Implementation of Big Data Analytics for Machine Learning Model Using Hadoop and Spark Environment on Resizing Iris Dataset

1st Tresna Maulana Fahrudin
Department of Data Science
Universitas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
tresna.maulana.ds@upnjatim.ac.id

2nd Prismahardi Aji Riyantoko
Department of Data Science
Universitas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
prismahardi.aji.ds@upnjatim.ac.id

3rd Kartika Maulida Hindrayani
Department of Data Science
Universitas Pembangunan Nasional
"Veteran" Jawa Timur
Surabaya, Indonesia
kartika.maulida.ds@upnjatim.ac.id

2022 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS) | 978-1-6654-7327-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICIMCIS56303.2022.10017465

***Abstract*— The concept of Big Data to refer to huge volumes of data and attributes, but data samples through the use of a diverse set of features gathered from various sources. A significant amount of time is spent constructing a pre-processing workflow and an analysis process that make possible impactful for machine learning. Big Data analytics is being driven by the need to process Machine Learning data, actual real-time processing, and graphics processing. Hadoop and Spark, both accessible data warehousing frameworks that allow for the distribution and computation of massive datasets across several clusters of computer nodes, are the most efficient prospects for Big Data analysis in a distributed setting. To test the ability of these Big Data Tools, this research use Iris dataset as experimental data which is resized to a larger file. Multinomial Naive Bayes algorithm was employed to create a classification model for Iris flowers using Spark Machine Learning Library. The experimental result reported that there is a difference in accuracy and execution time during testing machine learning performance in Hadoop. The experiment given the best performance used Iris dataset is resized to 148 MB consisting of 5,184,000 samples, the model accuracy reached 95.32% with an execution time of 1 minute 4 seconds. The increase in the number of samples in the dataset is also positively correlated with increasing execution time. However, execution time is relatively cheap in the Hadoop Environment.**

Keywords—big data analytics, machine learning, hadoop, spark, iris dataset

I. Introduction

Big Data is directly related to an increase in the peak in various data streams as new technologies are gradually deployed. Knowledge is increasingly widely obtainable than it ever has been, thanks to the rise of the internet, and the consumption of social platforms, phone app, connected, and demodulated things is increasing at an alarming rate [1]. Big Data Analytics are methods for analyzing and developing Big Data in the context of strategic planning. Data mining is a subset of big data analysis that seeks to discover the relationship between previously unknown aspects of a dataset by employing a variety of field approaches such as machine learning algorithm, database, statistical method, and mathematics formulations [2]. Data analytics approach provides both granted and interpreted technology in a variety of domains for future predictions. [3][4]. One of the most critical data volumes is the ability to handle a large amount of complex content from an increasing number of different and autonomous sources. Quite a companies were using the concept "Big Data" to refer to huge volumes of data and attributes, but data samples through the use of a diverse set of features gathered from various sources have also been referred to as Big data [5][6].

A significant amount of time is spent constructing a pre-processing workflow and an analysis process that make possible impactful for machine learning. Data pre-processing identifies an issues such as data redundancy, instability, noise, variability, difficulty, unsupervised machine learning, and transformation. Human knowledge is widely used especially for data pre-processing and planning, in addition to the availability of a variety of alternative methods. Sophisticated data interpretations do not apply to large-scale datasets, rendering processing ineffective. As a result, massive amounts of data identify the opportunity to reduce reliance on human insight by drawing from larger, more complicated and difficult, and occasionally improved data sets [7].

The potential for heterogeneous data to be used to amount of coverage machine learning techniques to unique varieties of market opportunities and behaviors also seems to be greater than ever before, but their legitimacy is mostly questioned [8][9]. According to the source, huge data provides an exceptional amount of informative depth, but traditional machine learning is hindered by the enormous number of variables. All are getting bigger and more intricate, necessitating extensive research and advances in machine learning [10]. Because learning algorithms are incredibly strong and can conduct continuous learning, which reduces the need for human contact, machine learning will swiftly displace multiple people's employment in the future [11].

Big Data analytics is being driven by the need to process Machine Learning data, actual real-time processing, and graphics processing. Hadoop and Spark, both accessible data warehousing frameworks that allow for the distribution and computation of massive datasets across several clusters of computer nodes, are the most efficient prospects for Big Data analysis in a distributed setting [12]. Hadoop, the main software that forms the basis of an ecosystem consisting of software that works together. Primarily, as a system for processing very large volumes of data. Besides Hadoop, there are Hadoop Distributed File System (HDFS) that provide high throughput access to application data. Apache Spark is useful as an open-source unified analytics engine for large-scale data processing.

Therefore, the research presents the performance of Hadoop and Spark as Big Data analytic environment. We use Hadoop and Spark to solve big data analytics and employee the machine learning model using Multinomial Naïve Bayes to solve classification task on resizing Iris dataset. The standard size of benchmark Iris dataset is in kilobytes. However, the dataset is resized to up to megabytes in the experiment. The purpose is to test the performance of the big data environment and to evaluate the accuracy, execution time of different file sizes and validation sampling.

II. Related Works

Despite the fact that several applications are attempting to run Big data including a variety of existing approaches that managed datasets, in this section two, the applications that were chosen to be reviewed and determined the conclusions by authors in recent years, more or less every research in below for one implementation from big data, machine learning, Hadoop, and Spark, which also implementing several algorithms that performed inside applications to represent the obvious impact of these conceptual methods.

Ilham Kusuma, et al. focused on the effectiveness of smart K-means throughout the environment of Hadoop using Spark [13]. The idea behind using Spark to develop smart K-means is that many types of data, such as genomes, are relatively large and continue to grow in size, making it relatively simple to expand the Hadoop environment. The open-source design that aids various computations in both Map-reduce and Hadoop. In the Big Data mining process, the spark design is scalable. In place of the standard Resilient Distributed Dataset, the design included data batching (RDD). Using the first data RDD, compare its specification to the implementation. According to experience, data batch implementation is faster than first RDD implementation.

Anjuman Prabhat and Vikas Khullar employed the method of machine learning classifiers using Naïve Bayes and logistics-type were utilized in this work to cope with mission challenges that included Twitter comments [14]. They also included Hadoop and Mahout in the classifier. To improve the efficiency of the experiment, an extra module for instance the observation controller is inserted. A further examination of logistics regression analysis yields 10.1% and delivers 4.34% more accuracy for the same dataset scale (sample size). This article contains some supplementary language that describes tweets as a potential future job. This may boost categorization performance by combining text and graphics. Bi-gram, trigram, and other formalized forms may be more accurate.

TABLE I. The Comparison Between Related Works and Proposed Work in Big Data Research

No.	Author(s)	Dataset	Method and Tools	Evaluation
1.	I. Kusuma, et al. [13]	▪ First dataset has characteristic of 5 features and each feature has 5 peaks ▪ The second dataset has. characteristic of 10 features and it is created from 10 different centroids	▪ K-Means Based on Spark for Big Data Clustering ▪ Cluster has 4 slave node and one master node ▪ Every node utilizes with Intel Core i7 and RAM 32 GB	• Speed up computational time in big data problem reach 58.4-3075.2 seconds • Higher silhouette value than original k-means using synthetic data reach 0.628 - 0.7476
2.	A. Prabhat, V. Khullar. [14]	▪ Real time twitter with two categories: positive and negative reviews (6 MB)	▪ Naïve Bayes and Logistic Regression ▪ Hadoop 2.7.1 and Mahout 0.9 ▪ Single node with Intel Core i3 and RAM 4 GB	• Accuracy of Naïve Bayes reach 66.67% and Logistic Regression reach 76.76% • Computational time of Naïve Bayes reach 15732 mile-seconds and Logistic Regression reach 3689 mile-seconds
3.	I. R. Prabaswara, R. Saputra. [15]	▪ Mapping of dengue fever incidence based on twitter data in Southeast Asia (4.056.690 tweets)	▪ Visualization of dengue fever ▪ Hadoop 3.1.2 and Spark 2.4.0 ▪ Cluster has one slave node and one master node ▪ Every node with Intel Core i7, RAM 16 GB in master-node and 8 GB in slave-node	• The minimum execution time reach 5,3 minutes • The optimal allocation of memory is 3 GB and maximum memory scheduler is 4 GB
4.	V. Suriya Narayanan, et al. [16]	▪ Protein interaction problem using graphs and its semantic representation	▪ Large scale distributed graphs using Apache Spark ▪ Word2Vec language for model vocabulary	• Achieved approximately 97% accuracy using a 128-dimensional embedding as compared to around 95% using a 2-dimensional embedding
5.	T. M. Fahrudin, et al. (Proposed Research)	• Resizing Iris Dataset in 5 KB - 148 MB	▪ Apache Hadoop 3.2.1 ▪ Apache Spark 3.0.0 ▪ Machine Learning Model using Multinomial Naïve Bayes ▪ Single node with Intel Core i3 and RAM 4 GB	• Accuracy and execution time of building classification model based on resizing Iris dataset (5 KB – 148 MB)

Irwan Rizqi Prabaswara and Ragil Saputra carried out research about trend mapping on the fever data Dengue Hemorrhagic Fever (DHF) from social media twitter. This research desires to construct a visualization of data obtained from twitter with using Hadoop and spark in tracking the growth of dengue in the Southeast Asia area [15]. The findings of trend mapping reveal that there is a substantial association between twitter data and the original data on dengue incidence collected from WHO. This research also investigated the performance of Hadoop and Spark. The larger the memory allocation of the executor that is applied and the larger and similar the maximum allocation of the memory scheduler applied to each node, the shorter the time required to complete the task. However, at some point Hadoop and Spark configurations hit a breaking point, so if the allocation is increased it produces the same result.

In the other works, V. Suriya Narayanan, et al. demonstrated how and when to use Apache Spark to implement a distributed learning algorithm that operates through large graphs [16]. It is technologically applicable making it easy and applicable for large-scale deployments, and it has a fairly clear set of powerful predictive capabilities. It is advantageous for graph-based developers to use comprehensible word embedding as physical world focuses. The use of the Spark design and optimization for the efficient loading of distributed edge records and in-graph frameworks. The application Word2Vec then used Spark (MLlib) fetch node interconnection with a random walk according to each graph node. Excellent results have been achieved. Typically, 70% of the data could be used to visualize and reinforce learning. However, this approach uses only 1% of the dataset to training the models. By implementing this model, they help in achieving a prediction precision of close to 95%.

Table I show the comparison between related works and proposed works in Big Data research. Big Data tools and methods used in the research commonly are Hadoop, Spark, and Mahout. However, what makes the difference is the dataset tested using the Iris dataset which has been resized to a larger size. We test the performance of big data when faced with exponentially growing data sizes. Then, we evaluate the accuracy and execution time.

III. System Design

In this chapter, the system design implemented in the proposed research will be explained. It will be discussed regarding Iris dataset, Hadoop, Hadoop Distributed File System (HDFS), Spark and MLlib, Multinomial Naïve Bayes, and the evaluation model. Fig.1 show the system design of proposed research.

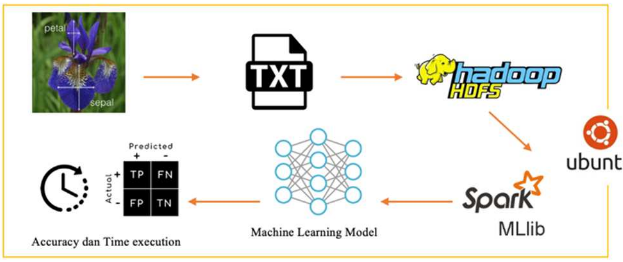


Fig. 1. Sytem design of proposed research

A. Iris Dataset

The Iris dataset is a popular dataset used in the learning and experiment about data science. The dataset stores tabular data about flower such as sepal length and width, petal length and width. Types of flowers are classified into three categories namely Setosa, Versicolour, and Virginica. The dataset was chosen because it is benchmark dataset that has been widely used and popular. The type of the features is continuous, while the categories are discrete. The number of instances on Iris dataset is 150 which each category is proportional to 30 instances. Almost all studies using iris datasets by researcher achieve good accuracy above 90%.

B. Hadoop

Hadoop is a big data technology product of the open-source Apache software [17]. The function of Hadoop is to solve the problem of large amounts of data and computing with a set of computer networks. Hadoop has an architecture with three components, namely HDFS (Hadoop Distributed File System), MapReduce, and YARN. The characteristics of Hadoop include:

- Hadoop is optimally used to handle large amounts of structured, semi-structured, and unstructured dataset.
- Hadoop replicates data across multiple computers (clustering). If one computer has a problem, the data can be processed from one of the other computers that are still alive.
- The Hadoop process is a batch operation handling a very large amount of data, so the response time is not real time.

C. Hadoop Distributed File System

Hadoop Distributed File System (HDFS) is part of Hadoop that functions as a consistent data storage place [18]. An important process in HDFS is data replication to different partitions by massive and parallel. The replication is balanced in different blocks. Distributed file systems designed for fault-tolerant file systems can run on multiple servers with low-cost specifications. HDFS is designed to support applications with large data sets, even terabytes of files. When a file is processed via HDFS, it is split into smaller parts and then the smaller part of the file is distributed across multiple nodes in the cluster system thus enabling parallel processing.

D. Spark and MLlib

Spark is an open-source framework that is suitable for use in iterative algorithmic processes [19]. Spark allows connecting analytics engines with high-scale data processing. MLlib is part of Spark with Machine Learning libraries. The goal is to make machine learning easier and more scalable. MLlib has many uses including regression, classification, clustering, can perform linear and statistical algebraic calculations and handle pipelines.

E. Multinomial Naïve Bayes

Multinomial Naïve Bayes event in the model is a multinomial vector $(P_i, \ldots.., P_n)$ where P_i is the probability that event i will occur. Vector $(x_1, \ldots.., x_n)$ in the form of a histogram. x_i is the total number of events occurring within a certain range. The Multinomial Naïve Bayes formula follows equation 1.

$$\text{p}\,(\text{x} \mid C_k) = \frac{(\sum_{i=1}^{n} x_I)!}{\prod_{i-1}^{n} x_I!} \prod_{i=1}^{n} Pki^{x_i} \tag{1}$$

F. Evaluation Model

The evaluation of the model that will be used is accuracy and execution time. Accuracy is how accurate the predictions made by the model are with the actual predictions. The accuracy formula follows equation 2.

$$\text{CR} = \frac{C}{A} \tag{2}$$

Where:
CR : The correct rate
C : The number of samples recognized correctly

A : The number of all samples.

While the execution time is recorded from start time to end time during the running code script program. The execution time formula follows equation 3.

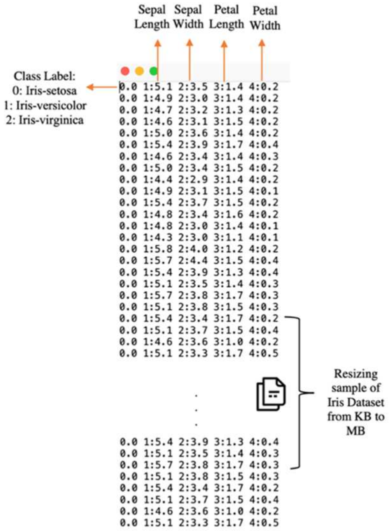
$$\text{Execution time} = \mathit{End\ time} - \mathit{Start\ time} \quad (3)$$

IV. Experimental Result and Discussion

In this section will discussed about dataset preparation, configuration Hadoop and Spark environment, building model of Multinomial Naïve Bayes, evaluation model and execution time.

A. Dataset Preparation

Iris dataset consists of 4 features including Sepal Length (cm), Sepal Width (cm), Petal Length (cm), and Petal Width (cm), while 3 species as class label including Iris-setosa, Iris-versicolor, and Iris-virginica which the total number of samples for each class is 50 proportionally. If the experiment uses an iris dataset of 150 samples (5 kb) to test machine learning models into a big data environment, this will certainly not have much impact. Therefore, the iris dataset in this experiment was resized to 72 MB of 2,592,000 samples and 145 MB of 5,184,000 samples. Fig. 2 show the resizing samples of Iris dataset.



```
0.0 1:5.1 2:3.5 3:1.4 4:0.2
0.0 1:4.9 2:3.0 3:1.4 4:0.2
0.0 1:4.7 2:3.2 3:1.3 4:0.2
0.0 1:4.6 2:3.1 3:1.5 4:0.2
0.0 1:5.0 2:3.6 3:1.4 4:0.2
0.0 1:5.4 2:3.9 3:1.7 4:0.4
0.0 1:4.6 2:3.4 3:1.4 4:0.3
0.0 1:5.0 2:3.4 3:1.5 4:0.2
0.0 1:4.4 2:2.9 3:1.4 4:0.2
0.0 1:4.9 2:3.1 3:1.5 4:0.1
0.0 1:5.4 2:3.7 3:1.5 4:0.2
0.0 1:4.8 2:3.4 3:1.6 4:0.2
0.0 1:4.8 2:3.0 3:1.4 4:0.1
0.0 1:4.3 2:3.0 3:1.1 4:0.1
0.0 1:5.8 2:4.0 3:1.2 4:0.2
0.0 1:5.7 2:4.4 3:1.5 4:0.4
0.0 1:5.4 2:3.9 3:1.3 4:0.4
0.0 1:5.1 2:3.5 3:1.4 4:0.3
0.0 1:5.7 2:3.8 3:1.7 4:0.3
0.0 1:5.1 2:3.8 3:1.5 4:0.3
0.0 1:5.4 2:3.4 3:1.7 4:0.2
0.0 1:5.1 2:3.7 3:1.5 4:0.4
0.0 1:4.6 2:3.6 3:1.0 4:0.2
0.0 1:5.1 2:3.3 3:1.7 4:0.5
              .
              .
              .
0.0 1:5.4 2:3.9 3:1.3 4:0.4
0.0 1:5.1 2:3.5 3:1.4 4:0.3
0.0 1:5.7 2:3.8 3:1.7 4:0.3
0.0 1:5.1 2:3.8 3:1.5 4:0.3
0.0 1:5.4 2:3.4 3:1.7 4:0.2
0.0 1:5.1 2:3.7 3:1.5 4:0.4
0.0 1:4.6 2:3.6 3:1.0 4:0.2
0.0 1:5.1 2:3.3 3:1.7 4:0.5
```

Fig. 2. Resizing samples of Iris dataset

B. Configuration Hadoop and Spark Environment

Before implementing big data analytics, there are several tools and software that need to be prepared. The following are the specifications of the device used in the experiment:

- Processor : Intel core i3
- RAM: 4.00 GB
- System type: 64-bit Operating System, x64-based processor
- OS: Windows 10 Home Single Language

While the specifications of the software used in the experiment as follow:

- Oracle VM VirtualBox Manager
- Ubuntu 64-bit: Virtual hard disk 15 GB
- Open Java Development Kit (OpenJDK)
- Apache Hadoop 3.2.1 (stable version)
- Apache Spark 3.0.0 (stable version)

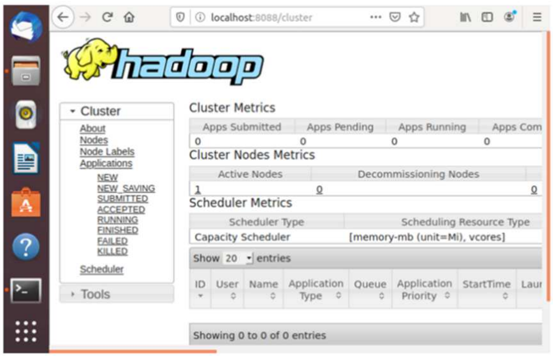
To install and configure Hadoop on Ubuntu, first step is install Java via Software Development Kit Manager (SDKMAN), check the path $JAVA_HOME, and then use JDK version 8.0.242.hs-adpt. Unzipped Apache Hadoop 3.2.1 (file extension *.tar.gz) and will extract several files such as start-all.sh, stop-yarn.sh, workers.sh, start-dfs.sh, httpfs.sh, stop-balancer.sh, and etc. Hadoop directory needs to have permissions set to change the ownership of its username by using the command "chown username:username -R name_directory". Bash shell script (~/.bashrc) also set Hadoop home directory with a variable named HADOOP_HOME and set the binary files in Hadoop home directory which is located in HADOOP_HOME/bin. To check Hadoop configuration is running well or not through the command "hadoop version". Master files such as hadoop-env.sh also need to be set the Java home directory in path so that Yarn, HDFS, MapReduce, and others can run correctly.

To install and configure Spark on Ubuntu, unzipped Apache Spark 3.0.0 (file extension *.tar.gz) and will extract several libraries such as Machine Learning Library (MLlib), R, Kubernetes, and etc. Spark directory needs to have permissions set to change the ownership of its username. Bash shell script (~/.bashrc) also set Spark home directory with a variable named SPARK_HOME and set the binary files in Spark home directory which is located in SPARK_HOME/bin. To check Spark configuration is running well or not through the command "spark-shell --version".

After installing Hadoop and Spark, the next step is Hadoop Distributed File System (HDFS) configuration. There are several files that must be configured in the /opt/hadoop/etc/hadoop/ directory, including:

- **core-site.xml**: set the default file system, localhost address and port
- **hdfs-site.xml**: set directory locations of name node and data node
- **mapred-site.xml**: set the MapReduce framework name
- **yarn-site.xml**: set manage node manager and handling the shuffle on MapReduce

The final step is formatting HDFS, make sure there is no important data in HDFS because the data will be deleted. Formatting HDFS with command "hdfs namenode -format -force". Then boot HDFS with the commands "start-dfs.sh && start-yarn.sh". To check HDFS is running correctly or not use the command "jps", the terminal shows the information the currently active services such as the Java Virtual Machine Process Status Tool (JPS), ResourceManager, NodeManager, DataNode, NameNode, and SecondaryNameNode. Hadoop provides monitoring dashboards for cluster node metrics, scheduler metrics, nodes, datanodes, startup progress and more as shown in Fig. 3.



localhost:8088/cluster

Cluster

- About
- Nodes
- Node Labels
- Applications
  - NEW
  - NEW_SAVING
  - SUBMITTED
  - ACCEPTED
  - RUNNING
  - FINISHED
  - FAILED
  - KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Com
0	0	0	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	
1	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type
Capacity Scheduler	[memory-mb (unit=Mi), vcores]

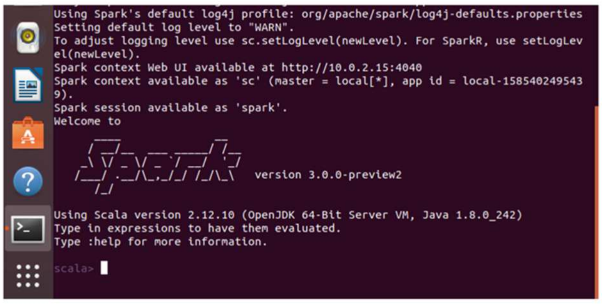
Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	Laur

Showing 0 to 0 of 0 entries

Fig. 3. Cluster metrics dashboard monitoring on Hadoop

While to start Spark by using the command "spark-shell" as shown in Fig. 4.



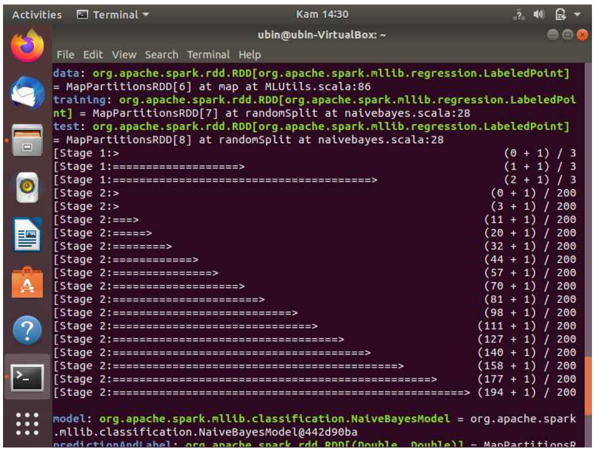
```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLev
el(newLevel).
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-158540249543
9).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.0.0-preview2
      /_/

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_242)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Fig. 4. Spark-shell on Spark 3.0.0

After Hadoop and Spark are properly configured, machine learning can be implemented.

C. Building Model of Multinomial Naïve Bayes

Spark provides alternative programming languages to implement Machine Learning such as Scala, Java, and Python The experiment used Scala programming language and importing the classification model (supervised learning) from MLlib such as Multinomial Naive Bayes algorithm. The Iris dataset is loaded into Scala, then splitting the dataset is 60% for training data and 40% for testing data. The performance evaluation of classification model based on accuracy matrix and execution time. Fig. 5 show the running Multinomial Naïve Bayes model using Scala on Spark, the program running the code script execution until the stage completely.



```
data: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint]
= MapPartitionsRDD[6] at map at MLUtils.scala:86
training: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoi
nt] = MapPartitionsRDD[7] at randomSplit at naivebayes.scala:28
test: org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint]
= MapPartitionsRDD[8] at randomSplit at naivebayes.scala:28
[Stage 1:>                                                          (0 + 1) / 3
[Stage 1:===================>                                       (1 + 1) / 3
[Stage 1:=======================================>                   (2 + 1) / 3
[Stage 2:>                                                        (0 + 1) / 200
[Stage 2:>                                                        (3 + 1) / 200
[Stage 2:===>                                                    (11 + 1) / 200
[Stage 2:=====>                                                  (20 + 1) / 200
[Stage 2:========>                                               (32 + 1) / 200
[Stage 2:============>                                           (44 + 1) / 200
[Stage 2:===============>                                        (57 + 1) / 200
[Stage 2:===================>                                    (70 + 1) / 200
[Stage 2:======================>                                 (81 + 1) / 200
[Stage 2:===========================>                            (98 + 1) / 200
[Stage 2:==============================>                        (111 + 1) / 200
[Stage 2:==================================>                    (127 + 1) / 200
[Stage 2:======================================>                (140 + 1) / 200
[Stage 2:===========================================>           (158 + 1) / 200
[Stage 2:================================================>      (177 + 1) / 200
[Stage 2:=====================================================> (194 + 1) / 200

model: org.apache.spark.mllib.classification.NaiveBayesModel = org.apache.spark
.mllib.classification.NaiveBayesModel@442d90ba
predictionAndLabel: org.apache.spark.rdd.RDD[(Double, Double)] = MapPartitionsR
```

Fig. 5. Running Multinomial Naïve Bayes Model using Scala on Spark

D. Evaluation Model and Execution Time

TABLE II. THE EXPERIMENTAL RESULT OF BIG DATA ANALYTICS FOR MACHINE LEARNING MODEL USING MULTINOMIAL NAÏVE BAYES IN HADOOP AND SPARK ENVIRONMENT ON RESIZING IRIS DATASET

No.	Size	Number of Samples	Validation Sampling	Accuracy	Execution Time
1.	5 KB	150	Pecentage Split (60:40)	65%	1.27 seconds
2.	8 MB	199,729	Pecentage Split (60:40)	96.04%	2.52 seconds
3.	14 MB	399,458	Pecentage Split (60:40)	95.31%	5.29 seconds
4.	27 MB	798,916	Pecentage Split (60:40)	95.27%	10.95 seconds
5.	87 MB	2,591,848	Pecentage Split (60:40)	95.31%	38.16 seconds
6.	148 MB	5,184,000	Pecentage Split (60:40)	95.32%	1 minute 4 seconds

Table II show when the number of samples in Iris dataset was resized from KB to MB, there is a difference in accuracy and execution time during testing machine learning performance in Hadoop. If the test used the original iris dataset of 5 KB consisting of 150 samples, the accuracy of the Multinomial Naïve Bayes model only reached 65% with an execution time of 3.66 seconds. On other hand, the test used Iris dataset that has been resized to 87 MB consisting of 2,592,848 samples, the accuracy of the model reached 95.31% with an execution time of 38,16 seconds. If the Iris dataset is resized to 148 MB consisting of 5,184,000 samples, the model accuracy reached 95.32% with an execution time of 1 minute 4 seconds. Table III show the performance of Multinomial Logistic Regression with the same size file, the number of samples, and validation sampling.

TABLE III. THE EXPERIMENTAL RESULT OF BIG DATA ANALYTICS FOR MACHINE LEARNING MODEL USING MULTINOMIAL LOGISTIC REGRESSION IN HADOOP AND SPARK ENVIRONMENT ON RESIZING IRIS DATASET

No.	Size	Number of Samples	Validation Sampling	Accuracy	Execution Time
1.	5 KB	150	Pecentage Split (60:40)	90%	3.66 seconds
2.	8 MB	199,729	Pecentage Split (60:40)	98.68%	12.29 seconds
3.	14 MB	399,458	Pecentage Split (60:40)	98.66%	25,73 seconds
4.	27 MB	798,916	Pecentage Split (60:40)	98.65%	55.78 seconds
5.	87 MB	2,591,848	Pecentage Split (60:40)	98.66%	2 minutes 9 seconds
6.	148 MB	5,184,000	Pecentage Split (60:40)	98,65%	4 minutes 11 seconds

Figure 6 and Figure 7 show that the Multinomial Logistic Regression has the highest accuracy, but is also followed by high execution time consumption, while the Naïve Bayes

Multinomial has the lowest execution time consumption, but reached a fairly good accuracy.

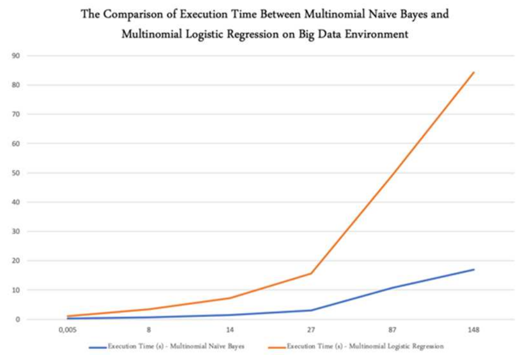


Fig. 6. The Comparison of Execution Time Between Multinomial Naïve Bayes and Multinomial Logistic Regression on Big Data Environment

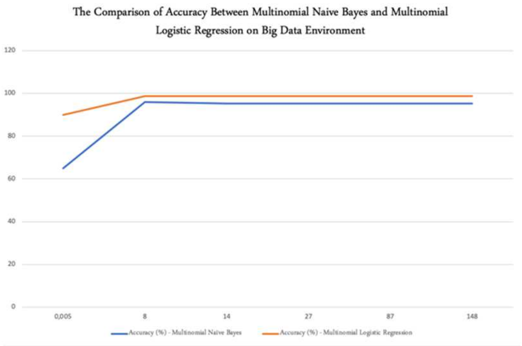


Fig. 7. The Comparison of Accuracy Between Multinomial Naïve Bayes and Multinomial Logistic Regression on Big Data Environment

The experimental results indicates that with the increase in the number of samples in the dataset is also positively correlated with increasing execution time. However, execution time is relatively cheap in the Hadoop Environment. The evaluation regarding the increase in accuracy needs to be investigated further because the Iris dataset is a benchmark dataset consisting of only 150 samples, while the iris dataset is resized in this experiment which allows the sample to be duplicated.

V. Conclusion

The implementations of big data analytics using Hadoop and Spark are configured and running well on resizing Iris dataset. Iris dataset load in Hadoop environment through Hadoop Distributed File System, while Spark provide several algorithms for classification, clustering, and regression. The experiment employed Multinomial Naïve Bayes to classify the Iris dataset. The experimental result reported that there is a difference in accuracy and execution time during testing machine learning performance in Hadoop. The experiment given the best performance used Iris dataset is resized to 148 MB consisting of 5,184,000 samples, the model accuracy reached 95.32% with an execution time of 1 minute 4 seconds. The increase in the number of samples in the dataset is also positively correlated with increasing execution time. However, execution time is relatively cheap in the Hadoop Environment. Further research needs to upgrade the device to experiment with a larger processor and RAM and the number of datasets that reach GB.

References

[1] B. Zerhari, A. A. Lahcen, and S. Mouline, "Big Data Clustering: Algorithms and Challenges," in Proceedings of the International Conference on Big Data, Cloud, and Applications (BDCA'15), pp. 1-6, 2015.

[2] S. W. Kareem, "Secure Cloud Approach Based on Okamoto-Uchiyama Cryptosystem," Journal of Applied Computer Science and Mathematics, vol. 14, no. 29, pp. 9-13, 2020.

[3] H. B. Patel and S. Gandhi, "A Review on Big Data Analytics in Healthcare using Machine Learning Approaches," in 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 84-90, 2018.

[4] S. Suthaharan, "Machine Learning Models and Algorithms for Big Data Classification," Integrated Series in Information Systems, vol. 36, pp. 1-12, 2016.

[5] V. Ajin and L. D. Kumar, "Big Data and Clustering Algorithms," in 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS), pp. 1-5, 2016.

[6] S. W. Kareem, R. Z. Yousif, S. M. J. Abdalwahid, and C. Science, "An Approach for Enhancing Data Confidentiality in Hadoop," Indonesian Journal of Electrical Engineering and Computer Science, vol. 20, no. 3, pp. 1547-1555, 2020.

[7] A. L'heureux, K. Grolinger, H. F. Elyamany, and M. A. Capretz, "Machine Learning with Big Data: Challenges and Approaches," Ieee Access, vol. 5, pp. 7776-7797, 2017.

[8] E. Hossain, I. Khan, F. Un-Noor, S. S. Sikander, and M. S. H. Sunny, "Application of Big Data and Machine Learning in Smart Grid, and Associated Security Concerns: A Review," IEEE Access, vol. 7, pp. 13960- 13988, 2019.

[9] M. Mohammadi, A. Al-Fuqaha, S. Sorour, M. Guizani, "Deep Learning for IoT Big Data and Streaming Analytics: A Survey," IEEE Communications Surveys and Tutorials, vol. 20, no. 4, pp. 2923-2960, 2018.

[10] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on Big Data: Opportunities and Challenges," Neurocomputing, vol. 237, pp. 350- 361, 2017.

[11] H. K. Tripathy, B. R. Acharya, R. Kumar, and J. M. Chatterjee, "Machine Learning on Big Data: A Developmental Approach on Societal Applications," in Big Data Processing Using Spark in Cloud: Springer, pp. 143- 165, 2019.

[12] Benlachmi, Y., Yazidi, A.E., Hasnaoui, M.L."A Comparative Analysis of Hadoop and Spark Frameworks Using Word Count Algorithm". International Journal of Advanced Computer Science and Applications. Vol. 12, No. 4, pp. 778-788, 2021.

[13] I. Kusuma, M. A. Ma'Sum, N. Habibie, W. Jatmiko, and H. Suhartanto, "Design of Intelligent K-Means based on Spark for Big Data Clustering," in 2016 International Workshop on Big Data and Information Security (IWBIS), pp. 89-96, 2016.

[14] A. Prabhat and V. Khullar, "Sentiment Classification on Big Data using Naïve Bayes and Logistic Regression," in 2017 International Conference on Computer Communication and Informatics (ICCCI), pp. 1-5, 2017.

[15] Prabaswara, I.R., Saputra, R. "Implementation of Hadoop and Spark for Analysis of The Spread of Dengue Hemorrhagic Fever based on Twitter Data," in IT Journal Research and Development (ITJRD), vol. 4, no. 2, pp. 164-171, 2020.

[16] V. S. Narayanan, V. B. Vijayakumar, S. R. Venkatraman, and P. K. Baruah, "Semantic Node Embeddings of Distributed Graphs using Apache Spark," in 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), pp. 709-713, 2016.

[17] O. Azeroual and R. Fabre, "Processing Big Data with Apache Hadoop in The Current Challenging Era of COVID-19," *Big Data Cognitive and Computing: MDPI.*, vol. 5, no. 1, pp.1-18, 2021.

[18] D. Veeraiah and J. N. Rao, "An Efficient Data Duplication System based on Hadoop Distributed File System," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, pp. 197–200, 2020.

[19] A. Mostafaeipour, A. Jahangard Rafsanjani, M. Ahmadi, and J. Arockia Dhanraj, "Investigating The Performance of Hadoop and Spark Platforms on Machine Learning Algorithms," *The Journal of Supercomputing.*, vol. 77, no. 2, pp. 1273–1300, 2021.