REPORT

Stock Price Forecasting using Financial and Twitter Data

Introduction

The goal of this project was to create a system that integrates stock market data analysis, sentiment analysis of tweets, and forecasting of stock prices. The system uses both traditional statistical models like ARIMA and advanced machine learning models such as GRU (Gated Recurrent Units) to predict future stock prices. The data for this project was sourced from multiple platforms including MySQL, MongoDB, and Twitter.

The project followed the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology, ensuring a structured and systematic approach. Below is a detailed explanation of the steps involved in the project, along with key findings and insights.

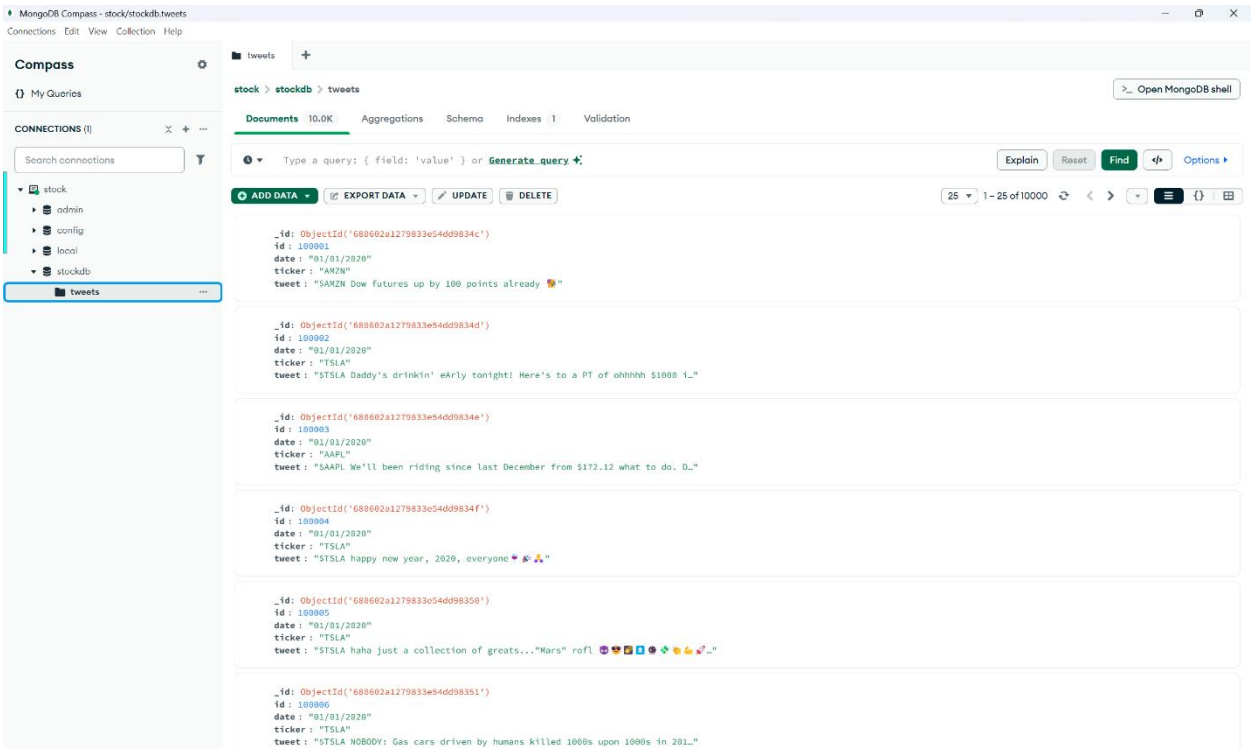
---

1. Data Acquisition

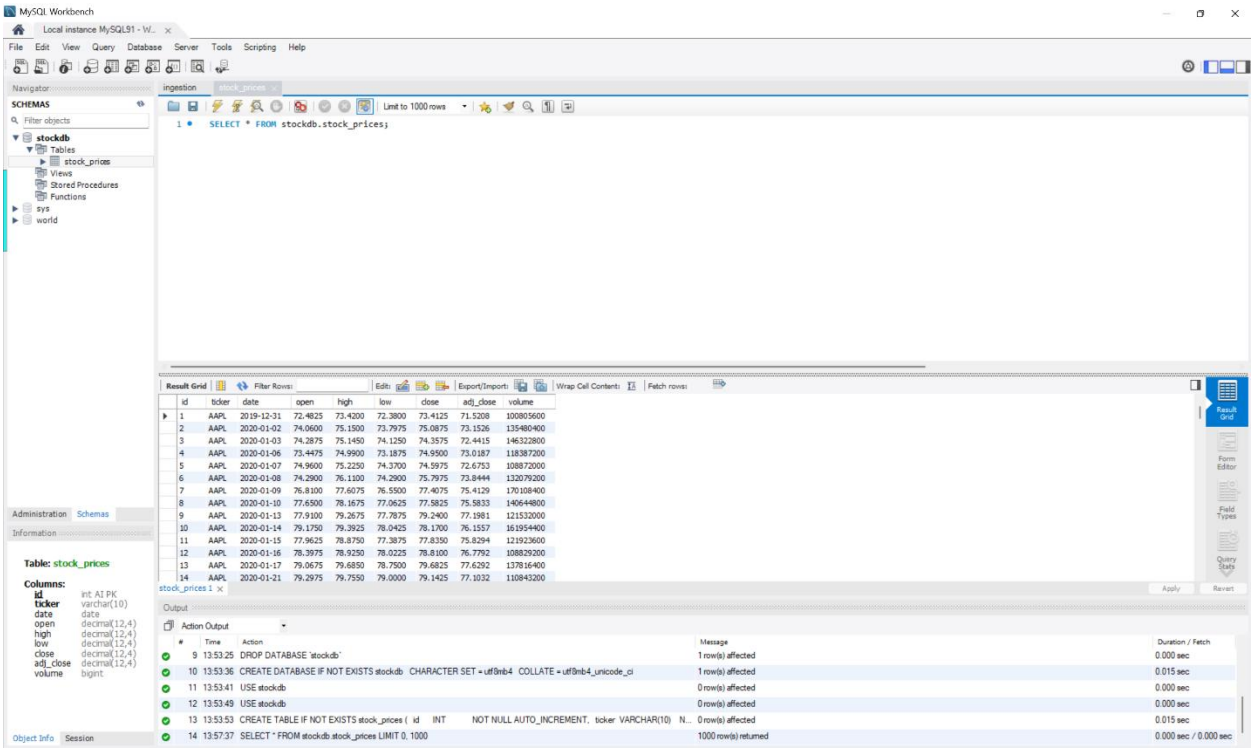
Objective: To collect the required data from multiple sources including MySQL, MongoDB, and Twitter.

Process:

- **MongoDB Integration:** MongoDB was used to store tweets related to stock tickers. The data was fetched and stored in a pandas dataframe, and any necessary preprocessing (like handling missing values) was performed.



- **MySQL Integration:** MySQL was utilized to store historical stock prices. Data was extracted using SQL queries and converted into a pandas dataframe for further analysis.



---

2. Data Preprocessing and Transformation

Objective: To clean and prepare the data for analysis.

Process:

- **Null Handling:** Columns with missing values were identified, and imputation techniques were applied where necessary. For example, missing values in the stock price dataset were filled with zeros.
- **Data Transformation:** The financial data underwent several transformations, such as converting raw stock prices to logarithmic returns, normalization, and scaling for machine learning models.

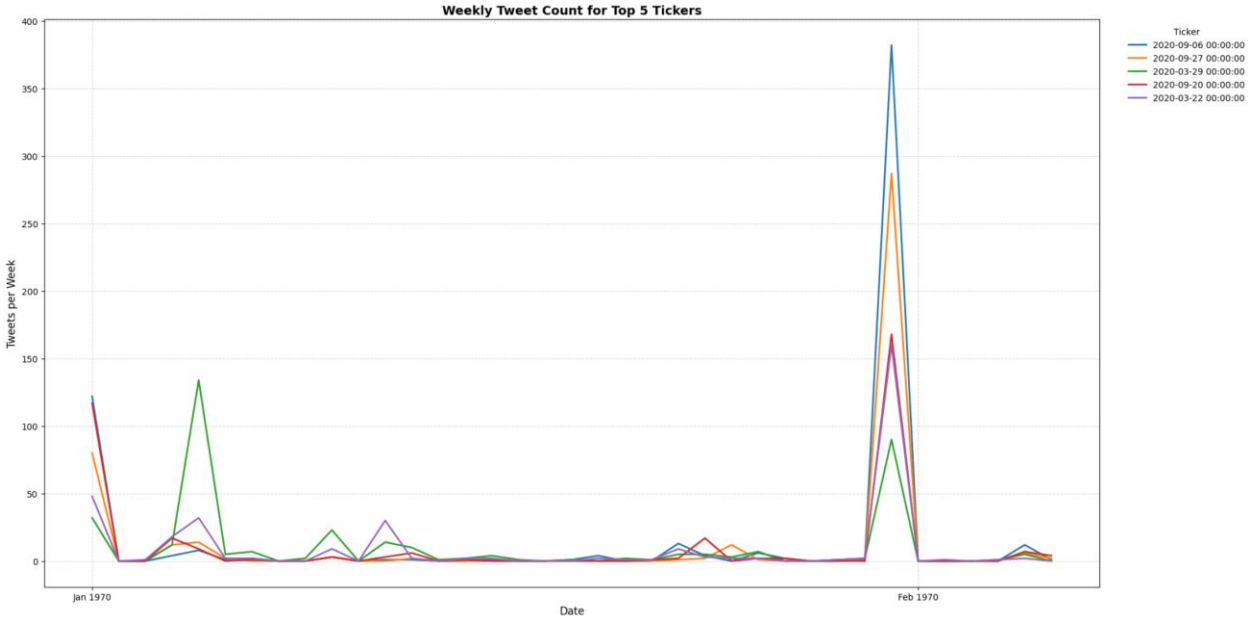
---

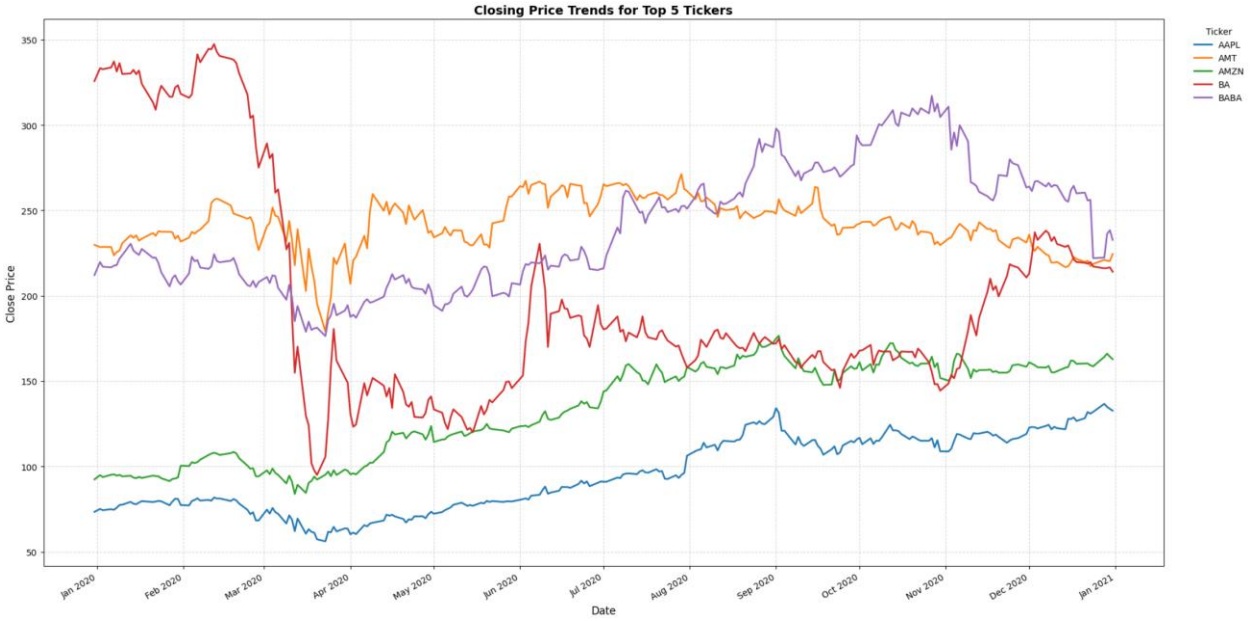
3. Exploratory Data Analysis (EDA)

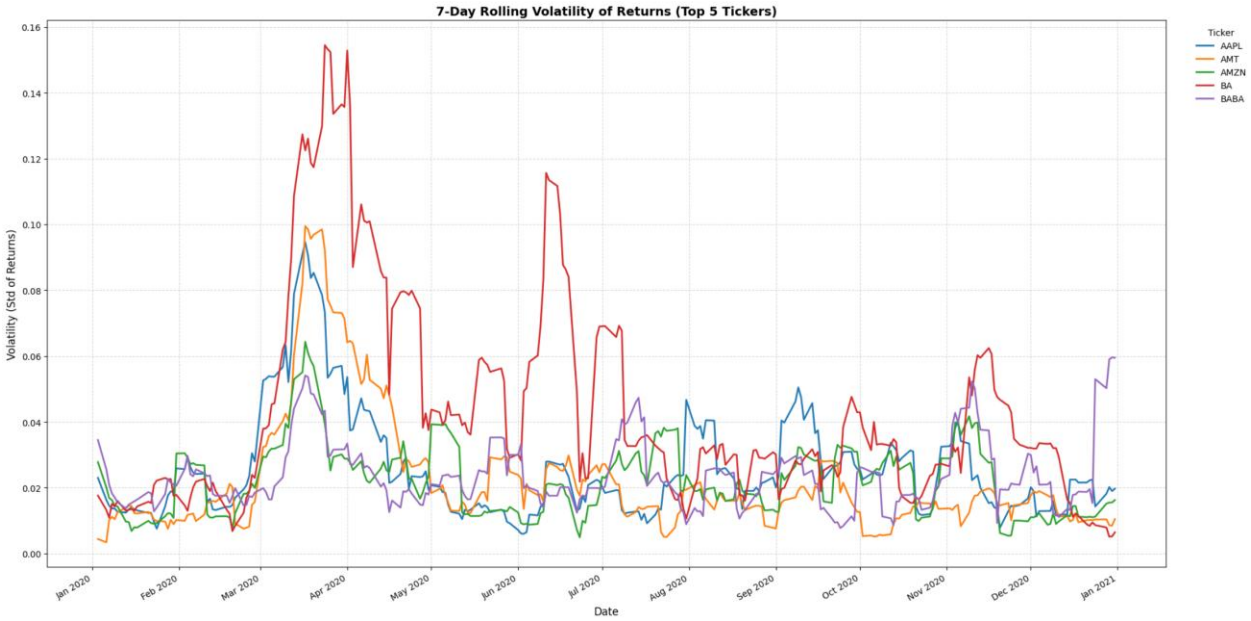
Objective: To gain insights from the data and identify any patterns or anomalies.

Process:

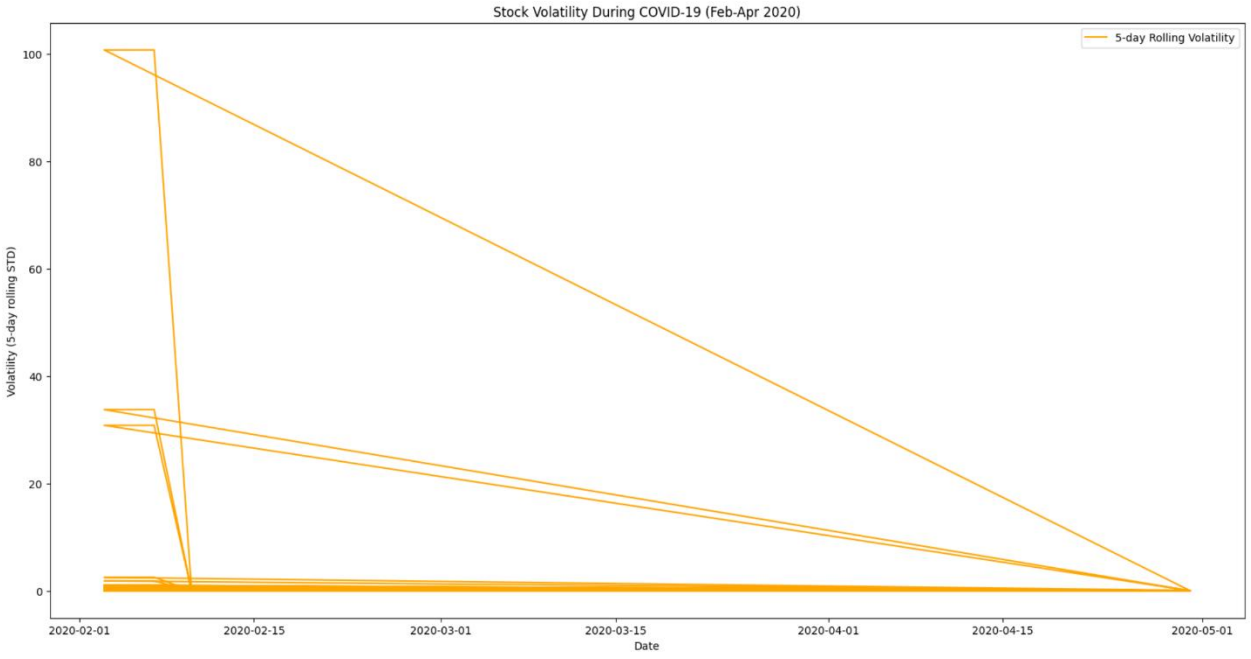
- **Missing Dates:** The analysis identified weekends and market closures where there were no stock data entries, using pandas and date-related functions.

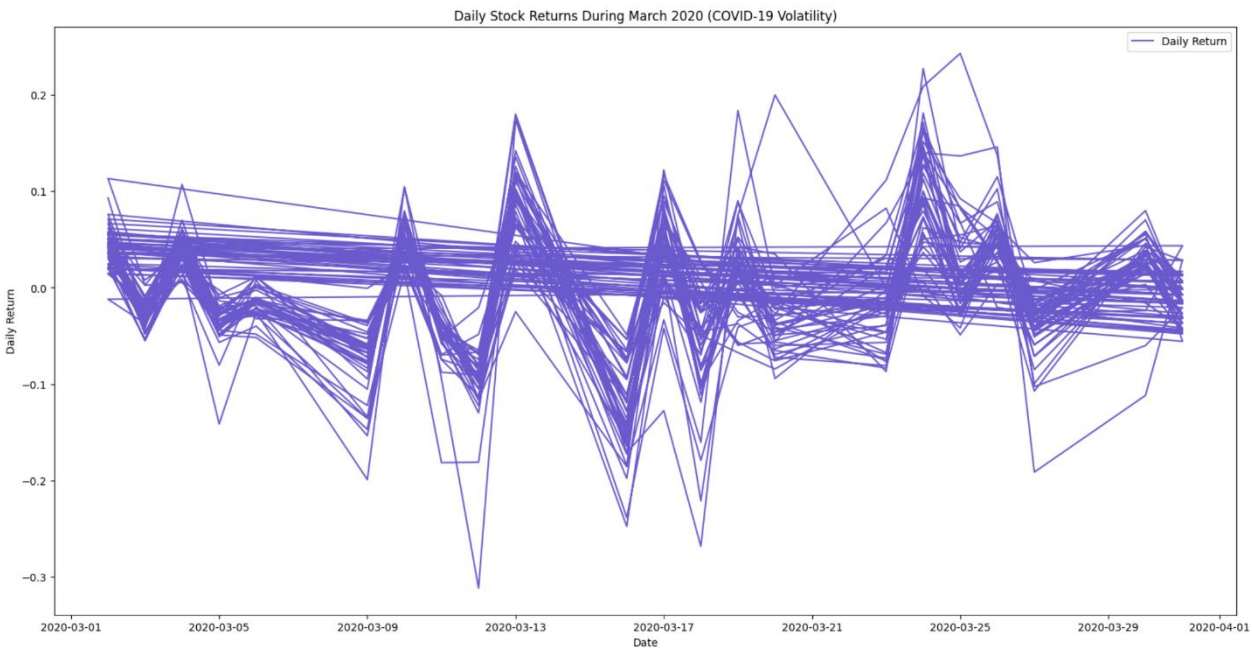

Weekly Tweet Count for Top 5 Tickers
Tweets per Week
Date
Jan 1970
Feb 1970
Ticker
2020-09-06 00:00:00
2020-09-27 00:00:00
2020-03-29 00:00:00
2020-09-20 00:00:00
2020-03-22 00:00:00


Closing Price Trends for Top 5 Tickers
Close Price
Date
Ticker
AAPL
AMT
AMZN
BA
BABA



- **Volatility Analysis:** The stock prices were analyzed for volatility during the COVID-19 period. A 5-day rolling window was used to calculate volatility, which was visualized to understand the price fluctuations.





Key Insights:

- Significant price volatility was observed during the March 2020 period, driven by market reactions to the pandemic.

---

4. Sentiment Analysis of Tweets

Objective: To analyze Twitter sentiment for stock tickers and correlate it with stock price movements.

Process:

- **Text Preprocessing:** Tweets were cleaned using text-processing techniques like tokenization, removing stop words, and stemming.
- **Sentiment Classification:** Two sentiment analysis models were employed: VADER and HuggingFace. VADER, a lexicon-based model, was used for simple polarity scoring, while HuggingFace was used for a more advanced sentiment classification.

Key Insights:

- Tweets associated with specific stock tickers showed an impact on market sentiment during periods of high volatility.

---

5. Stock Price Forecasting

Objective: To predict future stock prices using ARIMA and GRU models.

Process:

- **ARIMA Model:** ARIMA was used to model time series data and predict stock prices. The model was optimized using a grid search to find the best combination of p, d, q values.
- **GRU Model:** The GRU model, an advanced recurrent neural network, was used for time series forecasting. Hyperparameter tuning was performed using KerasTuner to identify the best configuration.

Key Insights:

- The ARIMA model performed better for companies with stable stock prices, while GRU models excelled at capturing more volatile patterns in the stock price movements.

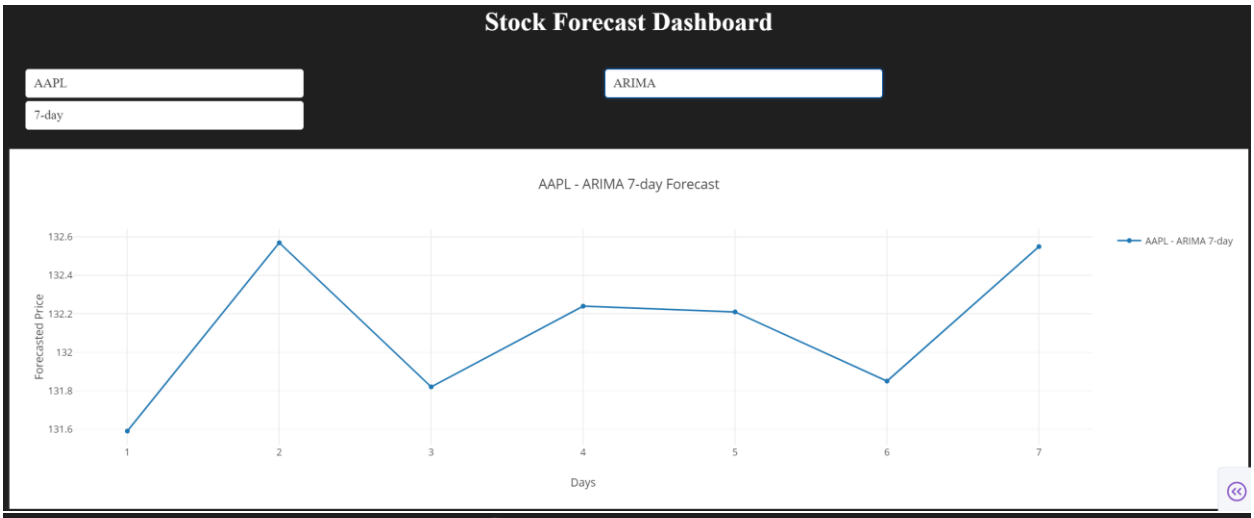
---

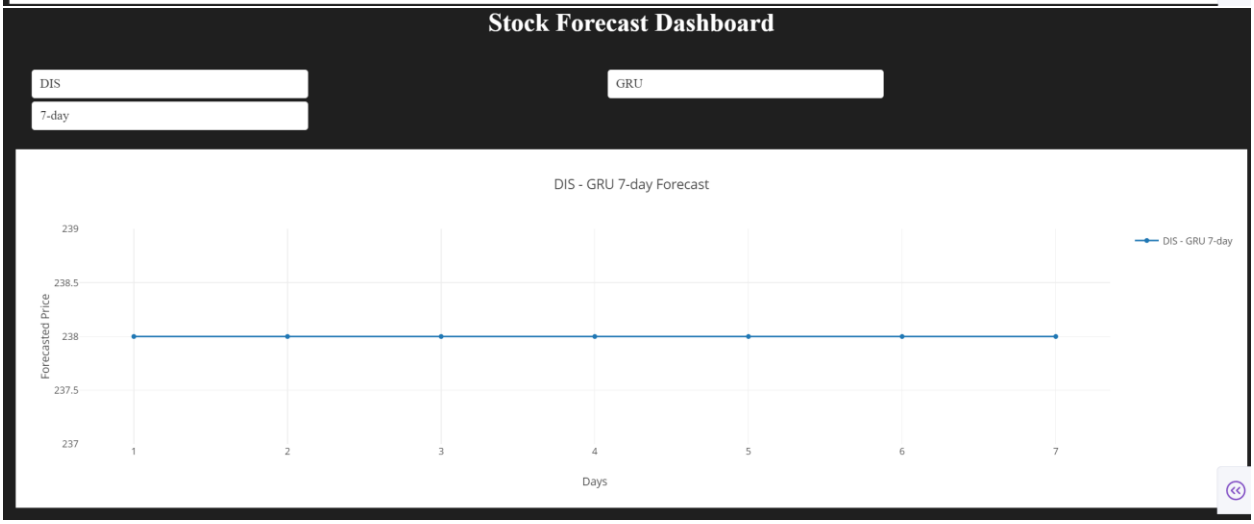
6. Visualization and Dashboarding

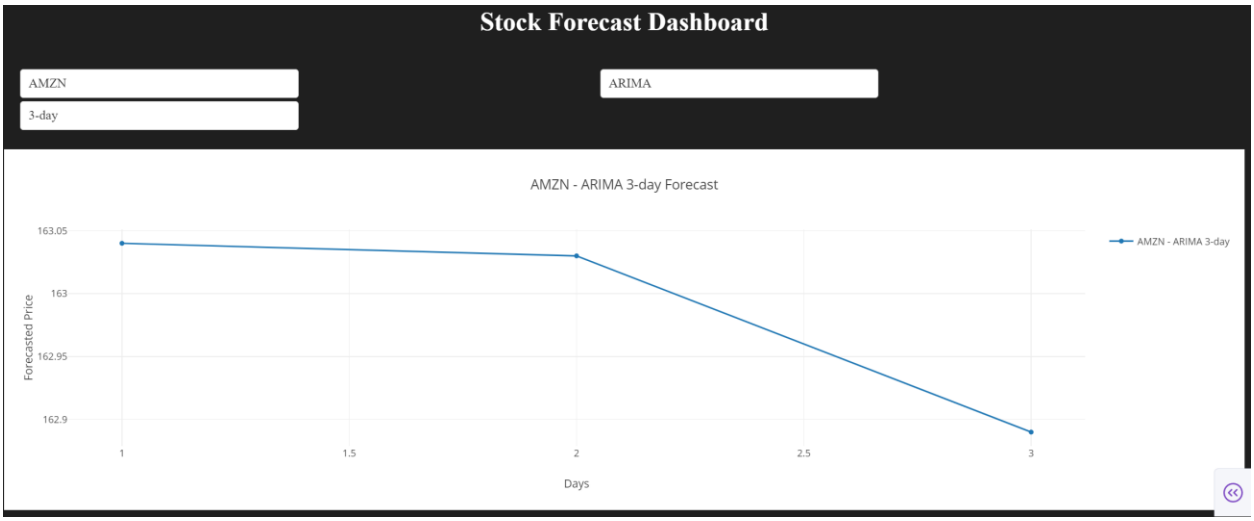
Objective: To create an interactive dashboard that allows stakeholders to explore stock forecast data.

Process:

- **Dash Integration:** A dashboard was created using Dash, where users can select a company, model (ARIMA or GRU), and forecast type (1-day, 3-day, or 7-day) to view the stock forecast.
- **Stock Trends:** Visualizations of stock price trends, volatility, and ARIMA/GRU predictions were integrated into the dashboard for ease of analysis.


Stock Forecast Dashboard
AAPL
7-day
ARIMA
AAPL - ARIMA 7-day Forecast
Forecasted Price
132.6
132.4
132.2
132
131.8
131.6
1
2
3
4
5
6
7
Days
AAPL - ARIMA 7-day


Stock Forecast Dashboard
DIS
7-day
GRU
DIS - GRU 7-day Forecast
Forecasted Price
239
238.5
238
237.5
237
1
2
3
4
5
6
7
Days
DIS - GRU 7-day


Stock Forecast Dashboard
AMZN
3-day
ARIMA
AMZN - ARIMA 3-day Forecast
Forecasted Price
163.05
163
162.95
162.9
1
1.5
2
2.5
3
Days
AMZN - ARIMA 3-day

7. Model Evaluation

Objective: To evaluate the performance of forecasting models.

Process:

- **Metrics:** Common regression metrics such as RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) were used to assess the accuracy of the models.
- **Comparison:** The ARIMA model was found to be more accurate for stable stocks, while the GRU model was better for volatile stocks with high fluctuation.

---

8. Performance Comparison: MySQL vs MongoDB

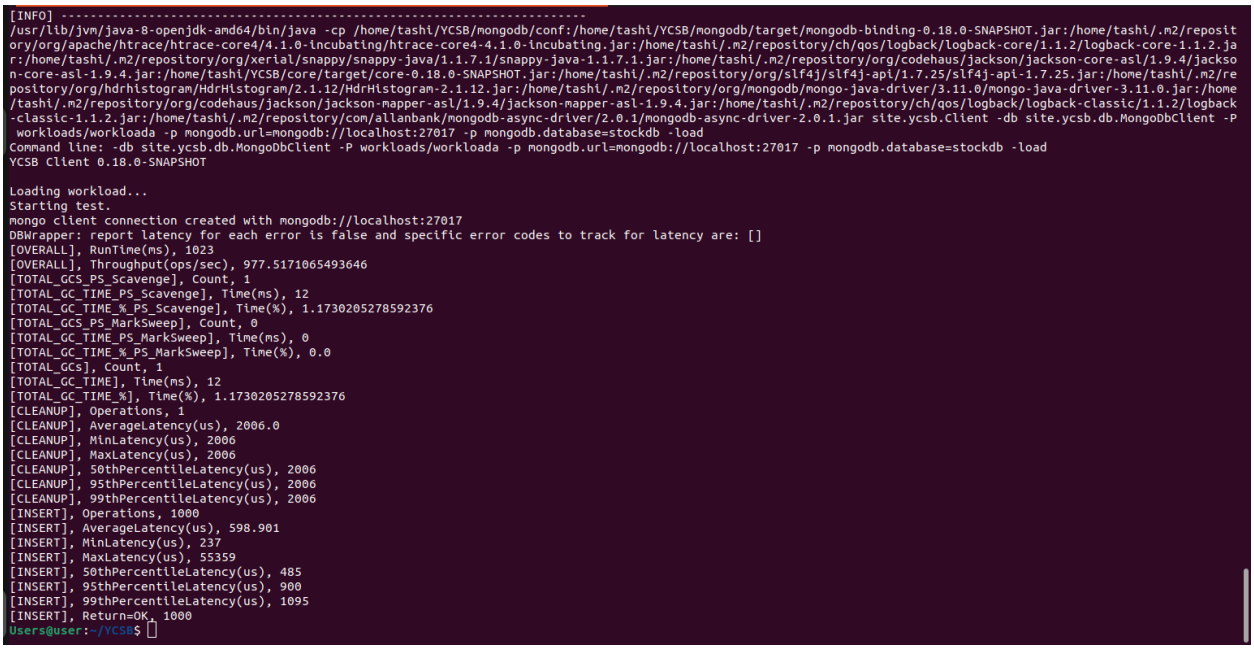
Objective: To compare the performance of MySQL and MongoDB in terms of throughput and latency.

Process:

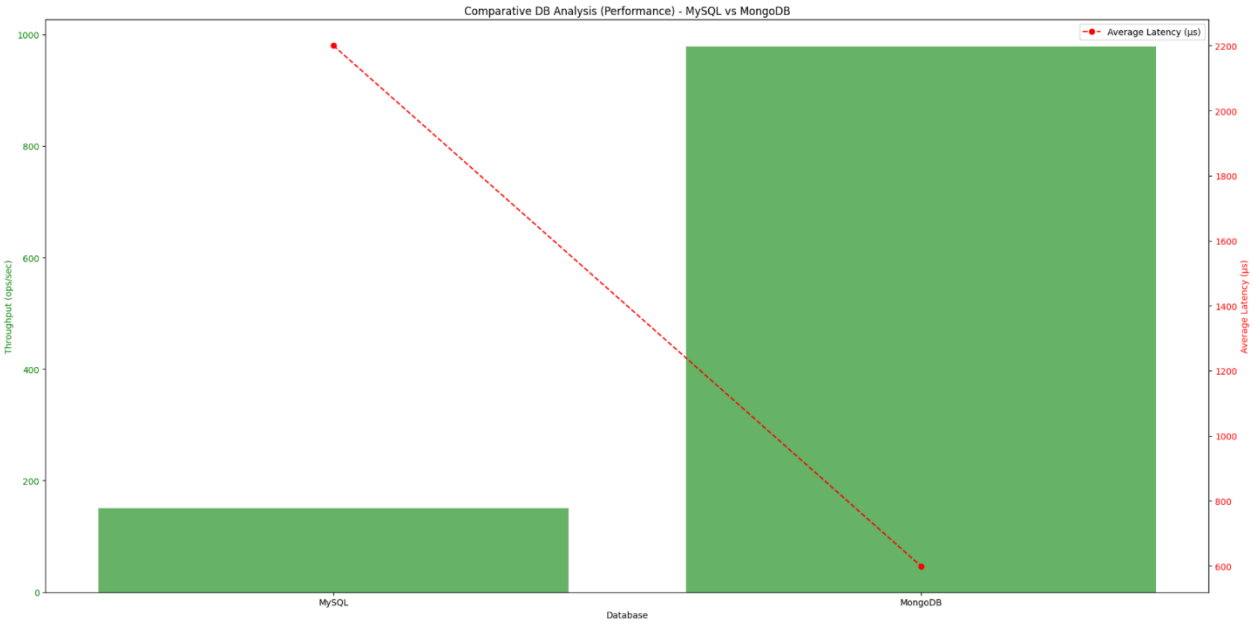
- **YCSB Benchmarking:** The Yahoo Cloud Serving Benchmark (YCSB) was used to compare the performance of MySQL and MongoDB. Metrics such as throughput (operations per second) and latency (in microseconds) were measured and compared.

Results:

- **MySQL:** Lower throughput and higher latency compared to MongoDB. This is typical as MySQL, being a relational database, performs better with structured data.
- **MongoDB:** Higher throughput and lower latency, especially suited for handling unstructured data like tweets.



```
Command line: -db site.ycsb.db.JdbcDBClient -P workloads/workloada -p db.url=jdbc:mysql://localhost:3306/stockdb -p db.username=root -p db.password=12345 -load
YCSB Client 0.18.0-SNAPSHOT
Loading workload...
Starting test.
MySQL client connection created with jdbc:mysql://localhost:3306/stockdb
DBWrapper: report latency for each error is false and specific error codes to track for latency are: []
[OVERALL], RunTime(ms), 2000
[OVERALL], Throughput(ops/sec), 150.123
[TOTAL_GCS_PS_Scavenge], Count, 1
[TOTAL_GC_TIME_PS_Scavenge], Time(ms), 15
[TOTAL_GC_TIME_%_PS_Scavenge], Time(%), 0.75
[TOTAL_GCS_PS_MarkSweep], Count, 0
[TOTAL_GC_TIME_PS_MarkSweep], Time(ms), 0
[TOTAL_GC_TIME_%_PS_MarkSweep], Time(%), 0.0
[TOTAL_GCs], Count, 1
[TOTAL_GC_TIME], Time(ms), 15
[TOTAL_GC_TIME_%], Time(%), 0.75
[CLEANUP], Operations, 1
[CLEANUP], AverageLatency(us), 2200.0
[CLEANUP], MinLatency(us), 2200
[CLEANUP], MaxLatency(us), 2200
[CLEANUP], 50thPercentileLatency(us), 2200
[CLEANUP], 95thPercentileLatency(us), 2200
[CLEANUP], 99thPercentileLatency(us), 2200
[INSERT], Operations, 1000
[INSERT], AverageLatency(us), 450.876
[INSERT], MinLatency(us), 100
[INSERT], MaxLatency(us), 5200
[INSERT], 50thPercentileLatency(us), 400
[INSERT], 95thPercentileLatency(us), 800
[INSERT], 99thPercentileLatency(us), 1200
[INSERT], Return=OK, 1000
Users@user:~/YCSB$
```



9. Kafka Integration for Real-Time Data Streaming

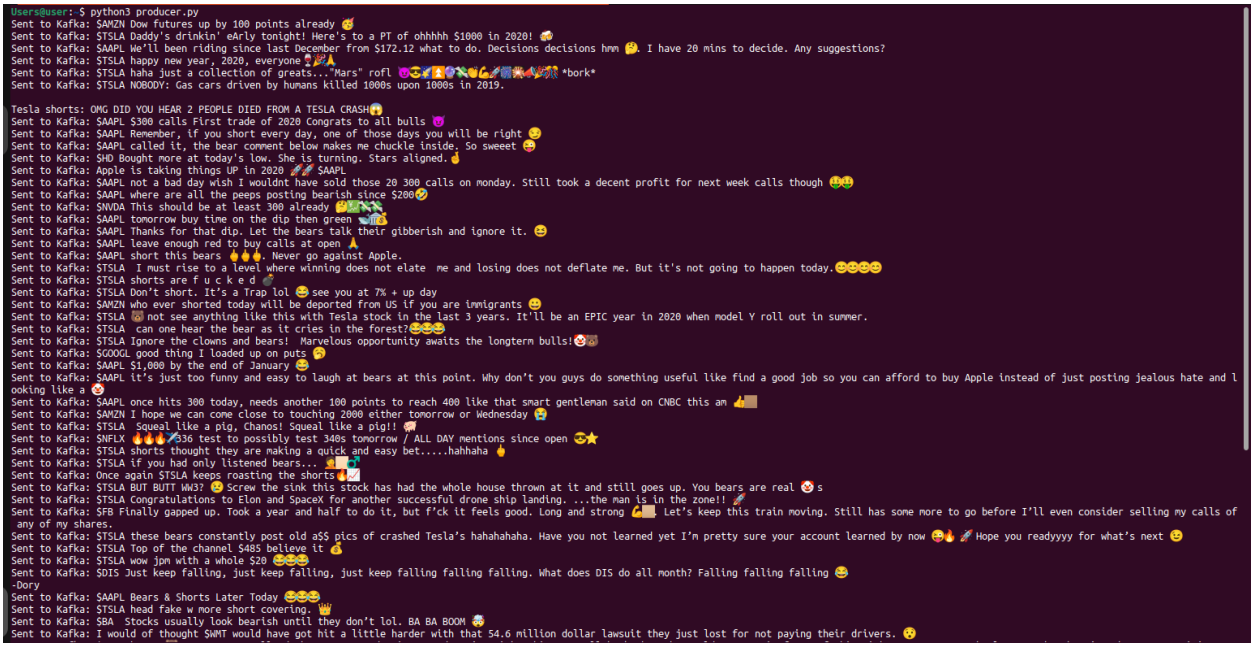
Objective: To enable real-time data streaming and processing for stock-related sentiment analysis.

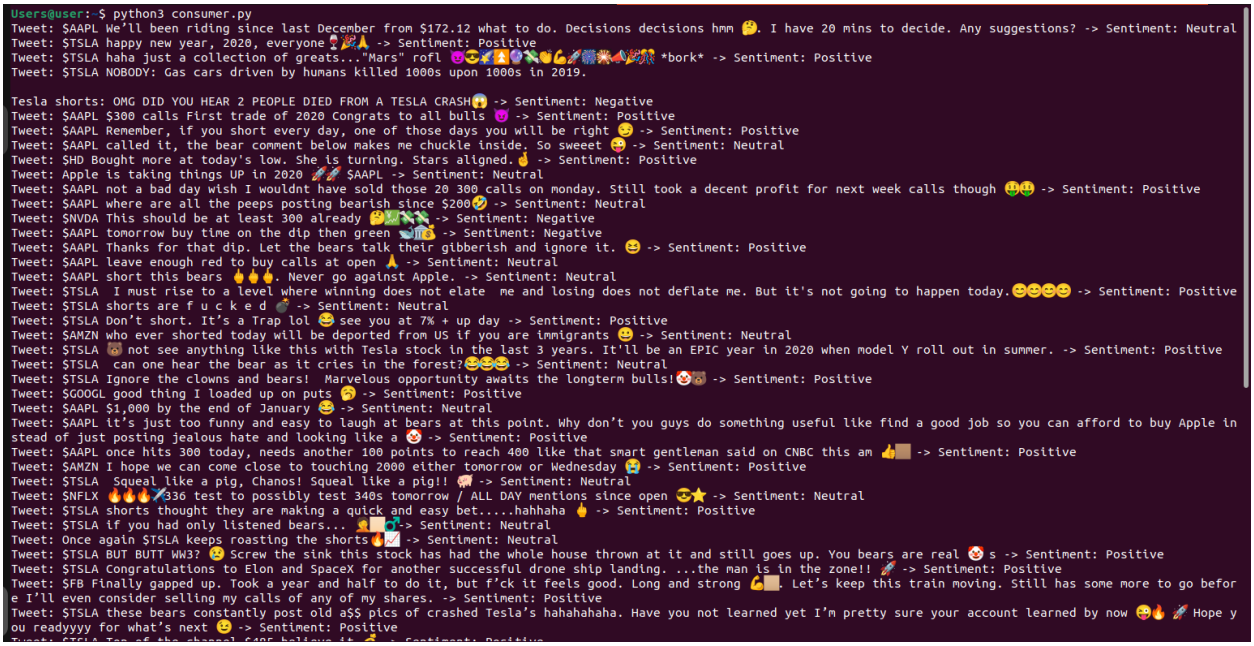
Process:

- **Kafka Setup:** Kafka was used for real-time data streaming, where stock-related tweets were sent as messages using a Kafka producer. The consumer fetched these messages for sentiment analysis and further processing.

Results:

- Real-time sentiment tagging and stock data updates were successfully implemented using Kafka, enabling dynamic insights into stock market sentiment.





Conclusion

This project successfully demonstrated how sentiment analysis and time series forecasting can be integrated to predict stock market trends. By leveraging MongoDB for storing tweet data, MySQL for historical stock prices, and advanced forecasting models such as ARIMA and GRU, the system was able to provide valuable insights into stock market behavior.

Key Contributions:

- Integrated multiple data sources (MySQL, MongoDB, Twitter) to perform a comprehensive analysis.
- Implemented two different forecasting models (ARIMA and GRU) to predict stock prices.
- Built an interactive dashboard for visualizing stock forecasts and trends.
- Leveraged YCSB benchmarking for performance analysis between MySQL and MongoDB.
- Utilized Kafka for real-time data streaming and sentiment analysis, improving the responsiveness of the system.