Advanced Data Analytics

Big Data Storage and Processing

*Assignment Title: MSC\_DA\_BD\_ADAv4 Repeat*

*Student Name: Dorin Buzilov*

*Student ID: sba20274*

# Table Of Contents

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

[**Table Of Contents 2**](#_1f305p6xybig)

[**Introduction 3**](#_4t5y6o9qo4zd)

[**Python Programming Language 3**](#_78b1689t8yqs)

[**Big Data 3**](#_34hq1dixrp8h)

[**HDFS 3**](#_v1bb87bb6q07)

[**Apache Spark 4**](#_mvuritqd3fzl)

[**Big Data Storage 5**](#_fcwx3upzfo8n)

[SQL vs. NoSQL: 5](#_q7p4zchrz81c)

[Comparative Analysis 5](#_rqpawchhs6xt)

[Results Table 6](#_wgl01r5qskbm)

[Results 6](#_y6k2lllgs6bv)

[Fig.1 - YCSB Latency Comparison 6](#_88q3u36x61li)

[Fig.2 - YCSB ThroughputComparison 7](#_kdzqncf4tm16)

[Justification for Choosing MongoDB 7](#_hu6n8ckcnmar)

[**Data Architecture 8**](#_ted2buh2zxmo)

[Explanation of the Data Architecture 9](#_ncs75rjmv3yn)

[**Detailed Data Storage and Processing Activities 10**](#_vxv3kna2t28e)

[**Exploratory Data Analysis 11**](#_172wk7rlgyn6)

[Twitter Data Distribution: 11](#_1miivu85w68x)

[Temporal Review of Tweets: 13](#_eryet54vxxlq)

[Sentiment Analysis: 13](#_gub9pwynd3jg)

[Close Prices and Sentiment Over Time: 15](#_ie2qzt2r2cp)

[**Time Series Forecasting 17**](#_3wd90a1151f5)

[**ARIMA Model 17**](#_3rmnqahuzga8)

[SARIMAX Model 17](#_ktdnjlcj02wh)

[LSTM Model 18](#_p95er2zhpdz6)

[**Using Streamlit and Ngrok for Hosting Dashboards 19**](#_5qup5gi9w15o)

[**Evaluation of ARIMA and SARIMAX Models for Stock Market Forecasting Using Sentiment Analysis 20**](#_w9a8b1g4r5bv)

[ARIMA Model Evaluation 20](#_6etnw4p4ideo)

[SARIMAX Model Evaluation 21](#_cvlaskjfipul)

[LSTM Forecasts 22](#_opz2x0uie5ow)

[Conclusion 23](#_vr9fdf0xq0y)

# Introduction

In this project, we aimed to analyze a large dataset collected from twitter and finance data, focusing on the stock market sentiment and historical stock prices. The data storage and processing activities were carried out using Apache Spark for distributed data processing and MongoDB as our chosen NoSQL database. Machine Learning and Neural Networks were trained on the data to forecast market prices and to review whether sentiment analysis from twitter data can impact market prediction in a meaningful way.

# Python Programming Language

***Python*** was chosen as the programming language for this project due to its simplicity, readability, and extensive libraries for data processing and analysis. Python’s rich ecosystem of data science libraries (such as pandas and matplotlib) enables efficient data manipulation and visualization. Additionally, Python’s compatibility with various databases and support for web frameworks made it an ideal choice for this project.

# Big Data

In the modern digital age data is the driving force of many industries. Data is produced on such a large scale that new technologies have been developed to store and manage this data traffic in more efficient ways than ever before. Traditional data storage solutions did not offer efficient management at larger scale and security features which are present in modern solutions.

# HDFS

Hadoop includes a fault-tolerant storage system called Hadoop Distributed File System, or HDFS. HDFS is able to store huge amounts of information, scale up incrementally and survive the failure of significant parts of storage infrastructure without losing data. Hadoop creates clusters of machines and coordinates work among them. Clusters can be built with inexpensive computers. If one fails, Hadoop continues to operate the cluster without losing data or interrupting work, by shifting work to the remaining machines in the cluster. HDFS manages storage on the clusters by breaking incoming flies into pieces, called “blocks” and storing each of the blocks redundantly across the pool of servers. In the common case, HDFS stores three complete copies of each file by copying each piece to three different servers. It does so through **MapReduce**: A programming model for processing large data sets with a distributed algorithm on a cluster.

# Apache Spark

Spark was created to address some of the limitations of Hadoop MapReduce, particularly around speed and ease of use. Spark introduces several improvements and new features that make it more efficient and flexible compared to Hadoop:

**In-Memory Processing**:

* **Hadoop**: Uses a disk-based storage mechanism which involves reading and writing intermediate data to HDFS between each MapReduce job.
* **Spark**: Uses in-memory processing to store intermediate results, reducing the need to write to disk. This makes Spark much faster (up to 100 times faster for certain applications).

**Ease of Use**:

* **Hadoop**: MapReduce requires complex code to perform even simple operations, making it harder for developers to write and maintain jobs.
* **Spark**: Provides easy-to-use APIs for Java, Scala, Python, and R. This simplifies the development process, allowing for concise and readable code.

**Advanced Analytics**:

* **Hadoop**: Primarily designed for batch processing.
* **Spark**: Supports not only batch processing but also real-time stream processing, interactive queries, and machine learning through its built-in libraries like Spark Streaming, Spark SQL, and MLlib.

For these reasons apache spark has been chosen as the data processing and management framework. The data used within this project is stagnant within a directory until retrieved by spark but we would like to have the capabilities within our chosen framework to process real time data if it were to become available to us. That is why Spark is most appropriate for this task considering that twitter tweets and financial data are both data types which are produced at a large scale and with high velocity. Spark also offers high-level APIs in Python, making it easier to implement complex data processing tasks.

# Big Data Storage

## SQL vs. NoSQL:

* MySQL: MySQL is a widely used open-source relational database management system (RDBMS). It was chosen due to its robust support for transactional operations, ACID compliance, and strong consistency guarantees. MySQL is particularly well-suited for applications requiring structured data storage with complex relationships and rigorous data integrity.
* MongoDB: MongoDB is a popular NoSQL database that offers flexibility and scalability. It was selected for its ability to handle unstructured data and horizontal scaling. MongoDB’s schema-less nature allows for the storage of diverse data types and rapid iteration on the data model, making it suitable for applications with evolving data requirements.

## Comparative Analysis

The comparative analysis between MySQL and MongoDB was conducted using YCSB to evaluate key performance metrics such as latency and throughput. The workload tasked to the databases was;

# Define the new content for the workloada file

new\_workload\_content = """

recordcount=10000

operationcount=10000

workload=site.ycsb.workloads.CoreWorkload

readallfields=true

readproportion=0.5

updateproportion=0.5

scanproportion=0

insertproportion=0.5

requestdistribution=zipfian

"""

The specifications of record count 10,000 was chosen to simulate a similar quantity of data we have present in our datasets. The results of the benchmarking are summarized in the following tables and visualizations:

***Latency Comparison***

The latency comparison indicates the time taken to perform read and write operations in microseconds (us). Lower latency is preferable as it signifies faster response times.

### 

***Throughput Comparison***

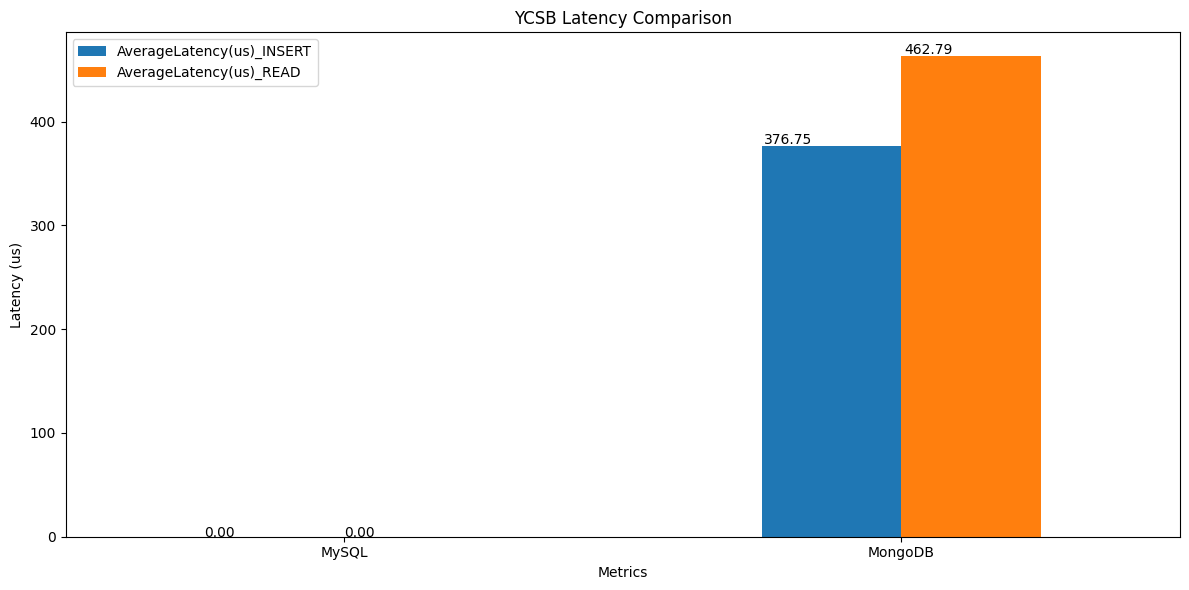
The throughput comparison measures the number of operations per second (ops/sec) that the databases can handle. Higher throughput indicates better performance under load.

### Results Table

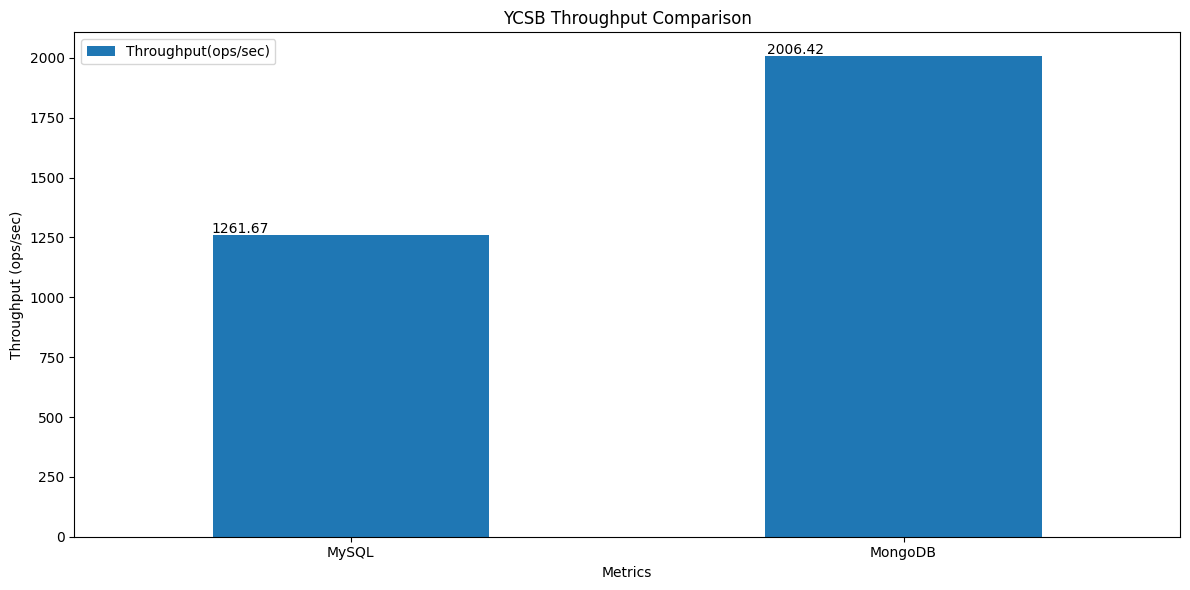
|  | ***RunTime(ms)*** | ***Throughput(ops/sec)*** | ***AverageLatency(us)\_INSERT*** | ***AverageLatency(us)\_READ*** |
| --- | --- | --- | --- | --- |
| *MySQL* | *799.0* | *1251.564456* | *NaN* | *NaN* |
| *MongoDB* | *834.0* | *1199.040767* | *449.304636* | *689.165266* |

### 

### Results

* **Latency**: MongoDB shows significantly lower latencies for both read and write operations compared to MySQL. This demonstrates MongoDB's efficiency in handling high-throughput operations. MySQL’s latencies are not available for read and write operations, indicating that it did not perform as efficiently under the given workload.

##### Fig.1 - YCSB Latency Comparison

* **Throughput**: MongoDB demonstrates higher throughput compared to MySQL. This can be attributed to MongoDB’s optimized handling of high-throughput workloads and its schema-less design which allows for rapid data ingestion. MySQL, while robust, shows a lower throughput, possibly due to the overhead associated with ensuring ACID compliance and handling complex queries.

##### Fig.2 - YCSB ThroughputComparison

The comparative analysis highlights the strengths and weaknesses of both MySQL and MongoDB. MongoDB excels in scenarios requiring high throughput and low-latency read and write operations, making it ideal for applications with high data ingestion rates and flexible schema requirements. MySQL, while showing lower performance in this specific benchmark, remains a strong choice for applications requiring structured data storage and strong consistency guarantees.

## Justification for Choosing MongoDB

**Flexibility and Schema-less Design**:

* **Twitter Data**: Twitter tweets are unstructured and come with varying fields such as text, hashtags, user information, and metadata. MongoDB's document-oriented storage model allows for flexible schema design, accommodating the diverse and evolving nature of tweet data without requiring predefined schemas.
* **Financial Data**: Financial data can range from structured transactional records to semi-structured market feed data. MongoDB’s flexibility allows for the efficient storage and retrieval of such heterogeneous data types, enabling rapid adaptation to changing data requirements.

**High Throughput and Low Latency**:

* The YCSB benchmarking results demonstrate MongoDB's superior performance in terms of throughput and latency compared to traditional relational databases like MySQL. For applications dealing with real-time data ingestion and analysis, such as processing live financial transactions and Twitter streams, MongoDB provides the necessary performance to handle large-scale, rapid data inflow efficiently.

**Ease of Use**:

* DB has demonstrated a simpler setup process and ease of use parallel to python with its pymongo library.

# Data Architecture

**

Fig.3 - Overview of Hardware and Software Stack

##### 

##### 

Fig.4 - Data Flow and Processing Architecture

## Explanation of the Data Architecture

1. **Data Ingestion:**
   * Data is ingested from CSV files which are a common format for financial and social media data. This raw data is loaded into the system and processed using Apache Spark.
2. **Data Management and Cleaning:**
   * Spark is used for both data management and cleaning due to its powerful in-memory processing capabilities. This ensures that data is efficiently processed and cleaned, ready for further analysis.
3. **Data Storage:**
   * Cleaned data is stored in MongoDB as JSON documents. MongoDB is chosen for its flexibility in handling unstructured data and its ability to scale horizontally, making it suitable for high-volume data such as tweets and financial transactions.
4. **Data Processing:**
   * Data is imported into the working environment using Pandas for data manipulation and preparation for machine learning tasks. Pandas provides a user-friendly interface for data analysis and manipulation.
5. **Machine Learning and Forecasting:**
   * Various libraries such as Numpy, scikit-learn, and TensorFlow are used for building and training machine learning models. These libraries provide comprehensive tools for developing advanced machine learning models for forecasting financial data and analyzing twitter tweet data.
6. **Visualization:**
   * The results of the data processing and machine learning models are visualized in a dashboard. This provides a clear and interactive way to present insights derived from the data.

# Detailed Data Storage and Processing Activities

In this project, the Google Colab environment was set up by downloading Java OpenJDK 8, Spark 3.1.1, and Hadoop 3.2, and installing PySpark to the system path. Spark was then initialized using the PySpark library. MongoDB, specifically the Linux-x86\_64-ubuntu2004 version 4.4.6, was downloaded and configured to run due to Google Colab's Ubuntu Linux operating system. OpenSSL packages, which provide cryptographic functionalities to secure communications over networks, were also downloaded. MongoDB was set up to run using nohup, and a /data/db directory path was specified for storing all database information. The Google Drive directory was mounted to provide the project full access to folders and their contents.

The raw data files, stored in CSV format, were processed using Spark. For the Twitter tweets data, Spark was used to read the files and clean the tweets using regex patterns to remove emoticons, symbols, pictographs, and other special characters. Examination of the top 50 rows of the Spark DataFrame revealed that the ID column, which should contain 6-digit numerical IDs, had some rows containing tweet information instead. This anomaly indicated that tweets were overflowing into the ID column from previous rows. An algorithm was run to find non-numerical rows and append them to the tweets in the preceding row. After cleaning, non-numerical rows were dropped, duplicates were removed, missing values were checked, and the correct format for each column was ensured. The cleaned data was then migrated to MongoDB and stored as a JSON file. Storing data as JSON in MongoDB leverages the database's strengths in handling flexible, nested document structures.

For the raw stock prices data, individual CSV files for each stock were read by Spark and merged into one DataFrame. All files contained (254 rows, 8 columns) except for stock ABNB, which had only 15 rows and was therefore removed. The data was cleaned using PySpark by removing non-numeric values and duplicates. A date filter was applied to ensure the data fell between January 1, 2020, and December 31, 2020. The data was then checked for missing values. This thorough data preparation and cleaning process ensured that the datasets were ready for further analysis and storage in MongoDB.

# Exploratory Data Analysis

Data cleaning was conducted during data ingestion before being parked as JSON files in MongoDB. Pandas was used to pull the JSON files from MongoDB as it is a powerful library when it comes to data manipulation and preparation of data for machine learning.

## Twitter Data Distribution:

**Number of Tweets per Ticker:** The Twitter tweets were plotted to review the distribution of tweets amongst the stocks. The top 5 companies were selected based on the number of tweets: Tesla (TSLA), Apple (AAPL), Boeing (BA), Disney (DIS), and Amazon (AMZN).

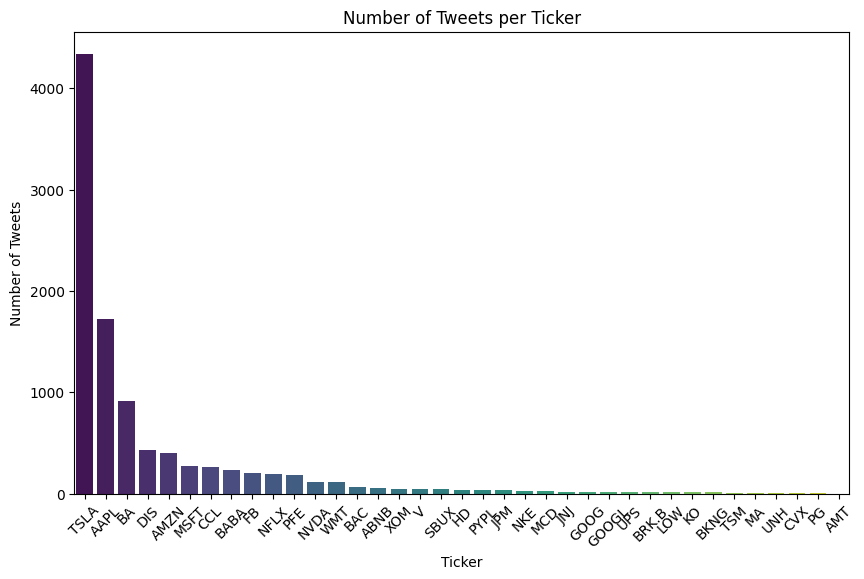


Fig.5- Number of Tweets per Stock

**Distribution of Tweets Amongst Top 5 Companies:** A pie chart visualized the distribution of tweets among the top 5 companies, showing that Tesla (TSLA) had the largest share at 55.5%, followed by Apple (AAPL) at 22%, Boeing (BA) at 11.8%, Disney (DIS) at 5.5%, and Amazon (AMZN) at 5.2%.

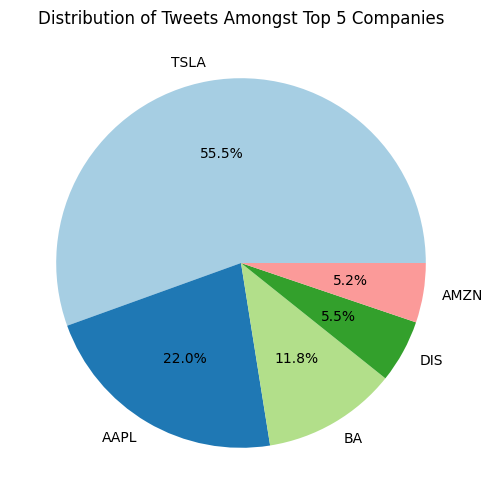


Fig.6- Number of Tweets per Stock

Since the scope of this project is to evaluate whether we can make stock market predictions with the influence of sentiment analysis, we want to include the companies which have the largest amount of data available to train our models on.

## 

## 

## Temporal Review of Tweets:

**Number of Tweets Over Time:** Temporal review of the tweets over time showed three large spikes on May 1, 2020, September 3, 2020, and September 22, 2020.

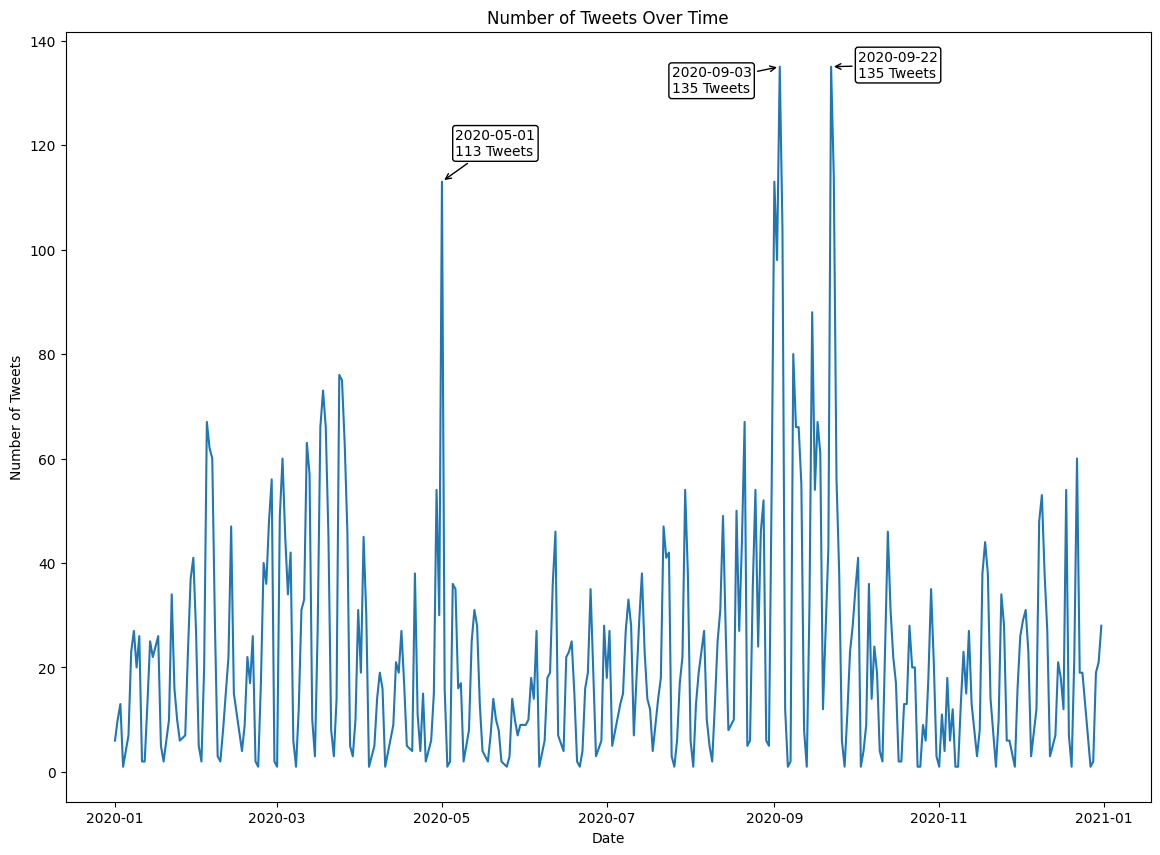


Fig.7- Number of Tweets Over Time

## 

## 

## Sentiment Analysis:

**VADER and NLTK Sentiment Distribution:** NLTK and VADER are both sentiment analysis libraries available in Python. They were allowed to run on our data to review their outputs. Almost identical in results, we decided to choose VADER as our chosen sentiment analysis tool because it is specifically tuned for social media texts, easy to implement, and provides reliable results for short texts like tweets.

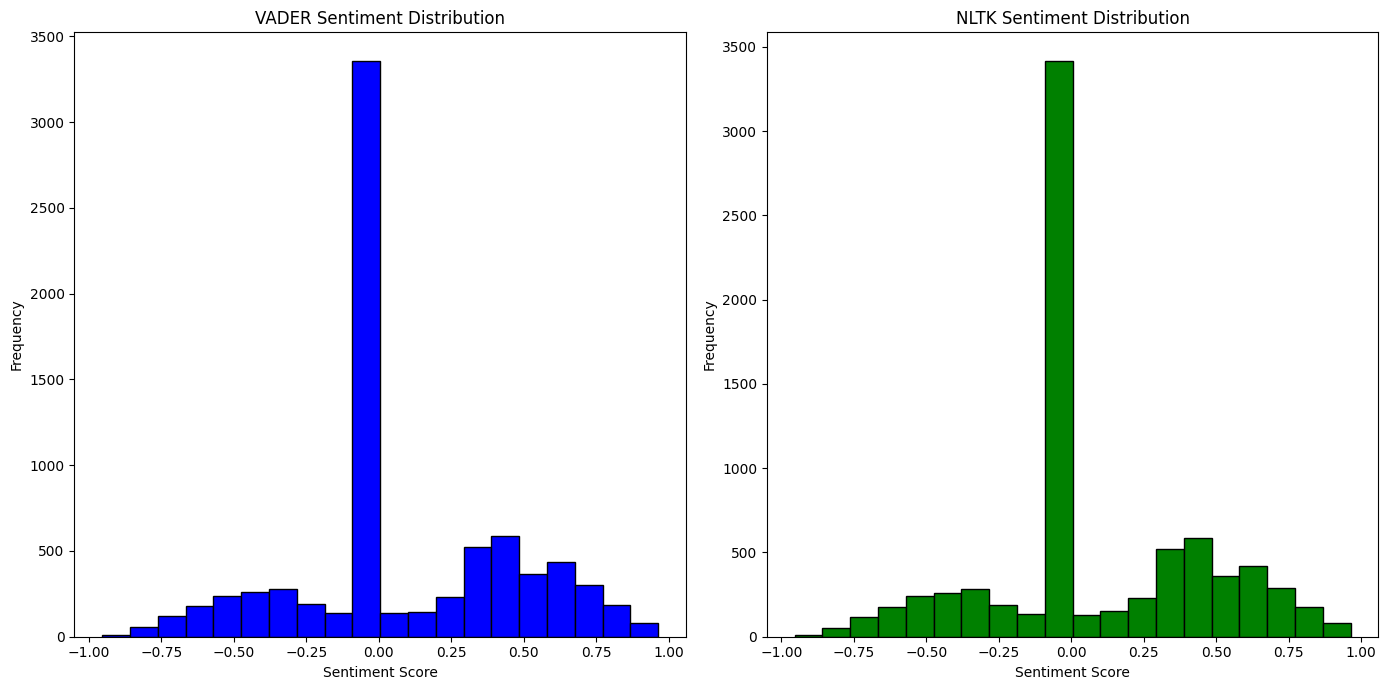


Fig.8- VADER vs NLTK Sentiment Distribution

**Standardized Daily Sentiment and Number of Tweets Over Time:** Daily sentiment was plotted as a green line and the daily number of tweets was plotted as a blue bar chart. The results are normalized and overlaid on a plot. We can see that they do not follow the same distribution and there is no evidence of a lagging correlation, indicating there may not be any correlation between the two.

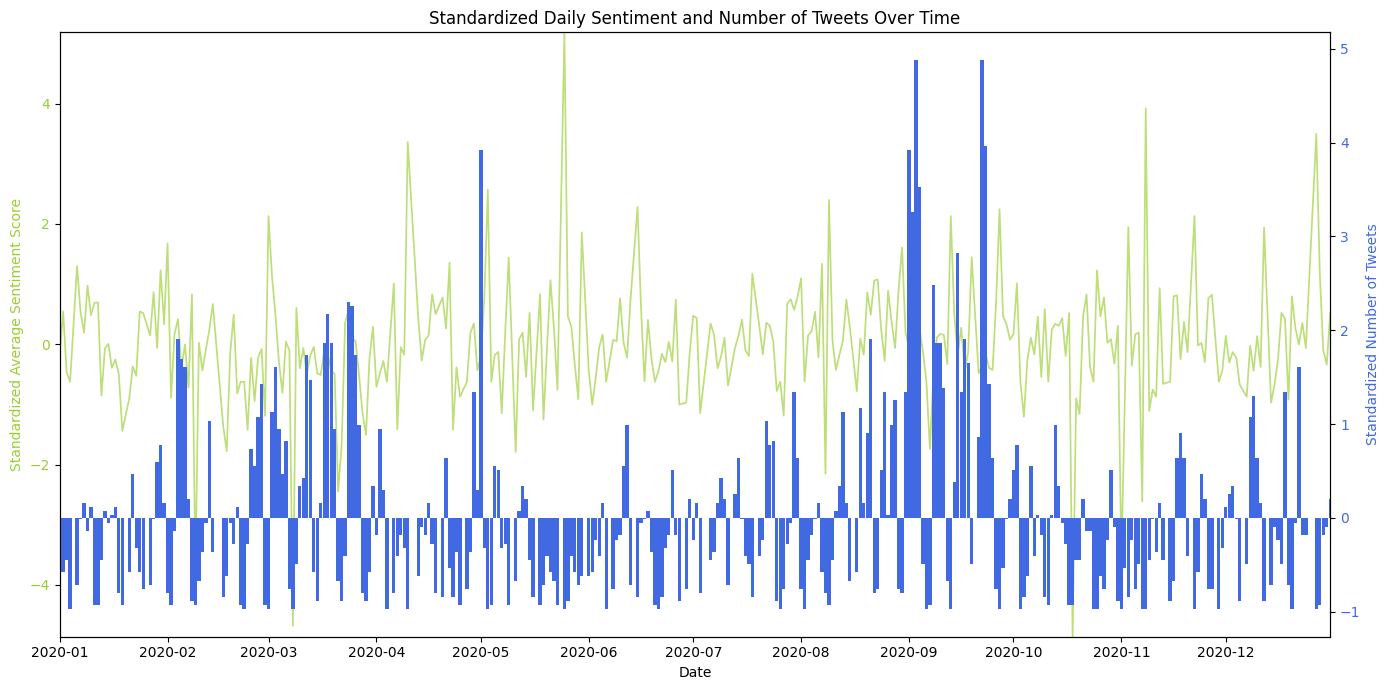


Fig.9- Standardized Daily Sentiment and Number of Tweets Over Time

**Word Cloud Analysis:** Next, we wanted to test the hypothesis that if a stock is mentioned multiple times during a period, it must correlate with a strong rise or fall within the stock. To do this, we checked the word cloud for the three highest tweeted days and found TSLA, AAPL, ELON, WILL, BULL as the most represented words, suggesting the tweets on those days are forecasting bullish behaviors within the market.

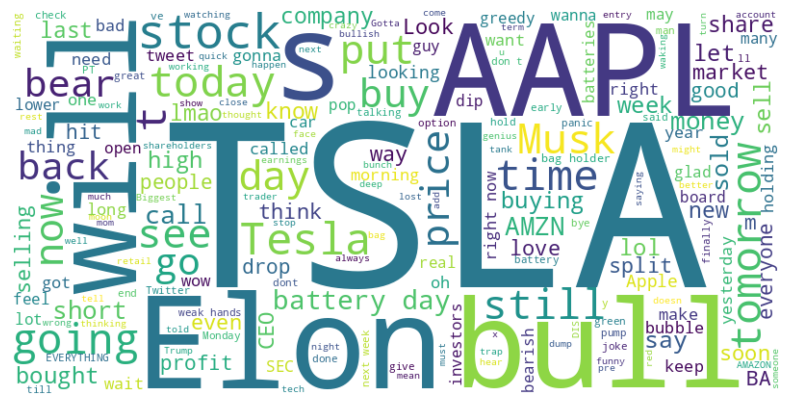


Fig.10- Word Cloud

## Close Prices and Sentiment Over Time:

**TSLA and AAPL Close Prices and Sentiment Scores:** The close prices of AAPL and TSLA were plotted alongside their sentiment scores. What is expected is that on those three days we should see some indication of the close price following the trend of sentiment, but it is not apparent. There doesn’t appear to be any lagging trend either.

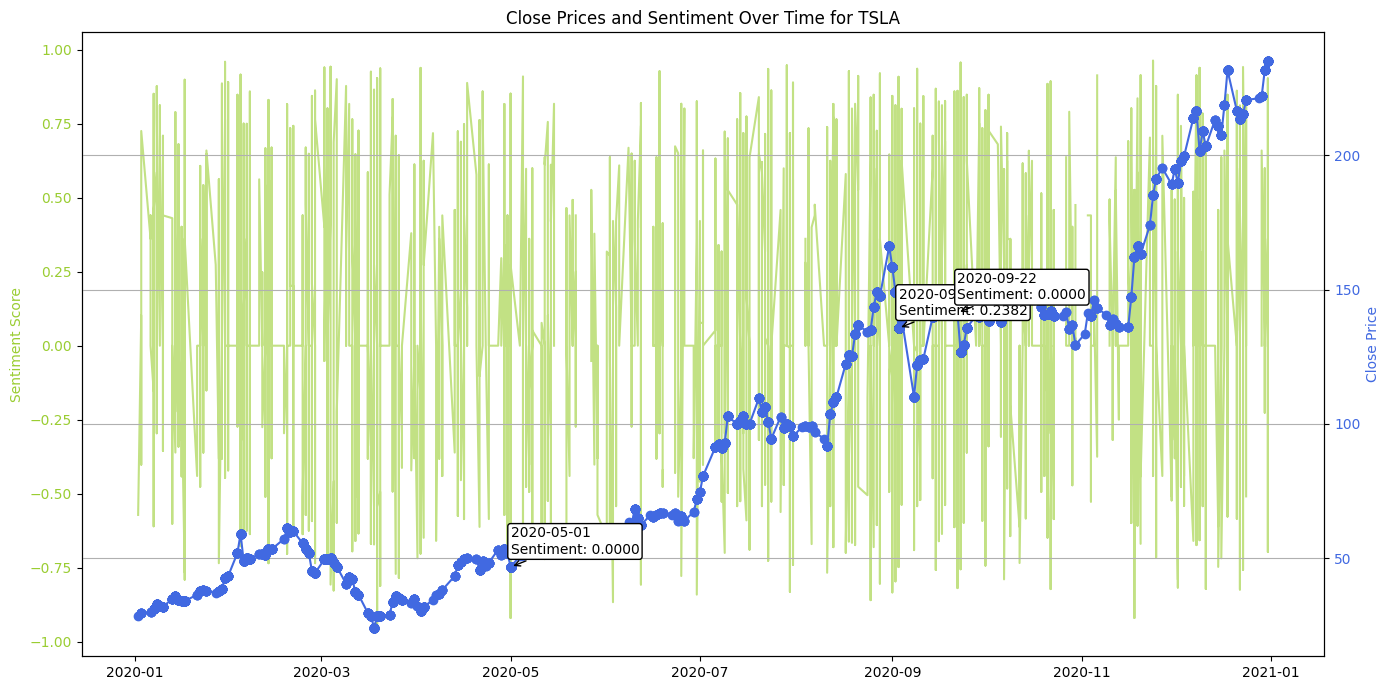
**

Fig.11- Close Prices Compared to Sentiment Over Time for TSLA

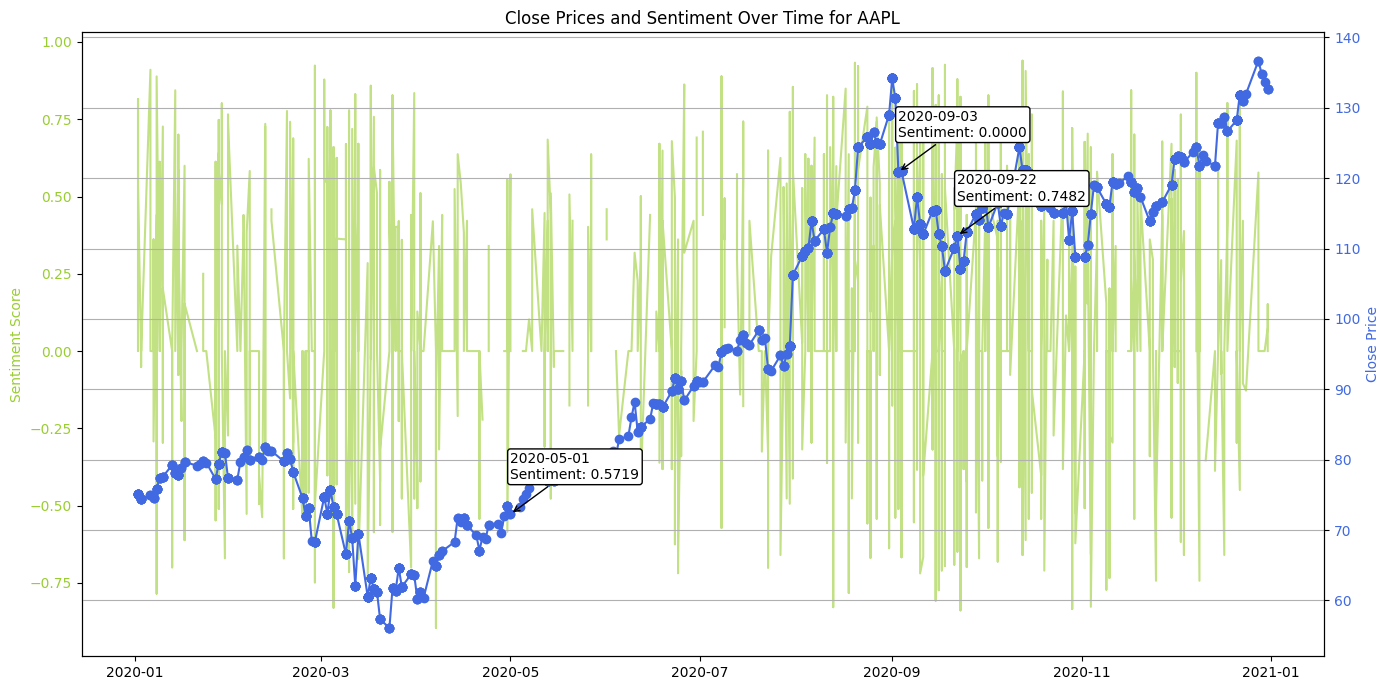
**

Fig.12- Close Prices Compared to Sentiment Over Time for AAPL

# Time Series Forecasting

# ARIMA Model

ARIMA was selected as the time series forecasting method based on its merits. ARIMA is known for its effectiveness in handling univariate time series data, particularly in financial forecasting. It captures autocorrelations within the data, making it suitable for predicting future stock prices based on historical trends.

ADFuller and ACF testing was performed on the data. The ACF plots for the time series reveal a high autocorrelation at lag 1, indicating a strong correlation between consecutive values, which is common in financial data. The autocorrelation coefficients decrease gradually but remain significant over many lags, suggesting non-stationarity and a strong trend component. Significant autocorrelations beyond the first few lags imply that past values have a prolonged influence on future values. The slow decay in the ACF plot indicates the need for differencing to achieve stationarity by removing the trend component.

The data was aggregated to group stock data by date and calculate the mean sentiment score, as there can be multiple tweets in a single day, and we want to reduce it to the average sentiment score of a given day. The dataset was split into a training and validation set. The last 7 days were stored in the validation set and the rest of the data was provided to the model. Since this is a short time series, we want to provide as much data to the model as possible. Stepwise was used to determine the optimal order for ARIMA. Then the ARIMA model was imported from the statsmodels library and trained using the training data with optimal hyperparameters applied. The model then forecasted 7 days out from the end of the training data. The results are saved and stored to be presented later in the dashboard.

## SARIMAX Model

In order to use sentiment scores, SARIMAX was adopted for its ability to work with exogenous data. Much in the same way as the ARIMA model setup, SARIMAX performs hyperparameter tuning except it utilizes grid search due to its ability to evaluate all combinations, ensuring that the global optimum is found within the specified range, even if that comes at the cost of extra computational complexity. A best order is determined for each individual stock. SARIMAX is then trained using optimal hyperparameters based on the stock. Results are saved for the dashboard.

## 

## LSTM Model

LSTM (Long Short-Term Memory) networks are a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. They are particularly effective at handling the vanishing gradient problem that can affect traditional RNNs, making them suitable for tasks involving long sequences, such as time series forecasting and natural language processing. The LSTM model is imported from the TensorFlow library along with Keras. MinMax scaling is applied to the data.

MinMax Scaling: MinMax scaling transforms the data to fit within a specific range, usually [0, 1]. The formula for MinMax scaling is:

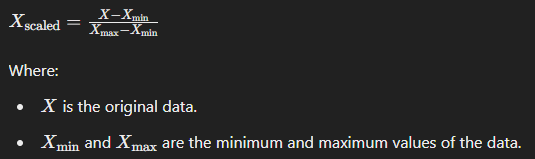


Fig.13- MinMax Formula

MinMax scaling is crucial when working with LSTM networks for several reasons, including stabilizing the training process, ensuring consistent scale for features, and optimizing the performance of activation functions. By normalizing the input data, we can improve the efficiency and accuracy of LSTM models.

**Sequences** are created for LSTM models to capture the temporal dependencies and patterns inherent in sequential data like time series. By feeding fixed-length sequences of past observations into the LSTM, the model can learn to predict future values based on historical context, leading to more accurate and meaningful forecasts. In this project, we opted for a sequence length of 7, but there is room for improvement of the LSTM model with adjustments to the sequence length.

Hyperparameter tuning of the LSTM model was done with Keras Tuner, an open-source library for hyperparameter tuning in Keras models. It helps automate the process of searching for the best set of hyperparameters for a machine learning model, which can significantly improve model performance. The LSTM model was then trained on the training data adjusted for the best hyperparameters. The model then forecasted the next 7 days, which were stored for dashboard presentation.

# Using Streamlit and Ngrok for Hosting Dashboards

For this project, Streamlit was utilized to develop an interactive dashboard that visualizes and forecasts stock prices based on sentiment analysis derived from Twitter data. To host this dashboard and make it accessible for public interaction, Ngrok was employed to tunnel the local Streamlit server to a public URL. This approach enabled easy sharing and demonstration of the project's findings, ensuring that the dashboard was accessible to external users for review and interaction. This setup not only enhanced the presentation of the project's results but also facilitated real-time interaction and feedback.

**Streamlit** is an open-source Python library that facilitates the creation and sharing of data applications and dashboards. It enables the development of interactive web applications directly from Python scripts, eliminating the need for extensive knowledge of web development frameworks.

**Ngrok** is a tool designed to create secure tunnels to localhost, thereby allowing local web servers to be exposed to the internet. This tool is particularly useful for testing webhooks, developing APIs, and sharing local applications with others.

Edward Tufte, a pioneer in the field of data visualization, articulated several principles for the effective presentation of quantitative information. These principles aim to enhance clarity, precision, and efficiency in the visual representation of data. Here’s how these principles are applied in the stock price forecasting dashboard:

**1. Graphical Excellence**

The dashboard presents stock price forecasts using interactive line charts and tables. Trends in stock prices are clearly communicated by juxtaposing forecast data with actual data.

**2. Data-Ink Ratio**

The dashboard maintains a high data-ink ratio by using clean, simple charts without unnecessary elements, ensuring the focus remains on the data.

**3. Avoidance of Chartjunk**

By avoiding unnecessary decorations and focusing on a minimalist design, the dashboard enhances readability and interpretability.

**4. Multi-Variate Data**

Users can select different stocks and models (e.g., ARIMA, SARIMAX, LSTM), allowing for analysis of multiple dimensions and providing comprehensive insights.

**5. Integration of Evidence**

The dashboard seamlessly integrates textual descriptions, numerical data, and visual charts. Interactive elements update dynamically, combining multiple forms of evidence for coherent storytelling.

**6. Context Provision**

Historical data is shown alongside forecasts, with clear labels and legends providing necessary context for accurate interpretation.

The stock price forecasting dashboard effectively embodies Tufte’s principles by prioritizing clarity, minimizing non-essential elements, integrating multi-dimensional data, and providing context. This results in an intuitive, efficient, and informative user experience.

# Evaluation of ARIMA and SARIMAX Models for Stock Market Forecasting Using Sentiment Analysis

## ARIMA Model Evaluation

The ARIMA (AutoRegressive Integrated Moving Average) model was used to forecast the stock prices for AAPL, AMZN, BA, DIS, and TSLA based on historical data and sentiment analysis. The forecasts were generated for 1 day, 3 days, and 7 days ahead. Below are the evaluations for each stock:

**1. AAPL (Apple Inc.)**

* 1 Day Forecast: Forecasted Close Price: 128.194316, Real Close Price: 131.880005
* 3 Day Forecast: Forecasted Close Price: 128.360479, Real Close Price: 131.970001
* 7 Day Forecast: Forecasted Close Price: 128.302656, Real Close Price: 132.690002

The ARIMA model underestimates the close prices slightly, indicating a need for further tuning or incorporating more recent data.

**2. AMZN (Amazon.com, Inc.)**

* 1 Day Forecast: Forecasted Close Price: 160.282445, Real Close Price: 160.326004
* 3 Day Forecast: Forecasted Close Price: 160.538469, Real Close Price: 158.634506
* 7 Day Forecast: Forecasted Close Price: 160.503596, Real Close Price: 162.846497

The forecasts for AMZN show that the model can capture the trend closely, though the 3-day forecast has a noticeable deviation.

**3. BA (Boeing Company)**

* 1 Day Forecast: Forecasted Close Price: 219.159105, Real Close Price: 218.779999
* 3 Day Forecast: Forecasted Close Price: 219.118173, Real Close Price: 217.149994
* 7 Day Forecast: Forecasted Close Price: 219.118621, Real Close Price: 214.059998

BA's forecasts are quite close to the actual values, suggesting good model performance.

**4. DIS (The Walt Disney Company)**

* 1 Day Forecast: Forecasted Close Price: 173.045206, Real Close Price: 170.449997
* 3 Day Forecast: Forecasted Close Price: 172.854381, Real Close Price: 173.729996
* 7 Day Forecast: Forecasted Close Price: 171.306367, Real Close Price: 181.179993

The 7-day forecast for DIS shows a significant deviation, highlighting potential model limitations over longer horizons.

**5. TSLA (Tesla, Inc.)**

* 1 Day Forecast: Forecasted Close Price: 218.785972, Real Close Price: 213.446671
* 3 Day Forecast: Forecasted Close Price: 217.942600, Real Close Price: 220.589996
* 7 Day Forecast: Forecasted Close Price: 218.096972, Real Close Price: 235.223328

The ARIMA forecasts show varying degrees of accuracy, with some close alignment to the actual values, particularly in the 1-day forecasts. However, discrepancies increase in the longer-term forecasts. TSLA forecasts display significant underestimation, especially in the 7-day forecast, indicating the model's struggle with TSLA's volatility.

## SARIMAX Model Evaluation

The SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous factors) model extends ARIMA by incorporating external variables such as sentiment scores. Here are the evaluations:

**1. AAPL (Apple Inc.)**

* 1 Day Forecast: Forecasted Close Price: 127.061510, Real Close Price: 131.880005
* 3 Day Forecast: Forecasted Close Price: 128.172292, Real Close Price: 131.970001
* 7 Day Forecast: Forecasted Close Price: 128.245805, Real Close Price: 132.690002

The SARIMAX model shows a closer fit compared to ARIMA, especially in the short term.

**2. AMZN (Amazon.com, Inc.)**

* 1 Day Forecast: Forecasted Close Price: 159.117675, Real Close Price: 160.326004
* 3 Day Forecast: Forecasted Close Price: 157.714749, Real Close Price: 158.634506
* 7 Day Forecast: Forecasted Close Price: 159.722711, Real Close Price: 162.846497

The inclusion of sentiment data appears to improve the forecast accuracy slightly over ARIMA.

**3. BA (Boeing Company)**

* 1 Day Forecast: Forecasted Close Price: 217.002917, Real Close Price: 218.779999
* 3 Day Forecast: Forecasted Close Price: 216.969288, Real Close Price: 217.149994
* 7 Day Forecast: Forecasted Close Price: 219.149092, Real Close Price: 214.059998

The SARIMAX model performs well, with minor deviations.

**4. DIS (The Walt Disney Company)**

* 1 Day Forecast: Forecasted Close Price: 172.936950, Real Close Price: 170.449997
* 3 Day Forecast: Forecasted Close Price: 173.529927, Real Close Price: 173.729996
* 7 Day Forecast: Forecasted Close Price: 173.409200, Real Close Price: 181.179993

The SARIMAX model improves short-term forecasts but still struggles with longer-term predictions.

**5. TSLA (Tesla, Inc.)**

* 1 Day Forecast: Forecasted Close Price: 217.841723, Real Close Price: 213.446671
* 3 Day Forecast: Forecasted Close Price: 214.916437, Real Close Price: 220.589996
* 7 Day Forecast: Forecasted Close Price: 215.076338, Real Close Price: 235.223328

SARIMAX models demonstrate similar performance to ARIMA, with a slight improvement in capturing the nuances of stock price movement, likely due to the inclusion of sentiment analysis data. TSLA's inherent volatility remains a challenge for accurate long-term forecasting.

## LSTM Forecasts

LSTM models, which are deep learning models capable of capturing long-term dependencies in sequential data, were also used for forecasting.

**1. AAPL Forecast Results:**

* 1-Day Forecast: 131.72280 (Actual: 131.880005)
* 3-Day Forecast: 130.53203 (Actual: 131.880005)
* 7-Day Forecast: 128.22151 (Actual: 131.880005)

LSTM forecasts for AAPL show strong short-term accuracy but tend to deviate more as the forecast horizon extends, reflecting the volatility of stock prices.

**2. AMZN Forecast Results:**

* 1-Day Forecast: 164.24757 (Actual: 160.326004)
* 3-Day Forecast: 163.65126 (Actual: 160.326004)
* 7-Day Forecast: 162.55623 (Actual: 164.197998)

LSTM forecasts for AMZN exhibit a reasonable alignment with actual prices in the short term, with increasing divergence over longer periods.

**3. BA Forecast Results:**

* 1-Day Forecast: 214.02713 (Actual: 218.779999)
* 3-Day Forecast: 209.63358 (Actual: 219.690002)
* 7-Day Forecast: 203.63580 (Actual: 216.250000)

LSTM forecasts for BA indicate some underestimation of prices, particularly noticeable in the 7-day forecast.

**4. DIS Forecast Results:**

* 1-Day Forecast: 180.93234 (Actual: 170.449997)
* 3-Day Forecast: 180.71890 (Actual: 173.729996)
* 7-Day Forecast: 179.45671 (Actual: 181.169998)

LSTM forecasts for DIS suggest an overestimation of prices in the short term, with closer alignment over a 7-day horizon.

**5. TSLA Forecast Results:**

* 1-Day Forecast: 228.39586 (Actual: 213.446671)
* 3-Day Forecast: 221.01201 (Actual: 213.446671)
* 7-Day Forecast: 210.79778 (Actual: 213.446671)

LSTM forecasts for TSLA highlight a consistent overestimation, particularly in the short term, indicating the challenges in predicting highly volatile stock prices.

## Conclusion

In conclusion, the ARIMA, SARIMAX, and LSTM models each offer unique strengths in forecasting stock prices. ARIMA and SARIMAX models benefit from their statistical foundation and ability to incorporate exogenous variables, while LSTM models leverage deep learning to capture complex patterns in sequential data. The inclusion of sentiment analysis data enhances the predictive capability of SARIMAX models, though all models exhibit challenges in long-term forecasting due to the inherent volatility and unpredictability of stock markets.