TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 3, June 2021, pp. 872~885
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i3.18159

Parallel classification and optimization of telco trouble ticket dataset

Fauzy Che Yayah, Khairil Imran Ghauth, Choo-Yee Ting
Multimedia University, Faculty of Computing and Informatics, Cyberjaya, Malaysia

Article Info

Article history:

Received Jul 16, 2020
Revised Sep 9, 2020
Accepted Sep 20, 2020

Keywords:

Classification
Hadoop
Optimization
Spark
Trouble ticket

ABSTRACT

In the big data age, extracting applicable information using traditional machine learning methodology is very challenging. This problem emerges from the restricted design of existing traditional machine learning algorithms, which do not entirely support large datasets and distributed processing. The large volume of data nowadays demands an efficient method of building machine-learning classifiers to classify big data. New research is proposed to solve problems by converting traditional machine learning classification into a parallel capable. Apache Spark is recommended as the primary data processing framework for the research activities. The dataset used in this research is related to the telco trouble ticket, identified as one of the large volume datasets. The study aims to solve the data classification problem in a single machine using traditional classifiers such as W-J48. The proposed solution is to enable a conventional classifier to execute the classification method using big data platforms such as Hadoop. This study's significant contribution is the output matrix evaluation, such as accuracy and computational time taken from both ways resulting from hyper-parameter tuning and improvement of W-J48 classification accuracy for the telco trouble ticket dataset. Additional optimization and estimation techniques have been incorporated into the study, such as grid search and cross-validation method, which significantly improves classification accuracy by 22.62% and reduces the classification time by 21.1% in parallel execution inside the big data environment.

Corresponding Author:

Fauzy Che Yayah
Faculty of Computing and Informatics
Multimedia University
63000, Cyberjaya, Malaysia
Email: akunyer@gmail.com

1. INTRODUCTION

Due to the growing data volume growth, and the advancement of computational system complexity, the need for fast machine learning has increased tremendously. Like telco, several businesses rely heavily on how much data they have monetized to survive relative to the situation some decades ago. Through that, importance now allows the critical task of extracting valuable knowledge from the information efficiently, so big data technology [1] was created. Big data is known for its 3(V) characteristics, including volume, velocity, and variety. The volume is interpreted as the dimension of data being measured. Velocity is described as how fast data is generated, and variety is the structure of data that can be structured or unstructured. Nowadays, the importance of machine learning is making it easier to execute and learn faster and more efficiently with less human intervention. There are a few hurdles ahead for the rapid implementation of machine learning. With the advent of the big data age, data collection is becoming very complicated for the high volume data set process. For example, the traditional machine learning technique [2] was designed to load all data into the memory, making data processing sometimes impractical for a regular single machine processing. This problem can be solved by applying the machine learning algorithm of the big data platform. For this study, the experimental dataset is based on the original trouble tickets [3] dataset from Malaysia's leading telco, consolidated in Hadoop since 2012.

This research aims to identify or predict the trouble ticket resolution code based on the symptom error code and other relevant information embedded in the consolidation dataset. Classification process problems occur with the machine learning approach, requiring large loading volumes of records and high dimensions of the trouble ticket dataset. The required classification accuracy results are hard to achieve due to computational constraints on a single computer. Data volume and dimensional data are two machine learning problems when implementing traditional classifiers. In this study, the classifier is selected based on the ability to interpret simple outcomes. The suggested algorithm selection relates to tree-based, rule-based [4], and Bayesian network [5]. Sixteen algorithms have been evaluated, and six of them meet the research criteria, which are PART, random tree [6], random forest [7], W-J48 [8], functional tree (FT), and Bayesian network [9]. The remaining algorithms are rejected due to lower accuracy results, highly resource-intensive computing, and model explainable format quality. Each algorithm has its parameter for precise tuning. For example, the classifiers used in this research, W-J48 (J48 Weka), has three mandatory parameters: C, M, and N. The default value of C (pruning confidence threshold) is 0.25, M (minimum instance number per leaf) is 2.0 and N (pruning numbers) is 3.

These parameters will change for further optimization based on the quality of the trained data set. Incorporating it into a limited memory computing environment, such as a single computer, is not advisable, leading to computational errors. The research findings suggested that solutions evolve from the traditional classification approach to parallel capability with hyper-parameter tuning. This new design machine learning approach is intended to ensure higher classification accuracy and faster learning. The suggested solution is using Radoop SparkRM [10] to improve computational strategy and transform the existing traditional machine learning technique is efficient with working within the big data environment. This significant contribution to the study reduces the processing time and increases the telco trouble ticket dataset's classification accuracy. The next section will show the study's literature, the problem description,the proposed approach, and the experimental setup. The final selection will address the final results and conclusion and this study's future progress.

Through the analysis of the literature, the research identified many approaches to addressing the classification issues. One of the optimal classifiers optimal for the telco trouble ticket dataset is applying with the W-J48 classifier. The dataset that works with the W-J48 classifier does not need to normalize and works correctly with the missing value and is easy to interpret by observing the model output. The classifier can also be configured for optimum accuracy by selecting the most optimal parameters, but there is a need for massive computation with more significant effort. The paper tackled this issue by proposing a transformation from the traditional classification system to big data classification using Hadoop, Radoop, and Apache Spark [11].

2. RELATED WORK

A few researchers have suggested strategies for optimizing the classification process on the traditional algorithm. Much of the initiatives relate to optimization based on improvements to the classifier's original formula and the features engineering to achieve maximum accuracy. A parallel approach to improving the optimization process and overcoming the large data set's dimensionality problems is applied in some studies. Chen [12] implemented a new classification method using softmax, which used the MapReduce. Softmax's key challenge is restricting the processing of massive data sets, which raises computing time and can be inefficient. One goal is to enhance the existing conventional softmax algorithm into parallel functionality based on the MapReduce framework. Implementing K-fold cross-validation [13] improves the accuracy of the model estimation and prevents overfitting. In a distributed system like Hadoop, the concluding model demonstrates enhanced overall performance and faster computation.

Du and Li [14] use the K-nearest neighbors (KNN) algorithm in MapReduce to achieve parallelism on the Hadoop network. The goal of this is to speed up the classification process and reduce computation time. The dataset in this study is about classifying text that is important to the public network's opinions. The experiment successfully converted the classical KNN into a MapReduce program. The result also improved classification performance with lower computation time and better classification accuracy when processing a larger dataset.

Nie *et al.* [15] suggested an approach to hybrid vector support to improve classification accuracy and reduce training time. This approach can be generated by transforming the classical support vector machine (SVM) into a MapReduce-based method that allows for parallel Hadoop computing. It is composed of a single master node and seven slave nodes. The dataset of this experiment has 123-dimensional features to solve two problems in classification. The original SVM kernel was modified with a linear, radial basis function (RBF), and polynomial mixed kernel. The experiment's test shows that the current hybrid SVM model significantly increases accuracy and computational time.

Shah and Patel [16] proposed a new approach to diabetes datasets being categorized in a distributed environment. The suggested method is to apply the process of classification within Hadoop using Spark. The study's main reason is to analyze the output results based on critical parameters such as accuracy, recall, and precision. The dataset's missing value was also rectified to increase the overall accuracy. The missing value is treated by imputing a mean and average estimate of the values. The experiment value outcome is based on applying multiple algorithms with missing value effects through the imputation process and without missing value.

Fang Fang Yuan [17] proposed a new approach to optimizing the decision-tree algorithm based on MapReduce. An optimized genetic algorithm (GA) [18] was implemented based on the decision tree algorithm, allowing it to be permitted in parallel mode. This new method will attempt to find the optimal global set of rules for the decision tree classifier. It will transform the current ruleset for the decision tree into a GA chromosome set and then use the fitness function to assess the individual chromosome's fitness by performing selection, crossover, and mutation. The final step is to decide the reasonableness of the currently transformed chromosome to apply the rules of increment, consolidation, or deletion when necessary. Since the GA method is an iterative operation, it can be performed in Hadoop simultaneously. It can finally independently and ready for parallel computational reconstruction of the decision tree structure by applying the reduction and the combination of the MapReduce matrix function.

Dai and Ji [19] utilizes the MapReduce framework to implement a new C4.5 decision tree algorithm method. The aim is to solve the problems when the decision tree classifier deals with a large dataset, which is very time-consuming. When data does not fit in the memory, some calculation needs to be performed to the external device, increasing the I/O cost. Therefore, the current decision tree C4.5 algorithm was transformed into the MapReduce version, which included mapping and reducing operation. Some experimental was conducted extensively with various large datasets and some data structure modifications to reduce computational and communication costs. The outcome of the MapReduce version of the C4.5 decision tree classifier demonstrates the computation time efficiency and can also be scalable to accommodate any dataset needs.

In summary, most of the current study implementing MapReduce for the traditional classifier based on the algorithm design structure and features. The requirement is that the selected classifier process must be able to split into a separate process. The cost of changing the existing classifier code into MapReduce based, however, is not a straightforward process and requires in-depth knowledge of how the algorithms work. Conversion to MapReduce requires information that part of the algorithms can be evaluated in parallel, and the criteria are selecting the key-pair value and the split ratio rate. Most previous studies that did not incorporate Cross-Validation is very important for resolving the overfitting issues and ensuring the best estimate of model results. The proposed solution addresses the previous limitation by applying alternative data processing frameworks, such as Apache Spark, which is more flexible and faster developing by most programming languages. Table 1 summarises a comparative study from the existing methods.

Table 1. A comparative study from the existing method

Approach	Framework	Algorithm	Enable Parameter Tuning	Enable Cross-Validation	Observation and limitation
Z. Chen and J. Cheng [12]	MapReduce	Softmax	No	Yes	a) Achieved parallelism for the selected classifier b) Manual parameter setting for each cycle. c) Custom development involved to convert to MapReduce compatible. d) Improved computational time and classification accuracy.
S. Du and J. Li [14]	MapReduce	K-Nearest Neighbors (KNN)	No	No	a) No parameter tuning was involved. b) Achieved parallelism for the classifier. c) Improved computational time and classification accuracy.
W. Nie, B. Fan, X. Kong [15]	MapReduce	Support vector machine (SVM)	No	Yes	a) Static parameter for each SVM kernel. b) It requires custom development, limiting the type of supported classifier, which may take a longer time for system development. c) Improved computational time and classification accuracy. d) The cluster memory is too low to execute for the MapReduce job (4GB)

Table 1. A comparative study from the existing method (continue)

Approach	Framework	Algorithm	Enable Parameter Tuning	Enable Cross-Validation	Observation and limitation
J. Shah and R. Patel [16]	Spark	Naïve Bayes, Bayesian Network, SVM, Neural Network	No	No	a) Only utilize the builtin Spark algorithms. b) No parameter tuning was involved. c) Achieved parallelism in the study. d) Naïve Bayes produces the best accuracy.
F. Yuan, F. Lian, X. Xu, and Z. Ji [17]	MapReduce	Decision Tree	No	No	a) No parameter tuning was involved. b) Achieved parallelism in the study. c) Only improved computational time.
W. Dai and W. Ji [19]	MapReduce	Decision Tree	No	No	a) No parameter tuning was involved. b) Achieved parallelism in the study. c) Improved computational time and classification accuracy. d) The cluster memory is too low to execute for the MapReduce job (1GB)

3. RADOOP AND W-J48 CLASSIFIER

3.1. Overview of Radoop

Rapidminer explicitly offers advanced data science tools [20] for machine learning, pre-processing data, and predictive analytics. Besides Rapiminer, there is a function called Radoop that was designed to integrate big data environments like Hadoop. As illustrated in Figure 1, Radoop can interact with Hadoop through a graphical user interface running Hive [21] for massive data processing. Hive is part of the big data ecosystem, which converts SQL syntax queries into MapReduce data processing jobs stored in the Hadoop distributed file system (HDFS) [22].

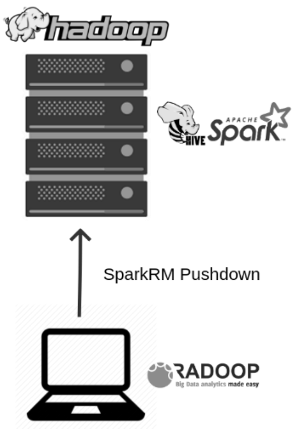


Figure 1. Radoop integration architecture

Radoop is an extension developed by Rapidminer to reduce the complexity when Hadoop is interfacing. Radoop converts and manages interaction within the Radoop Nest with the Hadoop cluster to execute the MapReduce job via SparkRM. Radoop Nest is the essential meta-operator that defines the Hadoop cluster connection settings. Any subprocess of Radoop operators inside the Radoop Nest determines the process that operates on that Hadoop cluster. Radoop Nest imports the data from the client's operative memory into the Hadoop cluster. SparkRM enables parallel process pushdown (translate) from the regular operation within Radoop Nest to parallel execution. Partitioning is the principal principle of the SparkRM. When running SparkRM, the data object is divided into multiple settings-based partitions. For each partition, the parallel execution [23] process is distributed through all cluster nodes. The supporting SparkRM partitioning method options are described in the following Table 2.

The justification for choosing RapidMiner over other data science tools is made by considering the simplification features bridging with less complexity between Hadoop and Apache Spark. Another deliberation is that RapidMiner also has an extensive machine learning algorithm library with other excellent tools such as built-in data transformation, model deployment, and optimization. Rapidminer is also listed as the leader in the Gartner Magic Quadrant of data science and machine learning platforms for the sixth consecutive year since 2013. The RapidMiner Radoop extension will automatically convert the Spark or Hive's internal execution code for further execution. Other essential features include the pushdown features that allow any components within RapidMiner to be auto-translated into the Spark binary code, thereby enhancing Hadoop's parallel performance and system integration. Table 3 demonstrates the similarities between Radoop, Hadoop, MapReduce, and Spark in tabular form. It allows for a description of the different components that are important to this analysis. The table also shows that Radoop is the only application with Hadoop focused on data science integration tools. The end-user has a graphical interface that is easy to use with all of the features. It only needs some information to be put in, and all is taken care of by the application. Apache Spark is the only application framework that only supports caching functionality is significantly improved to avoid any computational errors in the Spark's resilient distributed dataset (RDD) files. Most components were built in JAVA, except for Apache Spark, developed in Scala, which provided performance advantages and integrated with other languages. Therefore, it is necessary to conclude using Radoop and Apache Spark in this study.

Table 2. SparkRM partition method options

Partition Method	Description
Linear	It determines the number of partitions based on the size of the files. Depending on the size of the HDFS block size (default of 128 MB) of the Hadoop cluster, one partition size is assigned. This option is the fastest partitioning mode, reducing data movement, and where the information is processed.
Random	This approach can be used if the study prefers to know the number of optimal partitions and no control over the Hadoop cluster distribution. This can also produce higher overheads than linear partition mode on condition.
Attribute	This function helps one to define by manual option the distribution between the data and partition. The data are grouped according to the partition attributes, and each partition has to handle one particular attribute. It can also create significant network traffic and lead to unequal data distribution. Each partition may contain more data than the others.

Table 3. Comparison table of Radoop, Hive, Hadoop, MapReduce, and Spark

Components / Factors	Type	Speed	Language	Data Processing	Caching Capability	Integration Complexity
Rapidminer Radoop	Data Science Integration Tools Extension	Fast	Java	In-Memory (client)	No	Easy
Apache Hive	Data summarization on the top of Apache Hadoop	Medium	Java	Batch	No	Medium
Apache Hadoop	Distributed Storage Framework	Medium	Java	Batch	No	Hard
MapReduce	Data Processing Framework	Slow	Java	Batch	No	Hard
Apache Spark	Data Processing Framework	Fast	Scala	Batch and Streaming	Yes	Medium

3.2. Overview of W-J48 (Weka-J48) decision tree classifier

W-J48 is the statistical classifier used to create a decision tree that can be used to classify data. The additional W-J48 function is handling missing values, pruning the decision tree, promoting a continuous set of conditions, deriving rules, and reducing the classification of errors. The origin of this algorithm is the original design of the ID3 algorithm (iterative Dichotomiser 3). The theory of applying the W-J48 decision tree is based on the ratio of knowledge gain by calculating the amount of information gathered from the data. This primarily focuses on the basic features of attributes in the dataset. The criteria for splitting the decision trees are based on which attributes have more data. Visualization of the decision tree is essential to show all possible outcomes of a decision and each path's traces to the end. The W-J48 classifier operation is based on the benefit of knowledge and the reduction of entropy. The information benefit is that entropy by portioning behaviours based on attribute value, and the arbitrary sampling set's content determines the entropy. By adding more details to the dataset, complexity decreases. When modified, information gain correlates with prior information. Information gain (S, A) is defined as the following:

$$\text{Gain }(S, A) = \text{Entropy}(S) - \sum_{\text{vevalue}(A)} \frac{|S_V|}{|S|} \text{Entropy} \quad (1)$$

S is the original set's entropy value, and the next term is the expected entropy value. The typical entropy is the summation of entropy belonging to subset *S*. Thus, the *Gain* value *(S, A)* is the expected entropy loss due to the known *A* value. The Entropy formula is given as follows:

$$\text{Entropy}(S) = \sum_{i=1}^{c} \quad - P_i \log_2 \, P_i \quad (2)$$

where the *c* is the total class number and P_i is the proportion of *S,* which is owned to class *i*. W-J48 has a few hyper-parameters tuning as the following Table 4.

Table 4. W-J48 parameters tuning options

Parameter	Description
C	The confident threshold for pruning (Numeric)
M	The minimum number of instances per leaf (Numeric)
U	Use the unpruned tree (Yes/No)
R	Use reduce error pruning (Yes/No)
B	Use binary splits (Yes/No)
S	Do not perform Subtree raising (Yes/No)
L	Do not clean up after a tree has been built (Yes/No)
A	Laplace smoothing for predicted probabilities (Yes/No)

The selection of parameters in the W-J48 classifier in Table 4 is based on mixed groups' data set features, including numerical and categorical values. Parameter C is calculated when the decision tree is formed, based on controlling the pruning level. The higher-level C sum would lead to the deterioration of results in the group. The M parameter is an essential parameter for setting the minimum number of instances leaves for data separation and ensuring even distribution of the data per branch. The U parameter supports the error-based pruning method, which calculates the training dataset's confidence interval to generalize and estimate classification errors. R parameter is added to change the pruning error type from error-based pruning to reduced error pruning, which substitutes each node for the most common classes. The binary splits B is chosen to handle the training dataset's nominal attributes to minimize data fragmentation problems. Also, the option S parameter is necessary to substitute the subtree during the pruning process. The L-parameter choice is beneficial when the pruning tree is regenerated to boost performance. The final option for parameter A is to apply the Laplace smoothing correction, thus improving the decision tree's efficiency.

4. PROPOSED METHOD

The proposed parallel approach in Figure 2 uses the traditional classifier, such as W-J48, to boost classification accuracy, classification performance, and classification optimization using the computational framework of Apache Spark to perform the process within the Hadoop cluster. The critical steps to allow parallel execution of classification optimization as outlined in the following Table 5.

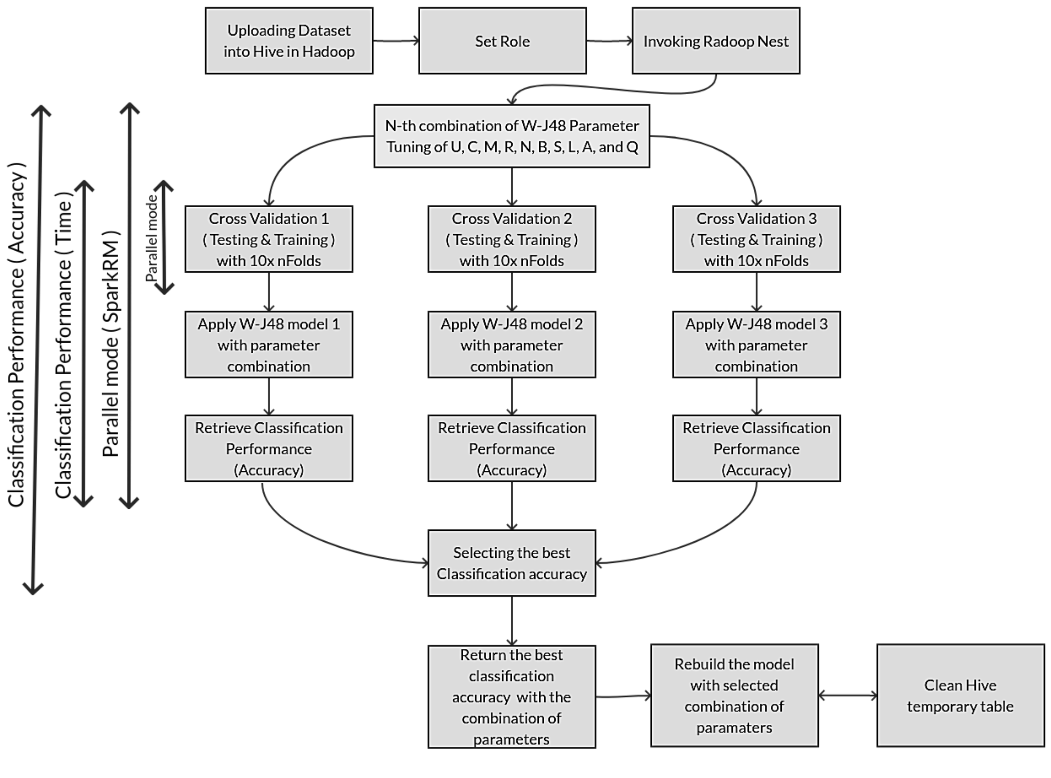

Uploading Dataset into Hive in Hadoop
Set Role
Invoking Radoop Nest
N-th combination of W-J48 Parameter Tuning of U, C, M, R, N, B, S, L, A, and Q
Classification Performance (Accuracy)
Classification Performance (Time)
Parallel mode (SparkRM)
Parallel mode
Cross Validation 1 (Testing & Training) with 10x nFolds
Cross Validation 2 (Testing & Training) with 10x nFolds
Cross Validation 3 (Testing & Training) with 10x nFolds
Apply W-J48 model 1 with parameter combination
Apply W-J48 model 2 with parameter combination
Apply W-J48 model 3 with parameter combination
Retrieve Classification Performance (Accuracy)
Retrieve Classification Performance (Accuracy)
Retrieve Classification Performance (Accuracy)
Selecting the best Classification accuracy
Return the best classification accuracy with the combination of parameters
Rebuild the model with selected combination of paramaters
Clean Hive temporary table

Figure 2. The proposed parallel process of W-J48 classifier parameter tuning

Table 5. List of steps of the proposed method

Steps	Methods	Details
1	Upload the dataset into Hadoop HDFS	This process is carried out through Apache Sqoop, connecting to the relational database management system (RDBMS), and copying the dataset to Hadoop HDFS.
2	Invoking Radoop Nest operator	a) Convert the text format dataset into parquet format via MapReduce operation. b) Create a temporary Hive table from the parquet format dataset. c) Constructing Radoop JAR, Radoop Hive JAR, and other dependencies JAR file, rebuild radoop_hive_v4.jar in the Radoop client local cache each time before uploading into Hadoop, Uploading Radoop JAR, Radoop Hive JAR, and other dependencies JAR file into Hadoop. d) Set Hive configuration to hive.warehouse.subdir.inherit.perms=true e) Upload rapidminer-extension-concurrency-9.6.0-all.jar, rmx_weka-7.3.0.jar, and rapidminer-extension professional-9.6.0-all.jar into Hadoop.
3	Invoking SparkRM operator.	a) Set Spark resource allocation policy is to dynamic resource allocation, which dynamically scales the cluster resources allocated to the application based on the workload. b) Upload rapidminer_pushdown-9.6.000.jar, radoop-common.jar, radoop-common-9.7.0.jar, radoop_spark15.jar, radoop-spark15-9.7.0.jar, radoop-spark15-pushdown.jar, radoop-spark15-pushdown-9.7.0.jar into Hadoop. c) Request for new application from cluster NodeManagers and wait for allocation AM (application master) container. d) Set container launch context for AM, change the view access control list (ACLs) to HDFS, set container launch context for AM, modify access control list (ACLs) to HDFS, Spark application submitted. (ACLs) to: HDFS, Spark application submitted.
4	Invoking Set Role operator - Change the role of one or more attributes in the dataset. The label role was selected for the variable name resolution code.	Next process → invoking optimize parameter operator - setting all possible values such as U, C, M, R, N, B, S, L, A, and Q for each W-J48 parameter. Higher combination parameters will require more computational time.
5	Invoking the Cross-Validation operator - Split the dataset from Hive into the testing and training set. The number of folds selected is set as 10. The execution is in parallel mode.	Next process → Invoking the W-J48 classifier with each parameter combination in Hadoop - Using Spark via SparkRM, each combination parameter for W-J48 is run parallel in slave nodes in Hadoop.
6	Invoking Apply Model operator - Applies the selected model on the dataset.	Next process → Retrieve the classification performance (Accuracy) - Retrieve the classification accuracy for each combination of parameters and select the best accuracy with its optimal parameter.
7	Rebuild the model with the list of optimal parameters - To rethink the model's inputs or algorithm and whether the model's output has been significantly degraded.	Next Process → Save the model - To reuse the model and compare it and other models to check the new data model.
8	Clean the temporary Hive table. - Manage the generated intermediate data automatically during large or complex query execution.	

5. THE EXPERIMENT

This experiment breaks down the hardware configuration into two components. The first part of the experimental setup is a single-node machine acting as the client and the second part as the Hadoop cluster. The Radoop is configured to the single-node computer. The suggested client configuration Table 6 is needed to ensure full integration effectively. In general, this study recommends the Linux-based operating system, which is equipped with sufficient memory, CPU, and disc storage. All necessary components, such as Java JDK, Spark components like SparkR, PySpark, and Scala, have to be installed with the correct version. The Hadoop cluster operates on one master node with three slave nodes. All computers must have the equivalent hardware specifications for optimal output and performance.

Table 6 displays the cluster system configuration, using Linux as its operating system. This study's Spark file format is a Parquet [24] format designed to improve performance effectively than the text format (default). The total percentage of the cluster resource was fixed at 70%, the Spark driver's memory is at 2 GB, and the executor's memory is set at 50% of resources. Apache Spark's resource allocation approach is dynamic resource allocation, which can dynamically adjust cluster resources based on the workload. If the application is no longer used, unused resources are returned to the cluster as needed. In the Radoop Nest operator, Radoop configures Hive as the default database in the Parquet format. This file format is also used to execute the

SparkRM container to speed up the process and enable parallelization. The partition is set to Random mode, and the number of partitions is fixed to three based on Hadoop's slave node number.

Table 6. Client and cluster configuration

Parameter	Client Configuration	Cluster Configuration
Operating System	Ubuntu Linux Desktop 16.04	Ubuntu Linux Server 18.04
Hadoop Distribution		Cloudera Enterprise 5.13
Data Mining Tools	Rapidminer Enterprise 9.6 + Radoop Extension	-
R	3.6.3	3.6.3
Spark	1.6.1	1.6.1
SparkR	SparkR version 2.0.1	SparkR version 2.0.1
PySpark	PySpark version 2.4.6	PySpark version 2.4.6
Scala	2.10.5	2.10.5
Java	JDK 1.8	JDK 1.8
CPU	8 x Core CPU	64 x Cores x 1 Master Node 64 x Cores x 3 Computing Nodes
RAM	32 GB RAM	64 GB x 1 Master Node 128 GB x 3 Computing Nodes
Storage	2 TB Storage	4 TB x 1 Master Node 8 TB x 3 Computing Nodes

5.1. Dataset overview

This study used the extensive data set extracted from the existing telco troubleshooting ticket system imported into Hadoop through extract-transform-load (ETL) tool like Apache Sqoop. The dataset's volume size is about 300 GB, consisting of more than 10 million compiled records from 2012 up to now. The data set metadata overview is illustrated in Table 7, which shows the variable's name, the type of variable, and the data set definition. The list of variables listed is extracted from feature selection using the information gain technique. Dataset is classified by area (zone). Each location will have different demographics and statistics based on problem ticket fault (customers, type of fault, symptom error code, cause code, and resolution code). The main objective is to predict each ticket's resolution code for each zone based on the information available; therefore, the accuracy output will differ. The second goal is to find the highest accuracy of the prediction results based on parameter tuning.

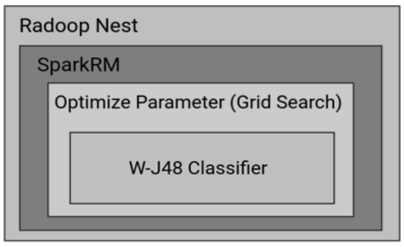
Table 7 shows that most columns are in categorical type. In this analysis, W-J48 classifier usage is suitable for this type of dataset. The previous workaround was accomplished using Weka ML software to predict resolution code without a parallel technique. Resolution codes describe how the service technician solved customer complaints (symptom error code) problems. Because of the memory constraint and computational power restriction on a single computer, the traditional W-J48 could not efficiently predict the large dataset size. This research, therefore, aims to overcome the previous experiment's limitations.

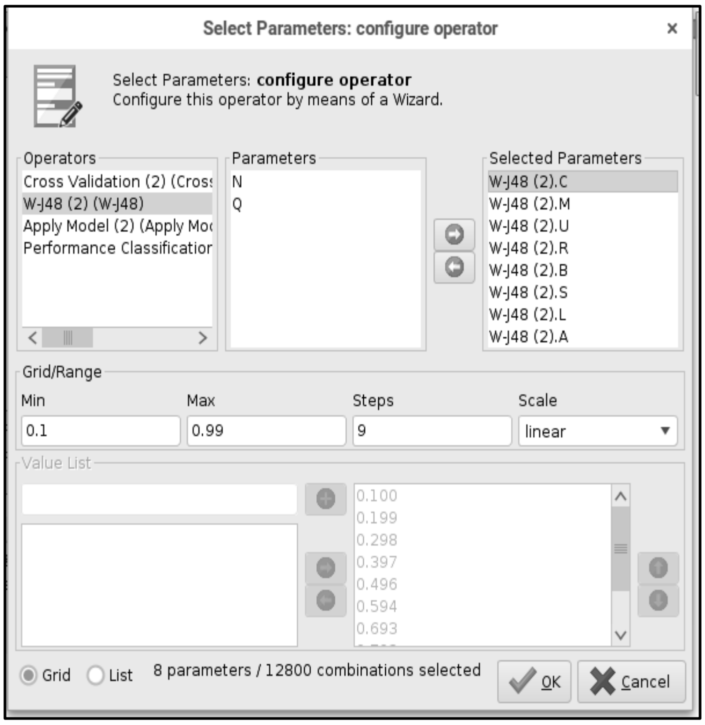
Table 7. Summary of trouble ticket dataset metadata

Variable Name	Variable Type	Description
Created Date	Date/Time	The date of the ticket is created.
Closed Date	Date/Time	The date of the ticket is closed.
Created By	Varchar (20)	The creator of the ticket.
Closed By	Varchar (20)	The info who closed the ticket.
Login Id	Varchar (50)	The Subscriber Login information.
Speed	Numeric	The Subscriber Package Speed.
Device Name	Varchar (20)	The Subscriber Device Name.
Btu Platform	Varchar (25)	The Subscriber BTU (Broadband Termination Unit) platform.
Btu Type	Varchar (20)	The Subscriber BTU (Broadband Termination Unit) type.
Resolution Code	Varchar (50)	The Code ID for Solving the Problem.
Symptom Error Code	Varchar (50)	The Code ID for when the Problem Found.
PTT	Varchar (20)	The local telco exchanges.
Zone	Varchar (20)	The name of the area of the fault occurs.
Exchange	Varchar (20)	The name of the exchange area of the fault occurs.
Description	Text	Free text notation about the fault.

5.2. Hyper-parameter grid search overview

This research uses Grid Search [25] as the method used to fine-tune the classification model. Grid Search is a brute-force approach where the algorithm uses the best to train the model and check all possible hyper-parameters. This is a comprehensive analysis of the model's hyper-parameter space, either over the whole or a subset. Since the W-J48 classifier has several parameters that can be a tuneup, all possible combinations of parameters are tested, searching for the best accuracy. Using the Optimize Parameter and SparkRM operator, which must be implemented within Hadoop, the parallelism method can be adapted to the traditional W-J48 classifier. The diagram in Figure 3 shows the embedding of the traditional W-J48 classifier within the Grid Search operator. Each layer reflects the various functionalities of each operator. Radoop Nest works to handle the client-to-Hadoop cluster interactions. On the other hand, SparkRM acts as a converter between the core operator and Hadoop, generating Spark's compatible code, executed in Hadoop. Any core operator inside SparkRM operators, such as the traditional W-J48 classifier, is automatically modernized for Hadoop's compatibility.




Select Parameters: configure operator

Select Parameters: **configure operator**
Configure this operator by means of a Wizard.

Operators: Cross Validation (2) (Cross; W-J48 (2) (W-J48); Apply Model (2) (Apply Mo; Performance Classificatior

Parameters: N; Q

Selected Parameters: W-J48 (2).C; W-J48 (2).M; W-J48 (2).U; W-J48 (2).R; W-J48 (2).B; W-J48 (2).S; W-J48 (2).L; W-J48 (2).A

Grid/Range

Min	Max	Steps	Scale
0.1	0.99	9	linear

Value List: 0.100, 0.199, 0.298, 0.397, 0.496, 0.594, 0.693

Grid List 8 parameters / 12800 combinations selected

OK Cancel

Figure 3. Grid search implementation with traditional W-J48 decision tree

The modernization process enables concurrent execution of any conventional operator which has previously been configured to operate in single-threaded, allowing the operator to access higher memory within the Hadoop cluster. In this analysis, with a combination of multiple parameters with different settings ranges (8 parameters) within the Grid Search operator, about 12,800 parameter combinations were created ultimately to search for the best classification accuracy in Hadoop. From the algorithm perspective, assume that each dimension of $A \in \mathbf{A}$ is a parameterization of the machine learning pipeline from data processing to model hyperparameter tuning. The Grid Search process is focused on finding $A*$ (3) to minimize the loss of trained dataset with joined of algorithms and the optimal hyperparameter values. The implementation of Grid Search is via the following formula:

$$A *= \underset{A \in \mathbf{A}}{\operatorname{argmin}} \frac{1}{K} \sum_{i=1}^{K} \mathcal{C}\,(A, D_{\text{train}}^{(i)}, D_{\text{valid}}^{(i)}) \tag{3}$$

$$\{D_{\text{train}}^{(1)}, \dots, D_{\text{train}}^{(K)}\} \tag{4}$$

$$\{D_{\text{valid}}^{(1)}, \dots, D_{\text{valid}}^{(K)}\} \tag{5}$$

$$\mathcal{C}(A, D_{\text{train}}^{(i)}, D_{\text{valid}}^{(i)}) \tag{6}$$

The formula (4) and (5) is the train and validation set acquired by K-Fold Cross-Validation, and (6) is the loss for pipeline data set (i) training and testing data set (i), which evaluated. There is no guarantee that the Grid Search will provide the best results, and often the incompatibility of the combination of parameters will cause the classifier to produce errors.

6. RESULTS

The experiment is conducted with three different classification processes in Hadoop to benchmark the classification performance (accuracy and computation time) with the traditional W-J48 classifier for example, the normal classification performance without parameter tuning (default parameter value), the non-parallel method with parameter tuning (tuned parameter value), and the parallel method with parameter tuning (tuned parameter value). For each iteration, the parameter tuning method involves 12,800 combinations of parameters (C, M, U, R, B, S, L, A) in Table 8, Figure 4 with 10x cross-validation. These three approaches are proposed to evaluate the efficient way of modernizing traditional classifiers, such as W-J48, to apply parallel and parameter tuning. Another aim of the proposed method is to research any difference in performance in accuracy and computation time. The initial experiment proves that the proposed method claims a shorter time to complete the classification process, around 0.28 minutes on average. The classification result, however, is considered to be at medium range accuracy, averaging 64.09%. The experiment applies parameter C (0.25), parameter M (2.0) by default, and the other parameter is set to false.

The second experiment in Table 9 shows that the highest accuracy is obtained from Zone Bangsar (BNR), which has a 93.05% accuracy but has taken about 2.40 minutes to complete the entire classification phase. The estimated classification process time is 5.378 minutes, and the average accuracy is approximately 76.35%. The third experiment in Table 10 shows significant progress in the classification's accuracy and the time is taken to complete the iteration.

Table 8. Summary of W-J48 normal classification performance

Zone Dataset	W-J48 Parameters								Time	Accuracy
	C	M	U	R	B	S	L	A		
BANGI	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.29	63.20%
BANGSAR	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.28	67.17%
BUKIT ANGGERIK	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.31	49.11%
CYBERJAYA	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.33	77.46%
GOMBAK	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.28	62.57%
KEPONG BATU	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.27	65.04%
KERAMAT	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.24	68.06%
KLANG	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.28	61.14%
MALURI	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.28	60.63%
PANDAN	0.250	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0.24	66.53%
Average									**0.28**	**64.09%**

Table 9. Summary of the non-parallel method with parameter tuning classification performance

Zone Dataset	W-J48 Parameters								Time	Accuracy
	C	M	U	R	B	S	L	A		
BANGI	0.396	2.0	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	6.36	68.57%
BANGSAR	0.594	3.0	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	2.95	74.38%
BUKIT ANGGERIK	0.792	1.0	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	2.45	76.44%
CYBERJAYA	0.594	1.0	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	2.40	93.05%
GOMBAK	0.891	1.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	9.50	68.88%
KEPONG BATU	0.891	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	10.16	71.01%
KERAMAT	0.594	1.0	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	9.20	75.05%
KLANG	0.298	3.0	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	5.45	72.28%
MALURI	0.594	1.0	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	3.16	86.13%
PANDAN	0.298	2.0	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	2.15	77.69%
Average									**5.378**	**76.35%**

Table 10. Summary of the parallel method with parameter tuning classification performance

Zone Dataset	W-J48 Parameters								Time	Accuracy
	C	M	U	R	B	S	L	A		
BANGI	0.396	2.0	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	5.10	71.23%
BANGSAR	0.594	3.0	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	2.12	75.58%
BUKIT ANGGERIK	0.792	1.0	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	2.06	78.65%
CYBERJAYA	0.594	1.0	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	1.25	94.13%
GOMBAK	0.891	1.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	8.07	72.48%
KEPONG BATU	0.891	2.0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	8.41	73.14%
KERAMAT	0.594	1.0	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	7.51	76.15%
KLANG	0.298	3.0	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	4.36	75.24%
MALURI	0.594	1.0	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	2.10	88.52%
PANDAN	0.298	2.0	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	1.45	80.78%
Average									**4.24**	**78.59%**

The average classification time taken is 4.24 minutes, and the average classification accuracy of 78.59%. The percentage accuracy and time difference from the second experiment result is 2.24% and 1.138 minutes. Figure 4 shows the difference in classification accuracy across the zone. The classification result is shared with the non-parallel and parallel approaches with a slight margin. Standard classification system accuracy is lower as the algorithm only applies default values of the traditional W-J48 classifier. The next Figure 5 shows the time gap in the classification result. The fastest classification results are with the normal method of classification. The approach to parameter tuning has two different modes: the parallel method and the non-parallel method. The parallel process is slightly lower than the non-parallel method for the classification time.

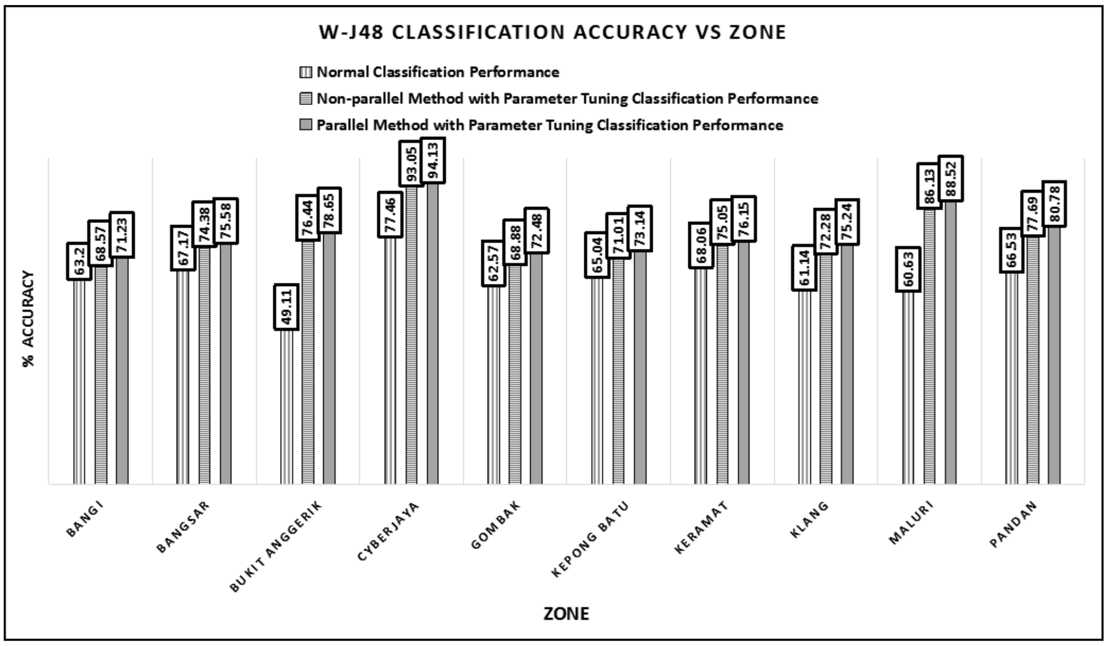


Figure 4. Comparison of classification accuracy in Hadoop

As for the summary, the result of parallel classification Table 10 or non-parallel tuning Table 9 produced better accuracy than the normal classification method Table 8. The classification accuracy Performance ($P_{Accuracy}$) and calculation time Performance (P_{Time}) can be described in the following formula based on the **a**verage values (A):

$$P_{Accuracy} = \frac{(A_{Tune} - A_{No.Tune})}{A_{No.Tune}} * 100 = \frac{(78.59 - 64.091)}{64.091} * 100 = 22.62\% \quad (7)$$

$$P_{Time} = \frac{(A_{TuneParallel} - A_{TuneSequential})}{A_{TuneSequential}} * 100 = \frac{(4.243 - 5.378)}{5.378} * 100 = 21.10\% \quad (8)$$

For this study, classification achievement using Hadoop's parallel tuning parameter (SparkRM) shows a substantial 22.62% increase. It takes almost 21.1% less time to complete the loop. All results are based on the small number node, 3 Hadoop slave nodes due to this study's budget constraints. Other disadvantages were discovered in the current architecture, which only allowed batch-oriented processing when invoking MapReduce on the data aggregation process. When the network is heavily used, latency becomes the problem, and the computation sometimes becomes slower. The study also discovers that there is no supporting caching functionality that could speed up the whole process. Nevertheless, the result is promising as Hadoop is exceptional and scalable [26] for further study. It has lots of research areas to be fine-tuned for improved accuracy and efficiency.

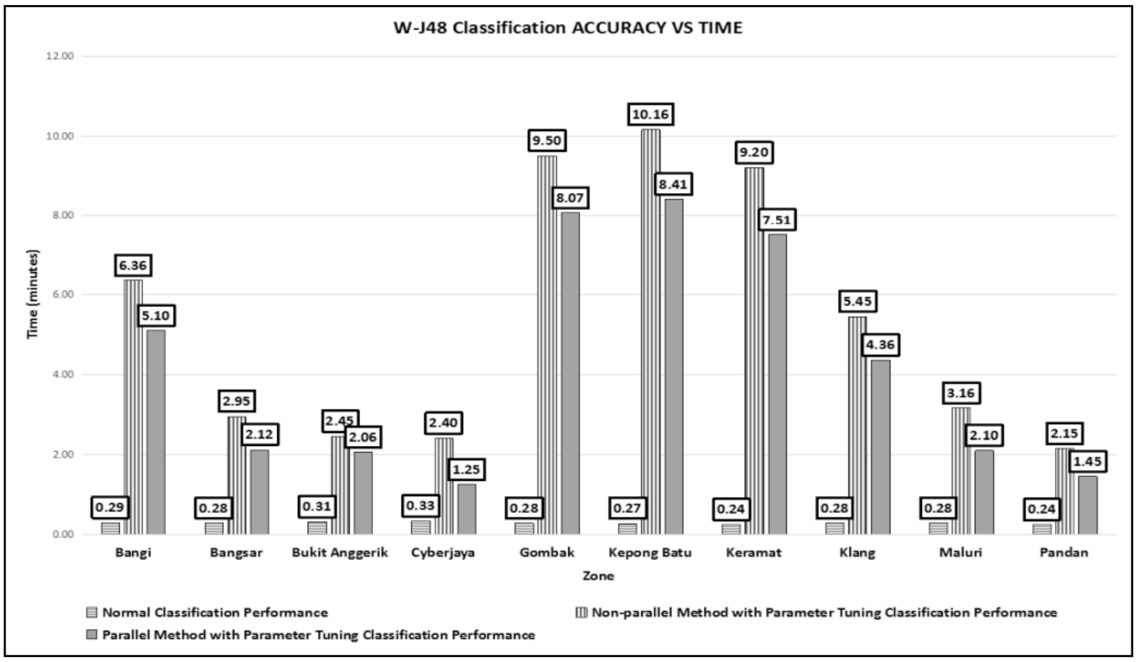


Figure 5. Comparison of classification time in Hadoop

7. CONCLUSION

The volume of data generated today presents a new challenge for traditional machine learning techniques. To obtain useful information from the large volume of data, such as feature selection, feature engineering, and classifier parameter optimization, involves different techniques and strategies to solve problems. One of the best classification techniques is to get the best combination of parameters for the new and unknown dataset. The solution can be achieved by migrating the traditional classifier to the Hadoop platform due to the large dataset volume. Hadoop storage and memory can be horizontally scaled, supporting parallel computing through computational frameworks like MapReduce and Spark. In this analysis, a new approach was introduced by combining the Grid Search with traditional classifiers such as W-J48 to enable parallel functions within the Hadoop framework. This integration ensures high accuracy for each dataset in the classification work. The study was conducted using Radoop, which supports Hadoop HDFS, MapReduce, and Apache Spark integration to utilize its unique functionality.

The proposed method shows significantly improved classification accuracy and reduced classification time with enabled Hadoop parameter tuning. The selected traditional classifier, such as W-J48, the parameter tuning operator, and the Cross-Validation operator, can now work alongside Radoop and SparkRM. The same parallel theory with SparkRM can also be applied in various applications, especially in other fields of machine learning such as artificial intelligence, data mining, and statistics. These include regression, ensemble method, clustering, dimension reduction, and bayesian, requiring intensive processing and memory use. Hadoop now solves the previous restriction on computing limitations. Future studies can be conducted using this study's essential parallelization principle to further boost classification efficiency by extending Hadoop slave node numbers and increasing current memory capacity to each node's limit. Another potential solution is to incorporate the large volume of datasets into the partitioning concept and expand the current approach to the cloud environment and distributed system kernel, for example, Apache Mesos.

REFERENCES

[1] D. Palacios, C. Morillas, M. Garcés and P. Tapia, “Big data-empowered system for automatic trouble ticket generation in IoT networks,” in *2019 IEEE 2nd 5G World Forum (5GWF)*, Dresden, Germany, 2019, pp. 63-68.

[2] N. K. Chauhan and K. Singh, “A review on conventional machine learning vs deep learning,” in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, Greater Noida, Uttar Pradesh, India, 2018, pp. 347-352.

[3] R. Li, X. Huang, S. Song, J. Wang, and W. Wang, “Towards customer trouble tickets resolution automation in large cellular services,” in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, 2016, pp. 479-480.

[4] W. Juan and W. Ping, “Optimization of fuzzy rule based on adaptive genetic algorithm and ant colony algorithm,” in *2010 International Conference on Computational and Information Sciences*, Chengdu, 2010, pp. 359-362.

[5] L. Nguyen, “Converting graphic relationships into conditional probabilities in bayesian network,” in *Bayesian Inference*, J. P. Tejedor, IntechOpen Limited, London, United Kingdom, 2017.

[6] Y.-C. Chen, T. Suzuki, M. Suzuki, H. Takao, Y. Murayama, and H. Ohwada, “Building a classifier of onset stroke prediction using random tree algorithm,” *International Journal of Machine Learning and Computing*, vol. 7, no. 4, pp. 61-66, 2017.

[7] A. V. Shichkin, A. G. Buevich, and A. P. Sergeev, “Comparison of artificial neural network, random forest and random perceptron forest for forecasting the spatial impurity distribution,” in *AIP Conference Proceedings*, vol. 1982, no. 1, 2018.

[8] N. Saravanan, V. Gayathri, “Performance and classification evaluation of J48 algorithm and kendalls based J48 algorithm (KNJ48),” *International Journal of Computer Trends and Technology*, vol. 59, no. 2, pp. 73-80, 2018.

[9] S. K. Singhal, “Bio-Inspired Bayesian Network Learning Algorithm For Bayesian Network Classifiers.,” *International Journal of Advance Engineering and Research Development*, vol. 2, no. 03, Mar. 2015.

[10] A. H. Ali and M. Z. Abdullah, “A parallel grid optimization of SVM hyperparameter for big data classification using spark Radoop,” *Karbala International Journal of Modern Science*, vol. 6, no. 1, 2020.

[11] S. Chellappan and D. Ganesan, “Introduction to apache spark and spark core,” *Practical Apache Spark*, pp. 79-113, 2018.

[12] Z. Chen and J. Cheng, “A parallel softmax classification algorithm based on MapReduce,” in *2018 13th International Conference on Computer Science & Education (ICCSE)*, Colombo, 2018, pp. 1-5.

[13] F. Budiman, “SVM-RBF parameters testing optimization using cross validation and grid search to improve multiclass classification,” *Scientific Visualization*, vol. 11, no. 1, pp. 80-90, 2019.

[14] S. Du and J. Li, “Parallel processing of improved KNN text classification algorithm based on Hadoop,” in *2019 7th International Conference on Information, Communication and Networks (ICICN)*, Macao, Macao, 2019, pp. 167-170.

[15] W. Nie, B. Fan, X. Kong, and Q. Ma, “Optimization of multi kernel parallel support vector machine based on Hadoop,” in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Xi'an, 2016, pp. 1602-1606.

[16] J. Shah and R. Patel, “Classification techniques for Disease detection using Big-data,” in *2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, Mysuru, India, 2019, pp. 140-145.

[17] F. Yuan, F. Lian, X. Xu and Z. Ji, “Decision tree algorithm optimization research based on MapReduce,” in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, 2015, pp. 1010-1013.

[18] T. Kuang, Z. Hu, and M. Xu, “A Genetic Optimization Algorithm Based on Adaptive Dimensionality Reduction,” *Mathematical Problems in Engineering*, vol. 2020, pp. 1-7, May 2020.

[19] W. Dai and W. Ji, “A MapReduce implementation of C4.5 decision tree algorithm,” *International Journal of Database Theory and Application*, vol. 7, no. 1, pp. 49-60, Feb. 2014.

[20] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, “YALE*,” Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining-KDD* ’06, 2006

[21] S. Shaw, A. F. Vermeulen, A. Gupta, and D. Kjerrumgaard, “Loading Data into Hive,” in *Practical Hive*, Berkeley, California, USA: Apress, 2016, pp. 99-114.

[22] Y. Zhu and X. Wang, “HDFS optimization program based on GE coding,” *Jisuanji Yingyong/ Journal of Computer Applications*, vol. 33, no. 3, pp. 730-733, 2013.

[23] M. N. Akhtar, J. Mohamad Saleh, and C. Grelck, “Parallel Processing of Image Segmentation Data Using Hadoop,” *International Journal of Integrated Engineering*, vol. 10, no. 1, Apr. 2018.

[24] D. Vohra, “Apache Parquet,” in *Practical Hadoop Ecosystem*, New York, USA: Apress, 2016, pp. 325-335.

[25] I. Syarif, A. Prugel-Bennett, and G. Wills, “SVM Parameter Optimization using Grid Search and Genetic Algorithm to Improve Classification Performance,” *TELKOMNIKA Telecommunication Computing Electronics and Control,* vol. 14, no. 4, p. 1502, Dec. 2016.

[26] W. Litke and M. Budka, “Scaling beyond one rack and sizing of Hadoop platform,” *Scalable Computing: Practice and Experience*, vol. 16, no. 4, pp. 423-436, 2015.

BIOGRAPHIES OF AUTHORS

Fauzy Che Yayah is a graduate with a Master of Software Engineering & Software Architecture from Multimedia University, Cyberjaya, Malaysia (2012). He obtained a Bachelor's Degree in Computer Science from Universiti Teknologi Malaysia, Skudai, Johor, in 2000. His research focuses on the areas of Big Data, Machine Learning, Artificial Intelligence, Data Mining, Predictive Analytics, and Advanced Analytics.
Further info on his LinkedIn page: https://my.linkedin.com/in/ts-fauzy-che-yayah-20628426

Khairil Imran Ghauth graduated with his degree in Bachelor of Information Technology (Hons.) from Multimedia University, Malaysia, the degree of Master of Information Technology from the University of Melbourne, Australia, and a Ph.D. degree in Computer Science from the University of Malaya, Malaysia. He has vast experience in research, and he has published articles in reputable journals primarily in the area of Recommender Systems and Information Filtering. He has also been invited as a reviewer and program committee in various journals. His researches are in fields of Information filtering, information retrieval, recommender system, web usage mining, web services, distributed system, and Internet technologies has been tackled.
Further info on his homepage: https://mmuexpert.mmu.edu.my/khairil-imran

Choo-Yee Ting is currently working as Associate Professor at the Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia. He is also the Deputy Dean of the Institute for Postgraduate Studies, Multimedia University. In the year 2002, Choo-Yee Ting was awarded the Fellow of Microsoft Research by Microsoft Research Asia, Beijing, China. He has been active in research projects related to predictive analytics and Big Data. Most of the projects were funded by MOE, MOSTI, Telekom Malaysia, MDeC, and industries.
Further info on his homepage: https://mmuexpert.mmu.edu.my/cyting**.**