TELKOMNIKA Telecommunication Computing Electronics and Control
Vol. 21, No. 1, February 2023, pp. 235~246
ISSN: 1693-6930, DOI: 10.12928/TELKOMNIKA.v21i1.24290

Big data classification based on improved parallel k-nearest neighbor

Ahmed Hussein Ali[1,2], Mostafa Abduhgafoor Mohammed[3], Raed Abdulkareem Hasan[4], Maan Nawaf Abbod[3], Mohammed Sh. Ahmed[5], Tole Sutikno[6]
[1]Department of computer, College of Education, AL-Iraqia University, Baghdad, Iraq
[2]Computer Science Department, AL Salam University College, Baghdad, Iraq
[3]Department of Arabic Languge, Imam Aadham University College, Baghdad, Iraq
[4]Department of Electrical techniques, Faculty of Al-Hweija Technical Institute/Noerthern Technical University, Mosel, Iraq
[5]Department of Petroleum Systems and Control Engineering, College of petroleum Processes Engineering, Tikrit University, Tikrit, Iraq
[6]Department of Electical Engineering, Universitas Ahmad Dahlan, Yogyakarta, Indonesia

Article Info

Article history:

Received Jul 14, 2022
Revised Oct 26, 2022
Accepted Nov 05, 2022

Keywords:

Big data
K-nearest neighbor
Machine learning
Parallel processing
Radoop
Spark

ABSTRACT

In response to the rapid growth of many sorts of information, highway data has continued to evolve in the direction of big data in terms of scale, type, and structure, exhibiting characteristics of multi-source heterogeneous data. The k-nearest neighbor (KNN) join has received a lot of interest in recent years due to its wide range of applications. Processing KNN joins is time-consuming and inefficient due to the quadratic structure of the join method. As the number of applications dealing with vast amounts of data develops, KNN joins get more sophisticated. The authors seek to save money on computer resources by leveraging a large number of threads and multiprocessors. Six popular datasets are used to apply the method and evaluate the sequential and parallel performance of the KNN technique. These datasets are used to compare the sequential and parallel performance of the KNN method. When compared to a matching multi-core solution, the final implementation saves computing resources. It has been optimized to utilize as little RAM as possible, allowing it to manage high-resolution photo data without sacrificing efficiency. The authors will use the technique they presented using Spark Radoop. Our performance research validates the supplied method's efficacy and scalability.

Corresponding Author:

Mostafa Abduhgafoor Mohammed
Department of Arabic Languge, Imam Aadham University College, Baghdad, Iraq
Email: alqaisy86@gmail.com

1. INTRODUCTION

At times, large amounts of data might be difficult to comprehend and even analyze [1], [2]. This challenge can be solved by using data mining techniques, which may be used to derive information from the data collected. The twenty-first century is the era [2] in which there is an enormous quantity of data, and that number is growing at an alarming rate. When dealing with such large amounts of data, the usual performance of computers is insufficient, and more intelligence is required to keep up with the current expansion of big data [3]–[17]. High-performance computing (HPC) should be taken into account while designing and creating software since it has the potential to increase computation speed while also giving more accurate results at a lower total cost than traditional computing [18]–[25]. HPC may be classified into a variety of categories, the most prominent of which are: computer clusters, grid computing, cloud computing, graphical processing units (GPU), microprocessors, and field programmable gate array (FPGA) [4], [20]–[22], [24].

In order to get high performance at the GPU level, it is possible to employ distributed memory programming with several processors, which is one of the ways that was used to achieve this outcome.

High-speed computing may be used to deal with large amounts of data in a variety of ways, one of which is through parallelizing machine learning algorithms and other methods. As a result of the simplicity and effectiveness with which it classifies data, the k-nearest neighbors (KNN) algorithm [26]–[41], is one of the most extensively used machine learning methods. KNN is a classification strategy that selects the item to category based on its proximity to the nearest point in the training set. If this approach is compared to other algorithms, it is considered to be straightforward, capable of dealing with a noisy dataset, and capable of producing decent results. The algorithm's disadvantage, on the other hand, is that it is difficult to compute and that the cost of putting it into practice is significant, as previously said. The method's parallel construction is utilized to reduce the complexity of the computation as well as the time required to do the distance calculation for each point in the dataset. As a result, the purpose of this article is to accelerate the KNN method through the use of multiple threads and multiple processors, thereby reducing the time required to run the algorithm.

Outlier identification, classification, molecular motion analysis, geographic databases, and pattern recognition are just some of the applications for which the KNN join and its variations have lately received a great deal of interest. It has also been stated that the performance of the popular k-means or k-medoid classification algorithms may be greatly enhanced by incorporating KNN joins into the process. When two relations, R and S, are joined together, the KNN join is performed. It yields every point in one relation R and the top-k nearest points in the other relation S. When dealing with a high-dimensional dataset, we take into consideration the difficulty posed by determining the k nearest neighbors of a query point. In order to efficiently handle this problem, we want to improve the performance of an existing method by parallelizing it as well as making it more tolerant to stragglers. While the KNN issue is not a new concept, it is frequently employed as a first step in a wide range of real-world applications, including genomics and customized search, network security, and web-based recommendation systems. As data and dimensionalities expand in the age of big data, KNN algorithms are frequently found to be a bottleneck. There is a large amount of research on the topic of quick closest neighbor retrieval [32]–[41].

An implementation of the parallel KNN technique based on MapReduce with a heterogeneous cluster was published in a work by [42] and it was carried out by employing the block nested loop strategy for KNN-joins. Data is partitioned into equal-sized blocks during the map stage, and these blocks are then subdivided into buckets during the sort step. Following that, during the reduce phase, the reducer does a block nested loop KNN join for each bucket that will be saved as depth first search (DFS) files, and the procedure is repeated for each bucket that will be stored as DFS files. Analysis of the findings revealed that the partitions were balanced with respect to the running duration and that speedup increased with the increasing size of the cluster. Communication using Hadoop block R-tree join, on the other hand, was found to be twice as effective as communication using Hadoop Z-value KNN Join. Furthermore, researchers discovered that as the dimension of communication increases, there is a decrease in recall and precision, leading to a lower ability to reach more than the average. It was published in the Journal of Scientific Computing in 2012 that a second study by Sismanis *et al*. [43] used the KNN algorithm for greater dimension with multi-core processors in the graphics processing unit (GPU). The parallelization of the KNN algorithm in the sorting process is accomplished through the use of truncated bitonic sorting techniques (TBiS). According to the conclusions of the study, it was found that the performance of the GPU with TBiS outperforms that of the sort and pick procedures.

Rajani *et al.* [44] investigated parallel k-nearest neighbor execution using tree-based data structures in open multi-processing) and the Galois framework. To implement KNN in OpenMP, four different forms of threading can be employed: 1, 4, 8, and 16 threads. Amdahl's law was used to assess how much quicker the implementation was compared to the original. With modified national institute of standards and technology (MNIST) datasets, the researchers observed that ball trees beat k-d trees in terms of performance in higher dimensions. They also discovered that ball trees are more efficient than Scikit-learn in Python when compared to other machine learning techniques. However, when the size of the datasets shrinks, the processing time gets faster, with a linear speedup for the datasets as the size of the datasets decreases. In [45]–[48] produced an OpenCL-based implementation of the KNN method for usage on FPGA and GPU systems, which was published in the Journal Scientific Reports. In order to allow the researchers to do distance calculation and distance ranking in parallel, they performed the bubble sort after data was sent from the CPU to the FPGA. Initially, a parallel distance calculation is carried out to estimate the distance between each item, and then a sorting operation is carried out to discover which of the items has the k-smallest distance between them. According to the findings of the study, when it comes to calculating speed, the GPU surpasses the CPU. When the average is used, the FPGA version of Joule, on the other hand, surpasses the GPU implementation. KNN performance depends on k and the proximity measure. If k is too low, the test sample will be impacted by noisy points and may overfit. Alternatively, if k is too high, the neighborhood may not adequately represent the class. In this study, we evaluated k values of 1, 10, and 100 (fine, medium, and coarse neighborhoods) to see if the model's accuracy was influenced.

Also, we used several distance metrics to evaluate their overall efficacy. When employing proximity-based classifiers, the size and range of the data values must be addressed. So, we used Z-score normalization to reduce problems with closeness predictions.

The purpose of this paper is to outline all of the particulars and requirements that must be met in order to constructs to save money on computer resources by leveraging a large number of threads and multiprocessors. Six popular datasets are used to apply the method and evaluate the sequential and parallel performance of the KNN technique. These datasets are used to compare the sequential and parallel performance of the KNN method. This paper is outlined as: section 2 reviewed the related works regarding the recommended method, while sections 3 provided a brief discussion of the suggested methodology. Section 4 analyzed the evaluation results from the several experiments conducted in this study, while section 5 concluded the paper.

2. RELATED WORKS

2.1. Apache Spark Radoop

RapidMiner Radoop expands the typical RapidMiner in-memory capabilities by offering advanced operators that are built for execution in the Hadoop data processing system [47], [49]–[53]. Data transformations and sophisticated and predictive modeling are supported by more than 60 operators that operate on a distributed Hadoop cluster in the same way as Hadoop itself. RapidMiner Radoop makes use of RapidMiner Studio's visual workflow designer to make the creation, execution, and maintenance of predictive analytics in Hadoop simpler. RapidMiner Radoop is available now. The code-free environment and built-in intelligence reduce the complexity of Hadoop, allowing you to concentrate on solving business challenges rather than being distracted by dead ends and technical obstacles. Radoop takes care of the workflow execution so that the user doesn't have to worry about it. All of the calculations are done in the Hadoop cluster, which is where the data lives. This makes predictive analytics effective and highly scalable, even for terabytes and petabytes of data [6], [54]–[75].

RapidMiner is an addon that we developed to guarantee that Hadoop is fully integrated with RapidMiner [76]. Radoop, as a data science software platform, reduces the difficulties associated with data preparation and machine learning on Hadoop and Radoop Spark, as shown in Figure 1. Radoop provides additional operators for RapidMiner and communicates with the Hadoop cluster in order to complete task execution. All operations and data processing in RapidMiner Studio operate in parallel as a result of the use of SparkRM, which is part of the Hadoop ecosystem. This is achieved by the use of Apache Spark as the job execution tool, which allows for the expansion of use cases and the development of more powerful algorithms as compared to MLlib. Several Hive and Mahout data analytics routines were reused in this study due to the highly optimized nature of certain of their data analytics capabilities. This research developed an addition that would aid in the achievement of tight integration while also providing the same Hadoop features that are typically used in memory-based RapidMiner operations. First, the RadoopNest meta-operator is added to the RadoopNest meta-operator, which holds the general cluster settings such as the IP address of the Hadoop master node, and then the RadoopNest meta-operator is removed from the cluster. The remaining Radoop operators can only be used within this metaoperator and not outside of it [47], [51]–[53], [77].

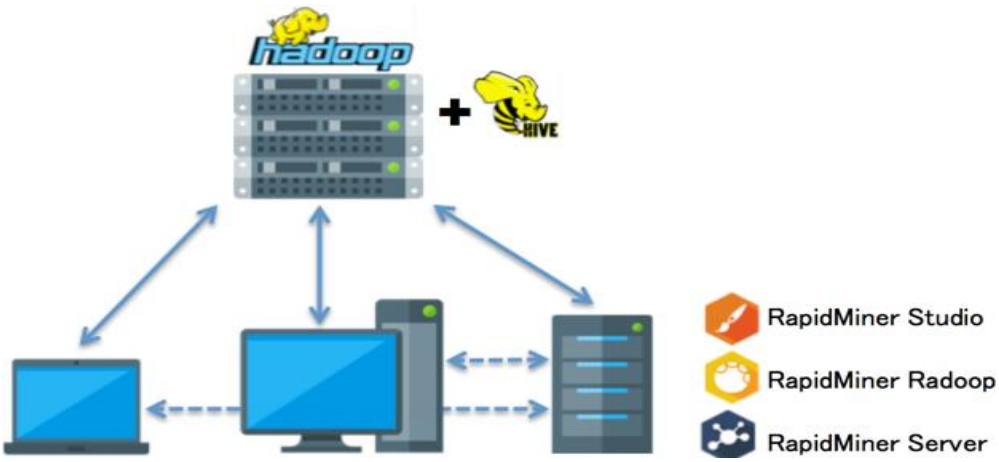

Figure 1. Spark Radoop architecture

2.2. MapReduce

MapReduce is a programming methodology created by Google to handle large-scale data analysis. It is based on the Hadoop framework [11], [58], [78]–[81], [64]–[67], [69]–[71], [74]. It is employed in a wide variety of applications. Data mapping and reduction are accomplished through the use of the MapReduce framework [82]. Work is divided and distributed across the nodes in the distributed cluster using the map functions, which split and distribute the work among them. It is necessary for them to process a collection of key/value pairs in order to generate a set of intermediate key/value pairs. In order to resolve the results, the reduction functions aggregate all of the intermediate values that share a common intermediate key, as well as all of the work that must be done in order to resolve the results. Apache Hadoop is a Java-based framework for MapReduce implementation that is available as a free download [83]. It provides support for the processing of large datasets in a distributed computing environment, as well as the storage of data in a distributed file system that is both fault-tolerant and scalable (HDFS). Hadoop breaks the project down into smaller tasks, such as mapping and reducing tasks, and then combines them all together to complete the job. It is necessary to divide the input data into segments of a predetermined size, which are referred to as input splits [84]. Each map task is executed in a separate thread and deals with a logical partition of the data that is stored on the HDFS [6], [59], [60], [64], [65], [69], [72], [85].

3. METHOD

The KNN [1], [2], [44], [50], [54]–[56], [86]–[88], [4], [26], [27], [32], [37]–[42] is a non-parametric pattern recognition approach that is used for classification and regression in pattern recognition. In our example, the algorithm is utilized for classification, with the outcome being a class membership [89]. Objects are categorized by a majority vote of their neighbors, with the item being given to the class that has the greatest number of members among its k most immediate neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of the object's nearest neighbor, which is the single nearest neighbor. Due to the fact that it memorizes the whole dataset, the KNN approach is considered a lazy learning classifier. Instead, it creates a discriminative model, similar to many of the classifiers that have been studied in the past (i.e., neural networks and decision trees). KNN plots nonlinear decision boundaries (Xie) on a graph and assists in assigning data points to one of the areas plotted on the graph [90]. It is the fundamental concept of KNN that if a test sample, which is an n-dimensional vector, is similar to a training sample, then it belongs to the training sample class [91]. In this case, the distance between the test sample and all of the other samples in the training set is calculated, and the sample is put into a class of the k-nearest samples, where k is an integer that shows how big the area around the test sample is.

The performance of KNN is influenced by the value of k and the proximity measure that is used. If the k value is set too low, the test sample will be influenced by the noisy points, and the test sample may suffer from an overfitting problem as a result. On the other hand, if the k number is excessively large, it is possible that the neighborhood may not accurately represent the true class. In this study, we examined k values of 1, 10, and 100 (i.e., fine, medium, and coarse neighborhoods) to investigate if the accuracy of the model was affected by the values of k. In addition, we employed a variety of different distance measurements to evaluate their overall effectiveness. One of the most essential difficulties to consider when using proximity-based classifiers is that they are sensitive to the dimensionality and range of the data values being considered. Therefore, we employed z-score normalization to reduce difficulties with proximity estimations, which we found to be particularly problematic. Figure 2 shows the KNN algorithm classifying a new sample.

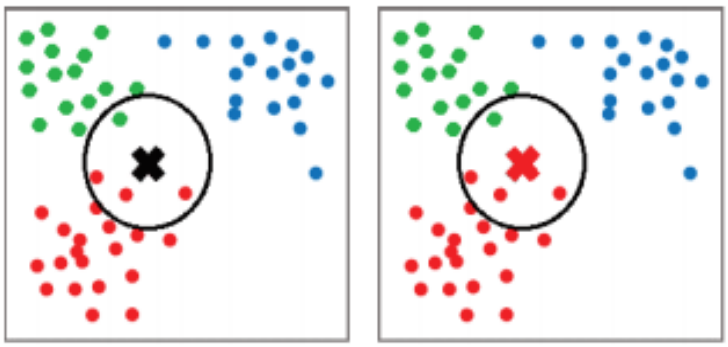

Figure 2. The KNN algorithm classifying a new sample

When predicting the classification of a new sample point, KNN uses a database in which the data points have been divided into multiple classes to make its predictions. There is no stated requirement for a training step in the algorithm. It is based on the resemblance of features. KNN selects the k points in the labeled data-set that most closely reflect the characteristics of each new sample point and determines the class for that data point based on which label appears more frequently in the k picked points from the training data for each new sample point. Using the KNN algorithm, the k nearest neighbors is determined by computing the distance between the test and training data points. In a similar way to Kmeans, the distance computations account for the majority of the calculations. Even though KNN doesn't need as many iterations as K-means, it doesn't need as many iterations to get to a result either. Once the k nearest neighbors is found, a conclusion can be made.

In this section, various aspects of the implementation of the algorithm with the Radoop framework are discussed in further detail. There are various phases involved in the parallelization process. When it came to executing the experiments, we employed three supercomputers: one for testing and tweaking the circumstances under which they were ultimately carried out, and one to run them on a virtual machine cluster. Because we are working with enormous amounts of data, it is possible that a dataset has a significant number of transactions. The suggested approach is implemented as a series of "map and reduce" operations. The algorithm demonstrates the parallel implementation of K-means in MapReduce for $K = 1$ and two features, although the WordCount technique is used as an example in the majority of MapReduce examples. $(Cx1, Cy1)$ and $(Cx2, Cy2)$ are the feature values for the centroids of the two clusters, respectively. $(Cx1, Cy1)$ and $(Cx2, Cy2)$ are the feature values for the centroids of the two clusters. During the map phase, the outputs for each data point (x, y) are represented as a key-value pair, with keys being either 1 or 2 and referring to the cluster to which the data point is most closely associated. The key and values are computed using (1).

$$Key = Argmin\ (1 \leq i \leq)(|x - Cxi|\ cyi)\ ł^{2}\ ) \quad (1)$$
$$Value = (x.y)$$

A further step involves shuffling and sorting the key-value pairs. The pairings with the same key are routed to reduction functions, where the total of values for both features (x and y) is determined for all of the data in each cluster using the reduced functions (key). Then, the output is divided by the number of data points in the sample to get the new centroid values.

```
Proposed Parallel K-Nearest Neighbor
Input: S (Dense array) Output: T (Reduced array)
Begin
        Run spark context (Slave)
        Master connection is being listened to
        Receive a dense array of data
        Check the length of the columns M
        Parallelize the data for processing;
                    N rows of data were gathered
                    do in parallel
Set the initial synaptic weights wij and thresholds j to tiny random values, such as [0, 1], and
 then repeat the procedure. Affect the learning rate parameter and the forgetting factor with
 modest positive values as well.
                    for each r in Ri do
                    Intilize KNN (r)with KNN (r .Si .K);
                    σ k (r.S) = maxσ ∈ KNN (r .Si .K)distr(r.8);

Calculate the output of the neuron at iteration T.
Make changes to the weights in the network.: wij(p + 1)=wij(p) + Δwij(p), // i, j=1, 2, ..., n
T should be sent as a reduction array.
Maintain a close connection
End
```

We focused on supervised classification algorithms created by Google, such as Naive Bayes, neural networks, k-nearest neighbors, and random forest. A summary of our experiments is shown in Table 1. The running times of the parallel generalized Hebbian algorithm (GHA) and parallel principal component analysis (PCA) on the identical hardware arrangement are presented in Table 2. Six unique, huge datasets were utilized to evaluate and compare the performance of the Apache Spark MLlib 2.0 package. Six datasets are available from the University of California, Irvine machine learning repository. The experimental architecture discussed in this article, in particular, consisted of a single Spark cluster that was developed in Java and used Apache Zeppelin 0.7.1 as an editor as well as an HDFS storage system. The Spark cluster is made up of several components, the most important of which are the master node, which is in charge of executing the driver application, and three worker nodes, which are in charge of data processing (including one worker node that runs on the master node). They all use the same software package and have the same

hardware setup (Intel® CoreTM i7-6700 CPU running at 3.40 GHz, 16 GB of RAM, 8 logical cores, and Windows 10 operating system) (see Table 2).

The three worker nodes were each allotted a total of 48 GB of RAM. In each worker node, four executors were installed, each of which had four gigabytes of RAM and two processor cores. A total of three executors were installed on each worker on the master node, each of which had five gigabytes of RAM and two CPUs. A total of 16 GB of accessible RAM was made available to the driver process. As previously indicated, the MLlib was executed using the Scala 2.11.8 programming language in a Spark 2.2.1 cluster with Hadoop 2.7.3 providing distributed storage, with the results being stored in Hadoop 2.7.3. By changing the amount of RAM available to the executors in each worker node while keeping as many data partitions as the architecture could handle, the execution time was made as short as possible. Several big classification datasets were used in this work, which were collected from the University of California, Irvine data repository. Table 1 presents the important properties of these datasets, including the number of records, attributes, and classes for each dataset, within each dataset.

Table 1. Datasets description

Data	No of record	No of attributes	No of classes
Covtype	581012	54	7
Covtype-2	581012	54	2
Higgs	11,000,000	28	2
Botnet Attacks	7,062,606	115	10
Dota2	102944	116	2
SUSY	5,000,000	18	2

Table 2. System description

Parameter	Description
Operating system	Windows10
CPU	Intel® Core™ i7-6700 CPU @ 3.40 GHz with 8 logical cores
Memory	16 GB
Number of workers	3
Computational framework	Apache Spark 2.2.1
compatible framework	Radoop
Distributed storage system	HDFS (Hadoop 2.7.3)
Editor for code development	Apache Zeppelin 0.7.1
Language used for coding	Scala 2.11.8

Table 3. Confusion matrices description

Matrices	Predicted positive	Predicted negative
Actual positive	True positive (TP)	False negative (FN)
Actual negative	False positive (FP)	True negative (TN)

4. RESULTS AND DISCUSSION

In this section, the results of the experiments that were conducted and the discoveries that were obtained after applying the indoor positioning k-nearest neighbor algorithm that was described in the methodology are broken down and discussed. The effectiveness of parallel KNN is evaluated for accuracy and speed in terms of data and model parallel KNN, as well as for the efficacy of parallel KNN for accuracy and speed in terms of parallel KNN. In Table 3, we can find a description of the confusion matrix.

4.1. Accuracy

According to the data presented in the table that came before it, the formula for determining how accurate a test is is $(TP + TN)/(TP + FP + FN + TN)$. This represents the total number of true positive and true negative instances as a percentage of the total number of cases. Figure 3 demonstrates how significantly more effective the enhanced parallel KNN algorithm is in comparison to its predecessor.

4.2. Recall

The recall (also known as sensitivity) of a genuine positive is the ratio of the number of true positives to the total number of true positives. In mathematical terms, the recall formula is: $TP/(TP + FN)$. Using this method, we can determine how well our model performs in terms of identifying the real, genuine outcome. Figure 4 shows the recall of the improved parallel KNN algorithm compared with the original algorithm.

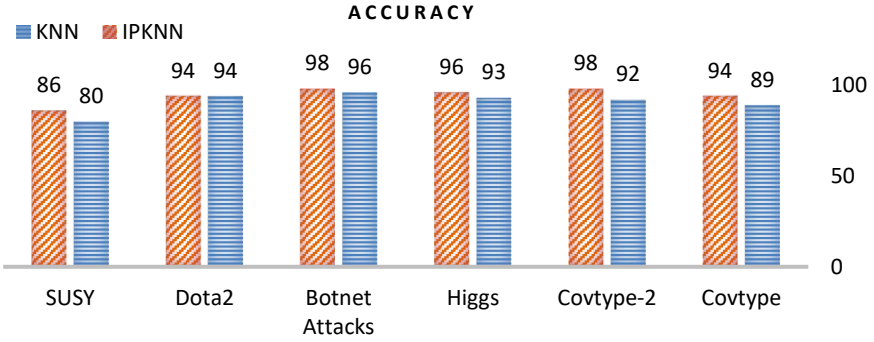

Figure 3. The accuracy of improved parallel KNN algorithm compared with original algorithm

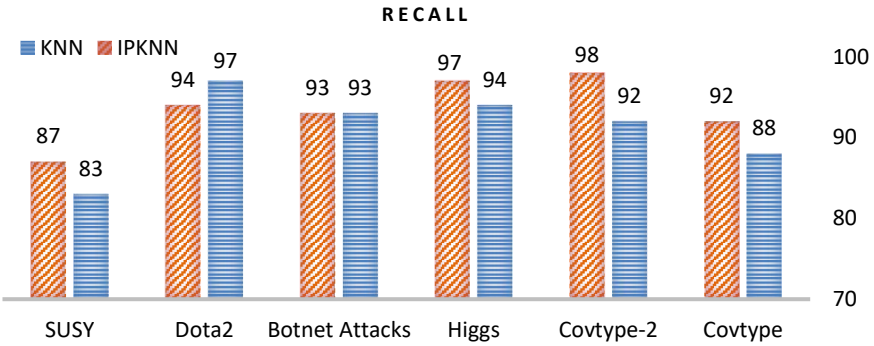

Figure 4. The recall of improved parallel KNN algorithm compared with original algorithm

For we utilize KNN to build the model, we can get the highest level of accuracy possible when forecasting absences. KNN has the highest success rate when aiming to ensure that the availability forecast is as accurate as possible. In this case, KNN may use similarity-based learning approaches to locate the not-so-easy-to-access dataset, or agent predictability. Once the system has found trends in the similarities of an employee's absences, it may predict when the individual will call in absent the following time. This forecast has significant restrictions, but it is generally accurate. When an employee calls in sick, the KNN uses the information to construct future instances of the employee. In this strategy, the call center will have enough data to construct a data set that will be utilized to estimate the predictability of each employee's agent predictability set. We compare it to the most fundamental generic techniques. For each approach, based on these comparisons, we can see the strength of our technique, which is more accurate than all of the classic approaches available and competitive with other powerful state-of-the-art methodologies. We approach our strategy in the same way we did previously.

4.3. Execution time

The default calculation method is serial calculation. Serial computation is characterized by the fact that each calculation pass is scheduled to be performed on a single processor. If the calculations are triggered by a calculation script, they are run sequentially in the order in which they occur in the calculation script, unless otherwise specified in (2). Parallel computing divides each calculation pass into a number of smaller sub-tasks. All of the sub-tasks that are capable of running independently of one another are scheduled to be executed simultaneously on up to 64 or 128 threads, depending on the configuration. Figure 5 shows the execution time comparison of serial KNN vs IPKNN.

$$S_{\text{Latency}} = \frac{L_{Serial}}{L_{Parallel}} \tag{2}$$

Across all data and model sets, the IPKNN implementation outperformed the single-node KNN implementation in all of our studies. Figure 5 shows that data parallel KNN is significantly faster than single-node KNN. The number of cores influences how fast the computer performs. This experiment was carried out on Internet cluster 1. The main purpose was to better understand the algorithm by determining how it behaves as the number of processing cores on the machines increased. There are a few aspects that must be considered

when assessing the data. As a result, the dataset is divided into exactly the same number of pieces as the total number of cores in the cluster, resulting in an overall result of n pieces. As a result of this configuration, we can ensure that every available core is always working on one or more bits of data from the dataset. This enables us to fully utilize the Spark configuration, keeping in mind that it is recommended that the dataset be broken into chunks at least equal in size to those accessible on the cluster's number of cores. The slave node could only have one executor for the two-core experiment, and each executor could only use 15 GB of RAM.

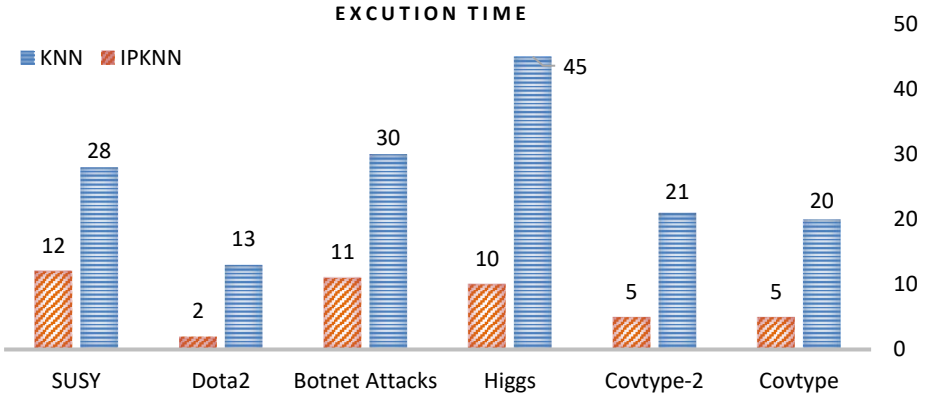

Figure 5. Execution time comparison of Serial KNN vs IPKNN

5. CONCLUSION

As a result, the necessity to develop new methods for processing huge datasets in order to reduce time costs prompted the creation of a data categorization strategy based on parallel partitions. Despite the increased number of clusters, separating data into batches aids in decreasing the classification algorithm's reaction time and associated processing costs. Parallel implementation of the proposed technique on a large number of nodes was found to be approximately twice as fast. This study compared the performance of the suggested technique on six datasets, and the results are reported in this report. This technique, which may be applied to big data analysis in medical, industrial, and business sectors, among others, can accomplish massive data categorization in a short period of time. Some of Python's parallel programming constraints are novel, as are some of the limits of other programming languages. In comparison to other programming languages, Python has fewer references to learn and fewer applications to implement. Experiments for the technique were also conducted on a single computer, which resulted in a number of system failures and difficulties with limiting the time of the recommended algorithm, among other issues. Future study could look into running the parallel algorithm in the cloud, as well as running the KNN in parallel on larger datasets. These are merely a few ideas for future research. In the future, the problem will be overcome by altering the number of centroids in each batch as the batch size changes. Furthermore, determining how to identify appropriate starting centroids, which is still a significant challenge in classification, will aid in increasing the accuracy of the parallel batch classification method, which is now not very accurate.

REFERENCES

[1] J. Maillo, I. Triguero, and F. Herrera, "A MapReduce-Based k-Nearest Neighbor Approach for Big Data Classification," in *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015, pp. 167-172, doi: 10.1109/trustcom.2015.577.

[2] M. El Bakry, S. Safwat, and O. Hegazy, "Big Data Classification using Fuzzy K-Nearest Neighbor," *International Journal of Computer Applications*, vol. 132, no. 10, pp. 8–13, 2015, doi: 10.5120/ijca2015907591.

[3] A. B. A. Hassanat, "Furthest-Pair-Based Binary Search Tree for Speeding Big Data Classification Using K-Nearest Neighbors," *Big Data*, vol. 6, no. 3, pp. 225–235, 2018, doi: 10.1089/big.2018.0064.

[4] A. Almomany, W. R. Ayyad, and A. Jarrah, "Optimized implementation of an improved KNN classification algorithm using Intel FPGA platform: Covid-19 case study," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, pp. 3815–3827, 2022, doi: 10.1016/j.jksuci.2022.04.006.

[5] S. -T. Nam, C. -Y. Jin, and S. -Y. Shin, "A forecasting of stock trading price using time series information based on big data," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2548–2554, 2021, doi: 10.11591/ijece.v11i3.pp2548-2554.

[6] S. Shetty, B. D. Rao, and S. Prabhu, "Growth of relational model: Interdependence and complementary to big data," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 2, pp. 1780–1795, 2021, doi: 10.11591/ijece.v11i2.pp1780-1795.

[7] Yatish HR *et al.*, "Massively scalable density based clustering (Dbscan) on the hpcc systems big data platform," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 1, pp. 207–214, 2021, doi: 10.11591/ijai.v10.i1.pp207-214.

[8] N. Seman and N. A. Razmi, "Machine learning-based technique for big data sentiments extraction," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 3, pp. 473–479, 2020, doi: 10.11591/ijai.v9.i3.pp473-479.

[9] N. P. -M. Esomonu, M. N. Esomonu, and L. I. Eleje, "Assessment big data in Nigeria: Identification, generation and processing in the opinion of the experts," *International Journal of Evaluation and Research in Education (IJERE)*, vol. 9, no. 2, pp. 345–351, 2020, doi: 10.11591/ijere.v9i2.20339.

[10] A. S. R. M. Sinaga, R. E. Putra, and A. S. Girsang, "Prediction measuring local coffee production and marketing relationships coffee with big data analysis support," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 5, pp. 2764–2772, 2022, doi: 10.11591/eei.v11i5.4082.

[11] H. K. Omar and A. K. Jumaa, "Distributed big data analysis using spark parallel data processing," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 3, pp. 1505–1515, 2022, doi: 10.11591/eei.v11i3.3187.

[12] S. Krishna and U. V. Murthy, "Evolutionary tree-based quasi identifier and federated gradient privacy preservations over big healthcare data," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 1, pp. 903–913, 2022, doi: 10.11591/ijece.v12i1.pp903-913.

[13] R. Karsi, M. Zaim, and J. El Alami, "Assessing naive bayes and support vector machine performance in sentiment classification on a big data platform," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 4, pp. 990–996, 2021, doi: 10.11591/IJAI.V10.I4.PP990-996.

[14] M. B. Mohammed and W. Al-Hameed, "New algorithm for clustering unlabeled big data," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 24, no. 2, pp. 1054–1062, 2021, doi: 10.11591/ijeecs.v24.i2.pp1054-1062.

[15] L. Rabhi, N. Falih, L. Afraites, and B. Bouikhalene, "Digital agriculture based on big data analytics: A focus on predictive irrigation for smart farming in Morocco," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 24, no. 1, pp. 581–589, 2021, doi: 10.11591/ijeecs.v24.i1.pp581-589.

[16] I. Sassi, S. Anter, and A. Bekkhoucha, "A spark-based parallel distributed posterior decoding algorithm for big data hidden markov models decoding problem," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 789–800, 2021, doi: 10.11591/ijai.v10.i3.pp789-800.

[17] T. Mouad, L. M. Driss, and K. Mustapha, "Big data traffic management in vehicular ad-hoc network," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 4, pp. 3483–3491, 2021, doi: 10.11591/ijece.v11i4.pp3483-3491.

[18] A. Rababah, "The best quintic chebyshev approximation of circular arcs of order ten," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 3779–3785, 2019, doi: 10.11591/ijece.v9i5.pp3779-3785.

[19] S. B. Lim, J. Woo, and G. Li, "Performance analysis of container-based networking Solutions for high-performance computing cloud," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 2, pp. 1507–1514, 2020, doi: 10.11591/ijece.v10i2.pp1507-1514.

[20] T. Hikida, H. Nishikawa, and H. Tomiyama, "Heuristic algorithms for dynamic scheduling of moldable tasks in multicore embedded systems," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 10, no. 3, pp. 157–167, 2021, doi: 10.11591/IJRES.V10.I3.PP157-167.

[21] H. Ramezani, M. Mohammadi, and A. S. Molahosseini, "An efficient look up table based approximate adder for field programmable gate array," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 1, pp. 144–151, 2022, doi: 10.11591/ijeecs.v25.i1.pp144-151.

[22] F. M. Saeed, S. M. Ali, and M. W. Al-Neama, "Notice of Retraction A parallel time series algorithm for searching similar sub-sequences," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 3, pp. 1652–1661, 2022, doi: 10.11591/ijeecs.v25.i3.pp1652-1661.

[23] S. S. Kolisetty and B. S. Rao, "Scalable epidemic message passing interface fault tolerance," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 2, pp. 998–1006, 2022, doi: 10.11591/eei.v11i2.3374.

[24] D. K. Siddaraju and R. Munibyrappa, "Design and performance analysis of efficient hybrid mode multi-ported memory modules on FPGA platform," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 11, no. 2, pp. 115–125, 2022, doi: 10.11591/ijres.v11.i2.pp115-125.

[25] N. K. Chebbah and M. Ouslim, "Medical diagnostic support system based on breast thermography using Raspberry Pi and cloud computing," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 28, no. 2, pp. 787–792, 2022, doi: 10.11591/ijeecs.v28.i2.pp787-792.

[26] M. M. Abed, U. Öztürk, and H. Khudhur, "Spectral CG Algorithm for Solving Fuzzy Non-linear Equations," *Iraqi Journal for Computer Science and Mathematics*, vol. 3, no. 1, pp. 1–10, 2022, doi: 10.52866/ijcsm.2022.01.01.001.

[27] V. Tsenev and M. Ivanova, "Statistical and machine learning approach for evaluation of control systems for automatic production lines," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 5, pp. 2527–2536, 2022, doi: 10.11591/eei.v11i5.3664.

[28] A. Ez-Zahout, H. Gueddah, A. Nasry, R. Madani, and F. Omary, "A hybrid big data movies recommendation model based k-nearest neighbors and matrix factorization," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 1, pp. 434–441, 2022, doi: 10.11591/ijeecs.v26.i1.pp434-441.

[29] W. Maharani and V. Effendy, "Big five personality prediction based in Indonesian tweets using machine learning methods," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 2, pp. 1973–1981, 2022, doi: 10.11591/ijece.v12i2.pp1973-1981.

[30] P. Shah, P. Swaminarayan, and M. Patel, "Sentiment analysis on film review in Gujarati language using machine learning," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 1, pp. 1030–1039, 2022, doi: 10.11591/ijece.v12i1.pp1030-1039.

[31] M. A. Muslim and Y. Dasril, "Company bankruptcy prediction framework based on the most influential features using XGBoost and stacking ensemble learning," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 6, pp. 5549–5557, 2021, doi: 10.11591/ijece.v11i6.pp5549-5557.

[32] E. Sutoyo and A. Almaarif, "Twitter sentiment analysis of the relocation of Indonesia's capital city," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 4, pp. 1620–1630, 2020, doi: 10.11591/eei.v9i4.2352.

[33] R. Sarno, D. R. Wijaya, and M. N. Mahardika, "Music fingerprinting based on bhattacharya distance for song and cover song recognition," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 2, pp. 1036–1044, 2019, doi: 10.11591/ijece.v9i2.pp1036-1044.

[34] Y. K. Liew, N. S. Ahmad, A. A. Aziz, and P. Goh, "Machine learning modeling of power delivery networks with varying decoupling capacitors," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 3, pp. 1049–1056, 2022, doi: 10.11591/ijai.v11.i3.pp1049-1056.

[35] I. D. Id, P. R. Sihombing, and S. Zakir, "Handling concept drifts and limited label problems using semi-supervised combine-merge gaussian mixture model," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 6, pp. 3361–3368, 2021, doi: 10.11591/eei.v10i6.3259.

[36] D. R. V. A. S. Kumar, C. S. Kumar, Ragamayi S., P. Sampath Kumar, K. S. Kumar, and S. H. Ahammad, "A test architecture design for SoCs using atam method," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 1, pp. 719–727, 2020, doi: 10.11591/ijece.v10i1.pp719-727.

[37] M. S. Farag, M. M. M. El Din, and H. A. El Shenbary, "Parking entrance control using license plate detection and recognition," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 15, no. 1, pp. 476–483, 2019, doi: 10.11591/ijeecs.v15.i1.pp476-483.

[38] P. P. Rokade, P. V. R. D. P. Rao, and A. K. Devarakonda, "Forecasting movie rating using k-nearest neighbor based collaborative filtering," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 6, pp. 6506–6512, 2022, doi: 10.11591/ijece.v12i6.pp6506-6512.

[39] M. Hassan *et al.*, "Sentiment analysis on Bangla conversation using machine learning approach," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 5, pp. 5562–5572, 2022, doi: 10.11591/ijece.v12i5.pp5562-5572.

[40] W. N. I. Al-Obaydy, H. A. Hashim, Y. A. Najm, and A. A. Jalal, "Document classification using term frequency-inverse document frequency and K-means clustering," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 27, no. 3, pp. 1517–1524, 2022, doi: 10.11591/ijeecs.v27.i3.pp1517-1524.

[41] A. S. Ahmed, Z. K. Obeas, B. A. Alhade, and R. A. Jaleel, "Improving prediction of plant disease using k-efficient clustering and classification algorithms," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 3, pp. 939–948, 2022, doi: 10.11591/ijai.v11.i3.pp939-948.

[42] C. Zhang, F. Li, and J. Jestes, "Efficient Parallel KNN Joins for Large Data in MapReduce," in *Proc. of the 15th International Conference on Extending Database Technology*, 2012, pp. 38–49, doi: 10.1145/2247596.2247602.

[43] N. Sismanis, N. Pitsianis, and X. Sun, "Parallel search of k-nearest neighbors with synchronous operations," in *2012 IEEE Conference on High Performance Extreme Computing*, 2012, doi: 10.1109/HPEC.2012.6408667.

[44] N. Rajani, K. McArdle, and I. S. Dhillon, "Parallel k nearest neighbor graph construction using tree-based data structures," in *1st High Performance Graph Mining workshop*, 2015, doi: 10.5821/hpgm15.1.

[45] Y. Zhang, Y. Liao, Q. Tang, J. Lin, and P. Huang, "Biomimetic Nanoemulsion for Synergistic Photodynamic-Immunotherapy Against Hypoxic Breast Tumor," *Angewandte Chemie International Edition*, vol. 60, no. 19, pp. 10647–10653, 2021, doi: 10.1002/anie.202015590.

[46] M. I. Hameed and B. N. Shihab, "On Differential Subordination and Superordination for Univalent Function Involving New Operator," *Iraqi Journal For Computer Science and Mathematics*, vol. 3, no. 1, pp. 22–31, 2022, doi: 10.52866/ijcsm.2022.01.01.003.

[47] S. Kunnakorntammanop, N. Thepwuttisathaphon, and S. Thaicharoen, "An Experience Report on Building a Big Data Analytics Framework Using Cloudera CDH and RapidMiner Radoop with a Cluster of Commodity Computers," in *International Conference on Soft Computing in Data Science (SCDS 2019)*, 2019, pp. 208–222, doi: 10.1007/978-981-15-0399-3_17.

[48] J. Venner, "Pro Hadoop," Apress, 2009, doi: 10.1007/978-1-4302-1943-9.

[49] G. singh Bhathal and A. S. Dhiman, "Big Data Solution: Improvised Distributions Framework of Hadoop," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2018, doi: 10.1109/iccons.2018.8663142.

[50] T. Zheng *et al.*, "Compositionally Graded KNN-Based Multilayer Composite with Excellent Piezoelectric Temperature Stability," *Advanced Materials*, vol. 34, no. 8, 2022, doi: 10.1002/adma.202109175.

[51] Z Prekopcs´ak, G Makrai, T. Henk, and C. G.-Papanek, "Radoop: Analyzing Big Data with RapidMiner and Hadoop," *Proc. 2nd RapidMiner Community Meet. Conf*. 2011. [Online]. Available: http://prekopcsak.hu/papers/preko-2011-rcomm.pdf

[52] R. R. Almassar and A. S. Girsang, "Detection of traffic congestion based on twitter using convolutional neural network model," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 4, pp. 1448–1459, 2022, doi: 10.11591/ijai.v11.i4.pp1448-1459.

[53] N. M. Samsudin, C. F. B. M. Foozy, N. Alias, P. Shamala, N. F. Othman, and W. I. S. W. Din, "Youtube spam detection framework using naïve bayes and logistic regression," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14, no. 3, pp. 1508–1517, 2019, doi: 10.11591/ijeecs.v14.i3.pp1508-1517.

[54] J. Lin *et al.*, "Ultrahigh energy harvesting properties in temperature-insensitive eco-friendly high-performance KNN-based textured ceramics," *Journal of Materials Chemistry A*, vol. 10, no. 14, pp. 7978–7988, 2022, doi: 10.1039/d2ta00203e.

[55] A. Bhaskar and R. Ranjan, "Optimized memory model for hadoop map reduce framework," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 4396–4407, 2019, doi: 10.11591/ijece.v9i5.pp4396-4407.

[56] M. B. Masadeh, M. S. Azmi, and S. S. S. Ahmad, "Available techniques in hadoop small file issue," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 2, pp. 2097–2101, 2020, doi: 10.11591/ijece.v10i2.pp2097-2101.

[57] J. S. Kumar, B. K. Raghavendra, S. Raghavendra, and Meenakshi, "Performance evaluation of Map-reduce jar pig hive and spark with machine learning using big data," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 4, pp. 3811–3818, 2020, doi: 10.11591/ijece.v10i4.pp3811-3818.

[58] Y. Choubik, A. Mahmoudi, M. M. Himmi, and L. El Moudnib, "STA/LTA trigger algorithm implementation on a seismological dataset using hadoop mapreduce," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 2, pp. 269–275, 2020, doi: 10.11591/ijai.v9.i2.pp269-275.

[59] A. H. Katrawi, R. Abdullah, M. Anbar, and A. K. Abasi, "Earlier stage for straggler detection and handling using combined CPU test and LATE methodology," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 5, pp. 4910–4917, 2020, doi: 10.11591/ijece.v10i5.pp4910-4917.

[60] Q. A. Abed, O. M. Fadhil, and W. L. Al-Yaseen, "Data mining in web personalization using the blended deep learning model," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 3, pp. 1507–1512, 2020, doi: 10.11591/ijeecs.v20.i3.pp1507-1512.

[61] S. W. Kareem, R. Z. Yousif, and S. M. J. Abdalwahid, "An approach for enhancing data confidentiality in hadoop," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 3, pp. 1547–1555, 2020, doi: 10.11591/ijeecs.v20.i3.pp1547-1555.

[62] A. H. Katrawi, R. Abdullah, M. Anbar, I. AlShourbaji, and A. K. Abasi, "Straggler handling approaches in mapreduce framework: A comparative study," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 1, pp. 375–382, 2021, doi: 10.11591/ijece.v11i1.pp375-382.

[63] M. Sornalakshmi *et al.*, "An efficient apriori algorithm for frequent pattern mining using mapreduce in healthcare data," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 1, pp. 390–403, 2021, doi: 10.11591/eei.v10i1.2096.

[64] B.-E. B. Semlali and C. El Amrani, "Big data and remote sensing: A new software of ingestion," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 2, pp. 1521–1530, 2021, doi: 10.11591/ijece.v11i2.pp1521-1530.

[65] H. Arslan and H. Arslan, "A new COVID-19 detection method from human genome sequences using CpG island features and KNN classifier," *Engineering Science and Technology, an International Journal*, vol. 24, no. 4, pp. 839–847, 2021,

doi: 10.1016/j.jestch.2020.12.026.
[66] R. Arief, A. B. Mutiara, T. M. Kusuma, and Hustinawaty, "Automated hierarchical classification of scanned documents using convolutional neural network and regular expression," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 1, pp. 1018–1029, 2022, doi: 10.11591/ijece.v12i1.pp1018-1029.
[67] L. Ammanna, Jagadeeshgowda, and J. Pujari, "An efficient and robust parallel scheduler for bioinformatics applications in a public cloud: A bigdata approach," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, no. 2, pp. 1078–1086, 2022, doi: 10.11591/ijeecs.v25.i2.pp1078-1086.
[68] A. H. Ali, R. A. I. Alhayali, M. A. Mohammed, and T. Sutikno, "An effective classification approach for big data with parallel generalized hebbian algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 6, pp. 3393–3402, 2021, doi: 10.11591/eei.v10i6.3135.
[69] J. García and C. Maureira, "A KNN quantum cuckoo search algorithm applied to the multidimensional knapsack problem," *Applied Soft Computing*, vol. 102, 2021, doi: 10.1016/j.asoc.2020.107077.
[70] A. Y. Mjhool, A. H. Alhilali, and S. Al-Augby, "A proposed architecture of big educational data using hadoop at the University of Kufa," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 6, pp. 4970–4978, 2019, doi: 10.11591/ijece.v9i6.pp4970-4978.
[71] P. Ganesh, K. S. Kumar, D. E. Geetha, and T. V S. Kumar, "Performance evaluation of cloud service with hadoop for twitter data," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13, no. 1, pp. 392–404, 2019, doi: 10.11591/ijeecs.v13.i1.pp392-404.
[72] S. Wilson and Sivakumar R., "Twitter data analysis using hadoop ecosystems and apache zeppelin," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, no. 3, pp. 1490–1498, 2019, doi: 10.11591/ijeecs.v16.i3.pp1490-1498.
[73] U. Narayanan, V. Paul, and S. Joseph, "A light weight encryption over big data in information stockpiling on cloud," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 1, pp. 389–397, 2019, doi: 10.11591/ijeecs.v17.i1.pp389-397.
[74] T. R. Shaha, M. N. Akhtar, F. T. Johora, M. Z. Hossain, M. Rahman, and R. B. Ahmad, "A noble approach to develop dynamically scalable namenode in hadoop distributed file system using secondary storage," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13, no. 2, pp. 729–736, 2019, doi: 10.11591/ijeecs.v13.i2.pp729-736.
[75] M. A. Rahman, A. Hossen, J. Hossen, C. Venkataseshaiah, T. Bhuvaneswari, and A. Sultana, "Towards machine learning-based self-tuning of hadoop-spark system," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 15, no. 2, pp. 1076–1085, 2019, doi: 10.11591/ijeecs.v15.i2.pp1076-1085.
[76] K. Kalia and N. Gupta, "Analysis of hadoop MapReduce scheduling in heterogeneous environment," *Ain Shams Engineering Journal*, vol. 12, no. 1, pp. 1101–1110, 2021, doi: 10.1016/j.asej.2020.06.009.
[77] S. Tangwannawit and P. Tangwannawit, "An optimization clustering and classification based on artificial intelligence approach for internet of things in agriculture," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 1, pp. 201–209, 2022, doi: 10.11591/ijai.v11.i1.pp201-209.
[78] R. A. Hasan, H. W. Abdulwahid, and A. S. Abdalzahra, "Using Ideal Time Horizon for Energy Cost Determination," *Iraqi Journal For Computer Science and Mathematics*, vol. 2, no. 1, pp. 9–13, 2021, doi: 10.52866/ijcsm.2021.02.01.002.
[79] A. AlDabbas and Z. Gal, "Cassini-Huygens mission images classification framework by deep learning advanced approach," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 3. pp. 2457–2466, 2021, doi: 10.11591/ijece.v11i3.pp2457-2466.
[80] E. M. Marouane and Z. Elhoussaine, "A fuzzy neighborhood rough set method for anomaly detection in large scale data," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 1, pp. 1–10, 2020, doi: 10.11591/ijai.v9.i1.pp1-10.
[81] A. Nakkiran and V. Thulasiraman, "Elastic net feature selected multivariate discriminant mapreduce classification," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 1, pp. 587–596, 2022, doi: 10.11591/ijeecs.v26.i1.pp587-596.
[82] N. M. Hussien, Y. M. Mohialden, N. T. Ahmed, M. A. Mohammed, and T. Sutikno, "A smart gas leakage monitoring system for use in hospitals," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 2, pp. 1048-1054, 2020, doi: 10.11591/ijeecs.v19.i2.pp1048-1054.
[83] K. Aggarwal *et al.*, "Has the Future Started? The Current Growth of Artificial Intelligence, Machine Learning, and Deep Learning," *Iraqi Journal For Computer Science and Mathematics*, vol. 3, no. 1, pp. 115–123, 2022, doi: 10.52866/ijcsm.2022.01.01.013.
[84] A. H. Ali, M. N. Abbod, M. K. Khaleel, M. A. Mohammed, and T. Sutikno, "Large scale data analysis using MLlib," *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 19, no. 5, pp. 1735-1746, 2021, doi: 10.12928/telkomnika.v19i5.21059.
[85] R. A. I. Alhayali, M. Aljanabi, A. H. Ali, M. A. Mohammed, and T. Sutikno, "Optimized machine learning algorithm for intrusion detection," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 24, no. 1, pp. 590–599, 2021, doi: 10.11591/ijeecs.v24.i1.pp590-599.
[86] L. Abualigah and B. Al Masri, "Advances in MapReduce Big Data Processing: Platform, Tools, and Algorithms," *Studies in Big Data*, pp. 105–128, 2021, doi: 10.1007/978-981-33-6400-4_6.
[87] A. Shokrzade, M. Ramezani, F. A. Tab, and M. A. Mohammad, "A novel extreme learning machine based kNN classification method for dealing with big data," *Expert Systems with Applications*, vol. 183, 2021, doi: 10.1016/j.eswa.2021.115293.
[88] P. Velentzas, M. Vassilakopoulos, and A. Corral, "GPU-Based Algorithms for Processing the k Nearest-Neighbor Query on Disk-Resident Data," in *International Conference on Model and Data Engineering (MEDI 2021)*, 2021, pp. 264–278, doi: 10.1007/978-3-030-78428-7_21.
[89] A. H. Ali, Z. F. Hussain, and S. N. Abd, "Big data classification efficiency based on linear discriminant analysis," *Iraqi Journal For Computer Science and Mathematics*, vol. 1, no. 1, pp. 7–12, 2020.
[90] Y. M. Mohialden, N. M. Hussien, Q. A. Z. Jabbar, M. A. Mohammed, and T. Sutikno, "An internet of things-based medication validity monitoring system," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 2, pp. 932-938, 2022, doi: 10.11591/ijeecs.v26.i2.pp932-938.
[91] M. I. Al-Mashhadani, K. M. Hussein, E. T. Khudir, and M. Ilyas, "Sentiment Analysis using Optimized Feature Sets in Different Facebook/Twitter Dataset Domains using Big Data," *Iraqi Journal For Computer Science and Mathematics*, vol. 3, no. 1, pp. 64–70, 2022, doi: 10.52866/ijcsm.2022.01.01.007.

BIOGRAPHIES OF AUTHORS

Ahmed Hussien Ali was born in Baghdad, Iraq, in 1988. He received the B.Sc. degree in Computer Science from the University of Al-Mustansiriyah, Iraq, in 2010, and the M.Sc. degree from BAMU University, India. He got Ph.D. from ICCI, Informatics Institute for Postgraduate Studies, Baghdad, Iraq and Faculty member, Computer Science Department, College of Education, Al-Iraqia University, Adhamyia, Baghdad, Iraq. He can be contacted at email: msc.ahmed.h.ali@gmail.com.

Mostafa Abdulghafoor Mohammed currently works at the Al-Imam AlAadham University college . Mostafa does research in Information technology , Computer Communications (Networks) , Cloud Computing and Communication Engineering . Their current project is 'offloading in mobile cloud computing'. He has finish his master from computer science department at BAMU university. India, and he finish Ph.D in Computer science and IT at University polytechnic of Bucharest, Romania. He can be contacted at email: alqaisy86@gmail.com.

Raed Abdulkareem Hasan currently works at the Al-Hawija Technical Institute, Northern Technical University. Raed does research in information systems (business informatics), computer communications (networks) and communication engineering. Their current project is 'offloading in mobile cloud computing'. He has finished his master from Computer Science Department at BAMU University, India. He can be contacted at email: raed.isc.sa@gmail.com.

Maan Nawaf Abbod 5th of January 1983 Baghdad BSc 2006 University of Baghdad\ Ibn Alhaitham College MSc IT 2011 DeMontfort University UK Lecturer in Al-Imam Al-Adham university college 2012 Assistant of Arabic Head department 2014-2018 Head of computer department 2014 Head of registration department 2018. Did researchs in (IT, Big Data, AI, Network, PLC). He can be contacted at email: alanimm2@gmail.com.

Mohammed Sh. Ahmed was born in Salahuddin, Iraq, in 1972. He received the B.Sc. and M. Sc. degrees in Computer Engineering from the University of Technology, Iraq, in 1995 and 2003, respectively. He got Ph.D. in the field of Networks communication from Newcastle University, UK. Currently, he working as a Dean of the College of Petroleum Processes Engineering, Tikrit University. He can be contacted at email: mohammed.shwash@tu.edu.iq.

Tole Sutikno is currently employed as a lecturer in the Electrical Engineering Department at Universitas Ahmad Dahlan (UAD), which is located in Yogyakarta, Indonesia. In 1999, 2004, and 2016, he graduated with a Bachelor of Engineering from Universitas Diponegoro, a Master of Engineering from Universitas Gadjah Mada, and a Doctor of Philosophy in Electrical Engineering from Universiti Teknologi Malaysia. All three degrees are in the field of electrical engineering. Since the year 2008, he has held the position of Associate Professor at the University of Ahmad Dahlan in Yogyakarta, Indonesia. His research interests include the areas of digital design, industrial applications, industrial electronics, industrial informatics, power electronics, motor drives, renewable energy, FPGA applications, embedded systems, artificial intelligence, intelligent control, digital libraries, and intelligent control. He can be contacted at email: tole@te.uad.ac.id.