Stock Price Forecasting using Financial and Twitter Data

1. Data Collection

- **MongoDB**: We connected to MongoDB using MongoClient and extracted tweets data from the "stockdb" database. The data was then converted into a pandas DataFrame for easy manipulation and analysis.
- **MySQL**: We connected to MySQL using mysql.connector and retrieved stock price data from the "stockdb" database. This data was also converted into a pandas DataFrame.

Assumptions:

- Both MongoDB and MySQL databases were assumed to contain valid data for tweets and stock prices.

---

2. Data Cleaning & Preprocessing

- **Null Handling in Tweets Data**: We identified and handled missing dates in the tweets data, removing rows with missing or invalid date entries.
- **Null Handling in Prices Data**: For the stock prices data, we checked for missing values and performed data imputation for fields like return and vol_7d using zero imputation.

Assumptions:

- Missing data in tweets was assumed to be due to market closures, and the missing values in prices were filled with zero to avoid biases in further calculations.

---

3. Feature Engineering & Transformation

- We performed various feature transformations on the stock price data:
    - **Log Transformation**: Applied log1p transformation to columns such as open, high, low, close, adj_close, and volume for normalization.
    - **Min-Max Scaling**: Used Min-Max scaling to normalize the data between 0 and 1.
    - **Z-Score Transformation**: Performed Z-score standardization for each of the columns.
    - **Robust Scaling**: Applied robust scaling using the IQR method for handling outliers.

Assumptions:

- The transformations were necessary to prepare the data for better analysis and modeling, especially considering potential outliers in stock price data.

---

4. Sentiment Analysis on Tweets

- **Text Preprocessing**: We tokenized and cleaned the tweets text, removing non-alphabetic characters, converting to lowercase, and stemming the words. This was done using the nltk library.
- **Sentiment Classification**: We used two different sentiment analysis models:
    - **VADER Sentiment Analysis**: Applied the VADER sentiment analysis tool to assign sentiment scores to each tweet.
    - **Hugging Face Sentiment Analysis**: Used the Hugging Face sentiment analysis pipeline for comparison.

Assumptions:

- Tweets were assumed to reflect sentiment about specific stock companies, and we used automated tools to gauge positive, negative, or neutral sentiment based on the content of each tweet.

---

5. Time Series Forecasting

- **ARIMA**: We used the ARIMA model to predict future stock prices. The ARIMA model was selected based on a grid search for optimal hyperparameters (p, d, q) to minimize the AIC (Akaike Information Criterion).
- **GRU (Gated Recurrent Units)**: We used a GRU-based deep learning model for forecasting stock prices. The model was tuned using keras_tuner for optimal architecture.

Assumptions:

- We assumed that stock prices and tweet data can be used in conjunction to predict future stock prices and identify trends.

---

6. Evaluation of Forecast Models

- **Model Performance**: The performance of both ARIMA and GRU models was evaluated using metrics such as RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error).
- **Forecasting**: We evaluated forecasts for 1-day, 3-day, and 7-day periods, depending on the model choice and data available.

Assumptions:

- We assumed that both ARIMA and GRU models were appropriate for time series forecasting of stock prices, and their performance was evaluated based on real data.

---

7. Dashboard for Real-Time Stock Forecast

- We built a dashboard using Dash and Plotly to visualize stock forecasts dynamically. Users could select a company, model, and forecast type (1-day, 3-day, or 7-day) to view the forecasted stock prices.

Assumptions:

- The dashboard was designed for interactive exploration of the forecast results and is accessible to users in real-time.

---

8. Comparative Database Performance Analysis

- **YCSB Benchmarking**: We used YCSB (Yahoo! Cloud Serving Benchmark) to compare the performance of MySQL and MongoDB. We measured metrics like throughput (operations per second) and latency (microseconds).
- **Visualization**: A dual-axis bar and line chart was created to visualize throughput and latency for both databases.

Assumptions:

- The performance comparison assumes that the databases are configured similarly and are operating under comparable workloads.

---

9. Final Insights & Conclusions

- **Sentiment Analysis**: Sentiment analysis showed clear patterns in market sentiment related to stock movements, which can be helpful for traders and analysts.
- **Forecasting**: The ARIMA and GRU models provided different approaches for stock price prediction, and their accuracy was assessed using traditional metrics like RMSE and MAE.
- **Database Performance**: The comparative analysis of MySQL and MongoDB highlighted significant differences in throughput and latency, helping us choose the right database for future applications.

Assumptions:

- We assumed that the models and methods used in this project were the most appropriate for the dataset and business objectives.

Conclusion

This project utilized multiple machine learning and statistical techniques, including sentiment analysis, time series forecasting, and database performance comparison, to provide a comprehensive analysis of stock market trends based on historical data. The integration of different tools such as MongoDB, MySQL, ARIMA, GRU, and Dash for visualization allowed for efficient handling of large datasets and real-time predictions.