Introduction/Background:

Financial analysis using machine learning has long been one of the most popular applications of artificial intelligence, and for good reason - with the recent surge of interest in ML techniques, data analysis has become much more readily available and efficient to compute. Machine learning empowers analysists to extract useful insights from data, enhancing risk management and stock prediction. One of the most commonly used ML algorithms for stock price analysis is linear regression. Its popularity stems from its ability to handle continuous stock price data, along with its simplicity and interpretability. In addition, sentiment analysis also plays a critical role in financial forecasting. Studies have consistently demonstrated a strong correlation between public sentiment and the market. By analyzing financial news and social media, researchers can discover new patterns that can be used alonside stock data to create a fuller picture. By leveraging both numerical and human-based insights, these ensemble models provide a more comprehesive view of the factors influecning stock prices leading to the possibility of greater predictive power. Our goal is to use this perspective by performing stock analysis by utilizing multiple ML methodologies functions to predict the price of a stock, combining sentiment and stock prices. For our research, we utilize several key datasets. These include historical stock ticker data, capturing open and close prices for indices like the S&P 500, major tech stocks, and U.S. stocks and ETFs. Additionally, we have a database of stock-related tweets, which include both labeled sentiment scores and raw, unlabeled data, providing a rich source of information for our sentiment analysis efforts. References to these potential datasets are provided below:

- https://www.kaggle.com/datasets/borismarjanovic/price-volume-data-for-all-us-stocks-etfs
- https://www.kaggle.com/datasets/alfredkondoro/major-tech-stocks-time-series-2019-2024 https://www.kaggle.com/datasets/camnugent/sandp500
- https://www.kaggle.com/datasets/equinxx/stock-tweets-for-sentiment-analysis-and-prediction
- https://www.kaggle.com/datasets/yash612/stockmarket-sentiment-dataset https://www.kaggle.com/code/prathamsoneja/stock-price-prediction-and-sentiment-analysis

Problem Definition:

Using ML to intelligently cluster, identify, and analyze financial data is a very popular undertaking to generate a profit. However, due to the sheer amount of data, different algorithms, and complex implementations that can be employed to perform this analysis. One common occurrence that we observed was that most stock prediction engines employ one main method for analyzing data. In doing so, there was the potential for single methodology bias, which may miss certain analysis that other methodologies may excel in. In our stock prediction engine, we attempt to instead use multiple methodologies on the same dataset in order to generate a more holistic idea of how a stock may perform by incorporating sentiment analysis into our linear regression algorithm [5].

Methods, Results, and Visualizations:

Contains
✅1+ Data Preprocessing Method
✅3+ Algorithms/Models
✅CS 4641: Supervised or Unsupervised Learning Method Implemented
✅Visualizations

✅Quantitative Metrics
✅Analysis of 3+ Algorithms/Models

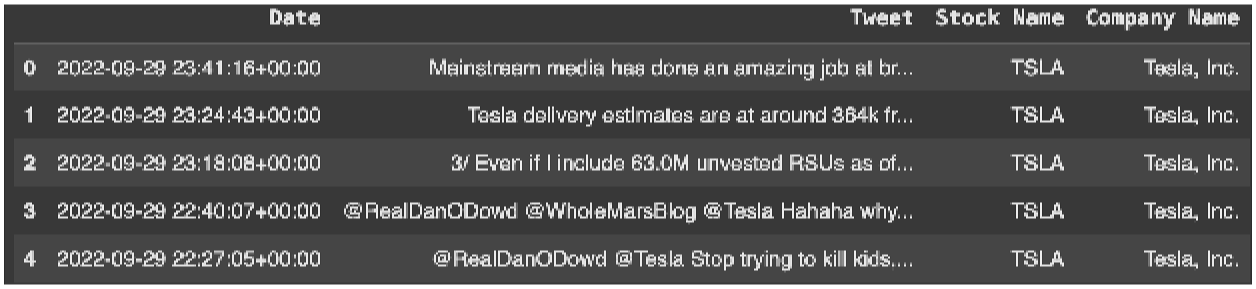
Data Pre-Processing Method

✅1 Data Preprocessing Method
✅Visualizations

For this project, we encountered a challenge in finding a comprehensive dataset that included both sentiment annotations and specific stock tickers. To address this, we decided to label the sentiment of tweets ourselves using a supervised machine learning approach. We utilized a labeled sentiment dataset of stock-related tweets to train two models: Logistic Regression and Multinomial Naive Bayes. These models were trained on preprocessed text data using TF-IDF vectorization, which transforms the text into numerical features. The preprocessing involved removing noise such as URLs, mentions, and hashtags, converting text to lowercase, and filtering out stopwords using the NLTK library.

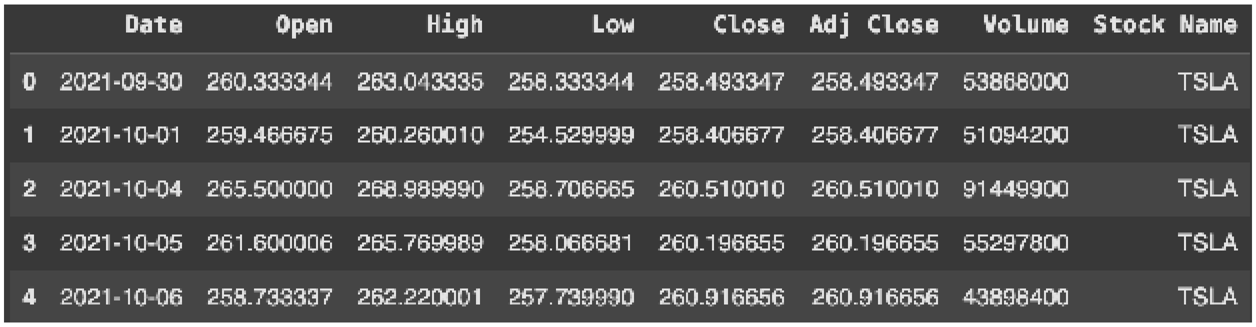
After training, we applied the best-performing model to the unlabeled dataset, predicting the sentiment of each tweet (positive or negative). Since multiple tweets could be associated with the same stock on the same day, we aggregated the sentiment scores by taking the average sentiment for each date. This allowed us to capture the general sentiment trend for each stock on a given day. For example, a sentiment score of 0.98 indicates strong positive sentiment, while a score closer to 0.76 might suggest mixed or neutral sentiment. We then merged the sentiment scores with the corresponding stock data (including opening prices and stock tickers) into a single DataFrame, which served as the input for the next phase of the analysis

Unlabeled TSLA tweets



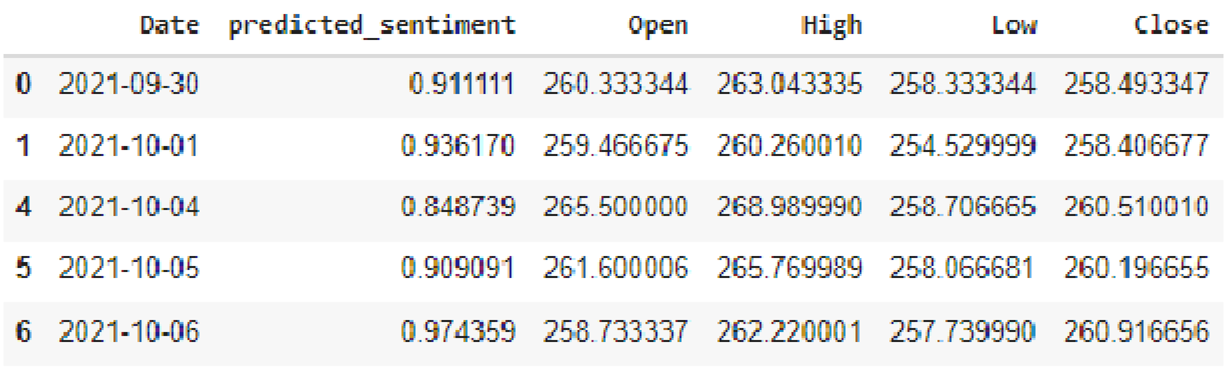
	Date	Tweet	Stock Name	Company Name
0	2022-09-29 23:41:16+00:00	Mainstream media has done an amazing job at br...	TSLA	Tesla, Inc.
1	2022-09-29 23:24:43+00:00	Tesla delivery estimates are at around 364k fr...	TSLA	Tesla, Inc.
2	2022-09-29 23:18:08+00:00	3/ Even if I include 63.0M unvested RSUs as of...	TSLA	Tesla, Inc.
3	2022-09-29 22:40:07+00:00	@RealDanODowd @WholeMarsBlog @Tesla Hahaha why...	TSLA	Tesla, Inc.
4	2022-09-29 22:27:05+00:00	@RealDanODowd @Tesla Stop trying to kill kids,...	TSLA	Tesla, Inc.

Ticker finance Data



	Date	Open	High	Low	Close	Adj Close	Volume	Stock Name
0	2021-09-30	260.333344	263.043335	258.333344	258.493347	258.493347	53868000	TSLA
1	2021-10-01	259.466675	260.260010	254.529999	258.406677	258.406677	51094200	TSLA
2	2021-10-04	265.500000	268.989990	258.706665	260.510010	260.510010	91449900	TSLA
3	2021-10-05	261.600006	265.769989	258.066681	260.196655	260.196655	55297800	TSLA
4	2021-10-06	258.733337	262.220001	257.739990	260.916656	260.916656	43898400	TSLA

Combined preprocessed dataframe



	Date	predicted_sentiment	Open	High	Low	Close
0	2021-09-30	0.911111	260.333344	263.043335	258.333344	258.493347
1	2021-10-01	0.936170	259.466675	260.260010	254.529999	258.406677
4	2021-10-04	0.848739	265.500000	268.989990	258.706665	260.510010
5	2021-10-05	0.909091	261.600006	265.769989	258.066681	260.196655
6	2021-10-06	0.974359	258.733337	262.220001	257.739990	260.916656

Core Algorithms / Models Utilized, and Why:

✅Analysis of 3 Algorithms/Models (1st time)
In this project, we employed three key machine learning algorithms: Logistic Regression, Multinomial Naive Bayes, and Linear Regression, each chosen for its suitability to the specific task at hand. For sentiment analysis, we first utilized Logistic Regression, a widely used algorithm for binary classification. This model was selected due to its interpretability and efficiency. Logistic Regression allowed us to clearly classify tweets as positive or negative based on the probabilities it assigned, making it easy to understand the modelâ€™s decision-making process. Additionally, the model performed well with our text data after being vectorized using TF-IDF, which transformed the raw tweets into numerical features. The linear nature of Logistic Regression also allowed us to assess the importance of individual features, making it a strong baseline model for our sentiment classification task.

To complement our analysis, we also trained a Multinomial Naive Bayes classifier, which is well-suited for text classification, particularly when working with high-dimensional, sparse data like TF-IDF features. This algorithm was chosen because of its computational efficiency and robustness with text data. Multinomial Naive Bayes leverages the probabilistic distribution of word counts, making it ideal for handling the categorical features derived from our tweet text. Its efficiency and performance in text classification tasks made it a valuable model to use alongside Logistic Regression. By training and evaluating both models, we ensured that our sentiment labeling was robust and reliable. The best-performing model was then used to label the sentiment of the unlabeled tweets, allowing us to create a dataset annotated with positive or negative sentiment scores.

Following the sentiment analysis, we integrated the labeled sentiment data with stock market data to investigate potential correlations between public sentiment and stock price movements. For this task, we used Linear Regression, a straightforward yet powerful algorithm for predicting continuous numerical values. [6] Linear Regression was chosen for its simplicity and interpretability [4], enabling us to easily quantify the relationship between independent variables (the opening stock price and average sentiment score) and the dependent variable (the closing stock price). This model helped us analyze how variations in public sentiment might influence stock performance, providing a baseline understanding of the potential impact of social media sentiment on the stock market. By combining these models, we were able to label tweet sentiments, merge them with stock data, and effectively explore the relationship between public sentiment and stock prices.

Model 1: Linear Regression

✅1 Algorithms/Models
✅Visualizations
✅Quantitative Metrics

✅Analysis of 1 Algorithms/Models

As mentioned, we utilized Linear Regression to perform our base fundamental stock valuation. Since linear regression lends itself quite nicely to numerical time-series data, we opted to perform this first to get a better idea of the general movement of a stock-for instance, TSLA-before introducing any further complexity with sentiment analysis. To perform this, we separated our stock dataframe into an X and Y split, based on the open and close values respectively. These were then split into training and testing iterations:

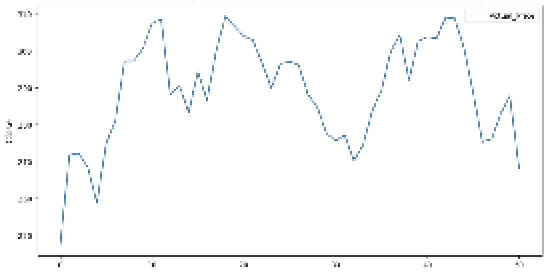
```
 X = tesla_data[['Open']].values
Y = tesla_data['Close'].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.2, shuffle=False)
```

From there, we set a simple LinearRegression() model based on these splits, fit our model, and got the following metrics:

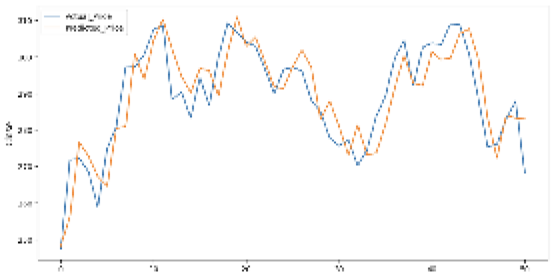
```
 Mean Absolute Error (MAE): 5.99
Mean Squared Error (MSE): 53.44
Root Mean Squared Error (RMSE): 7.31
R-squared (RÂ²): 0.73
```

Our overall observations were that there was moderate accuracy, with some prediction errors in high volatility scenarios. Our visualization also showed a reasonable alignment between our y_hat and y_predicted, though there was still room for improvement.

TSLA closing price over ~50 Days



TSLA closing price over ~50 Days w/ Predicted closing price from Linear Regression model



`Model 2: Logistic Regression`

✅1 Algorithms/Models
✅Visualizations
✅Quantitative Metrics
✅Analysis of 1 Algorithms/Models

✅CS 4641: Supervised Learning Method Implemented

Our next step in our multi-modal pipeline was to perform our first layer of sentiment classification. For this, we opted for logistic regression, essentially taking in various stock-related tweets and classifying them as -1 for negative sentiment and +1 for positive. To do this, however, we needed to find a way to make our character-heavy tweets parsable in our actual regression model. Below is a snippet from preprocess_text(), a function that essentially cleaned and preprocessed text to make it easier to assign sentiments:

```
 def preprocess_text(text):
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)  # Remove URLs
    text = re.sub(r'@\w+|\#', '', text)  # Remove mentions and hashtags
    text = text.lower()  # Convert to lowercase
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word.isalpha()]  # Remove non-alphabetic tokens
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)

# Apply preprocessing. Here, we added two new columns related to our cleaned text.
labeled_data['clean_text'] = labeled_data['Text'].apply(preprocess_text)
unlabeled_data['clean_tweet'] = unlabeled_data['Tweet'].apply(preprocess_text)
```

We also utilized the TF-DF Vectorizer library to make our text data even easier to fit in our logistic regression model. It essentially transformed our existing text data into feature vectors (numerical representations) by assigning 'scores' to each term in the document, allowing the model to avoid extraneous words like 'the', 'like', and 'is' and assigning higher importance to more unique, and thus meaningful, terms.

Our precision, recall, and F1-scores highlighted good performance for positive sentiment class but slightly lower performance for negative sentiment. Output for LogReg metric tests:

```
 Accuracy: 0.7739430543572045
              precision    recall  f1-score   support

          -1       0.81      0.51      0.62       427
           1       0.76      0.93      0.84       732

    accuracy                           0.77      1159
   macro avg       0.79      0.72      0.73      1159
weighted avg       0.78      0.77      0.76      1159
```

Note the much better recall and f1 scores for positive sentiment (+1). For overall market sentiment, this is still not a bad result, but our next model in the pipeline aimed to improve the performance after hybridizing the reuslts of the two.

We visualized the output of out predicted sentiment by concatenating it with the assosciated tweet as seen below.

```
      Date    Tweet   Stock Name  predicted_sentiment
0    2022-09-29 23:41:16+00:00   Mainstream media has done an amazing job at br...   TSLA    1
1    2022-09-29 23:24:43+00:00   Tesla delivery estimates are at around 364k fr...   TSLA    1
2    2022-09-29 23:18:08+00:00   3/ Even if I include 63.0M unvested RSUs as of...   TSLA    1
3    2022-09-29 22:40:07+00:00   @RealDanODowd @WholeMarsBlog @Tesla Hahaha why...   TSLA    1
4    2022-09-29 22:27:05+00:00   @RealDanODowd @Tesla Stop trying to kill kids,...   TSLA    1
```

Model 3: Multinomial Naive Bayes

✅1 Algorithms/Models
✅Visualizations
✅Quantitative Metrics
✅Analysis of 1 Algorithms/Models
MNB is a classification algorithm that is traditionally used for text data through the application of Bayes' theorem to predict the sentiment of a tweet. Similarly to our logistic regression model, we train on the vectorized text data (X_train) to classify the sentiment into either +1 or -1. The Bayes' theorem comes into play with the idea that the probability of a class C for a document D is given by multiplying the prior of class C - P(C) by the total product of the likelihoods of each individual WORD in the document also appearing in that class. Eventually, P(C | D) is computed for each class, both +1 and -1, and assigns the tweet the class with the higher conditional probability.

Here are our metrics from the Naive Bayes classifier:

```
 Accuracy: 0.7454702329594478
              precision    recall  f1-score   support

          -1       0.85      0.37      0.52       427
           1       0.72      0.96      0.83       732

    accuracy                           0.75      1159
   macro avg       0.79      0.67      0.67      1159
weighted avg       0.77      0.75      0.71      1159
```

Overall, we had a slightly better performance than logistic regression for sentiment prediction. It also handled skewed distributions pretty well, and benefited from simpler feature relationships.

HYBRIDIZATION OF MODELS - Multimodal Linear Regression Layer

By combining our numerical features (Open, High, Low, etc) with predicted sentiment, we were able to come up with a much more holistic forecast for the closing price of our stock of interest. In order to accomplish this, we aggregated our existing dataframes together, placing yfinance data along with our earlier sentiment-related findings together in one final merged dataframe:

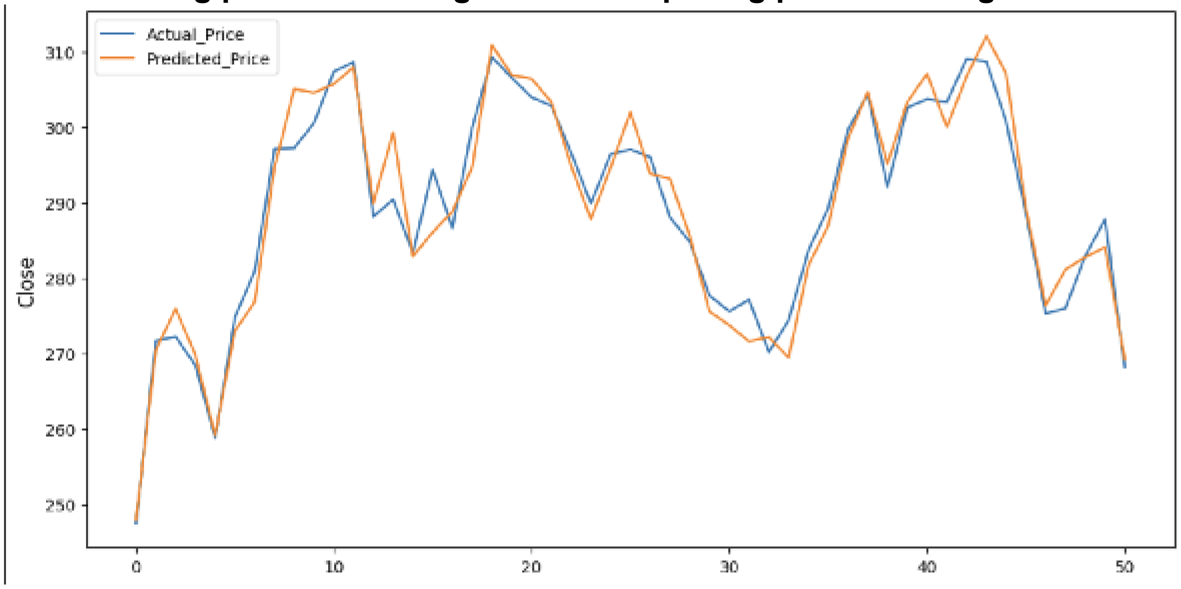
```
 merged_data = pd.merge(stock_data, yfinance_tsladf, on='Date', how='left')
merged_data.dropna(subset=['Open'], inplace=True)
merged_data.head()
```

```
     Date     predicted_sentiment     Open        High          Low           Close
0    2021-09-30  0.911111    260.333344  263.043335  258.333344  258.493347
1    2021-10-01  0.936170    259.466675  260.260010  254.529999  258.406677
4    2021-10-04  0.848739    265.500000  268.989990  258.706665  260.510010
5    2021-10-05  0.909091    261.600006  265.769989  258.066681  260.196655
6    2021-10-06  0.974359    258.733337  262.220001  257.739990  260.916656
```

From there, we were able to fit one final LinearRegression() model, using [['Open', 'predicted_sentiment', 'High', 'Low']] as our input features and ['Close'] as our output. Upon training, fitting, and testing our prediction on this final layer, we received the following metrics:

```
 Mean Absolute Error (MAE): 2.68
Mean Squared Error (MSE): 11.63
Root Mean Squared Error (RMSE): 3.41
R-squared (RÂ²): 0.94
```

Overlayed onto the actual data, the model performs as follows.
TSLA Closing price Linear Regression w/ Opening price & average sentiment



Discussion:

Initially, starting with a plain linear regression to predict the closing price of the TSLA stock at the close of the day based off of the initial opening price and previous days results showed some promise and had baseline results. In order to improve tracking and prediction to the stock, we wanted to integrate real world sentiment into the analysis of the stock. Adding this extra dimension of aggregating human input into the predictor we thought would assist the regression into making a more accurate guess. In order to accomplish this, we generalized a dataset containing labeled sentiment on stock events from users on Twitter. From this model we could then create our own data with regards to specific TSLA sentiment from tweets. This sentiment and timestamps we then linked to the previous inputs into our model and retrained the model. Proving our hypothesis correct, both numerically via our metrics and graphically, we can see that adding this new dimension to our model increased the tracking and predicting capability and reduced the MSE from 53 to 11. Additionally, we found that our MAE decreased by a factor of two, and our final R^2 value - which measures how well the inputs explain the variance in our outputs - was 0.94, which is considered a very great prediction without fear of overfitting to training data. We take this as a successful addition to our model and overall with regards to the graph and the metrics, we infer that our model would perform well. While there are places where the results are lacking, we have designed the process in such a way to be modular and benifit from future additions, as expanded upon in the next steps.

Next Steps:

As we look to build upon this project, there are several promising directions for further development and improvement. One of the key enhancements we plan to implement is the use of pre-trained word embeddings such as Word2Vec, GloVe, or BERT embeddings to improve the quality of sentiment labeling. While our current approach relies on traditional machine learning models like Logistic Regression and Multinomial Naive Bayes, these models use TF-IDF features that treat words as independent entities. This can sometimes lead to a loss of semantic context. By utilizing embeddings like Word2Vec, we can capture the semantic meaning and relationships between words in the tweets, allowing us to better understand context and sentiment. For instance, Word2Vec can help recognize that words like 'rise,' 'soar,' and 'increase' are similar in sentiment even if they don't appear frequently in the training data. This context-aware representation can significantly enhance the performance of our sentiment analysis, especially when dealing with complex financial jargon or nuanced language used in stock-related tweets.

Additionally, expanding our analysis to cover more stock tickers is a logical next step. Currently, our focus was limited to a specific stock (e.g., TSLA), but the same methodology can be applied to other stocks in different sectors. By incorporating a wider variety of tickers, we can evaluate the generalizability of our sentiment analysis model across different companies and industries. We can also compare sentiment trends across various stocks to identify sector-wide patterns or anomalies in sentiment that may indicate broader market movements.

Another area of improvement is to explore more sophisticated models for time-series analysis. While we used Linear Regression as a baseline model for predicting stock prices, there are more advanced models like LSTM (Long Short-Term Memory) networks or Prophet by Facebook that are specifically designed for time-series forecasting. These models can take into account temporal dependencies and patterns in the data, potentially providing more accurate predictions of stock price movements based on historical data and sentiment trends.

Overall, in the case that this project is revisted in the future when the technologies that we mentioned are improved upon, we suggest implimenting them with our current architecture and methodology in order to expand upon our results we found here. We hypothesize that the ML techniques will only improve give the new components and will not be affected by new patterns emerging in the market as it will generalize and pickup on them.

References:

1. "Classification: Roc and AUC | machine learning | google for developers," Google, https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc (accessed Oct. 3, 2024).
2. R. Adusumilli, "DBSCAN clustering for trading," Medium, https://towardsdatascience.com/dbscan-clustering-for-trading-4c48e5ebffc8 (accessed Oct. 3, 2024).

3. B. GÃ¼lmez, "Stock price prediction with optimized deep LSTM network with Artificial Rabbits Optimization Algorithm," Expert Systems with Applications, vol. 227, no. 120346, p. 120346, May 2023, doi: https://doi.org/10.1016/j.eswa.2023.120346.
4. S. A, "Predict stock market moves with logistic regression magic!," LinkedIn, https://www.linkedin.com/pulse/predict-stock-market-moves-logistic-regression-magic-shubham-a-h7agf/ (accessed Oct. 3, 2024).
5. L. S. Parvatha, "Stock market prediction using sentiment analysis and incremental clustering approaches | IEEE conference publication | IEEE Xplore," IEEE.org, https://ieeexplore.ieee.org/document/10112768/ (accessed Oct. 4, 2024).
6. K. Senevirathne, "How Iâ€™m using machine learning to trade in the stock market," Medium, https://medium.com/analytics-vidhya/how-im-using-machine-learning-to-trade-in-the-stock-market-3ba981a2ffc2 (accessed Oct. 3, 2024).

Contribution Table

✅Contribution Table
Ethan - Writeup, analysis help, debugging code, visualizations, github page
Nathan - Project Proposal - Intro/Background edits, sentiment datasets found, Problem Definition, Methods
Tarun - Sentiment analysis datasets found, trained sentiment models, preprocessing method, Next Steps, write up edits.
Jacob - Intro/Background edits, sentiment dataset training, Methods & Models, References
Eddy - Created the logistic regression model, helped train sentiment models, write up edits made

Download Gantt Chart

✅Gantt Chart
https://github.gatech.edu/nlin47/StockPredictor_CS4641/blob/main/GanttChart.xlsx

Link to Colab

https://colab.research.google.com/drive/15QMgVPxjrw2Y9A6cdrovW7CMLkNlka8r?usp=sharing#scrollTo=JZeJayom3wPA