ISSN: 2377-2220
http://dx.doi.org/10.21174/jomi.v2i1.101

LAZARUS SCIENTIFIC PUBLISHING

Big Data Classification Using Distributed Optimized Hoeffding Trees

Sharmishta Desai[1,*], Dr. S. T. Patil [2]

[1]*Savitribai Phule Pune University, India*
[2] *Vishwakarma Institute of Technology, Pune University, India*

***Abstract*—**Large usage of social media, online shopping or transactions gives birth to voluminous data. Visual representation and analysis of this large amount of data is one of the major research topics today. As this data is changing over the period of time, we need an approach which will take care of velocity of data as well as volume and variety. In this paper, author has proposed a distributed method which will handle three dimensions of data and gives good results as compared to other method. Traditional algorithms are based on global optima which are basically memory resident programs. Our approach which is based on optimized hoeffding bound uses local optima method and distributed map-reduce architecture. It does not require copying whole data set onto a memory. As the model build is frequently updated on multiple nodes concurrently, it is more suitable for time varying data. Hoeffding bound is basically suitable for real time data stream. We have proposed very efficient distributed map-reduce architecture to implement hoeffding tree efficiently. We have used deep learning at leaf level to optimize the hoeffding tree. Drift detection is taken care by the architecture itself no separate provision is required for this. In this paper, with experimental results it is proved that our method takes less learning time with more accuracy. Also distributed algorithm for hoeffding tree implementation is proposed.

Keywords—Big Data; Hoeffding Trees; Social Media; Decision Trees.

I. INTRODUCTION

While Big Data is perhaps not an entirely new concept, it is certainly a hot topic of the modern, digital era. However, it is an evolving concept, and any standards created must take into account that what is classed as 'Big Data' today is likely to change rapidly over the next few years. Standards must therefore look beyond the 'here and now' of how Big Data is currently being used and instead seek to establish frameworks for dealing with data sets that represent a significant logistical challenge [11]. Taking decisions based on large amount of past and current data is one of emerging topic today [4,5]. For example, predicting exchange rates or forecasting stock prices based on past records. This data is enormous in volume. Data is changing with respect to time. Also, data format is changing enormously. Decision tree algorithms are more suitable for this kind of data as it gives visual representation of data which is beneficial for analysis. Traditional decision tree algorithms like C4.5,ID3, Random forest cannot handle large volume of data due to their large memory requirement[16]. These methods are based on global optima which need to copy whole data onto memory to design a model. So the new approach is required which will take care three dimensions of Big data (Volume, Variety and Velocity).Our approach is based on distributed implementation of Hoeffding trees. Hoeffding trees are based on Hoeffding bound which is used to reduce tree height and increase accuracy. MapReduce architecture is used to implement distributed system. In the paper below, we have explained literature survey in section-II, Proposed Methodology in section-III and Experimental Results in section-IV.

II. LITERATURE SURVEY

In literature, many authors have used different machine learning algorithms to analyses time changing big data. In [1], author has used neural network to analyze behaviors of customers using social media data set. In [2], author has explained a way to find link between to users' social media like Twitter or Facebook using machine learning algorithm. In [3,4,5],

* Corresponding author can be contacted via the journal website.

authors have explained big data architecture, challenges etc. In work done by Isvani Frıas-Blanco, Jose del Campo-Avila [29], moving average method is suggested which is used for Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds.Short-Term Load Forecasting Based on Big Data Technologies by Pei Zhang [31], explain decision tree framework to forecast short term load like electricity. Petra Perner has explained how decision tree induction is suitable than traditional methods in his paper "Decision Tree Induction Methods and their Applications to Big Data".Also there is lot of work available on social media data analysis. Java et al. [21] have studied the topological and geographical properties of Twitter's social network. Characteristics of social activities and patterns of communication in Twitter are studied by Naaman et al. [23]. Davidov et al. [24] have used hash tags and other sentiment labels for sentiment analysis. An effective and efficient followed recommender system built by Hannon et al [25]. Methods to recommend influential users proposed by Kwak et al [26]. Twitter use within and across organizations and geographic markets comparison is proposed by Burton et al. [27]. Kim et al. [28], explained how to maximize the outcomes of SMM through Word-of-Mouth (WOM) marketing by identifying the core group of users. On distributed implementation of decision trees some work is available. In [28], author has implemented C4.5 decision tree algorithm using map-reduce. In [30], author has explained how to extract knowledge using decision Tree and Naïve Bayes Algorithm for Classification and Generation of Actionable Knowledge for Direct Marketing. Distributed implementation of support vector machines is proposed by [10]

III. Proposed Methodology

Authors are requested to submit papers as the data is dynamic, we need an approach which is based on local optima model. This model does not require whole data set present on a memory as well as it can be modified as and when required. To avoid missing, biased and concept drift data, we have concentrated on preprocessing technique. Proposed technique is based on distributed architecture which contains all steps from data preprocessing to tree optimization. This technique is based on horizontal splitting of samples. It is shown in figure 1.

A. Overall Flow of Processing

Algorithm *A.1* explains the entire process.

Central node is responsible for continuous stream reading and splitting it among different mapper nodes. Every mapper node will collect stream assigned to it. Stream will be processed to remove noisy, missing data. It will calculate statistics like entropy and information gain for every attribute Ai Є A, i varies from 1 to n. n is the number of attributes. Based on statistics, mapper will create local optimal tree based on set of samples assigned to it. This tree will be forwarded to respective reducer.

Every reducer will combine the trees received from respective mapper. The algorithm for combining local trees is given below. After combining a tree, tree pruning will be done to remove unused attribute. This technique will be applied by every mapper node to prune local trees also.

As the data preprocessing, statistics computation and tree creation tasks are divided among multiple mapper and reducer nodes, these tasks will be done parallel and the learning time will get reduce. Tree pruning will be done at every node to minimize the tree size and increase the accuracy.

At leaf level, different other classifiers like Naïve Bayes, Naïve Bayes Adaptive and majority class are used to avoid biased data and to increase accuracy.

This technique is based on vertical splitting of data. Data samples will be divided attribute wise. Set of attributes (Ai….At) along with class label attribute (Ac) will be assigned to each node. As the number of attributes increases, number of map-reduce slots can be increased. So, vertical splitting is more suitable for big data.

Drift is detected at very local mapper node and accordingly tree model is modified. This change will be automatically considered by reducer node.

Algorithm A.1: Tree Creation
INPUT
X: Stream of Samples
A: Set of attributes
f(A)- Function to split node
δ – Priority of choosing correct attribute
OUTPUT
T(A,C)- Tree with A attributes and C class labels
PROCEDURE: Create_Tree(X,A,f(A), δ)
Read Data Stream X.
Divide stream X into set of streams
X1,X2,…….where every set will contain t samples.
Assign stream Xi to MapperNodei
Distribute Stream X among different nodes using
MAP_DATA (Xt, A, f(X), δ)
Collect class labels from different nodes using
REDUCE_DATA (Xt1,f(Xt1),Ct),
(Xt2,f(Xt2),Ct),…….

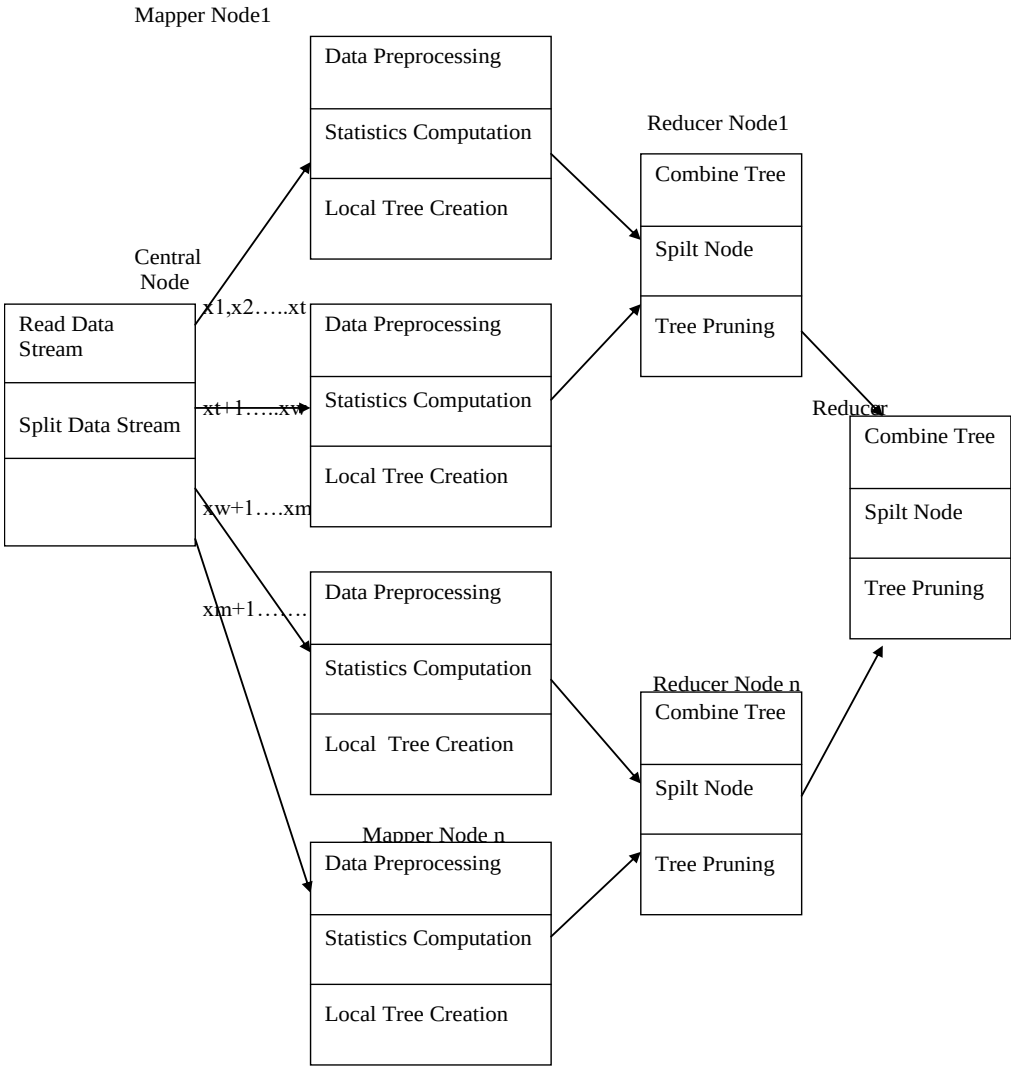

Figure 1. Distributed Architecture for Big Data Classification

B. Normalizing Leaf Node

If data samples at leaf node belong to different classes then we need normalize the leaf node. For this, Algorithm B.1 is applied to split a leaf node.

In this, entropy f(X) is calculated for all attributes X, Out of this, attribute Xta with max f(x) value and Xtb with second highest f(x) value is selected.

Two attributes entropy value is compared against hoeffding bound, HB.The attribute with highest entropy value f(x) is selected as parent node and nodes having less entropy value f(x) are added left side and rest are added on right

Algorithm B.1: Splitting Node
PROCEDURE: Splitnode(X,A,f(X), δ)
Select attribute f(Xta) > = max(f(Xt)) and max(f(Xt)) <= f(Xtb)< f(Xta) from each node.
Calculate HB = for each node
Select Xta as splitting attribute if f(Xta) – f(Xtb) > HB
Xta→left= all Xti ; f(Xti) < f(Xta)
Xtb→right= all Xti ; f(Xti) > f(Xta)

C. Mapper Processing

Every data stream is divided into sub data streams and assigned to different mapper nodes. Mapper node is responsible for doing data preprocessing, statistics calculations and creating local tree which is given in Algorithm C.1

Algorithm C.1 :Mapping Data
PROCEDURE MAP_DATA(Xt, A, f(X), δ)
Call Data_Preprocessing(Xt, A, f(X), δ)
Call Statistics_calc(Xt, A, f(X), δ)
Call Create_Tree(Xt, A, f(X), δ)

D. Reducer Processing

Reducer node task is to combine local trees received from respective mapper node. Then tree pruning is applied to remove same information value attribute.It is explained in algorithm 3.4.

Algorithm D.1: Reducing Data
PROCEDURE:
REDUCE_DATA(T1(A1,C1),T2(A2,C2))
Collect two local trees from two mappers
T1(A1,C1) & T2(A2,C2)
Combine two local tree into global tree(GT,Cg)
using procedure combine_tree
(T1(A1,C1),T2(A2,C2))
Call tree_prunning(GT,Cg) on global tree(GT,Cg)
Pass the pruned tree to next reducer

E. Data Preprocessing

In data preprocessing (Algorithm 3.5), missing values of attributes are replaced by mean values. Useless, meaningless data is removed by using variance.

Algorithm E.1: Data Preprocessing

PROCEDURE Data_Preprocessing(Xt, A, f(X), δ)

For all samples Xi ϵ {X1,X2,……}

For all Xi ϵ {‘ ‘ }

$$\hat{X} = \frac{\sum_{i=1}^{n} Xi}{n}$$

Calculate $\frac{\sum_{i=1}^{n} Xi - Xavg}{n-1}$

Calculate S2 =

If S2 > 90 % then remove(Xi)

F. Statistics Calculation

For attribute selection, different statistics like entropy, information gain are calculated. Attribute with highest information gain is selected as a node in a tree.

Entropy is measure of impurity level. Higher is the entropy more is the information value of an attribute. If we create decision tree based on entropy, then it tends to select attribute which forms a node with many branches.

So, Information gain is used which decides proper ordering of attributes. Information gain is difference between entropy before split and entropy after split of an attribute. It is explained in algorithm 3.6.

Algorithm F.1: Statistics Calculation

PROCEDURE Statistics_calc(Xt, A, f(X), δ)

Compute Entropy for all Ai,

$$E(Ai) = \sum_{i=1}^{t} P(Ai)\log(P(Ai))$$

Where t are the number of samples for Attribute Ai

Compute Information Gain for all Ai,

I(Ai)= E1(Ai) – E2(Ai)

Where E1(Ai) is entropy before split and E2(Ai) is entropy after split

G. Combining Local Trees

For combining two local trees, information gain of roots of two trees are compared with Hoeffding bound(HB). The root node with more probality is selected as root of combined tree, and node having less information gain is added on left side and other nodes on right side.

After forming single tree, leaf level nodes class distribution is checked. if it is biased or uneven then different classifiers like naïve Bayes are used.

Algorithm G.1: Combining Local Trees
PROCEDURE Combine_trees
(T1(A1,C1),T2(A2,C2))
Repeat For all R1εA1 and R2 ε A2,
Compute Information Gain, I
If I (R1) - I(R2) > HB then
Select R1 as root and add R1→Left=R2
Else:
Select R2 as root and add R2→Left=R1
End if all attributes of A1 and A2 are included
For all Xi ϵ {leaf Nodes},
Fnb:arg max r={n`i,j1,n` i,j,2…..,n`i,j,r}
n`i,j,r=P(X|cf).P(cf)/P(X)

H. Pruning Tree

After forming a tree at reducer node, pruning is done to remove less useful attributes. While creating local trees, it might have happened that one attribute has good information gain. But after combining trees this attribute may lose information or may have same information like other attributes, then their information gain is compared against hoeffding bound(HB). It is explained in Algorithm 3.8.

Algorithm H.1: Pruning Tree

PROCEDURE Tree_prunning(GT,Cg)

Traverse all Ai ε GT

Compute Information Gain, I

For attributes A1,A2

If I(A1) = = I(A2)

Calculate HB = $\sqrt{\frac{R2In(\frac{1}{\delta})}{2n}}$

If I(A1)-I(A2) > HB

Omit A2 from the tree

Readjust the tree

IV. Experimental Work

Experimental Wok is carried out on Windows 7 Operating System with Cygwin and Hadoop0.19.1 version. Hadoop cluster of 6 nodes is created. Distributed Hoedding tree is executed on Hadoop cluster. Hadoop is configured with Eclipse Europa.

Initially results are generated for sequential hoeffding tree and compared with other decision tree algorithms. These results (Table 1) are generated Using Java Weka 3.7.11 library and Eclipse. These results show that C.4.5 works well till 50000 attributes. Hoeffding tree works well though attributes beyond 50000.Though data increases accuracy of hoeffding tree remains linear.

Table 1. Comparison of Different Decision Tree Algorithms.

Algorithms	No of Attributes	Time (in Seconds)	Accuracy (%)
C4.5	10000	0.28	72.74
Random Forest	10000	0.47	70.59
Hoeffding Tree	10000	0.17	73.83
C4.5	50000	1.45	73
Random Forest	50000	Fails	Fails
Hoeffding Tree	50000	0.78	73.662
C4.5	100000	Fails	Fails
Random Forest	100000	Fails	Fails
Hoeffding Tree	100000	1.56	73.9
C4.5	1000000(1GB)	Fails	Fails
Random Forest	1000000(1GB)	Fails	Fails
Hoeffding Tree	1000000(1GB)	74.78	74

Results of distributed Hoeffding tree are compared with sequential Hoeffding tree results. Though the numbers of samples are increasing, learning speed of distributed Hoeffding tree is lesser than sequential Hoeffding Tree. It is shown in Figure 2. Here,Number of attributes are kept same i.e.100.

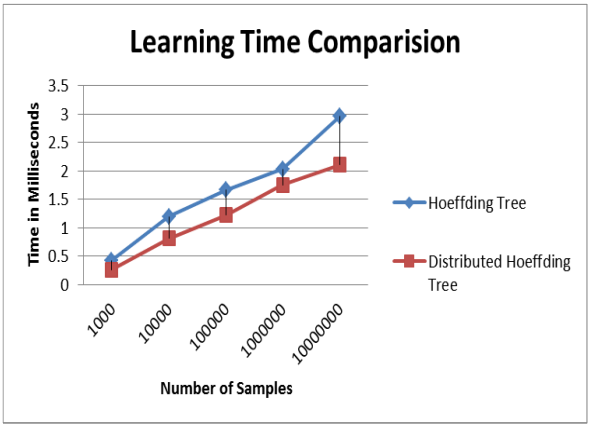

Figure 2. Learning Time Comparison (Number of attributes=100)

Learning speed of distributed and sequential hoeffding tree algorithm is compared for different number attributes. It is shown in Figure 3. As the number of attributes increases, the learning speed of distributed as well as sequential hoeffding tree increases. But distributed algorithm's learning speed is three times more while sequential algorithm is four times more.

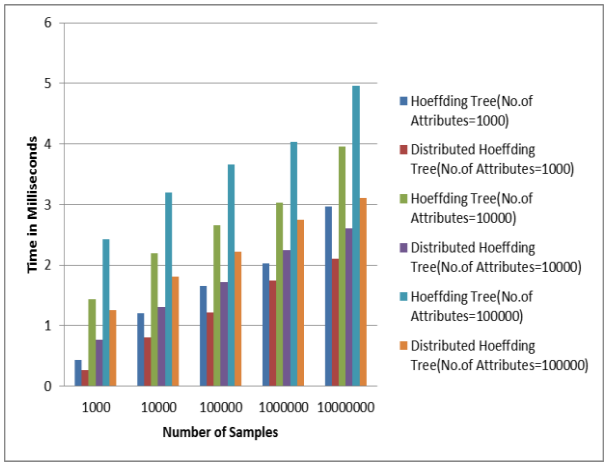

Figure 3. Learning Time Comparison (Different Number of Attributes)

As we are implementing distributed hoeffding tree, the learning time is get reduced. This is obvious. But though the model is distributed the accuracy is more than sequential architecture. It is shown in Figure 4. As the number of samples increases, sequential hoeffding tree accuracy declines while distributed hoeffding tree accuracy is steady. This is because we are creating local tree at every mapper node, we are splitting leaf node if class distribution is biased. At every reducer node, tree pruning is applied to remove irrelevant attributes.

To increase the accuracy of Distributed Hoeffding tree, we have used different classifiers at leaf level. This helps to remove biased data. The results are shown in Table-2.

Table 2. Comparison of Algorithms Used at Leaf Level

Algorithms Used at leaf level	No of attributes	Correctly Classified	Incorrectly Classified	Time in Milliseconds	Accuracy (%)
Majority Class	10000	1050	8950	0.13	10.5
Naïve bayes	10000	7383	2617	0.05	73.83
Naive bayes Adaptive	10000	7383	2617	0.14	73.83

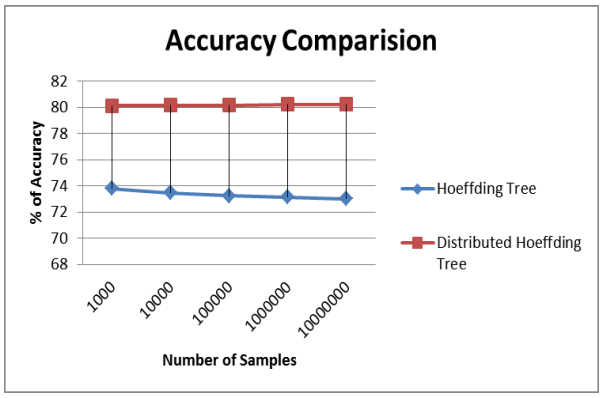

Figure 4. Accuracy Comparison (Number of attributes=100)

Comparison Accuracy of different classifiers used at leaf is shown in Figure 5. Naive Bayes and Naïve Bayes Adaptive shows better results than majority class.

I. Conclusion

Big Data analysis is one of the emerging topic today which is used to analyze past and current data any domain like financial, medical, social etc. This analysis is useful to take many decisions. As data changes in volume, velocity and variety, we need an approach

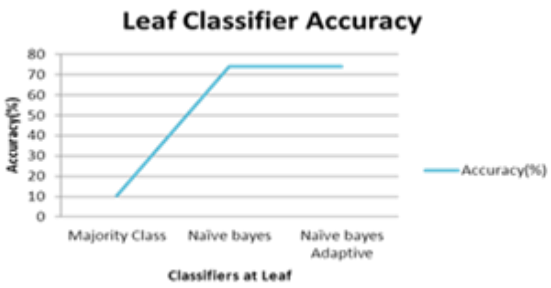

Figure 5. Leaf Classifier Accuracy Comparison (Number of attributes=100)

which will take all these three aspects. In our paper, we have proposed a distributed model which is based on hoeffding trees and map-reduce architecture. This model does not require whole data set to be present on a memory.

Local optimum tree models are generated at every node by taking care of drift (changed data) and also biased data.Tree pruning is done to remove same meaning or less useful attributes which keeps tree height minimum. With experimental results, it is proved that our proposed model gives better accuracy as well as learning speed. Different other classifiers like Naïve Bayes are used at leaf level to increase accuracy of model.

Acknowledgement

We would like to thank MIT College of Engineering, Pune, India for providing infrastructure to do this research. Also, we like to thank Savitribai Phule Pune University, Pune, India for sponsoring our research topics under BCUD research grant scheme.

References

[1] Badea, Laura Maria. "Predicting Consumer Behavior with artificial neural networks." *Procedia Economics and Finance* 15 (2014): 238-246. DOI=https://doi.org/10.1016/S2212-5671(14)00492-4

[2] Mengshoel, Ole J., Raj Desai, Andrew Chen, and Brian Tran. "Will we connect again? machine learning for link prediction in mobile social networks." *In Eleventh Workshop on Mining and Learning with Graphs*. 2013.

[3] Yadav, Chanchal, Shuliang Wang, and Manoj Kumar. "Algorithm and approaches to handle large Data-A Survey." *arXiv preprint arXiv*:1307.5437 (2013).

[4] Bakshi, Kapil. "Considerations for big data: Architecture and approach." In *Aerospace Conference, 2012 IEEE*, pp. 1-7. IEEE, 2012. DOI=https://doi.org/10.1109/aero.2012.6187357

[5] Pulse, Global, "Big Data for Development: Challenges and Opportunities", (2012)

[6] Zaharia, Matei, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. "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, pp. 2-2. USENIX Association, 2012.

[7] Bawa-Cavia, Anil. "Sensing the urban: using location-based social network data in urban analysis." *In Pervasive PURBA Workshop*. 2011.

[8] Cho, Eunjoon, Seth A. Myers, and Jure Leskovec. "Friendship and mobility: user movement in location-based social networks." In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1082-1090. ACM, 2011. DOI=https://doi.org/10.1145/2020408.2020579

[9] Tang, Lei, and Huan Liu. "Leveraging social media networks for classification." *Data Mining and Knowledge Discovery* 23, no. 3 (2011): 447-478. DOI=https://doi.org/10.1007/s10618-010-0210-x

[10] Catak, F. Ozgur, and M. Erdal Balaban. "CloudSVM: training an SVM classifier in cloud computing systems." In *Joint International Conference on Pervasive Computing and the Networked World*, pp. 57-68. Springer Berlin Heidelberg, 2012.

[11] The Big Data and standards market research report, January 2016

[12] Yeh, Chi-Yuan, Wen-Pin Su, and Shie-Jue Lee. "Employing multiple-kernel support vector machines for counterfeit banknote recognition." *Applied Soft Computing* 11, no. 1 (2011): 1439-1447. DOI=https://doi.org/10.1016/j.asoc.2010.04.015

[13] Kim, Kwang In, Keechul Jung, Se Hyun Park, and Hang Joon Kim. "Support vector machines for texture classification." *IEEE transactions on pattern analysis and machine intelligence* 24, no. 11 (2002): 1542-1550. DOI=https://doi.org/10.1109/TPAMI.2002.1046177

[14] Vaidya, Jaideep, Basit Shafiq, Wei Fan, Danish Mehmood, and David Lorenzi. "A random decision tree framework for privacy-preserving data mining." *IEEE transactions on dependable and secure computing* 11, no. 5 (2014): 399-411. DOI=https://doi.org/10.1109/TDSC.2013.43

[15] Desai, Sharmishta, and S. T. Patil. "Differential Evolution algorithm with Support Vector Machine to classify objects efficiently." *International Journal* 2, no. 3 (2014).

[16] Desai, Sharmishta, and S. T. Patil. "Efficient regression algorithms for classification of social media data." In *Pervasive Computing (ICPC), 2015 International Conference on*, pp. 1-5. IEEE, 2015. DOI=https://doi.org/10.1109/pervasive.2015.7087040

[17] Alabhya, F. "Analyzing trends in social media marketing." *Int. J. Comput. Sci. Manag. Stud* 14, no. 11 (2014): 1-5.

[18] Tom Mitchell, “Decision Tree Learning”, Princeton University

[19] Tang, Liyang, Zhiwei Ni, Hui Xiong, and Hengshu Zhu. "Locating targets through mention in Twitter." *World Wide Web* 18, no. 4 (2015): 1019-1049. DOI=https://doi.org/10.1007/s11280-014-0299-8

[20] Naaman, Mor, Jeffrey Boase, and Chih-Hui Lai. "Is it really about me?: message content in social awareness streams." In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pp. 189-192. ACM, 2010. DOI=https://doi.org/10.1145/1718918.1718953

[21] Davidov, Dmitry, Oren Tsur, and Ari Rappoport. "Enhanced sentiment learning using twitter hashtags and smileys." In *Proceedings of the 23rd international conference on computational linguistics: posters*, pp. 241-249. Association for Computational Linguistics, 2010.

[22] Hannon, John, Mike Bennett, and Barry Smyth. "Recommending twitter users to follow using content and collaborative filtering approaches." In *Proceedings of the fourth ACM conference on Recommender systems*, pp. 199-206. ACM, 2010. DOI=https://doi.org/10.1145/1864708.1864746

[23] Kwak, Haewoon, Changhyun Lee, Hosung Park, and Sue Moon. "What is Twitter, a social network or a news media?." In *Proceedings of the 19th international conference on World wide web*, pp. 591-600. ACM, 2010. DOI=https://doi.org/10.1145/1772690.1772751

[24] Burton, Suzan, and Alena Soboleva. "Interactive or reactive? Marketing with Twitter." *Journal of Consumer Marketing* 28, no. 7 (2011): 491-499. DOI=https://doi.org/10.1108/07363761111181473

[25] Kim, Yong Seog, and Vinh Loc Tran. "Selecting Core Target Users for Online Social Networking Marketing with Target Marketing: A Preliminary Report." In *AMCIS*. 2011.

[26] Adomavicius, Gediminas, and YoungOk Kwon. "Improving aggregate recommendation diversity using ranking-based techniques." *IEEE Transactions on Knowledge and Data Engineering* 24, no. 5 (2012): 896-911. DOI=https://doi.org/10.1109/TKDE.2011.15

[27] Tang, Lei, and Huan Liu. "Leveraging social media networks for classification." *Data Mining and Knowledge Discovery* 23, no. 3 (2011): 447-478. DOI=https://doi.org/10.1007/s10618-010-0210-x

[28] Dai, Wei, and Wei Ji. "A mapreduce implementation of C4. 5 decision tree algorithm." *International Journal of Database Theory and Application* 7, no. 1 (2014): 49-60. DOI=https://doi.org/10.14257/ijdta.2014.7.1.05

[29] Frías-Blanco, Isvani, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Rafael Morales-Bueno, Agustín Ortiz-Díaz, and Yailé Caballero-Mota. "Online and non-parametric drift detection methods based on Hoeffding’s bounds." *IEEE Transactions on Knowledge and Data Engineering* 27, no. 3 (2015): 810-823. DOI=https://doi.org/10.1109/TKDE.2014.2345382

[30] Karim, Masud, and Rashedur M. Rahman. "Decision tree and naive bayes algorithm for classification and generation of actionable knowledge for direct marketing." (2013).

[31] Zhang, Pei, Xiaoyu Wu, Xiaojun Wang, and Sheng Bi. "Short-term load forecasting based on big data technologies." *CSEE Journal of Power and Energy Systems* 1, no. 3 (2015): 59-67. DOI=https://doi.org/10.17775/CSEEJPES.2015.00036