Journal of Information Technology and Computer Science
Volume 5, Number 3, Desember 2020, pp. 302-312
Journal Homepage: www.jitecs.ub.ac.id

Development of Big Data App for Classification based on Map Reduce of Naive Bayes with or without Web and Mobile Interface by RESTful API Using Hadoop and Spark

Imam Cholissodin[1], Diajeng Sekar Seruni[2], Junda Alfiah Zulqornain[3], Audi Nuermey Hanafi[4], Afwan Ghofur[5], Mikhael Alexander[6], Muhammad Ismail Hasan[7]
[1,2,3,4]Faculty of Computer Science, Computer Science, Brawijaya University, Malang, Indonesia
[5,6,7]FGA Big Data Analytics Batch 2, Brawijaya University, Malang, Indonesia
[1]imamcs,@ub.ac.id, [2] diajengskr@gmail.com, [3] jundaa63@gmail.com, [4] udiafie27@gmail.com, [5] afwanghofur5@gmail.com, [6]sandermik13@gmail.com, [7] muhisan10@gmail.com

Received: 05 August 2020; Accepted: 21 Desember 2020

Abstract. Big Data App is a developed framework that we made based on our previous project research and we have uploaded it on github, which is developing lightweight serverless both on Windows and Linux OS with the term of EdUBig as Open Source Hadoop Distribution. In this study, the focus is on solving problems related to difficulties in building a frontend and backend model of a Big Data application which by default only runs scripts through consoles in the terminal. This will be quite a tribulation for the end users when the Big Data application has been released and mass produced to general users (end users) and at the same time how the end users test the performance of the Map Reduce Naive Bayes algorithm used in several datasets. In accordance to these problems, we created the Big Data App framework to make the end users, especially developers, feel easier to build a Big Data application by integrating the frontend using the Web App from Django framework and Mobile App Native, while for the backend, we use Django framework that is able to communicate directly with the script either hadoop batch, streaming processing or spark streaming very easily and also to use the script for pig, hive, web hdfs, sqoop, oozie, etc. the making of which is extremely fast with reliable results. Based on the test results, a very significant result in the ease of data computation processing by the end users and the final results showing the highest classification accuracy of 88.3576% was obtained.

Keywords: big data, map reduce of naive bayes, serverless, web and mobile app, restful api, django framework

1 Introduction

Development of Big Data applications in the blended form between conventional coding with the native language of Big Data, which is java, and those already using coding with high-level programming languages, for example using hadoop streaming, which initially can only be run in batch processing to be able to run streaming using python language, and pyspark streaming, all of which lead and / or have been serverless-based, is very promising to produce applications that are fast in making and reliable in terms of results [1][2][3]. Conditions that often arise in the making of Artificial Intelligence (AI) algorithm-based Big Data applications as machine learning such as Map Reduce Naive Bayes to a high level of Map Reduce Deep Learning

algorithm [4][5] is that the majority applications are limited to the form of notebooks (it is insufficient to be just like jupyter notebooks in local or public like Google colab, AWS sage maker and others), or even scripts that are incomplete and fragmented into small files that are very difficult to build in the form of a masterpiece application production for Big Data App [6]. It is admittedly considered suitable for the need of learning and training scale. If it is made for large-scale application production for the society or for the initiative to begin making start-ups, however, it is considered very far away from the goal as the final step of production that must be achieved.

Thus, the problem that often comes up when developing Big Data applications is the infrastructure readiness both the on-premise (the physical form in local or private) and the public (the non-physical in the cloud such as GCP or AWS or others) from the hardware as an adequate server, or in the form of a combination of private and public, both of which are associated with the development of backend and frontend that by default is usually in the form of a console. Consequently, developers are difficult to create visual applications in the form of a web or mobile interface as the frontend [7][8][9][10]. Therefore, in this study, a link between the frontend and the backend is created and as a test of the performance of the algorithm for the computational results of the dataset, the Map Reduce Naive Bayes algorithm, where the backend is the default tool of apache, which by standard uses WebHDFS and Spark API, both of which are accessible and run through the console in the terminal, is utilized. After that, we made an easier solution by using Django framework from python which acts as the frontend of the Web App (Web Interface Big Data App) and backend web services at the same time. Meanwhile, the Mobile App only serves as the frontend (Mobile Interface Big Data App) while the backend still uses Django framework. We use both of these solutions to communicate between the data using the Representational state transfer and Application program interface (RESTful API) which will later be converted into a file in json format.

2 Method

2.1 About Dataset

The dataset used in this study are of two kinds, the first of which is the data on activity as the smallest data sampling that we take from Naik (2016) as the data material to create a simulation of testing process within the Big Data App framework [11]. The data of this activity consist of three features and four classes, which are slightly modified. The second data is Nursery School, which contains the data about preschool education or commonly called kindergarten (TK), which is a form of formal education that is employed to assist the physical and spiritual growth and development of children in order to obtain the readiness to enter further education acquired from the UCI Repository. This dataset has 12,960 data and 8 attributes / features, i.e. parents referring to the work of parents (usual, pretentious, great_pret), has_nurs referring to child caretakers at home (proper, less_proper, improper, critical, very_crit), form referring to family completeness (complete, completed, incomplete, foster), children referring to the number of children (one, two, three, more), housing referring to house condition (convenient, less_conv, critical), finance referring to financial condition (convenient, inconv), social referring to social condition (non-prob, slightly_prob, problematic) , and health referring to health condition (recom, priority, not_recom).

Although both data can be considered quite small, which means that they are not able to represent Big Data in terms of volume yet, they are still able to meet the criteria in terms of Map Reduce computational complexity in the utilized algorithms since it is definitely required to adjust to every stage of the Big Data ecosystem, where the Big

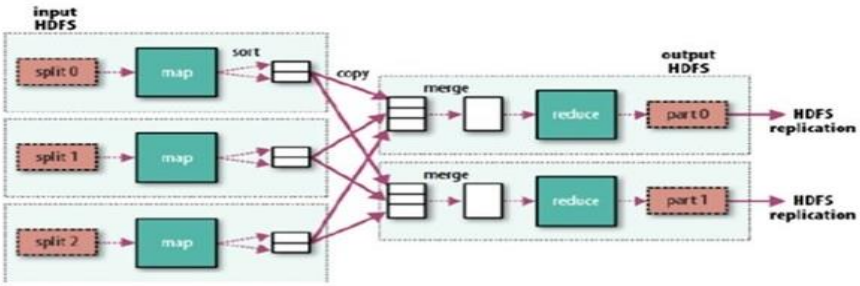
Data process is available in the form of Map and Reduce, both batch (Map Reduce Hadoop) or Streaming processing (Spark RDD or Spark SQL) on distributed computation. The illustration of the activities is displayed in Fig. 1.

Figure 1. Daily activities accessed [12**]**

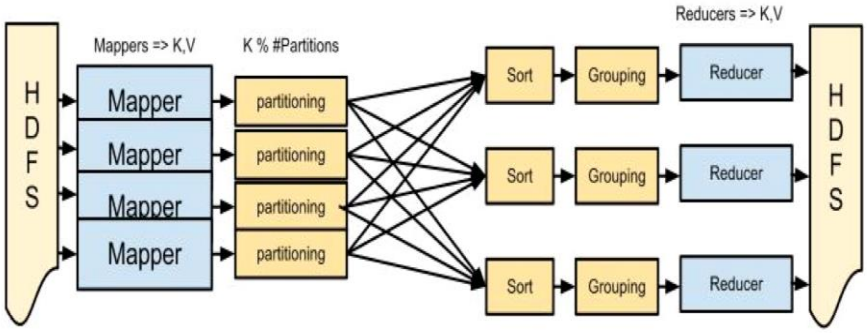
2.2 Naive Bayes Map Reduce

Naive Bayes can be implemented by using Non Map Reduce (Non-MR) and Map Reduce (MR). For utilizing MR, Hadoop framework, which has 2 kinds of components, i.e. Hadoop Distributed File System (HDFS) as the data storage medium and Map Reduce as the very large data processor (Big Data), is applicable. Map Reduce initially had two components, i.e. Job Tracker and Task Tracker. Job Tracker is a component on the Master computer, and Task Tracker is a component on the Slave computer. Since Hadoop 2.x, however, "Job Tracker" has been deprecated and replaced by "Resource Manager", and "Task Tracker" has also been deprecated and replaced by "Node Manager" in Map Reduce version 2 (MRv2) [13].

Figure 2 (a) and (b) show MR pipeline. First, the mapper receives (key, value) and the output is (key, value) where #partitions expresses the number of nodes in the cluster (multi-node), the second is partitioning / sorting / grouping that carries out Iterable[value] and scaling, and finally, the reducer process accepts (key, Iterable[value]) the output of which is (key, value) as the final result. Meanwhile, Figure 3 represent the MR manualization process in Word Count. Map Function is utilized to read the file input in the form of pairs of key and value, then to produce pairs of key and value output that have already been grouped with a base key, and to add "sort and shuffle" process. On the other hand, Reduce Function is employed to read the results of "key and value" output that have already been grouped with a base key from the results of the Map Function, then to calculate or counter the total value for each key group. Therefore, Reduce Function produces a "key and value" output with a unique key accompanied by a value from the counter results.



(a)



(b)

Figure 2. MR Pipeline [14,15]

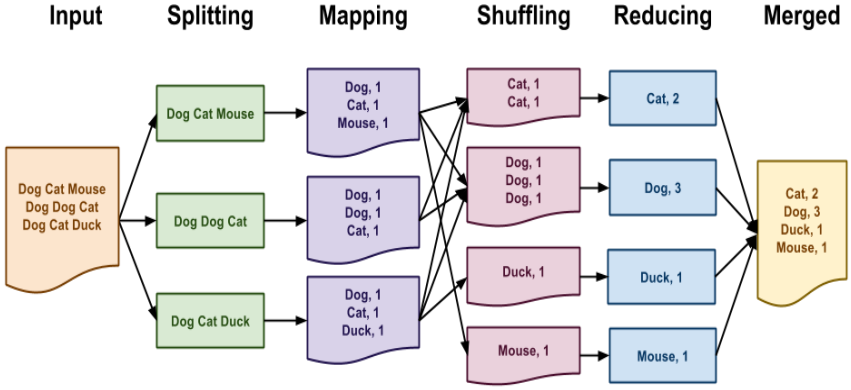
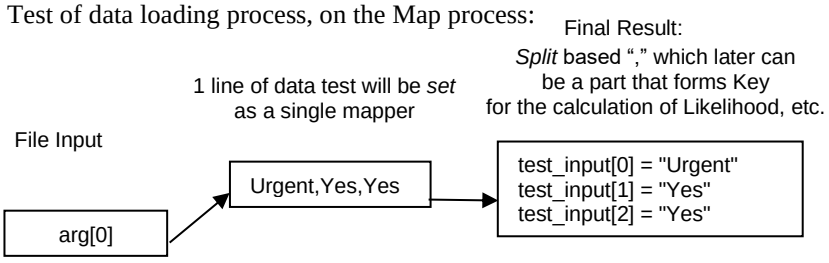


Figure 3. MR Word Count for fundamental MR of Naive Bayes [16, 17]

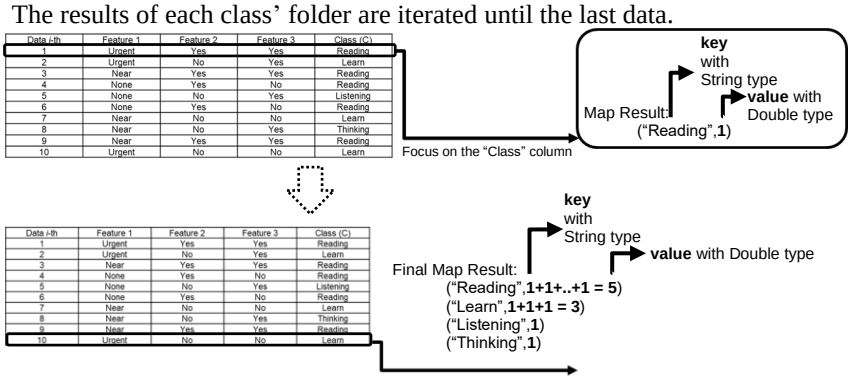
The following is the steps of Naive Bayes algorithms using MR, for example simple data [11][18][19][20] are used in this detailed process:

1. Loading test and training data
   - Test of data loading process, on the Map process:



     Where in the test data, feature 1 = "Urgent", feature 2 = "Yes", feature 3 = "Yes". Then, it will obtain the result of Class = "?". Feature 1 states "What is the nature of task deadline?", feature 2 states "Is there no Coding task? , and feature 3 states "Are there not many tasks?".
   - Loading training data (in the next step, i.e. Calculate Prior Opportunities and Likelihood)
2. Calculating Prior Opportunities of each class

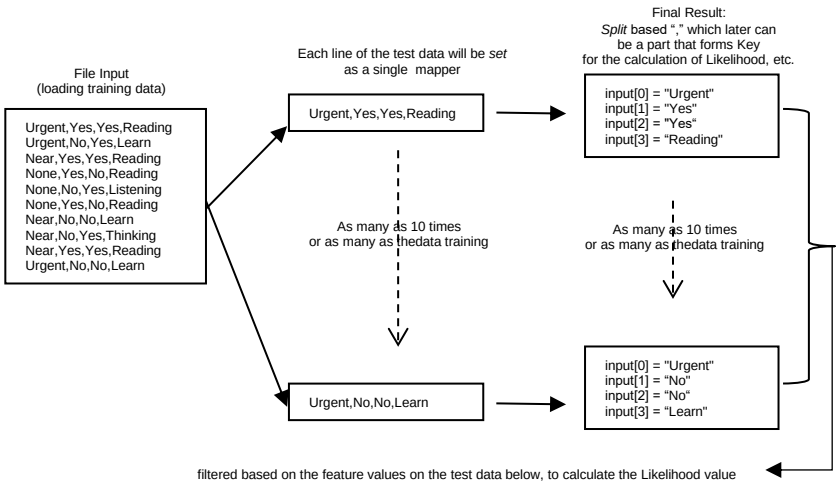
The results of each class' folder are iterated until the last data.



The map process, for example, with "prior_count" variable automatically performs iteration (increments +1 if a suitable key is found in the class column) until the last data even though the Mapper program code does not show the "for" looping syntax. Finally, in the Final Map Results, each value for example is stored in the prior_ count <key, value> variable in the form of HashMap, and has not been shared with many data trains. Hence, the value is not yet in the interval [0,1].

3. Calculating Likelihood Opportunities (Opportunities for discrete or qualitative features to Class)

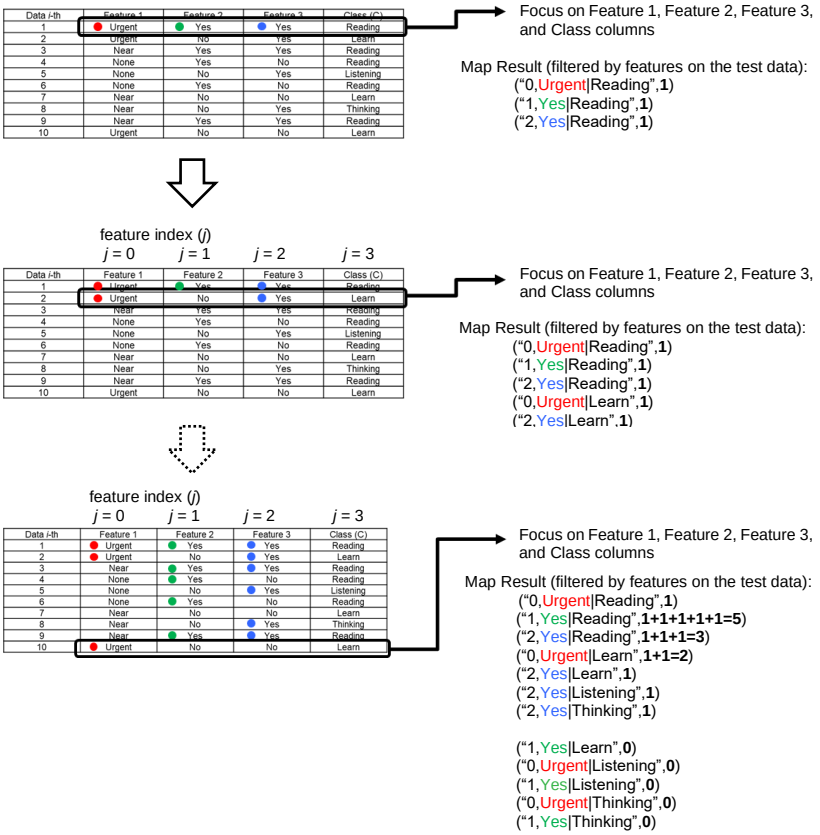
The thing that is required to perform first is to make a variable declaration, for example with the name of "count" to calculate the amount of training data and variable declaration for likelihood opportunities, for example with the name of "features_count" to accommodate the likelihood value, i.e. by carrying out the HashMap class instantiation (to create features_count objects), then, to put in as content on the String as key = {" feature index"+","+" feature values"+"|"+" class"}, while at Double as value = double type value, the application on the calculation of which states "the number of feature emergence to the class divided by the number of class emergence in the training data" = [0;1].



It is known in the test data that feature 1 = "Urgent", 2 = "Yes", 3 = "Yes".

feature index (*j*)

j = 0 *j* = 1 *j* = 2 *j* = 3



The map process will automatically carry out iteration (increment +1 if the corresponding key is found in feature 1 to 3 columns to the class) until the last data. In addition, the number of training data on the count variable = 10 from the result of ++ count code from the scanning process of all training data is also obtained. In the likelihood of zero, a smoothing process can be used, which is carried out by replacing the zero value with a number > 0, for example 10^{-8} to 10^{-5} or the others.

4. Calculating Posterior Opportunities (Class opportunities to features)
   Posterior opportunities derive from multiplying the Likelihood opportunities and the class emergence opportunities (Prior).
5. Determining Classification Results
   After obtaining all the posterior values from each class, the next step is to determine the class for the test data. Class determination is carried out by comparing the posterior values between classes. The highest posterior value will be the class for the test data.

2.3 Proposed Development of Big Data App Based on Map Reduce of Naive Bayes Algorithm on Desktop, Web and Mobile Interface by RESTful API Using Hadoop and Spark

The developed desktop-based implementation utilizes Netbeans and the web employs Python and Django Framework programming language for Backend, and CSS, HTML, Javascript for Frontend to integrate with Spark and Hadoop. Meanwhile, the Android-

based application utilizes Java programming language so that the compiling result is an android native application. The database is not employed in this application because the data process utilizes ResftFul API created in Django Framework and the development uses Android Studio. Specific for web-based implementation, it is deployed on cloud computing from AWS, i.e. the EC2 where the environment is the place where the application is run, and the data processing is on Spark and Hadoop.

```
def function_post_nb_run(request):
        if request.method == 'POST':
                cetak = nb_run(request, False)
                #render
                return render(request, page.html', {
                'prediction' : cetak
                })
        else:
                return render(request, page.html')
```

Source Code 1. View.py from Django Framework (part 1)

```
def nb_run(request, type_api=True):
        # make unique directory
        output_dir = randomString(10)

        if type_api:
                payload = json.loads(request.body)
                feature_1 = payload['feature_1']
                feature_2 = payload['feature_2']
                feature_3 = payload['feature_3']
        else:
                feature_1 = request.POST['feature_1']
                feature_2 =  request.POST['feature_2']
                feature_3 =  request.POST['feature_3']
        # merge input
        dataInput = feature_1 + "," + feature_2 + "," + feature_3

        #run hadoop
        exitcode, stdout, stdin = run_process([HADOOP_BIN, 'jar',
               'hadoop/NBMapReduce/NBMapReduce.jar', 'NBCDriver', \
               dataInput, '/user/ubuntu/nb-input/dataset.txt', \
               '/user/ubuntu/nb-output/'+output_dir])
        #Return if error occured
        if exitcode:
            save_result = exitcode
        else:
            save_result = run_process([HADOOP_BIN, 'fs', '-cat' , \
             '/user/ubuntu/nb-output/'+output_dir+'/*'])
        #delete output dir
        run_process([HADOOP_BIN, "fs", "-rm", "-r", \
                '/user/ubuntu/nb-output/'+output_dir])
        #return
        return save_result[1]
```

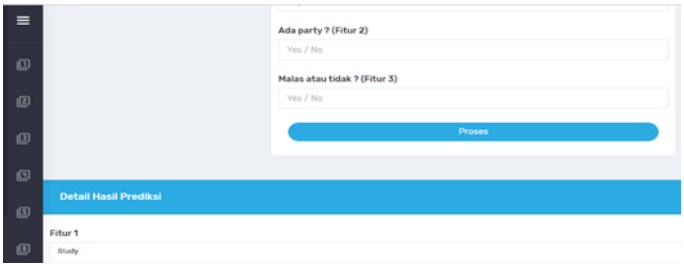
Source Code 2. View.py from Django Framework (part 2)

```
#run spark
run_process([SPARK_BIN, 'spark/pyspark_nb.py', number_features, \
"file://"+file_input, "file://"+dir_output])
```

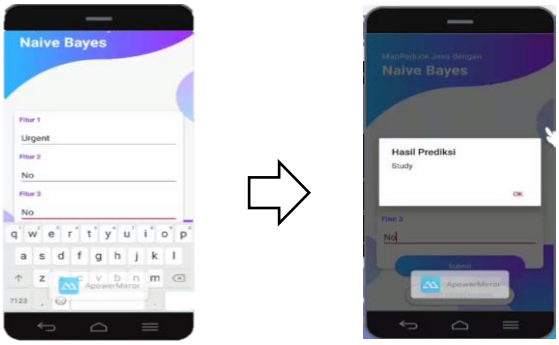
Source Code 3. View.py from Django Framework (part 3)

In accordance with Source Code 1 and 2, "#run hadoop" section is to run Hadoop MapReduce Java, the "print = run_process" is to read its results that are stored in variable, and "run_process" is to delete the temporary folder that is used to save the output file. Meanwhile, in Source Code 3, the "#run spark" section is to run Spark. In Fig. 4, as an instance, "HADOOP_BIN jar NBMapReduce.jar NBCDriver Urgent,No,No /user/ubuntu/nb-input/dataset.txt /user/ubuntu/nb-output/+output_dir" is input here. During the data processing, it is possible to observe the log on the terminal that is running the django server. The output readings will also be displayed, including the temporary folder.

(a)



(b)



(c)

Figure 4. Terminal, Web and Mobile that are running the django server

3 Results and Discussion

Testing the number of training data in the classification program using Naïve Bayes with Hadoop which was run was carried out 4 times (50%, 60%, 70%, and 80% training

data of the total 12,960 data) where each experiment had the same 5 types of test data. The test data utilized can be observed in Table 1.

Table 1. Test Results of the Number of NB Training Data on Hadoop

Experiment number *i*	Number of Training Data	Test Data number *j*	Execution Time (*s*)	Average Execution Time (*s*)
1	6480	1	3	3.2
		2	3	
		3	3	
		4	3	
		5	4	
2	7776	1	3	3
		2	3	
		3	3	
		4	3	
		5	3	
3	9072	1	3	3
		2	3	
		3	3	
		4	3	
		5	3	
4	10368	1	3	3.4
		2	3	
		3	4	
		4	3	
		5	4	

The results of testing the number of training data for the Naïve Bayes classification on Hadoop displayed in Table 1 produce an average value of execution time that is not much different. Of the four experiments, the shortest execution time was generated in the 2nd and 3rd experiments even though both trials had different number of training data. The resulting execution time was, however, the same. From these experiments, it can be suggested that the number of training data has no significant effect on the execution time when using Hadoop with MapReduce.

Table 2. Test Results of the Number of NB Training Data on Spark

Trial number *i*	Number of Training Data	Total of Test Data	Test error	Accuracy = (1- Test Error)	Total Execution Time (*s*)
1	6364	6596	0.150849	0.849151	10.95
2	12479	481	0.116424	0.883576	6.87
3	12640	320	0.140625	0.859375	10.74
4	12738	222	0.193694	0.806306	9.11
5	12841	119	0.117647	0.882353	8.61

From the test results shown in Table 2, the obtained result of the 2nd experiment

producing the shortest execution time and smallest error test value indicated that the classification can be considered good because the error rate was only 0.116424 or equivalent to the accuracy value of 0.883576, with training data of 12479 and test data of 481. Meanwhile, the longest execution time was generated in the 1st experiment with a total execution time of 10.95 seconds. The thing that causes the execution time to run longer is the number of test data that exceeds the training data so that the classification process runs longer.

4 Conclusion

Based on the test results, the number of training data and test data for student enrollment classification in Nursery School using Naïve Bayes and Tools Big Data (Hadoop and Spark), there are some conclusions to draw. First, Naïve Bayes algorithm can solve problems on student enrollment classification in Nursery School. This is carried out using the MapReduce technique in the Naïve Bayes classification. Second, the smallest error value and the shortest execution time are generated by experiments with the comparison of training data and test data of 0.25:0.01 respectively with an error value of 0.116424 or equivalent to the accuracy value of 0.883576 (88.3576%) and an execution time of 6.87 seconds. Then, as a suggestion for the next research, it is important to attempt to perform multi-node testing, on/off node on Hadoop or Spark, which will be more precise and detailed in calculating the computation time and the accuracy results as a measure of performance as well.

References

1. Juneja, P., and Kaur, P., (2019). "Software Engineering for Big Data Application Development: Systematic Literature Survey Using Snowballing," 2019 International Conference on Computing, Power and Communication Technologies (GUCON), NCR New Delhi, India, 2019, pp. 492-496.
2. Hui, Y., and Zesong, L., (2019). "Research on Real-time Analysis and Hybrid Encryption of Big Data," 2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 2019, pp. 52-55, doi: 10.1109/ICAIBD.2019.8836992.
3. Gunaratna, K., Anderson, P., Ranabahu, A., and Sheth, A., (2010). "A Study in Hadoop Streaming with Matlab for NMR Data Processing," 2010 IEEE Second International Conference on Cloud Computing Technology and Science, Indianapolis, IN, 2010, pp. 786-789, doi: 10.1109/CloudCom.2010.70.
4. Hiranandani, P., Pilli, E. S., Chand, N., Ramakrishna, C., and Gupta, M., (2018). "Big Data Analytics Using Multi-Classifier Approach with Rhadoop," 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, 2018, pp. 478-484, doi: 10.1109/CONFLUENCE.2018.8442876.
5. Khader, M., Awajan, A. and Al-Naymat, G., (2018). "The Effects of Natural Language Processing on Big Data Analysis: Sentiment Analysis Case Study," 2018 International Arab Conference on Information Technology (ACIT), Werdanye, Lebanon, 2018, pp. 1-7, doi: 10.1109/ACIT.2018.8672697.
6. Herraiz, I., (2018). “Notebooks are not enough: how to deliver machine learning products without getting killed”. https://www.bigdataspain.org/2018/talk/notebooks-are-not-enough-how-to-deliver-machine-learning-products-without-getting-killed/ accessed on July 30, 2020.
7. Miao, K., Li, J., Hong, W. & Chen, M. (2020). “A Microservice-Based Big Data Analysis Platform for Online Educational Applications”. Annual Review of Anthropology, 2020, ["6929750"]. Available from: https://doi.org/10.1146/annurev.anthro.33.070203.144008
8. Roy, S., et al., (2017). "IoT, big data science & analytics, cloud computing and mobile app based hybrid system for smart agriculture," 2017 8th Annual Industrial Automation and

Electromechanical Engineering Conference (IEMECON), Bangkok, 2017, pp. 303-304, doi: 10.1109/IEMECON.2017.8079610.
9. Dabek, F., (2016). "Leveraging Big Data to Provide a Web Service That Provides the Likelihood of Developing Psychological Conditions after a Concussion," 2016 IEEE International Conference on Mobile Services (MS), San Francisco, CA, 2016, pp. 160-165, doi: 10.1109/MobServ.2016.32.
10. Andy a., (2016). "Cloud Computing Part 5: SaaS (Software as a Service)". https://andypi.co.uk/2016/05/23/cloud-computing-part-5-saas-software-as-a-service/ accessed on July 30, 2020.
11. Naik, P. (2016). "MLHadoop". https://github.com/punit-naik/MLHadoop/tree/master/Naive_Bayes_Classifier_MapReduce accessed on October 1, 2016.
12. Gan, K. L., (2020). "Vector - Outdoor Club Games and Recreational Activities. Stick figure depict outdoor games lawn bowling, canoe, archery, horse riding, roller coaster, wall climbing, water park, swimming pool, and golf course". https://www.123rf.com/photo_81783678_stock-vector-outdoor-club-games-and-recreational-activities-stick-figure-depict-outdoor-games-lawn-bowling-canoe-.html accessed on March 8, 2020.
13. Putra, N. A., Putri, A. T., Prabowo, D. A., Surtiningsih, L., Arniantya, R., Cholissodin, I. (2017). "Klasifikasi Sepeda Motor Berdasarkan Karakteristik Konsumen Dengan Metode K-Nearest Neighbour Pada Big Data Menggunakan Hadoop Single Node Cluster". Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) FILKOM UB Vol. 4 No. 2, 81-86.
14. Naveen, N. (2020). "Hadoop MapReduce – Key Features & Highlights". https://intellipaat.com/blog/tutorial/big-data-and-hadoop-tutorial/hadoop-mapreduce-key-features-highlights/ accessed on March 8, 2020.
15. stackchief. (2017). "MapReduce Quick Explanation". https://www.stackchief.com/blog/MapReduce%20Quick%20Explanation accessed on March 8, 2020.
16. Kodžoman, V. (2019). "The hidden cost of shuffle – MapReduce". https://datawhatnow.com/mapreduce-shuffle-sort/ accessed on March 8, 2020.
17. Lee, D. (2018). "RaspPi-Cluster". https://github.com/daviddwlee84/RaspPi-Cluster/blob/master/Notes/Distributed_Computing/MapReduce.md accessed on March 8, 2020.
18. Cholissodin, I., Riyandani, E. (2016). "Analisis Big Data". Fakultas Ilmu Komputer (Filkom), Universitas Brawijaya (UB), Malang.
19. Maryamah, M., Asikin, M. F., Kurniawaty, D., Sari, S. K., Cholissodin, I. (2016). "Implementasi Metode Naïve Bayes Classifier Untuk Seleksi Asisten Praktikum Pada Simulasi Hadoop Multinode Cluster". Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) FILKOM UB Vol. 3 No. 4, 273-278.
20. Cholissodin, I. and Supianto, A. A., "Enhancement Full Open Source Hadoop Distribution Universal Big Data Up Projects (UBig) From Education To Enterprise," 2019 International Conference on Sustainable Information Engineering and Technology (SIET), Lombok, Indonesia, 2019, pp. 90-93, doi: 10.1109/SIET48054.2019.8986040.