A Comparative study of Machine Learning Algorithms Using RDD Based Regression and Classification Methods

A.VETTRISELVI[1], N. GNANAMBIGAI[2*],
P. DINADAYALAN[3], S.SUTHA[4]
[1]Research scholar ,Bharathiyar University, Coimbatore, India
[2]Departmen of computer Science ,Indra Gandhi Arts and Science College, Puducherry, India
[3]Departmen of computer Science ,Kanchi Mamunivar Govt. Institute for Postgraduate Studies and Research, Puducherry, India
[4]Research scholar ,Bharathiyar University, Coimbatore, India

ABSTRACT
Today we live in the world of big data, where we find it difficult to store and process the data by the traditional devices. This paper main aim is to help the researchers and professional who are already familiar with machine learning but not experienced with MLlib package for both Regression and classification methods using RDD-based in big data. This paper provides a list of scenario. Firstly, we go on Hadoop environment and have to integrate and deploy spark on yarn, secondly we explore how to allocate resources dynamically executors cores and memory which improves the performance of spark application on YARN, Thirdly, we have implemented RDDs API in spark on YARN, which we evaluate throughout the experiments. Finally, we compare and evaluate few machine learning algorithms in spark using RDD-based regression and classification methods for Random forest, decision tree, gradient boosted tree, Logistic regression, Linear regression Ridge and Lasso Regression with SGD and LBFGS optimization Technique. The comparative study is performed to analyze their performance on the basis of defined parameters. Additionally, we also compare and analyze the performance of both Hadoop and spark framework using the tool HiBench 7.0 benchmark suite. The results clearly depict how spark give more promising outcome than Hadoop in terms of execution duration (i.e. speed) and throughput per node (i.e. node/bytes).
Keywords*: machine learning, RDD, spark, Hadoop, Hibench, Regression, Classification, MAE, RMSE.*

Date of Submission: 28-04-2021 Date of acceptance: 12-05-2021

I. BACKGROUND

In recent developments, cluster computing has combined with big data, therefore machine learning has been pushed to the top most position of computing. The aim of machine learning is to study, train and improve mathematical models, it is designed to work on large data sets and give accurate results that would take much longer to process by humans.

Understanding Big Data and Hadoop:

The term Big Data [2] is used to describe as the data that is huge in size and more complex that traditional database tools cannot store and process it efficiently. The main problem of traditional databases when it is combined with big data is that it uses a scale up architecture which means that we need to add more CPU cores, RAM and disk storage to existing servers, In Big data, Hadoop distributed file system adopt a scale out architecture. It helps to grow the storage capacity by continuously adding more small commodity servers. The Goal of this paper is to provide a comprehensive review of Hadoop and spark which is an open source data processing engine [1] [19], This paper will be useful for the reader who has basic knowledge about machine learning concepts and tools Additionally, in this paper, we explored Hibench and its working towards machine learning algorithms and the comparative study between Hadoop and spark based on their performances. This paper will be very useful for researcher, engineer or software managers.

The remainder of this paper will be organized as follows: The first section "Introduction to Hadoop "along with its feature and its components. The second section almost the related work done previously based on machine learning and Hibench. The third section is "Architecture of Hadoop and its services. The fourth section "Yarn Architecture and working of hdfs. The fifth section about Apache spark and its important concepts the sixth section explain about machine learning algorithms and in our experiments. The seventh section explains about the parameter metrics which is used to evaluate the ml algorithms. The eighth section explain about the optimization techniques used to solve the problem. The ninth section about HiBench benchmark suite and its workload. The tenth section about the spark MLlib package and Scala programming language used for our experiments. In the eleventh section, it shows the experimental and result evaluation and the datasets used for both machine learning and hibench working experiments. And finally the last sections gives the conclusion and future enhancement to be done.

II. RELATED WORKS

Machine Learning:

In this paper, the author (MO Hai,2017) experiment how to evaluate performance of two classification algorithm in spark mllib namely Random forest and naive Bayes. The experiments are performed based on dataset and clusters of different scale. The results show that RF reaches its peak when number of modes is less than 4 and decreases its scale up when the no of modes is greater than 4 [38].

Authors (Mehdi and Ahmad,2017) in this paper analyzed that the machine learning algorithms the author evaluate multiple common big data machine learning models. The author compared the performance between spark and weka based on various hardware and software configuration. The comparative study shown that the Apache spark Mllib is able to be faster in comparison with the weka components [26].

In this paper, the author (Neelam,2017) explored about Apache Hadoop based distributed environment. The author proposed a comparative study of binary classification methods such as decision tree, Gradient boosting tree and random forest tree to judge their performance. The results show that Random forest tree performs best among all the three algorithms for the considered dataset [59].

In this paper, the author (Zaharia and Franblin) introduces a new framework called spark RDD's called Resilient distributed datasets. The author analyzed that an RDD's is a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. And also spark can outperform Hadoop by 10 X in iterative machine learning jobs [63].

Authors (zaharia and Justin,) should that RDD's are expressive enough to capture a wide class of computations and provide coarse-grained transformation rather than fine-grained. The author implemented RDD in a system called spark to evaluate variety of user applications and benchmarks [40].

HiBench

In this paper, the author (Mo Hai Ahmed,2016) deals with the deployment of spark cluster. The author explored Hibench benchmark suite to compare the performance of spark cluster both as a service and conventional. The results depict the outcome in terms of time, effect and throughput [28].

In this paper, the author (Bo Huang,2010) introduce, HiBench, a new benchmark suite for Hadoop. The author evaluates and characterize Hadoop using Hibench, hdfs bandwidth, system resources and data access patterns [32].

Author (Blesson ,2017) ph. proposed play plug and play bench to simplify the setup and configuration of the benchmarking process. The paper further proved that most workload cost increase as the number of modes increase but with an advantage of lower execution times [49].

In this paper, the author(Maryam,2010) explored how virtualization in a distributed environment hinder the possibility of achieving the maximum performances [64].

In this paper, the author (Rui Han,2015) developed data generators which is capable of preserving the 4V's properties of big data and also implemented application specific workload [50].

Introduction to Hadoop:

Apache Hadoop [1] is an open source software for reliable scalable and distributed processing of large data sets across clusters of computers. Anything more than a single machine will technically constitute a cluster the HDFS is a distributed, scalable and portable file system written in java for the Hadoop framework.

Features of Hadoop and its components [2]:

Handle large data sets: it can handle large amount of data its file sizes ranging from gigabytes to terabytes.

Streaming data: Hadoop was designed for batch processing such as indexing, iterative processing, search and stream processing.

Data consistency: Apache Hadoop data files employ a write –once-read-many model. There are no data consistency issues with Hadoop file system, because only a single writer can write to a file at any time.

Batch, Interactive and Real-Time Data access with Hadoop			
Script Pig	In-Memory Spark	Machine Learning Spark ML	Scala cascading
YARN : Operating system for Hadoop (Hadoop cluster Resource Management)			
HDFS (Hadoop Distributed File System)			

Fig.1 Components of Big Data

Architecture of Hadoop and its services:
Hadoop cluster [61] is a collection of machines that uses the Hadoop software on the foundations of a distributed file system(HDFS) and a cluster resource manager(YARN). It Provides the processing framework for running not only map reduce but also other frameworks such at Tez and spark.
Map reduce:
Map reduce is a distributed processing framework. It consists of two primary phases the map phase and the reduce phase. The Map function 's job is to map the input data to sets of by/value pairs and the reduce function then taken these key/value pairs and produces the output.
Hadoop services:
The name node service runs on a master node and maintains the metadata relate to HDFS Storage, such as the file system directory and location of the files, If the client wants to read or write to hdfs, it must contact the name node service. The secondary name node must run on master mode on each cluster. It removes the burden of the name node by performing tasks such as updating the metadata file. The Data node kept contact with the name node and update all the changes that occur in the file system.
Yarn services:
HDFS has several services that run on both the master and worker nodes. The resource manager runs on the master nodes. They are responsible for allocating the cluster resources and scheduling jobs on the worker nodes. The Application master, this is a master service and there is one for each application. Each time a new application starth, the Resource Manager deploys a container running the application master on one of the cluster nodes. The Node manager services runs and manages tasks on the worker nodes. The node manager has contact with the Resource manager and update it status and health of the tasks they are running.
YARN Architecture:
It is a Framework for managing distributed applications executed on multiple machines within a network. It is a processing layer that manages all resources in a Hadoop cluster. Yarn that makes multiple processing frameworks to run on the same Hadoop cluster such as map reduce or spark [60]. It depends on a cluster wide Resource Manager. The Resource manager works together with a node manager that sums on every data node in the cluster. Every application that run on yard has an Application master is to work with the node manager to execute the tasks that are part of each application.
Hadoop Distributed File System:
HDFS is a distributed file system. It is designed for fast, fault-tolerant processing. HDFS enables users to store data in files. Which are split into multiple blocks. The default block based file system, where files are broken up into blocks. Data replication is one of the pillars of Hadoop, since it provides fault tolerance. Hadoop maintains multiple copies of data so it is hard to lose data stored in a cluster.

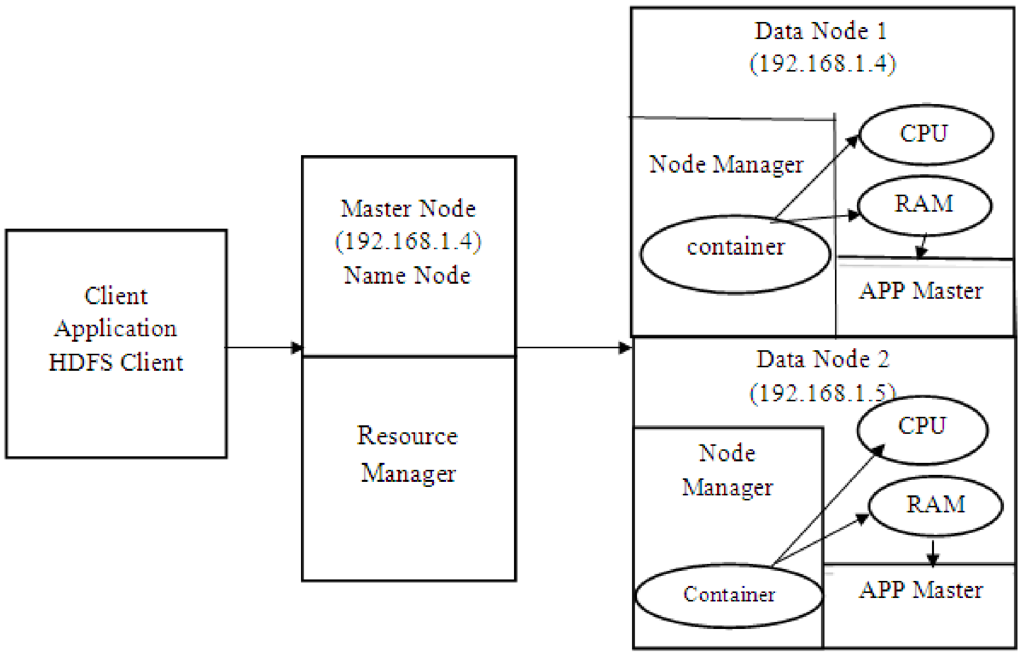

Fig 2. Yarn Architecture

Working of HDFS:
HDFS is a distributed fault tolerant file system, HDFS employs a write once record many access nodes for the files it stores. It stores data in hdfs files, each of which consists of a number of blocks. These blocks are replicated on multiple data nodes.
Client and HDFS Communication:
Application incorporate the hdfs client library and it's the client library that manages communication between the application on one hand and the name node and the data node on the other.
Name node and Data Node Communications:
Data node started and gets registered with the name node to let it know that its available to handle hdfs read and write operations. For every three seconds all data nodes periodically send a heartbeat to the name node. This heartbeat lets the name node know that it can send information such as block replication or deletion to the data nodes. If a data node fails to send its periodic heartbeat even after a long time, the name node will mark that data node as dead.
Data replication and Data storage:
Hadoop does not rely on a data protection mechanism such as RAID. It replicates data on multiple nodes for reliability. The default HDFS replication factor is 3. There are few Strategies to place the replicas across the data nodes in a cluster. The first replica is placed on the same nods where the client is running. Place the second replica on a data node in a randomly chosen rack that's different from the rack where the first mode was placed and place the third replica on a random data node on the remote rack not more than one replica of a block is placed on any mode and not more than two replicas are placed on any rack.

Spark:
Spark [8] is an open source computational framework and is widely considered the successor to the map reduce processes and analyzes huge amount of data using commodity servers. Spark was designed to overcome the inefficiency of the map reduce model in performing iterative computations. It supports a wide variety of workloads including batch processing, Streaming graphs and machine learning. In a Hadoop cluster, both spark and map reduce can be used side by side to provide distributed data processing.

Directed a cyclic graph:
Spark offers a 100x improvement in performance over map reduce. Spark can use memory-based computation. It is much faster than map reduce. Spark offers several features that map reduce does such as fault tolerance and scalability, Spark applications can be written in Scala, Python or Java, Spark transforms a job into a directed acyclic graph (DAG), the more the performance improvement of spark as comparted to map reduce. S pack is especially useful for running iterative data processing job and for interactive analysis. In the cluster, the

Appmaster will ask the node managers on the worker nodes to each launch a specific number of containers. Spark and yarn together will take the responsibility of adjusting the number of executors based on the workload.

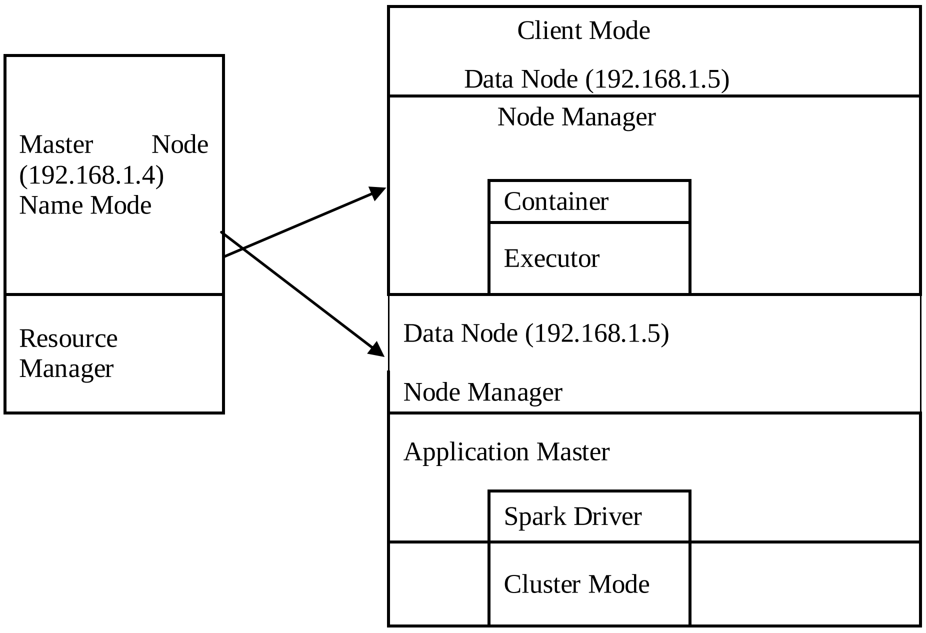

Fig 3. Spark Architecture in 2-node cluster

Spark on YARN :

Yarn [61] runs both runs both map reduce and spark applications in the same cluster Yarn let's all the applications to share the same pool of cluster resources dynamically spark can use data from various sources by connecting with the data sources. Spark works with Hadoop hdfs storage. It`s easy to secure the authentication between the processes enabling Kerberos on Hadoop cluster. It supports both a client and a cluster mode deployment of spark applications. In the client mode the spark driver program runs inside the client process that deploys the spark application and not in the yarn cluster. In the cluster Mode operations. The spark driver program run within the spark specific application master process, which is managed by yarn on the cluster.

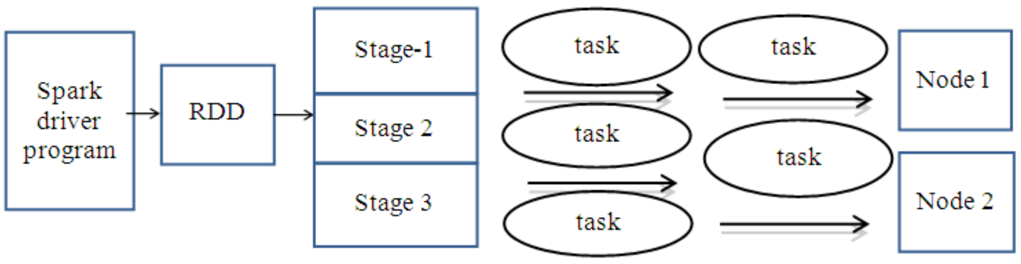

Fig 4. Spark RDD workflow

Spark RDDs:

RDD's is simply a collection of objects that split into partitions, to be computed on different nodes in the cluster. RDD can perform two operations called transformations and action on the RDD. A transformation will create an RDD from an existing RDD. It uses a Process called pipelining while it transforms RDDs. Pipelining means that, whenever possible spark performs sequences of RDD transformations by row without storing any data. Each time we execute an action on an RDD. Spark recomputes the RDDs, Spark stores the RDD contents in memory. It uses a lazy execution model, which means that it will wait to process the data in an RDD until it performs an action.

Components of a Spark:
Spark run in a distributed fashion over a cluster of nodes [19]. The Driver program. A driver program is the application that contains the processing code that spark will execute on each of the worker nodes in a cluster. The driver program can launch more than one job on the cluster.
Worker Nodes:
Worker process run on each of the worker nodes in a cluster and provide the CPU, memory and storage resources necessary to execute spark applications.
Executors: The executer process executes the application code and also caches data in memory or disk storage. Each application has its own executors when the application completes, the executer process goes away.

Dynamic Resource Allocation for spark on Yarn:
When running spark jobs, yarn uses the spark specific application master. It allocates resources such as memory and CPU, when spark application master request resources from the resource manager. It does so by estimating the resource requirement of the job and requesting a specific number of containers to complete the job. Based on the availability of resources in the cluster, the application master will ask the node managers on the worker nodes to each launch a specific number or containers. Spark and yarn together will take the responsibility of adjusting the numbers of executors based on the workload.

Machine learning and Big Data:
The main goal of machine learning [12] is to study, engineer and improve mathematical models which can be trained to infer the future and to make decisions without complete knowledge of all external factors. Machine learning just predicts the output with accuracy of 85 percent. It is broadly classified into three categories.
Supervised Learning:
A supervised learning is the concept of a teacher or supervisor (i.e.) to learn the relationship between other variables and a target variable. The main task is to provide the agent with a precise measure of its error directly compared with the output values. Based on the information, the agent can connect its parameters so as to reduce the loss function. After each iteration, if the algorithm is flexible enough and data elements are coherent the overall accuracy increases and the difference between the predicted and expected value becomes close to zero. The Model avoid a common problem caller over fitting, in this paper we will be mainly focusing on both the classification and regression machine learning problems.

Machine learning models and its working
Throughout, in this paper we will be experimenting few machine learning [4] RDD-based classification and regression methods in spark Regression analysis is a form of predictive modeling technique between a target variable and independent variable, It is an Important tool for modeling and analyzing data.

Linear Regression:
This is used for the prediction of continuous variable. It utilizes error minimization to fit the best possible line. This task can be easily accomplished by least square method. It also has a high bias and a low variance error. Regression models try to fit the best possible hyperplane by minimizing the error between the hyperplane and actual observations. In this paper,we will be using the closed form formula to solve Linear Regression problem by using Normal equations which is an alternative to Gradient Descent (GD) and LBFGS[65].

$$b = (X^1 X)^{-1} (X^1)Y$$

Lasso Regression:
Lasso is usually Prepared to reduce the number of inputs in the early stages. Lasso plays a substantial role in machine learning due to its ability to select a subset of the weights based on the threshold. Spark's RDD-based lasso regression can select a subset of parameters by setting the other weights to Zero. Lasso (least Absolute shrinkage and selection operator) ADI, a regression method that performs both variable selection and regularization at the same time in order to eliminate non-contributing explanatory variables therefore enhancing the prediction accuracy. In lasso regression, a penalty is applier also known as shrinkage penalty on coefficient values to regularize the coefficients with the tuning parameter.

RSS = Residual sum of squares
$RSS(\beta) = {}^{n}\sum_{i=1} (y_{i} - \beta_{o} - {}^{p}\sum_{j=1} \beta_{j,}xij)^2$
Where $\lambda = 0$, the penalty has no impact, lasso produces the same result as linear regression
$\lambda = \infty$, will bring coefficients to zero.
Ridge Regression:

Ridge regression API is meant to deal with multicollinearity. Ridge is about shrinking some of the parameters, therefore reducing their effect and in turn reducing complexity. Ridge regression only shrinks the parameter and does not set then to zero. Ridge regression uses L2 to penalize (ie) shrink some of the parameters.

Ridge Regression = $RSS(\beta) + \lambda\, Ƥ\sum_{j=1} \beta_j^{2.}$

Logistic Regression:

Logistic regression is another technique of machine learning. It applies maximum likelihood estimation after transforming the dependent variable into a logit variable. It is a popular method to predict a categorical response. Logistic regression estimates the probability of a certain event occurring.

log(odds) = log $\left(\frac{p}{1-p}\right) = \beta o + \beta 1 * x1 + \beta 2 * x2 + \cdots . + \beta n * Xn$

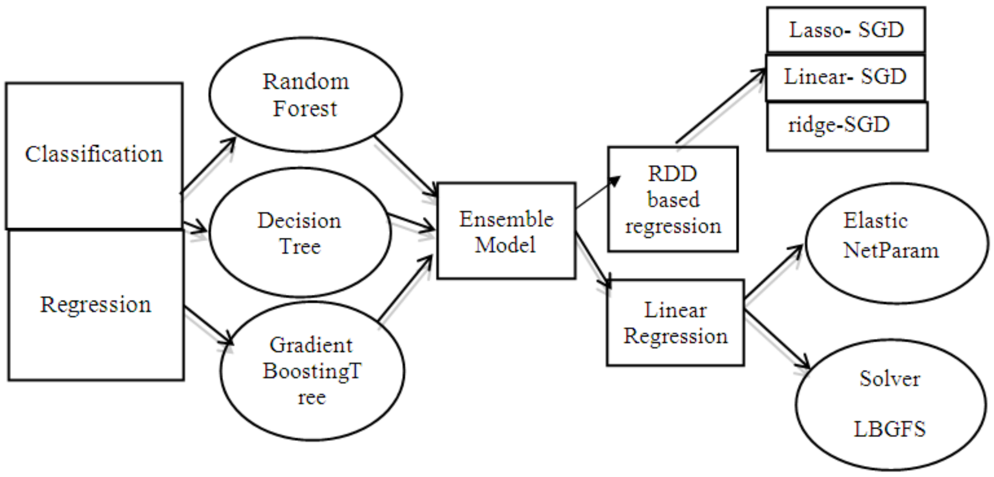

Fig 6. Spark RDD-based Regression and classification

Decision Trees:
Decision Trees are one of the oldest and more widely used methods of machine learning, A smart partitioning algorithm that tries to minimize a loss function. A decision tree in spark is a parallel algorithm to fit and grow a single tree into a dataset that can be categorical or continuous.

Random Forest:
A Random forest [47] is a set of decision trees built on random samples with a different policy, Random forest provides excellent accuracy among current classification and regression algorithms It can handle thousands of input features and variables at a time. It has an effective technique embedded for estimating missing or null values. In spark, RF implementation work very fast due to spark's exploitation of parallelism.

Gradient Boosting Tree:
Gradient boosting is a machine learning technique for regression and classification problems which produces a prediction model in the form of an ensemble of weak prediction models, same as decision trees , it is one of the most powerful techniques, for building predictive models. It is used with decision trees for especially CART trees of a fixed size as base learners.

Parameter Metrics used in Machine Learning Algorithms
Entropy:
Entropy [47] came from information theory and is the measure of impurity In data. If the sample is completely homogeneous, the entropy is Zero and if the sample is equally divided. It has entropy of one. In decision trees, the predictor with most heterogeneous will be considered nearest to the root node to classify the given data into classes in a greedy mode.

$$\text{Entropy} = -p1 * \log_2 p_1 - \ldots\ldots p_n * \log_2 p_n$$

Information Gain:
IG is the expected reduction in entropy caused by partitioning the examples according to a given attribute. The idea is to start with mixed classes and to continue portioning until each node reaches its observations of purest class. It every stage, the variable with maximum information gain is chosen in a greedy fashion.

Information Gain + Entropy of Parent – sum (weighted %* Entropy of child)
Gini Impurity :
Gini impurity[20] is a measure of misclassification, which applies in a multiclass classified context. Gini works similar to entropy, except Gini is quicker to calculate. It can be used where we have complex multi-dimensional data Gini is used for higher-dimensional and noisy data.

Confusion Matrix:
This is the matrix of the actual versus the predicted. The confusion matrix is relatively simple; it is a 2 x 2 matrix.

		Prediction	**Value**
		Yes	No
Actual	Yes	True Positive (TP)	False Negative (FN)
Value	No	False Positive (FP)	True Negative (TN)

(i) Accuracy $= \frac{TP + TN}{TP+FP+TN+FN}$

(ii) Error $= \frac{(FP + FN)}{TOTAL}$

Mean Speared error (MSE):
MSE is mean square error measures the average of the squares of the "errors" that is the difference between the estimator and what is estimated.
Calculating the MSE
MSE $= \frac{1}{n}\sum_{i=1} (y_i\hat{}_{- Y1})^2$

Root mean square error (RMSE):
RMSE is a frequently used measure of the different between values predicted by a model and the values actually observed from the environment that is being modeled. These individual differences are called residuals when the calculations are performed over the data sample that was used for estimation, and are called prediction errors when computed –Out-of-sample.
The RMSE of a model prediction with respect to the estimated variable X model is defined as the square root of the mean squared error.

$$\text{RMSE} = \frac{\sum_{i-1}^{N} (X_{\text{obs,I}} - \text{Xmodel,i})^2}{\eta}$$

Optimization Techniques used for machine learning algorithms
Stochastic gradient descent (SGD):
Stochastic gradient descent is a technique used to minimize the error of a model on a training data, each training instance is shown to the mode one at a time. The model makes a prediction for a training instance, the error is calculated and the model is updated in order to reduce the error for the next prediction. A Stochastic sub gradient is a randomized choice of a vector. Gradient is a class that computes the stochastic gradient of the function being optimized. While includes gradient classes for common loss functions e.g. Hinge, logistic, least-squares.
L-BFGS:
Limited-memory BFGS is an optimization algorithm is MLlib. It is limited memory of BFGS, which is in the family of quasi-Newton methods that approximate the BFQS algorithm, which utilizes a limited amount of computer memory. It is most effective and it is an optimization algorithm in the family of – quasi-Newton-method. The L-BFGS method approximates the objective function locally as a quadratic without valuating the second partial derivatives of the objective function to construct the Hessian matrix. The Hessian matrix is approximated by previous gradient valuations, so there us the vertical scalability issue when computing the Hessian Matrix explicitly in Newton's Method.

Benchmark Tool
Apache Hadoop come with several useful benchmarking tools, by running these tools, we can check the performance of the cluster and compare it with the performance of other clusters.

HiBench Benchmark:
HiBench [29] is a big data benchmark suite that helps to evaluate different lug data frameworks in terms of speed. Throughput and system resource utilizations. It contains a set of Hadoop, Spark and streaming workloads, there are totally 19 workloads in Hibench.The workloads are divided into six categories which are micro machine learning, Sql, graph, web search and streaming benchmark. Running a workload in Hibench involves two phases, a prepare phase and an execution phase. In the prepare phase hibench generates the necessary input data to run the benchmark and an execution Phase where the application logic of the benchmark run over the prepared dataset. Hibench suite is more realistic and comprehensive benchmark suite for Hadoop including not only synthetic micro-benchmarks, but also real world Hadoop applications.

Hibench workload:
In this paper, we use hibench tool [29] to test the performance between Hadoop and spark cluster. Hibench suite is a comprehensive and representative benchmark suite is a comprehensive and representative benchmark suite for Hadoop, spark, storm, Storm-Trident and samza.

Micro Benchmarks:
Sort
This workload sorts it text input data, which is generated using Random Text writer.
Word count
This workload counts the occurrence of each word in the input data, which are generated using Random Text writer.
Terasort
This workload generates input data by Hadoop TeraGen

Machine Learning
Bayesian classification:
This workload is a simple multiclass classification algorithm. It is implemented in spark. Mllib and uses the automatically generated documents whose words follow the zipfian distribution.
K-means clustering:
This workload tests the K-means clustering The input data set is generated by GenKmeans dataset based on uniform distribution and Gaussian distribution.
Logistic Regression:
This workload is a popular method to predict a categorical response. It is implemented with LBFGS optimizer and the input dataset is generated by Logistic Regression Data Generator.
Alternating Least squares:
ALS algorithm is a well-known algorithm for collaborative filtering. This workload is implemented in spark mill and the input data set is generated by rating data generator.
Gradient Boosting Trees:
This workload is implemented in spark. Mllib and the input dataset is generated by gradient boosting tree data generator. It is a popular regression method using ensembles of decision tree.
Linear Regression:
LR is a workload that implemented in spark with SGD optimizer. Input data set is generated by linear Regression Data generator.
LDA:
The input data set is generated by LDA Data generator. Is a topic model which infers topics from a collection of text documents.
Principal components Analysis:
PCA is a statistical method to find a rotation. The input data set is generated by PCA Datagenerator.This is used widely in dimensionality reduction.
Random Forest:
RF are ensembles of decision trees, the input data set is generated by Random forest data generator. They combine many decision trees in order to reduce the risk of overfitting.
Support Vector machine:
SVM is a standard method for large scale classification tasks, the input data set is generated by SVM data generator.

Singular value Decomposition:
SVD factorizes a matrix into three matrices. The input data set is generator.

SQL:
Scan, Join and Aggregation these workloads are developed based on SIQMOD 09 paper. It contains Hive queries performing the typical OLAP queries. Its input is also automatically generated web data with hyperlinks following the zipfian distribution.
Web search Benchmarks:
PageRank algorithm implemented in spark and Hadoop. The data source is generated from web data whose hyperlinks follow the zipfian distribution.
Graph benchmark:
Nweight is an iterative graph-parallel algorithm implemented by spark graphx and pregel. The algorithm computer associations between two vertices that are –n-hop away.

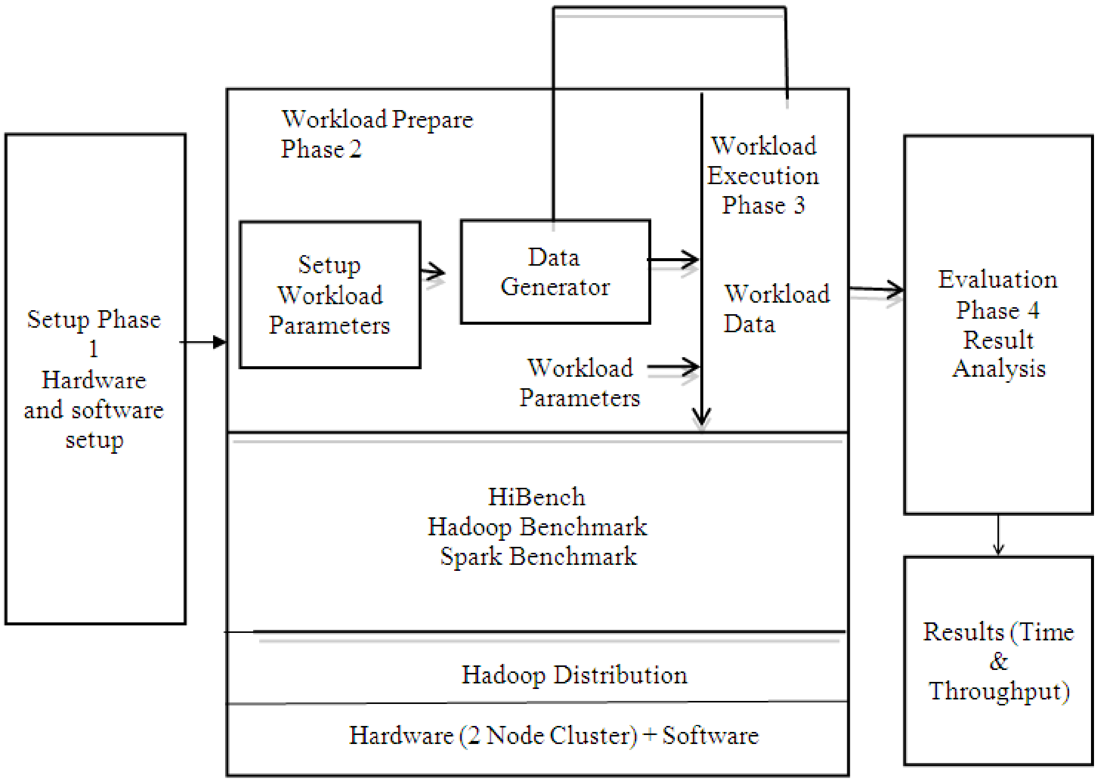
Setup Phase 1 Hardware and software setup
Workload Prepare Phase 2
Setup Workload Parameters
Data Generator
Workload Parameters
Workload Execution Phase 3
Workload Data
Evaluation Phase 4 Result Analysis
Results (Time & Throughput)
HiBench Hadoop Benchmark Spark Benchmark
Hadoop Distribution
Hardware (2 Node Cluster) + Software

Fig 7. HiBench Architecture

MLlib and scala
Scala:

Is a modern Programming language that is flexible and perfect for interacting with spark Mllib. Scala [41] is an excellent match for machine learning programming due to its support for algebra-friendly data types, anonymous functions, variance and higher-order functions, it is a java-based language. It has ability to use Java libraries with Scala and a rich environment for software engineers to build modern and complex machine learning systems.
MLlib:
MLlib [20] is one of the four Apache sparks' libraries. It is a scalable machine learning library. MLlib could be developed using Java spark's APIs using MLlib, we can access HDFS (Hadoop data file system) MLlib algorithms run fast and contains high quality algorithms for classification Regression, clustering, Recommendation etc.

Ganglia Monitoring System:
This is an open-source software and it is widely used by enterprises for metric collection and teaching. Ganglia can monitor very large clusters. It collects metrics such as CPU usage and free disk space and can also help

monitor failed nodes. It provider useful graphical information about the state of the cluster and its nodes. There are four main components in a ganglia monitoring system such as gmond ,gmetad, rrdtool and gweb.

Empherical Analysis:
In this section we further explain about the experimental setup. Hardware and software configuration and the parameters used for the two node cluster namely master and the slave. We begin with the architecture of our cluster and then introduce spark MLlib components datasets and then cod focused on machine learning algorithms which is separated by four experiments in this paper and finally we introduce Hibench benchmark suite to compare the performance between Hadoop and spark framework

III. EXPERIMENT AND RESULT EVALUATION

Experimental setup:
For the experiment, we deploy two separate virtual box vin on two different personal Laptop. Since, the concept of bug data is distributed computing all the frame works in this study implemented over a cluster of servers. A small cluster includes a single master and two or more worker nodes. Our experiment run with a two node duster. Both Hadoop and spark engine are deployed and integrated with yarn distributed environment. Since, we have only two node cluster, the master node also acts as the slave node which is totally fine with small cluster. But with bigger real clusters the server responsibility is even more separated.

Table 1: Hardware and software configuration.

HARDWARE	MASTER NODE	SLAVE NODE
Operating systerm	Ubuntu 16.04	Ubuntu 16.04
Processor	Intel® Core ™ I5-43000 CPU @ I90 GHZ 2.50 GHZ	Intel® Core ™ I7 – 75000 CPU@ 2.70 GHZ 2.90 GHZ
CPU	16.0 GB	16.0 GB
Type	64 bit	64 bit
Cores	8	8
Storage	SATA	SATA
Network	10 GB ethernet	10 GB Ethernet
N/W Adapter	Bridged	Bridged

Software:

Java	Java – 1.8.0 – open jock – and 64	
SCALA	Scala – 2.11.6 tgz	
Spark		Spark – 2.2.0 – bin – hadoop 2.7
Hadoop	2.7.2 version	Mahout
Hibench	7.0 version	Hive – 1.2.2 – bin tar.gz

Configuration Parameters:
Both the Hadoop and spark has numerous configuration parameters. In order to get going with fully distributed cluster we use few parameter settings in our experiment.

Hadoop configuration in Master Node:

Table 2: Mapred

Mapreduce.framework. name	yarn
Mapred.job.tracker	Master : 54311
Mapred. io. Compression. codecs	Snappy code
Mapreduce. map. output. compress	True
Mapreduce.jobhistory. webapp.address	Master:19888

Table 3: HDFS

Dfs. Datanode. data. dir	Directory path
Dfs. Replication	2
Dfs. Block. size	134217728
Dfs. Namenode. Acls. enabled	True
Dfs. Permission. Enabled	True
Dfs. Permission. superuser. Group	Username
Dfs. Datanode.address	Master: 50010

Dfs. Datanode. ipc. Address	Master: 50020
Dfs. Datanode. http. address	Master: 50075
Dfs. Secondary. http. address	Master: 50090

Table 4: Yarn

Yarn. Nodemanager.aux - services	Mapreduce – shuffle
Yarn. Nodemanager.aux – services. Maproduce. Shuffle. class	org.apache.hadoop. mapred. Shuffle handler
Yarn. Nodemanager. Resource. Memory. mb	8192
Yarn. Schedules. Minimum – allocation – mb	1024
Yarn. Nodemanager. Vmem – pmem - ratio	3.1
Yarn. Resourcemanager. Hostname	Master
Yarn. Resourcemanager.schedules. address	8035
Yarn. Resourcemanager.resource.tracker.address	8025
Yarn. Timeline – service. hostname	Master
Yarn. Timeline – service. address	10200
Yarn. Nodemanager. Recovery. enabled	False
Yarn. Resourcemanager.address	8050
Yarn. Timeline –service.enabled	false

Table 5: Core

Fs. Default FS	Master: 54310
Hadoop. tmp.dir	Tmp directory

Table 6: Slave Node

Dfs. Datanode. address	Slave: 50010
Dfs. Datanode. Ipc. address	Slave: 50020
Dfs. Datanode. http. address	Slave: 50075
Dfs. Namemode. Name. dir	Namenode dir
Dfs. Datanode. Data. dir	Datanode dir

Spark configuration parameters**:**

Spark. master	Yarn
Spark. serializer	Kryoserializer
Spark. Driver. memory	4 G
Spark. Yarn. Driver. memory	1 G
Spark. Executor. cores	2
Spark. Executor. memory	2 G
Spark. Dynamic., allocation. enabled	true
Spark.yarn.am.memory	777m
Spark.yarn.am.cores	2
Spark. Yarn. Submit. File. replication	2
Spark. Yarn. Executor. Memory overhead	384 m

In our experiment, we set the same set of parameters for both master and slave node in our fully distributed cluster.

Result evaluation and analysis
Spark machine learning:
To evaluate hadoop and spark experiments we first set up a passwordless connection SSH on the cluster.
SSh – keygen – t dsa
And then we setup two node Hadoop cluster for our experiments namely name node and data node service (Our cluster is off with a single node, where the master service running one one machine and the data node service running on the other. We connect the two node using bridged adapter network.

Experiments and datasets
In this paper, we do three experiments for the first experiment we use a housing dataset from the UCI machine library repository. The dataset comprises 14 columns with the first 13 columns being independent variables. We explore linear regression in spark MLlib with lasso and ridge regression using Lbfgs and auto optimization techniques. We complete and evaluate the results using the error metrics MSE (mean squared error) and RMSE (root mean squared error). We use the algorithms to train and predict the median price of the house. We load the housing datasets from hdfs for our experiments. The result shows that linear regression with Lbfgs outperform the other. Lbfgs is a hessian free quasi – Newton method. Fig. shows the regression type and parameter metrics used for experiments.

Table 7: Regression

Type	Parameter metrics
Lasso and auto	MaxIter (1000) Elastic Net Param (0.0) Reg Param (0.01) Solver (auto)
Lasso and Lbfgs	MaxIter (10) Solver (L-lfgs)
Ridge and auto	MaxIter (1000) Elastic Net Param (1.0) Reg Param (0.01) Solver (auto)

For the second experiment, we use Wisconsin Breast cancer dataset which is publicly available at UCI Machine Learning repository [13] we use the dataset to predict whether a patient in malignant or not. The dataset has 699 instances, with 458 classified as benign tumors and 241 as malignant cases. We split the whole dataset into training (70%) and test data (30%) randomly. The dataset is used to examine and measure the effectiveness of the tree model in spark namely Random forest, Decision Tree and gradient boosting tree using various impurity and confusion matrix measurements. For regression based algorithms we use variance as impurity and as a result, we evaluate MSE (mean squared error) as a key parameter for the measurement of a model in spark MLlib.
Variance method = 1/N $\sum_{i=1}^{n}$(yi μ)2
Whereas for classification based algorithms we use 'Gini and Entropy' as impurity and as a result we evaluate and compute both the accuracy and error.

$$\text{Accuracy} = \frac{\text{True positive (TP) + True Negative (TN)}}{\text{Total}}$$

$$\text{Error} = \frac{\text{False positive (FP) + False Negative (FN)}}{\text{Total}}$$

From the results, we depict that gradient boosted tree classifier outperform the other two algorithms namely RF and DT.

Table 8: Parameter Metrics

Classification	Num classes (2 Max depth (5) Max bins (32) Impurity (entropy), (Gini) Feature subset stratergy = 'auto'
Regression	Numclasses (2) Numtrees (3) Impurity (Variance) Maxdepth (4) maxBini (32) feature subset stratergy = 'auto'

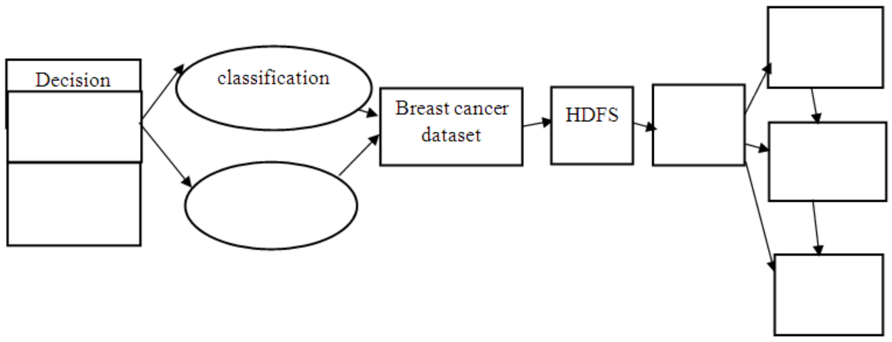

Fig 8. Workflow of Experiment

Table 9: Classification and Regression Evaluation

Model	Dataset	Classification (Impurity)	Regression (confusion Matrix)		Accuracy	Error	MSE
Decision Tree	Breast Cancer	Gini	115.0 0	5.0 88.0	0.9	0.03	
		Entropy	116.0 9.0	4.0 82.0	0.9	0.06	
		Variance					0.03
Random Forest	Breast Cancer	Gini	118.0 4.0	1.0 59.0	0.9	0.02	
		Entropy	115.0 0.0	4.0 63.0	0.97	0. 0.021	
		Variance					0.028
Gradient	Breast Cancer	Classification	124.0 2.0	2.0 64.0	0.97	0.020	
		Regression					0.053

In the third experiment, we use SGD (schostic gradient descent) optimizer technique for comparing linear, Ridge and Lasso regression. We use spark RDD – based regression API and show have to use an iterative optimization technique SGD to minimize the cost function to solve linear regression problem. The housing dataset is used for the experiment to predict the housing prices. SGD expressed in a formula,

$$W := w + n\left(y\,(i) - y\,\big(z(i)\big)\right)x(i)$$

SGD is used to compute intercept and weight for the model and we evaluate RMSE to quantify the fit for the model.

$$\text{RMSE} \;=\; \frac{\sum_{i-1}^{N} \,(y\,t - y))^2}{\eta}$$

And also both Ridge and lasso regression use the same housing datasets for our experiments. We use to fit and train the model using predict () API. To train both the ridge and lasso with SGD we use spark RDD – based regression method. Ridge regression reduces the parameter's weight but does not eliminate or reduce the weight to zero. Lasso performs both variable selection and regularization at the same time in order to eliminate variables.

Table 10:

Type	Parameter metrics
Linear	Num Iter (1000) Step SGD (.001)
Ridge and lasso	numIter (1000) step SGD (.001) regularization Param (1.13)

Finally, the fourth experiment we explored is our alternative to gradient descent and L-BFUS optimization method to solve linear regression using normal equations. We use the same dataset to solve LR using the closed form formula

$$b = (x^1\,x)^{-1}\,(x^1)y$$

Table 11:

Type	Parameter metrics
Linear regression	Maxiter (1000) Elastic Net Param (0.0) Reg param (0.01) Solver ("normal")

Table 12: Results of SGD

Type	Dataset	MSE(Mean Squared error)	RMSE (Root mean squared error)
Lasso - SGD	housing	99.8	9.99
Ridge - SGD	housing	92.6	9.6
Linear - SGD	housing	91.4	9.5

Table 13: Evaluate Linear Regression

Type	Dataset	MSE	RMSE
Linear regression with		13.608	3.689
Lasso and L-BFGS	housing	13.608	3.689
Lasso and auto	housing	13.609	3.689
Ridge and auto	housing	13.611	3.689
Normal equation	housing	13.609	3.689

Experimental evaluation and results HiBench

In this paper, we additionally study, examine and evaluate HiBench benchmark suite to compare and analyse the performances between Hadoop and spark. HiBench[30] is a big data benchmark suite that helps to evaluate different big data frame works in terms of speed and throughput and (bytes / node). There are totally 19 workloads in HiBench. The workloads are divided into six categories, which are micro, ml, sql, graph and web search benchmarks.

Micro Benchmark:

There are three micro benchmarks namely sort, word count and terasort used in HiBench suite. For both Hadoop and spark framework, the input data of sort and word count are generated using the Random writer and Random Text writer program. The input data for terasort is generated by the Teragen program.

Web Search Benchmark:

The nutch Indexing workload is an open source apache search engine. This workload used the crawler subsystem in hutch to crawl an in – house Wikipedia mirror. The PageRank workload calculates the rank of webpages.

Machine Learning Benchmark:

To evaluate machine learning algorithm in hadoop, we use mahout an open-source machine learning library built on top of hadoop and for spark MLlib is used for experiments. For all the machine learning algorithms used in hibench suite we use a random data generator to generate the input for the workload. Our goal is to show the performance between hadoop and spark based on the execution time and throughput per node bytes. Spark frame work is faster than hadoop. But in some cases hadoop outperform spark engine.

Results of HiBench

Table 14: Hadoop and spark performances

BENCHMARK	DATA SIZE	DURATION	THROUGHPUT (BYTES)	THROUGHPUT (NODE)	DURATION	THROUGHPUT (BYTES)	THROUGHPUT (NODE)
Micro Benchmarks sort	328501033	116.609	2817115	939038	194.091	1692510	564170
Terasort	3200000000	330.815	9673080	4836540	800.004	3999980	1999990
Wordcount	328495592	49.835	6591664	3295832	92.726	3542648	1771324
Sleep	0	68.826	0	0	74.339	0	0
Machine Learning Bayesian	111385907	968.197	115044	57522	108.021	1031150	515575
k-means	602462583	347.899	1731716	865858	103.738	5807539	2903769
Linear Regression	4003010600	--	--	--	192.583	20785898	10392949
Random Forest	11316	--	--	--	43.416	260	130
Support Vector Machine	8062008	--	--	--	111.831	72090	36045
Singular Value Decomposition	805000	--	--	--	90.879	8857	4428
Logistic Regression	808432	--	--	--	95.941	8426	4213
Gradient Boosting Tree	11316	--	--	--	159.005	71	35
PCA	88432				357.893	247	123
LDA	21876012				234.896	93130	46565
ALS	65728				136.808	480	240
SQL	0						
Join	0	116.512	0		115.252	0	0
Scan	205615	79.331	2591		65.036	3171	3171

Aggregation	37988	72.629	522		111.937	339	169
Websearch	0						
Benchmark	0						
Pagerank	10823	117.721	91		41.437	261	261
Nutch indexing	254658711	317.420	802276				
Graph							
Benchmark							
Weight	4353413				94.706	45967	22983

IV. CONCLUSION

The increasing adoption of big data analytics has led to a high demand for efficient technologies in order to manage and process large datasets. We have implemented RDD's in a system called spark and provided coarse – grained transformations rather than fine – grained. We worked on four experiments. The first one explored linear regression with Lasso, ridge and lbfgs using housing dataset to predict the housing price. The second experiment use cancer dataset for evaluating both classification and regression methods in Random forest, gradient boosted tree and decision trees. In the third experiment, we predict the housing prices to solve ridge and Lasso regression by using SGD optimizer technique. Finally, we experiment linear regression with normal equation against GD and L-bfgs techniques. And additionally, in this paper we use HiBench suite to compare Hadoop and spark frameworks.

Future Enhancement

Our future work will be focused on Apache spark MLlib and ML package. And also work on unsupervised machine learning algorithms in future with different data sets and different language using python.

REFERENCES

[1]. Apache Hadoop. https://hadoop.apache.org/

[2]. White T. Hadoop. The Definitive Guide, 3rd edn. Sebastopol, CA: O' Reilly Media, Inc;2012.

[3]. M. Zaharia, M. Chowdhury, T.Das, A. Dave, J. MA, M.Mc Cauley, M.J.Franklin, S. Shenker I. Stocia, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, in proceedings of the 9th USENIX conference on Networked systems design and Implementation. USENIX Association, 2012, pp.2-2.

[4]. "Apache hadoop releases," http://hadoop.apache.org/releases.html.

[5]. "Apache spark Mllib, "http://spark.apache.org/mllib/.

[6]. M. Zaharia, R.s. Xui, P. Wendell, T.Das, M.Armbrunt A. Dave, X.Meng, J.Rosen, s. Venkataraman, M.J, Franklin et.al., "Apache spark: a unified engine for bug data processing," communications of the ACM, Vol.59. no.11, pp. 56-65, 2016.

[7]. M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." Hotcloud, vol.10. no. 10-10, p/.95, 2010.

[8]. M. Frampton, Mastering Apache Spark. Packt Publishing Ltd., 2015.

[9]. X. Heng, J. Bradley, B. Yuvaz, E. Sparks, S. Venkataraman, DlLill, J.Freeman, D,Tsai, M. Ande, S. Owen et al., "Mllib: Machine learning in apache spark," JMLR, vol, 17, no. 34, pp.1-7, 2016.

[10]. C-Y. Lin, C.-H. Tsai, C-P.Lee, vand C.-J Lin, "Large scale logistic regression and linear support vector machines using spark," in Big Data (Big Data), 2014 IEEE International conference on. IEE, 2014, pp.519-528.

[11]. "Github-apache spark," https://githuf.arm/apache/spark/.

[12]. Landset, Sara, Taghi M.Khoshgoftaar, Aaron N. Richter, and Taufiq Hasanin. "And Survey of open source tools for machine learning with big data in the hadoop ecosystem. Journal of Big Data 2, no.1 (2015): 1.

[13]. "Uci machine learning repository." http://archive.ics' Uci edu/ml/index.html.

[14]. J.A. Hanley and B.J.McNeil, "The meaning and use of the area under a receiver operating Characteristic (roc) curve, "Radiology, vol.143, no.1. pp.29-36. 1982.

[15]. Apache Hadoop 2.7.2 Documentation. http://hadoop.apache.org/docs/current/.

[16]. Apache Hwi, http://hwi.apache.org/.

[17]. Dean J, Ghemawat S. Mapreduce: simplified Data processing on Large clusters. In: proceedings of the 6th symposium on Operating systems design and Implementation, 2004.

[18]. Zaharia M, Chowdhury M, Das T, Dave A. Fast and interactive analytics over hadoop data with spark. USENIX Login 2012; 37(4): 45-51.

[19]. Spark. https://spark.apache.org./.

[20]. MLlib.httpsL//spark.apache.org/mllile/.

[21]. Graphx. https://spark.apache.org/graphx/.

[22]. Mahout. http://mahout.apache.org/.

[23]. Albert M.Big Data and Machine Learning: A cake study with Bump Boost. Theses, Free University of Berlin; 2014.

[24]. Lin CY, Tsai CH, Lee CP, Lin CJ, Large-scale logistic regression and linear support vector machines using spark, In: 2014 IEEE International conference on Big Data, 2014. Pp 519-28.

[25]. Kraska T, Talwalkar A, Duchi J, Griffitch R, Franklin MJ, Jordan M, ML base: A distributed Machine- learning system. In: 6th Biennial conference on Innovative Data systems research, 2013.

[26]. Mehdi Assefi, Ehsun Behravesh, Big Data Machine Learning using Apache spark MLlib, In: 2017 IEEE Big Data.

[27]. Wael Etauir, Marian Biltawi, Evaluation of classification algorithms for banking customers behavior under Apache Spark Data Processing system, In: 4th International symposium on Emerging Information, communication and Networks, 2017. Pp 559-564.

[28]. Hameeza Ahmed, Muhammad Ali Ismail, Performance comparison of spark Clusters configured conventionally and a cloud service, In:symposium on Data Mining Applications, 2016. Pp.99-106.

[29]. "HiBench Benchmark suit." [Online]. Avoidable: https://githule.com/intel-hadoop/HiBench/.

[30]. Huang, Shengsheng, et al., “Hibench: A representative and comprehensive hadoop benchmark suit. “Proc. ICDE Workshops, 2010.
[31]. M.F.Hyder, M.A. Ismail and H.Ahmed. “Performance comparison of Hadoop clusters configured on virtual machines and as a cloud service,” in proceedings of the 10th IEEE International conference on Emerging Technologies (ICET), Islamabad, Pakistan, 8-10 December, 2014, pp.60-64.
[32]. Shengsheng Huang, Jie Huang, The HiBench Benchmark Suit; Characterization of the MapReduce-Based Data Analysis, April 2010.
[33]. Hadoop homepage. http://hadoop.apache.org/.
[34]. Mahout homepage. http://lucene.apache.org/mahout/
[35]. a. Pawie, A. Rasein, S. Madden, M. Stonebraker, D. Dewin and DJ. Abadi. “A comparison of Approaches to Large-scale Data Analysis, SIGMOD, June, 2009.
[36]. Matei Zaharia, Andy Kowiski, Anthony D.Joseph, Randy Katz, and Ion Storcai. “Improving MapReduce Performance in Hetercogenous Environments,” ISDI’ 08, December, 2008.
[37]. Worcwent program. Available in Hadoop source distribution: src/examples/org/apache/hadoop/examples/wordcount.
[38]. Mo Hai, You Zhang and Yuejin Zhang, A performance Evaluation of classification Algorithms for Big Data, In: Information Technology and Quantitative Management, 2017. Pp. 1100-1107.
[39]. Luo Yuan-Shuai Research on Parallel Text Classification Algorithm based on Random forest and spark Master Degree Thesis, southwest Jiaotong University, 2016.
[40]. Matei Zaharia, Mosharaf Chowdhury and Justin Ma, Resilient Distributed Datasets: A Fault – Tolerant Abstraction for In-Memory Cluster Computing, UC Berkeley, 2011.
[41]. Scala. http://www.scala-long.org.
[42]. Jakayla Alston and Babak Yadranjiaghdam, Performance Analysis of sparks Machine Learning Library, In: Transactions on Machine Learning and Datamining, 2017. Pp.67-77.
[43]. Apache Software foundation (2015). Apache spark-Lightning-fast cluster computing. URL http://spark.apache.org/.
[44]. Zhou L., Pan S., Wang J., Vasilakos A.V.: Machine Learning on Big Data. Opportunities and Challenges. Neurocomputing 237, 2017, pp. 350-361.
[45]. White T, Hadoop: The Definitive Guide. Sebastopol: O ‘Reilly Media, Inc; 2012.
[46]. Mengx, Bradley J, Yavuz B, SparksE, Venkataraman S, Liu D, Freeman J, Tsai D, Ande M, Owens, Xui D, Xni R, A. Mllib Machine learning in apache spark. JMach Learn Res. 2016; 17(34) pp.1-7.
[47]. MLlile Machine Learning Library (MLlib) for spark. http://spark.apache.org./docs/latest/mllile-guide.html.
[48]. M. Zaharia, A. Konurinski and I. Stocia. “Impressing mapreduce performance in heterogenerous environments.” In Osdi, vol. 8, no.4, 2008, p.7.
[49]. Adam Barber and Blesson Varghese, “Plug and play Bench: Simplifying Big Data Benchmarking using containers, November 2017.
[50]. Rui Han, Zhen Jai and Lei Wang, “Benchmarking Big Data Systems: State-of-the-Art and future directions, June 2015.
[51]. . Pavlo, E. Paulson and A.Rasin, “A comparison of approaches of the 2009 ACM SIGMOD International conference on Management of data. ACM, 2009, pp.165-178.
[52]. R. Han, X.Lee, and J.Xu, “On big data benchmarking,” in Big Data Benchmarks, Performance optimization, and emerging Hardware. Springer, 2014, pp.3-18.
[53]. Neelam Naik and Seema Purwhit, “Comparative study of Binary classification Methods to Analyze a Massive Dataset on virtual Machine, In: International conference on Knowledge based and Intelligent Information and Engineering Systems, KES2017, Sep. 2017, pp.6-8.
[54]. Zaharia M., Xui R.S., Wendell P., Das T., Stotica I. Apache Spark: A Unified Engine for Big Data Processing. Communications of the ACM, November 2016.
[55]. Zaharia M., Wendell P., Karau H. Learning Spark. O’Keilly Media, Inc; 2015.
[56]. N. Spangenberg, M. Roth, and B. Franczyk, “Evaluating new approaches of Big Data analytics frameworks,” in Proc. Of the 18th International conference on Business Information Systems (BIS, 15), Poznan, Poland, 2015, pp.28-37.
[57]. J. Shi et al., “Clash of the titans; MapReduce vs. spark for large scale data analytics,” in Proc. Of the very Large data base (VLDB) Endovement, vol.8, no.13, 2015, pp.2110-2121.
[58]. Apace Mahout: Scalaule machine learning and data mining, http://mahout.apache.org/.
[59]. Neelam Naik and Seema Purohit, “Comparative study of Binary Classification Methods to Analyze a Masive Dataset on Virtual Machine, In:KES 2017, September, pp.6-8.
[60]. San R. alapati, “Expert Hadoop Administration,” Pearson “Education Inc. 2017.
[61]. Siamak Amirghodsi, Shuen Hei and Broderick hall, “Apache spark 2.x Machine Learning Cookbook”, September 2017.
[62]. Dween S., Anil R., Dunning T., Friedman E. “Mahout in Action”, 2012, Manning Publication.
[63]. Matei Zaharia, Ion Storica, “Spark: Cluster computing with working sets”. Hotcloud, vol. 10. No.10-10, p.95, 2010.
[64]. Maryam Kontagora and Horacio, “Benchmarking a MapReduce environment on a full virtualization Platform”, In: IEEE, 2010.
[65]. Pratap Dangeti, “Statistics for Machine Learning”, July 2017.

A.VETTRISELVI, et. al. " A Comparative study of Machine Learning Algorithms Using RDD Based Regression and Classification Methods." *International Journal of Computational Engineering Research (IJCER*), vol. 11, no.4, 2021, pp 07-23.