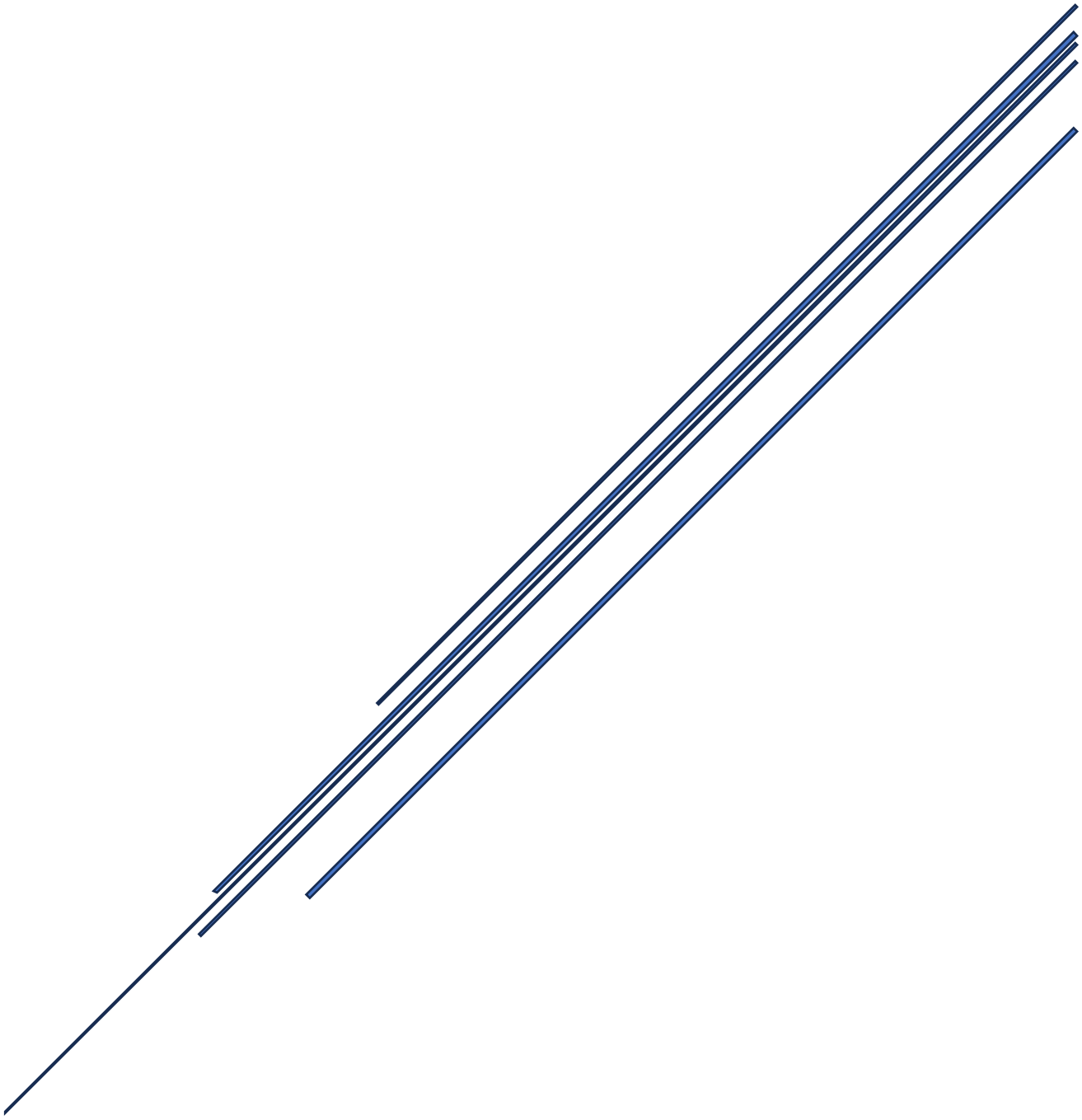
Federico Ariton

Stock Price Forecasting Using Sentiment and Big Data



Index

Big Data and Deep Learning for Stock Price Forecasting Using Sentiment Analysis

Introduction

This project presents an end-to-end predictive analytics system designed to forecast short-term stock prices using both historical market data and public sentiment from tweets (Bollen, Mao and Zeng, 2011). By combining traditional time series models (ARIMAX, SARIMAX) with deep learning models (LSTM, GRU), the system aims to enhance the accuracy of financial forecasting across multiple horizons (1, 3, and 7 days).

The architecture follows a Lambda-based design using Apache Spark for distributed processing and HDFS for storage. Cleaned and enriched data is exported to four databases (Cassandra, MongoDB, MySQL, and PostgreSQL), which are benchmarked using YCSB to identify the most efficient storage solution — MongoDB (Ieee.org, 2025).

The final predictions and evaluations are delivered through an interactive Streamlit dashboard, allowing users to explore forecast results, sentiment trends, and RMSE comparisons in real time.

Big Data

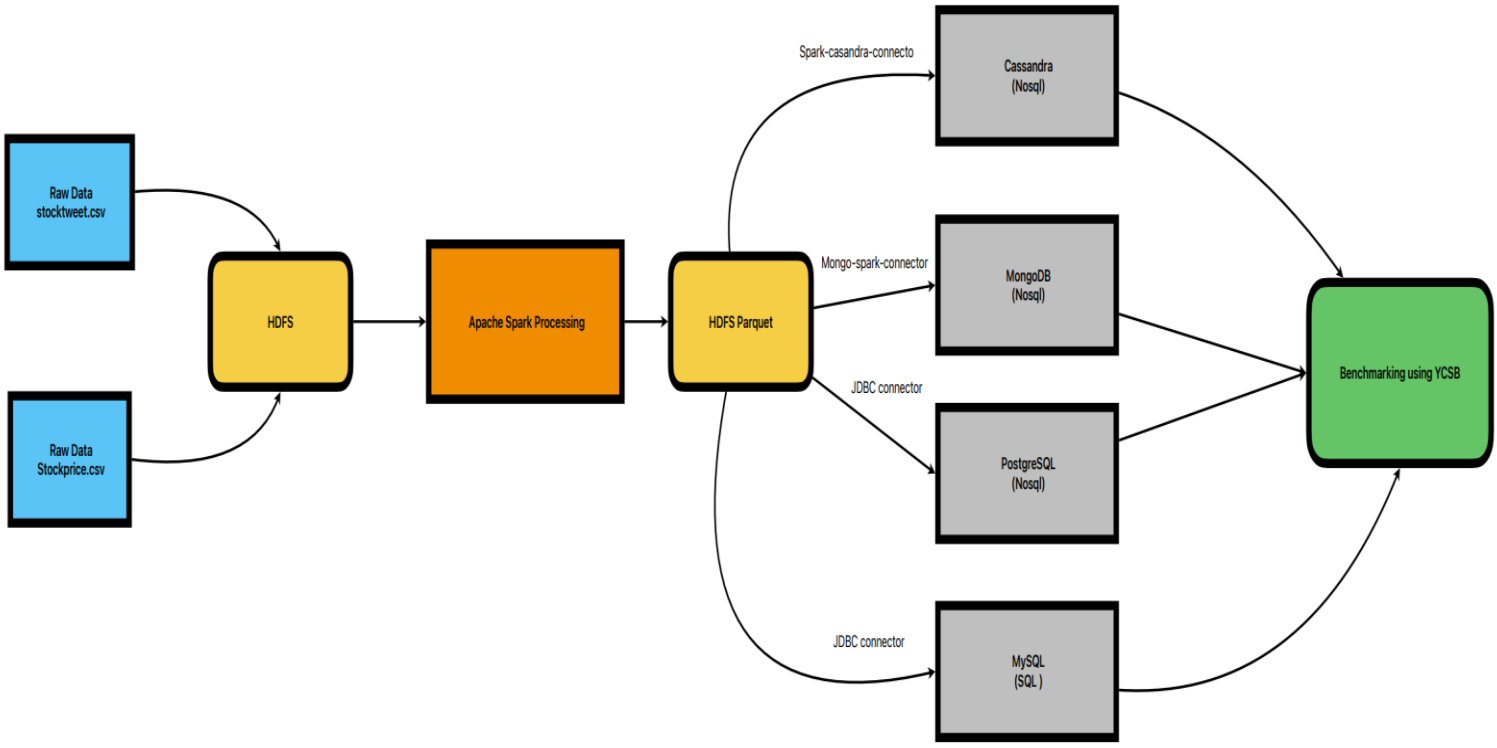


Figure 1 First Design of Bigdata Architecture

Big Data Architecture

This project adopts a Lambda architecture to efficiently process and analyze large-scale stock market and Twitter sentiment data. The architecture separates concerns between batch and real-time layers, providing flexibility, fault tolerance, and scalability (Ieee.org, 2025).

At the storage layer, all raw and processed data is stored in HDFS, enabling distributed access for high-volume processing(hadoop, n.d.). Apache Spark is used in the batch layer to transform the data computing lag features, rolling statistics, and aggregating tweet sentiment making it ready for machine learning forecasting models.

To evaluate performance and integration flexibility, processed datasets are saved into four databases: MongoDB, Cassandra, MySQL, and PostgreSQL.

Each connection was made using official Spark connectors such as:

- mongo-spark-connector for MongoDB
- spark-cassandra-connector for Cassandra
- JDBC connectors for MysSQL and PostgreSQL

Big Data Tools and technologies

The project integrates several tools from the big data ecosystem, each selected for its role in processing, storage, and evaluation:

- **HDFS**: Used as the central data lake to store large-scale structured and unstructured datasets (stock prices and tweet data).
- **Apache Spark**: The core processing engine, chosen for its in-memory distributed computing capabilities. Spark was responsible for feature engineering (lag variables, moving averages), and sentiment aggregation using PySpark, making it suitable for scalable model preparation(Zaharia et al., 2016).
- **MongoDB, Cassandra, MySQL, PostgreSQL**: Served as storage targets to evaluate database performance. Spark connectors enabled efficient writing of batch-processed data into each system for benchmarking.

Additional Tools Considered

- **Apache Pig** and **MapReduce** were also explored as potential processing alternatives. While both are powerful for batch processing in Hadoop environments, they lack native support for text-based sentiment analysis, which was central to the project(Olston et al., 2008). Implementing sentiment analysis would have required a complete redesign of the data flow, so they were excluded from the final pipeline but acknowledged as comparable batch-processing options.

Benchmarking and Database Selection

To determine the most efficient storage solution for the project's analytical workload, four different databases were benchmarked using the Yahoo! Cloud Serving Benchmark (YCSB) tool: MongoDB, Cassandra, MySQL, and PostgreSQL(Cooper et al., n.d.). Each database was tested across standard YCSB workloads (A–F), which represent varied read/write operations to simulate real-world access patterns.

Performance Overview

- **MongoDB** demonstrated strong overall performance with a balance of high throughput (up to 1379 ops/sec) and low latency, especially for read-heavy workloads like B and D. Its flexible document structure made it suitable for storing enriched time series data with sentiment attributes.
- **PostgreSQL** achieved the highest throughput (1595 ops/sec in workload C) with negligible latency, but its rigid schema and lack of native document support made it less suitable for evolving analytical records like tweets and forecasts.
- **Cassandra** showed moderate throughput (272 ops/sec) but suffered from higher read and update latencies. Its write-optimized nature wasn't ideal for the project's read-focused evaluation tasks.
- **MySQL** had the lowest performance overall, with high latencies (up to 5885 µs) and limited throughput (263 ops/sec), making it unsuitable for large-scale analytics in this context.

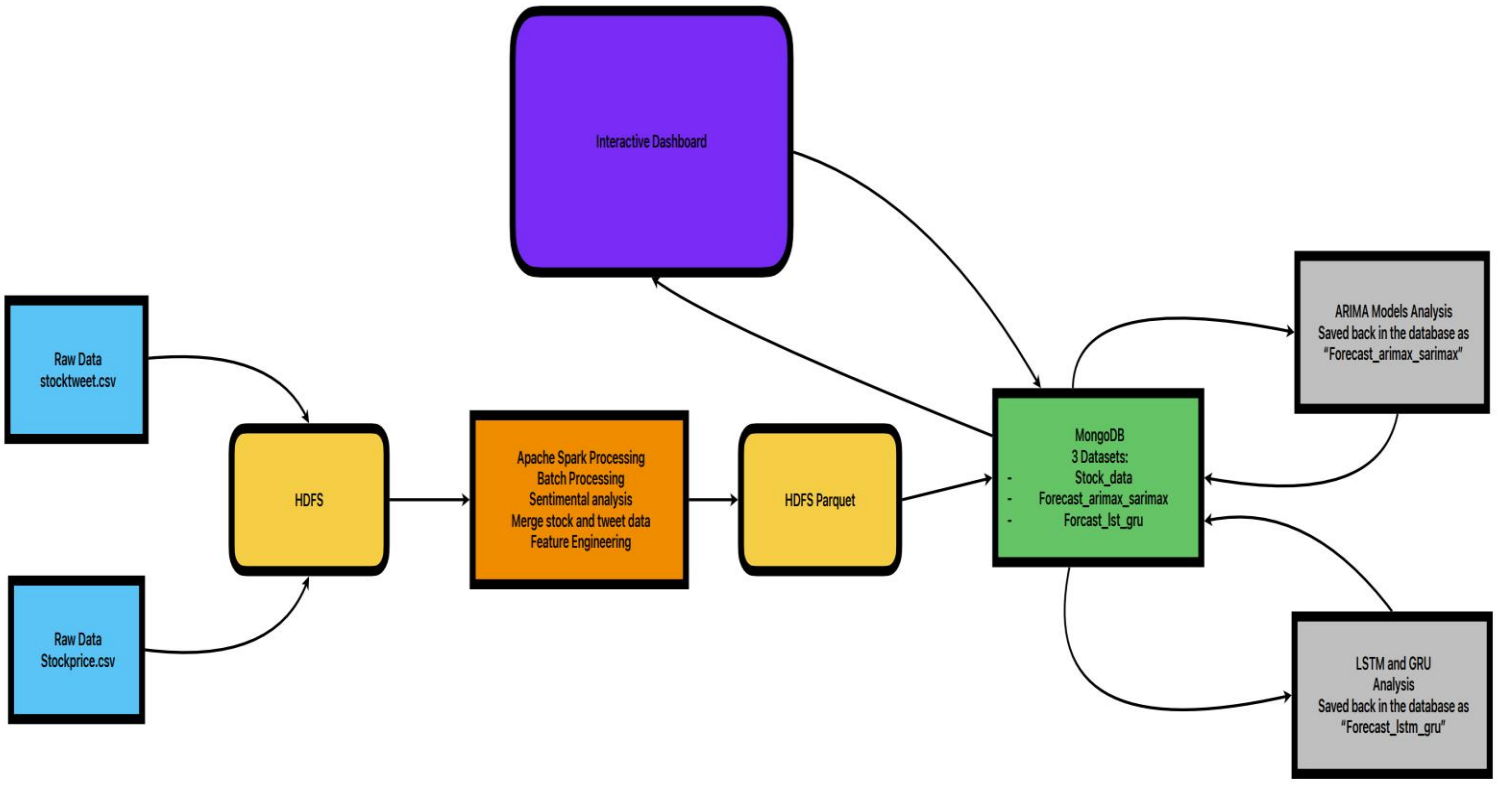


Figure 2 Final Design of Bigdata Architecture

Final Decision

Considering the need for scalable storage, flexibility in schema evolution, and low-latency access to mixed-format data (time series + sentiment + forecasts), MongoDB was selected as the primary database. It not only provided robust support for document-based records but also integrated smoothly with Spark through the MongoDB Spark Connector, enabling fast read/write operations within the big data pipeline.

Apache Kafka

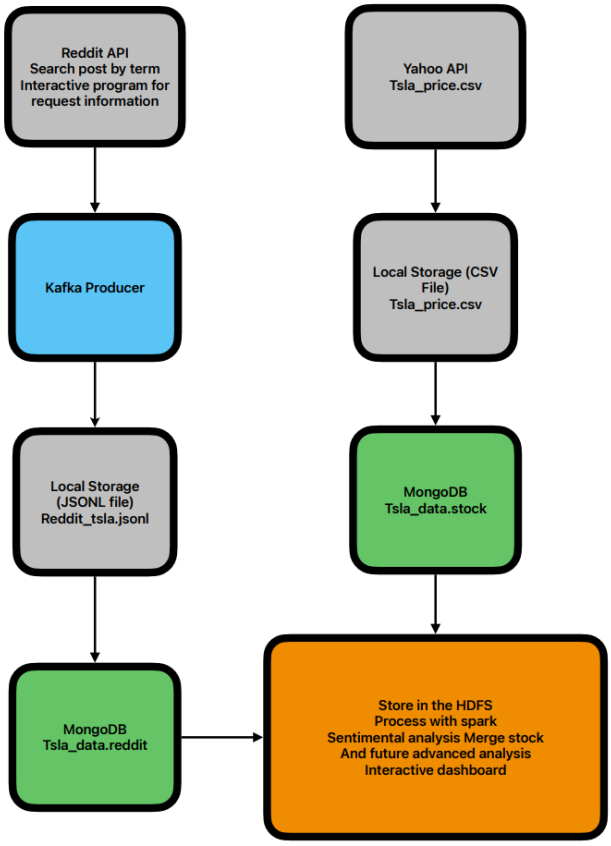


Figure 3 Streaming Architecture using Kafka

This study presents a data acquisition and storage framework combining financial stock data and social media content for future analytical tasks(Faisal Rafiq Khan, 2021). TSLA (Tesla Inc.) stock price data was retrieved using the Yahoo Finance API (yfinance) at one minute intervals and stored locally as a CSV file containing timestamps and corresponding closing prices. Simultaneously, Reddit posts related to TSLA were collected via the praw API, filtered by keyword and subreddit, and enriched with metadata such as title, post content, score, comment count, and timestamp. The posts were first published to a Kafka topic (reddit-news) for potential

real-time streaming applications and then stored locally in a JSON Lines file. Both the Reddit and stock datasets were subsequently ingested into a MongoDB database, where tsla_data.reddit_posts and tsla_data.stock_prices collections were used to facilitate structured storage.

Although Apache Spark was not employed in the current workflow, the data pipeline was designed with extensibility in mind. Future iterations of the system will integrate Spark for distributed processing of larger volumes of social and financial data. Planned enhancements include applying sentiment analysis to Reddit content, time-series forecasting models to stock prices, and the development of interactive dashboards for exploratory and predictive analytics.

Data preparation with Apache Spark

The data preparation pipeline was implemented using Apache Spark, chosen for its scalability and distributed data processing capabilities essential for handling large volumes of financial and textual data. The pipeline integrated stock market price data with social media sentiment extracted from tweets to generate a rich, feature-complete dataset for forecasting.

Sentiment Analysis

To analyze the sentiment of tweets, the **Spark NLP** (Kocaman and Talby, 2021) library was employed, leveraging its high-performance pretrained pipeline components:

- **DocumentAssembler**: Converted raw tweet text into a structured document format compatible with Spark NLP.
- **Tokenizer**: Split tweets into individual tokens (words), enabling more precise linguistic analysis.
- **SentimentDetector (ViveknModel)**: Applied supervised learning to classify each tweet as **positive**, **negative**, or **neutral** based on context and lexical cues.

Each tweet was processed to yield a sentiment label, and the results were aggregated per ticker per date to compute:

- positive, negative, and neutral tweet counts.
- Average sentiment score per day, normalized across all tweets mentioning a given stock.

This output was then merged with the financial dataset using the stock ticker and date as keys.

Stock Data Integration and Merging

After parsing and validating dates and ticker symbols, the stock data was joined with the tweet-level sentiment aggregates using a **Spark join operation** on ticker and date.

Feature Engineering

Several engineered features were constructed to capture historical patterns and short-term volatility:

- **Lag Features**: Included lag_Close_1 (previous day's closing price) and lag_sentiment_1 (previous day's average sentiment).
- **Rolling Statistics**: Computed avg_Close_5 (5-day moving average of closing price) and volatility_5 (5-day standard deviation of closing price), capturing short-term trends and market variability.
- **Sentiment-Volume Interactions**: Added tweet_volume to quantify daily social media engagement, which was later used as a feature in time-series models.

All features were validated for consistency, and missing values were handled to ensure reliable model training. The final dataset was saved to **MongoDB** using the official **MongoDB Spark Connector**, providing efficient write operations in a distributed environment.

Advanced Data Anlytics

In the advanced analytics phase, we implemented a combination of traditional statistical models (ARIMAX, SARIMAX) (Prakhar et al., 2022) and deep learning architectures (LSTM, GRU) (Chung et al., 2014) to forecast stock closing prices. These models were chosen to leverage both historical price patterns and external sentiment signals derived from Twitter. Prior to Spark based data processing, a sentiment analysis was conducted to evaluate the volume and quality of tweet data across 38 companies. To ensure analytical robustness and model interpretability, only the top six companies with the highest tweet volumes AAPL, TSLA, BA, DIS, AMZN, and MSFT were selected. This targeted approach ensured stronger correlations between stock price movements and sentiment-based exogenous variables, reducing noise and improving model performance for downstream forecasting and dashboard integration.

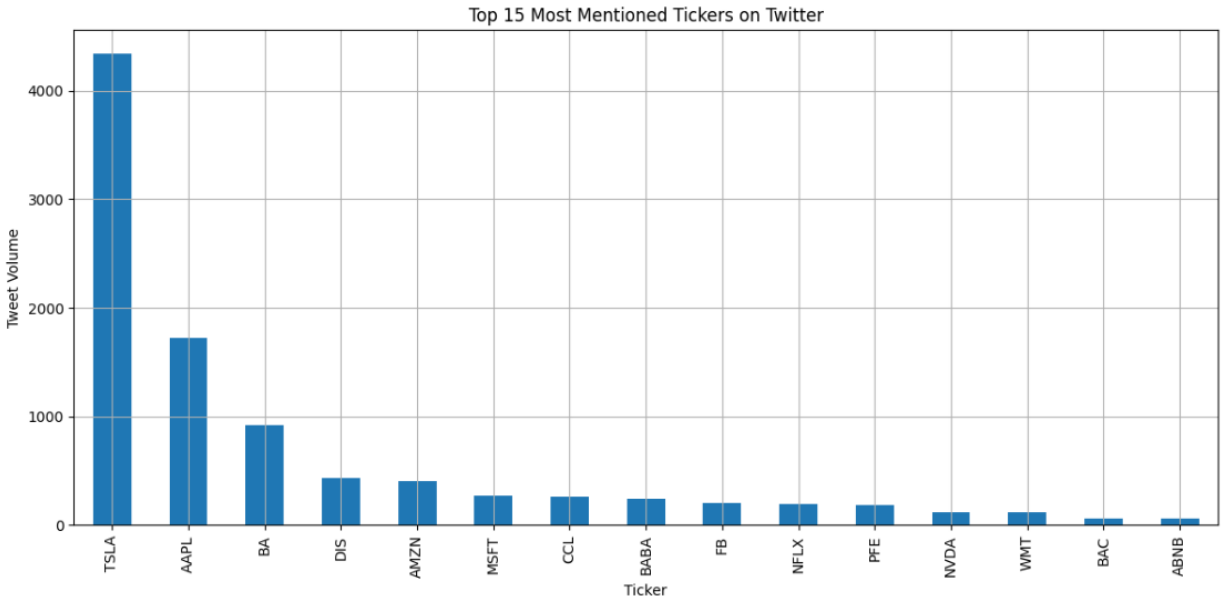


Figure 4 Top Companie with tweet volume

Handling Short times series

Due to the short time series in the dataset, a sliding window transformation was applied to convert sequences into supervised learning format for neural models (Andrii Gakhov, 2018). For traditional autoregressive models, differencing and model selection using AIC were applied to maintain predictive power.

ARIMA, ARIMAX, and SARIMAX Model Implementation

To initiate the time series forecasting phase, Apple Inc. (AAPL) was selected as a reference stock for developing and validating ARIMA models(Otexts.com, 2019). This decision was not based on tweet volume but rather to serve as a baseline example to perform a manual model analysis and contrast it with an automated parameter tuning approach. This methodical exploration enabled both interpretability and reproducibility before generalizing the pipeline to the remaining stocks. Before modeling, an exploratory data analysis (EDA) phase was conducted to better understand the dynamics of stock price movements and the correlation with sentiment data derived from tweets. The sentiment scores positive, negative, and neutral were averaged daily and aligned with stock price data to form a time indexed dataset

Stationarity and Model Order Identification

An initial **Augmented Dickey-Fuller (ADF)** test confirmed that the raw Close price series was **non-stationary** ($p \approx 0.947$). Applying **first-order differencing (d = 1)** resolved this issue, as the differenced series passed stationarity tests.

Analyzing the **ACF** and **PACF** plots:

- The **PACF** showed a significant spike at lag 1, followed by quick decay, indicating an **AR(1)** structure.
- The **ACF** displayed a strong spike at lag 1, also supporting an **MA(1)** component.
- No substantial autocorrelation was observed beyond lag 2, suggesting a simple **ARIMA(1,1,1)** model.

Seasonal and Exogenous Considerations

Given the nature of stock market data, potential seasonal effects (weekly patterns) were also considered. The SARIMAX(1,1,1)(1,1,1,5) model was constructed, incorporating a seasonal order with a periodicity of 5 (trading days). In addition, sentiment features, specifically a smoothed sentiment score, were used as an exogenous variable to explore its influence on future price movements.

Despite the inclusion of sentiment (sentiment_smooth), its coefficient was not statistically significant ($p = 0.792$), indicating weak predictive power in isolation. However, its presence slightly improved the model's AIC, justifying its role in a multivariate context.

Model Comparison and Evaluation

All three models ARIMA, ARIMAX, and SARIMAX, were evaluated using Root Mean Squared Error (RMSE) and Akaike Information Criterion (AIC)

	Model	RMSE	AIC
2	SARIMAX	4.388699	1166.243726
0	ARIMA	5.602023	1167.732155
1	ARIMAX	5.869502	1169.223328

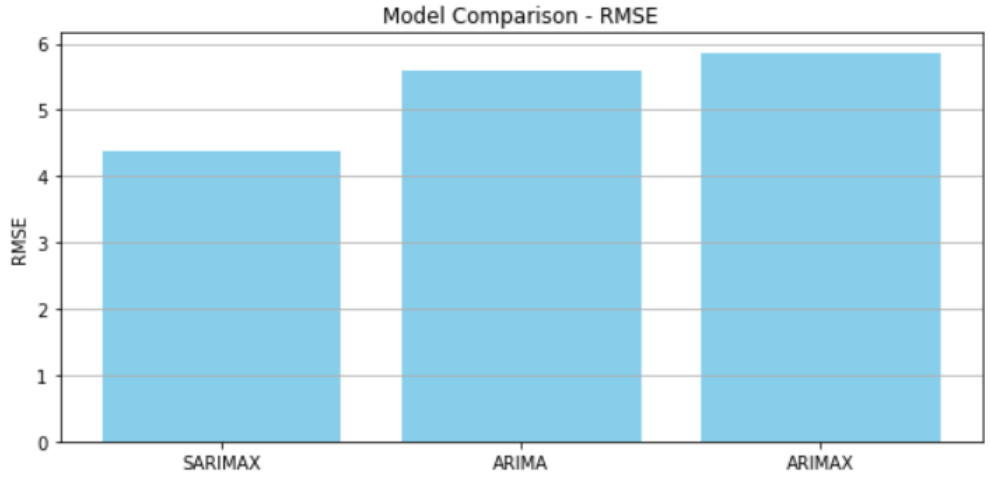


Figure 5 Comparison of Arima family models

The SARIMAX model emerged as the most accurate, yielding the lowest RMSE and AIC values, demonstrating its ability to capture both autoregressive structures and seasonal dependencies effectively. This outcome guided the project's decision to adopt SARIMAX and ARIMAX for future time series forecasting tasks, as both support the inclusion of exogenous variables like sentiment. In contrast, the ARIMA model, lacking this capability, was excluded from further analysis despite being considered during the initial evaluation.

Neural Network Forecasting: LSTM and GRU Models

To enhance predictive performance and capture the complex sequential structure inherent in financial time series, this project implemented Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks. These recurrent neural networks (RNNs) were selected for their ability to learn long-term dependencies, mitigate vanishing gradient issues, and model non-linear relationships particularly useful when integrating both lagged price data and exogenous sentiment signals from Twitter.

Unlike traditional statistical models such as ARIMA, these deep learning models do not assume linearity or stationarity, making them suitable for modeling the volatility and complexity of stock prices influenced by external factors.

Feature Selection and Correlation Analysis

Before model training, a correlation matrix was computed to assess multicollinearity and understand the relationship between the available features and the target variable (Close). Highly correlated features were either retained to preserve informative signals or removed to avoid redundancy. Final selected features included:

Lag_Close_1, avg_Close_5, volatility_5, tweet_volume, avg_sentiment and lag_sentimnet_1.

These features integrate both historical pricing dynamics and sentiment signals derived from pre-processed Twitter data, enriching the model's temporal and contextual understanding.

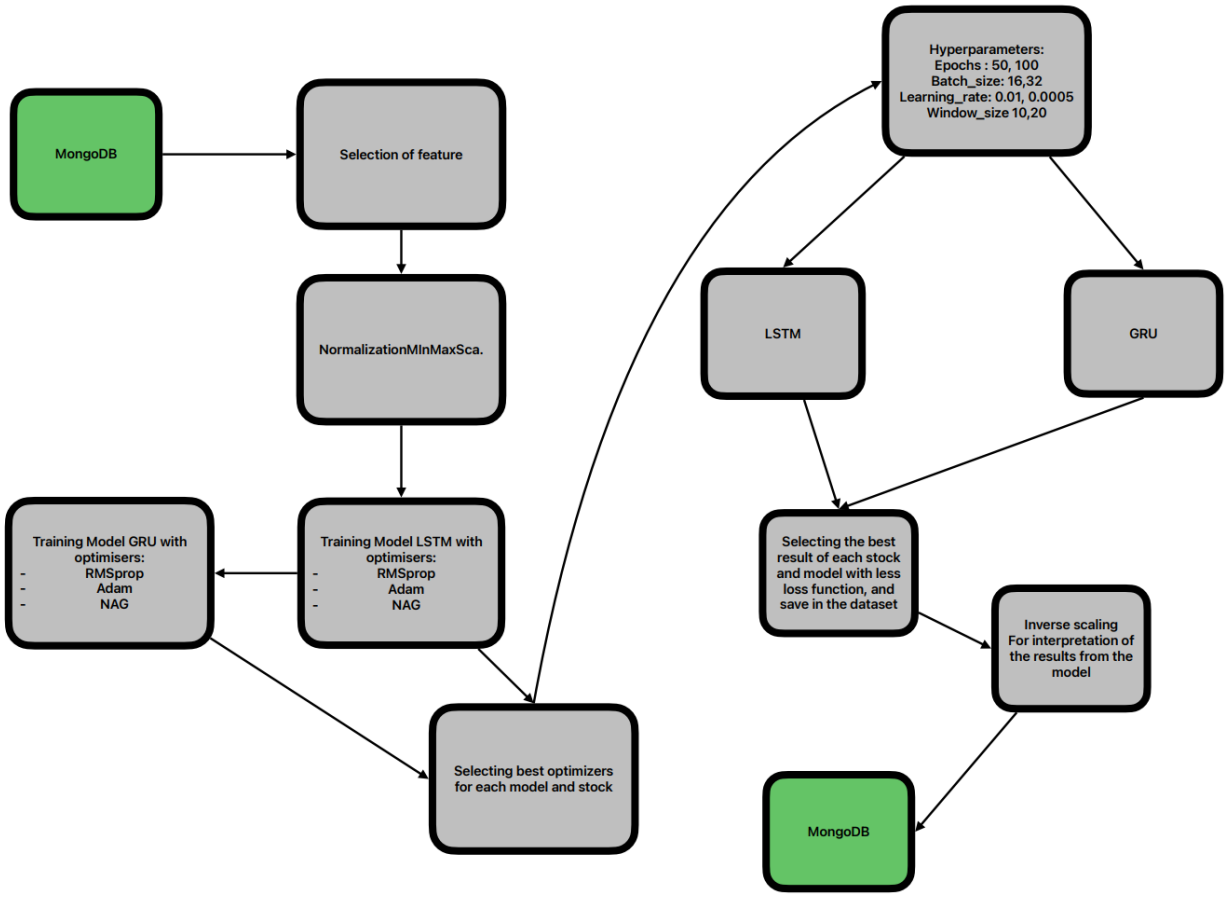


Figure 6 Architecture of Neural Network

Network Architecture and Model Training

To effectively model the temporal structure and complex non-linear dependencies present in stock and sentiment data, this project implemented two advanced recurrent architectures: **Long Short-Term Memory (LSTM)** and **Gated Recurrent Unit (GRU)**.

These models were chosen for their ability to retain long-range dependencies and manage vanishing gradients more effectively than traditional RNNs. The models were built using **Keras with TensorFlow backend**, allowing for streamlined configuration and scalability.

Architecture of Neural Network

Each model followed a uniform architecture composed of a **recurrent layer** (LSTM or GRU with 50 units), followed by an optional **dropout layer** to prevent overfitting, and concluded with a **dense output layer** producing forecasts for the next **7 consecutive days**. The recurrent layers utilized **tanh activations**, while internal gating mechanisms (input, forget, output gates) relied on **sigmoid functions** to regulate memory flow. The final layer used a linear activation to regress real-valued stock price predictions. The architecture was compiled with **Mean Squared Error (MSE)** as the loss function, chosen for its sensitivity to large errors.

Optimizers

A variety of **optimizers** were tested to enhance convergence stability and performance across tickers. These included **RMSprop**, effective for noisy data; **Adam**, known for its adaptive learning rate capabilities; and **NAG (Nesterov Accelerated Gradient)**, which anticipates gradient directions for smoother convergence. The optimizer was selected per model-ticker pair based on validation RMSE and loss behavior.

This section is visually supported by a **diagram showing the neural architecture (figure 6)**, detailing the data flow from time series input windows (including lagged prices and sentiment scores), through the recurrent layer, dropout layer, and into the multi-step dense forecast output. This helps clarify how temporal features are processed and transformed into future predictions.

Hyperparametes

Hyperparameters such as epochs, batch size, learning rate, and window size were tuned via grid search. The best-performing configurations were selected based on the lowest RMSE over a 7-day forecast horizon. After training, predictions were inverse transformed from their normalized form to recover interpretable values for integration into the dashboard.

Inverse Scaling and Forecast Recovery

Since the input features and output (Close) were normalized using **MinMaxScaler**, the final forecasted values were inverse transformed to recover the actual stock price scale. This step ensured interpretability and comparability with historical data and allowed seamless integration into the interactive dashboard.

Interactive Dashboard

As the final step in the project, an **interactive dashboard** was developed using **Streamlit**, enabling dynamic exploration and visualization of stock price forecasts, sentiment trends, and model performance. The dashboard serves as the front-end interface for end-users to engage with the results of advanced predictive models and big data analytics in a highly accessible format.

The dashboard connects directly to **MongoDB**, which stores the cleaned historical stock data, sentiment scores derived from tweets, and multi-model forecast outputs (from LSTM, GRU, ARIMAX, and SARIMAX). This real-time data connection ensures that visualizations remain responsive and up to date as new forecasts are generated.

Key features of the dashboard include:

- **Stock and Model Selection:** Users can choose from six major stocks (AAPL, TSLA, AMZN, MSFT, DIS, BA) and toggle between LSTM, GRU, ARIMAX, and SARIMAX models.
- **Forecast Modes:** Two forecast views are available — *Evaluation Mode*, which displays actual vs. predicted prices along with RMSE scores, and *Future Forecast Mode*, which shows multi-day predictions only.
- **Forecast Horizon Selection:** Users can specify the prediction window (1, 3, or 7 days) and explore how each model performs across different time spans.

- **Sentiment Analysis Panel:** Visualizations of positive, negative, and neutral tweet counts, along with tweet volume over time, help contextualize stock movements with public sentiment.
- **Date Range Filter:** A date picker allows users to restrict the analysis window to a custom timeframe, enhancing interpretability over specific market periods.
- **Performance Metrics Table:** In Evaluation Mode, RMSE values are summarized in a comparison table for all selected models and forecast windows.
- **Data Export:** Forecast data can be downloaded as CSV files for external analysis or reporting.

The dashboard was also designed with **Tufting visualization principles** in mind, prioritizing clarity, minimalism, and direct comparisons. Custom colors, hover interactions, and connector lines between historical and forecasted values help communicate model behavior effectively. The historical price line was set to dodger blue for visual emphasis, while forecasts use consistent color coding and tooltips for precision tracking.

Overall, the dashboard bridges the gap between **big data processing, predictive analytics**, and **end-user interpretation**, making it a valuable decision support tool for financial analysts and stakeholders.

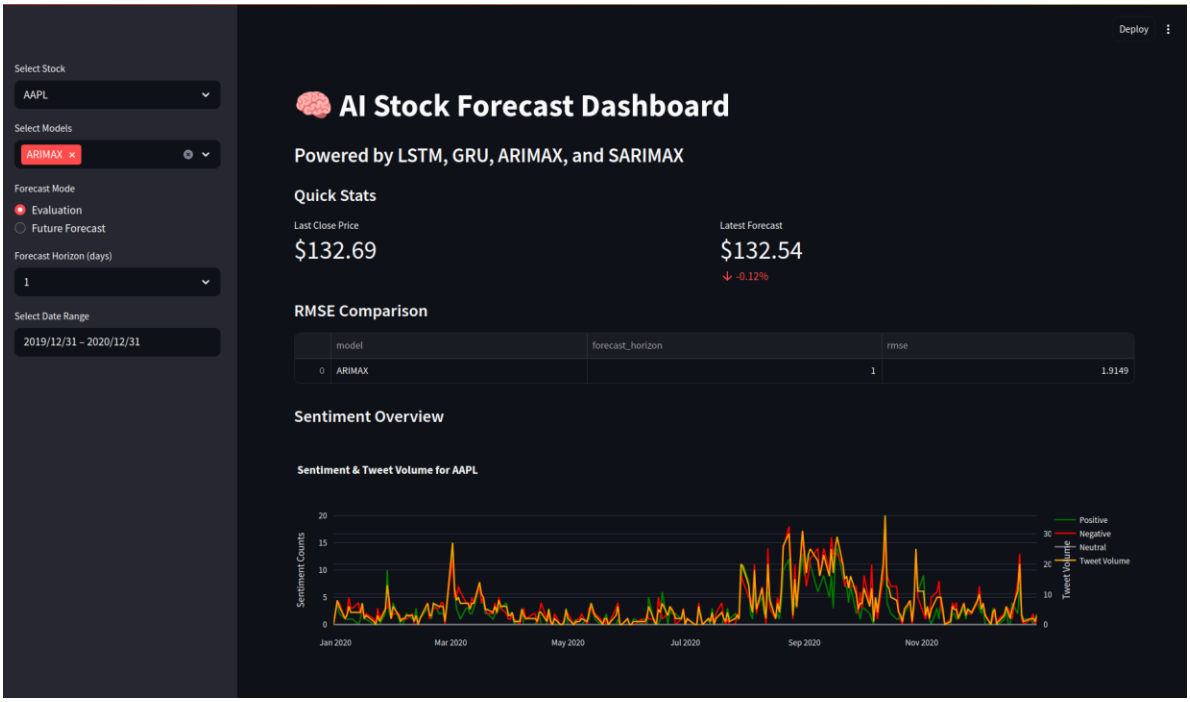


Figure 7 Interactive Dashboard

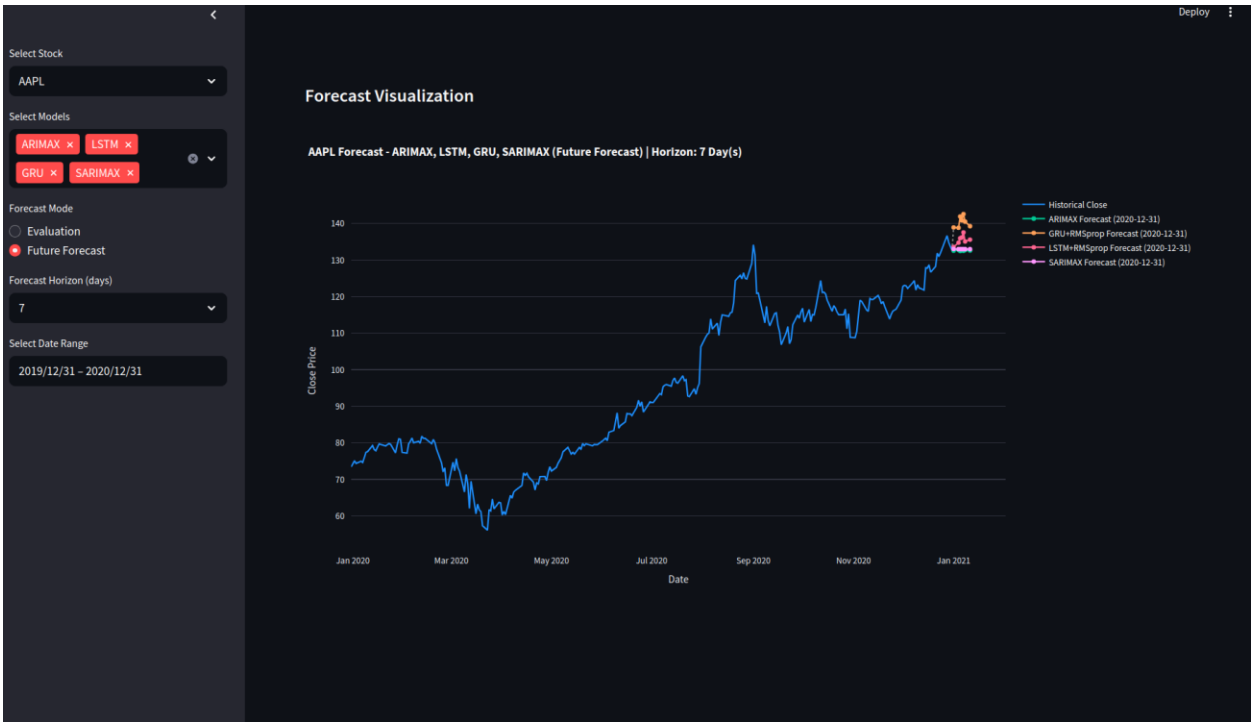


Figure 8 Interactive Dashboard Stock AAPL

Future Improvements

While this project successfully integrates time-series forecasting with sentiment analysis in a big data environment, several enhancements could further improve its robustness and scalability. Firstly, implementing real-time streaming capabilities using Apache Kafka and Spark Structured Streaming would allow continuous ingestion and forecasting, enabling more responsive insights aligned with market fluctuations. Secondly, the sentiment analysis module could be expanded by incorporating transformer-based models such as BERT or FinBERT, providing deeper contextual understanding of financial language beyond the rule-based approach currently applied. Thirdly, the hyperparameter tuning process could be enriched by testing a broader range of optimizers and configurations (e.g., learning rate decay, advanced scheduling strategies) and adopting smarter techniques like Bayesian optimization. Additionally, acquiring larger volumes of tweet data or integrating sentiment from alternative platforms like Reddit or news APIs would enhance the quality and reliability of the exogenous variables. Finally, to maximize forecasting accuracy for each stock, a manual model selection process for SARIMA parameters should be conducted per asset, ensuring seasonal patterns and lag structures are optimally captured based on ACF/PACF diagnostics and information criteria. These improvements would increase the system's precision, adaptability, and real-world applicability.

Conclusion

This project successfully designed and implemented an end-to-end predictive analytics pipeline combining big data processing, time series forecasting, and interactive visualization. By integrating stock price data and public sentiment extracted from tweets, the system provides short-term forecasting using both classical models (ARIMAX, SARIMAX) and advanced neural networks (LSTM, GRU). The Lambda architecture enabled scalable, distributed processing through Apache Spark and HDFS, while the evaluation of multiple databases using YCSB benchmarking ensured optimal data storage and dashboard performance, with MongoDB emerging as the preferred solution.

The project demonstrated how sentiment signals can be engineered, merged, and effectively leveraged to improve forecasting accuracy. An interactive Streamlit dashboard was developed to visualize model comparisons, RMSE, sentiment trends, and future forecasts, offering a user-friendly interface for stakeholders

Overall, the solution reflects a practical application of machine learning and big data technologies in financial analytics. It sets the foundation for future enhancements such as real-time prediction, deeper NLP integration, and broader financial data sourcing to further improve precision and usability in production environments.

Reference

Bollen, J., Mao, H. and Zeng, X. (2011). Twitter mood predicts the stock market. Journal of Computational Science, 2(1), pp.1–8.

Ieee.org. (2025). IEEE Xplore Full-Text PDF: [online] Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9300187 [Accessed 16 May 2025].

hadoop (n.d.). https://hadoopabcd.wordpress.com/wp-content/uploads/2015/05/oreilly-hadoop-the-definitive-guide-3rd-edition-may-2012.pdf.

Zaharia, M., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J. and Venkataraman, S. (2016). Apache Spark: a Unified Engine for Big Data Processing. *Communications of the ACM*, 59(11), pp.56–65. doi:https://doi.org/10.1145/2934664.

Andrii Gakhov (2018). An Introduction to Time Series Forecasting with Python. [online] doi:https://doi.org/10.13140/RG.2.2.18053.86249.

Faisal Rafiq Khan (2021). Apache kafka with real-time data streaming. [online] ResearchGate. Available at: https://www.researchgate.net/publication/348575301_Apache_kafka_with_real-time_data_streaming.

Kocaman, V. and Talby, D. (2021). Spark NLP: Natural Language Understanding at Scale. *Software Impacts*, 8, p.100058. doi:https://doi.org/10.1016/j.simpa.2021.100058.

Prakhar, K., S, S., E, S., M, K. and B, S.K. (2022). Effective Stock Price Prediction using Time Series Forecasting. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICOEI53556.2022.9776830.

Olston, C., Reed, B., Srivastava, U., Kumar, R. and Tomkins, A. (2008). Pig latin. Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08. doi:https://doi.org/10.1145/1376616.1376726.

Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R. and Sears, R. (n.d.). Benchmarking Cloud Serving Systems with YCSB. [online] Available at: https://courses.cs.duke.edu/fall13/compsci590.4/838-CloudPapers/ycsb.pdf.

Otexts.com. (2019). Chapter 8 ARIMA models | Forecasting: Principles and Practice. [online] Available at: https://otexts.com/fpp2/arima.html.

Chung, J., Caglar Gulcehre, Cho, K. and Y. Bengio (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. [online] ResearchGate. Available at: https://www.researchgate.net/publication/269416998_Empirical_Evaluation_of_Gated_Recurrent_Neural_Networks_on_Sequence_Modeling.