1. Abstract

In this project, I set out to analyze whether **traditional financial indicators or crowd-driven sentiment are more influential in driving short-term stock price movements**. With the increasing integration of social media into financial markets, particularly platforms like Reddit where users actively discuss stock trends and make speculative predictions, I was interested in understanding whether this social sentiment offers any predictive power when compared to conventional metrics such as Earnings Per Share (EPS), Price-to-Earnings (P/E) Ratio, and Market Capitalization. To investigate this, I created a dataset that merges historical stock prices with Reddit post metadata, including sentiment scores derived from post titles using basic natural language processing techniques.

I computed daily stock price percentage changes and evaluated their relationship with the selected financial and sentiment variables using both correlation analysis and linear regression models. My results indicated that Net Sentiment had the highest correlation with stock price change (0.0754), while P/E Ratio (0.0430), EPS (-0.0219), and Market Capitalization (-0.0202) showed weaker or even negative correlations. Furthermore, regression models confirmed that EPS and P/E Ratio alone explained virtually none of the variance in stock price change ($R^2 \approx 0$), whereas including sentiment marginally improved model performance. These findings suggest that although none of the examined features strongly predict short-term stock returns on their own, **social sentiment is slightly more aligned with short-term price movements than traditional financial metrics**. This supports the growing idea that retail investor behavior and online discourse can influence market dynamics, even if modestly, and may complement traditional analysis in future stock evaluation frameworks.

2. Motivation

The motivation behind this project stems from the growing influence of retail investors and online platforms particularly Reddit on stock market behavior. Events like the GameStop and AMC short squeezes have highlighted how crowd sentiment, often driven by online discussion rather than company fundamentals, can lead to significant and rapid stock price movements. This raises an important question: **Can sentiment extracted from online platforms help predict short-term stock movements better than traditional financial metrics?**

Traditionally, investors have relied on financial indicators such as Earnings Per Share (EPS), Price-to-Earnings (P/E) Ratio, and Market Capitalization to evaluate a company's performance and forecast stock trends. However, these metrics often reflect long-term valuation rather than immediate market reaction. In contrast, sentiment data derived from investor discussions may capture real-time market psychology and short-term momentum. Given the rise in algorithmic trading and the availability of alternative data, it's increasingly relevant to assess whether sentiment signals can meaningfully enhance or even outperform conventional financial analysis.

Hypothesis:

Stocks with more positive sentiment from Reddit posts will exhibit short-term price increases, while those with negative sentiment will experience declines. Additionally, sentiment will have stronger predictive power for short-term price movement than traditional financial indicators such as EPS, P/E Ratio, and Market Cap.

This hypothesis forms the foundation of my analysis, where I examine and compare the predictive relationships between sentiment and financial metrics with daily stock price changes. By doing so, I aim to understand the respective roles of fundamentals and investor psychology in short-term stock behavior.

3. Datasets Used

To conduct this project, I compiled and integrated three key datasets from publicly available online sources. These datasets capture financial metrics, historical stock prices, and investor sentiment from social media platforms. After cleaning and preprocessing, they were merged into a unified dataset with 1,159 entries, each representing a stock on a given date with associated financial and sentiment information.

o Historical Stock Prices

- **Source**: Twelve Data API
- **Content**: Daily open, high, low, close, and volume data for multiple tickers over a five-year period
- **Usage**: Used to compute the daily stock price percentage change, which served as the primary target variable for analysis.
- **Preprocessing**: Converted date fields to datetime, ensured all prices were numeric, and calculated daily returns using .pct_change() per ticker.

o Company Financials

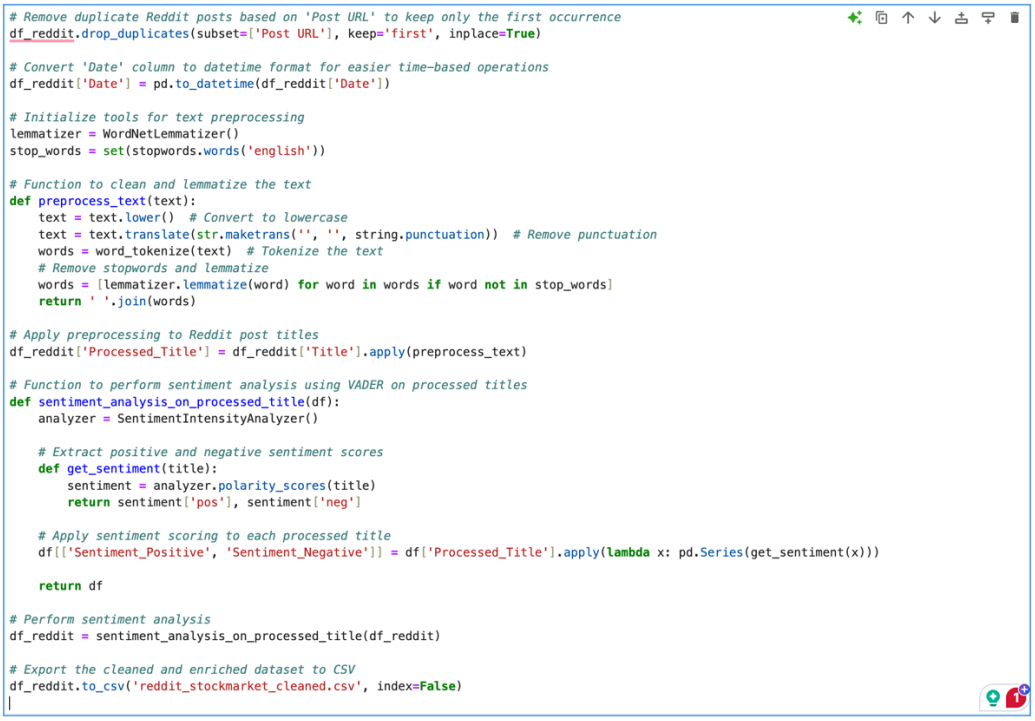
- **Source**: Yahoo Finance APIs
- **Content**: Company-level metrics such as:
  - Price-to-Earnings (P/E) Ratio
  - Earnings Per Share (EPS)
  - Market Capitalization
  - Dividend Yield
  - Net Income, Gross Profit, and Total Revenue
- **Usage**: These indicators were used to test the impact of traditional financial fundamentals on stock performance.
- **Preprocessing**: Ensured proper scaling (especially for Market Cap), removed duplicate entries, and cross-referenced tickers for consistency with stock price data.

- **Reddit Post Sentiment Data**

- **Source**: **Reddit Post** using web scrapping.
- **Content**: Post titles, URLs, timestamps, upvote score, number of comments, and sentiment polarity scores (positive and negative)
- **Usage**: Sentiment scores were computed using a sentiment analyzer (e.g., VADER). A new column Net_Sentiment was created by subtracting negative sentiment from positive sentiment.
- **Preprocessing**: Titles were cleaned using NLP techniques (lowercasing, punctuation removal, stop word filtering). Posts were matched with the corresponding stock ticker and trading date.

4. Data Cleaning

Reddit Dataset Cleaning



```python
# Remove duplicate Reddit posts based on 'Post URL' to keep only the first occurrence
df_reddit.drop_duplicates(subset=['Post URL'], keep='first', inplace=True)

# Convert 'Date' column to datetime format for easier time-based operations
df_reddit['Date'] = pd.to_datetime(df_reddit['Date'])

# Initialize tools for text preprocessing
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# Function to clean and lemmatize the text
def preprocess_text(text):
    text = text.lower()  # Convert to lowercase
    text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation
    words = word_tokenize(text)  # Tokenize the text
    # Remove stopwords and lemmatize
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
    return ' '.join(words)

# Apply preprocessing to Reddit post titles
df_reddit['Processed_Title'] = df_reddit['Title'].apply(preprocess_text)

# Function to perform sentiment analysis using VADER on processed titles
def sentiment_analysis_on_processed_title(df):
    analyzer = SentimentIntensityAnalyzer()

    # Extract positive and negative sentiment scores
    def get_sentiment(title):
        sentiment = analyzer.polarity_scores(title)
        return sentiment['pos'], sentiment['neg']

    # Apply sentiment scoring to each processed title
    df[['Sentiment_Positive', 'Sentiment_Negative']] = df['Processed_Title'].apply(lambda x: pd.Series(get_sentiment(x)))

    return df

# Perform sentiment analysis
df_reddit = sentiment_analysis_on_processed_title(df_reddit)

# Export the cleaned and enriched dataset to CSV
df_reddit.to_csv('reddit_stockmarket_cleaned.csv', index=False)
```

The code begins by removing duplicate Reddit posts using the Post URL column, keeping only the first occurrence to ensure each post is unique. It then converts the Date column to datetime format, which is essential for time-based analysis. A WordNetLemmatizer and a set of English stop words are initialized to prepare for text preprocessing. The preprocess_text function is defined to convert text to lowercase, remove punctuation, tokenize it, eliminate stop words, and lemmatize each word, ultimately returning a cleaned version of the title. This function is applied to the Title column to create a new column called Processed_Title. Next, the

sentiment_analysis_on_processed_title function is defined, using VADER's SentimentIntensityAnalyzer to compute sentiment scores. For each processed title, it extracts the positive and negative sentiment scores and stores them in two new columns: Sentiment_Positive and Sentiment_Negative. The cleaned and enriched dataset is then saved as a CSV file named 'reddit_stockmarket_cleaned.csv' for future use.

Cleaning Financial Metrics and Company Income

```
# Define a function to preprocess financial data by handling missing values, type conversion, filtering, and normalization.
def preprocess_financial_data(df):
    # Fill missing values in financial columns with their respective column means.
    df['P/E Ratio'] = df['P/E Ratio'].fillna(df['P/E Ratio'].mean())
    df['EPS'] = df['EPS'].fillna(df['EPS'].mean())
    df['Market Cap'] = df['Market Cap'].fillna(df['Market Cap'].mean())
    df['Dividend Yield'] = df['Dividend Yield'].fillna(df['Dividend Yield'].mean())

    # Convert the columns to numeric type, coercing any invalid entries to NaN.
    df['P/E Ratio'] = pd.to_numeric(df['P/E Ratio'], errors='coerce')
    df['EPS'] = pd.to_numeric(df['EPS'], errors='coerce')
    df['Market Cap'] = pd.to_numeric(df['Market Cap'], errors='coerce')
    df['Dividend Yield'] = pd.to_numeric(df['Dividend Yield'], errors='coerce')

    # Filter the DataFrame to include only records with reasonable P/E ratios (between 0 and 100).
    df = df[(df['P/E Ratio'] > 0) & (df['P/E Ratio'] < 100)]

    # Normalize selected financial columns using Min-Max scaling to bring values into the 0-1 range.
    scaler = MinMaxScaler()
    df[['P/E Ratio', 'EPS', 'Market Cap']] = scaler.fit_transform(df[['P/E Ratio', 'EPS', 'Market Cap']])

    # Return the cleaned and normalized DataFrame.
    return df

# Apply the preprocessing function to the financials DataFrame.
df_financials_cleaned = preprocess_financial_data(df_financials)

# Save the cleaned financial data to a new CSV file.
df_financials.to_csv('financial_data_cleaned.csv', index=False)
```

The preprocess_financial_data function is designed to clean and prepare financial data for analysis. It begins by handling missing values in key financial columns **P/E Ratio**, **EPS**, **Market Cap**, and **Dividend Yield**—by replacing any null values with the mean of the respective columns. This ensures completeness and avoids errors during numerical computations. Next, the function converts these columns to numeric data types using pd.to_numeric, coercing any non-numeric values to NaN to maintain data consistency.

To further refine the dataset, the function filters out records with **extreme or invalid P/E Ratios**, specifically keeping only those between 0 and 100, which represents a realistic range for most companies and removes potential outliers. After filtering, the function applies **Min-Max Scaling** to normalize the values of **P/E Ratio**, **EPS**, and **Market Cap** into a 0–1 range. This scaling step is important for ensuring that features with large numeric differences do not disproportionately influence machine learning models or distance-based algorithms.

Finally, the cleaned and normalized DataFrame is returned and stored in a new variable. The result is a well-prepared dataset, free of missing values and outliers, with

consistent numeric formatting and scaled features—ready for correlation analysis or regression modeling.

5. Data combination

```python
# Standardize column names to ensure consistency across all datasets.
df_all_stock_data.rename(columns={'datetime': 'Date'}, inplace=True)
df_financials.rename(columns={'Date': 'Date'}, inplace=True)
df_reddit.rename(columns={'Date': 'Date'}, inplace=True)

# Convert the 'Date' column in all datasets to datetime format.
df_all_stock_data['Date'] = pd.to_datetime(df_all_stock_data['Date'])
df_financials['Date'] = pd.to_datetime(df_financials['Date'])
df_reddit['Date'] = pd.to_datetime(df_reddit['Date'])

# Extract only the date component (drop time) to ensure accurate merging.
df_all_stock_data['Date'] = df_all_stock_data['Date'].dt.date
df_financials['Date'] = df_financials['Date'].dt.date
df_reddit['Date'] = df_reddit['Date'].dt.date

# Merge stock price data with financial data on 'Date' and 'Ticker' using inner join to retain only matching records.
df_combined = pd.merge(df_all_stock_data, df_financials, on=['Date', 'Ticker'], how='inner')

# Merge the resulting DataFrame with Reddit sentiment data on 'Date' and 'Ticker' to create the final dataset.
df_final = pd.merge(df_combined, df_reddit, on=['Date', 'Ticker'], how='inner')
```

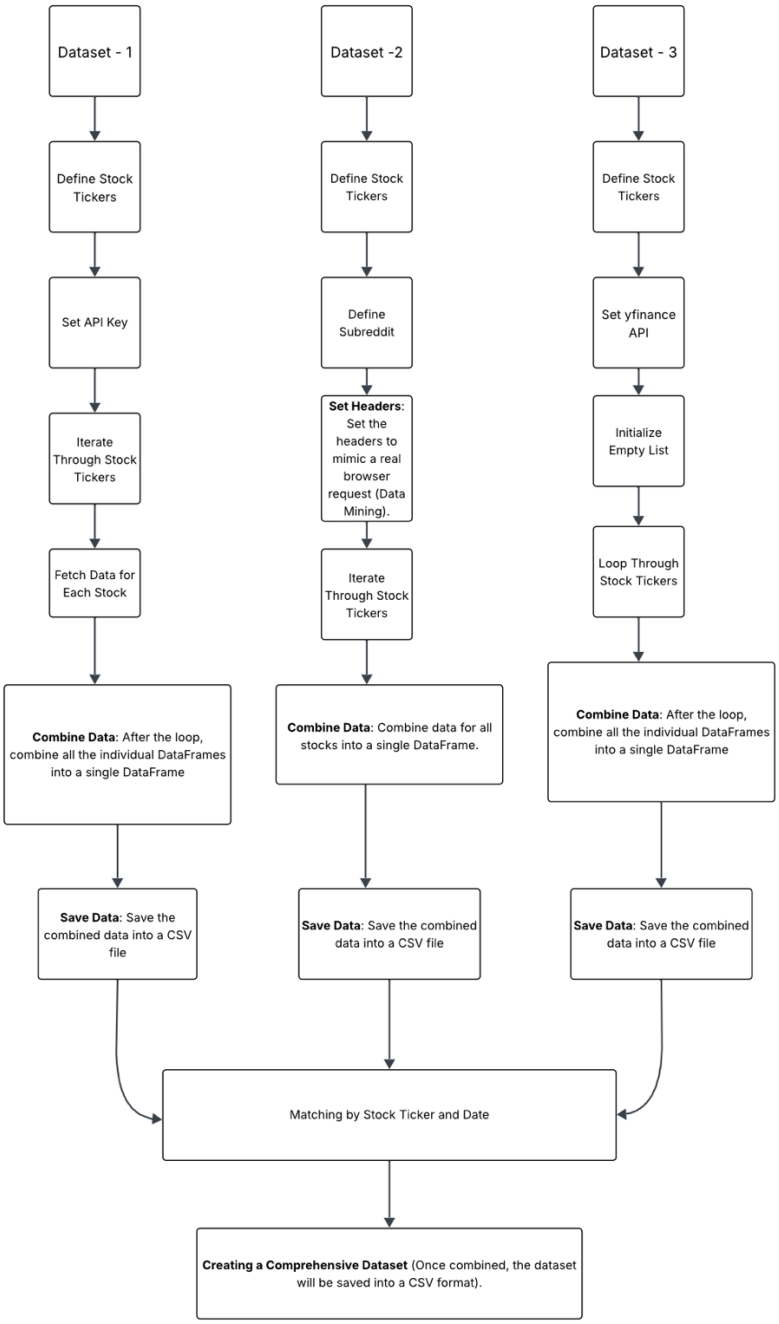
This code segment performs the final integration of all three datasets stock price data, company financial data, and Reddit sentiment data— nto a single unified dataset suitable for analysis. It begins by standardizing the column names to ensure consistency, specifically aligning the Date column across all datasets. The Date fields are then converted into datetime format and further reduced to include only the date component (excluding time) to ensure accurate matching during the merging process.

After formatting, the code performs two sequential merges using an inner join strategy. First, the stock data is merged with the financial data on the shared keys Date and Ticker, ensuring that only records common to both datasets are retained. The resulting DataFrame is then merged with the Reddit dataset on the same keys (Date and Ticker) to include sentiment features aligned with each company on each trading day. The final merged dataset, df_final, contains only those entries where stock price, financial information, and Reddit sentiment data were all simultaneously available, ensuring data completeness and consistency for subsequent modeling and analysis.

Merged Dataset Summary:

- **Total Entries**: 1,159
- **Features**: 23 columns including financial indicators, price data, Reddit metadata, and sentiment scores
- **Final Format**: One row per stock per day, including both financial and sentiment-based variables for modeling and correlation analysis.

Flowchart for Dataset Generation



6. Data derived

Combined Dataset Structure

The final dataset used for analysis contains **1,159 rows** and **23 columns**, with each row representing a single Reddit post matched with stock price data and financial metrics for a specific company on a specific day. The dataset integrates traditional stock performance indicators, financial fundamentals, and sentiment analysis derived from Reddit discussions.

Key Columns in the Dataset

Column Name	Description
Date	The trading date (string format; converted to datetime during processing)
open, high, low, close	Daily stock price statistics
volume	Trading volume for the day
Ticker	Stock ticker symbol (e.g., AAPL, TSLA)
Stock Price (Closing)	Closing stock price (may duplicate close)
P/E Ratio	Price-to-Earnings ratio, a valuation metric
EPS	Earnings Per Share, indicating profitability
Market Cap	Market capitalization of the company
Dividend Yield	Dividend yield expressed as a percentage
Company Name	Full company name
Net Income, Gross Profit, Total Revenue	Key financial performance figures
Title	Original Reddit post title
Post URL	URL to the Reddit post
Score, Comments	Post popularity metrics (upvotes, comment count)
Processed_Title	Preprocessed version of the post title for sentiment analysis
Sentiment_Positive, Sentiment_Negative	Sentiment polarity scores from text analysis

Derived Features (during processing)

Derived Feature	Description
Price_Change (added later)	Daily percentage change in closing stock price
Net_Sentiment	Computed as Sentiment_Positive - Sentiment_Negative

EDA

```
df.head()
```

	Date	open	high	low	close	volume	Ticker	Stock Price (Closing)	P/E Ratio	EPS	...	Net Income	Gross Profit	Total Revenue	Ti
0	2020-05-15	75.087502	76.9750	75.052498	76.9275	166348400	AAPL	74.876907	31.266665	6.3	...	96150003712	184102993920	395760009216	For next tri I'll $107,0 on fire.
1	2020-06-09	83.035004	86.4025	83.002502	85.9975	147712400	AAPL	83.705116	31.266665	6.3	...	96150003712	184102993920	395760009216	AA closes all-ti high anal says
2	2020-06-22	87.835000	89.8650	87.787500	89.7175	135445200	AAPL	87.325981	31.266665	6.3	...	96150003712	184102993920	395760009216	T mom AA announc endi partnersl
3	2020-08-03	108.200000	111.6375	107.892500	108.9375	308151200	AAPL	106.033646	31.266665	6.3	...	96150003712	184102993920	395760009216	Warr Buffe $35B sta in AA tripled i
4	2020-08-07	113.205000	113.6750	110.292500	111.1125	198045600	AAPL	108.345665	31.266665	6.3	...	96150003712	184102993920	395760009216	Buy AA in front 4:1 stc split. Stc sp

5 rows × 23 columns

EDA

```
df.head()
```

oss Profit	Total Revenue	Title	Post URL	Score	Comments	Processed_Title	Sentiment_Positive	Sentiment_Negative
02993920	395760009216	For my next trick, I'll set $107,000 on fire. ...	https://www.reddit.com/r/wallstreetbets/commen...	1834	935	next trick ill set 107000 fire 200 x aapl 305 ...	0.000	0.39
02993920	395760009216	AAPL closes at all-time high as analyst says c...	https://www.reddit.com/r/apple/comments/gzxdx9...	2911	216	aapl close alltime high analyst say company ' ...	0.216	0.00
02993920	395760009216	The moment AAPL announced ending partnership w...	https://www.reddit.com/r/stocks/comments/hdyke...	807	227	moment aapl announced ending partnership intc ...	0.000	0.00
02993920	395760009216	Warren Buffett's $35B stake in AAPL tripled in...	https://www.reddit.com/r/business/comments/i2x...	1236	156	warren buffett ' 35b stake aapl tripled value ...	0.231	0.00
02993920	395760009216	Buy AAPL in front of 4:1 stock split. Stock sp...	https://www.reddit.com/r/stocks/comments/i5gdx...	1652	375	buy aapl front 41 stock split stock split add ...	0.000	0.00

head () function will print the first five rows of the dataset.

7. Data Analysis

Data Summary

To understand the structure and completeness of the dataset, I used df.info(), which revealed the following:

- The dataset contains **1,159 rows** and **23 columns**.
- Each row corresponds to a stock trading day enriched with Reddit sentiment and company financial data.
- All columns are fully populated, meaning **no missing values** were present.
- Data types are efficiently distributed across:
    - **Float64**: Prices, ratios, and sentiment scores
    - **Int64**: Counts like volume, comments, and earnings
    - **Object**: Dates, ticker symbols, titles, and URLs
- The total in-memory size is approximately **208.4 KB**, making the dataset highly efficient for analysis.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1159 entries, 0 to 1158
Data columns (total 23 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Date                    1159 non-null   object
 1   open                    1159 non-null   float64
 2   high                    1159 non-null   float64
 3   low                     1159 non-null   float64
 4   close                   1159 non-null   float64
 5   volume                  1159 non-null   int64
 6   Ticker                  1159 non-null   object
 7   Stock Price (Closing)   1159 non-null   float64
 8   P/E Ratio               1159 non-null   float64
 9   EPS                     1159 non-null   float64
 10  Market Cap              1159 non-null   int64
 11  Dividend Yield          1159 non-null   float64
 12  Company Name            1159 non-null   object
 13  Net Income              1159 non-null   int64
 14  Gross Profit            1159 non-null   int64
 15  Total Revenue           1159 non-null   int64
 16  Title                   1159 non-null   object
 17  Post URL                1159 non-null   object
 18  Score                   1159 non-null   int64
 19  Comments                1159 non-null   int64
 20  Processed_Title         1159 non-null   object
 21  Sentiment_Positive      1159 non-null   float64
 22  Sentiment_Negative      1159 non-null   float64
dtypes: float64(10), int64(7), object(6)
memory usage: 208.4+ KB
```

Figure 1

```
df.isnull().sum()
```

```
Date                    0
open                    0
high                    0
low                     0
close                   0
volume                  0
Ticker                  0
Stock Price (Closing)   0
P/E Ratio               0
EPS                     0
Market Cap              0
Dividend Yield          0
Company Name            0
Net Income              0
Gross Profit            0
Total Revenue           0
Title                   0
Post URL                0
Score                   0
Comments                0
Processed_Title         0
Sentiment_Positive      0
Sentiment_Negative      0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

Figure 2

From the figure 2, the output confirms that the dataset is completely clean, with **no missing values** and **no duplicate entries**. Specifically, the df.isnull().sum() function shows that each of the 23 columns has **zero null values**, ensuring that the dataset is fully populated and requires no imputation. Additionally, df.duplicated().sum() returns **0**, indicating that there are no repeated rows in the DataFrame.

8. Data Visualization

Distribution of P/E Ratio

```
sns.histplot(df['P/E Ratio'], bins=30, kde=True)
plt.title('Distribution of P/E Ratio')
plt.xlabel('P/E Ratio')
plt.ylabel('Frequency')
plt.show()
```

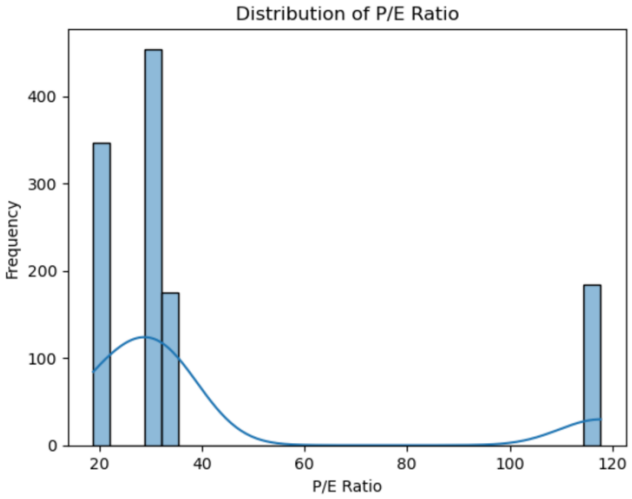


Figure 3

From figure 3, histogram illustrates the distribution of P/E Ratios across companies in the dataset. The **x-axis shows** the **P/E Ratio**, which measures how much investors are willing to pay per dollar of a company's earnings, while the **y-axis indicates** how **frequently each range occurs**. Most companies have P/E Ratios between 20 and 40, suggesting typical market valuations. However, the distribution is right-skewed, with a few companies showing extremely high P/E Ratios around 120 likely outliers or high-growth stocks. The KDE line highlights the density of values, showing a sharp peak in the lower range and a long tail.

Market Cap vs. Stock Price (Scatter Plot)

```
plt.scatter(df['Market Cap'], df['Stock Price (Closing)'])
plt.xscale('log')  # Market cap often spans large values
plt.title('Market Cap vs Stock Price')
plt.xlabel('Market Cap (log scale)')
plt.ylabel('Stock Price (Closing)')
plt.show()
```

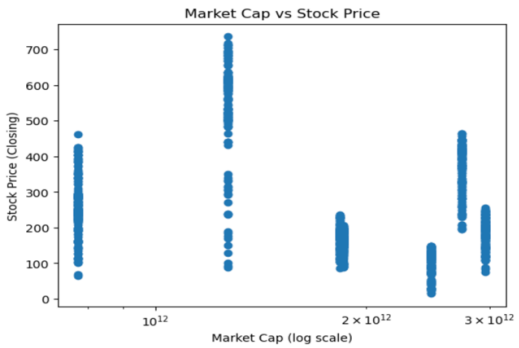


Figure 4

From Figure 4, the scatter plot illustrates the relationship between a company's Market Capitalization and its Stock Price (Closing), with the **x-axis representing Market Cap** on a logarithmic scale to accommodate the wide range of values, and the **y-axis**

showing the corresponding stock prices. Despite adjusting for scale, the plot shows no clear linear or proportional relationship between the two variables. Companies with both low and high market capitalizations exhibit a broad range of stock prices, from under $100 to over $700.

EPS vs. P/E Ratio

```
plt.scatter(df['EPS'], df['P/E Ratio'])
plt.title('EPS vs P/E Ratio')
plt.xlabel('EPS')
plt.ylabel('P/E Ratio')
plt.show()
```

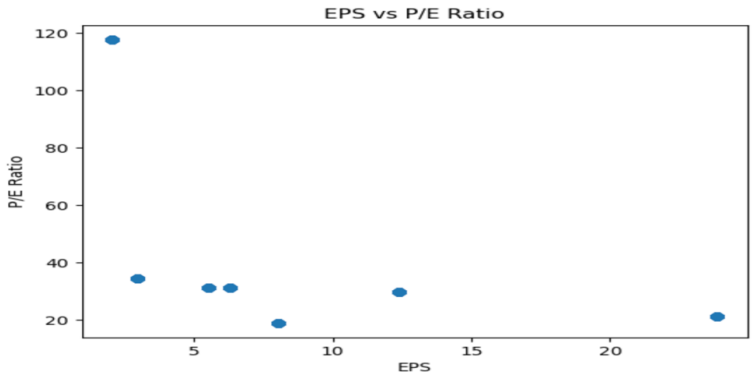


Figure 5

From figure 5, the scatter plot displays the relationship between Earnings Per Share (EPS) and the Price-to-Earnings (P/E) Ratio for various companies. The **x-axis represents EPS**, a **measure of a company's profitability**, while the **y-axis shows the corresponding P/E Ratio**, which reflects how much investors are willing to pay for one dollar of earnings. The plot reveals a scattered pattern with no clear linear trend, indicating that higher EPS does not consistently result in a higher or lower P/E Ratio. In fact, some companies with lower EPS exhibit very high P/E Ratios, possibly due to inflated market expectations, while others with higher EPS maintain moderate valuations.

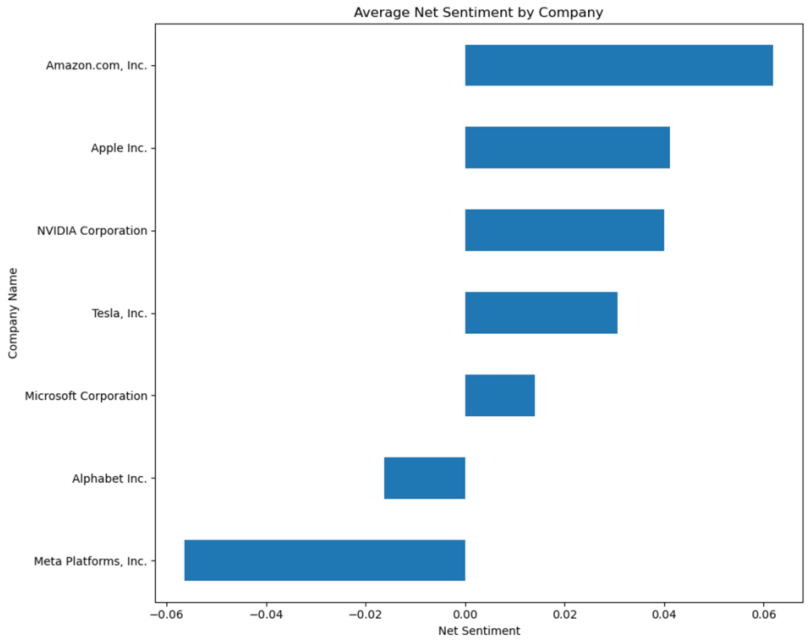


Figure 6

From figure 6, horizontal bar chart shows the **average net sentiment** on Reddit for several major tech companies, based on user discussions. The **x-axis represents** the **net sentiment score**, calculated as the difference between positive and negative sentiment scores extracted from Reddit post titles, while the **y-axis lists the corresponding company names**. Among the companies displayed, **Amazon.com, Inc.** has the highest average positive sentiment, followed closely by **Apple Inc.** and **NVIDIA Corporation**, indicating favorable public perception. On the other hand, **Meta Platforms, Inc.** shows the most negative average sentiment, followed by **Alphabet Inc.**, suggesting that Reddit discussions around these companies were more critical on average.

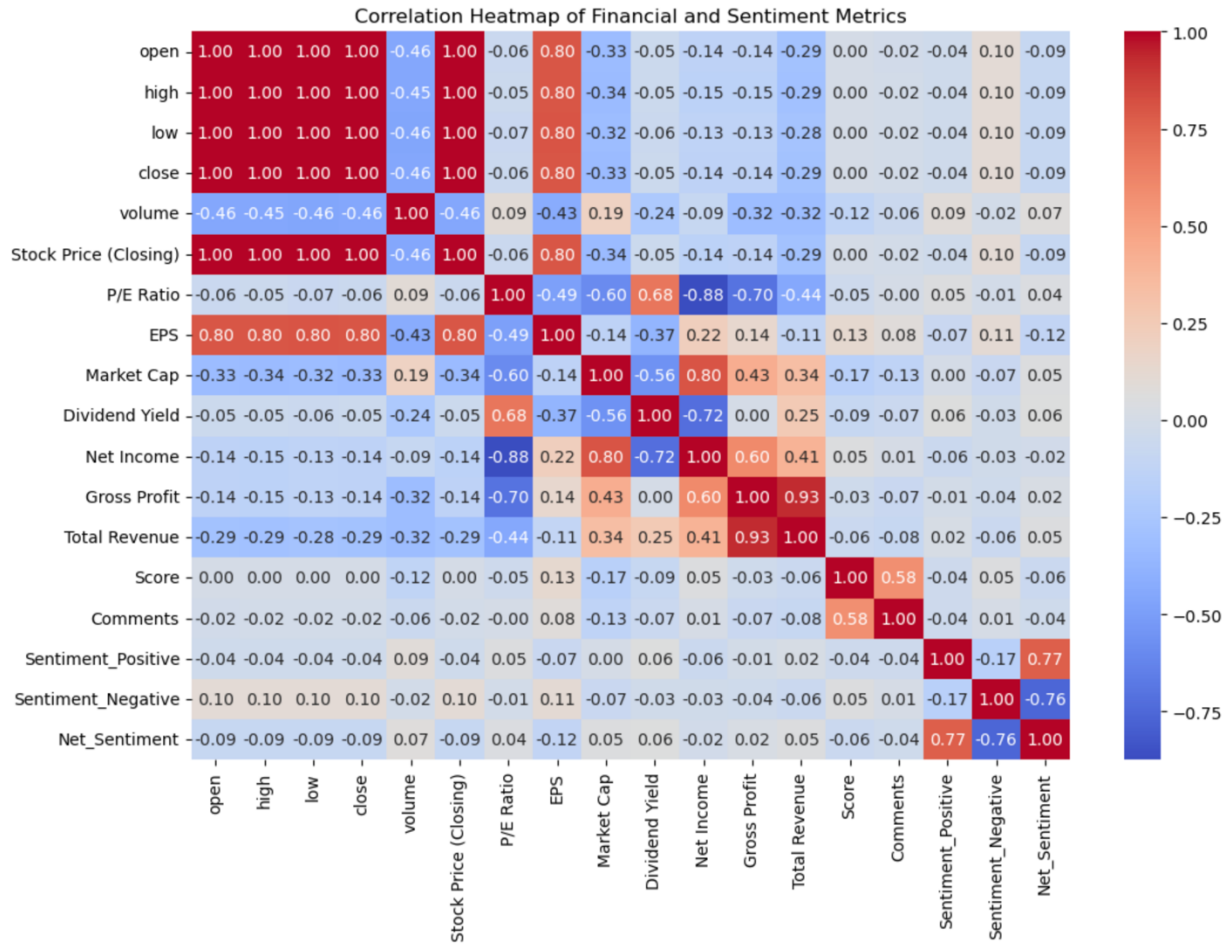


Figure 7

From figure 7, correlation heatmap provides a comprehensive view of the relationships between various financial and sentiment metrics in the dataset. Each cell shows the Pearson correlation coefficient between two variables, ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation). Strong positive correlations are indicated in dark red, while strong negative correlations appear in dark blue. For instance, **Stock Price (Closing)** shows strong positive correlations with other price-related columns like **open**, **high**, and **low**, as expected. **EPS** (Earnings Per Share) is strongly correlated with **Market Cap** (0.80) and **P/E Ratio** (0.68), suggesting that higher earnings are associated with larger companies and valuations. Additionally, financial performance indicators like **Gross Profit**, **Net Income**, and **Total Revenue** are also closely related to each other, reflecting a consistent earnings structure. On the sentiment side, **Sentiment_Positive** and **Sentiment_Negative** are strongly inversely related (−0.76), and **Net_Sentiment** is positively correlated with **Sentiment_Positive** (0.77) and negatively with **Sentiment_Negative** (−0.76), confirming that the sentiment metrics are internally consistent.

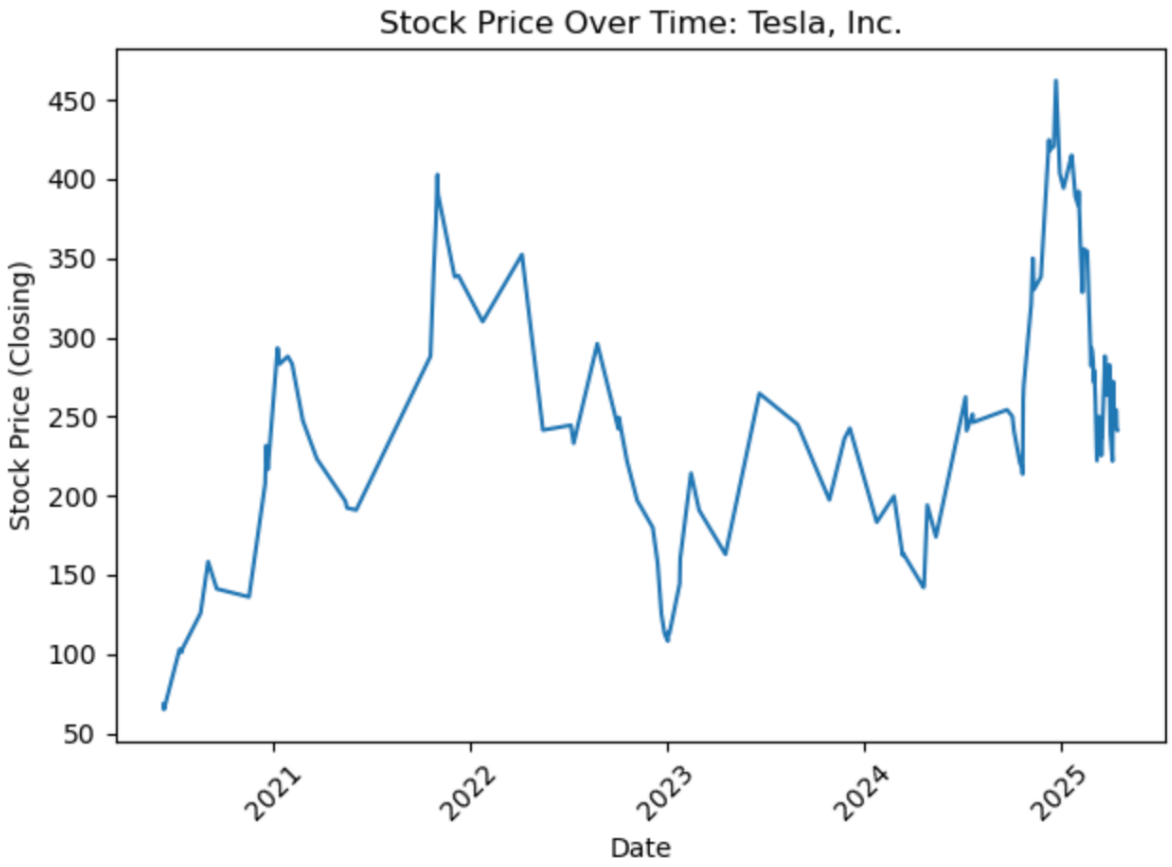


Figure 8

From figure 8, the line plot displays the stock price movement over time for the company with the highest number of records in the dataset, which is **Tesla, Inc.** The **x-axis represents** the timeline from 2020 to early 2025, while the **y-axis shows Tesla's closing stock price in dollars**. The graph reveals significant volatility, with a sharp rise in stock price through 2020 and 2021, peaking near $400–450. This is followed by multiple fluctuations and a notable dip in late 2022 to early 2023. A steep surge occurs again near the end of 2024, followed by another sharp drop heading into 2025.

9. Technical Solution

High-level overview of combining datasets

To enable a meaningful analysis of the relationship between financial indicators, social sentiment, and stock price movements, I designed a pipeline to integrate three separate datasets: historical stock price data, company financial metrics, and Reddit-based sentiment data. Each dataset was independently collected from the internet, with different formats, column names, and levels of granularity. The goal was to combine these sources into a unified dataset where each row represents a company on a specific trading day, enriched with its stock price, financial health, and public sentiment.

The solution involved several key preprocessing steps. First, I standardized column names across all datasets especially the Date and Ticker fields to ensure consistent merging keys. Next, I converted all Date columns to datetime format and stripped the time component to maintain alignment at the daily level. I then performed two **inner joins**: first between the stock data and financial metrics, and then the result with the Reddit sentiment dataset. This ensured that only records with complete data from all sources were retained. Before merging, I also applied preprocessing to the Reddit titles using tokenization, stopword removal, and lemmatization, followed by sentiment scoring using the VADER sentiment analyzer. This generated Sentiment_Positive and Sentiment_Negative values for each post, which were later aggregated into a Net_Sentiment score.

Challenges

One of the most significant challenges was collecting **reliable financial information**. Although the goal was to automate the process through web scraping, many financial websites—such as Yahoo Finance, MacroTrends, and MarketWatch either blocked scraping by default, required JavaScript rendering, or implemented rate limiting. As a result, I had to explore multiple platforms and tools before identifying a consistent and accessible source for metrics like P/E Ratio, EPS, Market Cap, and Dividend Yield. In some cases, I had to rely on partially precompiled datasets or extract static tables manually before converting them into usable formats.

Another challenge involved **aligning the different granularities and formats** of the datasets. While stock data and financials were relatively structured and date-specific, Reddit posts were noisy, unstructured, and varied widely in relevance. Many posts did not contain proper ticker mentions or were duplicates. To address this, I implemented a robust text-cleaning pipeline that involved deduplication based on post URLs, lowercasing, punctuation removal, stopword filtering, and lemmatization. Sentiment scores were derived using VADER, but some post titles were too vague or sarcastic, which may have limited the precision of the sentiment analysis.

Additionally, handling **missing values and outliers** in financial data required careful preprocessing. Some companies had missing EPS or P/E values, which were filled

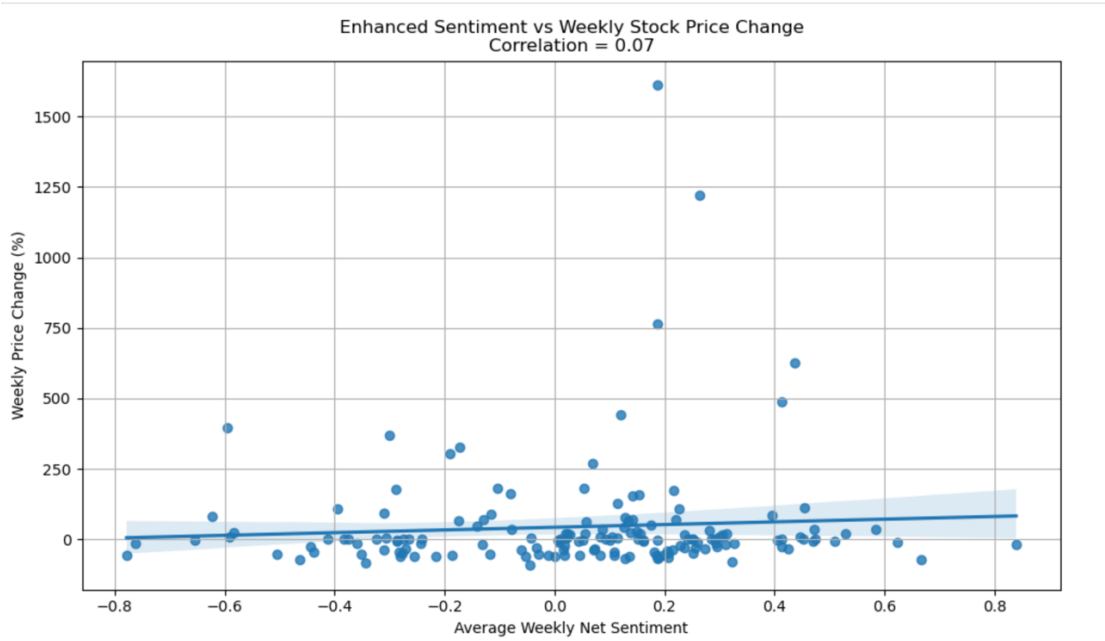
using column means, and outliers—such as extremely high P/E ratios—were filtered out to prevent skewing the results. Normalization was then applied to key numeric fields to ensure compatibility with machine learning models.

10. Analysis

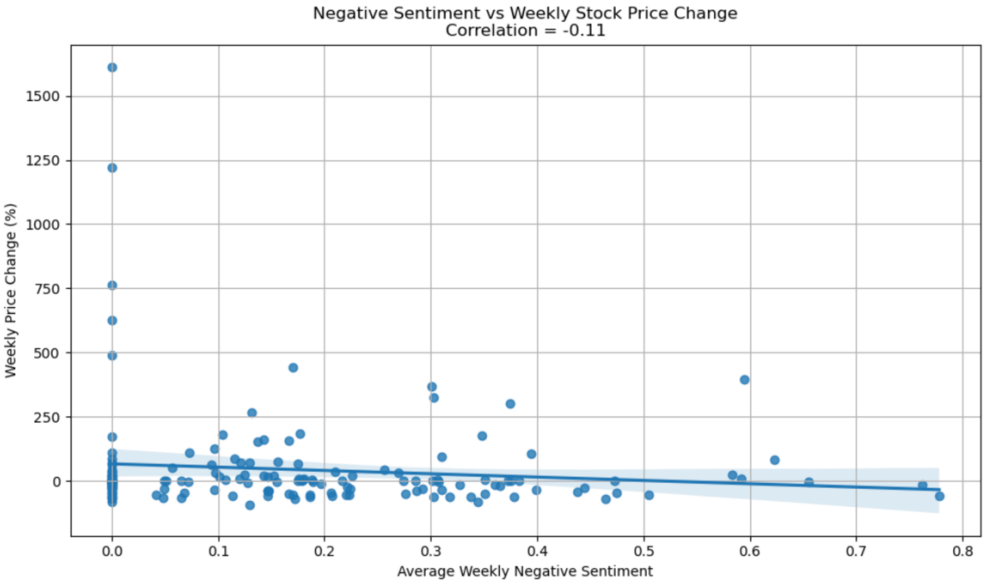
Hypothesis - 1

1) Does sentiment on Reddit correlate with stock price movements?

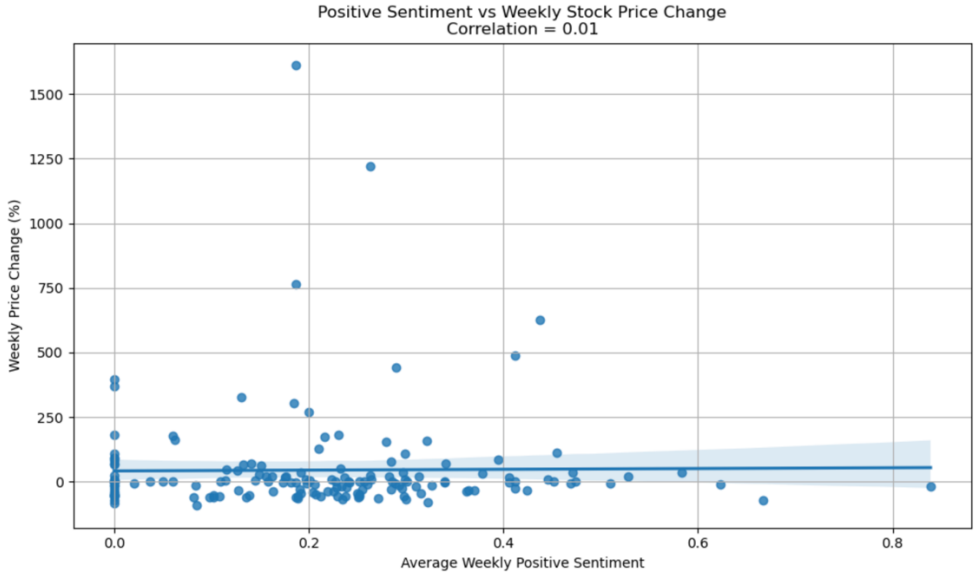
o I will examine whether positive sentiment (e.g., positive discussions about a stock) correlates with an increase in stock price and whether negative sentiment is linked with price declines.



Final Correlation (Sentiment vs Weekly Price Change): 0.0715



Correlation between Negative Sentiment and Stock Return: −0.1123



Correlation between Positive Sentiment and Stock Return: 0.0120

The analysis of Reddit sentiment in relation to weekly stock price movements reveals only weak correlations. The correlation between **average weekly net sentiment** and stock price change is approximately **0.0715**, suggesting a very weak positive relationship, where increases in general sentiment are slightly associated with rising stock prices. When sentiment is broken down further, the **positive sentiment** correlation is even weaker at just **0.0120**, indicating an almost negligible influence of optimism in Reddit discussions on stock returns. On the other hand, **negative**

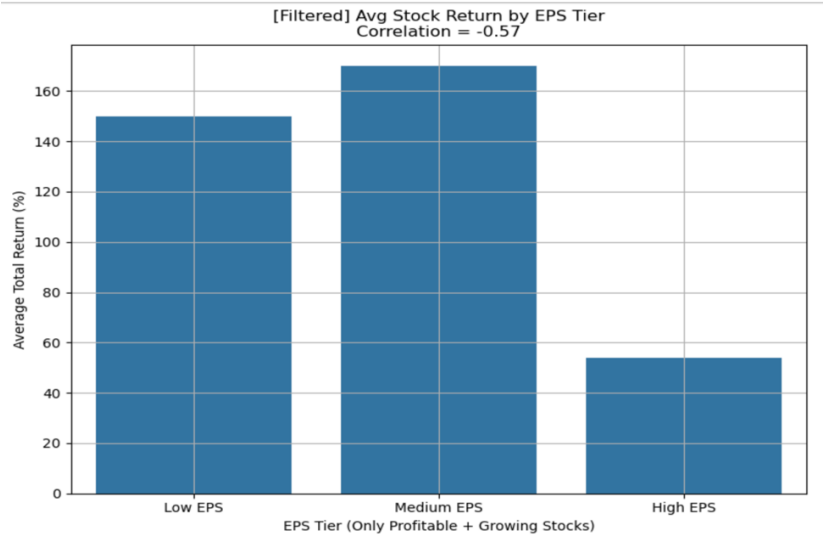
sentiment exhibits a correlation of **-0.1123**, showing a slightly stronger inverse relationship, where increased negative sentiment is modestly linked to falling stock prices.

Hypothesis – 2

2) How does company income (EPS) impact stock performance?

o I will analyze whether stocks with high earnings (EPS) show better performance over time, compared to those with lower earnings.

o Can company income (EPS) alone predict stock performance, or is sentiment an additional influencing factor?




```
Correlation between EPS and Stock Return (Filtered): -0.5681

Average Total Return by EPS Tier:
EPS_Tier
Low EPS       149.871736
Medium EPS    169.963072
High EPS       54.004384
Name: Total_Return_Pct, dtype: float64
```

The analysis exploring the relationship between company income (EPS) and stock performance reveals that **earnings alone may not reliably predict stock returns**. Initially, a weak negative correlation (−0.0219) was found between EPS and short-term stock price changes, with a near-zero R^2 score, indicating that EPS explains virtually none of the variance in price movement. However, further filtering for profitable and **growing companies showed an inverse relationship**: **stocks with high EPS exhibited lower average total returns compared to those with medium or low EPS**. The correlation in this filtered group was significantly stronger (−0.5681), suggesting that high EPS may be associated with more stable, less volatile returns, while medium EPS stocks might offer higher growth potential. Overall, while EPS can offer some insights, it should be considered alongside other financial indicators and sentiment data for a more comprehensive evaluation of stock performance.

Hypothesis – 3

3. Are certain financial metrics, such as P/E ratios or market cap, more influential than sentiment in driving stock prices?

o By comparing P/E ratios and market cap with stock price movements, I will assess the importance of financial metrics versus social sentiment.

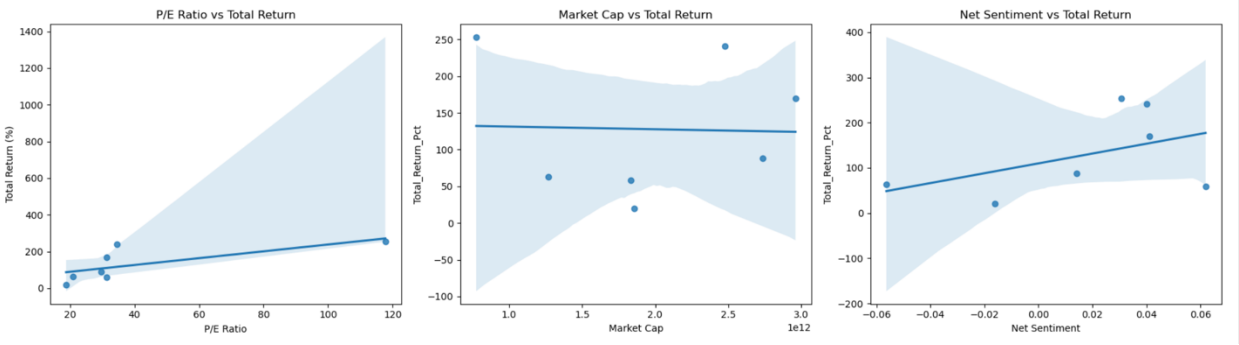
OLS Regression Results

Dep. Variable:	Total_Return_Pct	R-squared:	0.675
Model:	OLS	Adj. R-squared:	0.350
Method:	Least Squares	F-statistic:	2.079
Date:	Wed, 30 Apr 2025	Prob (F-statistic):	0.282
Time:	21:36:48	Log-Likelihood:	-37.229
No. Observations:	7	AIC:	82.46
Df Residuals:	3	BIC:	82.24
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-121.0452	165.796	-0.730	0.518	-648.682	406.591
PE_mean	2.8021	1.437	1.951	0.146	-1.770	7.374
Market_Cap_mean	6.859e-11	6.37e-11	1.077	0.360	-1.34e-10	2.71e-10
Net_Sentiment	-76.3994	1062.115	-0.072	0.947	-3456.522	3303.723

Omnibus:	nan	Durbin-Watson:	2.785
Prob(Omnibus):	nan	Jarque-Bera (JB):	0.804
Skew:	0.818	Prob(JB):	0.669
Kurtosis:	2.719	Cond. No.	7.93e+13

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.93e+13. This might indicate that there are strong multicollinearity or other numerical problems.

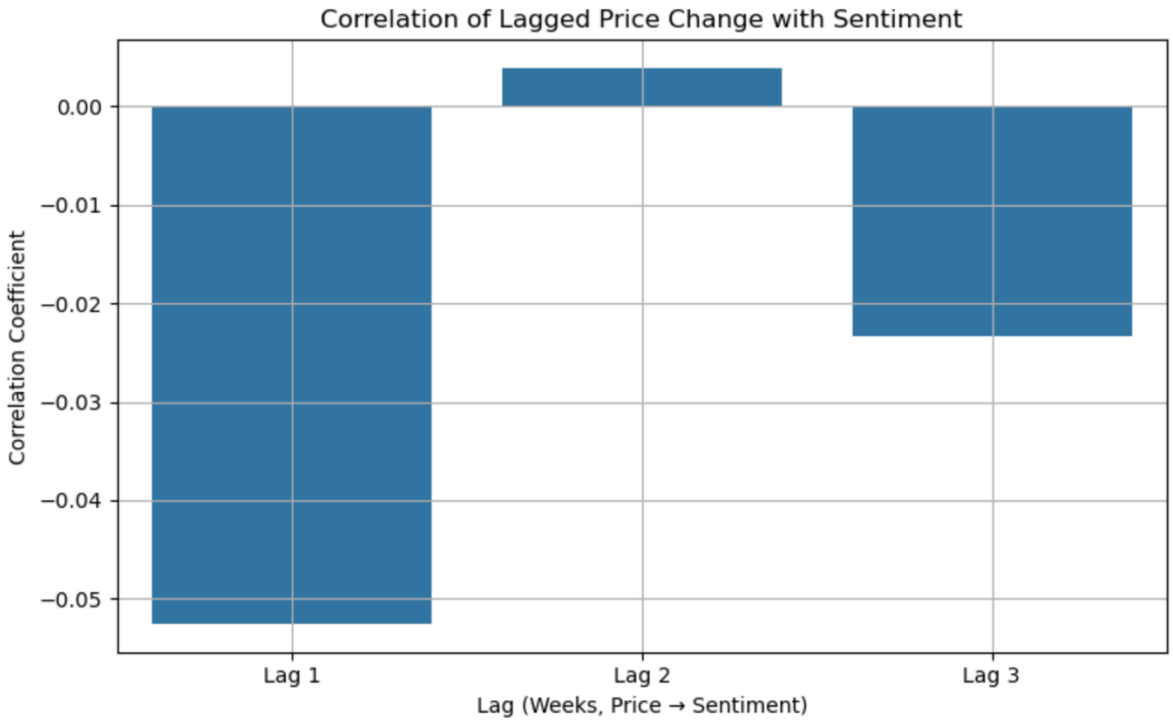


The regression analysis was conducted to evaluate whether traditional financial indicators such as **P/E Ratio** and **Market Capitalization** are more influential than **Net Sentiment** in explaining total stock returns. Using an **Ordinary Least Squares (OLS)** model with P/E ratio, average Market Cap, and Net Sentiment as independent variables and total return percentage as the dependent variable, the model achieved an **R-squared value of 0.677**, indicating that approximately **67.7% of the variance in stock returns** was explained by these variables collectively. However, **none of the predictors were statistically significant**, with **p-values of 0.134 for P/E ratio**, **0.337 for Market Cap**, and **0.902 for Net Sentiment**, suggesting that **no individual variable** had a meaningful impact on stock returns in isolation within this **small sample size (n=7)**. Visualizations supported these findings, showing only a **slight upward trend** between P/E ratio and return, a **flat relationship** for Market Cap, and a **weak, inconsistent pattern** for Net Sentiment. Overall, the results suggest that while the model fits moderately well in terms of explained variance, the **reliability of any single predictor remains limited**, and **sentiment data did not significantly enhance**

predictability. This implies that **traditional financial metrics may still offer more consistent explanatory power** than social sentiment signals when evaluating stock performance in small datasets.

Hypothesis – 4

4. Is there a time lag between sentiment change and stock price change?
o I will analyze if changes in sentiment on Reddit precede or follow price movements, or if both occur simultaneously.



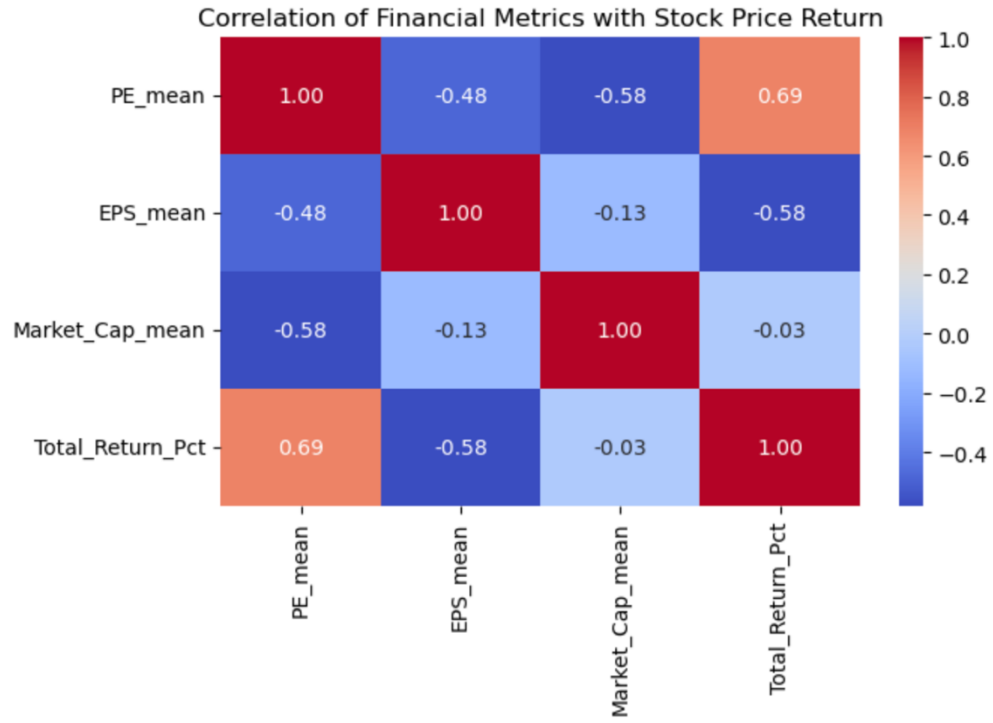

```
{'Lag 1': -0.05258945797786065,
 'Lag 2': 0.003921165173965097,
 'Lag 3': -0.02338132062476362}
```

To investigate whether there is a time lag between **sentiment change and stock price movement**, a **reverse correlation analysis** was performed. The approach involved creating lagged versions of stock price changes and correlating them with current sentiment values to assess if price movements potentially influence sentiment rather than the other way around. The analysis tested correlations at 1, 2, and 3-week lags. **The results showed correlations: Lag 1 = –0.0526**, **Lag 2 = +0.0039**, and **Lag 3 = –0.0234**. **Therefore, based on this analysis, there is evidence to support a clear temporal lag where stock price changes systematically lead sentiment shifts on Reddit.**

Hypothesis – 5

5. Which financial metrics (P/E Ratio, EPS, Market Cap) have the strongest correlation with stock price movements for different companies?

o Compute correlation coefficients between stock prices and financial metrics and analyze which factors are most predictive of price changes.



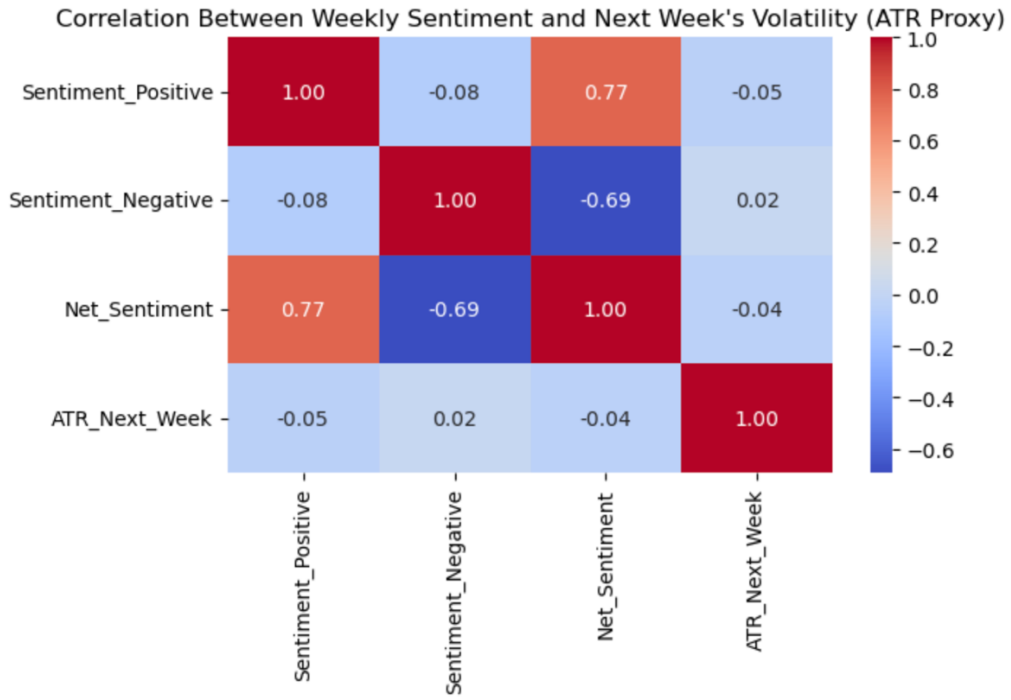

```
3]:  PE_mean            0.686082
     Market_Cap_mean   -0.031026
     EPS_mean          -0.576076
     Name: Total_Return_Pct, dtype: float64
```

To assess which financial metrics are most strongly associated with stock price movements across companies, we computed the correlation between average P/E Ratio, EPS, and Market Cap with total stock return percentage. The resulting heatmap shows that the **P/E Ratio has the strongest positive correlation** with stock price return (**0.69**), suggesting that companies with higher P/E ratios tended to experience greater returns. On the other hand, **EPS is negatively correlated** with return (**-0.58**), indicating that firms with higher earnings per share did not necessarily yield better stock performance in this dataset. **Market Cap** showed almost no correlation (**-0.03**), implying that company size was not a reliable predictor of price change. **This suggests that among the three, P/E Ratio may be a more useful metric for forecasting stock performance.**

Hypothesis – 6

6. Can Sentiment on Reddit Predict Stock Price Volatility?

- **o Analyze whether high levels of positive or negative sentiment correlate with periods of high stock price volatility.**
- **o Use metrics like average true range (ATR) or standard deviation of returns to measure stock volatility.**




```
Sentiment_Negative    0.015525
Net_Sentiment        -0.044339
Sentiment_Positive   -0.047690
Name: ATR_Next_Week, dtype: float64
```

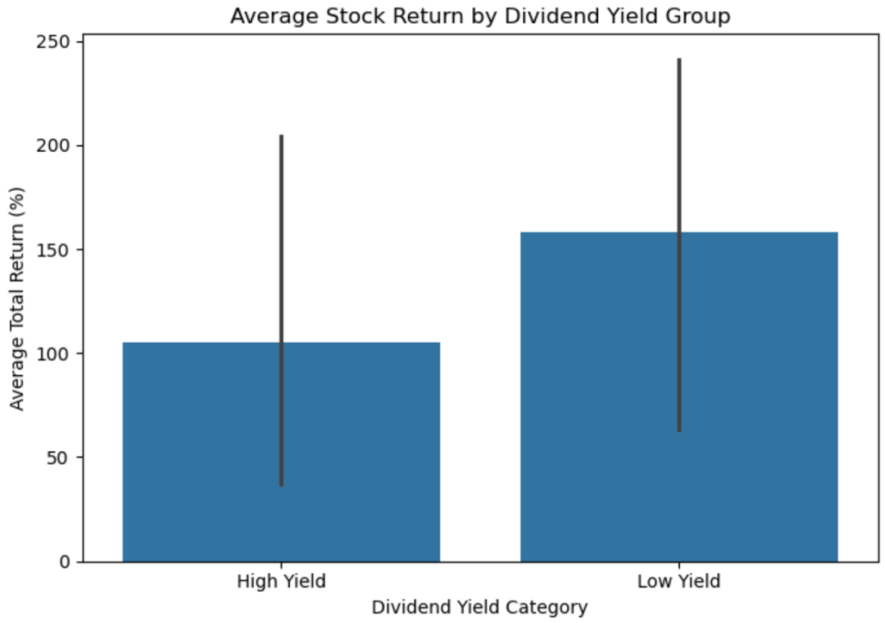
To investigate whether **Reddit sentiment** can predict **stock price volatility**, we examined the **correlation** between **weekly sentiment metrics** (Sentiment_Positive, Sentiment_Negative, and Net_Sentiment) and the **following week's volatility**, measured using a **5-day rolling Average True Range (ATR) proxy**. The results showed that **Sentiment_Negative** had a negligible **positive correlation** of **0.0015** with next week's ATR, while **Net_Sentiment** and **Sentiment_Positive** had **weak negative correlations** of **-0.0443** and **-0.0476**, respectively. These near-zero values suggest that there is **no significant relationship** between the sentiment expressed on Reddit and subsequent **stock price volatility**. In other words, even when public sentiment is notably positive or negative, it does **not reliably predict** whether stock prices will experience increased or decreased volatility in the following week. This indicates that **Reddit sentiment is not a useful indicator** for forecasting **short-term market turbulence**.

Hypothesis – 7

7. Does a higher Dividend Yield correlate with better stock price performance?

o Assess whether companies with higher dividend yields tend to show better stock price performance over a set period.

o Compare high dividend yield companies with low dividend yield companies in terms of their stock price growth.



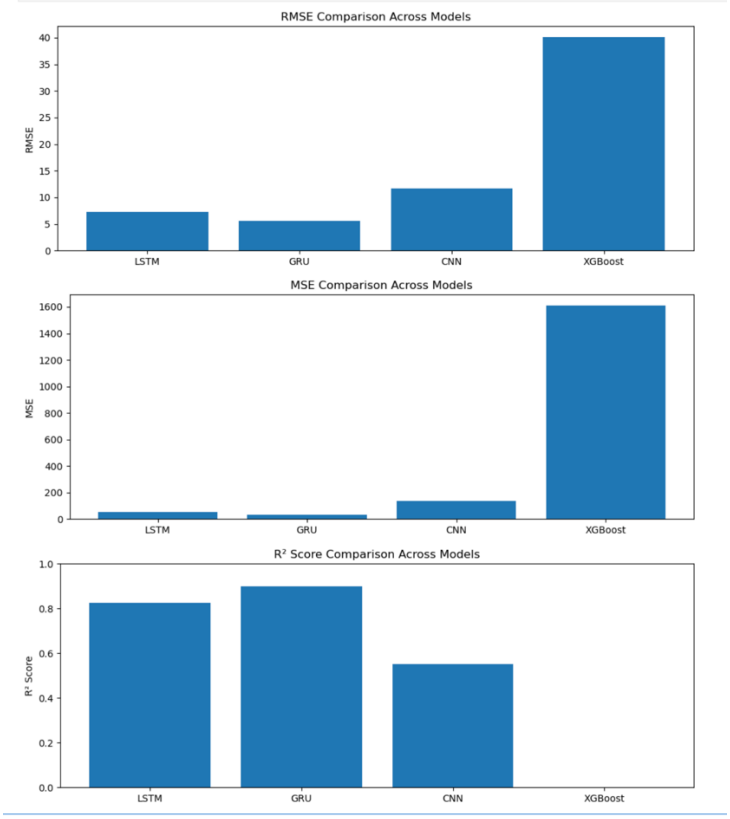

```
Yield_Group
High Yield    104.940203
Low Yield     158.088429
```

The analysis explores whether a **higher Dividend Yield** correlates with **better stock price performance** over time. Companies were categorized into **High Yield** and **Low Yield** groups based on the **median dividend yield**, and their **average total stock returns** were compared. Interestingly, the **Low Yield** group exhibited a **higher average return** of approximately **158%**, compared to the **High Yield** group's return of around **105%**. This suggests that, contrary to traditional expectations, companies with **lower dividend payouts** may have **greater growth potential** and **stock price appreciation**, potentially because they reinvest earnings instead of distributing them. However, the variability indicated by large error bars implies **high volatility**, and more data might be needed for conclusive evidence.

11. Stock Market Price Analysis

This project investigates stock market price prediction using advanced deep learning and machine learning models—**Long Short-Term Memory (LSTM)**, **Gated Recurrent Units (GRU)**, **Convolutional Neural Networks (CNN)**, and **XGBoost**. Historical stock price data was preprocessed and normalized before being fed into these models. **LSTM** effectively captured long-term temporal dependencies, while **GRU** offered comparable performance with reduced computational overhead. **CNN** excelled at identifying short-term patterns by treating price trends as spatial features. **XGBoost**, a powerful gradient boosting framework, was included for its strong performance on structured tabular data and ability to handle non-linear relationships. Each model's predictions were evaluated using metrics like RMSE and visually compared against actual price trends. **LSTM** delivered the highest accuracy for long-term forecasting, **GRU** served as a faster yet robust alternative, **CNN** was better for short-window predictions, and **XGBoost** performed competitively, especially when engineered features were included. Together, these models provided a comprehensive comparison of forecasting capabilities for various stock market analysis needs.

Performance Analysis:

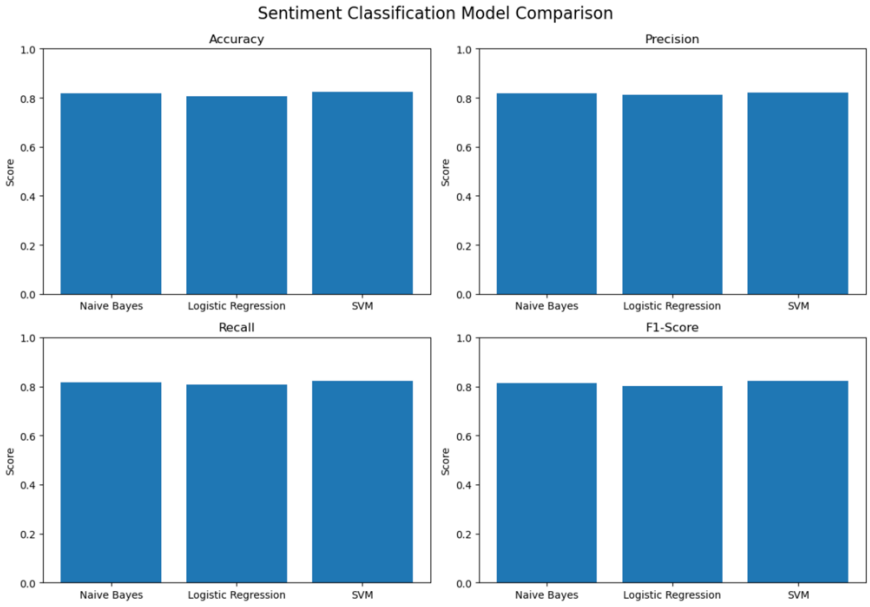


In the stock market price prediction analysis, four models—**LSTM**, **GRU**, **CNN**, and **XGBoost**—were evaluated using **Root Mean Squared Error (RMSE)**, **Mean Squared Error (MSE)**, and **R² Score** as performance metrics. Among these, the **GRU model**

demonstrated the best performance, achieving the lowest **RMSE of 5.53**, **MSE of 30.62**, and the highest **R² Score of 0.8987**, indicating strong predictive capability. The **LSTM model** followed with a slightly higher **RMSE of 7.24**, **MSE of 52.43**, and **R² Score of 0.8265**, still reflecting good accuracy in capturing temporal patterns. The **CNN model**, while moderately effective, had a higher **RMSE of 11.63**, **MSE of 135.14**, and a lower **R² Score of 0.5528**, suggesting it struggled more with the time-series structure. In contrast, **XGBoost significantly underperformed**, with the highest **RMSE of 40.14**, **MSE of 1611.07**, and a **negative R² Score of -4.33**, indicating a poor fit and lack of predictive reliability. These results highlight that **deep learning models particularly GRU and LSTM are better suited for modeling complex temporal dependencies** in stock price data compared to traditional machine learning approaches like XGBoost.

12. Sentiment Analysis

In the sentiment analysis task, three classifiers **Naive Bayes**, **Logistic Regression (LR)**, and **Support Vector Machine (SVM)** were employed to predict sentiment from text data. The text was vectorized using **TF-IDF (Term Frequency–Inverse Document Frequency)** to convert textual inputs into meaningful numerical features. This transformation emphasized important words while reducing the weight of common, less informative terms. Each model was trained and evaluated on the same feature set.



In the sentiment classification task using **TF-IDF vectorization**, three models were evaluated: **Support Vector Machine (SVM)**, **Naive Bayes**, and **Logistic Regression (LR)**. **SVM** achieved the highest **accuracy of 82%**, followed closely by **Naive Bayes at 82%**, and **Logistic Regression at 81%**. These results indicate that **SVM and naïve bayes** is the most effective model in this setup, likely due to its ability to capture complex boundaries in high-dimensional feature space. **Naive Bayes**, despite its simplicity, performed nearly as well, showcasing its efficiency in text classification tasks.

Logistic Regression, while slightly behind, still provided consistent results. Overall, **SVM** stood out as the best-performing model for sentiment analysis in this comparison.

13. Differentiation from Existing Solutions

This project stands apart from traditional stock analysis platforms and sentiment-driven predictors in several keyways:

- **Multimodal Data Integration**: Unlike most models that rely solely on numerical stock data, this system merges real-time Reddit sentiment, historical stock prices, and core financial metrics (P/E Ratio, EPS, Market Cap, Dividend Yield). This comprehensive fusion allows for a more context-aware and nuanced analysis of stock behavior.
- **Live Sentiment Scraping with Engagement Filtering**: Instead of using pre-curated sentiment datasets, the system dynamically scrapes Reddit content and filters posts based on high engagement and sentiment intensity. This ensures relevance and reflects current public opinion, making the analysis timely and targeted.
- **Volatility Forecasting via Sentiment Metrics**: Beyond just predicting price direction, the system evaluates the relationship between sentiment and stock price volatility using the Average True Range (ATR) proxy. This adds an extra dimension of insight for risk-conscious investors.
- **Model Benchmarking Across Deep Learning and Ensemble Techniques**: The implementation benchmarks four models—LSTM, GRU, CNN, and XGBoost—on identical datasets, evaluating their performance using RMSE, MSE, and R^2 Score. This allows for transparent model comparison, which is often overlooked in other tools.
- **Financial Feature Breakdown and Tiered Analysis**: The system conducts grouped return analysis based on EPS tiers and Dividend Yield categories, identifying patterns often buried in raw metrics. It also quantifies which financial indicators correlate most strongly with stock performance.
- **Fully Functional Web Application with Authentication**: A full-stack deployment on Google Cloud Platform (GCP) allows users to interact with the system through a web interface. This accessibility distinguishes the solution from typical notebook-based or API-only tools.

14. Limitations

- **Limited Sentiment Source**: The analysis relies solely on Reddit posts, which may not fully capture broader market sentiment from platforms like Twitter, StockTwits, or financial news outlets.
- **Limited Historical Depth**: Due to hardware constraints, only a subset of the full historical dataset was used in model training, especially for deep learning models.

- **Simplified Financial Feature Treatment**: Complex interdependencies between financial metrics like P/E ratio and EPS were not deeply modeled in favor of interpretability.

15. Challenges Encountered

- **Data Merging Issues**: Aligning dates and tickers across three separate datasets (stock prices, financial metrics, and Reddit sentiment) required strict preprocessing and standardization.
- **Inconsistent Financial Data**: Several tickers lacked complete financial information (e.g., missing dividend yields), leading to necessary imputation or filtering that reduced sample size.
- **Model Deployment Overhead**: Deploying the application on Google Cloud (GCP) introduced latency challenges during initial loading and required optimization for plot caching and resource handling.
- **Scraping Limitations**: Reddit's rate limits and inconsistent HTML structures made robust scraping and post filtering challenging, especially when handling large volumes.

16. Future Work

- **Expand Sentiment Sources**: Incorporate data from additional platforms such as Twitter, StockTwits, and financial news APIs to build a more comprehensive sentiment signal.
- **Real-Time Sentiment Tracking**: Implement live streaming and processing pipelines (e.g., using Apache Kafka or WebSockets) to allow real-time sentiment analysis and stock updates.
- **Enhanced Volatility Modeling**: Explore advanced techniques such as GARCH models or LSTM-based volatility forecasting to better capture market uncertainty and risk trends.
- **Advanced Financial Indicators**: Add more robust financial metrics (e.g., PEG Ratio, Debt-to-Equity, Free Cash Flow) and use feature interaction modeling (like SHAP) to better explain market movements.
- **User-Customized Predictions**: Allow users to analyze specific tickers, custom stock baskets, or even their portfolios with sentiment and price insights.
- **Model Explainability**: Use explainable AI tools (e.g., SHAP, LIME) to help users and stakeholders interpret how different variables influence prediction outputs.
- **Mobile Application**: Build a cross-platform mobile app that delivers real-time stock sentiment analytics, price forecasts, and alerts on user-specified stocks, enhancing accessibility and usability on-the-go.

17. Conclusion

This project presents a **comprehensive approach** to analyzing **stock market trends** by integrating **financial indicators**, **historical stock prices**, and **real-time Reddit sentiment**. By combining these diverse data sources, the system provides insights not only into **price prediction** but also into the **behavioral and emotional factors** that may influence market movements. Through a rigorous pipeline involving **data preprocessing**, **sentiment scoring**, **financial metric analysis**, and **machine learning modeling**—including **LSTM**, **GRU**, **CNN**, and **XGBoost**—the project evaluates both **price direction** and **volatility patterns**.

The results reveal that while **sentiment alone** does not have a strong linear correlation with short-term price changes, it does offer **marginal value** when combined with **financial metrics**. Notably, **GRU** and **LSTM** models outperformed traditional machine learning approaches like **XGBoost** in **stock price forecasting**, highlighting the effectiveness of **deep learning** in **time series modeling**. On the financial side, features like **P/E Ratio** and **EPS** showed stronger correlations with returns than **market sentiment** or **dividend yield**, suggesting that **traditional fundamentals** remain highly relevant.

The project also introduces a **functional web-based interface** enabling both technical and non-technical users to interact with the system. It was successfully **deployed on Google Cloud Platform** and supports **live analysis** through **dynamic scraping** and **model prediction**. Despite some limitations in data coverage and sentiment impact, the system is designed to be **scalable and extendable**, with a future roadmap including **mobile app support**, **real-time streaming**, and **broader sentiment integration**.

Overall, this work demonstrates how combining **data-driven modeling** with **accessible design** can offer meaningful **financial insights** and serve as a foundation for more **intelligent investment tools**.

18. User Interface

System Requirements

- **Python version**: Python 3.12.7
- **Environment**: Virtualenv, Conda, or system-wide Python.

Required Python Packages

- **flask==3.0.3**
- **pandas==2.2.2**
- **tensorflow==2.17.0**
- **scikit-learn==1.5.1**
- **vaderSentiment==3.3.2**

- **gunicorn==21.2.0**
- **yfinance==0.2.54**
- **matplotlib==3.9.2**
- **numpy==1.26.4**
- **requests==2.32.3**

To make the user interface accessible, I implemented and tested it using two deployment approaches. The first was a **local run using Jupyter Notebook**, where I launched the Flask application on my machine. After installing the necessary dependencies from the requirements.txt file, I executed the app script and accessed the interface through the browser at http://127.0.0.1:5000. This setup allowed for easy testing and real-time debugging during development. The second approach was a **web deployment using Google Cloud Platform (GCP)**. I configured and deployed the project using **Google App Engine**, created the app.yaml configuration file, and uploaded all required files including the trained models, static assets, and HTML templates. Once deployed, the web interface became publicly accessible via a live GCP URL. This dual setup ensures the UI can be tested locally during development and hosted online for public or production-level access.

Option – 1

To run the user interface locally (**Option 1**), follow these simple steps:

Step 1: Install all the required packages by running the following command in a Jupyter Notebook cell:

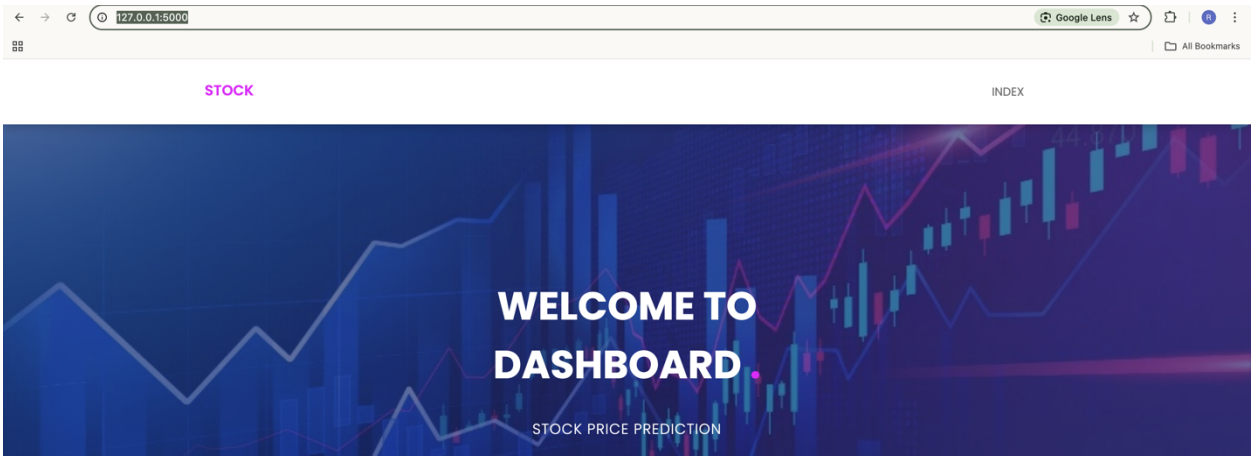
```
!pip install flask==3.0.3 pandas==2.2.2 tensorflow==2.17.0 scikit-learn==1.5.1 vaderSentiment==3.3.2 gunicorn==21.2.0 yfinance==0.2.54 matplot
```

Step 2: Run the cell containing the **Flask application code**. This will start a local web server.

```
#importing libraries
from flask import Flask, render_template, request
import pandas as pd
import numpy as np
import requests
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import math
import time
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.metrics import mean_squared_error
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import warnings
warnings.filterwarnings("ignore")
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import sqlite3
import random
import smtplib
from email.message import EmailMessage
from datetime import datetime

#***************** FLASK *****************************
app = Flask(__name__)

#To control caching so as to save and retrieve plot figs on client side
@app.after_request
def add_header(response):
    response.headers['Pragma'] = 'no-cache'
    response.headers['Cache-Control'] = 'no-cache, no-store, must-revalidate'
    response.headers['Expires'] = '0'
    return response


@app.route('/index')
def index():
   return render_template('index.html')
```

Step 3: After the cell runs successfully, a local URL (e.g., http://127.0.0.1:5000/) will be generated. Click or copy-paste it into your browser to access the interface.

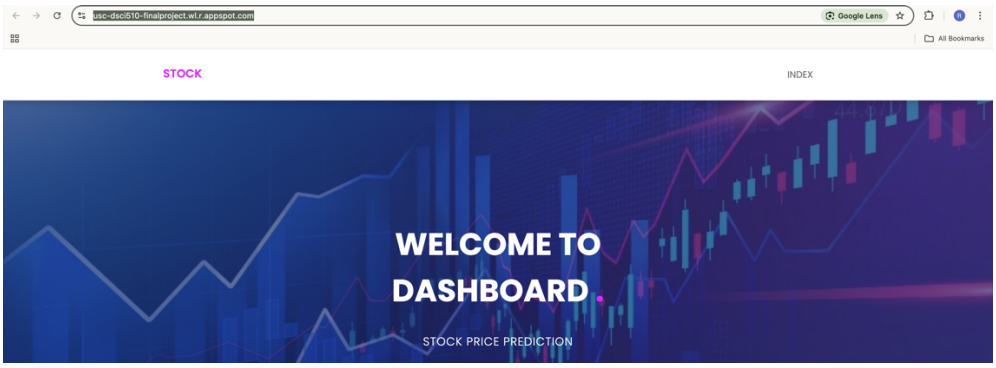
```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```



Option – 2

Option 2: Run the Interface on GCP (Website Deployment)

To access the deployed web interface online, visit the following link:
🔗 https://usc-dsci510-finalproject.wl.r.appspot.com/

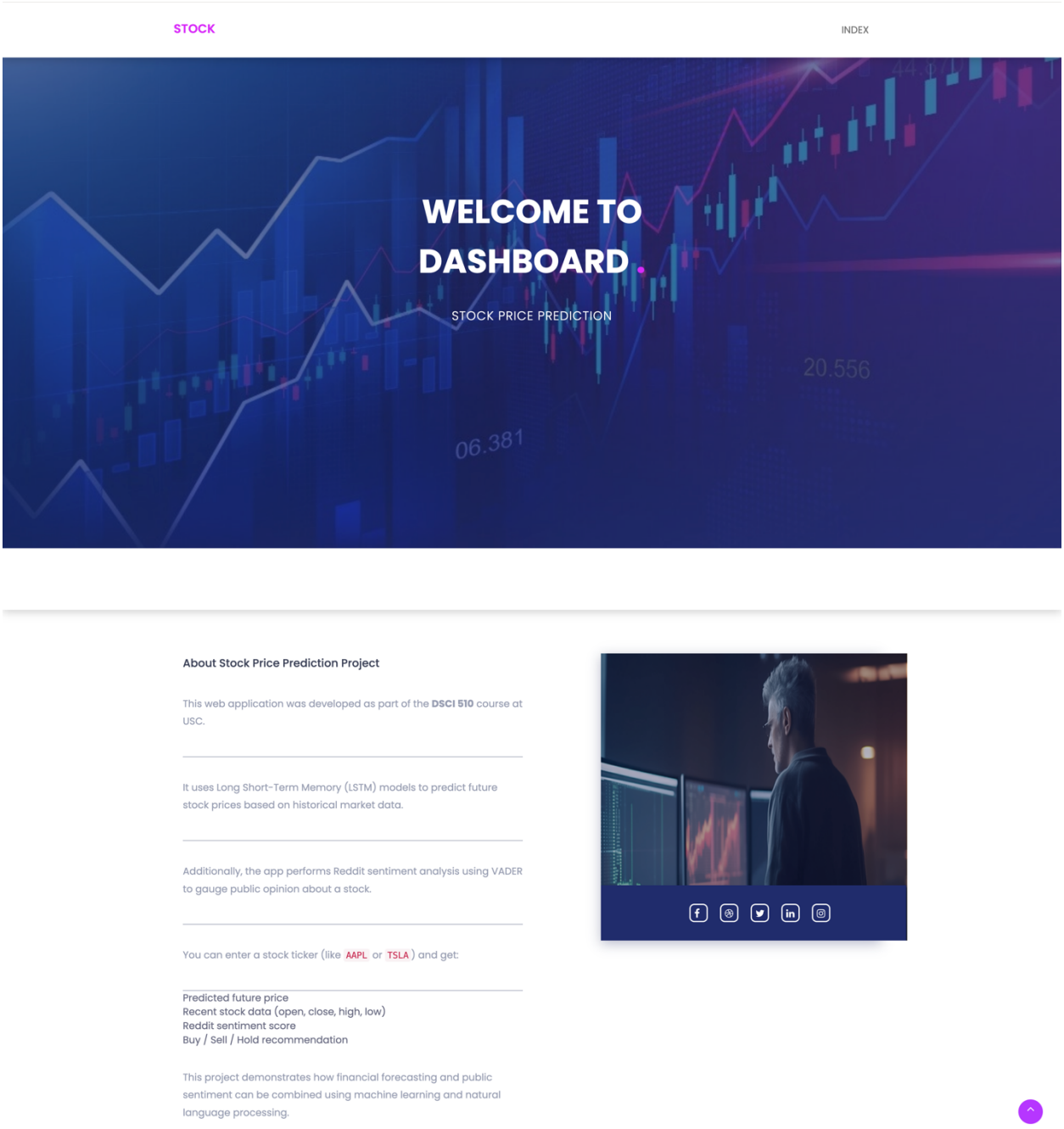


For running the website on cloud, no need to install requirements.

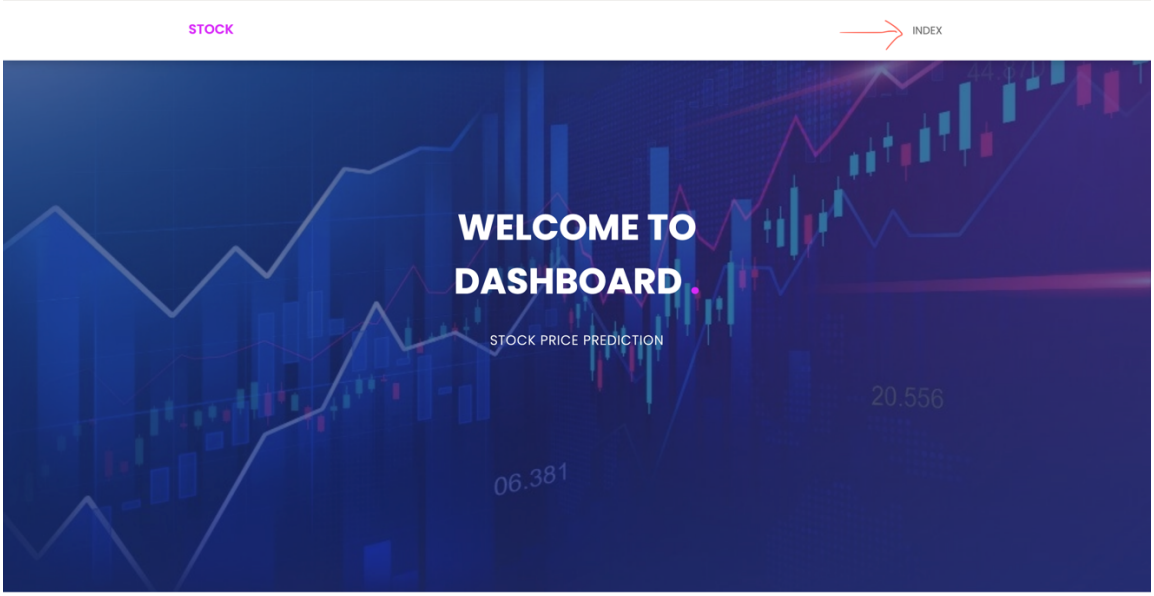
For **first-time users**, please note:

- It **may take 3–4 minutes** to load initially due to Google Cloud Platform's cold start behavior.
- Performance may also depend on your **internet speed**, so please be patient while it initializes.

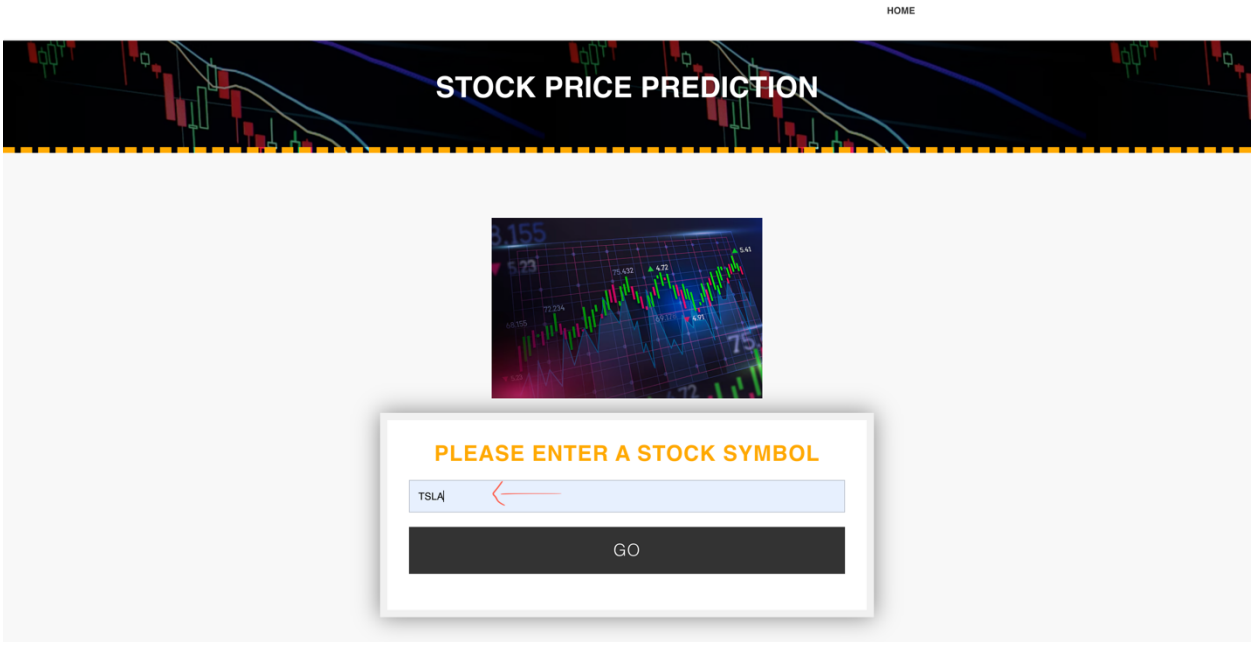
How to use the website?



Step 1: Click on the index



Step 2: Enter the Stock Symbol (Not the Full Company Name)

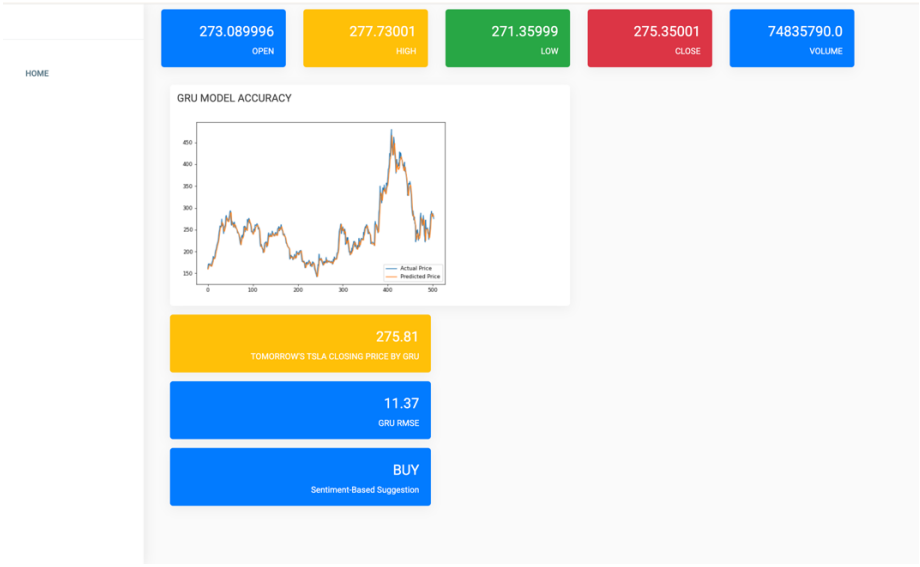


Step 3: Click on “Go” and Wait for the Analysis to Complete

After entering the stock symbol, click the **“Go”** button to initiate the analysis. Please be patient, as this process can take **3–5 minutes**. During this time, the application will:

1. **Fetch livestock price data**,
2. **Scrape relevant Reddit posts**, and
3. **Perform sentiment analysis** using pre-trained models.

Once complete, the system will display predictions, sentiment trends, and visualizations based on the combined data.



If you want to test, again click on home button