Acta Informatica Pragensia
2025, Volume 14, Issue 3, 460–473
https://doi.org/10.18267/j.aip.275

PRAGUE UNIVERSITY OF ECONOMICS AND BUSINESS

Article Open Access

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

Chitra Joshi, Chitrakant Banchorr, Omkaresh Kulkarni, Kirti Wanjale

Vishwakarma Institute of Information Technology, Pune, India

Corresponding author: Chitra Joshi (chitra.joshi@viit.ac.in)

Editorial Record

First submission received:
February 16, 2025

Revisions received:
April 18, 2025
June 6, 2025

Accepted for publication:
June 8, 2025

Academic Editor:
Zdenek Smutny
Prague University of Economics and Business, Czech Republic

This article was accepted for publication by the Academic Editor upon evaluation of the reviewers' comments.

How to cite this article:
Joshi, C., Banchorr, C., Kulkarni, O., & Wanjale, K. (2025). In-Memory Versus Disk-Based Computing with Random Forest for Stock Analysis: A Comparative Study. *Acta Informatica Pragensia*, 14(3), 460–473. https://doi.org/10.18267/j.aip.275

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 such as 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 become 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 uses 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 value were used to evaluate the goodness of fit of the models.

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 optimized 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 large amounts 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 big data machine learning applications.

Index Terms

Apache Spark; MapReduce; Big data; Random forest; Performance comparison; Data processing; In-memory processing; Disk-based processing.

1 INTRODUCTION

In the age of digital change, the amount of data created every day is increasing exponentially, resulting in what is called "big data". This refers to large sets of data that are challenging to manage and analyse using conventional data management tools and techniques (Badshah et al., 2024). Big data analysis is vital for any organization in today's world irrespective of the industry, be it finance, health, e-commerce, etc. (Tosi et al., 2024). Since global organizations are now trying to capitalize 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 take advantage of distributed computing across multiple servers. These tools facilitate parallel processing, enabling efficient execution of complex computations on vast volumes of data. Among the most prominent frameworks supporting such capabilities are MapReduce and Apache Spark, both are now core components in big data analytics (Badshah et al., 2024) thanks to their dependable and effective performance in processing massive datasets.

Stock price predictions are naturally difficult because financial data are changeable, abundant and fast-moving (Kulkarni et al., 2025). 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 such as 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 (Ahmed et al., 2020), 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 widely employed for large-scale data processing systems. The 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 (Demirbaga et al., 2024). While MapReduce is designed for batch processing, it is also designed for the fundamental principle of information being stored on a disk. This produces a resource lag particularly in situations which repeatedly work with the same data, which in turn affects efficiency especially in use cases which employ machine learning computational algorithms (Hedayati et al., 2023).

The MapReduce framework works by dividing tasks into two main phases: the map phase, where the input data are 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 (Abdalla, 2022). This system exhibits significant scalability and resilience to faults, rendering it ideal for 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 (Hedayati et al., 2023).

Apache Spark has been developed as a challenging alternative to traditional frameworks such as MapReduce (Salloum et al., 2016). 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 (Apache Software Foundation, n.d.). For these reasons, there are many situations where Spark is more competent 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 making use of in-memory computing. Spark uses resilient distributed datasets (RDDs) to store intermediate results in memory, which minimizes the necessity for disc reads and writes, thereby significantly accelerating computation (Aziz et al., 2019). 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 (Salloum et al., 2016).

The objective of this study is to conduct an empirical analysis and performance evaluation of both the Apache Spark and the MapReduce frameworks in terms of their implementation of random forest algorithms for stock price predictions. With an emphasis on performance-based metrics such as scalability, processing speed and the predictive accuracy of each integrated development environment, we can present 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:

- **RQ1:** 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?
- **RQ2:** 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?
- **RQ3:** How do the two frameworks differ in terms of residual error distribution and model fit, as reflected by R-squared values and histogram analysis?

The rest of the paper is structured as follows: Section 2 surveys associated studies in the areas of stock price prediction and big data processing. The experimental setting and performance metrics used 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. Finally, the work is summarised in Section 5.

2 LITERATURE REVIEW

In recent years, the quantity of data has increased at a breathtaking rate, making it necessary to have big data processing frameworks that are efficient. Two of the more popular frameworks are Apache Spark and MapReduce, both of which have their own advantages. Research in this domain includes 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 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 15 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.

Ibtisum et al. (2023) conducted a comparative analysis of MapReduce and Apache Spark for processing large-scale healthcare datasets. Their findings showed that the memory-centric processing of Spark 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 employing random forest for stock price prediction, offering a deeper insight into model accuracy and error analysis.

A comparative study by Gopalani and Arora (2015) highlighted the differences between MapReduce and Apache Spark, focusing on their performance utilizing *k*-means clustering for big data insights. The study concluded that Spark significantly outperforms MapReduce due to its in-memory processing capability, enabling accelerated and more efficient data operations. While the paper centred on clustering rather than predictive modelling, it reinforced 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 HDFS and MapReduce models offer scalable solutions for both storage and processing. Through the implementation of MapReduce jobs on unstructured stock datasets, the work emphasized the ability of the framework to manage large-scale financial data efficiently and encouraged further exploration of optimizing unstructured data processing using big data technologies.

Lulli et al. (2019) proposed ReForeSt, a novel distributed and memory-optimized implementation of the random forest (RF) algorithm, designed to address the computational and scalability challenges inherent in big data environments. Taking advantage of the distributed in-memory processing capabilities of Apache Spark, the framework introduces advanced data partitioning strategies and a greedy model selection mechanism to optimize training efficiency while minimizing memory consumption. A key contribution of their work lies in the integration of random rotation ensembles (RRE), which apply random transformations to the feature space prior to tree construction, thereby enhancing decision boundary delineation and improving generalization performance. Empirical evaluations conducted on large-scale datasets, including Covertype and Higgs, demonstrate the superiority of ReForeSt in terms of classification accuracy, resource utilization and training time compared to existing Spark-based solutions such as MLlib. This study underscores the significance of combining algorithmic innovations with distributed computing architectures to facilitate robust and scalable machine learning in high-dimensional data scenarios.

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 MapReduce, which makes Spark more appropriate for use during a number of machine learning cycles. The authors demonstrated how the Spark principles allow it to excel in tasks where data need to be written and read repetitively, meaning that it has a strong foundation for being used in real-time and interactive analytics.

Meng et al. (2016) examined the scalability of Spark MLlib library and highlighted its distinctive advantages over other machine learning libraries. Their study emphasized that the seamless integration of MLlib with Spark significantly improves the execution of iterative algorithms such as random forest and *k*-means on large-scale datasets. Moreover, the paper outlined how MLlib surpasses MapReduce, which lacks native support for machine learning tasks, thereby underscoring the suitability of Spark 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.

Adil et al. (2019) 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 such as competitive intelligence because of the demand for instantaneous data processing. However, the authors add, the deep learning libraries of Spark 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 (2020) 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. However, the authors stressed the filing system deficiencies of Spark 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 MapReduce 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 applications that are not time-sensitive. Nevertheless, the cost of using Hadoop MapReduce is still much lower 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) investigated big data analytics challenges within the context of scalable random forest algorithms and looked into utilizing Spark and MapReduce. Their research optimized SRF hyperparameters and carried out dimensionality reduction with the goal of enhancing scalability and precision. Additionally, the study explored the capabilities of Spark in dealing with large, high-dimensional datasets and explained the advantage of

using it for machine learning purposes, especially when the tasks involve multiple iterations. However, the authors acknowledged that both frameworks pose some limitations when dealing with large datasets with many dimensions.

Chaudhari et al. (2019) employed clustering techniques and classification strategies such as *k*-means and support vector machines through Spark and MapReduce. Their 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 is that the in-memory processing of Spark 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.

Gao et al. (2018) 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 showed that, even though Spark is popular, MapReduce can still be harnessed to complete certain classes of big data workloads, especially those that are very dense and have high-dimensional data structures that can take advantage of the disk-based and scalable operations of MapReduce. 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 into big data processing frameworks such as Apache Spark and MapReduce has brought about tremendous change in the field of data analytics, but 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 require 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 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 (2019) 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 in terms of not only execution time, but also prediction accuracy (MSE, RMSE, MAE, R^2). Furthermore, unlike prior work, this research emphasizes the impact of in-memory versus 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.

3 RESEARCH METHODOLOGY

The increasing sophistication of financial markets, combined with the growing 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 aims to tackle these issues and investigate the application of various big data processing frameworks in predicting stock prices. This project aims 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 of models, and evaluation of models. This study is carried out to predict stock prices by utilizing big data concepts, while also highlighting the importance of scalability and computational power.

The current research has 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 has 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 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.

3.1 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, see Data Availability statement or (Shah**,** 2022). 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:** (Shah**,** 2022)
- **Format:** CSV
- **Size:** The dataset consists of 1010 rows and 7 columns.

The dataset consists of multiple columns representing various stock features (see also Figure 1), including:

Date: The specific day of trading.

Open: The initial price of the stock at the beginning of the trading day.

High: The peak price attained during the trading session.

Low: The minimum price recorded during the trading period.

Close: The final price of the stock at the end of the trading day.

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 that 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.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

Figure 1. *Dataset overview.*

3.2 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 (Lulli et al., 2019), as well as its robustness in machine learning tasks, see Figure 2.

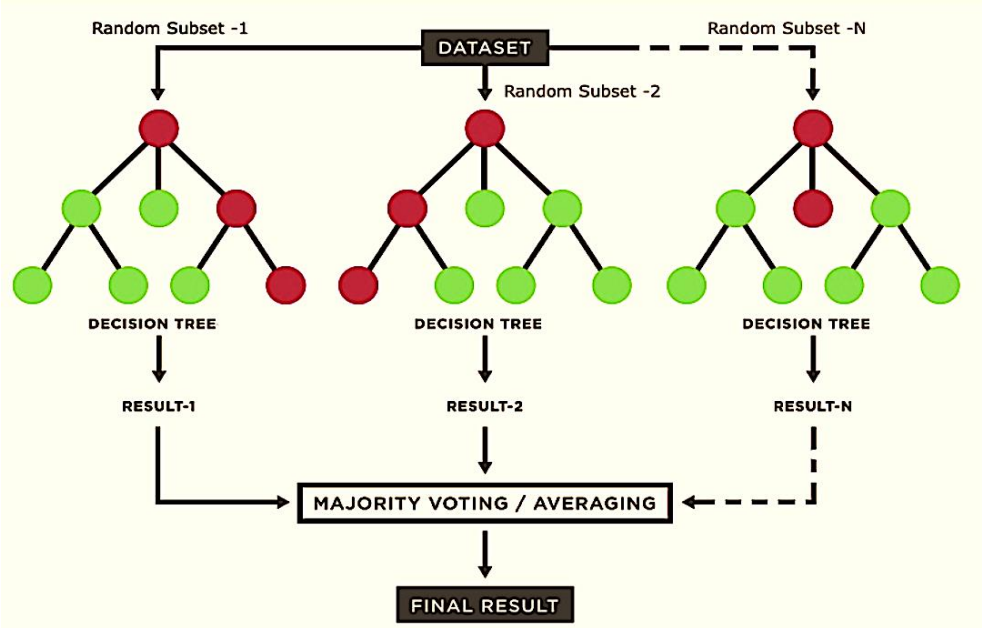

Figure 2. *Random forest illustration.*

MapReduce approach: The MapReduce model was developed using a disk-based, batch-processing pipeline. The mapping phase for the training data consisted of feature extraction and the reducing phase aimed 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.

Random forest is based on the principle of ensemble learning (Ronaghan, 2018), combining many decision trees to make predictions. Below is its mathematical framework:

1. **Bootstrap aggregation (bagging):**
   Given a dataset D = $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ with n samples:
   - Randomly sample m subsets $D_i \subseteq D$ with replacement.
   - Each subset D_i is used to train an independent decision tree $T_i(x)$.

2. **Splitting criterion:**
   For regression, at each split, the algorithm minimizes the variance of the target values:
   $$Variance\ reduction = Var(S) - \left(\frac{|S_L|}{|S|} Var(S_L) + \frac{|S_R|}{|S|} Var(S_R)\right)$$
   Where S is the set of data at the current node, S_L and S_R are subsets for the left and right child nodes, respectively.

3. **Prediction from a single tree:**
   Each tree T_i predicts the output for the input x:
   $$T_i(x) = \frac{1}{|L|} \sum_{x \in L} y$$
   Where L is the set of samples in the leaf node where x falls, and y is the target variable.

4. **Ensemble prediction (aggregation):**
   For M trees, random forest combines predictions by averaging (for regression):
   $$\hat{y} = \frac{1}{M} \sum_{i=1} T_i(x)$$

This mathematical method underpins the performance of random forest 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 such as the law of large numbers, as demonstrated by its mathematical backgrounds.

3.3 Model evaluation

In this study, we compare and evaluate the effectiveness of MapReduce and Apache Spark in stock price prediction using 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 (Omar et al., 2022). 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 lower sensitivity to outliers compared to root mean squared error (RMSE). The R-squared (R^2) 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. 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. **Mean squared error (MSE):**
   MSE quantifies the mean of the squared deviations, between the predicted ($\hat{y}$) and actual (y) values:
   $$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2. **Root mean squared error (RMSE):**
   RMSE is the square root of MSE, providing error magnitude in the same units as the target variable:
   $$RMSE = \sqrt{MSE}$$
   RSME is useful for interpreting the scale of prediction errors.

3. **R-squared (coefficient of determination):**
   R-squared measures the proportion of variance (Ronaghan, 2018) in the actual data explained by the model:
   $$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}\left(y_i - \underline{y_i}\right)^2}$$
   Where $\underline{y}$ is the mean of actual values. R^2 ranges from 0 to 1. Higher values indicate better performance.

4. **Mean absolute error (MAE):**
   MAE calculates the average of the absolute differences between predicted and actual values:
   $$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$
   Unlike MSE, MAE is less sensitive to outliers.

The random forest model was trained using both frameworks on the pre-processed dataset. The evaluation metrics included:

1. **RMSE:** Used to gauge how accurate a forecast is.
2. **MAE:** To evaluate the average magnitude of errors.
3. **R²:** To quantify the goodness-of-fit of the model.
4. **MSE:** To evaluate error sensitivity.

The performance of the two frameworks was evaluated using accuracy, processing time and scalability criteria. The Spark-based version had better RMSE, MAE and MSE values due to its optimized 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 such as 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 emphasize the superiority of Spark in cases that need both speed and precision in big data analytics.

4 RESULTS AND DISCUSSION

Table 1 presents a direct comparison of Model A (MapReduce) and Model B (Apache Spark) for stock price prediction using the 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 (R^2)	0.9746	0.9947

Table 1. *Metric analysis of MapReduce and Apache Spark.*

Mean squared error (MSE):

The MSE value of 179.45 for Model A (MapReduce) indicates a large average squared deviation between forecasted stock prices and their actual counterparts. The large prediction errors produced by the model indicate that it will deliver less precise stock price forecasts. On the other hand, Model B (Apache Spark) achieved 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, figures 3 and 4. Model B shows improved predictive accuracy because its MSE value is lower than that of Model A. Higher MSE values indicate that the predictions are distant from actual values, which might negatively affect stock price forecasting decisions. Real-world applications such as stock market analysis benefit from Model B because it offers precise price prediction capabilities.

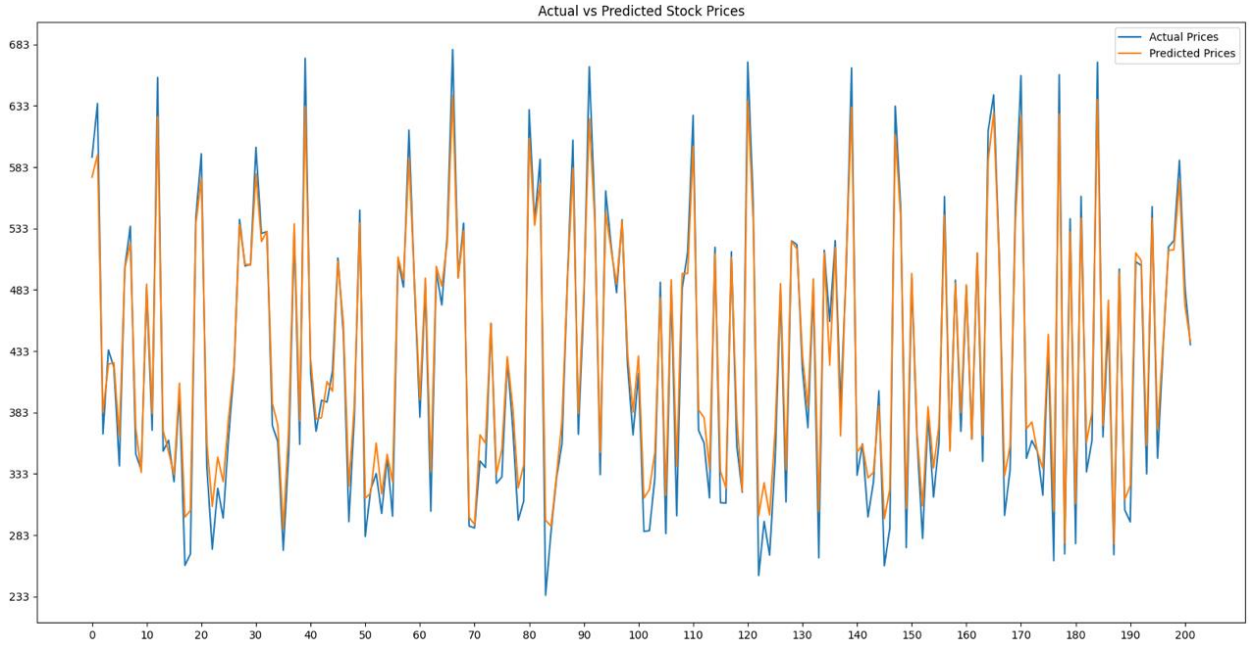

Figure 3. *Model A (MapReduce) – Comparison of actual and predicted stock prices. The X-axis represents time steps in the test dataset; the Y-axis shows the stock prices.*

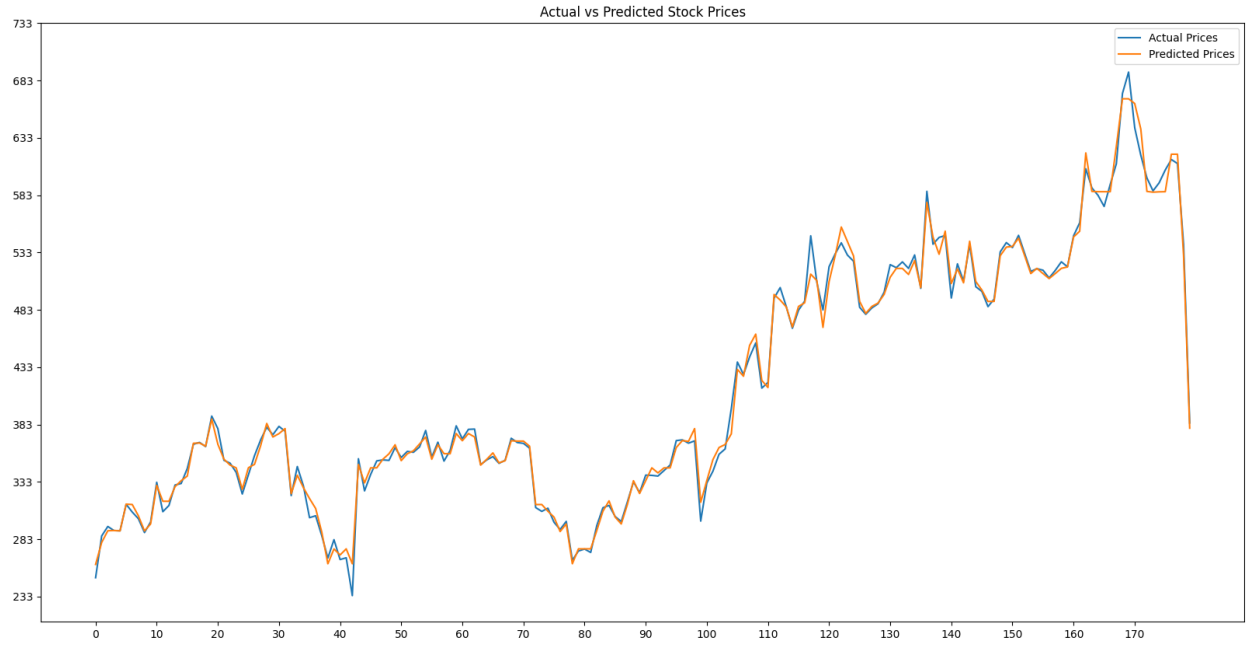

Figure 4. *Model B (Apache Spark) – Comparison of actual and predicted stock prices. The X-axis represents time steps in the test dataset; the Y-axis shows the stock prices.*

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 the considerably reduced RMSE of Model B signifies that 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.

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; lower values indicate better accuracy results. The prediction accuracy of Model B (Apache Spark) reaches a MAE of 5.50, whereas the average prediction errors of Model A (MapReduce) become larger at 9.41. 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.

R-squared (R^2):

With an R^2 value of 0.9846, Model A explains 98.46% of the variance in stock prices, see figures 5a and 5b. 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 R^2 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 higher R^2 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.

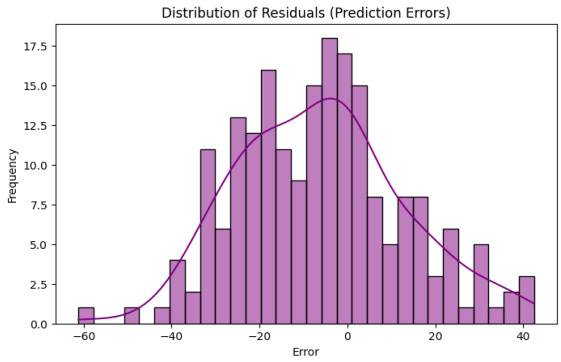

Figure 5a. *Error distribution – MapReduce.*

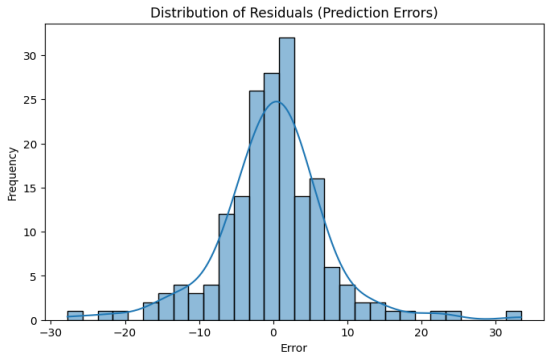

Figure 5b. *Error distribution – Apache Spark.*

The X-axis represents the difference between actual and predicted prices (error); the Y-axis indicates the frequency of these errors.

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 that the assumptions of normality in the model are probably achieved. There is a slight positive skew, which is characterized 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 significant issue. The residual distribution of Model A is adequate, suggesting that the model effectively captures the majority of the patterns present within the data.

The histogram for Model B has a bell-like form that is perfectly symmetrical around zero. This is highly noticeable and is a good sign, indicating that all aspects of the assumptions of normality in the model are likely met. The residuals are tightly grouped around zero, which indicates that the model 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 the advantage of Spark in batch processing and iterative algorithms due to its in-memory computation model. Similarly, Gopalani and Arora (2015) concluded that the performance of Spark significantly surpassed MapReduce when executing the *k*-means algorithm, reinforcing the suitability of Spark for machine learning workloads. Our results also support the observations made by 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 R^2 metrics. This deeper focus not only validates earlier insights but also offers empirical benchmarks for stock prediction tasks.

5 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, they are efficient with an eye towards scalability of processed stock price data. The findings show clear variations between the two models; Spark showed better accuracy and simplicity of use than the other one. Model B was clearly more accurate than Model A across all metrics (MSE, RMSE, MAE and R^2). It produced predictions that were closer to the actual stock prices and with 99.47% variance explained, it was 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, the in-memory computing of Apache Spark markedly reduced processing durations, resulting in lower error rates and higher R-squared values. The integration of Spark with MLlib and its support for iterative algorithms was beneficial for optimizing 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, the versatility and performance of Spark 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 the ability of MapReduce and Apache Spark 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 hyper-parameters of the random forest model to get even better results. For that purpose, methods such as grid search, random search or Bayesian optimization 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.

5.1 Future work

Further research may focus on optimizing the parameters of the random forest algorithm 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.

ADDITIONAL INFORMATION AND DECLARATIONS

Conflict of Interests: The authors declare no conflict of interest

Author Contributions: C.J.: Data curation, Formal analysis, Methodology, Software, Writing – Original draft preparation, Writing – Reviewing and Editing. C.B.: Conceptualization, Project administration, Writing – Reviewing and Editing, Validation. O.K.: Visualisation, Supervision. K.W.: Resources.

Statement on the Use of Artificial Intelligence Tools: The authors declare that they didn't use artificial intelligence tools for text or other media generation in this article.

Data Availability: The data that support the findings of this study are openly available in Kaggle at https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction.

REFERENCES

Abdalla, H. B. (2022). A brief survey on big data: Technologies, terminologies and data-intensive applications. *Journal of Big Data*, 9, Article 107. https://doi.org/10.1186/s40537-022-00659-3

Adil, B., Abdelhadi, F., Mohamed, B., & Haytam, H. (2019). A Spark based big data analytics framework for competitive intelligence. In *2019 1st International Conference on Smart Systems and Data Science (ICSSD)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICSSD47982.2019.9002837

Ahmed, N., Barczak, A. L. C., Susnjak, T., & Rashid, M. A. (2020). A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench. *Journal of Big Data*, *7*(1), Article 110. https://doi.org/10.1186/s40537-020-00388-5

Apache Software Foundation. (n.d.). Apache cluster model. Apache Spark. https://spark.apache.org/docs/3.5.3/cluster-overview.html

Aziz, K., Zaidouni, D., & Bellafkih, M. (2019). Leveraging resource management for efficient performance of Apache Spark. *Journal of Big Data*, 6(1), Article 78. https://doi.org/10.1186/s40537-019-0240-1

Badshah, A., Daud, A., Alharbey, R., Banjar, A., Bukhari, A., & Alshemaimri, B. (2024). Big data applications: overview, challenges and future. *Artificial Intelligence Review*, 57(11), Article 290. https://doi.org/10.1007/s10462-024-10938-5

Barvaliya, V. (2024, 20 June). How Netflix uses Apache Spark: A technical deep dive. *Medium.com*. https://medium.com/data-engineer/how-netflix-uses-apache-spark-a-technical-deep-dive-096ee486f54b

Benlachmi, Y., & Hasnaoui, M. L. (2020). Big data and Spark: Comparison with Hadoop. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4),* (pp. 811–817). IEEE. https://doi.org/10.1109/worlds450073.2020.9210353

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*, 7(4), 593–596.

Demirbaga, Ü., Aujla, G. S., Jindal, A., & Kalyon, O. (2024). Big data analytics platforms. In *Big Data Analytics* (pp. 79–126). Springer. https://doi.org/10.1007/978-3-031-55639-5_5

Gao, Z., Fan, Y., Niu, K., & Ying, Z. (2018). MR-Mafia: Parallel Subspace Clustering Algorithm Based on MapReduce for Large Multi-dimensional Datasets. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, (pp. 257–262). IEEE. https://doi.org/10.1109/bigcomp.2018.00045

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.

Gupta, Y. K., & Sharma, N. (2020). Propositional Aspect between Apache Spark and Hadoop Map-Reduce for Stock Market Data. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS),* (pp. 479–483). IEEE. https://doi.org/10.1109/iciss49785.2020.9315977

Hedayati, S., Maleki, N., Olsson, T., Ahlgren, F., Seyednezhad, M., & Berahmand, K. (2023). MapReduce scheduling algorithms in Hadoop: a systematic study. *Journal of Cloud Computing Advances Systems and Applications*, *12*(1), Article 143. https://doi.org/10.1186/s13677-023-00520-9

Ibtisum, N. S., Bazgir, N. E., Rahman, N. S. M. A., & Hossain, N. S. M. S. (2023). A comparative analysis of big data processing paradigms: Mapreduce vs. apache spark. *World Journal of Advanced Research and Reviews*, *20*(1), 1089–1098. https://doi.org/10.30574/wjarr.2023.20.1.2174

Kulkarni, M. S., Bharathi, S. V., Perdana, A., & Kilari, D. (2025). A quest for Context-Specific Stock Price Prediction: A comparison between time series, machine learning and deep learning models. *SN Computer Science*, 6(4), Article 335. https://doi.org/10.1007/s42979-025-03848-y

Lulli, A., Oneto, L., & Anguita, D. (2019). Mining big data with random forests. *Cognitive Computation*, 11(3), 294–316. https://doi.org/10.1007/s12559-018-9615-4

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.

Omar, A. B., Huang, S., Salameh, A. A., Khurram, H., & Fareed, M. (2022). Stock market forecasting using the random forest and deep neural network models before and during the COVID-19 period. *Frontiers in Environmental Science*, 10, Article 917047. https://doi.org/10.3389/fenvs.2022.917047

Oo, M. C. M., & Thein, T. (2019). Hyperparameters optimization in scalable random forest for big data analytics. In *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*. IEEE. https://doi.org/10.1109/CCOMS.2019.8821752

Peddi, P. (2019). An efficient analysis of stocks data using MapReduce. *Journal of Applied Science and Computations*, 6(1), 4076–4087.

Ronaghan, S. (2018, 12 May). The mathematics of decision trees, random forest, and feature importance in Scikit-learn and Spark. Towards Data Science. *Medium.com*. https://medium.com/towards-data-science/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3

Salloum, S., Dautov, R., Chen, X., Otrok, H., & Yassine, A. (2016). Big data analytics on Apache Spark. *International Journal of Data Science and Analytics*, 1(3–4), 145–164. https://doi.org/10.1007/s41060-016-0027-9

Shah, J. (2022). Netflix stock price prediction dataset [Data set]. *Kaggle.com*. https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction

Tosi, D., Kokaj, R., & Roccetti, M. (2024). 15 years of big data: A systematic literature review. *Journal of Big Data*, 11, Article 73. https://doi.org/10.1186/s40537-024-00914-9

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)*, (pp. 1–14). USENIX.

Acta Informatica Pragensia is published by the Prague University of Economics and Business, Czech Republic | eISSN: 1805-4951