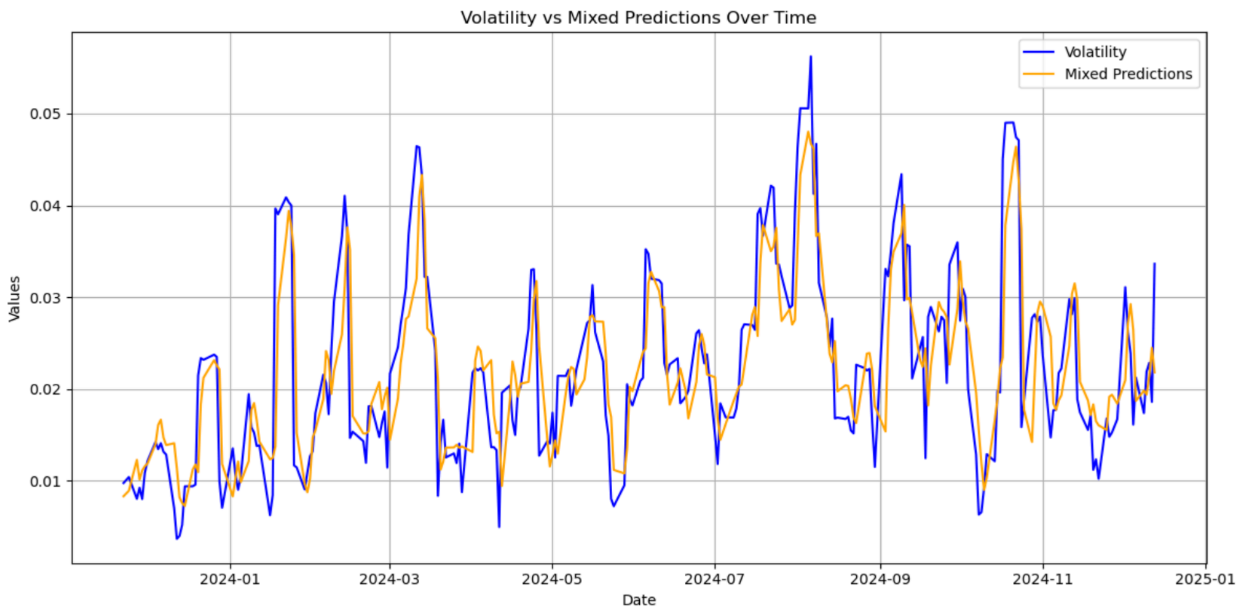


Figure 14: Predictions using Mixed model vs actual volatilities

9. Advantages & Limitations

We believe that the biggest advantage of our project is that it provides a novel and nuanced approach to options pricing by examining the relationship between sentiment and volatility. Furthermore, the overall structure of our models can be adapted to conduct sentiment analysis using different sources of information and incorporate other natural language processing tools. (Insert specific examples)

Of course, the actual accuracy of our model is a major limitation of our project, but from a process standpoint, the biggest limitation was data collection. Throughout this project, we realized how difficult, expensive, and time-consuming it can be to gather and process sentiment data. For example, we initially hoped to utilize Twitter's API to extract tweets for sentiment analysis, but this would require us to pay a significant amount of money for a subscription. One way we struggled with time constraints was how our Reddit data gathering script had to perform a 1000 item search for each query for each date (taking over 30 minutes), so we decided to just run it once and export the data to a JSON file that we could import everytime we ran our code for the LSTM model with the Reddit data. A limitation with this approach was a very limited amount of Reddit data, so our model likely had trouble learning patterns in volatility because it only had around 20 days of sentiment scores and post scores to work with.

Similarly, with the Google News data, extracting and processing the data was extremely time-consuming, so we had to limit the weekly data to five stocks and the daily data to just due to time and computational restraints. In order to test the true viability of our model and get a better understanding of how to improve it, we need significantly more data and test cases.

10. Conclusion & Future Work

In the end, the mixed model seemed to produce the best results as it had the lowest test error of MSE = 0.00003. Although the LSTM's predictions looked like they aligned with the actual volatilities very closely, its predictions tend to lag behind the actual values which accounts for the larger MSE.

Even though we had no breakthrough models, our results are still promising, and it appears that the mixed model in particular was effective in incorporating time series volatility data and sentiment data to estimate the volatility at a given moment. The models could even predict the volatility of the following day given the data from the previous. However, none of our models were accurate when it came to predicting volatilities that extended further into the future, i.e. no model could accurately simulate change in volatility over a prolonged period of time.

Take the following graph below for example. This graph was created using the LSTM, and volatilities during the test interval were "simulated" by the LSTM, meaning the LSTM calculated new values using its own values that it calculated previously. Obviously the results are not the best.

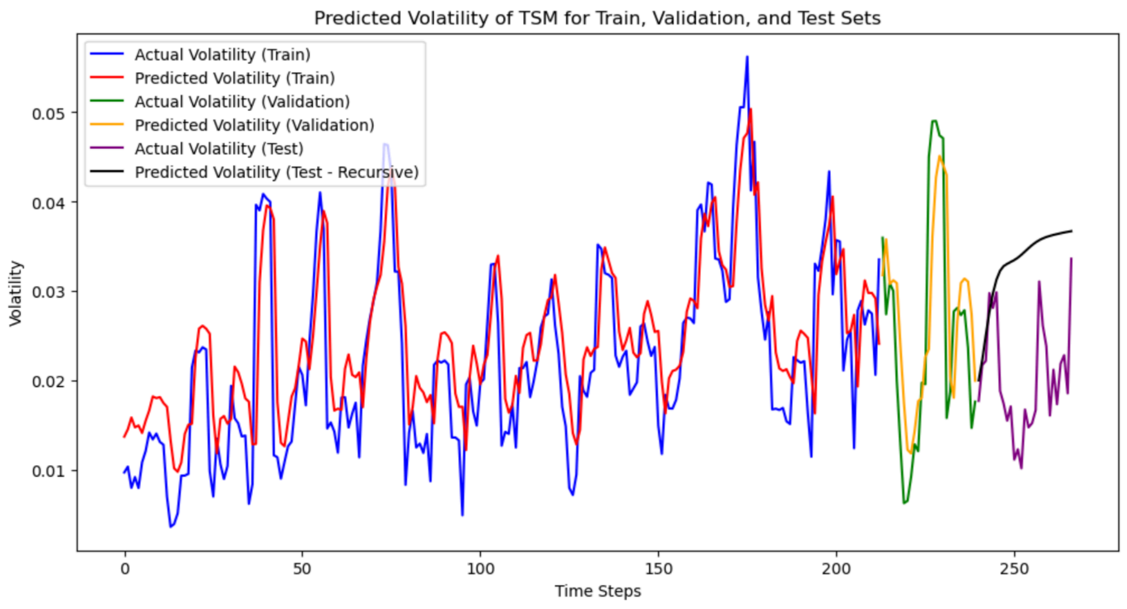


Figure 15: LSTM predictions with 'simulation' on the test set that did not perform very well

Thus, our project has yet to find a meaningful correlation between sentiment and volatility. And although all of our models are technically 'wrong', it seems that George Box would consider some more 'useful' than others. Future work will likely include analyzing more data and stocks since we were limited this time around. With enough time, money, and computing power, we could extract larger amounts of data from a diverse array of sources like Twitter, Reddit, and financial news articles. For this project, we used TextBlob to conduct sentiment analysis on our data due to its accessibility and simple API. There are, however, better resources for natural language processing like FinBert, which is specifically tailored towards financial language. Going forward, we would like to employ these more cutting-edge tools that could provide better, more detailed insights from the raw textual data. Finally, we also aim to further develop our Streamlit dashboard since it only displays numerical data so far. To make it more illustrative and help our user better understand our project, we can include figures like graphs and heat maps that visualize our data.

11. Roles

Alex: Led the development of our initial model, collected Reddit and Google News data and created LSTM models.

Andrew: Led the development of the Linear Regression and Feed Forward Neural Network models, and figured out how to use TextBlob to conduct sentiment analysis on our textual data.

Brian: Led the development of our mixed/ensemble model and wrote the script for our Streamlit program.