Al-Salam Journal for Engineering and Technology
Journal Homepage: http://journal.alsalam.edu.iq/index.php/ajest
e-ISSN: 2790-4822p-ISSN: 2958-0862

A Parallel Approach for Optimizing KNN Classification Algorithm in Big Data

Muna H. Aljanabi [1*], Kadhim B. S. Aljanabi [1]

[1]Department of Computer Science Faculty of CS and Mathematics, University of Kufa Najaf, 964, Iraq.

*Corresponding Author: Muna H. Aljanabi

DOI: https://doi.org/10.55145/ajest.2023.02.02.019
Received February 2023; Accepted April 2023; Available online May 2023

ABSTRACT: Big data classification is the study of how to classify large amounts of data, which conventional data mining methods typically find challenging to handle. The most popular data mining technique is the K-Nearest Neighbor(KNN) classifier because of its efficiency and simplicity. The sequential KNN classifier can't handle a huge amount of data due to its highly required calculation nature, so it is improved by using a parallel technique supported by Map Reduce. This model enables us to classify the massive amount of data that exceeds terabytes. Similar to the original KNN model, this parallel implementation offers the same classification rate but less time complexity. In this paper, parallel KNN with Map Reduce has been proposed using a Hadoop multi-data node cluster. First, the dataset split into n blocks, and in each node, the mapper and reducer will be executed. The mapper phase is responsible for calculating the Euclidean distance between the training set and the targeted point, and the output of the mapper is a set of pairs of <Distance, Class> that serve as the reducer's input. In reducer, the minimum k distances are determined, and the class with maximum occurrence will represent the predicted class.
The results showed a significant improvement in time complexity for the proposed approach over the traditional one. A New York criminal data set with a size of 6.5 million records was used in this work. The tuples Latitude and Longitude were used to determine the nearest neighbors, while the Patrol-BORO was used as class label.

Keywords: Parallel processing, classification, Map Reduce, KNN classifier

1. INTRODUCTION

Among the different types of classification techniques and algorithms such as DT, Naïve Bayes, Support Vector Machine (SVM) and others, KNN represents one of the most popular and widely used classifier. The K Nearest Neighbors (KNN) algorithm is a supervised machine learning technique that classifies an item based on the findings of a majority of its K neighbors. KNN is regarded as a "lazy" classification technique because it doesn't train at all when you supply the training data. In other words, the algorithm repeats computations for each input point [1]. For this reason, working with Big Data will increase computation time exponentially.

To reduce the time complexity of KNN when applied on huge data set, parallel approach with Hadoop streaming has been used. Map Reduce is the most important part of Hadoop, where it is the part responsible for executing the parallel process. Map Reduce programming divides the task into several parts that are executed concurrently [2]. In our research KNN algorithm applied to classify huge data, firstly the data are divided into several blocks (64 or 128 MB) to be stored in Hadoop Distributed file system(HDFS)number of blocks equal to number of Mappers. Mapper outputs will be <distance, class label> pairs, the output of each Mapper sorted and then merged with the other Mappers output and transferred to Reducer. Sorting process is done automatically when the data transferred from Mapper to Reducer. This process decrease the time required to apply KNN on Big data. A lot of previous researches dealt with the implementation of the KNN algorithm in standard Map Reduce, and it took a lot of time to sort the data.

However, in this work, we dispensed with the data sorting step and relied on the automatic sorting step in Hadoop Streaming and the data has been used is about 24 million records with 1.4 GB size. In this paper, section two contains the related works, section three contains the traditional KNN algorithm, improved KNN algorithm with small K and overview about KNN on Big data. Section four contains overview about Hadoop environment (Map Reduce and

Hadoop streaming). Improved KNN algorithm is proposed on section five and the last sections (six and seven) are contain the experimental setup and results of KNN algorithm on Hadoop.

2. RELATED WORKS

- **Vinoth, R.V.R., Kannan, K.N., and Shunmuganathan, K.S.D.K(2019)** proposed Novel technique using KNN to process agriculture data (big data) to predict the crop yield. They handle and process Big data using Hadoop by compose Hadoop cluster with three nodes(master and two slaves) [3]
- **Ding, Q., & Boykin, R. (2017)** used Hadoop environment to execute KNN algorithm. They composed a multi-node cluster that contains three computers wired together through Ethernet cables connected to a network hub. Each laptop had the Ubuntu operating system, and the hardware requirements were modest. The 179 lines in the used data set are divided into 165 lines for training and 14 for testing. They found that the classification accuracy varies with the provided k values [4].
- **Saadatfar, H., Khosravi, S., Joloudari, J. H., Mosavi, A., & Shamshirband, S. (2020)** proposed improved algorithm depends on using K-mean clustering to prune the input data for the first time. Then the classification will occur between each cluster's centroid and test set using KNN classifier [5]
- **Ma, C., & Chi, Y. (2022)** proposed an optimized KNN algorithm where the classic Euclidean metric is replaced with the Pearson correlation coefficient, and the distance calculation equation is then improved by using the entropy weight technique combined with Pearson's measure to account for attribute values of datasets. Once K had been set, they introduced the Gaussian Function to select the classification. This algorithm has been executed via Hadoop environment. The authors used four datasets: Iris, Breast cancer, Dry soybean, and Pulsar [6]

3. TRADITIONAL KNN -ALGORITHM

KNN algorithm is one of popular algorithms that used to classify unknown objects without training. The weak point in KNN is that it exhibits O(mn+ mlog2m) when classifying new instance over m instances and n attributes, where O(mn) is the time complexity for calculating the distances between the new instance and each of the training instances. In addition, O(m log2 m) is the time complexity for sorting the distances when finding the k-nearest neighbors of the new instance so that with the increase of data points the time complexity will increase[7]. The KNN has a number of weaknesses despite how straightforward it is. Listed below are a few drawbacks:

•Because it is necessary to store all training data in memory in order to classify a new sample, memory usage is high. On the other hand, the runtime of the KNN algorithm grows as the dataset volume increases.

• Expensive computations are required to compare each test sample to every training sample. As a result, the computing cost of this approach is high.

• Dependency on data: The KNN algorithm is very dependent on each sample in the training dataset. It, therefore, has a high noise sensitivity. The classification outcomes could significantly deviate from the ideal outcomes.

KNN Algorithm Pseudocode [9]
Input: Dataset D, K(number of required points), C(classes), and Data point P to be classified Output: Class Label of P Begin Calculate "Dist (x, xi)" i =1, 2, ….., n; where Dist denotes the Euclidean distance between the point x and all other points. Sort the calculated n Euclidean distances in non-decreasing order. Let k be a positive integer, retrieve the first k distances from this sorted list. Find those k-points corresponding to these k-distances. Let ki denotes the number of points belonging to the ith class among k points i.e. $k \geq 0$ If ki >kj $\forall$ i $\neq$ j then put x in class i.// choose the class label with maximum frequency within K Return class label End

The K-NN algorithm has some issues when dealing with big data. These issues can be listed as follows [8]

1. Running time.
2.Memory consumption

3.1 KNN-ENHANCEMENT FOR SMALL K VALUE

If K value is small, it is possible to improve the traditional KNN algorithm implementation time, so that the smallest k distances are retrieved instead of sorting the entire list of distances. For example, if we have K=10, and the training set has 1000 points, it is possible to find smallest 10 distances instead of sorting 1000 distances. This will enhance required implementation time very well.

3.2 KNN FOR MASSIVE DATA AND BIG K VALUE

The application of the KNN algorithm to big data requires a very long time, and to address this problem, modern methods have been used in processing big data, and one of the most famous of these methods is the parallelization using Map-Reduce programming model on Hadoop, a distributed software framework for mass data processing with advantages including reliability, efficiency and scalability. As one of the core designs of Hadoop, Map-Reduce highly abstracts the parallel computing processes running on large clusters into Map and Reduce, significantly simplifying the large-scale parallel data processing. Our suggested study is to using Hadoop to reduce the running time of KNN-algorithm.

4. HADOOP ENVIROMENT

Hadoop is an open-source platform used to store and process the huge amounts of data. It consists of three main components: (HDFS, Map-Reduce, and YARN). HDFS used for storing huge amounts of data in the form of blocks. The block size is 64 in default. Map--Reduce is a programming model used for processing data. It consists of two parts: job tracker and task tracker. Job tracker used to monitor the current job, while task tracker monitors the progress of the job processing and gives reports about the job to the name node. YARN is the abbreviation of (Yet Another Resource Negotiator). The responsibilities of YARN is resource allocation, it consists of resource manager and node manager. In Hadoop environment, two phases should be running: DFS and YARN. DFS runs the Name node, Data node and secondary Name node, while YARN runs resource manager and node manager.

4.1 MAP-REDUCE

A Java-based software framework called Map-Reduce enables programmers to run the exact computation simultaneously on several machines to process data much faster, reliably, fault-tolerantly, and effectively. Apache Hadoop is often used for this method because it is free and works well with the Map-Reduce programming language [9]

4.2 HADOOP STREAMING

Commonly, Hadoop environment is a java dependent. This can be changed by using Hadoop streaming where multiple languages, including Python, C++, and others, can be integrated with Hadoop Map Reduce. Python, an open source programming language, is used to carry out the proposed task. It is run in the Map Reduce framework using the Hadoop streaming interface, a tool that is included in the Hadoop distribution. With the help of this tool, we can construct and execute Map Reduce tasks using any script as the mapper and reducer. The mapper uses the Hadoop streaming tool Stdin to read the data, provides the mapped key value pairs to the reducer, and uses Stdout to output the results of the reducing operation, which are then stored in the HDFS[10], The mapper phase is responsible for calculating the Euclidean distance between the training set and the target point, and the output of the mapper is a set of pairs of <Distance, Class> that serve as the reducer's input. In reducer, the minimum k distances are determined, and the class with maximum frequency will represent the predicted class. Figure 1 illustrate KNN algorithm in Hadoop streaming.

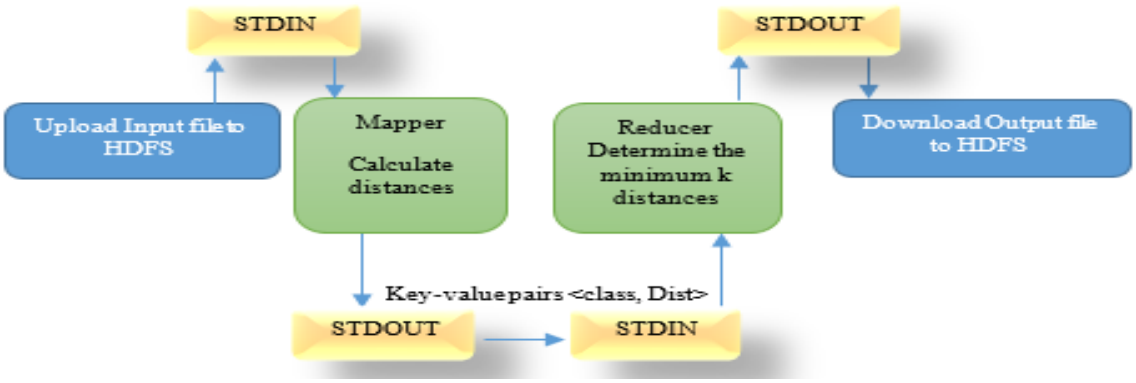

FIGURE 1. - KNN with Hadoop streaming

4.3 IMPROVED KNN-ALGORITHM (Proposed Approach)

In this paper, KNN algorithm using Hadoop streaming has been introduced. Hadoop streaming is a technique included into Hadoop. To process the algorithm using Hadoop streaming, input file should be transferred to Hadoop Distributed File System(HDFS). The meta data of input file will store on Name node while the actual data store in Data node. Each Data node have three replicas of file in default. These replicas prevent fault occurrence. After storage stage completed, the Map-Reduce can be executed on data. Map-Reduce is programming model where the user writes the algorithm. In the standard Hadoop, Map and Reduce programs written by java programming language. In this paper, python programming used to write Map-Reduce codes. To parallelize the KNN algorithm, Hadoop Map-Reduce programming has been utilized. In Map the data is read from Sys. Stdin, and the output is printed to Sys. Stdout. The output of Map will be the input data of Reduce phase. KNN- algorithm parallelization in Map-Reduce is done by calculation of Euclidean distance (between the new point and the data set points on Map phase, where the output in form of <distance, class>. In Reducer the output of mapper will be the input data. When the data transferred from Mapper to Reducer the stage of sorting is executed automatically, so the input of Reducer will be sorted. Reducer extracts the k <key, value> pairs. Since the data transmitted from the Mapper to the Reducer will be automatically arranged depending on the key, we will take advantage of this step in our paper by making the key is the distance between the new sample and training points, so the distances will sort from minimum to maximum. we will get the least k distances in Reducer. Note that, if the Reduce process ignored, the Mapper output will be unsorted.

Algorithm Mapper(split and map)

Input: data set D, N number of objects in D, Object to be classified M, K
Output: Distances from M to each object in D with the related class label

1- //split D into groups depending on the size of input file
2- R=N/Nodes// R number of objects in each node
3- For s=0 to R
4- X[s]=D[i++]// X is sent to Node 1,2,3
5- Y[s]=D[R+i++]// Y is sent to Node 1,2,3
6- Z[s]=D[2*R+i++]//Z is sent to Node 1,2,3
7- …
8- End for s
9- //Find the distance for each group
10- i ⟵ 0
11- **For** each line in X **do**
12- Elements ⟵ line. Split
13- **For** each element in Elements **do**
14- Distance[i++] ⟵ Euclidean (M, Elements)
15- **End for**
16- **End for**
17- Emit < Dist, class >
18- Return Emit

Algorithm Merging and Sorting

Input: < Dist, class>, k<represent the required neighbors >
Output: sorted Distances list

1- Merging the output of Mapper
2- Sorting the merged list
3- Step one and two is applied automatically before Reducer stage and the sorted distances stored on Sys. Stdin.

Reducer Algorithm(one reducer required)

Input: sorted list of <Dist, Class>
Output: predicated class for test data
Begin

1- i ⟵ 0
2- **For** each line in Sys. Stdin **do**
3- Elements ← line. split
4- Distance[i] ← (Elements)
5- **If** i < k
6- **i++**
7- **Else exit ()**
8- Let $\mathbf{k}_i$ denotes the number of points belonging to the i^{th} class among **k** points i.e. $k \geq 0$
9- If $k_i > k_j \; \forall \; i \neq j$ then put x in class i.

End

5. EXPERMENTAL SETUP

A wirelessly connected system with three nodes was used to implement the k-Nearest Neighbor approach. Three nodes were employed as data nodes and task trackers, while the remaining node served as a Name node and job tracker. On each node, Ubuntu 22.10 was the operating system in use. Python3 was used to program the KNN-algorithm in sequential and Map Reduce implementation. All the nodes have Apache Hadoop version 3.3.4 installed. The block size configured as 128 Megabyte. As we said previously, the data set which used in our paper is a New York criminal dataset, it has 6.5 million records and 8 tuples. In our paper, only three tuples were used, and other data sets (12,24) millions record were generated from the original dataset. Each data set implemented on sequential, and multi node Hadoop cluster. The generated data set consists of four tuples (Id, Latitude, longitude, PATROL_BORO) Latitude mean Latitude coordinate for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326) Longitude mean Longitude coordinate for Global Coordinate System, WGS 1984, decimal degrees (EPSG 4326) and PATROL_BORO refer to the Patrol BORO names in New York cities. Many samples of dataset are shown in Table 1

Table 1. - Some samples of data set

ID	PATROL BORO	Latitude	Longitude
1	PATROL BORO QUEENS SOUTH	40.68	-73.84
2	PATROL BORO BKLYN NORTH	40.69	-73.94
3	PATROL BORO MAN NORTH	40.83	-73.94
4	PATROL BORO BKLYN SOUTH	40.64	-73.91
5	PATROL BORO BRONX	40.82	-73.92
6	PATROL BORO BKLYN NORTH	40.67	-73.88
7	PATROL BORO MAN NORTH	40.85	-73.94
8	PATROL BORO BRONX	40.86	-73.89
9	PATROL BORO BKLYN SOUTH	40.64	-74.04
10	PATROL BORO STATEN ISLAND	40.59	-74.09
11	PATROL BORO BKLYN NORTH	40.68	-73.9
12	PATROL BORO BKLYN SOUTH	40.66	-73.94
13	PATROL BORO QUEENS NORTH	40.79	-73.83
14	PATROL BORO QUEENS NORTH	40.74	-73.88
15	PATROL BORO MAN NORTH	40.79	-73.94
16	PATROL BORO MAN SOUTH	40.76	-74
17	PATROL BORO BKLYN SOUTH	40.64	-74.02
18	PATROL BORO QUEENS NORTH	40.79	-73.8
19	PATROL BORO BKLYN NORTH	40.7	-73.92

6. EXPERMENTAL RESULTS

The sequential implementation of KNN-algorithm on big data is not efficient, because in each new sample classification it will need to calculate the distances between the sample and the points of the dataset as a whole, the time required to apply this step in sequential is more than the time taken in Hadoop Mapper job. Table 2 shows the time taken to perform this step on different dataset size. Table 3 shows distance calculation results of Mapper (without sorting) are same as results of sequential.

Table 2. - Time Required to calculate distances in KNN in sequential Approach

No. records	Size in megabytes	Time required In sequential
6 millions	305MB	185 s
12 millions	633MB	353 s
24 millions	1.4 GB	705 s

Table 3. - The distance calculation results of Mapper (without sorting) are same as results of sequential

Distance	Distance
119.53	PATROL BORO BKLYN NORTH
119.67	PATROL BORO MAN NORTH
119.48	PATROL BORO BKLYN SOUTH
119.66	PATROL BORO BRONX
119.51	PATROL BORO BKLYN NORTH
119.69	PATROL BORO MAN NORTH
119.7	PATROL BORO BRONX
119.48	PATROL BORO BKLYN SOUTH
119.43	PATROL BORO STATEN ISLAND
119.52	PATROL BORO BKLYN NORTH
119.5	PATROL BORO BKLYN SOUTH
119.63	PATROL BORO QUEENS NORTH
119.58	PATROL BORO QUEENS NORTH
119.63	PATROL BORO MAN NORTH
119.6	PATROL BORO MAN SOUTH
119.48	PATROL BORO BKLYN SOUTH
119.63	PATROL BORO QUEENS NORTH
119.54	PATROL BORO BKLYN NORTH
119.53	PATROL BORO BKLYN NORTH
119.67	PATROL BORO MAN NORTH

This step can be implemented using Hadoop streaming by Map-only job where the Reduce phase is ignored. Table 4 shows the time to perform this step on different data set size.

Table 4. - Time to execute distance calculation in KNN in Map-only job and number of splits of each input file

No. records	Size in megabytes	Time In Map -only job/second	No. splits
6 millions	305MB	77s	3
12 millions	633MB	158s	5
24 millions	1.4 GB	310 s	11

Due to the Reduce phase ignoring, the output of Map-only job is unsorted. The second step of KNN-algorithm represented by sorting the distances and extract k minimum distances. In sequential implementation the sorting phase takes a very large amount of time while the Map-Reduce job executes this step automatically when Reduce phase is called. Figure 2 shows Map-Reduce job of KNN-algorithm on 12 million records data set (633 Mega Bytes), Table 5 shows time required to implement Map-Reduce job.While Table 6 shows the results of reducer when k=10.

Table 5. - Time to execute KNN in Map-Reduce job k=1000

No. records	Size in megabytes	Time In Map -Reduce job/second
6 millions	305 MB	120 s
12 millions	633 MB	242 s
24 millions	1.4 GB	505 s

Table 6. - Reducer output when (k=10)

[Distance , Class]
[['119.48, PATROL BORO BKLYN SOUTH'],
['119.48, PATROL BORO BKLYN SOUTH'],
['119.48, PATROL BORO BKLYN SOUTH'],
['119.5, PATROL BORO BKLYN SOUTH'],
['119.51, PATROL BORO BKLYN NORTH'],
['119.52, PATROL BORO BKLYN NORTH'],
['119.53, PATROL BORO BKLYN NORTH'],
['119.58, PATROL BORO QUEENS NORTH'],
['119.6, PATROL BORO MAN SOUTH'],

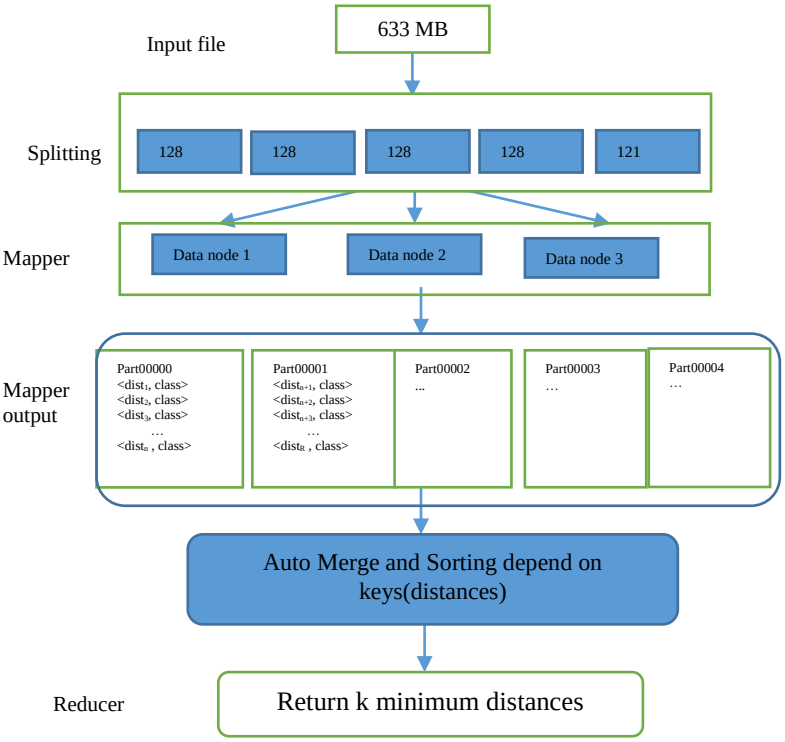

FIGURE 2. - Improved KNN

7. CONCLUSIONS

Due to the iterative nature of KNN algorithm, it suffers a lot when dealing with huge amount of data as shown in the results given in table (1).

The experimental results showed high improvement in time and time complexity when implementing KNN algorithm in parallel approach over the traditional one.To handle huge amounts of data, a suitable environment must be provided. Hadoop Streaming environment is one of the popular environments for parallel processing of big data. It is an efficient environment because it supports several languages, including Python. Python also features many libraries that facilitate user work.One of the worst problems that the user encounters when dealing with the KNN algorithm is the huge amount of computational operations that must be repeated in each classification operation, because this algorithm is lazy and cannot be learned, and to deal with this amount of computational operations, it is appropriate to use a parallel environment.The process that separating the Mapper from the Reducer is the sorting process. The data is arranged according to the key to be entered to reducer this will facilitate Reducer work.It is not recommended to implement some algorithms that do not need to sort the data in Map- Reduce because they will need more time due to the sort phase of the data after the mapping step.

8. FUTURE WORK

Proposing a new approach to deal with Big Data having larger number of features, where parallel approach has significant effect on time complexity.Converting The most popular classification algorithms and other Data Mining algorithms into parallel approach aiming to improve their time complexities. Proposing parallel approaches for data other than textual and structural data type such as audio and video files.

FUNDING

No funding received for this work

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their efforts.

CONFLICTS OF INTEREST

The authors declare no conflict of interest

REFERENCES

[1] Abu Alfeilat, H.A., Hassanat, A.B., Lasassmeh, O., Tarawneh, A.S., Alhasanat, M.B., Eyal Salman, H.S., and Prasath, V.S.: 'Effects of distance measure choice on k-nearest neighbor classifier performance: a review', Big data, 2019, 7, (4), pp. 221-248

[2] Hussain, T., Sanga, A., and Mongia, S.: 'Big data hadoop tools and technologies: A review', in Editor (Ed.)^(Eds.): 'Book Big data hadoop tools and technologies: A review' (2019, edn.), pp.

[3] Vinoth, R.V.R., Kannan, K.N., and Shunmuganathan, K.S.D.K.: 'A NOVEL PREDICTION APPROACH TO ANALYZE BIG DATA USING K-NEAREST NEIGHBOR ALGORITHM', Journal of Electrical Engineering, 2019, 19, (1), pp. 5-5

[4] Ding, Q., and Boykin, R.: 'A framework for distributed nearest neighbor classification using Hadoop', Journal of Computational Methods in Sciences and Engineering, 2017, 17, (S1), pp. S11-S19

[5] Saadatfar, H., Khosravi, S., Joloudari, J.H., Mosavi, A., and Shamshirband, S.: 'A new K-nearest neighbors classifier for big data based on efficient data pruning', Mathematics, 2020, 8, (2), pp. 286

[6] Ma, C., and Chi, Y.: 'KNN Normalized Optimization and Platform Tuning Based on Hadoop', IEEE Access, 2022, 10, pp. 81406-81433

[7] Zhao, Y., Qian, Y., and Li, C.: 'Improved KNN text classification algorithm with MapReduce implementation', in Editor (Ed.)^(Eds.): 'Book Improved KNN text classification algorithm with MapReduce implementation' (IEEE, 2017, edn.), pp. 1417-1422

[8] Maillo, J., Triguero, I., and Herrera, F.: 'A mapreduce-based k-nearest neighbor approach for big data classification', in Editor (Ed.)^(Eds.): 'Book A mapreduce-based k-nearest neighbor approach for big data classification' (IEEE, 2015, edn.), pp. 167-172

[9] Shamki, Z.A.M., and Rabee, F.: 'Image mining technique using Hadoop map reduce over distributed multi-node computers connections', Al-Salam Journal for Engineering and Technology, 2022, 1, (2), pp. 18-24

[10] Zarei, A., Safari, S., Ahmadi, M., and Mardukhi, F.: 'Past, Present and Future of Hadoop: A Survey', arXiv preprint arXiv:2202.13293, 2022

[11] https://dataaspirant.com/k-nearest-neighbor-classifier-intro