Hindawi
Mathematical Problems in Engineering
Volume 2023, Article ID 6048891, 10 pages
https://doi.org/10.1155/2023/6048891

Research Article

A Distributed Framework for Predictive Analytics Using Big Data and MapReduce Parallel Programming

P. Natesan,[1] V. E. Sathishkumar,[2] Sandeep Kumar Mathivanan,[3] Maheshwari Venkatasen,[3] Prabhu Jayagopal,[3] and Shaikh Muhammad Allayear[4]

[1]*Department of Computer Science and Engineering, Kongu Engineering College, Perundurai, Erode 638060, Tamilnadu, India*
[2]*Department of Industrial Engineering, Hanyang University, Seoul, Republic of Korea*
[3]*School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, TamilNadu 632014, India*
[4]*Department of Multimedia and Creative Technology, Daffodil International University, Daffodil Smart City, Khagan, Ashulia, Dhaka, Bangladesh*

Correspondence should be addressed to Shaikh Muhammad Allayear; headmct@daffodilvarsity.edu.bd

Received 18 May 2022; Revised 22 October 2022; Accepted 24 November 2022; Published 1 February 2023

Academic Editor: Adiel T. de Almeida-Filho

With the advancement of Internet technologies and the rapid increase of World Wide Web applications, there has been tremendous growth in the volume of digital data. This takes the digital world into a new era of big data. Various existing data processing technologies are not consistent and scalable in handling the complexity as well as the large-size datasets. Recently, there are many distributed data processing, and programming models have been proposed and implemented to handle big data applications. The open-source-implemented MapReduce programming model in Apache Hadoop is the foremost model for data exhaustive and also computational-intensive applications due to its inherent characteristics of scalability, fault tolerance, and simplicity. In this research article, a new approach for the prediction of target labels in big data applications is developed using a multiple linear regression algorithm and MapReduce programming model, named as MR-MLR. This approach promises optimum values for MAE, RMSE, and determination coefficient (R^2) and thus shows its effectiveness in predictions in big data applications.

1. Introduction

Linear regression is one of the most important prediction algorithms in statistics and machine learning. It is a well-understood algorithm that intends to build a linear association that exists between two attributes, namely, the dependent and the independent attributes. In the past years, the regression method has been widely used in forecasting, prediction, batch process analysis, and chemical calibration [1–5]. Multivariate linear regression (MLR) attempts to create a model to show the relationship between two or more attributes of a given domain [6–10].

In recent days, the regression algorithm has been widely used in data-intensive big data applications for prediction as well as classification. To improve the scalability, runtime management, and computational effectiveness, there are several parallel programming frameworks that implement classification and prediction techniques in a distributed environment. Google's MapReduce framework [11] was the revolutionary parallel programming tool to handle big data applications, built on its own distributed file system, Google File System (GFS) [12]. The MapReduce was a very popular tool in the parallel programming paradigm because of its scalability, simplicity, throughput, and fault tolerance [13]. Hadoop MapReduce is one of the publicly available and extensively used open-source frameworks for large-scale data processing [14], which supports many levels of programming details such as block storage, task scheduling, data management, fault tolerance, and load balancing. The various statistical and machine learning algorithms that were developed for sequential execution in a single-machine context have been modified to run concurrently in multiple machines in the cluster computing environment. Apache developed a set of machine learning libraries supported by the MapReduce model [15] called Mahout, and Yahoo uses a MapReduce-based programming model to analyze their big data [16] generated by their e-mail, news, sports, finance, entertainment, and shopping applications. Urbani et al. [17] implemented a MapReduce-based Web-scale inference engine (WebPIE) for the semantic web scalable ontology reasoning [18]. They demonstrated its performance on a Hadoop cluster with 64 nodes using Bio2RDF, LLDLSR, and LUBM datasets [18]. Ding et al. [19] developed a MapReduce-based multilayered massive data query processing system over the Skyline smart transportation dataset.

In cloud computing environment and commodity clusters, the distributed data processing MapReduce programming model is the best choice for data-intensive analysis due to its simplicity, scalability, and any complex tasks that can be parallelized in the underlying computing resources [20–22]. Currently, the MapReduce runtime is available as a cloud service in the cloud environment; one can easily build a data processing application [23]. Wang et al. [24] developed a MapReduce-based random forest machine learning algorithm called PaRFR (parallel random forest regression) for regression analysis in large-scale population genetic association studies, which involves multivariate traits.

Ashiq et al. [25] integrated the MapReduce framework with the graphics package OpenCV to process video streams in the cloud computing environment and showed that the performance of the system is better in terms of processing time. Yaseen et al. [26] demonstrated a cloud-based, parallelized, completely automated video stream processing system capable of handling large numbers of video streams in a short period of time. Swapnil et al. [27] proposed a novel architecture to use Hadoop-based image interfacing system [28] for processing a large number of images which are executed on a Hadoop cluster concurrently and deliver elevated throughput. Jatmiko et al. [29] developed a MapReduce framework to analyze the biomedical images which detect breast and brain cancer. Huang et al. [30] demonstrated Hadoop-based parallel processing for remotely sensed images. Maillo et al. [31] adopted the MapReduce programming model for k-nearest neighbor (MR-kNN) prediction as well as the classification algorithm. The outcome from the research shows that the experiment is scalable, has an exact parallel implementation, and achieved a good computational time as compared with the sequential version of k-NN.

1.1. The Contributions of This Research Paper

(i) A distributed machine learning multivariate linear regression model to process massively large-sized datasets is proposed

(ii) We executed experiments to show the scalability of the proposed model and compared it with a standalone sequential implementation of the linear regression algorithm

(iii) We evaluated the performance of the proposed research model for different split ratios of training and testing samples

2. Materials and Methods

2.1. Multivariate Linear Regression. A multivariate linear regression model with 'n' predictor variables $x_1, x_2, \ldots x_n$ and a response variable 'y' can be written mathematically as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n, \quad (1)$$

where$y$ represents the predicted or expected value of the response/dependent variable, x_1 through x_n are 'n' distinct independent variables, β_0 represents the intercept value of the regression model, β_i is the ith regression coefficient that decides the used weight by the equation on the ith independent attribute to give the estimated output.

MLR has two procedures, model building through learning and prediction. Learning a multivariate linear regression model is defined as estimating the values of the intercept and regression coefficients used in the model representation with a dataset considered. Given the coefficients, if we substitute in values for all the input variables, the learned model will give us the predicted values of the response variable.

3. The Proposed MapReduce Algorithm

The objective of making a machine learning algorithm is to enable an algorithm to gain knowledge from the past data, i.e., past or present events and gains knowledge, and the knowledge is represented as a model. The learned model is used to compose predictions or classifications/decisions regarding unidentified upcoming events. Figure 1 depicts the flow of the proposed MR-MLR algorithm, and the two phases of the algorithm are described as follows.

3.1. MapReduce-I for Training. The mapper program reads the training instances from the underlying distributed file system HDFS and computes the correlation that exists among the regression variables, the intercept value, and the coefficients for all attributes in the training dataset. The reducer program collects the intercept values and coefficients for all the attributes and computes the average value of it. From the intercept and regression coefficients, a distributed learned MR-MLR is constructed for the given training instances.

3.2. MapReduce-II for Prediction/Classification. The mapper program of MapReduce-II reads the test dataset instances from the underlying distributed file system HDFS and predicts the values of the response/predictor variables. This mapper also calculates the various measures pertaining to the prediction of the response/predictor variable for all the instances, and these measures are passed to the reducer component of the Mapreduce-II. The reducer part collects different performance metrics for the different blocks of data

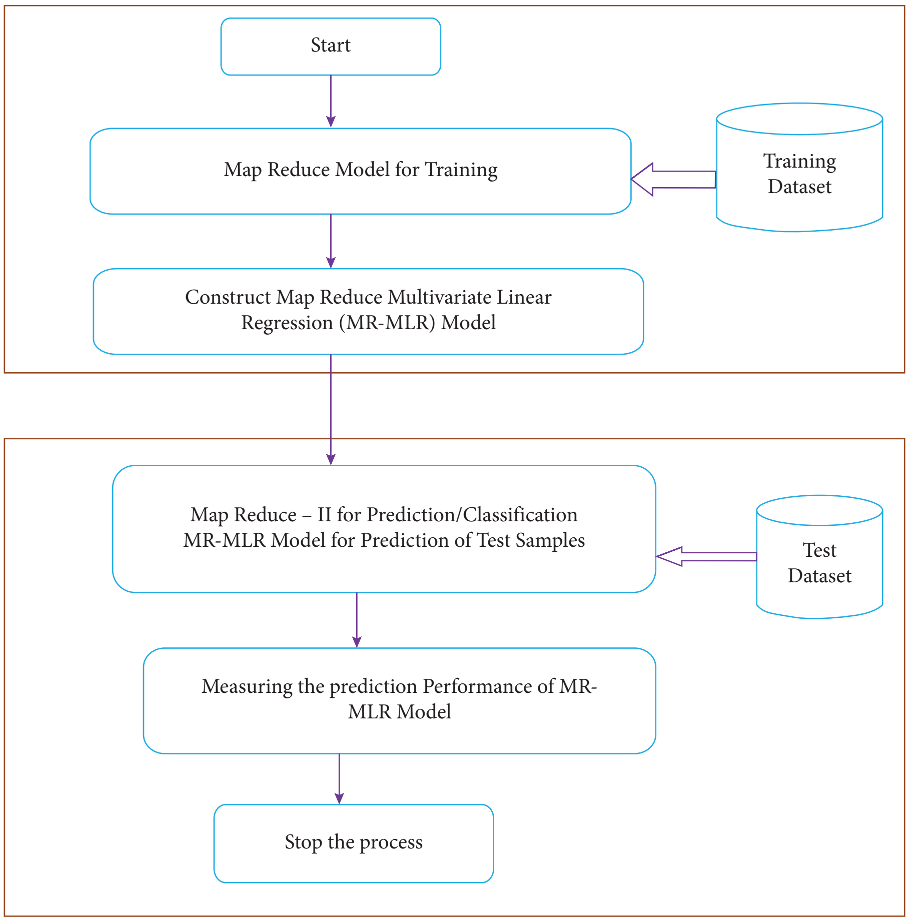

Start
Map Reduce Model for Training
Training Dataset
Construct Map Reduce Multivariate Linear Regression (MR-MLR) Model
Map Reduce – II for Prediction/Classification MR-MLR Model for Prediction of Test Samples
Test Dataset
Measuring the prediction Performance of MR-MLR Model
Stop the process

Figure 1: Flowchart of the proposed MR-MLR algorithm.

from the map tasks and computes the average value of the different performance metrics.

3.3. Hadoop Implementation of MapReduce-I. The instances are distributed uniformly among all the partitions, and each partition contains 'm/s' instances approximately. For each partition, a separate map task is created, and the regression coefficients are computed. Algorithm 1 presents the working flow of the map task operation. The reducer task reads the partially generated results computed by all instances of the map job and generates the average value of all the regression coefficients, as shown in Figure 2. The number of reducer tasks is smaller than the number of map roles in the system, which is due to the fact that only a small portion of the proposed MR-MLR algorithm is implemented in the reducer component. Finally, the reducer constructs a distributed machine learning model and writes the model into HDFS. (Algorithm 2)

3.4. Hadoop Implementation of MapReduce-II. The input test dataset TD with 'z' instances and 'n' columns is partitioned into 's' splits named as $p_1, p_2, \ldots, p_s$ in the underlying HDFS of Hadoop. For every block/partition, a separate map task is created and the machine learning model constructed in the MapReduce-I is validated as given in Algorithm 3. Finally, the reducer computes the average value of all the performance metrics as given in Algorithm 4 and writes the result in HDFS.

4. Experimental Setup

4.1. Datasets Description. In this proposed research work, four datasets from the UCI machine learning repository are used. The characteristics of each dataset are tabulated in Table 1 and shown in Figure 3.

4.2. Performance Measures

4.2.1. Mean Absolute Error (MAE). This shows a direct variation between the preferred and predicted target output values.

$$\text{Mean Absolute Error (MAE)} = \frac{1}{N}\sum_{i=1}^{N}\left|x_i - y_i\right|. \quad (2)$$

4.2.2. Mean Square Error (MSE). This measures the variation between the target of a model and what is going to be predicted pertaining to the target attribute.

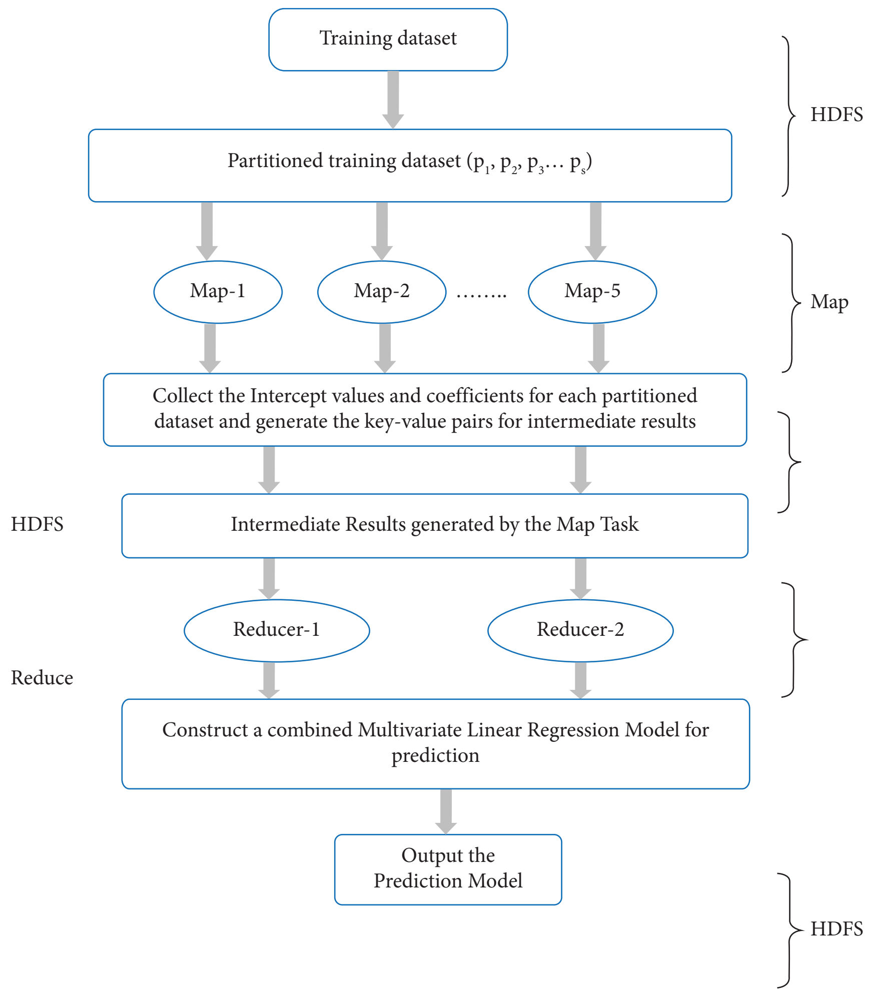

Training dataset
Partitioned training dataset (p1, p2, p3… ps)
HDFS
Map-1
Map-2
……..
Map-5
Map
Collect the Intercept values and coefficients for each partitioned dataset and generate the key-value pairs for intermediate results
HDFS
Intermediate Results generated by the Map Task
Reducer-1
Reducer-2
Reduce
Construct a combined Multivariate Linear Regression Model for prediction
Output the Prediction Model
HDFS

FIGURE 2: Hadoop implementation of MapReduce-I.

```
Function MAP-1 (training dataset)
Begin
    Input: training dataset $D$ with $m$ instances and $n$ attributes
    Partition the dataset $D$ into $s$ partitions as $p1$, $p2$, $p3$. . .. ... ps
    Read x_train[], y_train for each partitioned dataset
        Compute intercept and correlation coefficients for each block of instances
        Convert it into (key, value > pair as < Dataset_id, (intercept, coefficients)>
    Output < Dataset_id, <(intercept, coefficients[])>
end
```

ALGORITHM 1: Map function of MapReduce-I.

$$\text{Mean square error (MSE)} = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - y_i\right)^2. \quad (3)$$

4.2.3. Root Mean Square Error (RMSE). This measures the square root of the quadratic mean of the variations among the predicted and anticipated values of the intention feature.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i - y_i\right)^2}. \quad (4)$$

4.2.4. Determination Coefficient (R^2). This parameter assesses the mode in which the model estimates the actual information, which appraises the predictability grade of the model. The higher R^2, the more efficient the developed model. The value of R^2 is usually between 0 and 1. For example, if $R^2 = 1$ means, the model fits very well, i.e., all the data points lie on the straight line.

$$\text{Determination coefficient}\left(\mathrm{R}^2\right) = 1 - \frac{\sum_{i=1}^{N}\left(x_i - y_i\right)^2}{\sum_{i=1}^{N}\left(x_i - x\right)2}. \quad (5)$$

```
Function REDUCE-1(MAP-I output)
    read < Dataset id, (intercept, coefficients[]> from HDFS
        for $i = 1$ to $s$ partitions
            sum_intercept+ = intercept
        end for
        for $i = 1$ to $s$ partitions
                for $j = 1$ to $n$ attributes
            sum_coefficients[]+ = coefficients[]
                end for
            end for
    compute avg_intercept, avg_coefficients[]
    construct a learned MR-MLR model
    output < MR-MLR model>
end
```

Algorithm 2: Reduce function of MapReduce-I.

```
Function MAP-II (testing dataset)
    Input: testing dataset TD with $k$ instances and $n$ attributes
    Partition the dataset TD into $z$ partitions as $p1$, $p2$, $p3$. . ...$p_s$
    Read x_test[], y_test for each partitioned dataset
        Predict y_predict with the MR-MLR model
        Convert it into (key, value > pair as < Dataset_id, (y_predict, y_test)>
    Output < Dataset_id, (y_predict, y_test)>
End
```

Algorithm 3: Map function of MapReduce-II.

```
Function REDUCE-1I (MAP-II output)
    read < Dataset_id, (y_predict, y_test)>
    for all the $z$ partitions
        compute MSE, RMSE, and determination coefficient ($R^2$) from y_predict and y_test
    find the average value of MSE, RMSE, and determination coefficient ($R^2$)
    output < Dataset id, (MSE, RMSE, and determination coefficient ($R^2$)>
end
```

Algorithm 4: Reduce function of MapReduce-II.

Table 1: Summary description of the used datasets.

S. No.	Dataset	#Attributes	#Data types	#Instances	#File size (MB)	#Year
1	Combined cycle power plant	4	Multivariate	9568	1.93	2014
2	Wave energy converters	49	Multivariate	288000	123	2019
3	Year prediction MSD (subset of million song dataset)	90	Multivariate	515345	433	2011
4	Superconductivity data	81	Multivariate	21263	26.8	2018

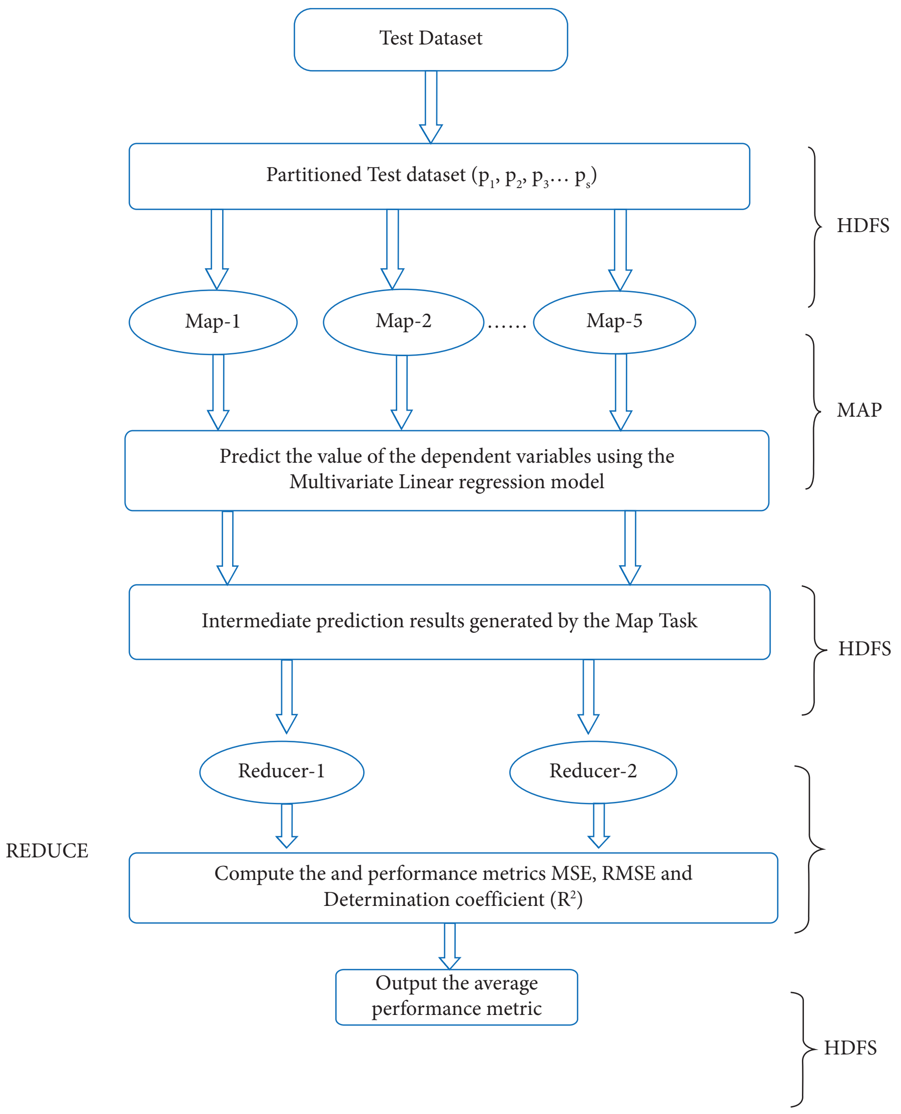
5. Experimental Result Analysis

(i) First, we compare the distributed learning MR-MLR model performance metrics with the standalone sequential learning MLR model as given in Section 5.1

(ii) Second, we analyze the scalability performance of the MR-MLR model, as reported in Section 5.2

(iii) Finally, we check the influence of the split ratio of training and testing samples in the performance metrics of the MR-MLR model, as described in Section 5.3

5.1. Comparison of MR-MLR Model with the Standalone MLR Model. Initially, we run the standalone sequential multivariate linear regression method among all four datasets, which is used as a baseline for comparison. To do this, 10% of the samples are chosen arbitrarily from each dataset and trained in the standalone MLR model. The model is trained with 80% of samples and tested with 20% of samples. The intercept, regression coefficients, and performance metrics are recorded. It is observed that there are four map tasks that have been created for each subset dataset, and all four map tasks learn the correlation among



Test Dataset
Partitioned Test dataset (p_1, p_2, p_3… p_s)
HDFS
Map-1
Map-2
Map-5
MAP
Predict the value of the dependent variables using the Multivariate Linear regression model
Intermediate prediction results generated by the Map Task
HDFS
Reducer-1
Reducer-2
REDUCE
Compute the and performance metrics MSE, RMSE and Determination coefficient (R^2)
Output the average performance metric
HDFS

Figure 3: Hadoop implementation of MapReduce-II.

the attributes and compute the regression coefficients on the given input dataset. Table 2 shows the intercept values generated by the map task process executed in the Hadoop cluster environment. The reduce task process collects all the regression coefficients and computes the average values. The MR-MLR model is constructed with the help of the regression coefficients generated from the Map-Reduce-phase I implementation.

The MapReduce-II implementation validates the model constructed in the earlier phase with 20% of the test dataset samples. The performance is measured with the various parameters and the results are tabulated and compared with the standalone MLR model performance, as shown in Table 3.

From these tables, we can observe the following:

(i) The regression coefficient intercept obtained from all the four map tasks, its average value is nearly the same as the value obtained from the standalone MLR model, which indicates that parallelism on data processing is highly efficient

(ii) There is not much deviation from the baseline values in the performance metrics MAE, RMSE, and the determination coefficient (R^2) of the MR-MLR model when compared with the standalone MLR model

5.2. Scalability. To study the scalability of the proposed MR-MLR algorithm, the influence of dataset split, and distribution on performance metrics, the input file is divided into <s> subsets of the same-sized files with a balanced distribution of instances. We run our experiment on all the four datasets with $s = 6$, 8, 10, 12, 14, and 16. There are about <s> map tasks initiated for each dataset processing, and each map task computes the correlation coefficients.

Table 2: Regression coefficient intercept values ($s = 4$).

Dataset	Intercept values generated by the map tasks				Average intercept value
	Map-1	Map-2	Map-3	Map-4	
Combined cycle power plant	$7.12E-04$	$7.93E-04$	$8.72E-04$	$8.29E-04$	$8.02E-04$
Wave energy converters	$2.45E-04$	$3.15E-04$	$2.75E-04$	$3.30E-04$	$2.91E-04$
Year prediction MSD	$4.15E-04$	$3.92E-04$	$3.85E-04$	$4.13E-04$	$4.01E-04$
Superconductivity data	$1.24E-04$	$2.43E-04$	$1.99E-04$	$2.12E-04$	$1.95E-04$

Table 3: Comparison of performance measures of standalone MLR and MR-MLR ($s = 4$).

Dataset	No. of samples	Intercept		Mean absolute error (MAE)		Root mean square error (RMSE)		Determination coefficient (R^2)	
		MLR	MR-MLR	MLR	MR-MLR	MLR	MR-MLR	MLR	MR-MLR
Combined cycle power plant	957	$7.93E-04$	$8.02E-04$	2.9145	3.3294	4.3289	4.14021	0.90898	0.89805
Wave energy converters	28800	$2.89E-04$	$2.91E-04$	13.1246	12.8326	436.2986	425.9171	0.98129	0.99938
Year prediction MSD	51534	$3.98E-04$	$4.01E-04$	17.6717	18.1748	643.7491	643.7491	0.99212	0.99473
Superconductivity data	2126	$1.88E-04$	$1.95E-04$	10.0976	10.1901	123.5647	124.7320	0.99834	0.99973

Table 4: Regression coefficient intercept (average) and standard deviation (average) values.

Dataset	Average intercept value						Average standard deviation value					
	$s = 6$	$s = 8$	$s = 10$	$s = 12$	$s = 14$	$s = 16$	$s = 6$	$s = 8$	$s = 10$	$s = 12$	$s = 14$	$s = 16$
Combined cycle power plant	$8.06E-04$	$8.28E-04$	$8.09E-04$	$8.13E-04$	$8.30E-04$	$8.00E-04$	4.14	4.76	4.64	4.17	4.52	4.92
Wave energy converters	$3.07E-04$	$3.35E-04$	$3.00E-04$	$3.23E-04$	$3.06E-04$	$3.37E-04$	42.93	43.03	42.23	42.53	42.93	43.33
Year prediction MSD	$4.12E-04$	$4.15E-04$	$4.02E-04$	$4.23E-04$	$4.23E-04$	$4.10E-04$	64.74	65.14	65.44	64.24	63.54	64.44
Superconductivity data	$2.25E-04$	$2.47E-04$	$2.32E-04$	$2.13E-04$	$2.13E-04$	$2.32E-04$	12.89	12.349	13.29	12.34	13.12	13.43

Table 5: Performance metrics: mean absolute error (MAE).

Dataset	Mean absolute error (MAE)					
	$s = 6$	$s = 8$	$s = 10$	$s = 12$	$s = 14$	$s = 16$
Combined cycle power plant	3.64322	4.32675	4.23487	3.89432	4.21264	4.10233
Wave energy converters	13.56213	13.94356	12.98244	13.32156	13.45322	13.76454
Year prediction MSD	20.23987	20.34224	20.52124	20.10224	20.02311	20.56440
Superconductivity data	11.99002	12.54097	12.12105	11.87220	12.10990	12.23154

Table 6: Performance metrics: root mean square error (RMSE).

Dataset	Root mean square error (RMSE)					
	$s = 6$	$s = 8$	$s = 10$	$s = 12$	$s = 14$	$s = 16$
Combined cycle power plant	4.47402	4.86432	4.91286	4.98186	4.67547	4.76547
Wave energy converters	434.45364	456.90675	466.12191	445.12911	439.41254	447.65445
Year prediction MSD	621.23901	676.90870	632.12909	656.23191	643.45324	653.21524
Superconductivity data	125.76329	129.96409	122.12396	131.67596	133.23763	127.36776

Initially, we set $s = 6$ and execute the experiment in the cluster on all four datasets. The intercept values from each map task are converted in to < dataset id, (intercept, regression coefficient)> key pair. The reducers receive the intercept, regression coefficients, and their average values are computed. Afterwards, a learned MR-MLR is constructed with the computed intercept coefficients and standard deviations. Similarly, the experiment is repeated with $s = 8$, 10, 12, 14, and 16 and the results obtained are shown in Table 4.

The MR-MLR model constructed from the average values of intercept and coefficients in MapReduce-I is tested with the test dataset. The performance metrics obtained from the experiments are shown in Tables 5–7.

According to these tables, we can conclude the following:

Table 7: Performance metrics: determination coefficient (R^2).

Dataset	Determination coefficient (R^2)					
	$s=6$	$s=8$	$s=10$	$s=12$	$s=14$	$s=16$
Combined cycle power plant	0.87995	0.86453	0.88615	0.87716	0.88715	0.88718
Wave energy converters	0.99865	0.97213	0.98672	0.98237	0.97817	0.98987
Year prediction MSD	0.98755	0.98213	0.98782	0.98712	0.97828	0.98978
Superconductivity data	0.99754	0.98324	0.99128	0.98783	0.98763	0.98975

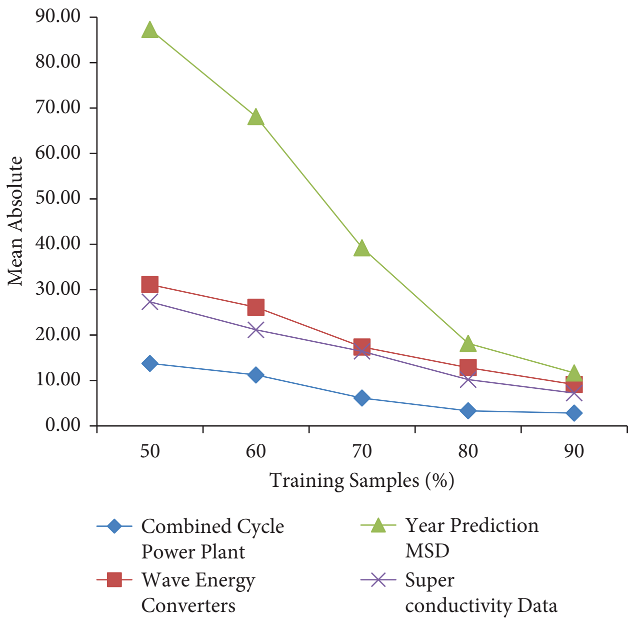


Figure 4: Mean absolute error (MAE) for the different split ratios of samples.

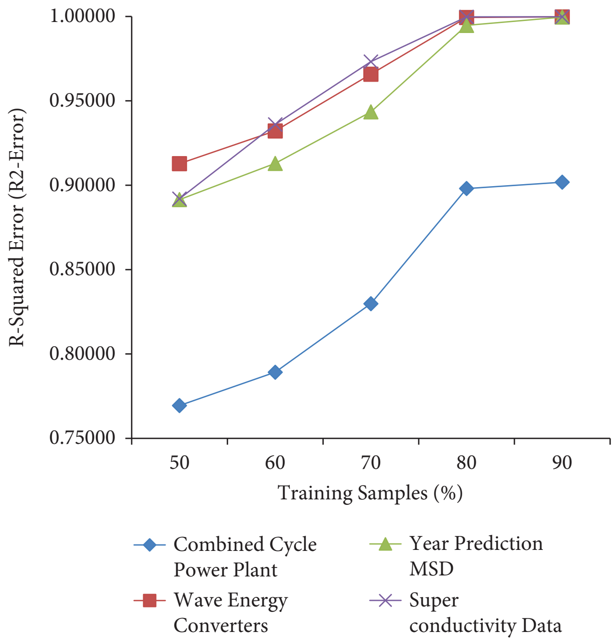


Figure 6: Determination coefficient (R^2) for the different split ratios of samples.

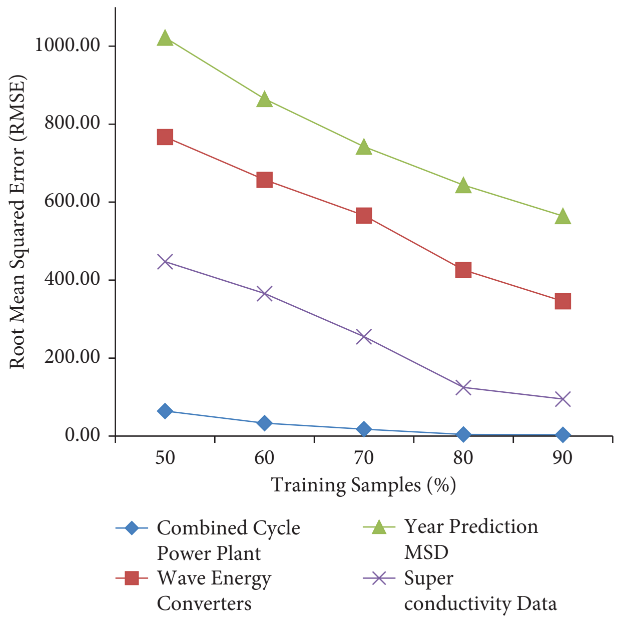


Figure 5: Root mean square error (RMSE) for the different split ratios of samples.

(i) The Hadoop cluster is capable of generating as many map tasks based on the availability of processor cores in the research cluster and the size of datasets

(ii) The MR-MLR model is scalable to handle any number of instances, map tasks, and reduce tasks provided sufficient number of CPU cores in the cluster

(iii) The performance metrics MAE, RMSE, and the determination coefficient ($R2$) of the MR-MLR model vary as 33.3% (from $s=6$ to $s=8$), 25% (from $s=8$ to $s=10$), 20% (from $s=10$ to $s=12$), 16.7% (from $s=12$ to $s=14$), and 12.5% (from $s=14$ to $s=16$) when the $<s>$ value increases for all the four datasets

5.3. Influence of Training and Testing Split Ratio on Performance Metrics. To analyze the influence of the split ratio of training and testing samples on performance metrics, we have chosen the percentage of training samples in the dataset as 50%, 60%, 70%, 80%, and 90%. The effectiveness of the designed MR-MLR model is measured in a 64 MB HDFS configuration, and $<s>$ is set as 4. From Figure 4, it is concluded that the mean absolute error (MAE) gradually decreases when the training sample size increases. In the case of the year prediction MSD dataset, there is a rapid fall in mean absolute error (MAE) as compared to the other three datasets.

Figure 5 shows that the root mean square error (RMSE) is very less in the case of the combined cycle power plant dataset, which is due to the less number of training samples as compared with the other three datasets. As the ratio of training samples increases, R^2 reaches 1, which indicates that the model fits very well with the chosen dataset, as shown in Figure 6.

6. Conclusion

In this article, we have designed and developed a two-stage MapReduce model called MR-MLR for the multivariate linear regression statistical/machine learning based on Apache Hadoop. The constructed model is trained and tested with large-datasets in the multinode Hadoop cluster environment. The use of the Hadoop framework enables us with a scalable, fault tolerance, and runtime management when dealing with a large dataset of millions of instances. The predictive effectiveness of MR-MLR is computed in terms of mean absolute error, root mean square error, and determination coefficient (R^2). The results obtained have shown that the main achievements of MR-MLR are the following:

(i) There is a consistency in terms of the MSE, RMSE, and determination coefficient (R^2) even when the subset of the dataset increases

(ii) When the train and test sample split ratio increases, the determination coefficient (R^2) moves near 1, indicating that the model learns and fits very well to the given dataset

(iii) MR-MLR is a scalable, exact parallel approach as many maps and reducers are instantiated and very good performance metrics achievement in large-sized big data applications.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag, Berlin, Germany, 2007.

[2] R. Bro, "Exploratory study of sugar production using fluorescence spectroscopy and multi-way analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 46, no. 2, pp. 133–147, Mar. 1999.

[3] J. Nilsson, S. de Jong, and A. K. Smilde, "Multiway calibration in 3D QSAR," *Journal of Chemometrics*, vol. 11, no. 6, pp. 511–524, 1997.

[4] N. Draper, H. Smith, and E. Pownell, *Applied Regression Analysis*, Vol. 706, Wiley, New York, 1998.

[5] D. Kleinbaum, L. Kupper, and K. Muller, *Applied Regression Analysis and Other Multivariable Methods*, Duxbury Pr, Florence, KY, 2007.

[6] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "MPCA: multilinear principal component analysis of tensor objects," *IEEE Transactions on Neural Networks*, vol. 19, no. 1, pp. 18–39, Jan, 2008.

[7] A. Shashua and A. Levin, "Linear image coding for regression and classification using the tensor-rank principle," in *Proceedings of the IEEE Comput. Soc.Conf. Comput. Vis. Pattern Recog*, vol. 1, pp. 42–49, Kauai, HI, USA, December 2001.

[8] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, "Incremental tensor analysis: theory and applications," *ACM Transactions on Knowledge Discovery from Data*, vol. 2, no. 3, pp. 1–37, 2008.

[9] J. Yang, D. Zhang, A. F. Frangi, and J.-Y. Yang, "Two-dimensional Pca: a new approach to appearance-based face representation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131–137, Jan, 2004.

[10] J. Ye, "Generalized low rank approximations of matrices," in *Proceedings of the Twenty-First International Conference on Machine Learning*, pp. 887–894, Banff, AB, Canada, July 2004.

[11] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[12] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pp. 29–43, ACM, Bolton Landing, NY, USA, June 2003.

[13] P. Mika and G. Tummarello, "Web semantics in the clouds," *IEEE Intelligent Systems*, vol. 23, no. 5, pp. 82–87, 2008.

[14] Apache Hadoop, "Apache Hadoop," 2014, http://hadoop.apache.org/ Accessed.

[15] Mahout, "Mahout," 2014, http://mahout.apache.org/ Accessed.

[16] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "PigLatin: a not-so-foreign language for data processing," in *Proceedings of the ACM SIGMOD International Conference of Management of Data*, Vancouver, Canada, June 2008.

[17] J. Urbani, S. Kotoulas, J. Maassen, F. Van Harmelen, and H. Bal, "WebPIE: a web-scale parallel inference engine using mapreduce," *Journal of Web Semantics*, vol. 10, pp. 59–75, 2012.

[18] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[19] L. L. Ding, J. C. Xin, G. R. Wang, and S. Huang, "Efficient skyline query processing of massive data based on MapReduce," *Chinese Journal of Computers*, vol. 34, no. 10, pp. 1785–1796, 2011.

[20] Y. Gaizhen, "The application of MapReduce in the cloud computing," in *Proceedings of the Intelligence Information Processing and Trusted Computing (IPTC)*, pp. 154–156, Wuhan, China, October 2011.

[21] B. Catanzaro, N. Sundaram, and K. Keutzer, "A map reduce framework for programming gpus," in *Proceedings of the Third Workshop Software Tools for MultiCore Systems, STMCS*, Berkeley, California, 2008.

[22] K. Hwang and W. Sung, "Load balanced resampling for real-time particle filtering on graphics processing units," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 411–419, 2013.

[23] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox, "MapReduce in the clouds for science," in *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science, (CloudCom)*, pp. 565–572, IEEE, Indianapolis, IN, USA, November 2010.

[24] Y. Wang, W. Goh, L. Wong, and G. Montana, "Random forests on Hadoop for genome-wide association studies of multivariate neuroimaging phenotypes," *BMC Bioinformatics*, vol. 14, no. S16, p. S6, 2013.

[25] A. Anjum, T. Abdullah, M. F. Tariq, Y. Baltaci, and N. Antonopoulos, "Video stream analysis in clouds: an object detection and classification framework for high performance video analytics," *IEEE Trans. Cloud Comput*, vol. 7, no. 4, pp. 1152–1167, 2019.

[26] M. U. Yaseen, A. Anjum, O. Rana, and R. Hill, "Cloud-based scalable object detection and classification in video streams," *Future Generation Computer Systems*, vol. 80, pp. 286–298, 2018.

[27] S. Arsh, A. Bhatt, and P. Kumar, "Distributed image processing using hadoop and HIPI," in *Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016*, pp. 2673–2676, Jaipur, India, September 2016.

[28] C. Sweeney, L. Liu, S. Arietta, and J. Lawrence, *HIPI: A Hadoop Image Processing Interface for Image-Based Mapreduce Tasks*, University of Virginia, Charlottesville, VA, USA, 2011.

[29] W. Jatrniko, D. M. S. Arsa, H. Wisesa, G. Jati, and M. A. Ma'sum, "A review of big data analytics in the biomedical field," in *Proceedings of the International Workshop on Big Data and Information Security (IWBIS)*, pp. 31–41, IEEE, Jakarta, Indonesia, 2016.

[30] W. Huang, L. Meng, D. Zhang, and W. Zhang, "In-memory parallel processing of massive remotely sensed data using an Apache Spark on Hadoop YARN model," *Ieee Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 1, pp. 3–19, 2017.

[31] J. Maillo, I. Triguero, and F. Herrera, "A MapReduce-based k-nearest neighbor approach for big data classification," in *Proceedings of the 9th International Conference on Big Data Science and Engineering (IEEE BigDataSE-15)*, pp. 167–172, Helsinki, Finland, August 2015.