Pakistan Journal of Scientific Research
Multidisciplinary & Peer Reviewed
Volume: 5, Number: 2, Pages: 74-86, Year:2025

Distributed Machine Learning with Spark MLlib: A Comprehensive Performance Evaluation

Kiran Hayat, Nargis Fatima and Sumaira Nazir

Department of Software Engineering, National University of Modern Languages, Islamabad, Pakistan

Correspondence: numl-s25-17501@numls.edu.pk, nfatima@numl.edu.pk, sunazir@numl.edu.pk

Submitted: 01-08-2025**, Revised:** 09-11-2025, **Accepted:** 15-12-2025

Abstract

The growing demand for big data processing and distributed machine learning is necessary to apply predictive models to millions, or even billions, of records. Apache Spark MLlib is one of the most well-known Spark libraries, providing machine learning algorithms for processing large-scale data both in memory and in a distributed manner. Researchers have explored the distributed and in-memory processing capabilities of the Spark framework, little research has been conducted regarding empirical comparison of predictive performance and system efficiency using the Spark framework. These pipelines, classified as classification, regression, and clustering, are implemented in Spark MLlib. The experimental datasets consist of 540,000 to 12 M records with unified feature encoding and normalisation, and partitioning techniques. The evaluation metrics of the developed models include F1-Score, accuracy, Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Silhouette Score, Time spent on training, Memory Usage, and Computational Speedup. The study results show that the ensemble models (Random Forest and Gradient-Boosted Trees) outperform the others in classification. Linear Regression has lower regression errors, K-Means clustering is stable across different data sizes, and distributed computation speeds up computation while also providing better accuracy when applied to more complex models. These results provide insights into design decisions for both algorithms and partitioning strategies that can improve performance and accuracy in the production of Spark MLlib workflows.

Keywords: Distributed Machine Learning; Apache Spark MLlib; Scalability; Parallel Computing; Data Partitioning; Big Data Analytics

1. Introduction

Distributed data Processing serves as a fundamental component of big data processing and analytics in an efficient manner [1]. Multiple distributed data processing framework are available for the processing of big data on of the known one is Apache spark. In modern applications, big data analysis, making regular machine learning methods that execute on a single machine infeasible. For data sets with millions or billions of records, training times and memory requirements exceed the capabilities of a single machine. This demand has led to the construction of distributed computing frameworks that partition computation workloads across clusters of computers. Apache Spark provides a distributed in-memory computation model and a library of scalable algorithms,

 https://doi.org/10.57041/ya5rmz93

MLlib [2]. Its approach of performing computations in parallel, with minimal input/output overhead, makes it especially suitable for large-scale analytics and machine learning [3].

The existing work on Spark MLlib has mostly focused on algorithm and system design, and on comparisons with previous systems such as Hadoop MapReduce [4]. They directly showed that Spark's in-memory computation model may provide speedups and lower latency than disk-based systems [4], [5]. Other work has considered the speed-up, fault tolerance, and scalability of distributed learning as data size increases [5]. While these experiments have shown the usefulness of distributed machine learning, they generally either use a specialised dataset or examine algorithms applicable only to a small class of distributed-learning problems [6]. Thus, there is a need for empirical studies that compare machine learning models across the major dimensions of performance and do not restrict attention to specific domains, datasets, or task types.

The paper aims to bridge this gap with a systematic empirical study of machine learning model implementations in Spark MLlib. It focuses on a wide range of models and is not tied to a specific data set or problem, but rather aims to understand their behaviour in distributed environments under commonly used metrics. To achieve study aims, the performance metrics between the two systems, including accuracy, F1-score, RMSE, MAE, training time, memory footprint, and speedup, were selected to provide insight into Spark's performance when scaling machine learning workloads. The study's research questions are given in Table 1. The rest of the paper is categorised as follows: Section 2 provides the study background, Section 3 describes the methodology, Section 4 discusses the study results, and Section 5 provides the discussion. Sections 6, 7, and cover the study conclusion, contribution and future work.

2. Background

In modern applications, the amount of data to be processed is continually growing; therefore, traditional machine learning methods that run on a single machine cannot be considered. In fact, when we deal with millions or even billions of data points, the training time and memory requirements become much larger than a single machine can manage [7]. This has led to the development of distributed computing frameworks that can handle the workload of many machines. The Apache Spark framework is commonly used for distributed data processing and introduced a distributed in-memory computation model and a library of scalable machine learning algorithms known as MLlib [3]. The ability to perform computation in parallel with minimal input/output overhead makes this framework best suited for machine learning operations [8] [9].

Apache Spark is a distributed computing framework that enables fast and efficient processing of large volumes of data. Unlike Hadoop's MapReduce, which stores data on disk, Spark utilises in-memory computing [10]. This feature of Spark makes it much more efficient in handling large volumes of data, especially in iterative processing, as seen in many machine learning algorithms [11]. Spark's primary abstraction for parallel and fault-tolerant data processing is the Resilient Distributed Dataset (RDD) [12]. A significant aspect of the Spark stack is the inclusion of MLlib, a scalable machine learning library that offers a wide range of algorithms for tasks like classification, regression, clustering, collaborative filtering, and dimensionality reduction [8]. The MLlib library is intended to run on Spark's distributed engine, enabling data scientists to implement machine learning workflows that can handle large datasets without worrying about the intricacies of distributed systems [11], [13].

The effectiveness of Apache Spark and MLlib in handling big data has been proven in a variety of fields. In healthcare, Apache Spark has been successfully employed in creating predictive

models to forecast cardiovascular diseases by utilizing classification algorithms like as logistic regression, random forests, and decision trees and found to be highly accurate in a distributed system environment [14] [15]. In another healthcare application, Apache Spark was successfully employed in managing diabetes by processing Electronic Health Records (EHRs), which proves its efficiency in handling big data in healthcare environments [5]. In cybersecurity, several studies have been conducted to prove the efficiency of Apache Spark's MLlib algorithms in processing network data to perform intrusion detection. Naive Bayes and random forests were found to have speedups when executed on multiple worker nodes [16] . Apache Spark was also successfully employed in the smart grid domain to analyze smart meter data to forecast loads and provide demand response, utilizing Apache Spark's SQL and MLlib modules to process billions of records and build predictive models accurately [11].

Although Spark MLlib is a powerful tool for machine learning tasks, it is not consistent in terms of performance for all the machine learning models and tasks it supports. According to the literature, there are several key factors that affect the performance and accuracy of Spark MLlib for machine learning tasks. For example, the selected machine learning model for the task may have trade-offs for level of accuracy that it offers for complex tasks, such as the fact that ensemble methods such as random forests and GBTs may offer higher accuracy than linear models for complex tasks, but may involve a computational cost that is significantly higher than that of linear models [17]. The scalability of Spark MLlib may also be affected by the number of nodes that it supports in a cluster, with experiments showing that the training time may be reduced by the addition of nodes to the cluster until a point is reached after which the addition of nodes may not offer the expected results [6]. Additionally, the size of the data set relative to the amount of available memory is a significant factor, and Spark is best suited to handle data sets that can fit into memory, although performance can suffer when data is spilled to disk [18].

The distributed characteristic of Spark MLlib also brings complexities that are not associated with a single machine setup. A typical example is the process of hyperparameter tuning, which is critical in optimizing machine learning models. The hyperparameter tuning process has been parallelized in a cluster setup but requires appropriate management to ensure that there is no significant overhead [17]. In addition, the integration of deep learning frameworks into the Spark pipeline is not always smooth and requires the creation of integration layers [19]. The efficiency of task-based parallelism in Spark MLlib has been evaluated in a study that concluded that linear models perform better in enhancing speedup compared to other supervised algorithms [12].

Existing works on Spark MLlib focused on algorithm design, system design, and comparison with existing framework like Hadoop MapReduce. They directly showed that Spark's in-memory computation model may provide speedup and latency advantage compared to disk-based systems [20]. Moreover, other research work has considered the speed-up, fault-tolerance, and scalability of distributed learning when scaling data size [10]. While these experiments have shown the usefulness of distributed machine learning, they generally either used a special dataset or examined an algorithm applicable only to a small class of distributed-learning problems. Thus, large empirical studies that compare machine learning models across the major dimensions of performance and do not restrict attention to specific domains, datasets, or task types remain needed.

3. Research Methodology

This section discusses the details of the methodology employed to conduct this research. It provides details about research questions, research objectives, data set, target variables, data preprocessing and cleaning, distributed partitioning strategy, distributed ML Model.

3.1 Study Research Questions and Objectives

The study research questions are as follows.

RQI: How does distributed training in Spark MLlib affect the predictive performance and computational efficiency of various machine learning models across datasets of increasing sizes.
RQ2: What are the trade-offs between training time, memory consumption, and model accuracy under different data partitioning strategies in a distributed environment?

The research objectives are as below.
RO1: To empirically evaluate the predictive performance (accuracy, F1-score, RMSE, MAE, silhouette score) of classification, regression, and clustering models implemented in Spark MLlib on datasets of varying sizes (500K to 12M records).
RO2: To analyze the system-level performance (training time, memory usage, computational speedup) of distributed machine learning and assess the impact of partitioning strategies on scalability and efficiency.

3.2 Datasets

To evaluate model performance and system overhead under varied dataset size and characteristics, this research work uses two real-world datasets and two synthetically generated datasets with gradually increasing sizes to answer the research questions. This allows validation of existing Apache Spark MLlib models under both simulated and experimental conditions. The details of data sets used in this research work are discussed in sub-sections.

3.2.1 Online Retail Dataset (Real-World)

The first data set is online retail dataset, a real-life transactional dataset with approximately 540,000 instances that contain many numerical and categorical features representing customer, product and transaction details. It is a semi-structured dataset with missing values, categorical data and class imbalance. It acts as a baseline for distributed machine learning algorithms on real-world transactional datasets and to understand how Spark handles common data preprocessing, encoding and partitioning problems found in real-world datasets.

3.2.2 Taxi Trip Dataset (Real-World)

The second dataset is a taxi trip dataset with about 12 M instances. It was obtained from the transportation sector. Continuous attributes include trip distance, fare-amount, and time-related variables. Categorical attributes include payment type, pickup area, and drop-off area. Because of its size and number of features, the dataset was suitable to examine the scalability, compute overhead and memory requirements of Spark MLlib models with real-world large datasets.

3.2.3 Synthetic Dataset – 1 M Records

The third dataset is a synthetic 1 M row dataset created to test performance on moderately sized workloads. It contains numerical and categorical features (both indexed and one-hot encoded) along with a binary classification label and a continuous regression target. The dataset provides a basis for repeatable, meaningful comparisons between implementations of different machine learning algorithms.

3.2.4 Synthetic Dataset – 10 M Records

The fourth dataset is synthetic and has 10 M rows. It uses the same schema as the previous dataset with a 10 M rows. The large dataset is used to study the changes in terms of training time, model accuracy, memory use, and speedup for increasing sizes of the dataset.

Keeping the schema constant while varying the number of records enables the systematic study of how dataset size can affect the behavior of a distributed learning algorithm. Data set summary is given in Table 3.

***Table 3**: Dataset Summary*

Dataset Name	**Dataset Type**	**Number of Rows**	**Number of Columns**	**Partition Size**
Online Retail Dataset	Real-world	~540,000	8	50
Taxi Trip Dataset	Real-world	~12,000,000	19	32
Synthetic Dataset (1M)	Synthetic	1,000,000	20	50
Synthetic Dataset (10M)	Synthetic	10,000,000	20	64

3.3 Target Variables

These datasets were created to compare the performance of models (classification, regression, and clustering) n datasets of different sizes.

Classification Task: The target class label was computed as a non-linear combination of the selected features enabling models to learn non-linear classification boundaries which can be encountered in many applications from the field of big data analytics. *Regression Task*: Continuous target variable was constructed by taking a weighted sum of the selected numerical features and adding random noise to simulate a real-world environment. Specifically, the target variable was generated using:

$$y = 0.3f_1 + 0.6f_3 + \varepsilon$$

Where f_1 and f_3 are numeric input features and ε is a small noise term. This provides a regression task with predictable behavior that has enough complexity to allow close analysis of the model's performance. *Clustering Task:* The models were trained under self-supervised conditions on feature vectors without target labels, learning the structure of the data based on relations between the feature vectors. A consistent target across all datasets allows comparison of classification accuracy, regression error, clustering quality, and system-level metrics such as training time, memory usage, and scalability across datasets of various sizes and dimensionalities.

3.4 Data Preprocessing

The datasets were preprocesses in the same way to ensure the reliability of experimental results. *Data cleaning* was performed to identify missing values. Real-world datasets required explicit handling of missing values. However, the synthetically generated datasets typically did not have missing values. During *categorical features encoding c*ategorical features were transformed into indexed categorical features using the StringIndexer transformer and subsequently transformed into sparse binary vectors using the OneHotEncoder transformer. This allows categorical features to be used efficiently by Spark MLlib models without introducing an ordinal bias associated with categorical integer values. During feature assembly the numeric and the encoded categorical features were combined as a single feature vector using the VectorAssembler, the input format required by Spark's machine learning algorithms. For models sensitive to the scale of features such as KNN classifiers, MinMaxScaler normalize the features to range from 0 up to 1 during *normalization.* Finaly during train-test split, each dataset split into a training set held 80% of all

data, and a test set held 20%. This data split, with the same ratio, was selected across all datasets to maintain consistency and to allow fair comparison among the models.

3.5 Distributed Partitioning Strategy

To observe the speed-up and memory impact of distributed processing, the experiments were executed under an idealized partitioning scheme for Spark, where the size of the partitions was controlled accruing to dataset size, available memory of the executors, and stability of the cluster. Under this controlled environment, the impact of the data and partitions on training latency and memory pressure, and the rest of the system was observed. Both datasets (the online retail dataset and the taxi dataset) were processed for both scenarios: without explicit repartitioning (using Spark's default partitioning) to investigate the model performance, and with explicit repartitioning to investigate the benefit of explicit repartitioning. Partitions of the synthetic datasets were pre-determined, having sizes that scaled with the input dataset, so that executors got similar amounts. The partitioning schemes provide a way to compare single partition (also called minimum partitioning) execution with fully distributed multi-partition training. The execution time and resource usage statistics provide a criterion for measuring the parallel processing power of Spark as well as the performance scalability as the size of the training dataset increases. This ensures balanced distribution and allows comparison of single-partition training, distributed multi-partition training, and speedup.

3.6 Distributed Machine Learning Models

Machine learning models were trained for three separate machine learning tasks classification, regression, and clustering on each dataset. The classification techniques included *Logistic Regression, Random Forest Classifier, and Gradient Increased Trees (GBT).* The evaluation criteria used for the performance of classification algorithms were *Accuracy, AUC, F1-Score, Model Training Time, Memory Usage and Speed-up.* For regression, the *Linear Regression, Decision Tree Regressor, Random Forest Regressor and Gradient Increasing Regressor* were trained to predict the regression values. RMSE, MAE, MASE, Training Time, RAM usage and Speed-up were used to evaluate the models. For clustering, KMeans and Bisecting KMeans algorithms were implemented, and their Silhouette score, convergence time, and memory consumption were tracked. Each algorithm was tested across the datasets as to whether they would scale in a distributed processing environment.

3.7 Performance Measurements

To evaluate the behavior of distributed training on these four datasets, a set of model-level and system-level metrics was recorded.

3.7.1 Model Performance Metrics

Model performance metrics are given in Table 4.

Table 4. Model Performance Metrics

Model	**Metric**	**Description**
Classification	Accuracy	Proportion of correctly classified instances
	F1-score	Harmonic mean of precision and recall
	AUC	Area Under the ROC Curve
Reg	RMSE	Square root of average squared prediction errors

	MAE	Average absolute prediction errors
	MASE	Scaled version of MAE for interpretability
Clustering	Silhouette Score	Measure of cluster cohesion and separation (-1 to 1, higher is better)

3.7.2 System-Level Performance Metrics

The metric for system level performance metrics utilized in this study were "*Training Time*", "*Computational Speed-up*" and "*Memory Usage*". *"Training Time"* is the time from model initialization to completion, measured in seconds. *"Memory Consumption"* **is** monitored through Spark executor metrics and Python runtime memory probes. "*Computational Speedup"* is Calculated as: Speed-up= (T-single partitioning/T-distributed). Where "*T_single_partition*" is the training time for a single partition and *"T_distributed"* is the time for multiple partitions. Each dataset was analyzed to observe the scaling trends as the dataset size increased from hundreds of thousands to millions and tens of millions.

3.8 Result Storage and Visualization

Model performance metrics and system level performance metrics were stored in structured files and visually represented through graphs. The research methodology is presented in Figure 2.

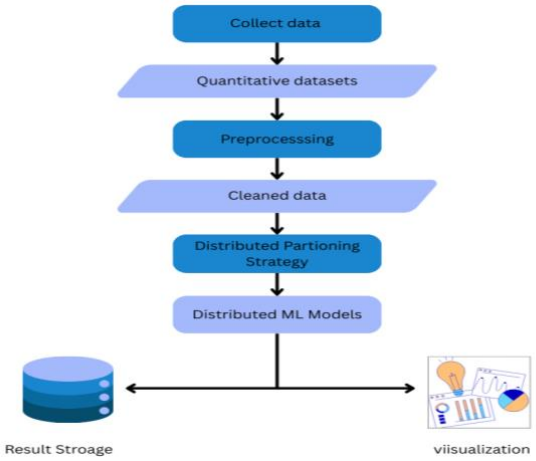

Figure 1 Research Methodology

4. Results

The study results based on the study research objectives and research questions. The results are discussed in upcoming subsections.

4.1 Results for Research Question 1 "Predictive Performance Across Datasets

RQ1 aims to evaluate the predictive performance of classification, regression, and clustering models implemented in Spark MLlib on datasets ranging from 500K to 12M records. The evaluation is based on accuracy, F1-score, RMSE, MAE, and silhouette score.

4.1.1 Results for Classification Model Performance

The study results for classification models show that models performed well for all data sets. In the Online Retail dataset, which contains 540K records, all models Logistic Regression, Random Forest, and GBT achieve accuracy and F1-scores close to 1. The Random Forest and Gradient Boosted Trees models performed slightly better than Logistic Regression. The same pattern is also

observed for the 1 M synthetic dataset. The accuracy and F1-scores close to 0.99. It clearly shows that all the models performed well with large, structured datasets. For 10 M dataset accuracy and F1- score for all the models was 1.0, it clearly shows that performance was improved with larger data set. The study results showed that performance decreased with Taxi Trip dataset where the accuracy of Logic regression, Random Forest and GBT were 0.78, 0.79, and 0.79 respectively. It indicates that models performed better with increased data sets.

, while the accuracy of Random Forest and GBT was around 0.79.

4.1.2 Results for Regression Model Performance

The results analysis of Regression model was obtained based on RMSE and MAE. The results showed that RMSE and MAE for simple linear regression models are low i.e. 0.01 for online retail dataset as compared to tree regression models. It indicates that there is linear relationship between features and target variables. Similar findings were observed for 1 M data set where RMSE and MAE values for linear regression model are close to 0 and for tree regression model the values of RMSE and MAE are 0.95 and 0.78. Similar findings were reported using the previous 1 M data, where the Simple Linear Regression produced RMSE and MAE values at close to 0 and the Tree Regression produced RMSE and MAE values of 0.95 and 0.78 respectively. Same trends were observed in 10 M datasets where linear regression model performed well. In case of Taxi trip Data, the RMSE value was 1.1 for both models while MAE value for linear regression models was lowest. The results indicated that the model complexity does not guarantee better results. The results also indicate that linear regression consistently provides the best regression performance across all dataset sizes.

4.1.3 Results for Clustering Model Performance

The silhouette score was used to evaluate the performance of the clustering algorithm. For Online Retail dataset, KMeans algorithm yields a silhouette score approximately equal to 0.70 it indicates that there is good separation between the clusters created by this algorithm. For the 1 M datasets, the silhouette score decreases to approximately 0.58, it shows that clustering produced in this case has a more complicated structure than that of the Online Retail dataset. But for the 10 M datasets, the silhouette score increases to 0.95, it indicates that the algorithm performance depends on the amount of data available for clustering. Likewise, the silhouette score for the Taxi Trip dataset is approximately 0.75, although we can see from the previous two datasets that this is a lower score. The results indicate that the performance of clustering models improves as dataset size increases. The predictive performance (accuracy, F1-score, RMSE, MAE, silhouette score) of classification, regression, and clustering models is given in Table 5.

Table 5. Predictive performance of classification, regression, and clustering models

Datasets	Classification Models	Accuracy	F1 score	Regression Models	RMSE	MAE	Clustering Model	Silhouette Score
Online Retail	Logistic Regression Random forest GBT	0.99 1.0 1.0	0.99 1.0 1.0	Linear Regression Decision Tree Regressor	0.01 0.75	0.01 0.3	K Means	0.7

1M Dataset (Syntheti	Logistic Regression Random forest GBT	0.99 0.99 0.99	0.99 0.99 0.99	Linear Regression Decision Tree Regressor	0.0 0.95	0.0 0.78	K Means	0.58
10M Dataset (Synthetic	Logistic Regression Random forest GBT	1.0 1.0 1.0	1.0 1.0 1.0	Linear Regression Decision Tree Regressor	0.0 0.95	0.0 0.78	K Means	0.95
Taxi Trip Dataset	Logistic Regression Random forest GBT	0.78 0.79 0.79	0.72 0.74 0.75	Linear Regression Decision Tree Regressor	1.1 1.1	0.93 0.92	K Means	0.75

The results analysis of RQ1 shows that the predictive capabilities of the models developed using Spark MLlib remain high even when the size of the data set increases from 500K to 12M data points. Classification models have high accuracy with high F1 scores, Linear Regression consistently has the lowest error values, and KMeans has consistent clustering quality.

4.2 Results for Research Question 2 "System Level Performance and Scalability"

RQ2 focuses on system performance, including training time, memory usage, and computational speedup in a distributed environment. The results are discussed in subsections.

4.2.1 Results for Training Time

The results analysis regarding training time shows that it varies depending on the model type. For Online Retail dataset, the training time for the Linear Regression model was only 4.86 seconds, while the training time for the Gradient Boosted Trees model was 83.99 seconds. For 1 M data set the training time for the Linear Regression model was the least compared to other models. Similarly, the training time for 10 M and 12 M datasets was more compared to the 1 M and 4 M datasets. Likewise, the training time for the 10M and 12M datasets was more for the Random Forest and Gradient Boosted Trees models as compared to the other models.

4.2.2 *Results for Memory Usage*

The results show that the models have same amount of memory used when implemented with the same datasets. The Online Retail dataset used approximately 493 MB of memory while the 1M dataset used approximately 221 MB of memory. The 10M dataset used about 142 MB of memory compared to the Taxi Trip dataset which used about 260 MB. It indicates that distributed processing can optimize memory consumption regardless of an increase in the size of a dataset.

4.2.3 Results for Computational Speedup

The speedup in computation depends on the type of model used. For the Online Retail dataset, the values of the speedup were small, i.e., 0.49, 1.13, which indicates that for small-sized datasets, the use of distributed computing does not result in significant benefits. For the 1M dataset, the speedup for models like Random Forest and Linear Regression increased, with the value of the speedup greater than 1.4 in some cases. When the size of the dataset increased to 10M and 12M records, the values of the speedup increased, especially for models like Random Forest, GBT, and Decision Tree. The results indicate that with the increase in dataset size, distributed data processing is beneficial, particularly for complex models.

The results analysis for RQ2 show that the system-level results verify the benefits of distributed machine learning for Spark MLlib's scalability and efficiency as the size of the data set gets larger. Although the simpler models remain more efficient for training speed, the more complex models benefit more from the distributed approach. The training speed, memory usage, and speedup benefits verify the advantages of distributed computing as the size of the data set gets larger. Table 6 presents the result analysis of system Level performance metrics.

Table 6 System Level Performance Metrics

Dataset	Task	Model	Training Time (s)	Memory Usage	Computational Speedup
Online Retail	Classification	Logistic Regression	47.25	493.28	0.49
		Random Forest	19.09	493.31	0.73
		GBT	83.99	493.34	0.49
	Regression	Linear Regression	4.86	493.36	1.13
		Decision Tree Regressor	10.42	493.39	0.56
	Clustering	KMeans	8.55		0.76
1M Dataset (Synthetic	Classification	Logistic Regression	21.95	221.81	1.26
		Random Forest	13.41	221.82	1.4
		GBT	58.45	221.84	0.92
	Regression	Linear Regression	1.43	221.86	1.67
		Decision Tree Regressor	5.52	221.87	1.10
	Clustering	KMeans	24.39		0.66
10M Dataset (Synthetic	Classification	Logistic Regression	201.46	142.11	1.38
		Random Forest	127.38	141.90	1.93
		GBT	330.87	141.92	1.49
	Regression	Linear Regression	19.33	141.96	2.09
		Decision Tree Regressor	60.40	145.59	1.51
	Clustering	KMeans	101.25		1.57
Taxi Trip Dataset	Classification	Logistic Regression	389.76	259.65	1.71
		Random Forest	2346.74	259.94	1.72
		GBT	2103.26	260.25	1.39
	Regression	Linear Regression	146.16	261.37	2.04
		Decision Tree Regressor	221.69	261.43	1.77
	Clustering	KMeans	114.57		1.43

5. Discussion

The experimental results indicate that the increasingly used Spark MLlib distributed pipelines lead to high predictive performance on large datasets. For classification, ensemble models like Random Forest or Gradient Increased Trees are superior to Logistic Regression. The benefit of using these models over Logistic Regression becomes smaller with larger datasets, making Logistic Regression suitable for smaller datasets, and situations where interpretability is preferred over

accuracy. Likewise, across all four datasets the Linear Regression model achieved the lowest error indicating that the underlying relationships were approximately linear. The tree-based regressors appear to be more sensitive to their dataset size and variance, and therefore further hyperparameter tuning or regularization may be required to prevent overfitting. For clustering model KMeans consistently achieved high silhouette scores across datasets. The lower score for real datasets was due to the noise and complexity in real-world data compared to synthetic datasets. Regarding computational efficiency the distributed execution speeded up the training of computationally intensive algorithms, such as Random Forest and GBT, on large datasets (10M, 12M records), where parallelization allows a reduction in training time of 40-60%. Simpler models (Logistic Regression, Linear Regression) do not benefit from parallelization as much due to their lower computational load; their coordination overhead also increases considerably. Regarding memory storage it was observed that memory usage did not increase between datasets and never filled up even in the case of the 12M record dataset. It confirms the efficiency of Spark's memory management, even on large datasets. Regarding Partitioning Strategy, the way partitions are distributed over these machines largely influence performance metrics such as load balancing. Too many partitions can create excessive coordination overhead, while too few partitions can lead to unbalanced workloads and less utilization of cluster resources which degrades system performance.

6. Conclusion

The study explores distributed machine learning based on the Apache Spark MLlib that allows for training models in small to very large datasets (from 540,000 to 12 M records) efficiently. Classification models such as Random Forest and Gradient Increased Trees outperform linear models like Logistic Regression on large datasets, but the latter remains competitive as well. Linear Regression exhibits the lowest MAE and RMSE across all datasets. Tree-based regressors perform comparably on occasion but require wide-ranging tuning to do so. For Clustering KMeans silhouette scores on various datasets indicate that KMeans clustering scales. The study shows that distributed execution helps speed up the training time of more complex models. The speedup is greater for larger datasets. Less complex models have lower arithmetic intensity and benefit less from execution being distributed. The study results can help us select algorithms, partitioning strategies, and cluster configurations to maximize their predictive accuracy and computational efficiency in production-scale distributed machine learning systems. This work adds to the distributed and big data processing body of knowledge on the performance characteristics of distributed machine learning processes empirically across models and tasks of different sizes and datasets. The study concludes that distributed machine learning is a faster and more efficient approach for large scale data.

7. Study Contribution and Implications

The paper provides in-depth empirical research on distributed machine learning by applying the Apache Spark MLlib to the classification, regression, and clustering tasks of datasets with thousands to millions of items. The research study compares the different machine learning algorithms in terms of the accuracy of the prediction, training time, memory consumption, and the speedup in comparison to the different approaches of data partitioning. The study findings can aid practitioners in deploying distributed machine learning pipelines.

8. Future Work Suggestions

In future the study can be replicated with larger datasets 100M rows to analyze maximum scalability of larger datasets. Real time streaming machine learning pipelines can be designed with spark framework. Multi-modal distributed learning on heterogeneous feature types including text, image, and time series data can also be evaluated.

Supplementary Materials:

Author Contributions: For research articles all three authors "Nargis Fatima, Sumaira Nazir and and Kiran Hayat" contributed for conceptualization, research background, methodology, data collection and analysis, writing and Editing; Formal Review; "Nargis Fatima, Sumaira Nazir" Supervision, "Nargis Fatima, Sumaira Nazir . All authors have read and agreed to the published version of the manuscript.".

Funding: This research received no external funding.

Conflicts of Interest: Declare conflicts of interest or state "The authors declare no conflicts of interest."

References

1. A. Rehan, N. Fatima, and S. Nazir, "Waste Generated in Distributed Data Processing Systems : Strate- gies and Future Directions," vol. 2, pp. 82–91, 2025.
2. A. Pratap, S. Kant Gupta, and S. Shahi, "Exploring Energy-Efficient Data Processing Technique for Sustainable Computing," 2025.
3. S. K. Gupta, "Machine Learning Integration in Spark-Based Pipelines," vol. 12, no. 4, pp. 3020–3025, 2025.
4. P. Sewal and H. Singh, "Learning Based Iterative Gbtr On Higgs And Covid-19 Datasets," vol. 25, pp. 1373–1386, 2024.
5. K. Sharma, D. Parashar, O. Mengshetti, R. Ahmad, R. Mital, and P. Singh, "Revue d ' Intelligence Artificielle Apache Spark for Analysis of Electronic Health Records : A Case Study of Diabetes Management," vol. 37, no. 6, pp. 1521–1526, 2023.
6. J. Zhang, Z. Yang, and Y. Benslimane, "Exploring And Evaluating The Scalability And Eficinecy Of Apache Spark Using Educational Datasets," *2019 Int. Conf. Mach. Learn. Cybern.*, pp. 1–6.
7. L. Belcastro, R. Cantini, F. Marozzo, A. Orsino, D. Talia, and P. Trunfio, *Programming big data analysis : principles and solutions*. Springer International Publishing, 2022.
8. I. Aouria, "Machine learning in big data : A performance benchmarking study of Flink-ML and Spark MLlib," vol. 21, no. 2, pp. 18–27, 2025.
9. M. Azeem and B. M. Abualsoud, "Mobile Big Data Analytics Using Deep Learning and Apache Spark," no. c, pp. 16–28, 2023.
10. F. Ullah, I. Mohammed, and M. A. Babar, "A Framework for Energy-aware Evaluation of Distributed Data Processing Platforms in Edge-Cloud Environment," pp. 1–10, 2022.
11. A. El *et al.*, "Big data resolving using Apache Spark for load forecasting and demand response in smart grid : a case study of Low Carbon London Project," *J. Big Data*, 2024.
12. V. Nejkovic, "The computational efficiency of task-based parallelism of spark MLlib," no. June, pp. 9–12, 2025.
13. Z. Shen, "A simple and efficient framework for big data processing and analysis," *2025 5th Int. Conf. Neural Networks, Inf. Commun. Eng.*, pp. 1754–1759, 2025.
14. A. Ed-daoudy, K. Maalmi, U. Sidi, and M. Ben, "Real-time machine learning for early detection of heart disease using big data approach Batch interval," *2019 Int. Conf. Wirel. Technol. Embed. Intell. Syst.*, pp. 1–5, 2019.
15. Y. K. Gupta and S. Kumari, "Performance Evaluation of Distributed Machine Learning for Cardiovascular Disease

Prediction in Spark," *2021 5th Int. Conf. Trends Electron. Informatics*, pp. 1506–1512, 2021.

16. R. Atefinia and M. Ahmadi, "Performance Evaluation of Apache Spark MLlib Algorithms on CSE-CIC-IDS2018 Intrusion Detection Dataset."
17. J. Chen, K. Li, S. Member, Z. Tang, and K. Bilal, "A Parallel Random Forest Algorithm for Big Data in a Spark Cloud Computing Environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 919–933, 2017.
18. Y. N. Seunghye Han*, Wonseok Choi**, Rayan Muwafiq**, "ImpactofMemorySizeonBigdataProcessingbasedonHadoopandSpark," in *RACS '17*, 2017.
19. N. J. Venkatesan, C. Nam, and D. R. Shin, "Deep Learning Frameworks on Apache Spark : A Review Deep Learning Frameworks on Apache Spark : A Review," vol. 4602, no. May, 2018.
20. P. Josyula, "Survey of Big Data Architectures and Frameworks on Kubernetes : Challenges , Solutions , and Future Directions," *2025 10th Int. Conf. Big Data Anal.*, pp. 1–8, 2025.