International Journal of Scientific Research in Computer Science and Engineering
Vol.5, Issue.3, pp.92-97, June (2017)

Research Paper

E-ISSN: 2320-7639

A Big Data Analytical Approach for Analyzing Temperature Dataset using Machine Learning Techniques

J.V.N. Lakshmi[1*], Ananthi Sheshasaayee[2]

[1*]Dept. of IT and MCA, Acharya Institutes of Management and Sciences, Peenya, India
[1*]Research Scholar, Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya University, Kancheepuram, India
[2] PG & Research Department of Computer Science, Quaid-E-Millath Govt College for Women, Chennai, India

[*]*Corresponding Author: jlakshmi.research@gmail.com, Tel.: 91-9498004429*

Available online at: www.isroset.org

Received 12th May 2017, Revised 24th May 2017, Accepted 15th Jun 2017, Online 30th Jun 2017

Abstract— Machine Learning algorithms are used to predictive analytics. These algorithms are put into practice for measuring the temperature data. To capture these data spark framework is being exploited. Machine learning applications are increasingly deployed not only to serve predictions using static models, but also as tightly-integrated components of feedback loops involving dynamic, real-time decision making. These applications pose a new set of requirements, none of which are difficult to achieve in isolation, but the combination of which creates a challenge for existing distributed execution frameworks: computation with millisecond latency at high throughput, adaptive construction of arbitrary task graphs, and execution of heterogeneous kernels over diverse sets of resources. We assert that a new distributed execution framework is needed for such ML applications. This paper describes evaluation of these algorithms on Hadoop, an open-source for spark implementation. The proposed methodology uses a temperature data set for analyzing the machine learning algorithms on spark data frame.

Keywords— Big Data; Machine Learning; HADOOP; Spark; Linear Regression; Gradient Boosting Tree

I. INTRODUCTION

The simplification and generalization capability of a machine learning algorithm depends upon the features of the dataset. Hence engineering the features to represent the salient structure of the data is important [3]. However, feature engineering requires domain knowledge and human ingenuity to generate appropriate features for the dataset.

Conventional Machine Learning methods were limited in building pattern recognition, vigilant engineering and designing the domain expert for extracting features from raw data which ensemble internal representation for understanding classification and prediction from raw input [7]. Perhaps the existing techniques afford little for applying statistical machine learning algorithms in handling Big Data. Systems built on storage requirements such as standard databases and Hadoop are not proposed for access patterns of machine learning methods but the developers are forced to construct adhoc solutions to mine and evaluate data with tools [1].

Iterative algorithms from machine learning techniques are applied as a function repeatedly to the same dataset to optimize the parameters. While each iteration can be expressed as a job on a spark framework for significant performance analysis by adapting the features such as scalability and fault tolerance of MapReduce.

Deploying analytics is forthcoming challenge as end users gather information. An open source framework for processing contaminated analytics on Big Data is Spark [8]. This unified framework gives us a wide-range of practices on diverse text data, graph data and structured either static or real time streaming as well. Spark uses MLlib for developing Machine Learning algorithms. These algorithms uses less memory, less processing time and are largely hand tuned on specialized architecture to parallelize large cluster of machines for data analytics [2].

In this article an algorithmic model has been developed to tune the databases using machine learning within the following sections. Section 2 briefs Spark framework and Hadoop. Section 3 presents the machine learning methods using spark framework. In section 4 model is build on PySpark framework for tuning machine learning techniques.

The computation of Random mean squared error using anaconda tool is discussed in section 5. The results and evaluation of the study conducted on temperature dataset is discussed in section 6. The paper is concluded in Section 7.

II. SPARK AND HADOOP MAP REDUCE

Spark is implemented on top of HDFS infrastructure to provide augmented and supplementary functionality. Spark uses improvised map reduce framework with facilitating in memory data storage, enhanced shuffle phase, rapid performance and real time processing. This operates on huge datasets than the average capacity of a cluster [6].

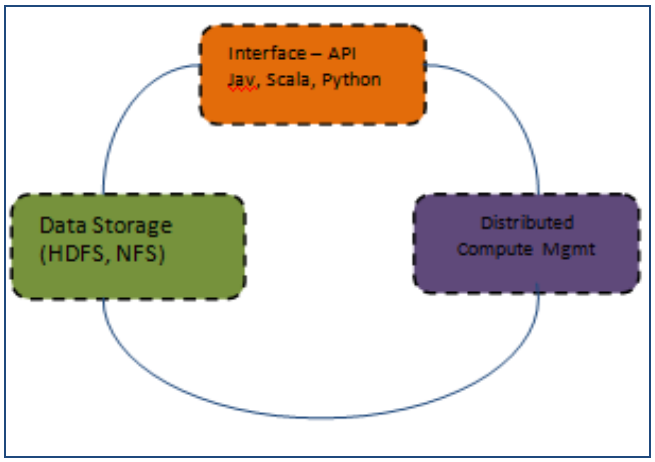

Figure 1. Spark Architecture

Spark Architecture: Model has three components Data Storage, API and Resource Management.

Data Storage: In spark is structured using HDFS attuned on data sources such as Hbase, Cassandra etc,

API: Spark afford wide range of Application interfaces such as Scala, java and Python for development

Resource Management: Spark can be deployed in two ways one as a standalone server and other as a distributed computing network like yarn.

III. MACHINE LEARNING ALGORITHMS

A set of machine learning algorithms compute certain statistics for expressing individual data points as a summation function. Machine learning applications are classified under single pass learning, iterative learning and query based learning [4]. These single pass learning applications erect a single pass through the dataset and extract relevant analysis for future usage during inference. This uncomplicated learning has several stages take place in natural language processing, from machine translation, information extraction and spam filtering. These simple learning techniques are composed of these statistics into estimating language model multinomials, feature extraction for classifications and syntactical translation modeling. The classes of iterative ML algorithms are chained together into multiple spark jobs which share some common properties are coordinated in the dataset via iterative expansion. While in the case of multi query instances a query is concurrently processed on various spark jobs.

A machine learning algorithm is elegantly expressed with a novel framework such as Spark for substantial improvement in performance on analytics. Iterative methods include statistical techniques such as Linear Regression, Random Forest, Decision Tree, Gradient Booster and Naïve Bayes are evaluated for optimization procedures. Numerous algorithms are expressed using multiple instances in a distributed environment as a summation functions for spark jobs.

A significant task of machine learning is classification and regression through which an attempt is deployed for recognizing patterns and predictions [5]. Algorithms constructed distinguish based on different examples relating distinguished patterns [9]. Decision Tree, Random Forest, Logistic Regression, Naïve Bayes and K-Nearest Neighbor are some of the classification methods. Linear Regression, Perceptron, Stochastic Gradient Descent and Gradient booster are some of the regression algorithms. A machine learning task aims to identify a function $f : x \rightarrow y$ that maps input domain x onto output domain y. the function f is selected from a certain function class which is varying for each learning algorithm. Elements of X and Y are application specific representations of data objects and predictions respectively [10]. Some of the machines learning applications were incorporated in spark for big data analytics in this article.

A. *Linear Regression*

Linear Regression assumes a linear or straight line relationship between the training and target variables. In simple linear regression uses the statistics on the training data to estimate the coefficients using least squares to make predictions on new data.

```
may_assembler = VectorAssembler(inputCols=["MAY"],outputCol="features")

may_assemblmn_df= may_assembler.transform(temparature_data)

(trainingData, testData) = may_assemblmn_df.randomSplit([0.8, 0.2],seed = 11)

may_lr = LinearRegression(labelCol="ANNUAL", featuresCol="features")

may_model= may_lr.fit(trainingData)

may_predictions = may_model.transform(testData)

may_predictions.select("prediction", "ANNUAL", "features").show(20)
```

Snippet 1: Linear Regression fit in pyspark

prediction	ANNUAL	features
23.81063681050591	23.86	[27.83]
24.251960882230392	23.77	[28.36]
24.41017139850898	23.96	[28.55]
24.20199966656347	24.11	[28.3]
25.45103005823653	24.05	[29.8]
23.477562039393096	24.15	[27.43]
24.301922097897315	24.61	[28.42]

Table 1: Prediction and Annual

The table 1 gives the prediction values for given annual temperature using linear regression method.

```
evaluator = RegressionEvaluator(labelCol="ANNUAL", predictionCol="prediction", metricName="rmse")

rmse = evaluator.evaluate(may_predictions)

print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
Root Mean Squared Error (RMSE) on test data=0.45328
```

The coordinates of the linear equation are given as

```
x1 = list(df1['YEAR'])

y1 = list(df1['ANNUAL'])

plt.plot(x1, y1)
```

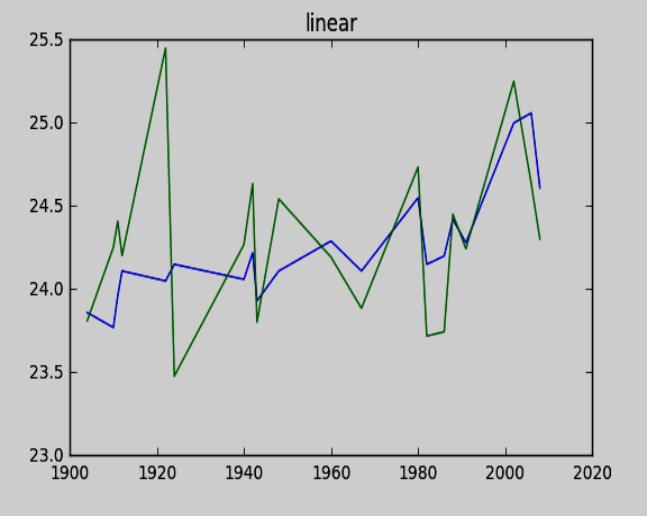

Figure 2: Linear Regression line projecting Annual and the predictions of the temperature dataset

To fit ɵ on our dataset to predict the temperature as a function of x with the training data set. The above results in figure 2 were obtained with the linear regression fit on the temperature dataset.

B. Gradient Boosting Tree

Gradient boosting is a greedy algorithm which overfits training dataset quickly. It constructs additive regression model by applying least squares method at each iteration on parameterized function to present pseudo residuals. The pseudo-residuals are the gradient of the loss functional being minimized, with respect to the model values at each training data point, evaluated at the current step. It is shown that both the approximation accuracy and execution speed of gradient boosting can be substantially improved by incorporating randomization into the present procedure.

Specifically, this algorithm at each iteration draws a subsample of the training data randomly (without replacement) from the full training data set. This randomly selected subsample is then used in place of the full sample to fit the base learner and compute the model update for the current iteration. Gradient Boosting Tree approach increases robustness against congestion of training sets. The following snippet shows the implementation in pyspark framework.

```
temparature_data.select(my_features).show()

my_features_assembler = VectorAssembler(inputCols=my_features,outputCol="features")

my_features_df= my_features_assembler.transform(temparature_data)

my_features_dt = GBTRegressor(labelCol="ANNUAL", featuresCol="features")

(my_features_trainingData, my_features_testData) = my_features_df.randomSplit([0.8, 0.2],seed = 11)

my_features_model = my_features_dt.fit(my_features_trainingData)

my_features_predictions = my_features_model.transform(my_features_testData)
```

Snippet 2: Gradient Boosting tree fit in pyspark

```
my_features_predictions.select("prediction", "ANNUAL", "features").show(5)
```

prediction	ANNUAL	features
19.210420539917976	19.22	[13.04,14.07,17.7...
19.21784604799296	19.01	[13.04,14.31,17.3...
19.339460590119 69	19.31	[13.81,13.8,17.09...
19.51264226929818	19.27	[13.5,15.62,17.19...
19.247095496060574	19.32	[13.65,14.95,17.9...

Table 2: Prediction and Annual

The table 2 gives the prediction values for given annual temperature using gradient regression tree method.

```
my_features_testData.select(mean('ANNUAL').al
       ias('ANNUAL_value')).collect()

evaluator = RegressionEvaluator(labelCol="ANN
UAL", predictionCol="prediction", metricName=
                     "rmse")

rmse = evaluator.evaluate(my_features_predict
                     ions) #
```

```
print("Root Mean Squared Error (RMSE) on test
data = %g" % rmse)

Root Mean Squared Error (RMSE) on test data =
0.197092

'GBTRegressionModel (uid=GBTRegressor_4f36814
e93c4fc654d3a) with 20 trees
```

```
Tree 0 (weight 1.0):
If (feature 6 <= 23.1)
      If (feature 1 <= 12.99)
            Predict: 0.003
      Else (feature 1 > 12.99)
            If (feature 0 <= 12.16)
                  Predict: 18.76
            Else (feature 0 > 12.16)
                  Predict: 18.62
Else (feature 6 > 23.1)
     If (feature 14 <= 23.47)
           If (feature 13 <= 20.29)
                 If (feature 14 <= 22.94)
                       If (feature 10 <= 15.87)
                             Predict: 18.799999999999997
                       Else (feature 10 > 15.87)
                             Predict: 18.9375
                 Else (feature 14 > 22.94)
                             If (feature 15 <= 16.39)
                                   Predict: 19.074166666666667
                             Else (feature 15 > 16.39)
                                   Predict: 19.277142857142852
                            Else (feature 13 > 20.29)
```

Output 1: Gradient Boosting tree structure

This above tree represents output of displaying 6 trees of Gradient Boosted Algorithm for the temperature data set.

```
                                  If (feature 12 <= 14.32)
                                       If (feature 15 <= 16.8)
                                             Predict: 19.28723513513513
                                        Else (feature 15 > 16.8)
                                             Predict: 19.478333333333335
                                  Else (feature 12 > 14.32)
                                        If (feature 3 <= 21.74)
                                             Predict: 19.543333333333333
                                        Else (feature 3 > 21.74)
                                             Predict: 19.71
        Else (feature 14 > 23.47)
              If (feature 15 <= 17.2)
                    If (feature 4 <= 22.98)
                          If (feature 0 <= 13.08)
                               Predict: 19.42
                          Else (feature 0 > 13.08)
                               Predict: 19.370000000000005
 Else (feature 4 > 22.98)
    Predict:   19.34000004
```

Output 2: Prediction from Gradient Boosting Tree

IV. MODEL AND METHODOLOGY

Machine Learning algorithms are modeled in the figure 3. Linear Regression and Gradient Boosted methods were applied on the temperature dataset. The data set is given as an input in the initial step. The job is distributed on Hadoop Spark frame work. The snippets which are used are written in python language in a spark environment. In the transform stage the model regulates to process three steps vector assembler => transform => regression evaluator.

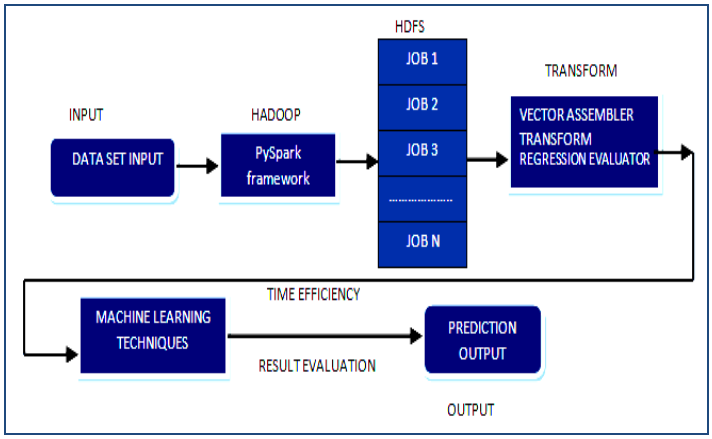

Figure 3: Model for handling the input data for prediction

As the job is splitted in the Hadoop Spark environment the machine learning algorithms are applied on the jobs for prediction from the training datasets. Each algorithm is tested by computing time as a parameter. Eventually time efficiencies and best prediction are obtained for further evaluations. The model is illustrated as a process in the figure -3. The prediction and time parameter constraints are calculated for efficiency learning methods.

V. ANALYSING TEMPERATURE DATA SET ON THE MODEL

The algorithm for analyzing the data through the model is illustrated as below.

Algorithm 1:

1: Read the data in spark data frame and use time method to invoke time.
2: Splitting the data into train data and test data using randomSplit function from spark.
3: Vector Assembler converts the data in terms of vectors.
4: Transform function modifies the vectors into necessary data frames.
5: Mapping the labelcol and featurecol using Machine Learning method from MLlib.
6: Pipeline consists of stages each acts as an estimator or a transformer when fit() is initiated.
7: Regression Evaluator uses to evaluate the prediction on the featured data.
8: Calculate the correlation coefficient from the data
9: Evaluate the coefficients from the covariance and correlation
10: Predict the values with the respective coefficients.
11: RMSE is calculated to find the mean square error.
12: Total evaluated time to process the data set is computed.

This fits the machine learning techniques to the data in spark environment using python language. The prediction for large real time data can be handled in spark using the algorithm1.

VI. RESULT AND EVALUATION

To better validate this framework real time datasets temperature of India corresponding to each month and year is been used. Spark tool is used as a framework. In this paper we build Linear Regression and Gradient Boosted tree model with the reduced training data, and evaluate it on the test data after applying the same feature selection. This model predicts temperature of year impression created with the test data.

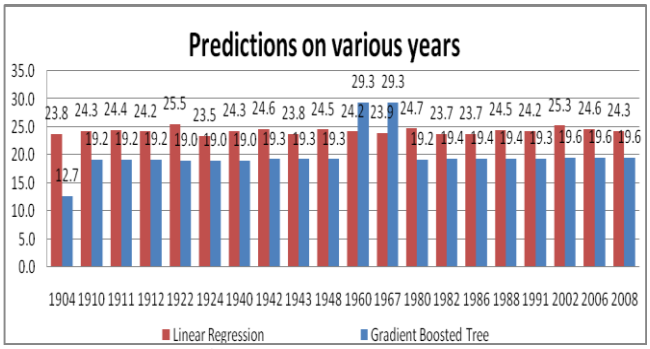

Figure 4: Predictions of Linear Regression and Gradient Boosted Tree with respect to annual

The results are illustrated on the Gradient Boosted tree and the Linear Regression Model. The prediction is made on the model created on the training dataset. If the model learning is effective, we expect the impression with high prediction to be actually resulting. Spark uses effective pipeline across various stages that gets updated to handle real time data efficiently.

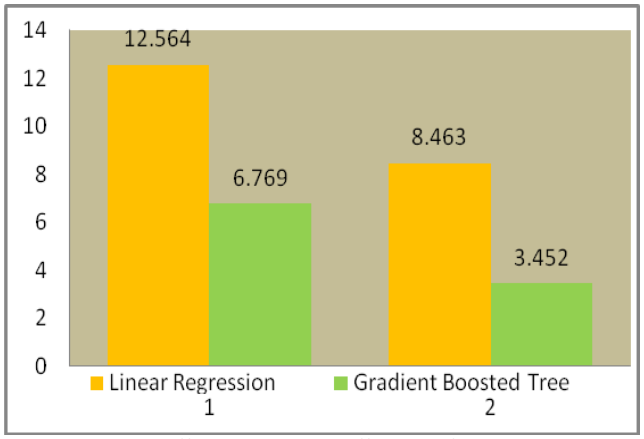

Figure 4: Time efficiency and space efficiency of machine learning techniques

The efficiency is calculated on comparisons on the datasets by the values such as time and space. The proposed algorithm takes o(n) times efficient for machine learning methods such as linear regression shows 12.564 Seconds it takes to store 750MB of data and gradient boosting tree 8.463 seconds for 750 MB. Another parameter space takes $o(n^2 \log n)$ in spaces as 6.769MB and 3.452MB is required for internal storage for linear regression and gradient boosting tree respectively in figure 4.

VII. CONCLUSION

In this paper a model is presented using pyspark that enables the development of large-scale machine learning algorithms. The languages in which machine learning algorithms are expressed are compared with highly optimized execution plans over existing techniques. Systematic empirical results in this paper have shown the benefit of a number of optimization strategies such as performance tuning reducing more no of disk I/O, latency, interactivity and the applicability scale up on diverse set of machine learning algorithms when compared with map reduce. Development of additional constructs to support machine learning meta-tasks such as model selection, and enabling a large class of algorithms to be probed at an unprecedented data scale.

These machine learning algorithms used for processing the dataset shows the better efficiency in context to time and space parameters. Gradient Boosting tree structure gives the appropriate results while comparing with Linear Regression. The usage of these algorithms in Spark framework improves the performance and also predicts the feature selection efficiently. So, application of Machine Learning algorithms using pyspark shows the competent results in prediction.

REFERENCES

[1] J.V.N. Lakshmi, S. Ananthi, "*A Theoretical Model for Big data Analytics using Machine Learning Algorithms*", In the Proceedings of the 2015 International Conference (WCI/ICACCI 2015), India, pp.632-636, 2015.

[2] Y. C. Kwon, H. Bill "*A Study of skew in MapReduce Application*", in International Conference , USA, pp. 234-245, 2014.

[3] J.V.N. Lakshmi "*Hadoop Spark Framework For Machine Learning Using Python*", In the proceedings of the 2016 National Conference on ACSE conference, India, pp.9-14, 2017 .

[4] Haroshi T., Shinji N., Takuyu A., "*Optimizing multiple machine learning jobs on map reduce"*, In IEEE – ICCCTS conference, Japan, pp. 59-66, 2011.

[5] C.-T. Chu, Lin, Y. Yu, G. R. Bradski, A. Y. Ng, K. Olukotun, "*Map-reduce for machine learning on multicore*", MIT Press, USA, pp. 281–288, 2006.

[6] Walisa, Wichan, "*An Adaptive ML on Map Reduce for improving performance of large scale data analysis 2013 Eleventh International Conference on ICT and Knowledge Engineering*, Bangkok, pp.1-7, 2013

[7] Asha, Sravanthi, "*Building Machine learning Algorithms on Hadoop for Big Data*", in IJET Journal, Vol 3, No 2, pp. 484-489, 2013.

[8] G. Schwarz, "*Estimating the dimension of a model*" , The annals of statistics, Vol.6, Issue.2, pp. 461–464, 1978.

[9] Caruana Rich, Nikos K, Ainur Y, "*An Empirical Evaluation of Supervised Learning in High Dimensions*", Proceedings of the

25th International Conference on Machine Learning, Finland, pp.96-103, 2008.

[10] M. Dhivya, D. Ragupathi, V.R. Kumar, "*Hadoop Mapreduce Outline in Big Figures Analytics*", International Journal of Computer Sciences and Engineering, Vol.2, Issue.9, pp.100-104, 2014.

Authors Profile

Ms. J V N Lakshmi is a Research Scholar of Research Department of Computer Science and Applications, Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya University at Kanchipuram, Tamil Nadu, and India. She received Master's Degree in Statistics in 2013, Master of Computer Applications (MCA) in 2008 and Bachelor of Science degree in Statistics and Computer Science in 2005 from Osmania University, Hyderabad, India. Her research interest is on Machine Learning, Big Data and Data Analytics. Her articles are published on Machine Learning Algorithms using Big Data in reputed journals include ACM and *IEEE* and also available online. She currently works as Assistant Professor at AIMS Institutes, Bangalore, India. She has 7 years of teaching experience

Dr Ananthi Sheshasaayee is a Research Supervisor of Research Department in , Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya University at Kanchipuram, Tamil Nadu, and India. She is Associate Professor heading the PG and Research Department in Quaid - E- Millath Government College, Chennai, India. She is a alleged Ph.D guide for many universities.

