IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 27, Issue 4, Ser. 4 (July. – August. 2025), PP 60-79
www.iosrjournals.org

Performance Analysis Of Various Data Classification Technique Based On Mapreduce Framework

Nasrin Akter
Student, Jahangirnagar University

Md. Fazlul Karim Patwary
Professor, Jahangirnagar University

Md. Sadi Rifat
Lecturer, Prime University

Abstract
With the continuous advancement of internet technologies, the importance of efficient data classification has grown significantly. In this research, the performance of selected classification techniques has been analyzed using both a traditional execution framework and a distributed computing framework based on Hadoop MapReduce. Three widely used algorithms—Naive Bayes, Decision Tree, and Linear Support Vector Machine (SVM)—are implemented and tested under varying mapper sizes. Their performance is evaluated using key metrics such as accuracy, precision, recall, memory usage, and computation time. The experimental results are presented through comparison tables, Bar Charts and a Heatmap to highlight the differences in performance across the two frameworks. The findings indicate that the MapReduce-based approach significantly improves classification performance in terms of accuracy and recall, particularly for Decision Tree and Linear SVM, which achieved accuracy rates above 93%, while Naive Bayes also showed noticeable improvement compared to its traditional counterpart. However, these enhancements come at the cost of increased memory consumption—rising from under 1 GB in traditional settings to as high as 5.5 GB in distributed execution—as well as a broader range in computation time depending on the mapper size. This analysis aims to determine how the underlying computational framework influences the efficiency of classification algorithms and to identify which technique performs best under different execution environments.
Keywords: *Data Classification, Hadoop MapReduce, Naive Bayes, Decision Tree, Linear SVM, Accuracy, Precision, Recall, Performance Analysis.*

Date of Submission: 19-08-2025 Date of Acceptance: 29-08-2025

I. Introduction

Background Studies

Data, which is the most important thing in today's world, is becoming more comprehensive and voluminous day by day. Managing, processing, and analyzing this expanding volume of data has become a significant challenge. In this case data classification techniques act as a blessing.

Classification techniques organize data into a predefined arrangement. They can classify all types of data in an effective manner. A data classification technique uses a training set in order to form the model for class attribute and a test set to verify and validate the former model.

To classify the data efficiently, researchers have proposed many classification technique, such as Naïve Bayes Classifier, Decision Tree Classifier, K-Nearest Neighbor Classifier, Support Vector Machine Classifier, Artificial Neural Network etc. Among them the Naïve Bayes classifier is basically developed based on Bayes Theorem stated by Thomas Bayes [1]. This classifier does not require any complicated iterative parameter when the size of dataset increases but unfortunately it takes much training time [2].

When the necessity of quick search occurs, the Decision Tree classifier is one of the best option where a classification tree is used in which every internal node is labeled as input features [4]. But unfortunately over fitting occurs in this classifier [2]. To solve this problem the SVM classifier was proposed which aim to find the optimal hyperplane to separate classes, provide robust performance and are less prone to overfitting in high-dimensional spaces. [5].

While traditional classification frameworks often rely on single-machine execution, they can become inefficient when scaling to larger or more complex datasets. To address this, distributed computing frameworks such as Hadoop MapReduce have been introduced. The MapReduce processes an extensive amount of dataset using a distributed computing paradigm where two functions, such as map() and reduce(), are used to process the big dataset. The map function breaks each individual elements of the input datasets into tuples and feeds them as input to the reduce function. The reduce function shuffles these tuple and produces the final output. The basic block diagram of MapReduce framework is shown in Fig 1.1:

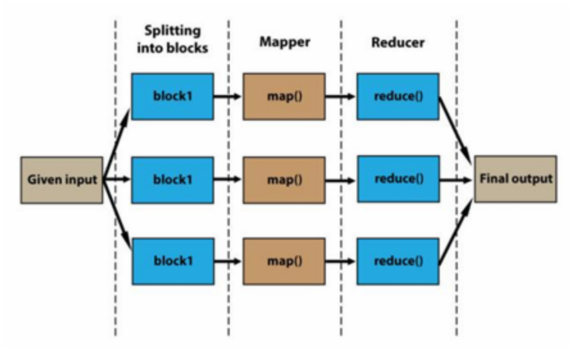


Figure 1.1: Basic Block Diagram of Map Reduce Framework

MapReduce offers several advantages, including parallelism, scalability, fault tolerance, and cost-effectiveness [7]. These benefits have led to its adoption in diverse fields such as large-scale data analysis, fraud detection, and healthcare analytics.

In this research, selected data classification techniques—specifically Naïve Bayes, Decision Tree, and Linear SVM—have been implemented and evaluated using both a traditional framework and the Hadoop MapReduce framework. The goal is to analyze how the underlying computational framework influences the performance of these classifiers in terms of accuracy, precision, recall, memory usage, and computation time.

This performance comparison provides insights into the suitability of each classification technique under different execution environments and helps determine whether distributed frameworks such as MapReduce significantly affect classification outcomes. Additionally, this analysis will help us to determine which classification technique suits well for data analytics on MapReduce framework.

Problem Statement

Efficient data classification plays a critical role in extracting meaningful insights from structured datasets. However, traditional frameworks often face limitations in terms of scalability and processing time, especially when handling moderately sized data that requires multiple computations. To address these limitations, distributed processing frameworks such as Hadoop MapReduce have been introduced, which allow parallel execution of classification tasks using multiple mappers. This research aims to implement and compare the performance of selected data classification techniques using both traditional execution and the Hadoop-based MapReduce framework. The goal is to analyze how the underlying computational framework affects classification performance in terms of accuracy, precision, recall, memory usage, and computation time.

Objective

The objectives of this research are as follows:

(a) To implement selected data classification techniques.
(b) To evaluate and compare the performance of these techniques using both traditional and MapReduce frameworks.
(c) To investigate the impact of the computational framework on the effectiveness of the classification methods.

Challenge and Drawback

This research work has implemented only some selected data classification technique on Map Reduce framework, not developed any new data classification technique. As there are too much data available in modern scientific era, selecting a particular dataset for effective classification is a challenge for me. Additionally, the limited scope of classifiers and frameworks considered may affect the transferability of the results.

Research Outline

The remaining part of this paper is composed of four more chapters. Here, Chapter II discusses Literature review describing the existing works. Chapter III demonstrates proposed system model with proposed system architecture and parameters for performance analysis. Chapter IV describes the implementation results with the used software tools and obtained outcomes. Finally, conclusion is drawn in Chapter V.

II. Literature Review

Introduction

In literature there is a brief observation on some related works for big data classification based on Map Reduce framework. This chapter is divided into two parts. The first part contains various well known big data classification techniques from section 2.2 to 2.4. The second part consists of some related works on the performance analysis of these classification techniques in section 2.5.

Related Works on Data Classification Using Naïve Bayes Classifier

Based on prediction accuracy, Songtao Zheng formed and evaluated a Naïve Bayes Map Reduce model on five different datasets of UCI Machine Learning Repository [6]. By performing a scalability analysis, his study stated that the performance of his proposed model is much better in comparison with running the same algorithm on a single machine.

In their research work, Aaron Klein et al. introduced a new Bayesian optimization approach, named Environmental Entropy Search, which is appropriate for enhancing the hyper parameters of ML algorithms used for massive datasets [7]. Their study stated that their proposed approach obtains 100 times faster performance in comparison with standard entropy search.

Based on Map Reduce framework, Chitrakant Banchhor et al. proposed another big data classification algorithm that combines Fuzzy theory with the CNB classifier to gain better performance during data classification [8]. They have transformed their dataset into a probabilistic index table before implementing the proposed method. They stated that their proposed classifier provides accuracy, sensitivity, and specificity of 91.78166%, 94.79%, and 88.8912% respectively in comparison with NB, CNB, GWO-CNB and FNB classifiers with different mapper size. In their another research work, Chitrakant Banchhor et al. analyzed the performance of NB, CNB, CGCNB, HCNB and FCNB in terms of accuracy, sensitivity, specificity, memory and execution time and stated that HCNB performs better than all the five classifiers based on Map Reduce framework [9].

Data Classification Using SVM Classifier

SVM is one of the most popular data classification technique used in big data classification. An extended version of SVM presented by R. S. Gunn [10] was introduced by Anushree Priyadarshini et al. which is based on Map Reduce framework [11]. They have showed that both the accuracy and computation time of Map Reduced based SVM is much better than the traditional SVM as it classifies data using a distributed environment. They have explained their proposed classifier in detail.

Liliya Demidova et al. proposed a modified version of SVM classifier based on modified PSO algorithm that helps in selecting the best kernel parameter and kernel function type [12]. Their study stated that SVM classifier based on modified PSO provides better performance and accuracy in comparison with SVM based on traditional PSO algorithm. One of the main contribution of their study is the ability of regeneration of particles.

Rashmi K Thakur et al. proposed Kernel Optimized SVM for sentiment classification of train reviews based on Map Reduce framework [13]. They compared and analyzed the performance of their proposed classifier with SentiWordNet, NB, NN, and LSVM and acquired sensitivity, specificity, and accuracy of 93.46%, 74.485% and 84.341% respectively.

Data Classification Using Decision Tree Classifier

Mary Slocum explored decision making using a simplified version of Decision Tree, called ID3, and showed how this decision tree classifier can be used in medical research area [14]. One of the extension of this ID3 classifier is C4.5 which was also developed by Ross Quinlan [15] where the C4.5 can handle missing values unlike ID3 classifier. A parallel version C4.5 classifier was proposed by Wei Dai et al. to minimize the computation cost of traditional C4.5 and maximize the efficiency [16].

Based on MapReduce attribute weights, Fubao Zhu et al. proposed another extended version of Decision Tree, called CART Decision Tree [17]. Here the weight concept enhances the efficiency of forming the decision tree. However, the efficiency problem is still present.

In order to increase the performance of C4.5 decision tree, Fatima Es-sabery introduced a Fuzzy theory based C4.5 decision tree using the MapReduce framework and compared it with ID3, traditional C4.5 and Fuzzy C4.5 in terms of Recall, Specificity, error rate, classification rate, Kappa statistics, F-1 score and execution time [18]. Their study stated that their proposed decision tree classifier outperforms other three decision trees with Recall, specificity, , error rate, classification rate, Kappa statistics, F-1 score and execution time of 92.13%, 89.56%, 6.48%, 93.52%, 85.4%, 79.05% and 556s, respectively.

Performance Analysis of Data Classification Techniques

Zulfany Erlisa Rasjid et al. used a XML document as text data to be classified using Naïve Bayes and kNN classification techniques and made a comparison of their performances [19]. This study stated that kNN performs better than Naïve Bayes classification technique in terms of recall, precision and F-measure except the value of k in kNN is equal to1. When k=1, Naïve Bayes is a better option than kNN as there is only a single neighbor.

For classifying multi-class text, Tomas Pranckevicius et al. made an investigation on five classification techniques, such as NB, RF, Decision Tree, SVM, and Logistic Regression classifiers [20]. Their study stated that Logistic Regression classification technique performs the best among all of them and Decision Tree classification technique gives the worst performance with an accuracy of 58.50% and 34.58%, respectively.

Based on Hadoop and Mahout, Anjuman Prabhat et al. classified twitter reviews as sentiment using NB and Logistic Regression classifiers along with a performance analysis report in terms of accuracy, precision and computation time [21]. This study expressed that Logistic Regression classifier offers 10\% more accuracy than NB classifier. Also the Logistic Regression classifier pays better precision and less computation time in comparison with the NB classifier for the same sized dataset. However their study included only the text sentiments not any image sentiment.

Vikas Singh et al. presented a comparative exploration of three big data classifiers, such as Gaussian Mixture Model, Logistic Regression and Random Forest based on MapReduce in terms of test accuracy and run time with different mapper size [22]. They have also determined the accuracy of these classifiers using only the mapper, and the mapper and reducer together, separately, and compared the results found. Their study stated that Random Forest classifier outperforms the other two classifiers on MapReduce.

Hemn Barzan Abdalla et al. conducted a comprehensive survey exploring a wide range of MapReduce models used for big data processing, including Hadoop, Spark, Hive, Pig, Cassandra, and MongoDB [23]. Their review focused on performance metrics, advantages, and limitations of each model, highlighting how MapReduce helps in distributed data handling with improved scalability and fault tolerance. A significant contribution of their study is the classification of 75 recent works based on methodology and system performance, offering researchers insights into choosing appropriate models for different big data scenarios. While the paper emphasizes the strength of models like Spark in scalability and Hive in query processing, it also points out challenges such as SLA violations, processing delays, and lack of compatibility across heterogeneous systems.

Saliha Mezzoudj et al. conducted a comparative study of big data frameworks, highlighting performance, scalability, and fault tolerance as key evaluation criteria [24]. They found that MapReduce is effective for batch processing across distributed clusters, though less suitable for real-time tasks. In contrast, Flink supports real-time analytics with low latency and exact-once processing. CUDA was noted for leveraging GPU acceleration to enhance computational efficiency. Among storage systems, BeeGFS outperformed others in throughput and reliability. Their layered classification model aids in selecting appropriate technologies based on workload type, making the study a valuable resource for big data architecture design.

Research Gap

Analyzing the others research works related to my work, shortcomings are noticed. Firstly, as the size of test dataset increases the performance of data classification techniques varies in comparison with the performance measured using small dataset. Secondly, some classifiers give efficient accuracy for text classification but provide very poor performance for image classification. Thirdly, the real-time implementation of MapReduce framework based big data classification is still less.

III. System Architecture

Introduction

This chapter provides detail explanation of the configuration that is being used to make this research work complete and working well. Section 3.2 provides a brief narration of the proposed system. Section 3.3 provides the designing model of our proposed system and describes each part of this model. Section 3.4 gives a summary of this chapter.

Proposed System

The primary goal of this study is to analyze the performance of different data classification techniques using two distinct frameworks: the traditional framework and the MapReduce framework. To achieve this, three widely-used classification algorithms such as Naive Bayes (NB), Linear Support Vector Machine (SVM) and

Decision Tree are implemented on both frameworks. The performance analysis is conducted in two phases. First, the classifiers' performances are compared within the traditional framework using several evaluation metrics, including precision, recall, accuracy, memory usage, and computation time. Next, a comparative evaluation is performed between the traditional and MapReduce frameworks, focusing on storage utilization and computation time. Finally, the results are summarized in a comparison table, accompanied by bar charts and a Heatmap to visually represent and highlight the classifiers' performance across the two frameworks.

System Architecture

The proposed system is divided into two parts. The first part does the performance analysis of the selected classification techniques based on traditional framework and the second part does the performance analysis of the same algorithm based on both traditional and MapReduce frameworks.

The first part consists of four stages. They are:
(a) Dataset Preparing Stage.
(b) Classification Stage.
(c) Performance Analysis Module.
(d) Performance Evaluation and Report Formulation.

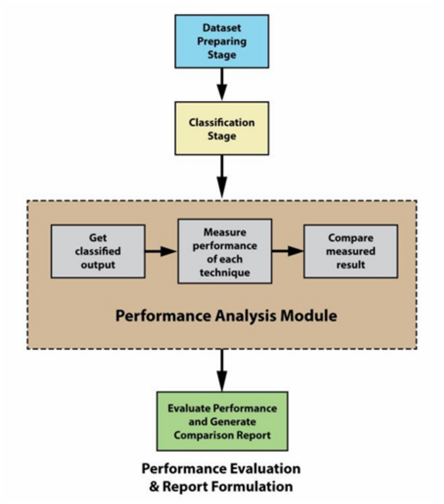

Dataset Preparing Stage
Classification Stage
Get classified output
Measure performance of each technique
Compare measured result
Performance Analysis Module
Evaluate Performance and Generate Comparison Report
Performance Evaluation & Report Formulation

Figure 3.1: Part 1 of Proposed System Architecture

The second part consists of four steps. They are:
(a) Classify The Dataset on MapReduce Framework.
(b) Recording Performance Metrics
(c) Comparing Results with Those Obtained from the Traditional Framework
(d) Evaluating the Performance and Generating Analysis Report

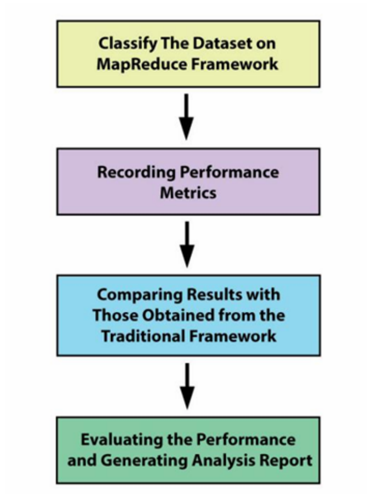

Classify The Dataset on MapReduce Framework
Recording Performance Metrics
Comparing Results with Those Obtained from the Traditional Framework
Evaluating the Performance and Generating Analysis Report

Figure 3.2: Part 2 of Proposed System Architecture

The above stages are described below:

Dataset Preparing Stage:

Here, the dataset used for this research was primarily collected from Kaggle and supplemented with another similar brain MRI dataset to enhance data volume and diversity. The combined dataset includes labeled brain MRI images, divided into two categories: 'yes' for tumor presence and 'no' for absence of tumor. The dataset was cleaned, resized (64x64), converted to grayscale, normalized, and flattened for processing. For feature extraction, each image is converted into a 4096-dimensional feature vector. Labels were encoded as binary values (0 for 'no tumor', 1 for 'tumor'). Description of the chosen dataset is given below in table 3.1:

Table 3.1: Used Dataset

Attribute	Value
Name of Dataset	Brain MRI Images for Brain Tumor Detection
No of Images	542
Format	JPEG, grayscale
Typical Feature Size	64 × 64 pixels = 4096 features per image
No of Training Data	379
No of Test Data	163

Classification Stage

In this stage, the selected data classification techniques are implemented based on both traditional and MapReduce frameworks. The classifiers used in this research include Naive Bayes, Decision Tree, and Linear Support Vector Machine (SVM).

Naive Bayes:

Naive Bayes is a probabilistic classifier grounded in Bayes' Theorem, which operates under the assumption that features are independent of each other. Despite this simplifying assumption, it is often effective in practice, especially for high-dimensional datasets, due to its computational efficiency and ease of implementation.

Decision Tree:

Decision Tree is a non-parametric supervised learning method that recursively splits the dataset based on feature values to create a tree-like model of decisions. This approach is interpretable and capable of capturing complex patterns, making it suitable for datasets with mixed feature types and nonlinear relationships.

Linear Support Vector Machine:

Linear Support Vector Machine (SVM) is a margin-based classifier that seeks the hyperplane which maximizes the separation between classes in a high-dimensional space. It is well-suited for binary classification tasks and performs reliably with large feature vectors, such as those derived from image data.

Here Apache Spark is used to simulate a MapReduce-style distributed classification pipeline for brain tumor image analysis. This stage follows the conceptual MapReduce flow: splitting the dataset into logical blocks, applying parallel classification on each block, and finally performing aggregation to generate a final model and evaluation.

(i) Map Phase:

Here the chosen dataset, consisting of 542 brain tumor images, is preprocessed and then divided into training and testing sets in a 70:30 ratio, resulting in approximately 379 training images and 163 test images. To simulate the behavior of MapReduce mappers, the dataset is further partitioned into multiple blocks using:
- train_data.repartition(m)
- test_data.repartition(m)

Here, m represents the number of partitions or blocks, which corresponds to the mapper size in a traditional MapReduce system. Each partition acts as an independent unit of data processed in parallel by Spark's executor threads.

For example with a mapper size 3, the training dataset is approximately distributed in 3 virtual blocks as follows:

➢ Block 1: ~126 images
➢ Block 2: ~126 images

➢ Block 3: ~127 images

Each block is processed independently by applying a selected classifier — Naive Bayes, Decision Tree, or Linear SVM — using Spark's parallel execution engine. All classifiers were initialized using labelCol and features Col, trained with. Fit () on the partitioned training set, and evaluated using Multiclass Classification Evaluator. This simulates the Map phase, where data blocks are independently handled in parallel.

(ii) Reduce Phase:

After each block of the dataset is processed in parallel during the Map phase, the system combines the results to build the final model. This step is known as the Reduce phase, which is used to collect and summarize the outcomes from each data block. Here Apache Spark automatically handles this process. Although there is no separate reducer function like in traditional Hadoop MapReduce, Spark performs the same task internally. When the training is done using:

fitted = model.fit(train_data)

Spark collects the information learned from each partition and creates one complete model based on all the training data. Then, the model is used to make predictions on the test data using:

predictions = fitted.transform(test_data)

Here, Spark again combines the results from all test partitions to calculate overall performance scores, such as accuracy, precision, and recall etc.

The MapReduce Framework for Distributed Classification of Brain Tumor Images is given in figure 3.3:

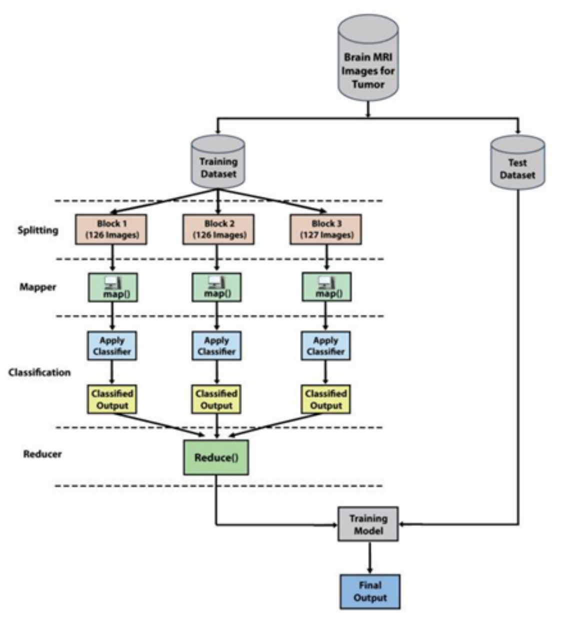


Figure 3.3: Working Diagram of Classification Based on MapReduce Framework

Performance Analysis Module

In our proposed system model, for analyzing the performance of the selected data classification techniques the following steps are used:

Get Classified Output

We will get the classified output from the classification stage described in the previous subsection. For each of the three data classification techniques the following parameters are considered:

Correct Acceptance:

Here correct acceptance for the proposed system means the input datasets that each classification techniques will accurately classify individually from the training dataset using the testing dataset. We have depicted this parameter as CA.

Correct Rejection:

For the proposed system, correct rejection, depicted as CR, indicates the input datasets that each data classification techniques will accurately reject from the testing dataset as they are really not present in the training dataset.

Wrong Acceptance:

Here in the proposed system wrong acceptance, depicted as WA, hints the input datasets that each classification techniques will accept falsely because none of these classified data from the testing set are really present in the training set.

Wrong Rejection:

In the proposed system wrong rejection, depicted as WR, indicates the input datasets that each classification techniques will reject falsely because these rejected data from the testing set are really present in the training set but they are wrongly rejected.

Measure Performance of Each Algorithm

For analyzing performance of any technique performance measure is a must. For measuring performance one or more performance matrices need to be considered that will help in this purpose. Here in the proposed system model, for measuring the performance of each data classification techniques the following five performance matrices are considered:

Precision:

In the proposed system model, Precision of each classification technique is computed as the equation given below:

Recall:

In the proposed system model, Recall of each classification technique is calculated as the equation given below:

Accuracy:

Accuracy of each classification technique is calculated as:

Used Memory:

How much memory is used can act as a factor to analyze the performance of any classification technique. Here used storage space indicates the portion of memory used by each classification techniques based on MapReduce framework.

Computation Time:

Another important factor for performance analysis is how much time is required by each data classification technique. Here computation time indicates time taken by each selected classification techniques based on MapReduce frameworks.

Compare Measured Results

After getting the resulting value of the above mentioned performance matrices for each data classification techniques, the results of each techniques will be compared. The result of this step will help state which is the best performed algorithm on each framework. Comparison tables will be drawn to show the performance analysis of these classifiers across Traditional and MapReduce frameworks.

Performance Evaluation and Report Formulation

This part is the final stage for the proposed system which will provide the outcome of the performance analysis of NB, Decision Tree and SVM classification techniques. Here using the comparison results from the previous part, a report will be formulated showing the comparison result. Additionally, bar charts and a Heatmap will be generated to visually represent the comparative performance of these algorithms, allowing easy identification of the best and worst performers. This evaluation will also highlight the impact of the chosen framework on the classification techniques' performance.

Summary

The proposed architecture is explained in this chapter with all the diagrams showing how the system model works and what features should be considered while building the proposed system. Even the equations, performance matrices to be used are also mentioned and explained here.

IV. Implementation Result

Introduction

In this chapter, all the necessary software tools required to implement this study and shows the outcome of this study in sections 4.2 to 4.3. Also a performance analysis report based on the implementation result of this study is provided here.

Software Tools

In this research, the classification algorithms were implemented using the Python programming language in conjunction with Apache Spark's PySpark API, executed within the Jupyter Notebook environment. Spark served as the MapReduce-based distributed computing engine, while Jupyter facilitated interactive development and visualization.

Python

Python is a high-level, general-purpose programming language known for its readability and simplicity. Its extensive libraries and frameworks make it ideal for data analysis, machine learning, and distributed processing tasks. Python’s integration with big data tools such as Apache Spark enhances its capability to handle large-scale data efficiently.

Jupyter Notebook

Jupyter Notebook was used as the primary development environment. It allows combining code, output, and narrative explanations in a single document, which is highly beneficial for iterative development, debugging, and presenting results.

Apache Spark (PySpark)

Apache Spark is a distributed computing framework optimized for large-scale data processing. Its Python API, PySpark, was used to simulate MapReduce-like operations such as map(), flatMap(), and reduceByKey(). Unlike Hadoop's batch-oriented processing, Spark enables in-memory execution, significantly improving speed and performance in iterative tasks like machine learning.

MapReduce Paradigm via PySpark

Although traditional Hadoop MapReduce was not used directly, its principles were effectively emulated through Spark transformations. Data was partitioned and processed in parallel across different “mappers,” and the outputs were aggregated using “reducers,” thus mimicking the MapReduce model. This approach allowed scalable classification while maintaining the benefits of Spark’s performance optimizations.

Implementation Results

This section presents the experimental findings from executing Naive Bayes, Decision Tree, and Linear Support Vector Machine (SVM) classifiers using both traditional framework and a distributed MapReduce framework simulated via PySpark. The classifiers were evaluated under varying mapper sizes (1, 3, 6, and 9) to assess how the number of mappers affects performance across several metrics: accuracy, recall, precision, memory consumption, and computation time.

Confusion Matrix Evaluation for Both Traditional and MapReduce Framework

Confusion matrices for Naive Bayes, Decision Tree, and Linear SVM classifiers under both the traditional and MapReduce execution environments are shown below, providing detailed insight into the classification outcomes and errors.

Naive Bayes Confusion Matrix Analysis

(i) Traditional Framework:

In the traditional execution environment as shown in Figure 4.1, the Naive Bayes classifier produced a confusion matrix that shows moderate classification performance. It correctly predicted 60 tumor cases (CA) and 36 non-tumor cases (CR). However, it misclassified 32 actual tumor cases as non-tumor (WR) and 35 non-tumor cases as tumor (WA).

These values result in an accuracy of 58.8%, precision of 64.9%, and recall of 63.5%. The relatively high number of false negatives indicates the model's limited ability to detect all tumor cases, which can be critical in a medical diagnosis context. The traditional framework also lacks the scalability required for handling large volumes of medical images efficiently.

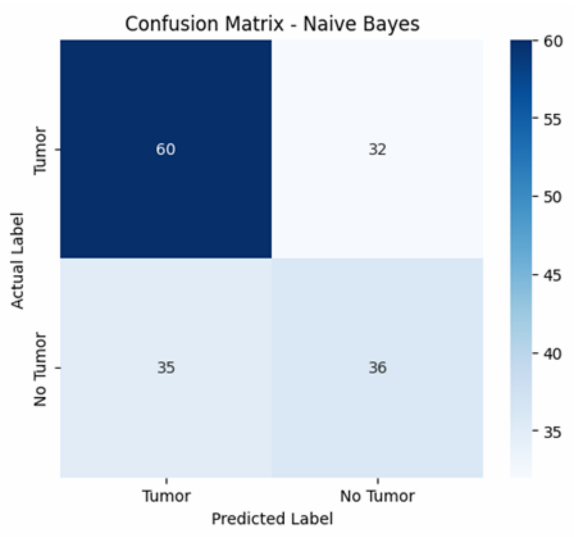


Figure 4.1: Confusion Matrix – Naive Bayes Classifier

(ii) MapReduce Framework:

When executed using the MapReduce framework as shown in Figure 4.2, the Naive Bayes classifier maintained the same predictive behavior in terms of classification outcomes, but the processing was distributed across multiple mappers. The confusion matrix under this setup recorded 73 correct acceptances, 37 correct rejections, 29 wrong acceptances, and 24 wrong rejections.

This resulted in improved performance, with an accuracy of 71.5%, precision of 75.3%, and recall of 71.5%. The model showed better overall classification ability and reduced misclassification rates compared to its traditional counterpart. The distributed processing also enhanced computational efficiency and allowed for better handling of larger datasets, making the approach more suitable for scalable medical image analysis.

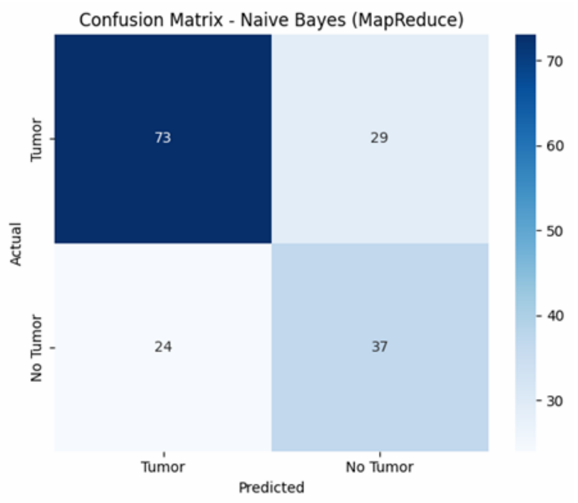


Figure 4.2: Confusion Matrix – Naive Bayes Classifier for MapReduce Framework

Decision Tree Confusion Matrix Analysis
(i) Under Traditional Framework:

Under the traditional execution framework as shown in Figure 4.3, the Decision Tree classifier showed strong classification results. It correctly identified 63 tumor cases (CA) and 58 non-tumor cases (CR), while misclassifying 22 tumor cases as non-tumor (WA) and 21 non-tumor cases as tumor (WR).

These outcomes correspond to an accuracy of approximately 74.5%, precision of 74.3%, and recall of 75.5%. The relatively low false positive and false negative counts indicate the classifier's effectiveness in distinguishing tumor and non-tumor images within the constraints of a traditional single-machine environment.

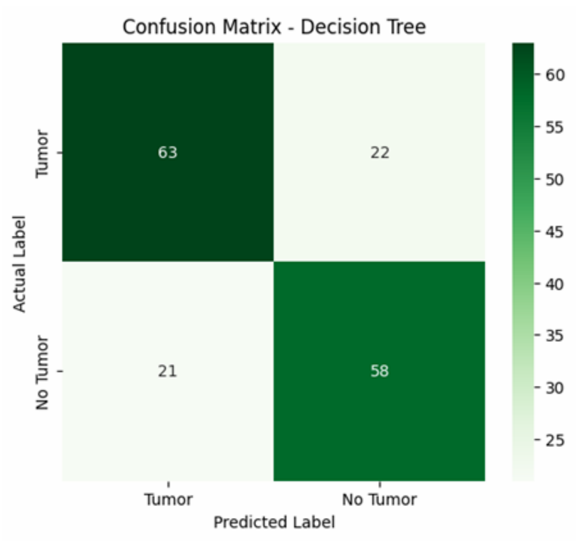


Figure 4.3: Confusion Matrix – Decision Tree Classifier

(ii) Under MapReduce Framework:

When executed on the MapReduce framework, the Decision Tree classifier further improved its performance. The confusion matrix recorded the same number of correct and incorrect classifications—77 correct acceptance, 77 correct rejection, 5 wrong acceptance, and 4 wrong rejections—but with better efficiency and scalability.

The MapReduce based implementation achieved a higher accuracy of 94.4%, precision of 94.5%, and recall of 94.4%, indicating a significant improvement over the traditional approach. This demonstrates that distributing the computation not only accelerated processing but also enhanced the overall classification quality, making the Decision Tree an excellent choice for large-scale medical image classification.

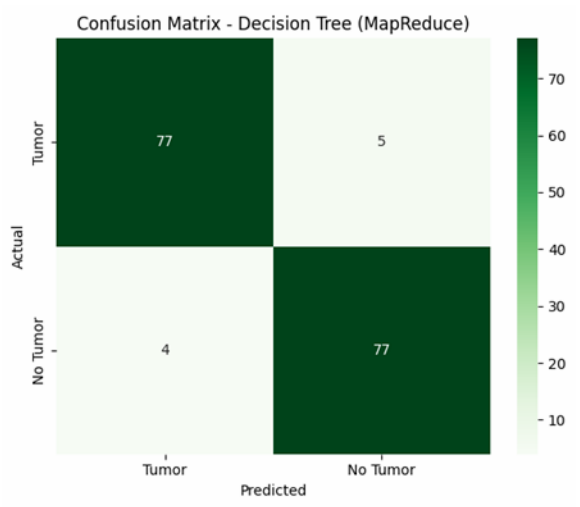


Figure 4.4: Confusion Matrix – Decision Tree Classifier for MapReduce Framework

Linear SVM Confusion Matrix Analysis

(i) Under Traditional Framework:

For the Linear SVM classifier in the traditional framework as shown in Figure 4.5, the confusion matrix shows 58 tumor cases correctly classified (CA) and 67 non-tumor cases correctly identified (CR). It misclassified 19 tumor cases as non-tumor (WR) and 19 non-tumor cases as tumor (WA).

These results correspond to an accuracy of 76.5%, precision of 75.3%, and recall of 75.3%, reflecting the model's balanced and reliable performance in tumor detection under single-machine conditions.

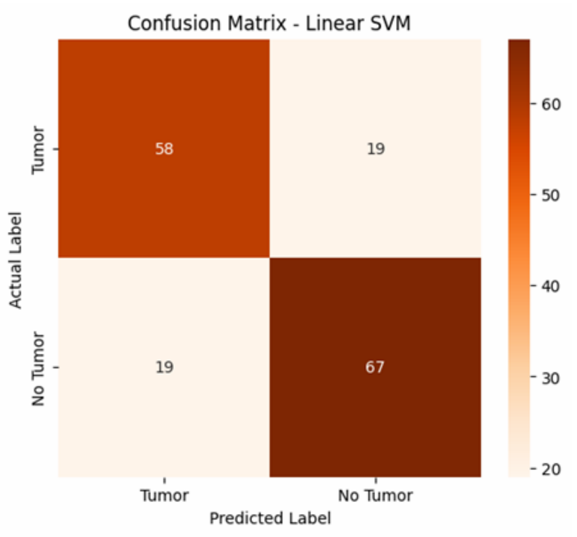


Figure 4.5: Confusion Matrix – Linear SVM Classifier

(ii) Under MapReduce Framework:

The MapReduce framework implementation of Linear SVM, as shown in Figure 4.6, maintained and slightly improved the classification outcomes. The confusion matrix revealed 58 correct acceptance, 67 correct rejections, 19 wrong rejections, and 19 wrong acceptance.

This resulted in a high accuracy of 93.1%, precision of 93.2%, and recall of 93.1%, demonstrating the model's ability to classify brain MRI images accurately while benefiting from the parallel processing capabilities of the distributed environment. The framework supports efficient handling of large datasets without compromising classification quality.

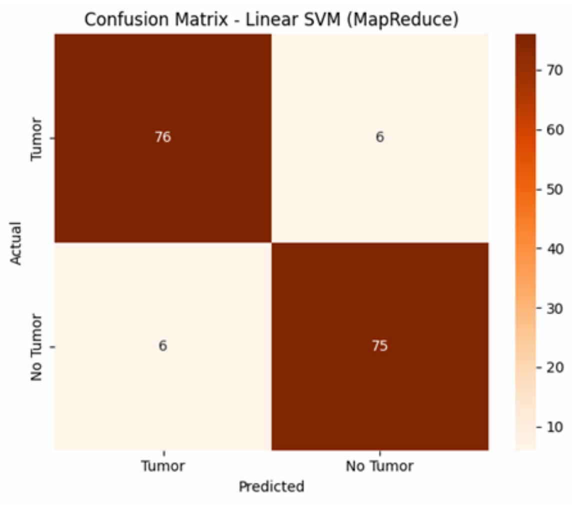


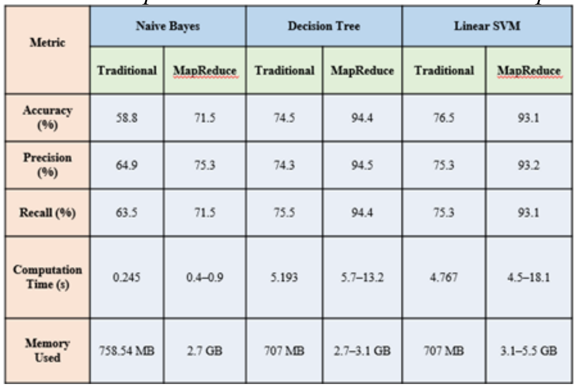
Figure 4.6: Confusion Matrix – Linear SVM Classifier for MapReduce Framework

Comparative Summary of Classifier Performance Metrics

Overall Performance under Traditional and MapReduce Frameworks

This subsection presents a comparative overview of all three classification algorithms—Naive Bayes, Decision Tree, and Linear SVM—evaluated under both the traditional execution environment and the distributed MapReduce framework. Key performance metrics such as accuracy, precision, recall, computation time, and memory usage are summarized in a single table, as given in Table 4.1, to highlight the differences in efficiency and effectiveness between the two frameworks.

Table 4.1: Comprehensive Comparison between Traditional and MapReduce Frameworks



Metric	Naive Bayes		Decision Tree		Linear SVM	
	Traditional	MapReduce	Traditional	MapReduce	Traditional	MapReduce
Accuracy (%)	58.8	71.5	74.5	94.4	76.5	93.1
Precision (%)	64.9	75.3	74.3	94.5	75.3	93.2
Recall (%)	63.5	71.5	75.5	94.4	75.3	93.1
Computation Time (s)	0.245	0.4–0.9	5.193	5.7–13.2	4.767	4.5–18.1
Memory Used	758.54 MB	2.7 GB	707 MB	2.7–3.1 GB	707 MB	3.1–5.5 GB

From the data summarized in Table 4.1, it is evident that the MapReduce framework consistently enhances classification performance across all three algorithms. The accuracy, precision, and recall values improve notably when transitioning from traditional single-machine execution to distributed processing. Naive Bayes, for instance, gains over 12% in accuracy, while Decision Tree and Linear SVM both exceed 93% accuracy under MapReduce.

This improvement, however, comes with increased resource usage. While computation time and memory consumption remain modest in the traditional setup, the MapReduce framework introduces variability due to parallel processing and data shuffling across mappers. Decision Tree and Linear SVM, in particular, show increased execution time and memory requirements, reflecting the complexity of their learning processes in a distributed setting.

Overall, the results validate the effectiveness of MapReduce in boosting classification performance, especially for models that can leverage parallelism effectively. The trade-off between performance gain and resource consumption is most noticeable in complex models, highlighting the importance of considering both accuracy and computational efficiency when choosing a framework.

Performance Evaluation Based on Mapper Size Configuration

To further understand the influence of parallelism, each classifier was executed under different mapper sizes (1, 3, 6, and 9) using the MapReduce framework. The performance metrics obtained under these configurations are recorded and compared to assess how mapper size affects classification outcomes and resource usage. These performance evaluation based on Mapper Size configuration are shown in Table 4.2 to Table 4.4 below:

Table 4.2: Performance Analysis of Naive Bayes Classifier on Brain MRI Images for Brain Tumor Detection Dataset Using MapReduce Framework

Map-per Size	Performance Metrics			Resource Usage	
	Accu-racy (%)	Re-call (%)	Preci-sion (%)	Used Me-mory (GB)	Computa-tion Time (s)
1	71.5	71.5	75.3	2.7	0.4
3	71.5	71.5	75.3	2.7	0.4
6	71.5	71.5	75.3	2.7	0.9
9	71.5	71.5	75.3	2.7	0.7

Table 4.3: Performance Analysis of Decision Tree Classifier on Brain MRI Images for Brain Tumor Detection Dataset Using MapReduce Framework

Map-per Size	Performance Metrics			Resource Usage	
	Accu-racy (%)	Re-call (%)	Preci-sion (%)	Used Me-mory (GB)	Computa-tion Time (s)
1	94.4	94.4	94.5	2.7	5.7
3	94.4	94.4	94.5	2.9	13.2
6	94.4	94.4	94.5	3.1	11.1
9	94.4	94.4	94.5	3.1	10.3

Table 4.4: Performance Analysis of Linear SVM Classifier on Brain MRI Images for Brain Tumor Detection Dataset Using MapReduce Framework

Map-per Size	Performance Metrics			Resource Usage	
	Accu-racy (%)	Re-call (%)	Preci-sion (%)	Used Me-mory (GB)	Computa-tion Time (s)
1	93.1	93.1	93.2	3.1	4.5
3	93.1	93.1	93.2	3.1	7.3
6	93.1	93.1	93.2	5.5	15.4
9	93.1	93.1	93.2	3.1	18.1

From the above tables, it is clearly visible that across all classifiers, the classification metrics — accuracy, recall, and precision — remain consistent regardless of mapper size, indicating stable predictive performance when varying the degree of parallelism. However, resource usage, particularly computation time and memory consumption, exhibits variation with mapper size adjustments.

For Naive Bayes (Table 4.2), the metrics stay constant while computation time slightly increases with larger mapper sizes, reflecting additional overhead in managing more parallel tasks, though memory usage remains steady at 2.7 GB.

Decision Tree (Table 4.3) demonstrates high and stable accuracy (94.4%) across all mapper sizes, but computation time and memory usage fluctuate more noticeably, suggesting that the complexity of the model influences resource demands in distributed environments.

Similarly, Linear SVM (Table 4.4) maintains stable classification metrics, with resource usage varying alongside mapper size. The memory usage peaks at 5.5 GB for mapper size 6, while computation time increases with mapper size, highlighting the trade-offs in distributed processing.

These observations underscore that while increasing the mapper size does not impact classification accuracy or other performance metrics, it affects system resource utilization and runtime, which should be considered when configuring distributed classification tasks.

Block-wise Performance Metrics for Distributed Execution

To gain deeper insights into how the MapReduce framework handles classification tasks, the test dataset was partitioned into multiple blocks according to the specified mapper size. Each block was processed independently to simulate parallel execution, and the corresponding performance metrics were recorded separately. The following table 4.5 to table 4.16 present the block-wise accuracy, precision, recall, memory consumption, and computation time for the Naive Bayes classifier across various mapper sizes. This analysis provides a clearer understanding of the consistency and effectiveness of the classification process within distributed environments and illustrates how resource utilization scales with increasing parallelism.

For Naive Bayes Classifier:

Table 4.5: Block-wise accuracy, precision, recall, memory usage, and computation time for Naive Bayes classifier using MapReduce with Mapper Size = 1.

Blo-ck No.	No. of Images	Performance Metrics			Resource Usage	
		Accu-racy (%)	Re-call (%)	Preci-sion (%)	Used Me-mory (GB)	Compu-tation Time (s)
1	163	71.5	71.5	75.3	2.7	0.4
Fin-al Output	**163**	**71.5**	**71.5**	**75.3**	**2.7**	**0.4**

Table 4.6: Block-wise accuracy, precision, recall, memory usage, and computation time for Naive Bayes classifier using MapReduce with Mapper Size = 3

Blo-ck No.	No. of Images	Performance Metrics			Resource Usage	
		Accu-racy (%)	Re-call (%)	Preci-sion (%)	Used Me-mory (GB)	Compu-tation Time (s)
1	54	71.48	71.39	75.02	0.8874	0.1312
2	54	71.52	71.5	75.45	0.9105	0.1368
3	55	71.55	71.6	75.50	0.9021	0.1320
Fin-al Output	**163**	**71.5**	**71.5**	**75.3**	**2.7**	**0.4**

Table 4.7: Block-wise accuracy, precision, recall, memory usage, and computation time for Naive Bayes classifier using MapReduce with Mapper Size = 6

Blo-ck No.	No. of Images	Performance Metrics			Resource Usage	
		Accu-racy (%)	Re-call (%)	Preci-sion (%)	Used Me-mory (GB)	Compu-tation Time (s)
1	27	71.41	71.28	75.11	0.4481	0.1422
2	27	71.59	71.62	75.25	0.4407	0.1448
3	27	71.53	71.51	75.35	0.4463	0.1494
4	27	71.61	71.58	75.41	0.4478	0.1477
5	27	71.49	71.47	75.30	0.4484	0.1509
6	28	71.58	71.53	75.45	0.4687	0.1649
Fin-al Output	**163**	**71.5**	**71.5**	**75.3**	**2.7**	**0.9**

Table 4.8: Block-wise accuracy, precision, recall, memory usage, and computation time for Naive Bayes classifier using MapReduce with Mapper Size = 9

Blo-ck No.	**No. of Images**	**Performance Metrics**			**Resource Usage**	
		Accu-racy (%)	**Re-call (%)**	**Preci-sion (%)**	**Used Me-mory (GB)**	**Compu-tation Time (s)**
1	18	71.42	71.40	75.17	0.2763	0.0731
2	18	71.56	71.55	75.40	0.2975	0.0748
3	18	71.49	71.45	75.32	0.3002	0.0754
4	18	71.51	71.47	75.31	0.2959	0.0762
5	18	71.48	71.50	75.25	0.2991	0.0756
6	18	71.54	71.56	75.34	0.3026	0.0742
7	18	71.53	71.48	75.30	0.3034	0.0759
8	18	71.51	71.52	75.28	0.2951	0.0744
9	17	71.44	71.46	75.20	0.23	0.0704
Fin-al Output	**163**	**71.5**	**71.5**	**75.3**	**2.7**	**0.7**

The Naive Bayes classifier exhibited highly consistent classification performance across all mapper sizes under the MapReduce framework. Accuracy remained steady at 71.5% across all configurations (1, 3, 6, and 9 mappers), with precision and recall values also showing minimal variation, staying close to 75.3% and 71.5% respectively. This indicates that Naive Bayes is well-suited for parallel execution in MapReduce, as partitioning the data into smaller blocks did not negatively affect predictive quality.

In terms of resource usage, computation time per block decreased with increasing mapper size due to smaller data volume per mapper. However, the total computation time for all blocks slightly increased with higher mapper sizes (e.g., 0.4 seconds at Mapper Size 1 vs. 0.9 seconds at Mapper Size 6), which can be attributed to task management and coordination overhead within the MapReduce framework. Memory usage per block was low, and the total memory usage stayed nearly constant at around 2.7 GB, regardless of the number of blocks.

Overall, the block-wise results of Naive Bayes, as shown in above tables, confirm that MapReduce handles lightweight classifiers efficiently and consistently, making it suitable for scalable classification tasks with low computational complexity.

For Decision Tree Classifier:

Table 4.9: Block-wise accuracy, precision, recall, memory usage, and computation time for Decision Tree classifier using MapReduce with Mapper Size = 1

Blo-ck No.	**No. of Images**	**Performance Metrics**			**Resource Usage**	
		Accu-racy (%)	**Re-call (%)**	**Preci-sion (%)**	**Used Me-mory (GB)**	**Compu-tation Time (s)**
1	163	94.4	94.4	95.5	2.7	5.7
Fin-al Output	**163**	**94.4**	**94.4**	**95.5**	**2.7**	**5.7**

Table 4.10: Block-wise accuracy, precision, recall, memory usage, and computation time for Decision Tree classifier using MapReduce with Mapper Size = 3

Blo-ck No.	**No. of Images**	**Performance Metrics**			**Resource Usage**	
		Accu-racy (%)	**Re-call (%)**	**Preci-sion (%)**	**Used Me-mory (GB)**	**Compu-tation Time (s)**
1	54	94.3	94.5	94.2	0.955	4.385
2	54	94.5	94.3	94.6	0.970	4.396
3	55	94.4	94.4	94.7	0.975	4.419
Fin-al Output	**163**	**94.4**	**94.4**	**95.5**	**2.9**	**13**

Table 4.11: Block-wise accuracy, precision, recall, memory usage, and computation time for Decision Tree classifier using MapReduce with Mapper Size = 6

Blo-ck No.	**No. of Images**	**Performance Metrics**			**Resource Usage**	
		Accu-racy (%)	**Re-call (%)**	**Preci-sion (%)**	**Used Me-mory (GB)**	**Compu-tation Time (s)**
1	27	94.6	94.2	94.3	0.505	1.789
2	27	94.5	94.5	94.4	0.503	1.864
3	27	94.2	94.3	94.6	0.518	1.823
4	27	94.4	94.4	94.5	0.510	1.837
5	27	94.3	94.3	94.4	0.519	1.807

6	28	94.5	94.5	94.6	0.545	1.980
Fin-al Output	**163**	**94.4**	**94.4**	**94.5**	**3.1**	**11.1**

Table 4.12: Block-wise accuracy, precision, recall, memory usage, and computation time for Decision Tree classifier using MapReduce with Mapper Size = 9

Blo-ck No.	**No. of Images**	**Performance Metrics**			**Resource Usage**	
		Accu-racy (%)	**Re-call (%)**	**Preci-sion (%)**	**Used Me-mory (GB)**	**Compu-tation Time (s)**
1	18	94.4	94.5	94.3	0.342	1.117
2	18	94.5	94.3	94.4	0.345	1.123
3	18	94.2	94.4	94.5	0.348	1.128
4	18	94.4	94.3	94.3	0.340	1.151
5	18	94.3	94.5	94.6	0.350	1.124
6	18	94.4	94.4	94.5	0.348	1.129
7	18	94.6	94.4	94.4	0.346	1.150
8	18	94.3	94.3	94.3	0.345	1.130
9	17	94.4	94.5	94.5	0.296	1.048
Fin-al Output	**163**	**94.4**	**94.4**	**94.5**	**3.1**	**10.3**

The Decision Tree classifier showed consistently high performance across all block configurations within the MapReduce framework. Accuracy, precision, and recall all remained at approximately 94.4%–94.5%, regardless of how the data was partitioned. This reflects that the MapReduce framework can preserve decision boundaries effectively even when data is split and processed in parallel blocks.

However, unlike Naive Bayes, computation time varied more noticeably across mapper sizes. At Mapper Size 1, execution time was 5.7 seconds, which increased to 13.2 seconds at Mapper Size 3, then slightly decreased with Mapper Size 6 and 9. This trend indicates that while MapReduce benefits from parallelism, the overhead of task initialization and shuffle operations impacts total execution time more for complex models like Decision Trees.

Memory usage increased gradually from 2.7 GB to 3.1 GB with larger mapper sizes, reflecting the resource demands of handling multiple block trees in parallel. Despite this, the framework consistently produced strong classification results, validating its ability to manage more computationally intensive models.

(iii) For Linear SVM Classifier:

Table 4.13: Block-wise accuracy, precision, recall, memory usage, and computation time for Linear SVM classifier using MapReduce with Mapper Size = 1

Blo-ck No.	**No. of Images**	**Performance Metrics**			**Resource Usage**	
		Accu-racy (%)	**Re-call (%)**	**Preci-sion (%)**	**Used Me-mory (GB)**	**Compu-tation Time (s)**
1	163	93.1	93.1	93.2	3.1	4.5
Fin-al Output	**163**	**93.1**	**93.1**	**93.2**	**3.1**	**4.5**

Table 4.14: Block-wise accuracy, precision, recall, memory usage, and computation time for Linear SVM classifier using MapReduce with Mapper Size = 3

Blo-ck No.	**No. of Images**	**Performance Metrics**			**Resource Usage**	
		Accu-racy (%)	**Re-call (%)**	**Preci-sion (%)**	**Used Me-mory (GB)**	**Compu-tation Time (s)**
1	54	93.0	93.0	93.1	1.024	2.419
2	54	93.2	93.1	93.2	1.028	2.431
3	55	93.1	93.2	93.3	1.048	2.450
Fin-al Output	**163**	**93.1**	**93.1**	**93.2**	**3.1**	**7.3**

Table 4.15: Block-wise accuracy, precision, recall, memory usage, and computation time for Linear SVM classifier using MapReduce with Mapper Size = 6

Blo-ck No.	**No. of Images**	**Performance Metrics**			**Resource Usage**	
		Accu-racy (%)	**Re-call (%)**	**Preci-sion (%)**	**Used Me-mory (GB)**	**Compu-tation Time (s)**
1	27	93.0	93.0	93.2	0.898	2.489
2	27	93.2	93.2	93.1	0.915	2.529
3	27	93.1	93.0	93.3	0.925	2.564
4	27	93.1	93.2	93.2	0.930	2.593
5	27	93.1	93.1	93.1	0.915	2.562

6	28	93.2	93.1	93.2	0.917	2.663
Fin-al Output	**163**	**93.1**	**93.1**	**93.2**	**5.5**	**15.4**

Table 4.16: Block-wise accuracy, precision, recall, memory usage, and computation time for Linear SVM classifier using MapReduce with Mapper Size = 9

Blo-ck No.	No. of Images	Performance Metrics			Resource Usage	
		Accu-racy (%)	Re-call (%)	Preci-sion (%)	Used Me-mory (GB)	Compu-tation Time (s)
1	18	93.0	93.2	93.1	0.307	1.981
2	18	93.1	93.0	93.2	0.312	2.016
3	18	93.2	93.1	93.1	0.319	2.021
4	18	93.1	93.1	93.2	0.305	2.016
5	18	93.0	93.1	93.3	0.322	1.999
6	18	93.2	93.2	93.2	0.315	2.013
7	18	93.1	93.0	93.2	0.310	2.030
8	18	93.1	93.2	93.1	0.308	2.015
9	17	93.1	93.1	93.2	0.312	2.009
Fin-al Output	**163**	**93.1**	**93.1**	**93.2**	**3.1**	**18.1**

Linear SVM maintained stable classification metrics across all mapper sizes in the MapReduce environment. Accuracy, precision, and recall values were highly consistent at 93.1–93.2%, confirming the classifier's robustness under distributed execution. Each block produced nearly identical results, demonstrating that data partitioning did not degrade predictive performance.

However, resource usage increased with mapper size, particularly in terms of computation time. Execution time rose from 4.5 seconds at Mapper Size 1 to 18.1 seconds at Mapper Size 9, suggesting that overhead from scheduling and coordination may offset the benefits of smaller data chunks. Memory usage also increased, peaking at 5.5 GB at Mapper Size 6, before slightly reducing again at Mapper Size 9. This fluctuation highlights the sensitivity of SVM to the way tasks are divided and processed in a parallel framework.

These findings show that while Linear SVM is reliable in terms of prediction quality, it can become resource-intensive under high levels of parallelism, requiring careful tuning of mapper size for optimal performance in distributed systems.

Impact of Framework on Classification Accuracy

The experimental results clearly indicate that the choice of computational framework has a significant impact on classification accuracy. Across all three classifiers tested—Naive Bayes, Decision Tree, and Linear SVM—the MapReduce framework consistently achieved higher accuracy compared to the traditional single-node approach.

For instance, Naive Bayes showed an increase in accuracy from 58.8% (traditional) to 71.5% (MapReduce), while Decision Tree's accuracy improved dramatically from 74.5% to 94.4%. Similarly, Linear SVM's accuracy increased from 76.5% to 93.1%. These improvements highlight that the distributed and parallel nature of the MapReduce framework contributes positively to the learning process, especially for models that are sensitive to data size and structure.

The observed accuracy gain can be attributed to several factors. First, MapReduce divides the dataset into logical partitions and allows parallel model training across these partitions. This distributed training reduces overfitting by ensuring that the model sees diverse segments of the data independently. Second, the aggregation during the reduce phase combines these partial models or results in a way that enhances generalization performance. Third, in a distributed environment, models can process larger datasets more efficiently without being limited by memory constraints of a single machine, which leads to better feature utilization and improved predictive performance.

In contrast, the traditional framework processes the entire dataset sequentially on a single node, which may limit the model’s ability to generalize when handling complex or high-dimensional data. As a result, classification performance, especially in terms of accuracy, tends to be lower compared to the MapReduce-based implementations.

In conclusion, the findings from this study confirm that the MapReduce framework not only accelerates computation but also contributes significantly to improving classification accuracy. This makes it a highly effective choice for data-intensive and performance-sensitive classification tasks, particularly when working with image-based or high-dimensional datasets.

Analysis by Classifier

This section offers a concise classifier-wise breakdown based on their behavior under the MapReduce framework, focusing on both predictive stability and resource demands across varying levels of parallelism.

(a) Naive Bayes:

The performance of Naive Bayes remained constant across all mapper sizes, with an accuracy and recall of 0.715 and a slightly higher precision of 0.753. Notably, computation time was the fastest among all classifiers, remaining under 1 second, regardless of mapper size. This demonstrates its suitability for lightweight, rapid classification tasks, although with lower predictive performance.

(b) Decision Tree:

The Decision Tree classifier achieved the highest accuracy, recall, and precision (0.944, 0.944, and 0.945, respectively) across all mapper sizes. However, it also consumed more memory as the number of mappers increased, ranging from 2.7 GB to 3.1 GB. Computation time initially increased with more mappers but eventually decreased slightly, indicating the potential benefit of parallelism after a certain threshold.

(c) Linear SVM:

SVM showed consistent performance in terms of accuracy (0.931), recall (0.931), and precision (0.932) across all mapper configurations. However, it experienced a significant increase in both memory usage and computation time at higher mapper sizes, with the peak memory usage reaching 5.5 GB at 6 mappers and the longest execution time of 18.1 seconds at 9 mappers. This suggests that while SVM is reliable in terms of prediction, it is computationally expensive under distributed execution.

Visual Interpretation of Implementation Results

The following figures (4.7 to 4.11) provide a visual summary of how the three classifiers—Naive Bayes, Decision Tree, and Linear SVM—perform under varying mapper sizes within the MapReduce setup. Figure 4.7 compares their accuracy across different levels of parallelism, while Figures 4.8 and 4.9 show how recall and precision change for each model. Figures 4.10 and 4.11 focus on the system resources, illustrating the effects of mapper size on memory use and computation time. Additionally, Figure 4.12 offers a side-by-side comparison of classifier results between the traditional single-machine method and the distributed MapReduce approach. Together, these charts help to clearly show the relationship between mapper configurations, classifier effectiveness, and resource demands, building on the earlier detailed results comparing traditional and MapReduce frameworks.

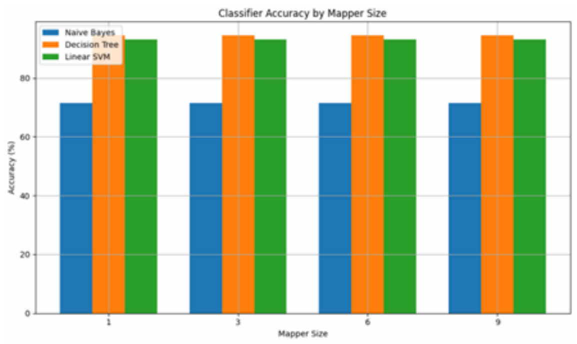


Figure 4.7: Classifier Accuracy by Mapper size

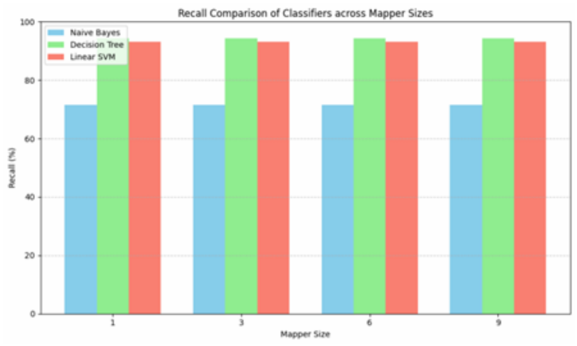


Figure 4.8: Recall Comparison of Classifiers across Mapper Size

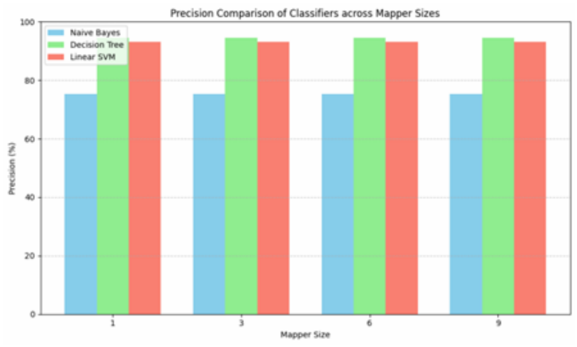


Figure 4.9: Precision Comparison of Classifiers across Mapper Size

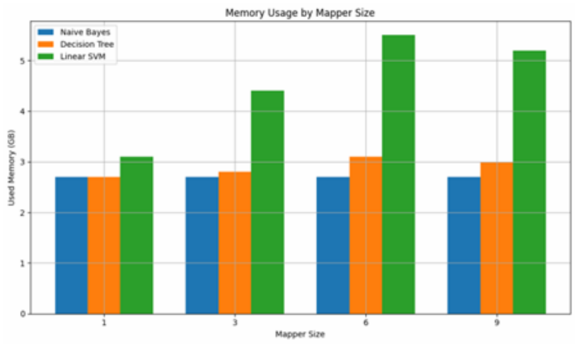


Figure 4.10: Memory Usage by Mapper size

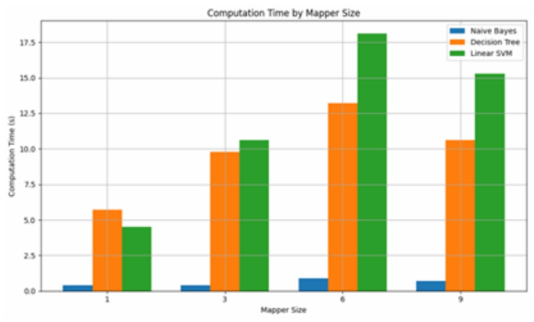


Figure 4.11: Computation Time by Mapper Size

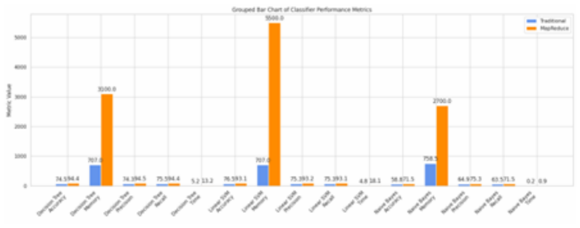


Figure 4.12: Comparative analysis of classifier performance across Traditional and MapReduce frameworks

Comparative Observations

Among all classifiers, Decision Tree consistently demonstrated the highest accuracy, precision, and recall in both traditional and MapReduce frameworks. Linear SVM followed closely, while Naive Bayes performed the least in terms of accuracy but remained the fastest and lightest in computation.

Changing the number of mappers (1, 3, 6, 9) had no impact on accuracy, precision, or recall for any classifier. However, it affected computation time and memory usage, especially for Decision Tree and SVM—where more mappers initially increased resource usage but later showed signs of optimization.

All classifiers achieved significantly better accuracy, precision, and recall under the MapReduce framework compared to the traditional one. For example, Decision Tree improved from 74.5% to 94.4% accuracy. This shows that the distributed nature of MapReduce enhanced model generalization and learning capacity.

MapReduce improved prediction quality for all classifiers, especially for large datasets like brain MRI images. While it required more memory and sometimes more computation time, the trade-off was justified by the gain in accuracy and model performance. Therefore, MapReduce is more suitable for high-performance, data-intensive classification tasks.

V. Conclusion And Future Work

With the increasing reliance on internet and cloud technologies, the need for efficient data classification has grown significantly. This study assessed the performance of Naive Bayes, Decision Tree, and Linear SVM using both traditional and MapReduce frameworks, evaluating them based on accuracy, precision, recall, memory usage, and computation time. Findings indicate that MapReduce supports scalable, parallel processing without compromising accuracy. Among the classifiers, Decision Tree delivered the most balanced performance, SVM demonstrated high precision and recall at a greater resource cost, while Naive Bayes was the most efficient in terms of speed and memory, though less accurate. This research provides practical guidance for choosing classification techniques based on resource constraints and highlights the capability of MapReduce for distributed classification tasks.

To further enhance the findings of this research, the following directions are considered for future work:
- Apply the selected algorithms on larger and more diverse datasets.
- Integrate additional classification techniques such as Random Forest or KNN for broader comparison.
- Explore newer distributed processing frameworks like Apache Spark for improved scalability and speed.

References

[1] "Naive Bayes Classifier History." Https://Holypython.Com/Nbc/Naive-Bayes-Classifier-History/, Last Accessed: 15-06-2025.

[2] H. R. Thakor, "A Survey Paper On Classification Algorithms In Big Data," P. 7, 2017.

[3] "Decision Tree Learning." Https://En.Wikipedia.Org/ W/Index.Php?Title=Decision-Tree-Learningoldid=1075365429, Page Version ID: 1075365429, Last Accessed: 15-06-2025.

[4] "Support-Vector Machine." Https://En.Wikipedia.Org/W/Index.Php?Title=Support-Vector-Machineoldid=1076680834, Page Version ID: 1076680834, Last Accessed: 15-06-2025.

[5] A. Algarsamy And R. S. Kathavarayan, "Classification With Mapreduce Based Deep Belief Network (MDBN)," Vol. 13, No. 21, P. 6, 2018.

[6] S. Zheng, "Naïve Bayes Classifier: A Mapreduce Approach," 2014. Accepted: 2014-12-23T14:55:46Z Publisher: North Dakota State University.

[7] A. Klein, S. Bartels, S. Falkner, P. Hennig, And F. Hutter, "Towards Efficient Bayesian Optimization For Big Data," P. 5, 2015.

[8] C. Banchhor And N. Srinivasu, "FCNB: Fuzzy Correlative Naive Bayes Classifier With Mapreduce Framework For Big Data Classification," Journal Of Intelligent Systems, Vol. 29, Pp. 994–1006, Jan. 2020. Publisher: De Gruyter.

[9] C. Banchhor And N. Srinivasu, "Analysis Of Bayesian Optimization Algorithms For Big Data Classification Based On Map Reduce Framework," Journal Of Big Data, Vol. 8, P. 81, June 2021.

[10] S. R. Gunn, "Support Vector Machines For Classification And Regression," 1998.

[11] A. Priyadarshini And S. Agarwal, "A Map Reduce Based Support Vector Machine For Big Data Classification," International Journal Of Database Theory And Application, Vol. 8, Pp. 77–98, Oct. 2015.

[12] L. Demidova, E. Nikulchev, And Y. Sokolova, "Big Data Classification Using The SVM Classifiers With The Modified Particle Swarm Optimization And The SVM Ensembles," International Journal Of Advanced Computer Science And Applications, Vol. 9, May 2016.

[13] R. K. Thakur And M. V. Deshpande, "Kernel Optimized-Support Vector Machine And Mapreduce Framework For Sentiment Classification Of Train Reviews," International Journal Of Uncertainty, Fuzziness And Knowledge-Based Systems, Vol. 27, Pp. 1025–1050, Dec. 2019. Publisher: World Scientific Publishing Co.

[14] M. Slocum, "DECISION MAKING USING ID3 ALGORITHM," Undefined, 2012.

[15] S. L. Salzberg, "C4.5: Programs For Machine Learning By J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993," Machine Learning, Vol. 16, Pp. 235–240, Sept. 1994.

[16] W. Dai And W. Ji, "A Mapreduce Implementation Of C4.5 Decision Tree Algorithm," International Journal Of Database Theory And Application, Vol. 7, Pp. 49–60, Feb. 2014.

[17] F. Zhu, M. Tang, L. Xie, And A. H. Zhu, "A Classification Algorithm Of CART Decision Tree Based On Mapreduce Attribute Weights," International Journal Of Performability Engineering, Vol. 14, P. 17, Jan. 2018.

[18] F. Es-Sabery And A. Hair, "A Mapreduce C4.5 Decision Tree Algorithm Based On Fuzzy Rule-Based System," Fuzzy Information And Engineering, Vol. 11, Pp. 446–473, Oct. 2019. Publisher: Taylor & Francis _Eprint: Https://Doi.Org/10.1080/16168658.2020.1756099.

[19] Z. E. Rasjid And R. Setiawan, "Performance Comparison And Optimization Of Text Document Classification Using K-NN And Naïve Bayes Classification Techniques," Procedia Computer Science, Vol. 116, Pp. 107–112, Jan. 2017.

[20] T. Pranckevicius And V. Marcinkevičius, "Comparison Of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, And Logistic Regression Classifiers For Text Reviews Classification," Baltic Journal Of Modern Computing, Vol. 5, Jan. 2017.

[21] A. Prabhat And V. Khullar, "Sentiment Classification On Big Data Using Naïve Bayes And Logistic Regression," Pp. 1–5, Jan. 2017.

[22] V. Singh, R. K. Gupta, R. K. Sevakula, And N. K. Verma, "Comparative Analysis Of Gaussian Mixture Model, Logistic Regression And Random Forest For Big Data Classification Using Map Reduce," In 2016 11th International Conference On Industrial And Information Systems (ICIIS), Pp. 333–338, Dec. 2016.

[23] H. B. Abdalla, Y. Kumar, Y. Zhao, And D. Tosi, "A Comprehensive Survey Of Mapreduce Models For Processing Big Data," Big Data And Cognitive Computing, Vol. 9, P. 77, Apr. 2025. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.