**A Comparative Study: In-Memory vs. Disk-Based Computing with Random Forest for Stock Analysis**

Chitra Joshi 1[https://orcid.org/0009-0000-3643-5557], Chitrakant Banchorr 2[https://orcid.org/0000-0002-7287-0801],   
Omkaresh Kulkarni 3[https://orcid.org/0000-0003-4775-1437], Kirti Wanjale 4[https://orcid.org/0000-0003-4271-504X],

1 B.Tech (Final Year) Student, Computer Engineering Department, Vishwakarma Institute of Information Technology (VIIT), Pune, India; [chitra.joshi@viit.ac.in](mailto:chitra.joshi@viit.ac.in)

2 Assistant Professor, Computer Science & Engineering (AI) Department, VIIT, Pune, India; [chitrakant.banchhor@viit.ac.in](mailto:chitrakant.banchhor@viit.ac.in)

3 Assistant Professor, Computer Science & Engineering (AIDS) Department, VIIT, Pune; [omkaresh.kulkarni@viit.ac.in](mailto:omkaresh.kulkarni@viit.ac.in)

4 Assistant Professor, Computer Engineering Department, VIIT, Pune, India; [kirti.wanjale@viit.ac.in](mailto:kirti.wanjale@viit.ac.in)

\* Correspondence: Chitra Joshi ([chitra.joshi@viit.ac.in](mailto:chitra.joshi@viit.ac.in))

**Abstract:**

**Background:** The advancement of big-data analytics calls for careful selection of processing frameworks to optimize machine learning effectiveness. Choosing the appropriate framework can significantly influence the speed and accuracy of data analysis, ultimately leading to more informed decision-making. In adapting to this changing landscape, businesses should focus on factors like how well a system scales, how easily it can be used, and how effectively it integrates with their existing tools. The effectiveness of these frameworks plays a crucial role in determining data processing speed, model training efficiency, and predictive accuracy. As data becomes increasingly large, diverse, and fast-moving, conventional processing systems often fall short of the performance required for modern analytics.

**Objective:** This research seeks to thoroughly assess the performance of two prominent big data processing frameworks —Apache Spark (in-memory computing) and MapReduce (disk-based computing)— with a focus on applying Random Forest algorithms to predict stock prices. The primary objective is to assess and compare their effectiveness in handling large-scale financial datasets, focusing on key aspects such as **predictive accuracy**, **processing speed**, and **scalability**.

**Methods:** The investigation utilised the MapReduce methodology and Apache Spark independently to analyse a substantial stock price dataset and to train a Random Forest regressor. Mean Squared Error (MSE) and Root Mean Square Error (RMSE) were employed to assess the primary performance indicators of the models, while Mean Absolute Error (MAE) and the R-squared (R²) value were used to evaluate the models' goodness of fit.

**Results:** The RMSE, MAE, and MSE obtained for the Spark based implementation were lower, compared to the MapReduce based implementation, although these low values indicate high prediction accuracy. It also had a big impact on the time it took to train and run models because of its optimised in-memory processing. As opposed to this, the MapReduce approach had higher latency and lower accuracy, reflecting its disk-based constraints and reduced efficiency for iterative machine learning tasks.

**Conclusion:** The conclusion supports the fact that Spark is the better option for complex machine learning tasks such as stock price prediction, as it is good for handling the large amount of data. MapReduce is still a reliable framework but not fast enough to process and not lightweight enough for analytics that are too rapid and iterative. The outcomes of this study are helpful for data scientists and financial analysts to choose the most appropriate framework for the big data machine learning applications.

**Keywords:** Apache Spark, MapReduce, Big Data, Random Forest, Performance Comparison, Data processing, In-memory processing, disk-based processing***.***

**Additional Information and Declarations**

**Data Availability:** The authors used a dataset available from <https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction>

# Introduction

In the age of digital change, the amount of data created every day is increasing exponentially which has resulted in what is called as the ‘big data’ (Bhadoria, Pandey, & Kundu, 2021). This refers to large sets of data that are challenging to manage and analyse using the conventional data management tools and techniques. Big data analysis is very vital for any organisation in the current world irrespective of the industry they belong to, be it finance, health, e-commerce etc. Since, the global organisations are now trying to capitalise on the available data, there has been a growing need for strong data processing frameworks.

Big data technologies empower organizations to design scalable architectures that leverage distributed computing across multiple servers. These tools facilitate **parallel processing**, enabling the efficient execution of complex computations on vast volumes of data. Among the most prominent frameworks supporting such capabilities are **MapReduce** and **Apache Spark**, both of them are now core components in big data analytics; thanks to their dependable and effective performance in processing massive datasets.

Stock price predictions are naturally hard because financial data is changeable, has a lot of it, and moves quickly. Big data technologies are needed to quickly handle large volumes of data, either instantly or within short response times. However, standard approaches frequently fail to handle unstructured and time-sensitive stock data efficiently. This highlights the importance of comparing distributed computing technologies like MapReduce and Apache Spark. Their performance, scalability, and ability to handle iterative machine learning algorithms, all have a direct impact on prediction accuracy and timeliness, which are crucial in financial forecasting because even minor delays or inaccuracies can result in considerable losses.

MapReduce is a programming methodology established by Google that is strongly employed for the process of large-scale data systems. MapReduce programming model utilizes the simple formulaic logic of fragmentation of tasks into smaller components called sub tasks, this programming model enables parallel processing throughout a cluster in a distributed event. While MapReduce is designed for batch processing, it is also designed to the fundamental principle that information is stored on a disk which projected resource lag particularly in situations which repeatedly work with the same data which in turn effects efficiency especially in use cases which make use of machine learning computational algorithms.

The MapReduce framework works by dividing tasks into two main phases: the Map phase, where the input data is split and processed in parallel to generate intermediate key-value pairs, and the Reduce phase, where these intermediate results are aggregated to generate the final output (Rajapandiyan & Subbaiyan, 2025). This system exhibits significant scalability and resilience to faults, rendering it ideal for the batch processing of extensive datasets. Nevertheless, due to its dependence on disk-based storage for each intermediate output between tasks, MapReduce can experience significant I/O overhead, which reduces its efficiency for operations that are often encountered in machine learning and real-time analytics.

Apache Spark has been developed as a challenging alternative to the traditional frameworks such as MapReduce. By bringing in the elements of ease of use and speedy SLAs on Spark, this framework has a far better performance with regards to data access latencies as data can be accessed directly. Because of these reasons, there are many situations where Spark is enabled to be more competent as compared to MapReduce especially when it comes to handling iterative and real time algorithms. Furthermore, a wide variety of primitive libraries such as entity classifiers, graphs and stream processing libraries are available within Spark that empower a data analyst or scientist with beneficial tools.

In contrast to MapReduce, as noted by Barvaliya (2024),“***Apache Spark*** *is an open-source unified analytics engine designed for large-scale data processing*”, offering a significant performance boost over MapReduce by leveraging **in-memory computing**. Spark utilises Resilient Distributed Datasets (RDDs) to store intermediate results in memory, which minimises the necessity for disc reads and writes, thereby significantly accelerating computation. Spark facilitates a comprehensive approach by enabling batch processing alongside streaming data, SQL queries, predictive modelling, and graph processing, all integrated within one structure. The flexibility of Spark, combined with its user-friendly interface and comprehensive API, positions it as a top option for modern big data applications that demand real-time performance and repetitive processing.

The objective of this study is to conduct an empirical analysis and performance evaluation of the performance metrics of both the Apache Spark and the MapReduce framework in terms of their implementation of Random Forest algorithms for stock price predictions. With an emphasis on performance-based metrics like scalability, processing speed and the predictive accuracy of each Integrated Development Environment, we can present the insights with regards to the strengths and weaknesses of both frameworks for machine learning tasks. An understanding of these nuances remains pivotal for any practitioner or researcher who seeks to utilize big data tools for the financial sector.

This study aims to evaluate and contrast the effectiveness of in-memory and disk-based distributed computing frameworks in stock price prediction using the Random Forest algorithm. The following research questions guide this investigation:

**Q1:** How does the predictive accuracy of Apache Spark (in-memory computing) compare with MapReduce (disk-based computing) when applied to stock price prediction using the Random Forest algorithm?

**Q2:** What are the differences in error metrics (such as Mean Squared Error, Root Mean Squared Error, Mean Absolute Error) between MapReduce and Apache Spark in the context of stock price forecasting?

**Q3:** How do the two frameworks differ in terms of residual error distribution and model fit, as reflected by R-squared values and histogram analysis?

This paper's succeeding sections have been structured as follows: Section 2 surveys associated studies in the areas of stock price prediction and large data processing. The experimental setting and performance metrics utilised in the comparative study are described in Section 3. The findings of our assessment are shown in Section 4, which is followed by a discussion. The work is ultimately summarised in Section 5.

# Literature Review

In recent years, the quantity of data has increased at a rambling rate making it necessary to have big data processing frameworks that are efficient. Two of the more popular frameworks are Apache Spark and Hadoop MapReduce, both of which have their own advantages. Research in this domain includes an analysis of these frameworks and their areas of use with respect to performance, scalability and efficiency.

To create an in-depth understanding of the performance comparison between MapReduce and Apache Spark in big data analytics, particularly with respect to forecasting stock prices, we conducted a directed literature review. We selected relevant peer-reviewed research articles, conference papers, and academic publications by searching digital databases. Preference was given to recent studies (within the last 10-12 years) that specifically examined the performance metrics, implementation strategies, and real-world applications of these distributed computing frameworks. This review helped in identifying research gaps and positioning our work within the existing body of knowledge.

M. R. S. Kumar and H. S. Mohan(2024) presented an optimisation strategy for enhancing big data processing efficiency. The study focused on addressing latency and throughput challenges in large-scale data environments using Hadoop’s MapReduce framework combined with Genetic Algorithm-Based Optimization. The study emphasised the trade-offs between old MapReduce and newer big data frameworks, proving that optimisation strategies can close performance gaps and make MapReduce more effective for large-scale data analysis.

Sifat Ibtisum and others (2023) conducted a comparative analysis on MapReduce and Apache Spark for processing large-scale healthcare datasets. Their findings showed that Spark’s memory-centric processing offered better performance than MapReduce, especially in batch workloads. While the study provided execution time comparisons, it did not address predictive modelling. Our research builds on this by applying Random Forest for stock price prediction, offering a deeper insight into model accuracy and error analysis.

A comparative study by Satish Gopalani and R. Arora (2015) highlights the differences between Hadoop MapReduce and Apache Spark, focusing on their performance utilizing K-Means clustering for big data insights. The study concludes that Spark significantly outperforms MapReduce due to its in-memory processing capability, enabling accelerated and more efficient data operations. While the paper centers on clustering rather than predictive modeling, it reinforces the growing industry trend of adopting Spark for scalable, multi-purpose data processing. Our work extends this comparison into the realm of supervised learning, specifically Random Forest regression for stock price prediction.

Peddi (2019) focused on the challenges of processing unstructured stock market data using the Hadoop MapReduce framework. The study highlighted the shortcomings of conventional RDBMS in handling Big Data and demonstrated how Hadoop’s HDFS and MapReduce model offer scalable solutions for both storage and processing. Through the implementation of MapReduce jobs on unstructured stock datasets, the work emphasized the framework's ability to manage large-scale financial data efficiently and encouraged further exploration into optimizing unstructured data processing using big data technologies.

In the paper by Zaharia et al (2012), Resilient Distributed Datasets (RDDs) are introduced which form the basis of the Apache Spark framework as it makes memory processing fault tolerant. RDDs enable Spark to overcome the high latency of Hadoop MapReduce which makes Spark more appropriate for use during a number of machine learning cycles. The authors, demonstrate how the Spark principles allow it to excel in tasks where data needs to be written and read repetitively, meaning that it has a strong foundation for being used in real-time and interactive analytics.

Meng and others (2016) examined the scalability of Spark’s MLlib library and highlighted its distinctive advantages over other machine learning libraries. Their study emphasized that MLlib’s seamless integration with Spark significantly improves the execution of iterative algorithms like Random Forest and K-Means on large-scale datasets. Moreover, the paper outlines how MLlib surpasses MapReduce, which lacks native support for machine learning tasks, thereby underscoring Spark’s suitability for big data and machine learning applications. This seamless support enables faster development and deployment of data-driven models in real-world scenarios. As a result, Spark has become a preferred choice for organizations dealing with complex analytical workloads and high-volume data streams.

Birjali et al (2020) presented an architecture for competitive intelligence that runs on spark and allows collection, storage, analysis, and visualization of data to assist organizations in decision making. The works above illustrate how the built in libraries of spark together with its in memory computing has enabled parallel computation of data, an important consideration in areas like competitive intelligence because of the demand for instantaneous data processing. However, the authors add, Spark’s deep learning libraries are still in high demand for areas where modelling complexity is needed, owing to the libraries already present in Spark.

Concerning financial data, Gupta and Sharma (2021) compared Apache Spark and Hadoop MapReduce for the purpose of evaluating stock market data influenced by the COVID 19 pandemic. According to their results, Spark offers a much greater speed than MapReduce when processing data and engaging in real-time analysis. This efficiency is particularly important in stock market evaluation as decisions can be made based on recent information. The authors however stress Spark’s filing system deficiencies and a minor, although present, sluggishness with some operations. These three components are still remaining areas in which MapReduce is asserted to perform well in batch-type scenarios.

Benlachimi et al. (2021) went into detail in their work on the comparison of Spark and Hadoop in terms of big data processing, focusing on the structural and competitive components of both. They noted that due to the in-memory processing model specifically designed for Spark, it is much faster than the disk based MapReduce even for non-time sensitive applications. Nevertheless, the cost of using Hadoop MapReduce is still much less and its scaling remains effective even on expansive unmoving datasets; there are therefore cases where MapReduce cannot be ignored because of its cost. The authors point out that unlike traditional computing tasks, iterative computing tasks associated with active data traffic such as machine learning and analysis of streamed data will benefit more from the use of Spark.

Oo and Thein (2019) investigating big data analytics challenges within the context of Scalable Random Forest algorithm looked into utilizing Spark and MapReduce. As further described, the research optimizes SRF hyperparameters as well as carries out dimensionality reduction with the goal of enhancing scalability and precision. Additionally, the authors of the study explore the capabilities of Spark in dealing with large, high-dimensional datasets and explain the advantage of using it for machine learning purposes, especially when the tasks involve multiple iterations. However, the authors acknowledge that both frameworks pose some limitations while dealing with large datasets with many dimensions.

Chaudhari and others(2019) resorted to applying clustering techniques and classification strategies such as k-Means and Support Vector Machines also through Spark and MapReduce. The findings revealed that Spark, alongside the MLlib machine learning library, was able to outperform MapReduce in terms of the baseline classification algorithms, and even more for particularly iterative tasks. The reason being that Spar’s in-Memory processing decreases the I/O costs in contrast to MapReduce which in turn shortens the duration of training and inferencing. Consequently, real-time analytics where prompt insights are paramount can greatly benefit from using Spark.

Benlachimi et al. (2021) were also able to extend further the existing research by performing a comparative analysis of Hadoop and Spark using the Word Count algorithm, benchmarking both technologies for performance evaluation. It was seen that the combination of real-time streaming and Spark is made possible due to the support inhibiting in memory processing. Alternatively, the research noticed that Hadoop is more effective for batch processing. This provides in-depth insight into the potential exhibits of Spark while further increasing the credibility of the existing research. MapReduce on the other end still remains an appealing alternative for businesses with a limited amount of resources and do not require immediate analytics.

Gao et al. (2020) presented the MR-Mafia algorithm. It is a parallel subspace clustering algorithm that employs MapReduce and is designed to cluster large multi-dimensional datasets. Their results and demonstration show that, even though Spark is popular, MapReduce can still be harnessed to complete certain classes of big data workloads, especially those that are heavily dense and have high-dimensional data structures that can leverage MapReduce’s disk based and scalable operations. This research emphasizes that although there are benefits of using Spark in terms of speed and flexibility, there are benefits of using MapReduce in terms of processing large amounts of data without consuming too much memory.

To summarize, the research on the big data processing frameworks such as Apache Spark and Hadoop MapReduce has brought about tremendous change in the field of data analytics and many more problems still remain unsolved. The comparative analysis of the two tools shows that the time efficiency of Spark is clearly superior when it comes to real-life data that requires immediate interpretation, due in particular to the MLlib. These characteristics make Spark very attractive for tasks with many repetitions and low latency requirements such as predicting future stock prices.

While previous studies have explored the performance of Hadoop MapReduce and Apache Spark for big data processing, including applications in stock market analysis, most have either used generic datasets, focused solely on one framework, or implemented traditional machine learning algorithms such as K-Means. For instance, Peddi (n.d.) processed unstructured stock data using MapReduce, and Gopalani and Arora (2015) compared Spark and MapReduce using clustering methods. In contrast, our study uniquely applies the Random Forest algorithm—a supervised learning model—on a real-world stock price dataset, providing a comparative analysis of MapReduce and Apache Spark not only from an execution time perspective, but also prediction accuracy (MSE, RMSE, MAE, R²). Furthermore, unlike prior work, this research emphasizes the impact of **in-memory vs. disk-based** distributed computing frameworks in a predictive financial analytics context, thereby offering **practical insights** for choosing appropriate tools in real-world forecasting applications.

However, the literature also identifies critical challenges that must be addressed. The limited support for deep learning within Spark, file management issues, and latency concerns can hinder its effectiveness in certain scenarios. Furthermore, both frameworks have trouble handling high-dimensional data, which is especially problematic when it comes to stock market analysis, where a lot of variables can affect results.

# Research Methodology

The escalating sophistication of financial markets, combined with the Rising volume of stock price information, makes accurate predictions and analysis challenging. The challenges associated with the scale and speed of data cannot be addressed by conventional computing methods, highlighting the necessity for distributed computing frameworks. This study aimed to tackle these issues and investigate the application of various big data processing frameworks in predicting stock prices. This project aimed to compare the effectiveness of two models, MapReduce, and Apache Spark, in predicting stock prices through the application of the Random Forest algorithm. The methodology for the investigation encompasses a systematic progression of tasks encompassing data gathering, feature extraction, training models, and evaluating models. This study was carried out to predict stock prices by utilising big data concepts, while also highlighting the importance of scalability and computational power.

The current research applied a range of tools and technologies to implement the MapReduce technique, train machine learning models, and evaluate their performance. The use of these technologies ensured efficient data processing, accurate forecasts, and reproducible outcomes. The following is an explanation of the tools used:

1. MapReduce Approach in Python

The MapReduce paradigm was implemented programmatically in Python to simulate the distributed computing process. Instead of relying on Hadoop, the approach was custom-coded and executed in a single-node environment using Jupyter Notebook. Key aspects of this approach include:

* Mapper and Reducer Functions: Python functions were designed to process and aggregate data in a way that mimicked the MapReduce process.
* Sequential Execution: This implementation adhered to the principles of MapReduce for data processing and analysis.

2. Apache Spark with MLlib

Apache Spark was used for in-memory distributed computing. Key features include:

* In-Memory Computation: Spark cached data in memory to improve processing speed.
* Machine Learning with MLlib: Spark’s MLlib library facilitated the training of Random Forest models, making the process scalable and efficient.

3. Python for Implementation

Python served as the primary programming language due to its simplicity and extensive ecosystem. It was used for both MapReduce and Spark implementations.

4. Python Libraries for Data Analysis and Visualization

* Pandas: Used for pre-processing and manipulating the stock price dataset.
* NumPy: Provided numerical operations for feature engineering and analysis.
* scikit-learn: Used to establish baseline machine learning tasks and performance metrics.
* Matplotlib and Seaborn: Employed for visualizing the dataset, model predictions, and performance comparisons.

## Dataset Description

The process of anticipating stock market trends has been found to be a considerable challenge for numerous researchers and analysts. Indeed, there exists a significant interest among investors in the scope of research in stock price forecasting. In pursuit of a sound and prosperous investment, numerous investors exhibit a strong interest in discerning the prospective dynamics of the stock market. Robust and efficient prediction systems for the stock market assist traders, investors, and analysts by offering valuable insights regarding the prospective trajectory of the market.

For this study, we utilized the **Netflix Stock Price Prediction dataset** sourced from Kaggle. This dataset contains historical stock price data (Currency in US Dollars ($)), for 5 years for Netflix Inc. (NFLX), which is used for predictive modelling using the Random Forest algorithm.

**Dataset Details:**

* **Source**: Kaggle – <https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction>
* **Format**: CSV
* **Size**: The dataset consists of *1010* rows and *7* columns.

The dataset consists of multiple columns representing various stock features, including:

1. *Date:* The specific day of trading.
2. *Open:* The stock’s initial price at the beginning of the trading day.
3. *High:* The peak price attained during the trading session*.*
4. *Low:* The minimum price recorded during the trading period*.*
5. *Close*: The stock's final price at the end of the trading day.
6. *Volume:* The total quantity of shares exchanged during the day.

The stock price dataset used in the study contained features such as opening price, closing price, high, low, and volume. The dataset was pre-processed to ensure data quality:

1. Missing values were handled through imputation.
2. Data normalization was performed to ensure all features were on a comparable scale.
3. Feature engineering techniques were applied to improve the predictive power.
4. The dataset was split, using 80% for training and 20% for testing purposes.

A screenshot of a graph

Description automatically generated

Figure 1 Dataset Overview

## Implementation

Based on their popularity and ability to handle distributed data processing, MapReduce and Apache Spark were chosen for comparison. The Random Forest algorithm was selected due to its efficacy in addressing non-linear interactions between dependent and independent variables (Ghayoumi, 2023), as well as its robustness in machine learning tasks.

MapReduce Approach: The MapReduce model was developed using a disk-based, batch-processing pipeline. The mapping phase of the training data consisted of feature extraction and the reducing phase was to aggregate the results to form the Random Forest model. The intermediate results were written to the disk which resulted in higher latency.

Apache Spark Approach: The Spark model utilized in-memory computation along with MLlib for Random Forest training processes. The optimization of iterative computations depended on Data processing through RDDs for caching and storage. Spark distributed its functionality to process data effectively throughout both analytical and machine learning executions.

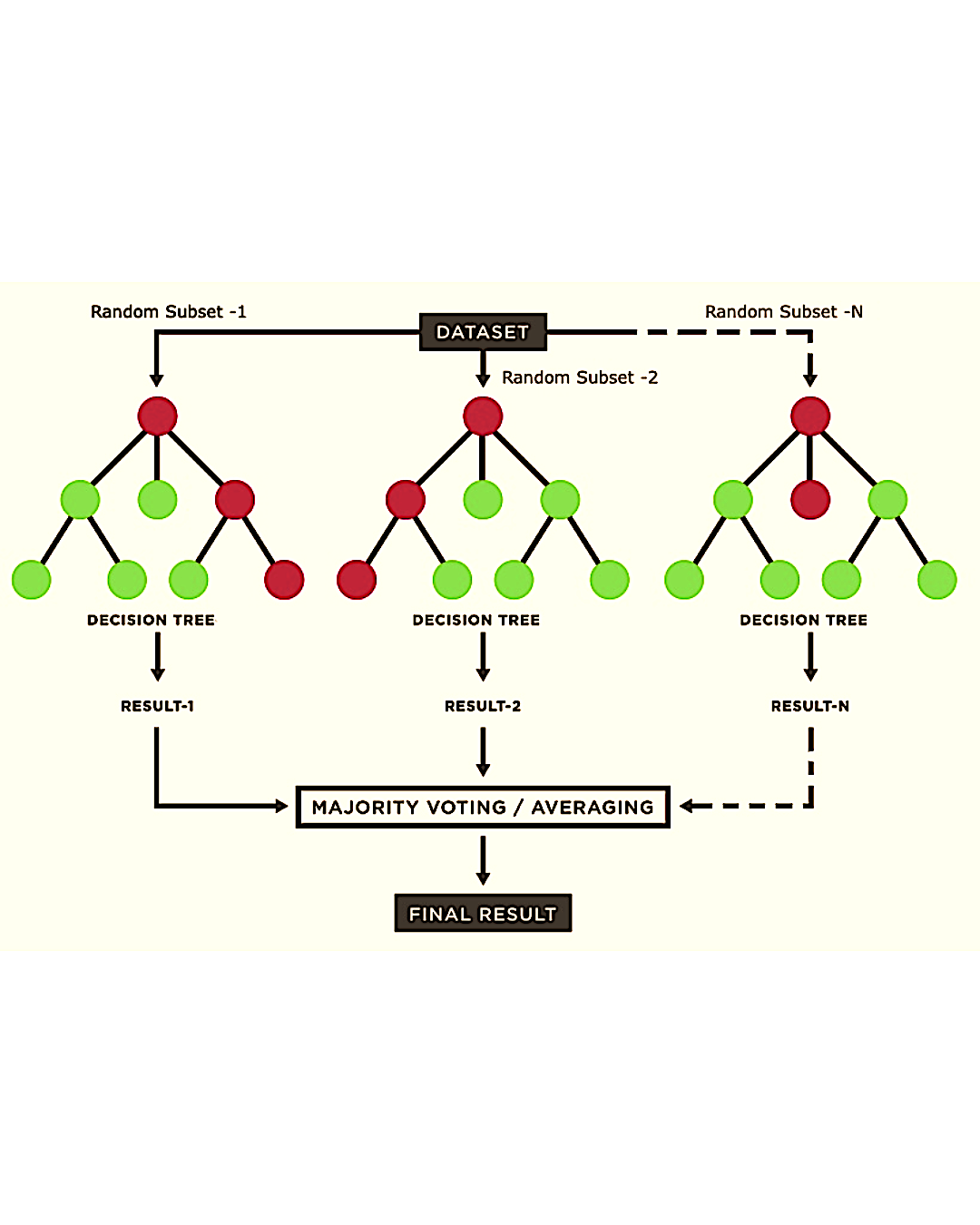


Figure 2 Random Forest Illustration

Random Forest is based on the principle of ensemble learning(Mariprasath, Cheepati, & Rivera, 2024), combining many decision trees to make predictions. Below is its mathematical framework:

1. **Bootstrap Aggregation (Bagging):**

Given a dataset D = {(), (), …, ()} with *n* samples:

* Randomly sample *m* subsets with replacement.
* Each subset is used to train an independent decision tree

1. **Splitting Criterion:**

For regression, at each split, the algorithm minimizes the variance of the target values:

Where *S* is the set of data at the current node, and are subsets for the left and right child nodes, respectively.

1. **Prediction from a Single Tree:**

Each tree predicts the output for the input *x*:

Where *L* is the set of samples in the leaf node where *x* falls, and *y* is the target variable.

1. **Ensemble Prediction (Aggregation):**

For M trees, Random Forest Combines predictions by averaging (for regression):

This mathematical method underpins Random Forest's performance in noisy and large-scale datasets, which makes it ideal for our study.

The predictions of several decision trees constructed from bootstrapped samples and random feature subsets are mathematically combined by the Random Forest technique. This ensemble approach strengthens predictive robustness and accuracy, making it an effective tool for stock price prediction and other predictive modelling tasks. In order to produce dependable results, Random Forest relies on statistical theories like the Law of Large Numbers, as demonstrated by its mathematical backgrounds.

## Evaluation of the Model

In this study, we compare and evaluate the effectiveness of MapReduce and Apache Spark in stock price prediction utilising the Random Forest algorithm employing a number of important assessment indicators. The Mean Squared Error (MSE) evaluates the average squared errors between predicted and observed values, reflecting the proximity of predictions to actual values; lower values signify superior performance. Root Mean Squared Error (RMSE), defined as the square root of Mean Squared Error (MSE), provides a measure that is interpretable and expressed in the same units as the predicted variable. Mean Absolute Error (MAE) computes the average of the absolute differences between predicted and actual values, showing less sensitivity to outliers compared to Root Mean Squared Error (RMSE). The R-squared (R²) value indicates the proportion of variance in the dependent variable that can be explained by the independent variables, with values closer to 1 indicating a better model fit (Wang, Wang, Yin, Gu, Lin, & Zhang, 2024). The metrics collectively offer a thorough assessment of the predictive accuracy and reliability of the two frameworks. To evaluate the Random Forest regression model mathematically, the following metrics were used.

1. **Means Squared Error (MSE):**

MSE quantifies the mean of the squared deviations, between the predicted and actual *(y)* values:

1. **Root Mean Squared Error (RMSE):**

RMSE is the square root of MSE, providing error magnitude in the same units as the target variable:

RSME is useful for interpreting the scale of prediction errors.

1. **R-Squared (Coefficient of Determination):**

R-squared measures the proportion of variance (Ronaghan, 2018), in the actual data explained by the model:

Where is the mean of actual values. ranges from 0 to 1. Higher values indicate better performance.

1. **Mean Absolute Error (MAE):**

MAE calculated the average of the absolute differences, between predicted and actual values (GeeksforGeeks, 2025):

Unlike MSE, MAE is less sensitive to outliers.

The Random Forest model was trained using both frameworks on the pre-processed dataset. Evaluation metrics included:

1. **Root Mean Squared Error (RMSE):** Used to gauge how accurate a forecast is..
2. **Mean Absolute Error (MAE):** To evaluate the average magnitude of errors.
3. **R-squared (R²):** To quantify the model’s goodness-of-fit.
4. **Mean Squared Error (MSE):** To evaluate error sensitivity.

The two frameworks' performance was evaluated using accuracy, processing time, and scalability criteria. The Spark-based version had better RMSE, MAE, and MSE values due to its optimised in-memory processing, whereas MapReduce had higher latency and lesser accuracy due to its disk-based limitations. Furthermore, the Spark-based solution outperformed the iterative calculations required for machine learning processes, making it better suited for applications like Random Forest training. In contrast, the MapReduce technique, while dependable and simple, lacked the efficiency required for fast data processing and real-time applications. These distinctions emphasise Spark's superiority in cases that need both speed and precision in large data analytics.

# Results and Discussion

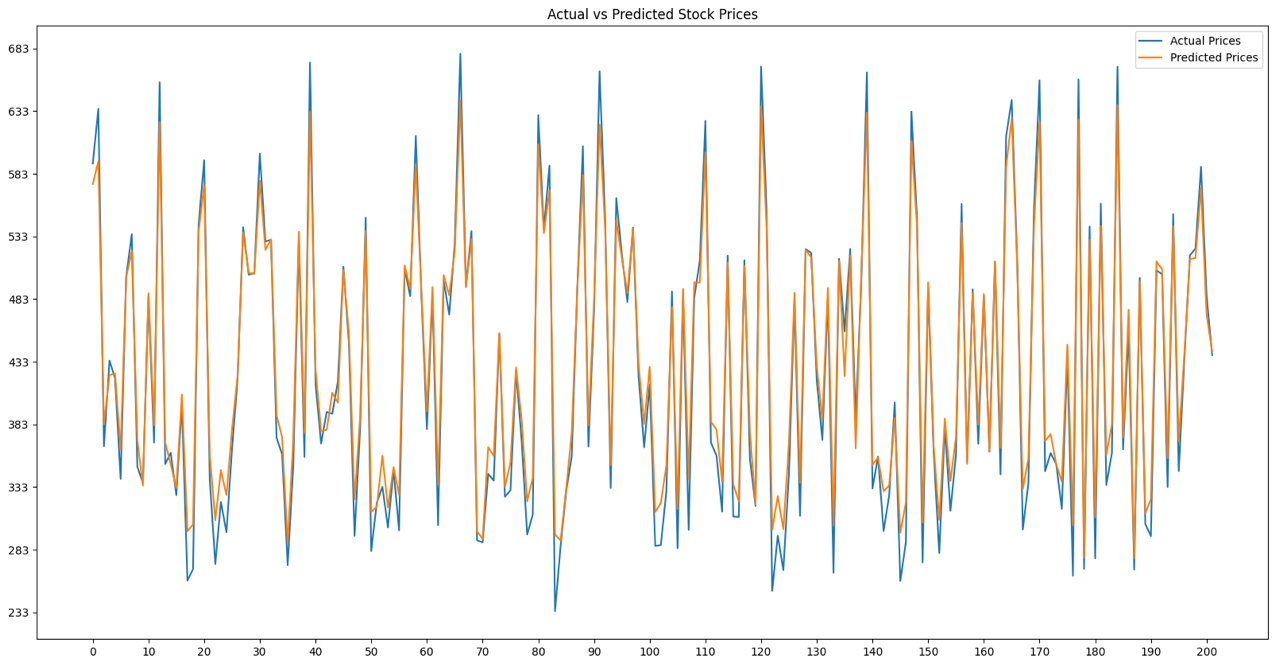
Here’s a direct comparison of **Model A (MapReduce)** and **Model B (Apache Spark)** for stock price prediction using Random Forest Regressor.

|  |  |  |
| --- | --- | --- |
| **Metric** | **Model A: MapReduce** | **Model B: Apache Spark** |
| Stock Price Range (in $) | 250 - 700 | 250 - 700 |
| Mean Squared Error (MSE) | 179.45 | 57.08 |
| Root Mean Squared Error (RMSE) | 13.40 | 7.56 |
| Mean Absolute Error (MAE) | 9.41 | 5.50 |
| R-squared (R²) | 0.9746 | 0.9947 |

**Table 1**: Metric Analysis of MapReduce and Apache Spark

1. **Mean Squared Error (MSE):**

The MSE value of 179.45 from Model A (MapReduce) indicates a large average squared deviation between forecasted stock prices and their actual counterparts. The high prediction errors produced by the model indicates that it will deliver less precise stock price forecasts. On the other hand, Model B (Apache Spark) achieves a much lower MSE of 57.08 which demonstrates superior performance in error reduction. A model with lower MSE offers advantages because its predictions stay nearer to actual values. Model B shows improved predictive accuracy because its MSE value is lower than Model A's MSE value. Higher MSE values indicates that the predictions are distant from actual values which might negatively impact stock price forecasting decisions. Real-world applications like stock market analysis benefit from Model B because it offers precise price prediction capabilities.



**Figure 3**: Model A (MapReduce) - Actual vs Predicted Stock Prices Plot

A graph showing the price of a stock price

Description automatically generated

**Figure 4**: Model B (Apache Spark) - Actual vs Predicted Stock Prices Plot

1. **Root Mean Squared Error (RMSE):**

Model A (MapReduce) exhibits a Root Mean Square Error (RMSE) of 13.40, indicating that its forecasts deviate, on average, by approximately 13.4 units from the actual stock values. The RMSE is represented in the same units as the stock price; so, a greater RMSE indicates poorer predictions. For, Model B (Apache Spark): The RMSE of 7.56 indicates that its predictions are significantly closer to the actual data, averaging a deviation of approximately 7.56 units. The implication is that Model B's considerably reduced RMSE signifies its forecasts are markedly more accurate and dependable than those of Model A. In stock price prediction, little deviations can significantly influence financial decisions; thus, Model B provides a far more accurate instrument for forecasting stock prices.

1. **Mean Absolute Error (MAE):**

The average error measured by Model A (MapReduce) amounts to 9.41 units between predicted and actual stock prices. The MAE measurement enables us to understand prediction accuracy while lower values indicate better accuracy results. The prediction accuracy of Model B (Apache Spark) reaches 5.50 MAE whereas the average prediction errors of Model A (MapReduce) become larger at 9.41 MAE. The lower MAE in Model B indicates that its predictions are more consistently closer to the true stock prices. . Strategic decision-making becomes more effective because decision-makers who use predictions for investment or trading benefit from lower MAE values.

1. **R-squared (R²):**

With an R2 value of 0.9846, Model A explains 97.46% of the variance in stock prices. This indicates that although the model accounts for most of the data variability, approximately 2.54% remains unexplained. Model B (Apache Spark) exhibits a superior R² score of 0.9947, indicating it accounts for 99.47% of the variance in stock prices. This indicates that Model B offers a superior fit for the data, accounting for nearly all the fluctuation in stock prices. The upgraded R² value in Model B demonstrates its superior ability to match the stock price analysis data. The model shows stronger capabilities to represent underlying patterns which makes it more reliable for long-term forecasting applications. Future stock price forecasting depends on precise measurement of variation.

A graph of a distribution of residuals

Description automatically generatedA graph of a distribution of residuals

Description automatically generated

**Figure 5**: Error Distribution - MapReduce **Figure 6**: Error Distribution - Apache Spark

The histogram for Model A displays a distribution that is simply bell-shaped, which indicates that the residuals have been roughly distributed in a normal fashion. Not only is this a positive indicator, but it also indicates the model's assumptions of normality are probably achieved.  
There is a slight positive skew, which is characterised by a tail that extends to the right. This may hint at a few larger errors that are in the positive direction. It is not, however, a serious issue. The residual distribution of Model A appears to be adequate, suggesting that the model effectively captures majority of the patterns present within the data.

The histogram for Model B has a bell-like form that is almost perfectly symmetrical around zero. This is highly noticeable and is a good sign, indicating that all aspects of the model's assumptions of normality are likely met. The residuals are tightly grouped around zero, which indicates that the model's predictions are often quite correct. This characteristic is referred to as concentration. All things considered; the residual distribution of Model B is excellent. It indicates that the model is well-fitted, with a low number of prediction errors and a high adherence to the assumption of normality.

The experimental results of this study reveal that Apache Spark consistently outperformed MapReduce in stock price prediction tasks, particularly in terms of processing speed, lower error metrics, and model scalability. These findings align with those of Ibtisum et al. (2023), who also observed Spark’s advantage in batch processing and iterative algorithms due to its in-memory computation model. Similarly, Gopalani and Arora (2015) concluded that Spark’s performance significantly surpassed MapReduce when executing the K-Means algorithm, reinforcing Spark’s suitability for machine learning workloads. Our results also support the observations made by Dr. Prasadu Peddi (2019), who demonstrated that while MapReduce is capable of handling unstructured stock data, its disk-based nature limits efficiency. Compared to these previous studies, our work extends the discussion specifically into the financial forecasting domain using the Random Forest algorithm and provides quantitative performance comparisons using MSE, RMSE, MAE, and R² metrics. This deeper focus not only validates earlier insights but also offers empirical benchmarks for stock prediction tasks.

# Conclusion

This project examined the efficacy of the Random Forest technique when combined with distributed computing frameworks, specifically MapReduce and Apache Spark, for the purpose of stock price prediction. Using big data approaches, we are efficiently with an eye towards scalability processed stock price data. The findings show clear variations between the two models; Spark shows better accuracy, and simplicity of use than the other one. Model B is clearly more accurate than Model A across all metrics (MSE, RMSE, MAE, and R²). It produces predictions that are closer to the actual stock prices, and with 99.47% variance explained, it is much better at capturing the trends in the data. This degree of precision is especially significant in financial forecasting, where small variations can result in substantial financial consequences.

Model A (MapReduce) exhibited higher latency due to its disk-based operations, even though it was dependable and most suitable for batch processing. The values of MSE and MAE for the model trained using MapReduce were higher, indicating its efficacy in managing iterative machine learning processes. In contrast, Apache Spark's in-memory computing markedly diminished processing durations, resulting in reduced error rates and enhanced R-squared values. The integration of Spark with MLlib and its support for iterative algorithms has been beneficial for optimising machine learning activities such as Random Forest training.

This project underscores the importance of choosing appropriate frameworks for big data analytics. While MapReduce remains a viable option for simpler batch processing tasks, Spark’s versatility and performance make it the preferred choice for complex, iterative, and real-time workflows. Future work could extend this study to real-time stock prediction and compare additional frameworks, further advancing the field of big data analytics for financial applications.

This study correctly compared MapReduce and Apache Spark's ability to use the Random Forest method to predict stock prices. Even so, there is still a lot of room for growth and more study. In the future, researchers may focus on improving the Random Forest model's hyper-parameters to get even better results. For that purpose, Methods like Grid Search, Random Search, or Bayesian Optimisation could be used. Through further development the study will address important issues to build better stock price prediction algorithms that improve accuracy and efficiency while also advancing knowledge in big data analytics.

This research offers practical insights for data scientists, financial analysts, and engineering teams working with large-scale prediction tasks. The results support Apache Spark as a better-suited framework for real-time stock price forecasting, enabling more accurate and faster predictions. Companies in the finance and tech sectors can use these findings to optimize their infrastructure choices for data-driven decision-making.

From a scientific standpoint, this work contributes to the comparative literature on distributed computing frameworks in big data analytics. By applying Random Forest models to real-world financial data, the study provides a benchmark for performance evaluation in predictive modelling. It also encourages future research into integrating other machine learning models, testing hybrid computing systems, and enhancing prediction through hyperparameter tuning using methods such as Grid Search or Bayesian Optimization.

**Future Work**

Further research may focus on optimizing the Random Forest algorithm's parameters to enhance model performance. While this study focused on MapReduce and Apache Spark, additional distributed computing frameworks such as **Apache Flink**, **Dask**, or **Ray** could be included in future benchmarks to provide a broader performance comparison. Each framework offers unique advantages in terms of latency, fault tolerance, and real-time processing. This progression would deepen understanding and drive innovation in the intersection of machine learning and distributed computing for financial applications.

# Appendix A

1. Coder, J. (n.d.). Netflix stock price prediction dataset. Kaggle. <https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction>
2. *Evaluation metrics*. (n.d.). FasterCapital. <https://fastercapital.com/startup-topic/evaluation-metrics.html#differentiating-performance-evaluation-metrics1>
3. Ibtisum, S., Bazgir, E., Rahman, S., & Hossain, S. (2023). A comparative study of MapReduce and Apache Spark in big data analytics. World Journal of Advanced Research and Reviews, 20(1), 100–105. <https://doi.org/10.30574/wjarr.2023.20.1.2174>
4. Gopalani, S., & Arora, R. (2015). Comparing Apache Spark and MapReduce with performance analysis using K-means. International Journal of Computer Applications, 113(1), 8–11.
5. Peddi, P. (2019). An efficient analysis of stocks data using MapReduce. Journal of Applied Science and Computations, 6(1), 22–34.
6. Ghayoumi, M. (2023). Generative adversarial networks in practice. CRC Press. <https://api.pageplace.de/preview/DT0400.9781003805496_A47077316/preview-9781003805496_A47077316.pdf>
7. Rajapandiyan, A., & Subbaiyan, T. (2025). Commitment-driven penalty mechanism with dynamic pricing for V2V energy trading: A multi-armed bandit reinforcement learning and game theoretic approach. *Energy, 322*, 135742. <https://doi.org/10.1016/j.energy.2025.135742>
8. Khamaru, A., & Hiwarkar, T. (2022). A dynamics of machine learning on Map-Reduce architecture for enhancing big data analysis performance. International Journal of Computer Science and Mobile Computing, 11(11), 108–117. <https://ijcsmc.com/docs/papers/November2022/V11I11202213.pdf>
9. Bhadoria, R. S., Pandey, M. K., & Kundu, P. (2021). RVFR: Random vector forest regression model for integrated & enhanced approach in forest fires predictions. Ecological Informatics, 66, 101471. <https://doi.org/10.1016/j.ecoinf.2021.101471>
10. Barvaliya, V. (2024, June 20). *How Netflix uses Apache Spark: A technical deep dive*. Medium. <https://medium.com/data-engineer/how-netflix-uses-apache-spark-a-technical-deep-dive-096ee486f54b>
11. Wang, W. (2024). *Comparative analysis of ARIMA, Random Forest, and LSTM models for Mercedes-Benz stock price prediction*. *Advances in Economics, Management and Political Sciences, 134*, 1–8. <https://doi.org/10.1201/9781003563945>
12. Mariprasath, T., Cheepati, K. R., & Rivera, M. (2024). *Practical guide to machine learning, NLP, and generative AI: Libraries, algorithms, and applications*. <https://doi.org/10.1201/9781003563945>
13. Amalina, F., Hashem, I. A. T., Azizul, Z. H., Fong, A. T., Firdaus, A., Imran, M., & Anuar, N. B. (2019). Blending big data analytics: Review on challenges and a recent study. IEEE Access, 8, 3629–3645. <https://doi.org/10.1109/access.2019.2923270>
14. Narayanan, U., Veettil, N. P., Krishnankutty, R. T., Sunny, L. E., & Paul, V. (2024). Mitigating privacy and security risks in the era of big data: A comprehensive framework of best practices and protocols. Journal of Computer Science, 20(9), 1121–1145. <https://doi.org/10.3844/jcssp.2024.1121.1145>
15. Wang, D., Wang, L., Yin, H., Gu, G., Lin, Z., & Zhang, W. (2024). Graph-Driven Models for Gas Mixture Identification and Concentration Estimation on Heterogeneous Sensor Array Signals. *arXiv preprint arXiv:2412.13891*.
16. GeeksforGeeks. (2025, February 27). *Supervised vs unsupervised learning*. GeeksforGeeks. <https://www.geeksforgeeks.org/supervised-unsupervised-learning/?ref=header_search>
17. Apache Software Foundation. (n.d.). Apache cluster model. Apache Spark. <https://spark.apache.org/docs/3.5.3/cluster-overview.html>
18. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., ... & Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI'12). <https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf>
19. Oo, M. C. M., & Thein, T. (2019). Hyperparameters optimization in scalable random forest for big data analytics. https://doi.org/10.1109/CCOMS.2019.8821752
20. Kumar, M. R. S., & Mohan, H. S. (2024). Improving big data analytics data processing speed through MapReduce scheduling and replica placement with HDFS using genetic optimization techniques. Journal of Intelligent & Fuzzy Systems, 37(1), 1–14. <https://doi.org/10.3233/JIFS-240069>
21. Benlachimi, Y., & Hasnaoui, M. L. (2021). Big data and Spark: Comparison with Hadoop. https://doi.org/10.1109/WorldS450073.2020.9210353
22. Gupta, Y., & Sharma, N. (2021). Propositional aspect between Apache Spark and Hadoop MapReduce for stock market data. https://doi.org/10.1109/ICISS49785.2020.9315977
23. Birjali, M., Beni-Hssane, A., & Erritali, M. (2020). A Spark-based big data analytics framework for competitive intelligence. https://doi.org/10.1109/ICSSD47982.2019.9002837
24. Gupta, Y. K., & Kumari, S. (2020). A study of big data analytics using Apache Spark with Python and Scala. In Proceedings of the International Conference on Intelligent Sustainable Systems (ICISS). <https://doi.org/10.1109/ICISS49785.2020.9315863>
25. Gao, Z., Fan, Y., Niu, K., & Ying, Z. (2020). MR-Mafia: Parallel subspace clustering algorithm based on MapReduce for large multi-dimensional datasets. https://doi.org/10.1109/BigComp.2018.00045
26. Ronaghan, S. (2018, May 12). The mathematics of decision trees, random forest, and feature importance in Scikit-learn and Spark. Towards Data Science. <https://medium.com/towards-data-science/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3>
27. Khamaru, A., & Hiwarkar, T. (2022). A dynamics of machine learning on Map-Reduce architecture for enhancing big data analysis performance. International Journal of Computer Science and Mobile Computing, 11(11), 108–117. <https://ijcsmc.com/docs/papers/November2022/V11I11202213.pdf>
28. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... & Zaharia, M. (2016). MLLib: Machine learning in Apache Spark. Journal of Machine Learning Research, 17(1), 1235–1241. <https://www.jmlr.org/papers/volume17/15-237/15-237.pdf>
29. Chaudhari, R. S., Patil, S. S., & Ghorpade, S. J. (2019). Classification and clustering methods along with MapReduce, Apache Spark: A study. International Journal of Research and Analytical Reviews. <http://www.ijrar.org/papers/IJRAR2AA1352>