**Big Data Strategies for Sentiment Analysis: A Comparative Evaluation of Time Series,** **Forecaster Autoreg, and Deep Learning Models**

***Abstract***—This study evaluates various predictive models for sentiment analysis (David Zimbra, 2018) on Twitter data within a big data framework, emphasizing the critical role of Exploratory Data Analysis (EDA) (Gelman, 01 Jan 2012) and strategic missing value imputation. Utilizing a robust infrastructure including Hadoop, Spark, and MongoDB (Verch, 2016), the research focuses on the effectiveness of traditional machine learning models like ARIMA and Forecaster Autoreg (Athanasopoulos, 2018), alongside advanced deep learning models such as Bi-LSTM (Sunny, Maswood, & Alharbi, 2020) and Bi-GRU (Mohammed M.Abdelgwad a, 2021/08). Special attention is given to EDA and the application of the KNN mean (Jianping Gou a, 2018/08) technique for filling missing values, which significantly enhances the accuracy and reliability of time series forecasting. The study leverages PySpark for efficient in-memory computations and Keras Tuner (Mohamad Zaim Awang Pon, 2021)for optimal hyperparameter tuning, demonstrating improvements in model performance through rigorous data preparation.

**Keywords** : **Spark ,MongoDB,ForecasterAutoreg,KNN\_mean**

**Research Questions** :

How does the architecture of big data systems influence the scalability and efficiency of sentiment analysis tools in handling large volumes of social media analytics?

How do traditional time series models, the Forecaster Autoreg model, and deep learning approaches compare in terms of performance when analyzing sentiment data from social media?

**Introduction**

In the evolving field of data science, sentiment analysis is crucial, especially within social media analytics. As platforms like Twitter increasingly influence public discourse, accurately predicting sentiments from tweets using sophisticated models and big data technologies becomes essential for stakeholders such as marketers, policymakers, and researchers. This paper explores the effectiveness of various predictive models in analyzing Twitter data within a big data framework, highlighting the integration of advanced machine learning and deep learning techniques.

The advent of big data technologies has enabled the management of vast amounts of data from social media platforms. Utilizing a robust architecture that includes Hadoop for distributed storage, Spark for efficient data processing, and MongoDB (Verch, 2016) for scalable storage, this study creates a comprehensive environment to implement and compare different sentiment analysis models. This setup ensures the manageability of large-scale data and the agility to process data in real-time, crucial for effective sentiment analysis.

The study integrates deep learning models like Bidirectional LSTM (Bi-LSTM) (Sunny, Maswood, & Alharbi, 2020) and Bidirectional GRU (Bi-GRU) (Mohammed M.Abdelgwad a, 2021/08), along with traditional models such as ARIMA and ForcastAutoregre, to achieve a nuanced understanding of sentiment dynamics. These models are selected for their ability to effectively capture and interpret the sequential and contextual nuances of textual data. The research addresses the challenge of selecting and optimizing these models to maximize sentiment prediction accuracy, utilizing PySpark for in-memory computations and Keras Tuner (Mohamad Zaim Awang Pon, 2021)for hyperparameter tuning. This approach is expected to not only improve the accuracy of sentiment predictions but also provide insights into the computational efficiency and scalability of applying these models in a big data environment.

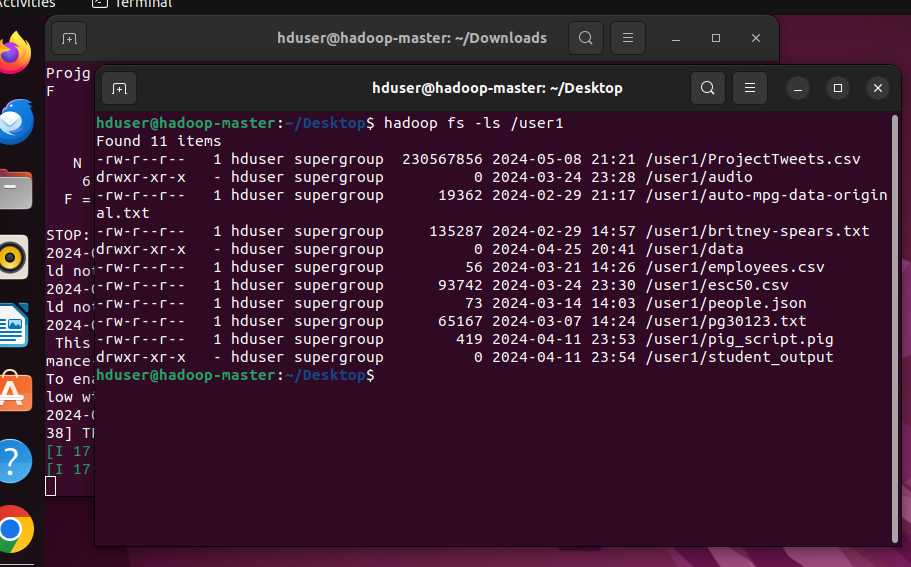
**Methodology**

**Big Data Storage & Processing**

Introduction to Project Environment

|  |  |
| --- | --- |
| Software | Version |
| Ubuntu | Unbuntu 22.04.3 ITS |
| Hadoop | 3.3.6 |
| Spark | 3.4.2 |
| Hbase | 2.4.17 |
| Cassandra | 3.11.16 |
| Mysql | 8.036 |
| MobgoDB | 3.2.10 |

1. **Data storage and processing**

****

Figure(1)

As figure(1) shows, Data is initially loaded into HDFS from local sources. In real-world scenarios, Python scripts continually fetch Tweets data from APIs and consistently upload it to HDFS. Once uploaded, the data becomes accessible to other nodes within the cluster, facilitating data sharing.

During the EDA phase, Spark is used to understand the scale and types of data through its processing interfaces, which help simplify the data and identify missing values. Spark’s lazy evaluation means transformations like map(), filter(), and drop() are registered in the DAG but not executed immediately. They are executed when actions like collect(), count(), or save() are invoked. EDA often requires output from previous steps to guide subsequent operations, necessitating frequent use of action operations.

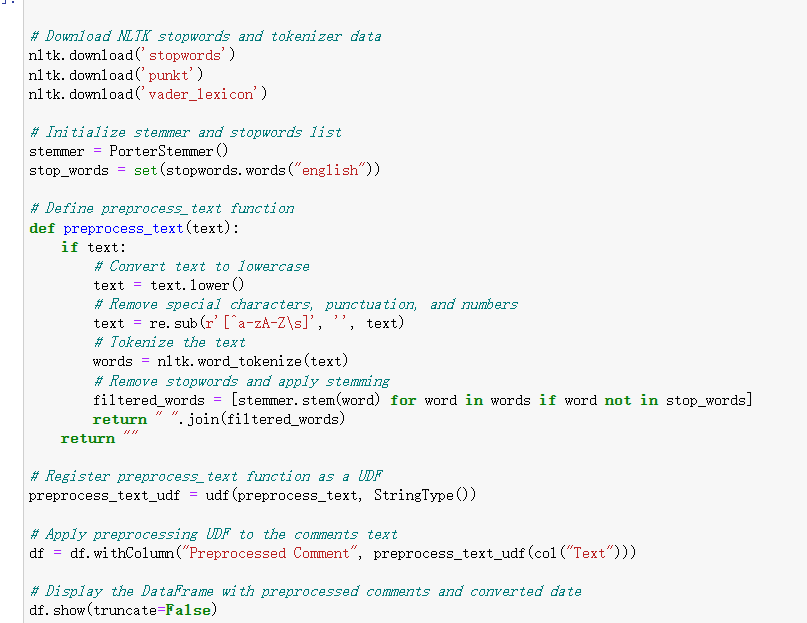


Figure (2)

As figure(2) shows, Use of User-Defined Functions (UDF) allow for the creation of custom functions to process and transform data into new columns. However, UDFs are typically slower than built-in functions as Spark cannot optimize them, potentially leading to performance trade-offs. Hence, built-in functions are generally preferred.

After processing comments for sentiment analysis, the need for extensive statistical computations and visualizations necessitates converting Spark DataFrames into Pandas DataFrames. This marks a transition from using Spark to using more traditional data storage solutions. Data can be stored on local disks or databases; two options provided here are MySQL and MongoDB. Ensuring these databases can connect with Spark is crucial.

|  |  |
| --- | --- |
| Data Base Connection Work log | |
| Mysql | Download the appropriate connector, place it in /path/spark/jars so Spark recognizes it at startup. Create databases and tables in MySQL as per data format requirements. |
| Mongodb | Download the connector suitable for the current MongoDB version used, as newer connectors might have compatibility issues with older versions, and place it in /path/spark/jars. |

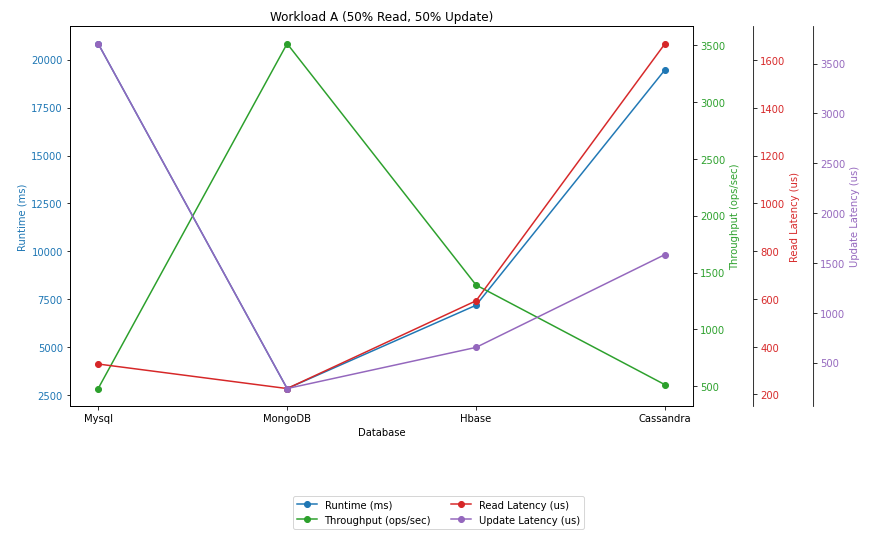
As a test, data is first stored in MySQL, then retrieved and stored in MongoDB using Spark, and accessed via Python from MongoDB. Depending on practical needs, any storage method can be utilized. This transition from a Spark environment to a Python environment is seamless with appropriate database connections.

**2 Comparative analysis for Database**

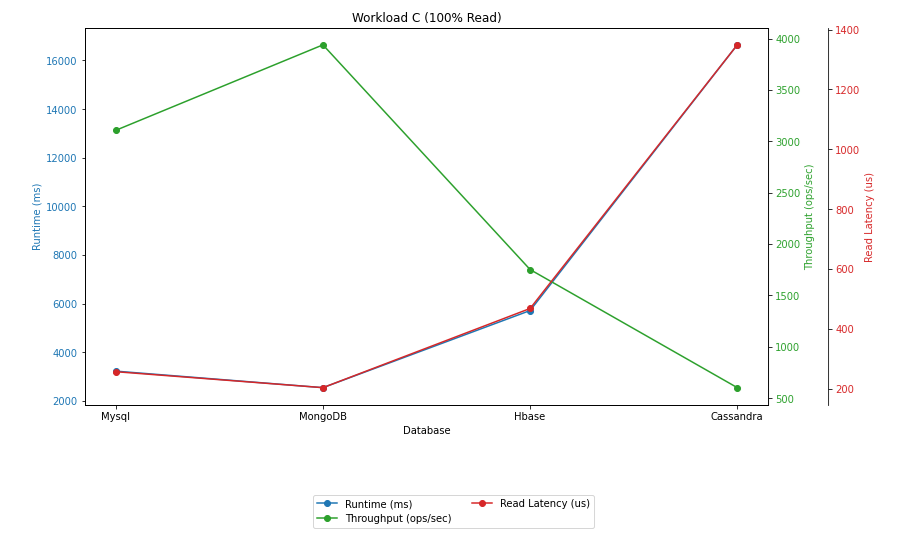
This experiment involved performance testing of MySQL, MongoDB, HBase, and Cassandra (Featherston, 2010)using the YCSB (Seghier & Kazar, 2021) workload. Connection strategies for HBase and Cassandra include:

|  |  |
| --- | --- |
| Ycsb Connection work log | |
| Hbase | Locate the binding file under ycsb-0.17.0 directory and use the appropriate command to invoke it, e.g., /bin/ycsb-0.17.0 load Hbase20... |
| Cassandra | Adjust parameters in the YCSB-0.17.0 binding file or use the direct interface command, e.g., bin/ycsb-0.17.0 load cassandra-cql.... |

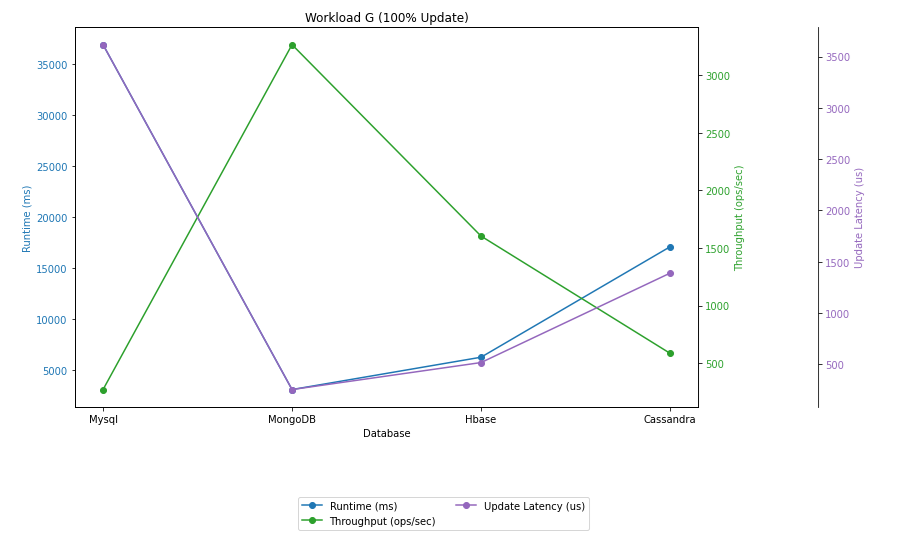
The test standards set by the workload files aimed for 10,000 Read and Update operations, with ratios of 50% read to 50% update, 100% read, and 0% update. Ensuring the database is operational before initiating data loads, running performance tests, and clearing lists for repeated tests are essential.



Figure(3)



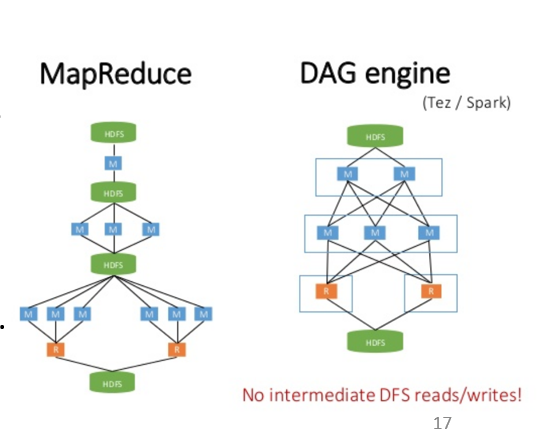
Figure(4)



Figure(5)

As Figure(3-5)show, MongoDB, in particular, demonstrated outstanding performance across all tested scenarios. It maintained high throughput levels and exhibited low latency during both read and update operations, which indicates its efficiency in managing simultaneous read and write demands. Such performance is particularly notable in environments where rapid data retrieval and updates are crucial, such as in real-time analytics and transactional systems.

1. **discussion of the rationale and justification**



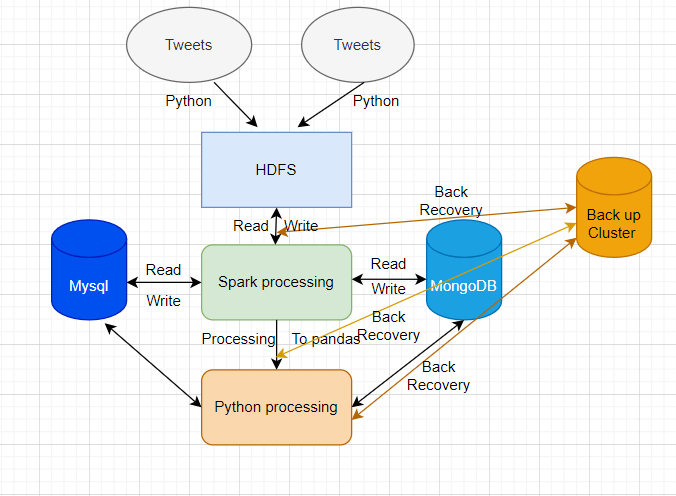
Figure(6)

Choosing Spark over MapReduce was dictated by Spark's design optimized for speed, utilizing in-memory computation (RAM) which can be up to 100 times faster than MapReduce when processing in-memory, and 10 times faster on disk. Unlike MapReduce, which requires disk writes after each Map or Reduce operation, leading to slower processing times, Spark’s design based on the Directed Acyclic Graph (DAG) significantly reduces disk read/write operations, enhancing processing speed. Furthermore, Spark’s compatibility with multiple programming languages and integration with various software systems like MySQL, HBase, and Cassandra, affirm its position as a leading tool in big data processing.

In terms of programming style, during the EDA processing phase, unavoidable action operations like collect(), count() are called, but it is possible to minimize the use of spark.stop() code as much as possible when memory allows. This reduces the overhead of starting and stopping SparkContext, thus enhancing overall performance.

Regarding data, MongoDB, which performs best in Ycsb tests, is chosen not only for its excellent write and update performance but also because it is a document-oriented database. This allows data to be stored without the need for pre-creating tables, making it very convenient to use.

1. **Architecture for big data processing**

.

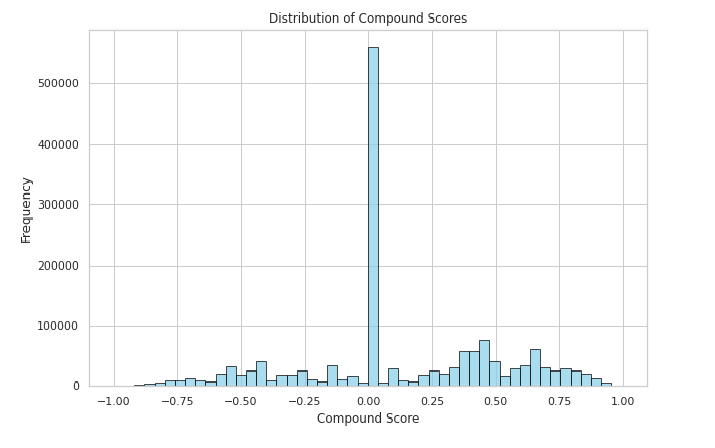
Figure(7)

As Figure(7)shows, In this expanded architecture for big data processing using Twitter data, the system ensures continuous data collection through automated Python scripts which interface directly with Twitter APIs. This data is then pre-processed to cleanse and standardize information before it's loaded into HDFS. Apache Spark, renowned for its fast processing capabilities, efficiently handles large-scale data transformations and complex analytics directly within HDFS. These capabilities allow for real-time analysis and immediate data manipulation, which are crucial for dynamic datasets like Twitter feeds.

After initial processing, the data is seamlessly integrated into SQL and NoSQL databases such as MySQL and MongoDB, facilitating structured storage and further analysis. This architecture also includes robust backup and recovery systems across all stages, ensuring data integrity and quick recovery in the event of failures.

**Advanced Data Analytics**

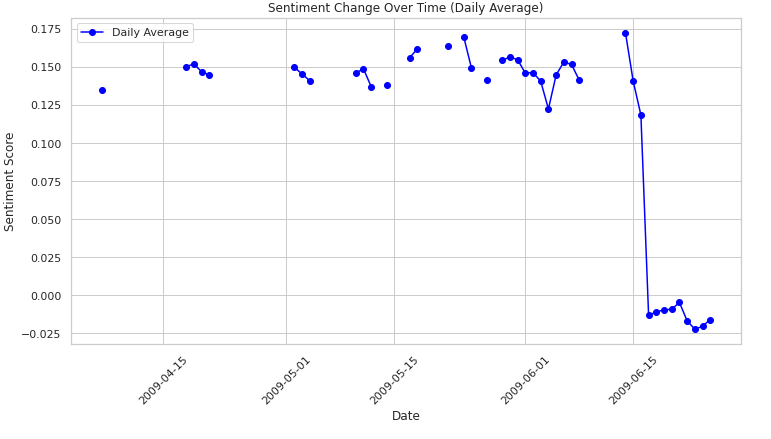
1. **Discussion of rationale, evaluation, and justification in terms of EDA**



Figure(8)

In the preliminary stages of Exploratory Data Analysis (EDA), using the NLTK library for sentiment analysis of tweets, a compound sentiment score was obtained.As figure(8)shows. The distribution pattern where more than half of the dataset centered around a score of zero suggests that the majority of tweets have a neutral sentiment. This distribution pattern provides crucial reference information for subsequent sentiment classification.

Notably, despite most tweets showing neutrality, a significant proportion displayed distinctly positive or negative sentiments. To capture these emotional shifts more accurately and effectively distinguish between sentiments, a threshold was set at 0.01. This means any tweet with a compound sentiment score above 0.01 is classified as positive, while those below -0.01 are classified as negative. This threshold considers the need to capture subtle emotional shifts while minimizing the risk of misclassifying essentially neutral tweets as positive or negative. This method ensures accuracy and sensitivity in sentiment classification, allowing for a more detailed observation and analysis of emotional fluctuations in tweets.

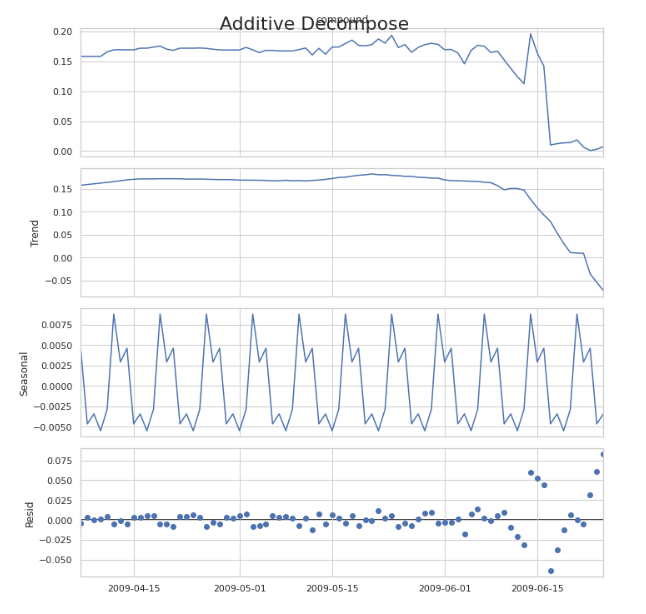


Figure(9)

As Figure(9)shows,after processing the dataset by daily averages, it was found that multiple days of data were missing. These missing values are discontinuously distributed throughout the time series, so careful selection of appropriate missing value filling methods is necessary to ensure data integrity and the accuracy of the analysis.

Common missing value filling methods like Forward Fill and Backward Fill rely on continuity and are not suitable in this case as the data gaps are discontinuous, which could lead to misinterpretation of the time series. Additionally, the Seasonal Mean method requires data to exhibit clear seasonal variation characteristics, but if the data does not display strong seasonal patterns, this method is also inappropriate. The Linear Fill method assumes a linear trend in the data, which is unsuitable for time series that may exhibit complex or non-linear trends.

Therefore, the KNN Mean method was chosen to fill missing values. This method considers the 'neighboring' data points of the missing points and uses their means for filling, which does not require continuity in the time series nor rely on the data's linear characteristics, making it highly suitable for handling scattered and irregular missing data situations. The KNN Mean method can flexibly adapt to the local characteristics of the data, providing a strategy to effectively handle missing values while preserving the original information characteristics of the data.



Figure(10)

In Figure(10) we can see,after decomposing the data using additive decomposition, the analysis revealed seasonal and trend components in the data. This decomposition clearly demonstrated the basic patterns of the data over time, providing a crucial basis for model selection and subsequent analysis. For these findings, choosing the appropriate time series analysis model is essential. ARIMA (Autoregressive Integrated Moving Average) and SARIMA (Seasonal ARIMA) models are powerful tools for handling non-stationary time series data. The ARIMA model is suitable for capturing non-seasonal trends and fluctuations in the data, while the SARIMA model adds consideration of seasonal factors, enabling more accurate predictions of time series with seasonal cycles.

In the field of deep learning, Bidirectional Long Short-Term Memory networks (Bidirectional LSTM) and Bidirectional Gated Recurrent Units (Bidirectional GRU) were selected to handle time series data. The bidirectional structure of these models allows them to learn both forward and backward dependencies in time series data, a capability that makes them perform well in handling complex time series data, especially when long-term dependencies are present.

Additionally, autoregressive models based on Random Forest and Ridge Regression (ForecasterAutoreg and ForecasterAutoregDirect) were considered. These models predict future data points by integrating multiple lagged values as inputs, with their core advantage being good adaptability and robustness to non-stationary sequences, suitable for handling complex nonlinear patterns that are difficult to capture with traditional statistical models.

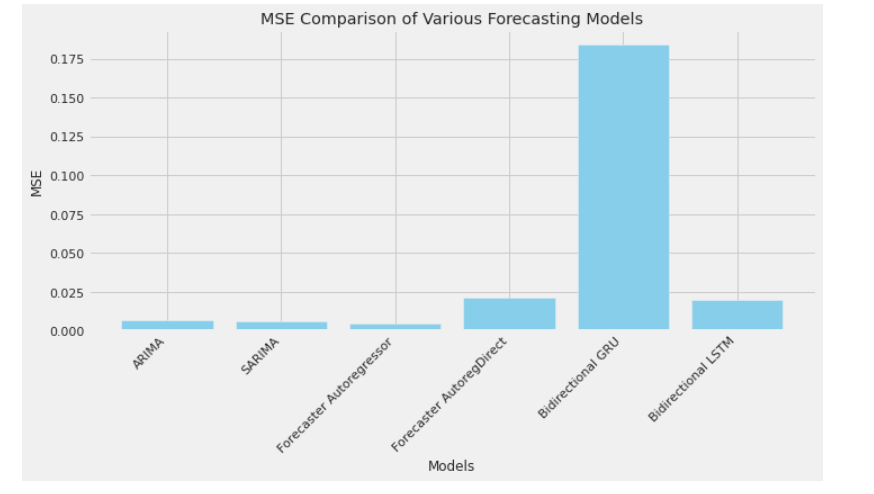
Overall, each model has its unique application scenarios and advantages. In practice, it is often necessary to evaluate and compare multiple models to find the one that best suits the current data characteristics.

1. **hyperparameter tuning**

Regarding hyperparameter tuning, using the pmdarima library for modeling and parameter optimization of ARIMA and SARIMA models offers significant advantages due to its ability to automatically select and optimize parameters, greatly reducing the complexity of manual configuration. The automation features of the pmdarima library not only simplify the model-building process but also provide detailed statistical information during parameter adjustment, allowing users to understand the logic behind model selection and adjustment clearly.

In the machine learning field, models such as RandomForestRegressor and RidgeRegressor, through the hyperparameter tuning capabilities provided by the skforecast library, can potentially improve the accuracy of model predictions. However, this training dataset-based tuning strategy may lead to insufficient generalization ability on unseen data. Therefore, when using these methods, analysts need to be careful to assess the model's performance on an independent test set to ensure its practicality and reliability.

For Bidirectional LSTM and Bidirectional GRU neural network models, manual hyperparameter tuning is usually required during the data preparation phase. In this process, analysts choose different lookback values based on experience, which are crucial parameters that determine the model input window size, typically choosing 7 and 12 as the window period. Choosing the appropriate lookback value is crucial for the model to capture dependencies in time series data. During the model training phase, the Keras Tuner library was used, allowing users to specify nearly all parameters that could affect model performance and automatically conducting extensive experiments to find the optimal configuration. Another significant feature of Keras Tuner is its ability to save each training result, avoiding repeated training and saving considerable computational resources and time.



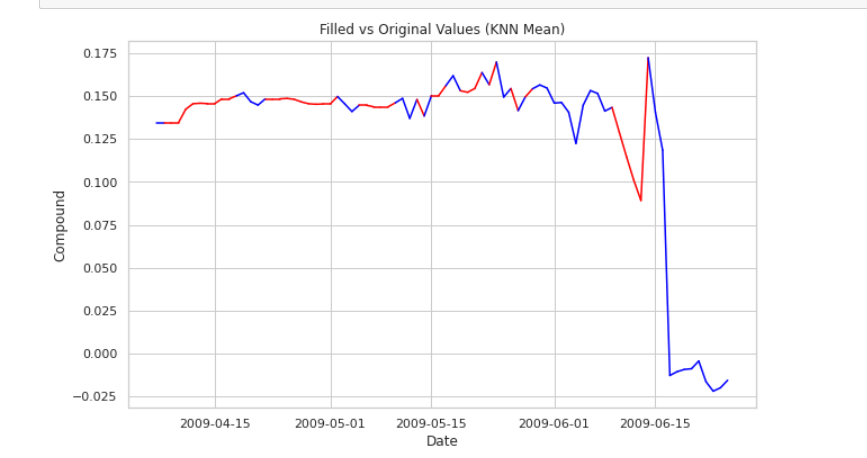
Figure(11)

From the Figure(11), it's evident that the Bidirectional GRU model shows the highest MSE, suggesting it may not perform as well as other models on the given dataset. This could be due to overfitting, insufficient training, or the data's characteristics not matching well with the GRU's capability to capture complex structures. On the other hand, another complex neural network model—the Bidirectional LSTM—displays significantly lower MSE, indicating it can more effectively capture the patterns in the data rather than fitting noise or irrelevant patterns.

Traditional time series models like ARIMA and SARIMA show relatively low MSE compared to more complex neural network-based models, demonstrating solid performance. This may suggest these models are sufficient to fit the data without needing to capture the more complex dependencies that neural network designs are intended to handle. Interestingly, the Forecaster Autoreg model, both in its regular and direct versions, displays very competitive low MSE values, emphasizing their effectiveness in scenarios that require a balance between simplicity and predictive power.

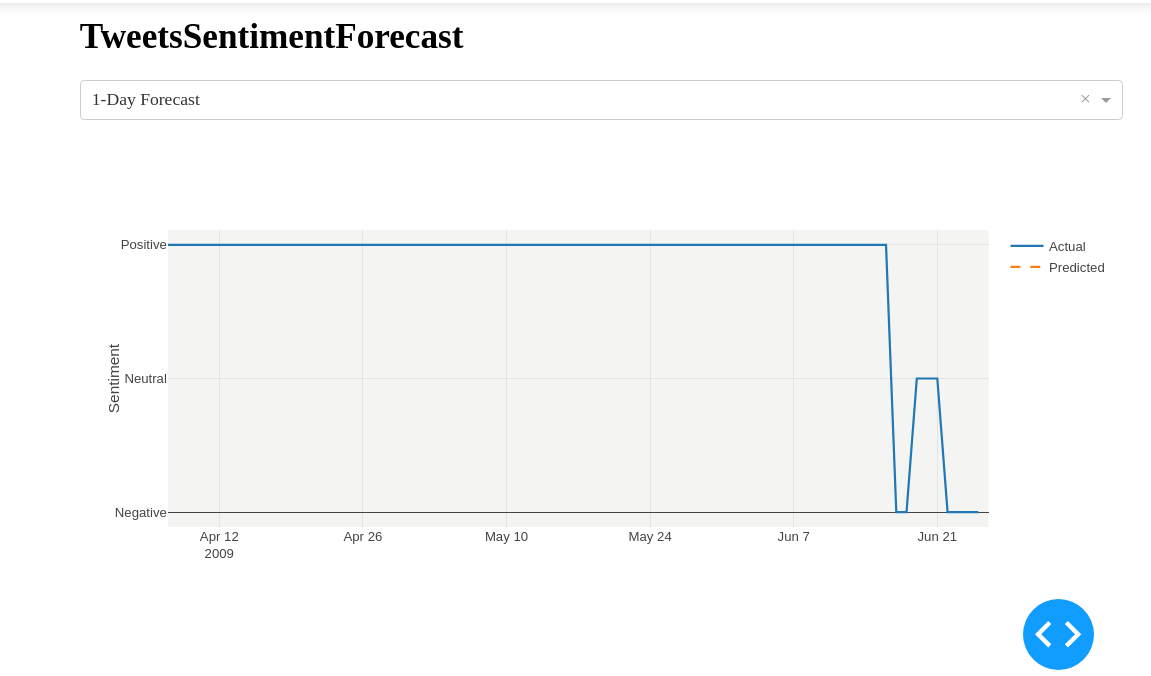
The final prediction will be made using the Forecaster Autoreg model.

1. **Sentiment Analysis**

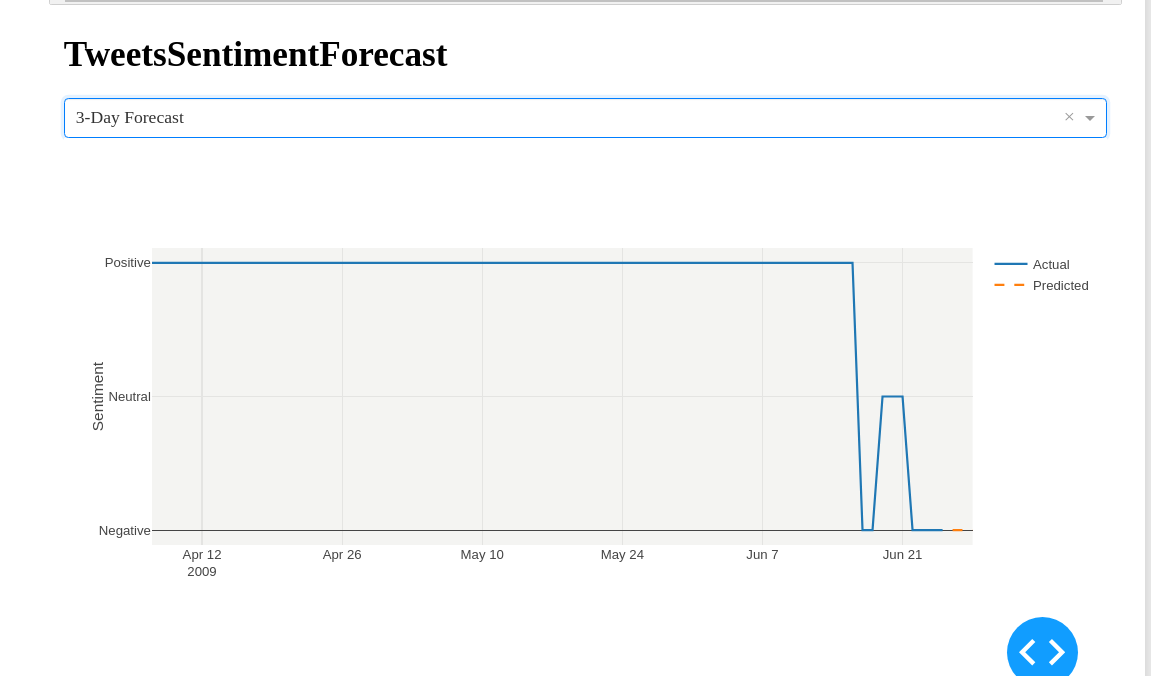


Figure(12)

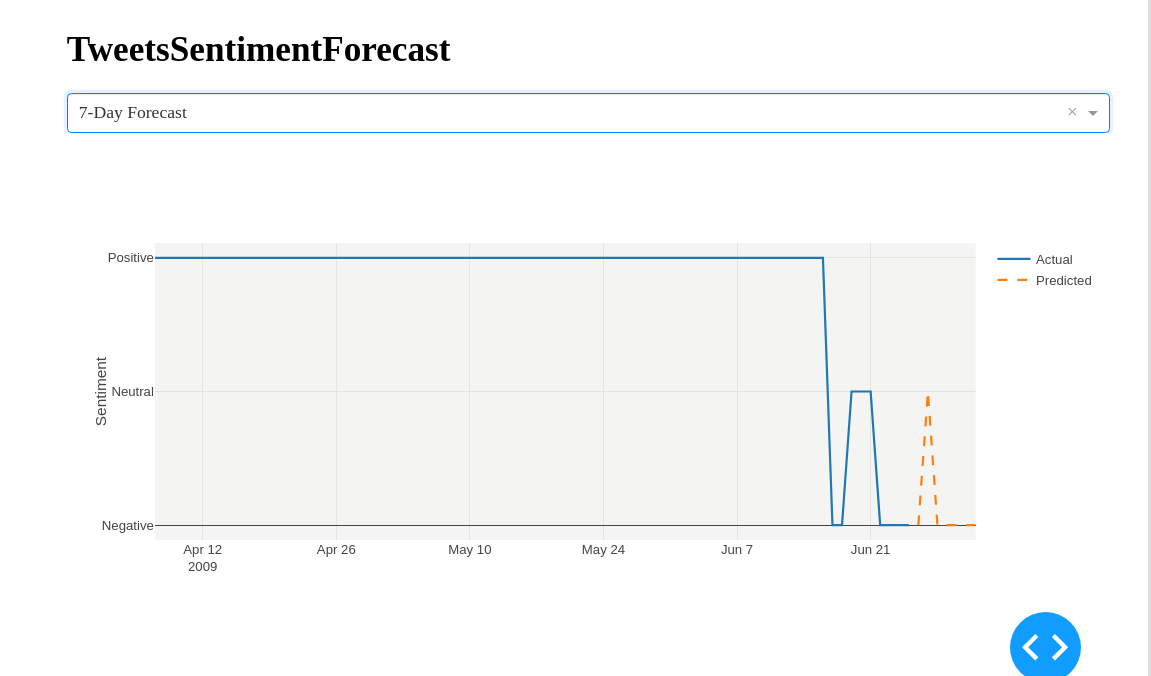
From figure(12)The analysis of sentiment change observed from the graph indicates that from April 15th to early June, both curves mostly remain in the positive region, suggesting that the overall sentiment of tweets during this period is positive. Despite some fluctuations, the positive nature of the sentiment trend is relatively stable. In early June, there is a noticeable drop in sentiment, with sentiment scores rapidly declining to near or below zero, indicating a sudden burst of negative sentiment possibly linked to specific events or news affecting public mood significantly.



Figure(13)



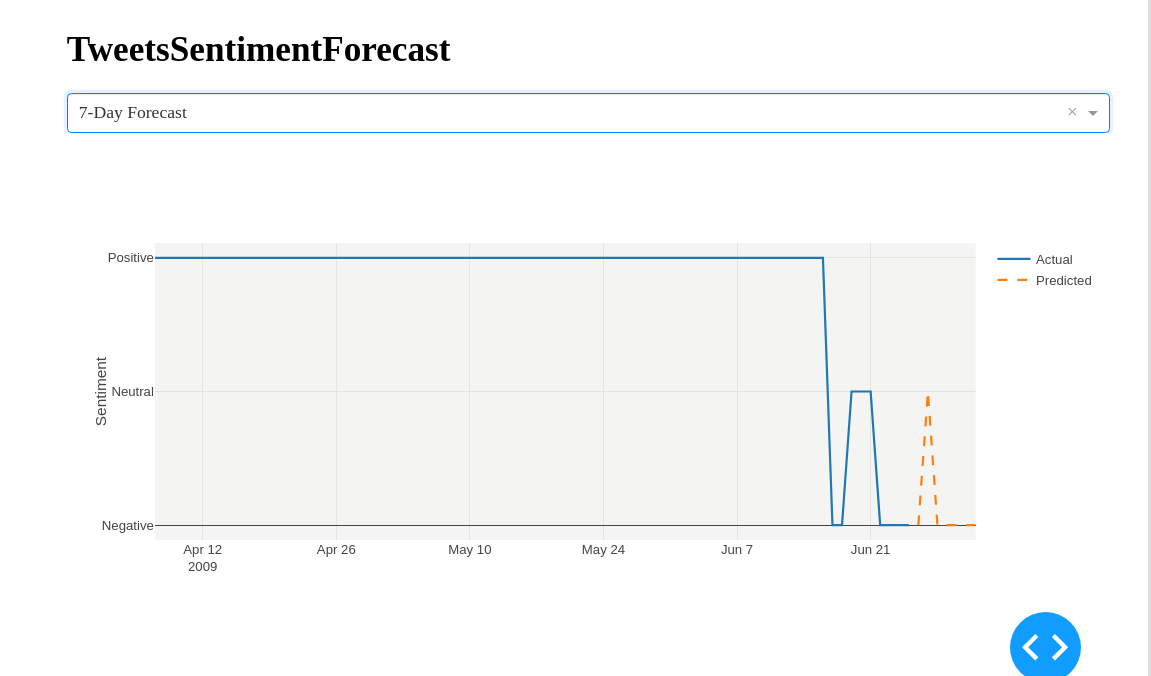
Figure(14)



Figure(15)

The graph presents tweet sentiment predictions and actual data from April 12th to June 21st, 2009. According to the chart, the predictions for one day and three days are close to the actual values, indicating minor fluctuations and maintaining a stable level. This suggests that in the short term, both positive and negative emotional changes are relatively mild, possibly reflecting no significant events or emotional triggers during this period. However, in the 7-day forecast, a larger fluctuation can be observed, particularly in early June. This sudden drop in sentiment may indicate the occurrence of some events with significant social impact, which could have triggered extensive discussion and emotional reactions on social media.

1. **Present the Dasboard**



Figure(16)

Based on Tufte's design principles, as figure(16) shows this dashboard showcases a clear and uncluttered interface. The charts feature crisp lines and vivid color contrasts, aiding users in quickly identifying key data points, such as the differences between forecasted and actual values. The titles "TweetSentimentForecast" and "7-Day Forecast" clearly mark the theme and purpose of the charts, allowing users to instantly understand the core message being conveyed. Additionally, the charts intuitively display sentiment categories (positive, neutral, negative), making the nature and scope of the data immediately apparent.

The design of the dashboard focuses on the clear presentation of data rather than excessive design elements, embodying Tufte's principle of minimizing non-data ink—reducing unnecessary elements in visual design. The chart designs prioritize data presentation over beautifying the graphics themselves, helping users focus on the most important information without being distracted by complex visual elements.

Overall, the dashboard is highly successful in its design, presenting key information in a simple and effective manner while faithfully adhering to Edward Tufte’s principles of information design. By effectively utilizing visual elements to enhance the understanding and interpretation of data, the dashboard not only provides information that is easy to absorb and analyze but also optimizes the user’s visual experience, making it both intuitive and practical. This design approach ensures transparency and readability of data, making it an ideal choice for displaying complex data and trends.

**Conclusion**

This study, conducted within a big data framework using various predictive models for Twitter sentiment analysis, demonstrates the critical role of big data technologies and databases in managing and processing large-scale data. Advanced technologies such as Hadoop, Spark, and MongoDB ensure effective data management and batch processing capabilities, essential for timely and accurate sentiment analysis.

By employing Exploratory Data Analysis (EDA) and filling missing values with methods like KNN mean, the research underscores the importance of data preprocessing in time series prediction. These steps not only enhance data integrity but also improve the models' ability to capture emotional shifts within time series, thereby increasing prediction accuracy.

Performance evaluations reveal that deep learning models like Bidirectional LSTM and GRU perform poorly with complex sequence data, while traditional machine learning models such as ARIMA and Random Forest provide reliable predictions for more stable data. Additionally, the Forecaster Autoreg model, due to its outstanding performance and low Mean Squared Error (MSE) on datasets, emerged as the best-performing model in this study.

In conclusion, this research not only confirms the effectiveness of big data technologies in handling and analyzing large-scale social media data but also validates the significance of EDA and proper data filling strategies in enhancing the accuracy of time series predictions. These findings offer valuable insights for data scientists in selecting and optimizing models, especially when dealing with rapidly changing social media data.

Reference

AthanasopoulosJ Hyndman · GeorgeRob. (2018). Forecasting: principles and practice.

David ZimbraAbbasi,Daniel Zeng,Hsinchun ChenAhmed. (2018). The State-of-the-Art in Twitter Sentiment Analysis: A Review and Benchmark Evaluation. Publication History.

FeatherstonDietrich. (2010). Cassandra: Principles and Application.

GelmanAndrew. (01 Jan 2012). Exploratory Data Analysis for Complex Models.

Jianping Gou aMa b, Weihua Ou c, Shaoning Zeng d, Yunbo Rao e, Hebiao Yang aHongxing. (2018/08). A generalized mean distance-based k-nearest neighbor classifier.

Mohamad Zaim Awang PonPrakash K KKrishna. (2021). Hyperparameter Tuning of Deep learning Models in Keras.

Mohammed M.Abdelgwad aHassan A Soliman a, Ahmed I.Taloba a, Mohamed Fawzy FarghalyTaysir. (2021/08). Arabic aspect based sentiment analysis using bidirectional GRU based models.

SeghierBenNadia, & KazarOkba. (2021). Performance Benchmarking and Comparison of NoSQL Databases: Redis vs MongoDB vs Cassandra Using YCSB Tool. IEEE.

SunnyArif IstiakeMd., MaswoodMohd ShahriarMirza, & AlharbiG.Abdullah. (2020). Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model.

VerchBanker · Douglas Garrett · Peter Bakkum · ShaunKyle. (2016). MongoDB in Action: Covers MongoDB version 3.0.