Map-Reduce based Distance Weighted k-Nearest Neighbor Machine Learning Algorithm for Big Data Applications

Gothai E
Kongu Engineering College
Usha Moorthy
REVA University
Sathishkumar V E (sathishkumar@scnu.ac.kr)
Sunchon National University https://orcid.org/0000-0002-8271-2022
Abeer Ali Alnuaim
King Saud University
Wesam Atef Hatamleh
King Saud University
Dirar Sweidan
University of Skovde: Hogskolan i Skovde

Research Article

Keywords: Supervised Learning, Big Data, MapReduce framework, Distance weighted k-NN, Determination Coefficient (R2), Accuracy, Scalability, Speedup.

Posted Date: July 9th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-684319/v1

Map-Reduce based Distance Weighted k-Nearest Neighbor Machine learning algorithm for Big Data Applications

E. Gothai[1], Usha Moorthy[2], Sathishkumar V E[3], Abeer Ali Alnuaim[4], Wesam Atef Hatamleh[5], Dirar Sweidan[6],

[1]Department of Computer Science and Engineering, Kongu Engineering College, Perundurai, Erode 638060, Tamilnadu, India

[2]School of Computer Science and Engineering, REVA University, Banglore, India

[3]Department of Information and Communication Engineering, Sunchon National University, Suncheon, Republic of Korea.

[4]Department of Computer Science and Engineering, College of Applied Studies and Community Services, King Saud University, Saudi Arabia.

[5]Department of computer science, College of Computer and Information Sciences, King Saud University, Saudi Arabia.

[6]School of Informatics, University of Skövde, Högskolevägen 1, 541 28 Skövde, Sweden.

Corresponding Email: sathishkumar@scnu.ac.kr

***Abstract**:* With the evolution of Internet standards and advancements in various Internet and mobile technologies, especially since web 4.0, more and more web and mobile applications emerge such as e-commerce, social networks, online gaming applications and Internet of Things based applications. Due to the deployment and concurrent access of these applications on the Internet and mobile devices, the amount of data and the kind of data generated increases exponentially and the new era of Big Data has come into existence. Presently available data structures and data analyzing algorithms are not capable to handle such Big Data. Hence, there is a need for scalable, flexible, parallel and intelligent data analyzing algorithms to handle and analyze the complex massive data. In this article, we have proposed a novel distributed supervised machine learning algorithm based on the MapReduce programming model and Distance Weighted k-Nearest Neighbor algorithm called MR-DWkNN to process and analyze the Big Data in the Hadoop cluster environment. The proposed distributed algorithm is based on supervised learning performs both regression tasks as well as classification tasks on large-volume of Big Data applications. Three performance metrics, such as Root Mean Squared Error (RMSE), Determination coefficient (R^2) for regression task, and Accuracy for classification tasks are utilized for the performance measure of the proposed MR-DWkNN algorithm. The extensive experimental results shows that there is an average increase of 3% to 4.5% prediction and classification performances as compared to standard distributed k-NN algorithm and a considerable decrease of Root Mean Squared Error (RMSE) with good parallelism characteristics of scalability and speedup thus, proves its effectiveness in Big Data predictive and classification applications.

Keywords: Supervised Learning, Big Data, MapReduce framework, Distance weighted k-NN, Determination Coefficient (R^2), Accuracy, Scalability, Speedup.

1. Introduction

The k-Nearest Neighbors (kNN) algorithm [1, 2] is one of the top 10 supervised machine learning algorithms that perform classification as well as regression analysis on the given dataset. Due to its non-parametric nature, easy implementation, and effectiveness, it becomes an important part of the machine learning domain and found in various diverse fields like pattern classification [3], image classification [4,5], big data classification[6], automated world wide web usage mining [7] and document classification[8-12].

There are several variants of the kNN algorithm have been proposed to handle various kinds of data and are shown to be very effective in its performance. Keller et al. [13] proposed a modified version of the kNN algorithm based on fuzzy concepts called fuzzy-kNN with three techniques to assign fuzzy memberships to the data points. Denoeux [14] proposed a method D-SkNN based on Dempster-Shafer theory to address the problem of unseen pattern classification in a dataset based on the kNN algorithm. The author demonstrated the performance of the new method D-SkNN with a real-time dataset as well as a simulated dataset and compared it with the standard kNN and majority voting methods. Kuncheva [15] specified an intuitionistic fuzzy version of the kNN rule called IF-kNN and incorporated the voting rule with assigned weights based on the membership and non-membership to a certain category of class. Based on the threshold value, the vote is categorized as positive or negative. Yang et al. [16] developed an enhanced version of the kNN algorithm with a fuzzy editing rule and incorporated a few asymptotic properties into kNN. The experimental results conducted on various datasets conformed that this approach outperforms the standard kNN algorithm. Huang et al. [17] developed a modified version of kNN called DCT-kNN which is based on feature weighting and class distribution. The experiments on UCI datasets had shown a considerable classification accuracy improvement.

Liu et al. [18] introduced the kNN method for Multi-class classification problems and demonstrated its performance in the classification of Multiclass datasets. Liu and Zhang [19] designed the variant of the kNN algorithm termed as mutual nearest neighbors (MkNN) to remove the noisy data and improved the data classification accuracy. Zhang [20] incorporated the certainty factor measure to the kNN algorithm to apply it over the imbalanced class distribution dataset and the variant is called kNN-CF. The authors also demonstrated that kNN-CF algorithm accuracy is better than the standard kNN algorithm. Shichao et al. [21] proposed a novel method called self-reconstruction to determine the k-value of the kNN algorithm for each training sample and applied on real datasets for data classification. The experimental results show that this method outperformed the standard data classification methods in terms of classification accuracy.

In recent years, the map-reduce based distributed and parallel machine learning algorithms are the focus of the research community. The MapReduce framework has emerged as a powerful, robust, and distributed parallel programming model [22-25] provides a solution with good performance and efficient execution to large-scale data analytic applications including data mining, web page access ranking, graph analysis, image classification and bioinformatics [26-44]. Kolb et al. [45] investigated the use of the MapReduce programming model for parallel entity resolution with automated data partitioning on real-world datasets.

The application of the kNN algorithm and its variants in the context of big data for classification and regression has been already considered. Triguero et al. [46] suggested a method for handling big data in kNN classification. They proposed a novel partitioning method based on the Map-Reduce framework and distributed the functioning of algorithms to multiple nodes in the cluster environment without any loss of classification accuracy. The performance of this method was tested with 5.7 million instances on Poker hand dataset, obtained an accuracy of 0.5171 for k=3 and the results show that kNN is a suitable algorithm to handle big data.

Ding et al. [47] proposed a clustering-based approach for processing large high-dimensional datasets. The authors added the Principal Component Analysis for dimensionality reduction in addition to the kNN classification algorithm in the processing of large datasets. Deng et al. [48] introduced the kNN algorithm in big data applications for classifications. The authors applied the k-means clustering algorithm on a large size dataset, as a result, it is split into several subsets. Finally, they applied the kNN method, its variants RC-kNN and LC-kNN, classified the samples in each subset of the dataset with an accuracy of 72.21%, 83.89% and 86.35% on MNIST dataset. In [49] the authors propose a new distributed and parallel kNN join operation on large real and synthetic datasets in the Map-Reduce platform. They demonstrated the scalability and efficiency of the method with hundreds of millions of records. In [50], the authors developed a cost-effective MapReduce-based k-Nearest Neighbor (MR-kNN) algorithm for Big Data classification and tested for different k-values against the Pokerhand dataset. The classification accuracy was about 0.5386 for k=7. In [51], an iterative Hadoop MapReduce method called iHMR-kNN was developed for kNN based classification and analysis of the image dataset.

The contributions of this paper include:
- A distributed and parallel Distance Weighted k-Nearest Neighbor model has been proposed to analyze big data
- Hadoop MapReduce framework cluster has been established to improve execution efficiency and scalability of the proposed system
- Different block size of data in the underlying HDFS chosen to handle large datasets with high efficiency rate.
- Several experiments have been designed and executed to show the speedup and scale-up of the proposed distributed algorithm

- Classification accuracy of the proposed system measured and analyzed for different size of k-Nearest neighbors
- The performance of the proposed system is evaluated on the authentic, standard and reliable dataset.

The rest of this paper is organized as follows. Section two provides an introduction to the distance weighted k-NN algorithm, we discussed the big data technology Hadoop framework and the architecture of the proposed MR-DWkNN model with the Hadoop implementation algorithm in section three. The experimental setup such as Hadoop computational cluster and its components, various benchmark datasets, and performance metrics for evaluation are described in section four. The conduct of experiments, performance analyses, and results in comparison with standard k-NN are presented in section five while section six draws conclusions from the experimental study and suggests the directions for future work.

2. Distance Weighted k-Nearest Neighbor (DWkNN) Algorithm

k-Nearest Neighbor algorithm is a lazy learner and nonparametric method which does not rely on building a model during the training phase, and whose classification rule is based on a given similarity function between the training samples and the query (testing) sample to be classified. Unlike the existing model-based classification algorithms (building a model with a given training dataset and predicting any test samples with the built model), the kNN algorithm needs to keep all the training examples in memory, to search for all the K nearest neighbors for a test sample.

Let $TR = \{x_1, \dots, x_n\}$ be the training dataset with n instances and m attributes. All the instances are class labeled data points(x_i, c_j), $j = 1, \dots, n_c$ the corresponding class labels of the instances. Here $n > n_c$. In the k-NN algorithm, the learned target function may be either categorical (discrete-valued) or continuous-valued (real-valued) function. Given a query instance x_q, from the collections of query instances dataset TQ, its unknown class c' is determined as follows.

Step 1: Compute the Euclidean distance between the query instances x_q to all instances in TR. Let the instance x_i in TR is described as a feature vector $\left(a_1(x_i), a_2(x_i), \dots, a_m(x_i)\right)$ where $a_z(x_i)$ denotes the z^{th} attribute value of instance x_i.
Let $d(x_i, x_q)$ is the Euclidean distance between the query instance x_q and the instance x_i in the dataset TR and is computed as per equation (1)

$$d(x_i, x_q) = \sqrt{\sum_{z=1}^{m} \cdot \left(a_z(x_i) - a_z(x_q)\right)^2} \qquad (1)$$

Step 2: Sort the instances in the dataset TR in ascending order of their Euclidean distances as given in equation (2).

$$sort\ (TR < (d(x_i, x_q) >),\ 1 \leq i \leq n \tag{2}$$

Step 3(a): For discrete-valued learning target function,

The general form of a discrete-valued learning function is $f = TR^m -> V$, where V is the finite set $\{v_1, \ldots, v_s\}$, here s is the number of classes.

Let $x_1, \ldots, x_k$ denote the k instances from TR that are nearest to the query instance x_q

Return the class label for the query instance x_q as given in equation (3)

$$f'(x_q) \leftarrow \sum_{i=1}^{k} \delta(v, f(x_q)) \tag{3}$$

Where $\delta\ (x, y) = 1$ if $x = y$ and where $\delta\ (x, y) = 0$ otherwise

Step 3(b) : For real-valued target function,

The general form of the function is $f : TR^m -> TR$

The target attribute value for the query instance x_q is computed as given in equation (4)

$$f'(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k} \tag{4}$$

The above steps illustrate the standard kNN algorithm and this can be enhanced with the inclusion of weight function as given in equation (5),

Step 4(a): Select the k-nearest neighbors from the dataset TR

Let $TR' = \{ x_1, \ldots, x_k\}$ be the k nearest instances. Assign a weight w_i to i^{th} nearest neighbor of the query x_q using the distance-weighted function as given in equation (5)

$$w_i = \begin{cases} \dfrac{d(x_q, x_k) - d(x_q, x_i)}{d(x_q, x_k) - d(x_q, x_1)}, if\ \ d(x_q, x_k) \neq d(x_q, x_1) \\ 1 \qquad\qquad\qquad\qquad , if\ \ d(x_q, x_k) \neq d(x_q, x_1) \end{cases} , \tag{5}$$

Based on the majority voting, assign the class label c_j to the query instance x_q of discrete-valued function as given in equation (6)

$$f'^{(x_q)} \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i(v, f(x_q)) \tag{6}$$

Step 4(b): For real-valued target functions, the target attribute value for the query instance x_q is computed as given in equation (6)

$$f'(x_q) = \frac{\sum_{i=1}^{k} w_i \, f(x_i)}{\sum_{i=1}^{k} w_i} \tag{7}$$

3. Hadoop Implementation of our proposed Algorithm

3.1 Hadoop Distributed Framework

Hadoop is an Apache project, initially introduced in 2007 as a free and open-source framework with two major components Hadoop Distributed File System (HDFS) and MapReduce (MR) programming model. It becomes very popular in recent years because of its simplicity, built-in fault tolerance, scalability on data-intensive tasks, parallel execution of tasks, and capable of running on a Hadoop cluster with 1000s of commodity hardware machines. Hadoop framework itself takes care of job scheduling, job execution, controlling of all underlying tasks in the cluster, and other runtime management tasks.

MapReduce is a software framework that enables the application developers to explore all the options of parallel procedures with Map and Reduce functions. The MapReduce programming model supports the developers to write and execute the applications upon a cluster consists of a few hundred to several thousand commodity hardware machines. The three major classes of the MapReduce program are Master/Driver, Mapper, and Reducer. The Master class is responsible for setting various execution parameters to the MapReduce job to run in the Hadoop cluster. The parameters are names of Mapper and Reducer classes, data types, and job names to be executed. The MapReduce framework operates on (key, value) pairs. Each Map task process an input split (block) generating intermediate data of (key, value) format. Then, they are sorted and partitioned by key, so later at the Reduce phase, pairs of the same key will be aggregated to the same reducer for further processing [52-53].

3.2 MapReduce Architecture of proposed Distance Weighted kNN Algorithm

The main purpose of the supervised machine learning technique is to facilitate an algorithm to learn from the previous historical data/events and extracts the knowledge from the events. The extracted knowledge is represented as an intelligent mathematical model that can be used to make predictions or classifications on the given new scenario/future events. In most general terms, the machine learning model consists of two phases, the training phase, and the testing phase. Fig.1 depicts the Map-Reduce architecture of the proposed Distance weighted kNN algorithm and its implementation.

The input training dataset TR with ‘m’ samples and ‘n’ features is split into ‘s’ partitions/blocks as p_1, p_2,.., p_s in the distributed file system of the Hadoop cluster. Each partition

takes one block storage size of HDFS (64MB/128MB/256MB) as set initially in the Hadoop file system. Each partition contains 'm/s' samples, distributed uniformly with a default replication factor of 3. The dataset TQ contains the query instances and since the K-Nearest Neighbor is a lazy learner, the pattern matching process initiated only on the submission of query instance x_q to the system. For each block of the input file, a separate map task is created and each map task computes the Euclidean distance between the query instance x_q and all the instances in the block. The Euclidean distance that is computed for all the blocks is converted into <key, value> pair as <instance_id, (Euclidean distance, attributes)> and these are stored in HDFS as intermediate results. Algorithm.1 provides the details of the Map function operation of Distance weighted kNN.

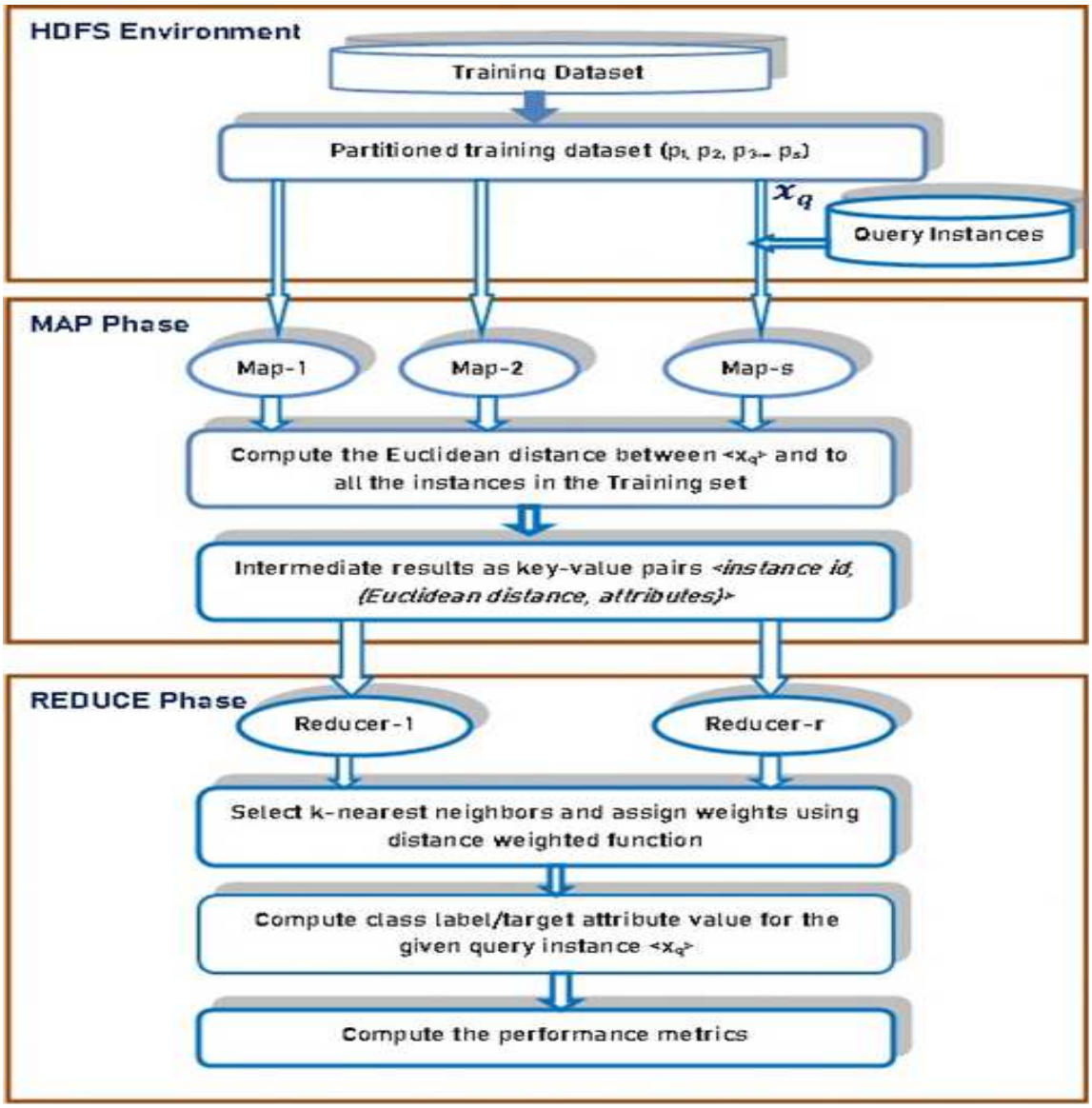

Figure.1: Proposed Distance Weighted kNN Algorithm (MR-DWkNN)

The reducer part of the MapReduce program collects the intermediate results generated by all the map tasks and sorts the instances on Euclidean distance from smallest to largest. After sorting instances, k-nearest samples are selected and weights are assigned to the samples based on Euclidean distance using the distance weighted function as in equation (5). Finally, the reducer finds the class label for the query instance x_q if the target attribute is a discrete-valued function as given in equation (6) and computes the target attribute value for the real-valued target function as given in equation (7). The detailed operation of the Reduce function is given in algorithm.2. The Map-reduce architecture of standard k-NN is the same as Fig.1 with the exclusion of distance weighted function as given in equation (5) and is called an MR-SDkNN algorithm.

Algorithm 1 Map function of Distance Weighted kNN

Procedure DWkNN-MAP (TR, TQ, k))

Input: TR-> Training dataset with n instances and m attributes
TQ-> Dataset with query instances
k-> number of neighbors
key-> instance_id,
value -> <Euclidean instance, instance features>

Output: Class label/target attribute value for the query instance x_q *and performance metrics of the proposed methodology.*

for a given query instance x_q *in* TQ

for t=1 to size(TR) do

$ED(x_q, x_i) < -compute\ Euclidean\ distance\ (TR(x_i), x_q\ )$

$result < -(< key : instance_{id(t)}; value: (ED(x_q, x_i), instance) >$

$context.write(t, ED(x_q, x_i))$

end for

end procedure

Algorithm 2 Reduce function of Distance Weighted kNN

Procedure DWkNN-REDUCE (key, value)

Key: instance_id

Value: Euclidean distance, Attributes

For all k-nearest instances of TR do the following

Compute w_i *as given in equation (5)*

Assign the distance-weighted value to all k-instances

If the target function is discrete-valued then

Find the class-label of x_q using equation (6)

Compute the performance metric classification Accuracy

Context.write(x_q, <class label, accuracy>)

else

Find the target attribute value of x_q using equation (7)

Compute the performance metrics RMSE and Determination coefficient (R^2)

Context.write(x_q, (class label, RMSE, Determination coefficient (R^2)))

endif

end for

4 Experimental Setup

This section describes the Multi-node Hadoop cluster configured for this research work (Sec 4.1), specifications of the datasets chosen (Sec 4.2), and the various performance metrics employed to measure the classification and regression task performances. (Sec 4.3).

4.1 Hadoop cluster environment.

The various experiments designed for this proposed research work are executed on a Multi-node Hadoop cluster established with 25 physical machines. One machine is designated as a master node (Name node) and configured to run Hadoop services and the remaining machines are configured as the worker nodes (data node). The configuration of all the physical machines is Core i5 four-core Processor, 2.1GHz clock speed, 16 GB RAM, 6MB Cache, 1 TB HDD with 1 Gbps network card.

The specific details of the software used in the cluster and Hadoop environment configuration parameters are the following:

Hadoop framework version 2.9.0
Operating system: Ubuntu Linux 18.04.03 LTS 64-bit
Replication factor : 3
HDFS size : 64 MB/128MB
Virtual memory for the map and reduce task: 8 GB

The total cores in the cluster are 50, enabling the hardware level hyper-threading features, the cores in the cluster become 100.

4.2 Datasets

In this research work, we will use six benchmark large-size datasets from the UCI machine learning repository. Among the six datasets, three datasets with real-valued target attributes used for the regression task (Table.1), and the remaining three datasets with discrete-valued target attributes are used for the classification task (Table.2). We tabulate the number of attributes (#attributes), Data type of attributes (#Data types), number of instances (#instances), file size (#File size), and the year of publication (#year). In our experimental work, all the datasets are partitioned into training and test dataset using an n^{th} fold cross-validation technique.

Table: 1 Summary description of the datasets with the real-valued target attributes

S.No	Dataset	#Attributes	#Data Types	# instances	#File size	#year
1	Superconductivity Data	81	Multivariate	21,263	26.8MB	2018
2	Year Prediction MSD (Subset of Million song Dataset)	90	Multivariate	515,345	433MB	2011
3	Wave Energy Converters	49	Multivariate	288,000	123MB	2019

Table: 2 Summary description of the datasets with discrete-valued target attributes

S.No	Dataset	#Attributes	#Data Types	# instances	#File size	#classes	#year
1	Higgs	28	Multivariate	11,000,000	7.4GB	2	2014
2	Susy	18	Multivariate	5,000,000	2.22GB	2	2014
3	Pokerhand	11	Multivariate	1,025,000	23.4 MB	10	2007

4.3 Performance measures

In our research work, the performance of the proposed MR-DWkNN algorithm is assessed with the following four metrics.

Root Mean Square Error (RMSE): The error value of the regression model is measured in terms of RMSE. This metric shows the square root of the quadratic mean of the differences between the predicted and expected values of the target attribute.

$$\text{Root Mean Square Error (RMSE)} = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \quad (8)$$

Determination coefficient (R^2): This metric analyzes how the differences in one variable (x) can be explained by a difference in another variable (y). This measure evaluates how a model approximates the real data points. The higher R^2, the more efficient is the prediction model and its value usually between 0 and 1.

$$\text{Determination coefficient } (R^2) = 1 - \frac{\sum_{i=1}^{N}(x_i - y_i)^2}{\sum_{i=1}^{N}(x_i - x)2} \quad (9)$$

Accuracy: The performance of classification model is measured using the metric accuracy. It is defined as the ratio of the number of correctly predicted samples to the total number of input samples in the dataset.

$$Accuracy = \frac{Number\ of\ correctly\ predicted\ samples}{Total\ Number\ of\ input\ samples} \quad (10)$$

These are three commonly used performance metric for measuring the performance of the standard predictors/classifiers

Scalability: This defines the capability of an algorithm in a parallel computing cluster environment to enhance both in terms of the number of processing cores/nodes in the cluster and the number of instances in the dataset. It can also be defined as the capability of an s-times larger computing system in the cluster environment to execute an s-times larger computational task in the same given job execution time as the original system and this can be expressed as

$$Scalability(s, TD) = \frac{JET(1,TD)}{JET(s,sTD)} \quad (11)$$

where‘s’ is the number of cores/nodes in the cluster environment, JET(1, TD) is the job execution time on one core/node with data size of TD, JET(s, sTD) is the job execution time of the parallel tasks with ‘s’ cores/nodes in the cluster environment with data size s times of TD. Ideal parallelism shows a constant scale up with an increasing number of computing cores in the cluster and dataset size [54-55]

Speedup: Speedup is also one of the measures employed in evaluating the performance of the parallel algorithms and is used to enhance the job execution time in the cluster. It is defined as the ratio of the time of sequential execution to the time of parallel execution. Speedup can be expressed as

$$Speedup(s, TD) = \frac{JET(1)}{JET(s)} \quad (12)$$

where‘s’ is the number of cores in the cluster environment, JET(1) is the job execution time on one core with data size of TD, JET(s) is the job execution time of the parallel tasks with ‘s’ cores in the cluster environment with the same data size TD.

5 Conduct of Experiments and discussion of results

In this section, we evaluate the proposed supervised MR-DWkNN algorithm on six public datasets in the UCI Machine Learning repository. Here we described the four experiments conducted and compare the results collected from these experiments with the MR-SDkNN algorithm.

- In the first two experiments, the MR-DWkNN method was executed against the three datasets under the regression category and another three datasets under the classification category. The performance metrics are compared with the standard MR-SDkNN method as given in section 5.1 and 5.2 respectively.
- Third, the scalability performance of the MR-DWkNN model on regression and classification datasets are analyzed, reported in section 5.3
- Finally, the performance of our proposed method for different k-values was tested and the performance metrics are described in section 5.4.

5.1 Performance of MR-DWkNN model with MR-SDkNN model

In this experiment, the HDFS block size is set as 64MB, and 10% instances from the Higgs dataset, 20% from Susy dataset, and the entire Pokerhand dataset are chosen. Initially, we run the parallel version of the standard kNN algorithm MR-SDkNN over all the six datasets which is used for comparison with its variants. To do this, 20% of the instances are chosen randomly from each dataset as query instances (TQ) and the remaining 80% of the instances are considered as Training instances (TR). The performance metrics root mean square error (RMSE) and Determination coefficient (R^2) are recorded for the regression task and accuracy is recorded for the classification task.

Afterward, we executed the MR-DWkNN algorithm over all the three datasets under the classification category. In this, the entire input dataset file is split into HDFS block size of equal-sized files and contains an equal number of samples. The samples in each file have been distributed evenly and studied the performance of the proposed MR-DWkNN algorithm. Initially, the Map-Reduce model is trained with 80% of the training samples. Table.3 shows the classification accuracy of MR-SDkNN and MR-DWkNN on three benchmark datasets for two different k-values. Similarly, the regression performance of MR-SDkNN and MR-DWkNN is tabulated in Table.4

Table 3: Performance metrics of the Classification task

Dataset	No. of Instances	No. of Blocks/Map Tasks	k=3		k=5	
			MR-SDkNN	MR-DWkNN	MR-SDkNN	MR-DWkNN
Higgs	1,100,000	12/12	0.67382	**0.69782**	0.68688	**0.71065**
Susy	1,000,000	8/8	0.76361	**0.79416**	0.78754	**0.80965**
Pokerhand	1,025,000	1/1	0.54784	**0.55329**	0.55064	**0.56548**

Table 4: Performance metrics of a Regression task

Dataset	No. of Blocks / Map Tasks	MR-SDkNN (k=3)		MR-DWkNN (k=3)		MR-SDkNN (k=5)		MR-DWkNN (k=5)	
		RMSE	Det.Coe (R^2)	RMSE	Det.Coe (R^2)	RMSE	Det.Coe (R^2)	RMSE	Det.Coe (R^2)
Superconductivity Data	1/1	63.65784	0.89327	**52.63257**	**0.90676**	55.75436	0.89878	**52.63257**	**0.91564**
Year Prediction MSD	7/7	364.73621	0.86213	**325.63850**	**0.88126**	343.65785	0.86899	**307.64545**	**0.89675**
Wave Energy Converters	2/2	245.71917	0.90234	**232.67719**	**0.91452**	241.43650	0.90768	**212.65432**	**0.92456**

From the above two tables, we can conclude that,

- In the case of the classification task, the proposed MR-DWkNN algorithm produces an increase of classification accuracy in the range of 1.5% to 3.5% for all three datasets as compared with the standard MR-SDkNN.
- In the case of regression task, the MR-DWkNN algorithm produces an increase of Determination coefficient (R^2) in the range of 1.5% to 3.2%, and consequently, there is a fall of RMSE in the range of 1.5% to 2.5% for all three datasets as compared with the standard MR-SDkNN.

5.2 Scalability, Dataset split and distribution

To demonstrate how well both the MR-SDkNN and MR-DWkNN scales up, two datasets under each category were chosen. The scalability experiments were performed where the instances of the dataset/size of the dataset were increased in proportion to the number of cores. Table.5 summarizes the datasets used for the scalability experiment, number of instances, computing processor cores, the number of map tasks, and the scalability results obtained for both MR-SDkNN and MR-DWkNN algorithms.

Table.5 Scalability performance of MR-SDkNN and MR-DWkNN

Dataset	#instances	#cores used	#Map Tasks	Scalability	
				MR-SDkNN	MR-DWkNN
Higgs	1,100,000	12	12	1.00000	1.00000
	2,200,000	24	24	0.98650	0.97964
	4,400,000	48	48	0.93256	0.92546
	8,800,000	96	96	0.89756	0.87659
Poker Hand	1,025,000	1	1	1.00000	1.00000
	2,050,000	2	2	0.98082	0.96702
	4,100,000	4	4	0.97125	0.94576
	8,200,000	8	8	0.94366	0.91342
	16,400,000	16	16	0.92453	0.87905
	32,800,000	32	32	0.89786	0.84704
Year Prediction MSD	515,345	7	7	1.00000	1.00000
	1,030,690	14	14	0.97908	0.95409

	2,061,380	28	28	0.96880	0.94378
	4,122,760	56	56	0.94788	0.91876
Wave Energy	288,000	1	1	1.00000	1.00000
Converters	576,000	2	2	0.98761	0.97699
	1,152,000	4	4	0.96034	0.95578
	2,304,000	8	8	0.95979	0.93456
	4,608,000	16	16	0.93435	0.90671
	9,216,000	32	32	0.87692	0.85674

From the Table.5, It is observed that the scalability of both MR-SDkNN and MR-DWkNN decreases slowly when the size of the dataset increases, and the number of processor cores used for the computation increases.

Fig.2 shows the performance results of our proposed model on Higgs datasets. The experiment was conducted with a size of 1.1 million, 2.2 million, 4.4 million, and million 8.8 million instances on 12, 24, 48, and 96 cores respectively. Since the initial size of the dataset with 1.1 million instances is 740 MB, it occupies about 12 HDFS blocks, hence 12 cores have been chosen initially for its execution. Afterward, the size of the cluster has been doubled. It is observed that the scalability of both MR-SDkNN and MR-DWkNN decreases slowly when the size of the dataset increases and it maintains a value of scale-up higher than 89.76% for MR-SDkNN and 87.65% for MR-DWkNN.

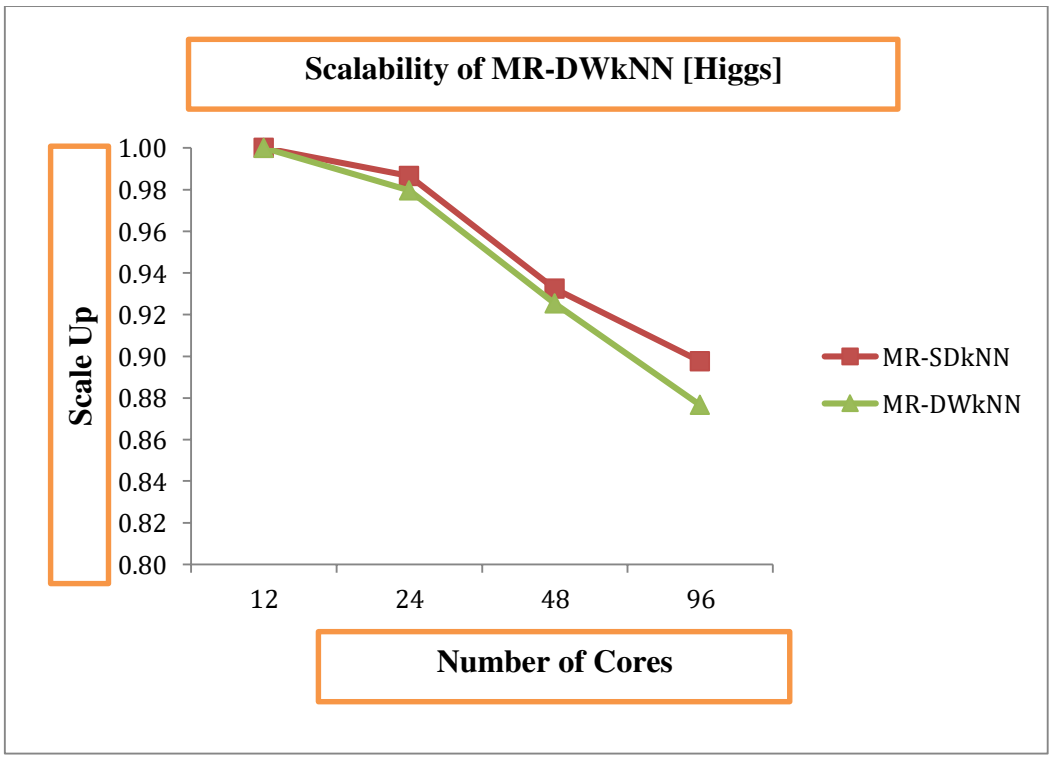

Fig.2 Scalability of MR-SDkNN and MR-DWkNN [Higgs Dataset]

The scalability performance of MR-SDkNN and MR-DWkNN algorithms on the Poker Hand dataset is shown in Fig.3. The initial size of the dataset is 23.4 MB with 1.025 million instances occupies one HDFS block of size 64MB. Initially, one core in the cluster is used to execute the proposed model, and subsequently, the experiment was conducted with a dataset size of 2.05 million, 4.1 million, 8.2 million, 16.4 million, and million 32.8 million instances on 2, 4, 8,16, and 32 cores respectively. From this experiment, it is observed that the scalability of both MR-SDkNN and MR-DWkNN decreases slowly when the size of the dataset increases, and it maintains a value of scale-up higher than 89.76% and 84.7% respectively.

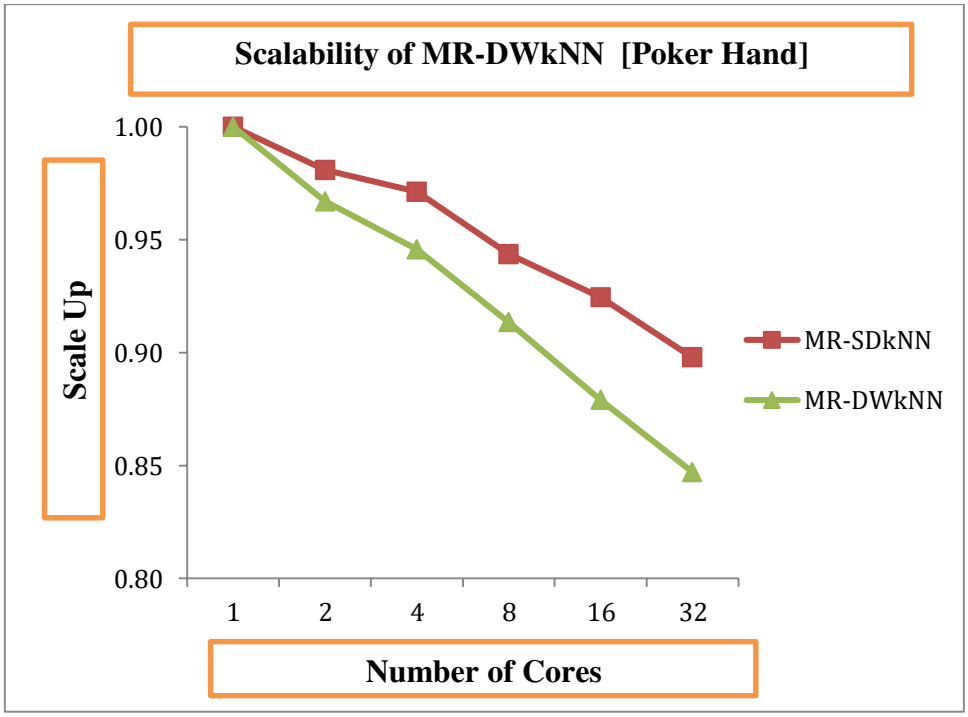

Fig.3 Scalability of MR-SDkNN and MR-DWkNN [Poker Hand Dataset]

Fig.4 shows the scalability performance results of MR-SDkNN and MR-DWkNN models on year prediction MSD datasets. Initially, the experiment was conducted with 0.515 million instances and the dataset size is 433MB. The dataset is split into 7 HDFS blocks and for each block, one map task is created. Hence, a total of 7 map tasks are created and a cluster of size 7 cores are used for its execution. Afterward, the size of the dataset and the number of cores in the cluster has been increased proportionately. The results show that both MR-SDkNN and MR-DWkNN scale up to higher than 94.78% and 91.87% respectively.

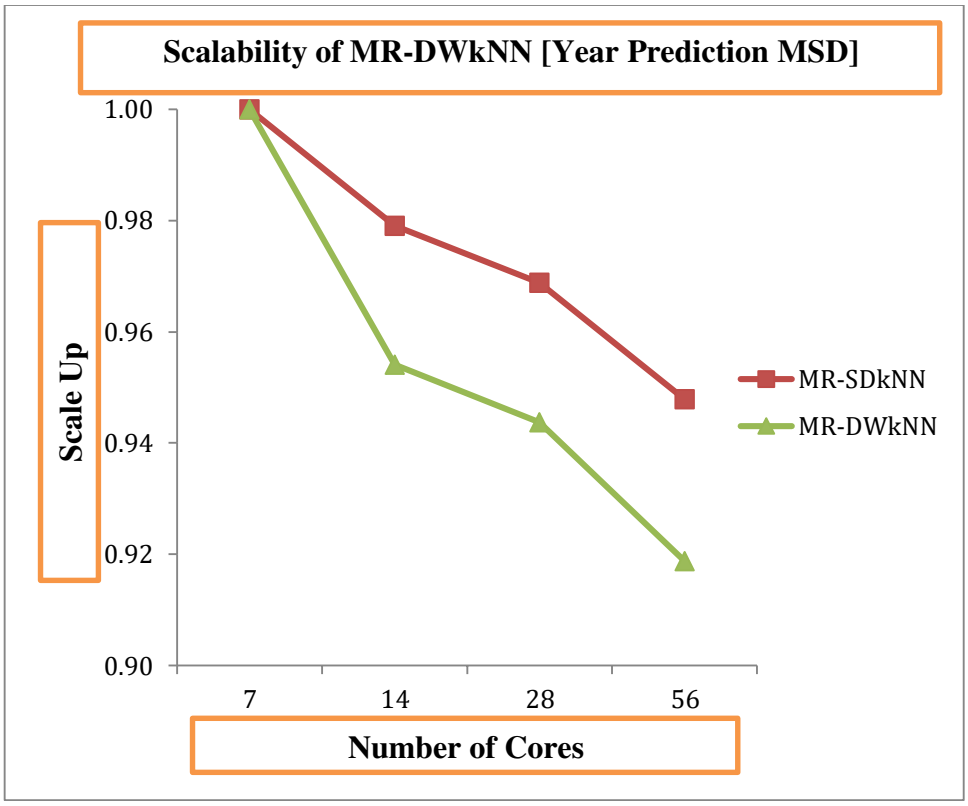

Fig.4 Scalability of MR-SDkNN and MR-DWkNN [Year Prediction MSD Dataset]

Fig.5 shows the scalability performances of our proposed MapReduce-based kNN versions on the Wave energy converters dataset. The initial size of the dataset is 123 MB with 0.288 million instances occupies one HDFS block of size 128 MB. For this experiment, the HDFS block size is configured as 128 MB. Initially, one core in the cluster is used to execute the proposed model, and subsequently, the experiment was conducted with a dataset size of 0.576 million, 1.152 million, 2.304 million, 4.608 million, and 9.216 million instances on 2, 4, 8, 16 and 32 cores respectively. From this experiment, it is observed that the scalability of both MR-SDkNN and MR-DWkNN decreases gradually when the size of the dataset and number of cores for its execution increases, and it maintains a value of scale-up higher than 87.69% and 85.67% respectively.

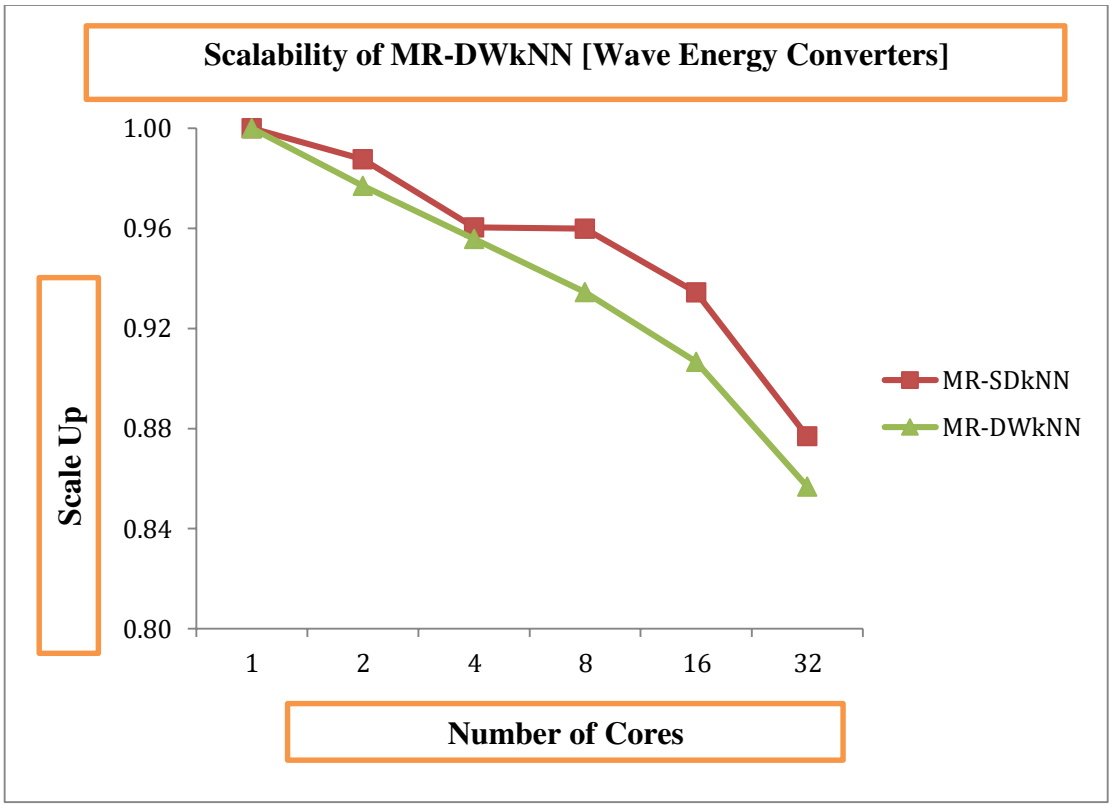

Fig.5 Scalability of MR-SDkNN and MR-DWkNN [Wave Energy Converters Dataset]

From these experiments, we can conclude that,

- Hadoop clusters can handle large volumes of datasets and provide as many processor cores as required for the execution of our proposed algorithm.
- Both MR-SDkNN and MR-DWkNN able to scale up when the size of the dataset and computing processor cores increases.

5.3 Speedup

To measure the speedup performance of both the MR-SDkNN and MR-DWkNN algorithms, two datasets under each category were chosen. The speedup experiments were performed where the number of dataset splits was increased in proportion to the number of computing cores in the cluster with a fixed size of the dataset. Table.6 summarizes the datasets used for the speedup experiments, number of instances, computing processor cores, the number of map tasks, and the speedup (x times) results obtained for both MR-SDkNN and MR-DWkNN algorithms. It is observed that the speedup of both MR-SDkNN and MR-DWkNN increases in proportion to the number of computing processor cores enabled in the cluster.

Table.6 Speedup performance of MR-SDkNN and MR-DWkNN

Dataset	#instances	#cores used	#Dataset Splits	Speedup (x times)	
				MR-SDkNN	MR-DWkNN
Higgs	1,100,000	5	5	3.66	3.56
		10	10	7.22	6.92
		15	15	10.55	10.31
		20	20	13.64	13.42
Poker Hand	2,050,000	1	1	1.00	1.00
		2	2	1.56	1.50
		4	4	3.01	2.97
		8	8	5.86	5.54
		16	16	10.91	10.59
		32	32	21.70	20.54
Year Prediction MSD	515,345	5	5	3.66	3.56
		10	10	7.12	6.62
		15	15	10.35	9.86
		20	20	13.24	12.82
Wave Energy Converters	576,000	1	1	1.00	1.00
		2	2	1.56	1.48
		4	4	2.97	2.89
		8	8	5.78	5.38
		16	16	10.59	10.11
		32	32	20.22	19.58

From these experiments, we can conclude that,

- Both MR-SDkNN and MR-DWkNN able to speedup when the computing processor cores increases for a fixed size dataset
- The execution efficiency of the proposed algorithms in the cluster improves proportionally with the size of the cluster

5.4 Influence of Neighborhood size k on performance metrics and comparison

To investigate the influence of neighborhood size k on performance metrics, we have chosen all the six datasets under two different categories. The performance of the proposed distributed algorithms is measured in a cluster with an HDFS block size of 64MB and the interval of neighborhood size k ranges from 3 to 31. Table.7 shows the influences of neighborhood size k on the classification accuracies on three benchmark classification datasets. It is observed that the classification performance of the MR-DWkNN classifier is better than the MR-SDkNN classifier with increasing neighborhood size k. However, after a certain value of k, the classification accuracy starts decreasing on all three datasets. The best classification accuracy of each classifier

on the benchmark data sets is shown in bold-faces against the corresponding k-value. The values in the parenthesis represent the corresponding dataset.

Table.No.7 Influence of neighborhood size k on the classification task

K-Value	Accuracy (Higgs)		Accuracy (Susy)		Accuracy (Poker Hand)	
	MR-SDkNN	MR-DWkNN	MR-SDkNN	MR-DWkNN	MR-SDkNN	MR-DWkNN
3	0.67382	0.69782	0.76361	0.79416	0.54784	0.55329
7	0.68688	0.71065	0.78754	0.80965	0.55064	0.56548
11	0.70214	0.74654	0.82435	0.83324	0.56462	0.57900
15	**0.71124**	**0.76428**	0.85864	0.87987	**0.57886**	**0.59809**
19	0.70892	0.76002	**0.87425**	**0.90231**	0.57031	0.58971
23	0.69745	0.75462	0.86548	0.89489	0.56112	0.57602
27	0.69126	0.74532	0.84387	0.87002	0.54387	0.56043
31	0.68542	0.73457	0.80563	0.85743	0.52301	0.54387

Table.8 shows the influences of neighborhood size k on the regression task and its performance metric determination coefficient (R^2). The Determination coefficient (R^2) shows that how close the data instances fit with the regression line and also it measures the strength of the relationship between the distributed machine learning model constructed in the training and the response variable. The three benchmark datasets Superconductivity, Year prediction MSD, and Wave Energy converters dataset under the regression category are chosen. From the results, it is observed that the prediction performance of the MR-DWkNN method is better than the MR-SDkNN method with increasing neighborhood size k. However, after a certain value of k, the R^2 value starts decreasing on all three datasets. The best R^2 value of each classifier on the benchmark data sets is shown in bold-faces against the corresponding k-value. The values in the parenthesis represent the corresponding dataset. Table.9 shows the classification accuracy of our proposed methods and compared with MR-kNN ((Maillo et al.) and MRPR – FCNN ((Triguero et al.)) methods. It is concluded that our model MR-DWkNN classifies instances with good accuracy rate for k=1,3,5 and 7 as compared other models on Poker hand dataset.

Table.No.8 Influence of neighborhood size k on the regression task

K-Value	Determination Coefficient -R^2 (Superconductivity Data)		Determination Coefficient -R^2 (Year Prediction MSD)		Determination Coefficient -R^2 (Wave Energy Converters)	
	MR-SDkNN	MR-DWkNN	MR-SDkNN	MR-DWkNN	MR-SDkNN	MR-DWkNN
3	0.89327	0.90676	0.86213	0.88126	0.90234	0.91452
7	0.89878	0.91564	0.86899	0.89675	0.90768	0.92456
11	0.91004	0.92764	0.88034	0.91502	0.91241	0.92650
15	**0.91678**	**0.93013**	0.89764	0.93013	0.92134	0.93613
19	0.90452	0.92648	**0.91245**	**0.94065**	**0.92876**	**0.94261**
23	0.88099	0.91438	0.90657	0.92344	0.91834	0.93744
27	0.87239	0.89723	0.89542	0.90723	0.90878	0.92372
31	0.86573	0.87564	0.87421	0.89756	0.89743	0.91076

Table.No.9 Comparison of classification accuracy

k-Value	Accuracy (Poker Hand)			
	MR-SDkNN	MR-DWkNN	MR-kNN (Maillo et al.)	MRPR – FCNN (Triguero et al.)
1	0.52318	**0.53786**	0.5019	0.5013
3	0.54784	**0.55329**	0.4959	0.5171
5	0.55098	**0.55971**	0.5280	-
7	0.55064	**0.56548**	0.5386	-

6 Conclusions and further work

In this research work, we have developed a MapReduce based on two different versions of the kNN algorithm called MR-SDkNN based on standard k-NN algorithm and MR-DWkNN based on distance weighted k-NN algorithm. The distributed learning model is constructed with 80% of training instances and the remaining 20% instances are used as test (query) instances. Various experiments are carried out on recently published six benchmark large volume datasets from the UCI repository. The predictive and classification performance of MR-DWkNN is evaluated in terms of three metrics as Root Mean Squared Error (RMSE), Determination Coefficient (R^2), and Accuracy. In addition to these, the scalability performance of the proposed algorithm was also tested on Hadoop Multi-node cluster. The results obtained from these experiments have shown that the main accomplishments of MR-DWkNN are the following:

- There is an increase in performance such as classification accuracy and determination coefficient (R^2) of the proposed MR-DWkNN as compared to the MR-SDkNN
- MR-DWkNN is a scalable approach in a multi-node cluster environment and a proven parallel approach with promising performance metrics achievement in Big Data applications.

The future work considered is the improvement of runtime execution of Map and Reduce tasks through the use of other big data handling frameworks such as Spark and Flink.

Authorship contribution statement:

E. Gothai: Conceptualization; Investigation; Writing – original draft; **Usha Moorthy**: Methodology, Writing – original draft; **Sathishkumar V E**: Methodology, Supervision; **Abeer Ali Alnuaim:** Writing - review & editing; **Wesam Atef Hatamleh:** Investigation, Writing - review & editing, **Dirar Sweidan**: Formal analysis, Data curation.

Acknowledgement

The authors extend their appreciation to the Researchers supporting project number (RSP-2021/384) King Saud University, Riyadh, Saudi Arabia.

Funding statement:

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflicts of Interest:

The authors declare that they have no conflicts of interest to report regarding the research work.

Data availability statement:

The datasets analysed during the current study are available in the UCI Machine Learning repository. URL: https://archive.ics.uci.edu/ml/datasets.php

References

1. T. Cover, P. Hart, nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13(1) (1967) 21–27.
2. A. Mucherino, P.J. Papajorgji, P.M. Pardalos, K-nearest neighbor classification, Data Mining in Agriculture, Springer, 2009, pp.83–106.
3. B. Yang, M. Xiang, Y. Zhang, Multi-manifold discriminant iso map for visualization and classification, Pattern Recognit. 55 (2016) 215–230.
4. J. Zou, W. Li, Q. Du, Sparse representation-based nearest neighbor classifiers for hyperspectral imagery, IEEE Geosci. Remote Sens. Lett. 12 (12) (2015) 2418–2422.
5. W. Yang, C. Sun, L. Zhang, A multi-manifold discriminant analysis method for image feature extraction, Pattern Recognit. 44 (8) (2011) 1649–1657.
6. J. Maillo, I. Triguero, F. Herrera, A MapReduce-based k-nearest neighbor approach for big data classification, IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, Volume 2, 2015, pp. 167–172.
7. D. Adeniyi, Z. Wei, Y. Yongquan, Automated web usage data mining, and recommendation system using k-nearest neighbor (KNN) classification method, Appl. Comput. Inform. 12 (1) (2016) 90–108.
8. Bijalwan, V., Kumar, V., Kumari, P. & Pascual, J. (2014). KNN based machine learning approach for text and document mining. International Journal of Database Theory and Application, 7(1): 61-70.
9. Zhang, W., Yoshida, T. & Tang, X.J. (2008). Text classification is based on multi-word with support vector machine. Knowledge-Based Systems, 21(8): 879-886.
10. Zhang, W., Yoshida, T. & Tang, X.J. (2011). A comparative study of TF* IDF, LSI, and multi-words for text classification. Expert Systems with Applications, 38(3): 2758-2765.

11. Chen, J.D. & Tang, X.J. (2017). The distributed representation for societal risk classification toward BBS posts. Journal of Systems Science & Complexity. DOI: 10.1007/s11424-016-5099-z.
12. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12:2461-2505.
13. M. Keller, M.R. Gray, J.A. Givens, A fuzzy *k*-nearest neighbor algorithm, IEEE Trans. Syst. Man Cybern. 4 (1985) 580–585.
14. T. Denoeux, A *k*-nearest neighbor classification rule based on Dempster-Shafer theory, IEEE Trans. Syst. Man Cybern. 25(5) (1995) 804–813.
15. L.I. Kucnehva, An intuitionistic fuzzy *k*-nearest neighbors rule, Notes IFS 1 (1995) 56–60.
16. M.-S. Yang, C.-H. Chen, on the edited fuzzy *k*-nearest neighbor rule, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 28(3) (1998) 461–466.
17. J. Huang, Y. Wei, J. Yi, and M. Liu, "An Improved kNN Based on Class Contribution and Feature Weighting," 2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Changsha, 2018, pp. 313-316, DOI: 10.1109/ICMTMA.2018.00083.
18. H. Liu, X. Li, S. Zhang, Learning instance correlation functions for multi-label classification, IEEE Trans. Cybern. 47 (2) (2016) 499–510.
19. H. Liu, S. Zhang, Noisy data elimination using mutual k-nearest neighbor for classification mining, J. Syst. Softw. 85 (2012) (2012) 1067–1074.
20. S. Zhang, KNN-CF approach: incorporating certainty factor to KNN classification, IEEE Intell. Inf. Bull. 11 (1) (2010) 24–33.
21. Shichao Zhang, Debo Cheng, Ming Zong, Lianli Gao, Self-representation nearest neighbor search for classification, Neurocomputing, Volume 195, 2016, Pages 137-142, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2015.08.115.
22. Hadoop. in: http://Hadoop.apache.org (Accessed on 13.01.16).
23. Jimmy Lin, Alek Kolcz, Large-scale machine learning at Twitter, Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, Scottsdale, AZ, USA, 2012, pp. 793–804.
24. Fan Zhang, Junwei Cao, Samee U. Khan, Keqin. Li, Kai Hwang, A task-level adaptive MapReduce framework for real-time streaming data in healthcare applications, Future Gener. Comput. Syst. 43–44 (2015) 149–160.
25. Panagiotis Moutafis, George Mavrommatis, Michael Vassilakopoulos, Spyros Sioutas, Efficient processing of all-k-nearest-neighbor queries in the MapReduce programming framework,Data & Knowledge Engineering, Volume 121,2019,Pages 42-70,ISSN 0169-023X,https://doi.org/10. 1016/j.datak. 2019.04.003.
26. Jianping Gou, Wenmo Qiu, Zhang Yi, Xiangjun Shen, Yongzhao Zhan, Weihua Ou, Locality constrained representation-based K-nearest neighbor classification, Knowledge-Based Systems,Volume 167,2019,Pages 38-52,ISSN 0950-7051, https://doi.org/10.1016 /j.knosys.2019.01.016.
27. Niloofar Rastin, Mansoor Zolghadri Jahromi, Mohammad Taheri,A Generalized Weighted Distance k-Nearest Neighbor for Multi-label Problems,Pattern Recognition,2020, 107526,ISSN 0031-3203,https://doi.org/10.1016/j.patcog.2020.107526.

28. Ching-Hsue Cheng, Chia-Pang Chan, Yu-Jheng Sheu,A novel purity-based k nearest neighbors imputation method and its application in financial distress prediction, Engineering Applications of Artificial Intelligence,Volume 81,2019,Pages 283-299, ISSN 0952-1976,https://doi.org/ 10.1016/j.engappai.2019.03.003.
29. Sarah M. Ayyad, Ahmed I. Saleh, Labib M. Labib,Gene expression cancer classification using modified K-Nearest Neighbors technique,Biosystems,Volume 176,2019,Pages 41-51, ISSN 0303-2647,https://doi.org/10.1016/j.biosystems.2018.12.009.
30. Yanhui Guo, Siming Han, Ying Li, Cuifen Zhang, Yu Bai,K-Nearest Neighbor combined with guided filter for hyperspectral image classification,Procedia Computer Science, Volume 129,2018,Pages 159-165,ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.03.066.
31. Aparicio-Ruiz, P., Barbadilla-Martín, E., Guadix, J. et al. KNN and adaptive comfort applied in decision making for HVAC systems. Annals of Operations Research (2019). https://doi.org/ 10.1007 /s10479-019-03489-4
32. Patel, D., Shah, Y., Thakkar, N. et al. Implementation of Artificial Intelligence Techniques for Cancer Detection. Augment Hum Res 5, 6 (2020). https://doi.org/10.1007/s41133-019-0024-3
33. Yadav, D.C., Pal, S. Thyroid prediction using ensemble data mining techniques. Int. j. inf. tecnol. (2019). https://doi.org/10.1007/s41870-019-00395-7
34. Owen O'Malley, Arun C. Murthy, Yahoo! Winning a 60 Second Dash with a Yellow Elephant. http://sortbenchmark.org/Yahoo2009.pdf (April 2009) (Accessed on 22.02.16).
35. Eric Anderson, Joseph Tucek, Efficiency matters!, ACM SIGPOS Oper. Syst. Rev. 44 (1) (2010) 40–45.
36. F.N. Aftari, J.D. Ullman, Optimizing joins in a MapReduce environment, Proceedings of the 13th International Conference on Extending DatabaseTechnology, EDBT, 2010, pp. 99–110.
37. Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M, Distributed GraphLab: a framework for machine learning and data mining in the cloud. Proc. VLDB Endow. **5**, 716–727 (2012).
38. Cormack, G.V., Smucker, M.D., Clarke, C.L., Efficient and effective spam filtering and re-ranking for large web datasets. Inf. Retr. **14**, 441–465 (2011).
39. Lin, J.: Brute force and indexed approaches to pairwise document similarity comparisons with MapReduce, Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.155–162 (2009).
40. Zhao, W., Ma, H., He, Q.: Parallel k-means clustering based on MapReduce, Cloud Computing, pp. 674–679. Springer, Berlin (2009).
41. Baraglia, R., De Francisci Morales, G., Lucchese, C., Document similarity self-join with MapReduce. In: 2010 IEEE 10th International Conference on Data Mining (ICDM), pp. 731–736 (2010).
42. Caruana, G., Li, M., Liu, Y.: An ontology enhanced parallel SVM for scalable spam filter training. Neurocomputing **108**, 45–57 (2013).
43. Liao, R., Zhang, Y., Guan, J., Zhou, S.: CloudNMF: a MapReduce implementation of nonnegative matrix factorization for large-scale biological datasets. Genomics Proteomics Bioinforma. **12**, 48–51 (2014).

44. Svendsen, M., Tirthapura, S.: Mining maximal cliques from a large graph using MapReduce: tackling highly uneven subproblem sizes. J. Parallel Distrib. Comput. **79**, 104–114 (2012).
45. Kolb, L., Thor, A. & Rahm, E. Multi-pass sorted neighborhood blocking with MapReduce. Comput Sci Res Dev 27, 45–63 (2012). https://doi.org/10.1007/s00450-011-0177-x.
46. I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, MRPR: a mapreduce so- lution for prototype reduction in big data classification, Neurocomputing 150, Part A (0) (2015) 331–345, doi: 10.1016/j.neucom.2014.04.078.
47. S. Ding, M. Du, H. Jia, Study on density peaks clustering based on k-nearest neighbors and principal component analysis, knowledge-based Systems. 99 (2016) 135–145, DOI: 10.1016/j.knosys.2016.02.00.
48. Z. Deng, X. Zhu, D. Cheng, M. Zong, S. Zhang, Efficient KNN classification algorithm for big data, Neurocomputing 195 (2016) 143–148.
49. C. Zhang, F. Li, J. Jestes, Efficient parallel KNN joins for large data in MapReduce, Proceedings of the 15th International Conference on Extending Database Technology, EDBT '12, ACM, New York, NY, USA, 2012, pp. 38–49, DOI: 10. 1145/2247596.2247602.
50. J. Maillo, I. Triguero, F. Herrera, A MapReduce-based k-nearest neighbor approach for big data classification, 9th International Conference on Big Data Science and Engineering (IEEE BigDataSE-15), 2015, pp. 167–172.
51. K. Sun, H. Kang, H.-H. Park, Tagging, and classifying facial images in cloud environments based on kNN using MapReduce, Optik 126 (21) (2015) 3227–3233, DOI: 10.1016/j.ijleo.2015.07.080.
52. Dobre, C., Xhafa, F. Parallel Programming Paradigms and Frameworks in Big Data Era. Int J Parallel Prog 42, 710–738 (2014). https://doi.org/10.1007/s10766-013-0272-7.
53. Ahmad, A., Paul, A., Din, S. et al. Multilevel Data Processing Using Parallel Algorithms for Analyzing Big Data in High-Performance Computing. Int J Parallel Prog 46, 508–527 (2018). https://doi.org/10.1007/s10766-017-0498-x.
54. Jin Qian, Duoqian Miao, Zehua Zhang, Xiaodong Yue, "Parallel attribute reduction algorithms using MapReduce", Journal of Information sciences, Elsevier, Vol. 279, pp. 671–690,2014.
55. Weaver, Jesse. "A scalability metric for parallel computations on large, growing datasets (like the web)." In Proceedings of the Joint Workshop on Scalable and High-Performance Semantic Web Systems. 2012.