Parallel Computing to Predict Breast Cancer Recurrence on SEER Dataset using Map-Reduce Approach

Umesh D. R.
Assistant Professor
Department of Computer Science & Engineering
PESCE, Mandya, Karnataka, India.

B. Ramachandra, PhD
Professor
Department of Electrical & Electronics Engineering
PESCE, Mandya, Karnataka, India.

ABSTRACT
Due to the late overpowering development rate of large scale data, the advancement of handling faster processing algorithms with optimal execution has turned into a critical need of the time. In this paper, parallel Map-Reduce algorithm is proposed, that encourages concurrent participation of various computing hubs to develop a classifier on SEER breast cancer data set. Our algorithm can prompt supported models whose speculation execution is near the respective baseline classifier. By exploiting their own parallel architecture the algorithm increases noteworthy speedup. In addition, the algorithm don't require singular processing hubs to communicate with each other, to share their data or to share the knowledge got from their data and consequently, they are powerful in safeguarding privacy of computation also. This paper utilized the Map-Reduce framework to implement the algorithms and experimented onSEER breast cancer data sets to exhibit the execution as far as classification accuracy and speedup.

Keywords
Breast cancer; Big dataanalytics, Classification; Parallel Computing; MapReduce, SEER.

1. INTRODUCTION
Data evolution has been quickly moving from the Terabytes to the petabytes age as an aftereffect of the blast of data. The potential worth and bits of knowledge which could be derived from massive data sets have pulled in gigantic enthusiasm for an extensive variety of business and scientific applications [1–3]. It is turning out to be increasingly critical to arrange and use the huge measures of data at present being created. However, with regards to massive data, it is troublesome for current data mining algorithms to build classification models with serial algorithm running on single machines, also exact models. In this way, the requirement for proficient and powerful models of parallel computing is obvious.

Fortunately, with the assistance of the MapReduce [3–6] infrastructure, specialists now have a basic programming interface for parallel scaling up of numerous data mining algorithms on larger data sets. It was indicated [7] that algorithms which fit the Statistical Query model [8] can be composed in a specific "summation form". They represented 10 different algorithms that can be effectively parallelized on multi-core personal computers applying the MapReduce paradigm.

Despite the fact that MapReduce handles extensive scale computation, it doesn't support iteration. Since there are no loop steps available in Hadoop, to execute loops, an outside driver is expected to repeatedly submit MapReduce jobs. Since each MapReduce jobs works independently, keeping in mind the end goal to reuse information between MapReduce jobs, the outcomes created by a previous MapReduce jobs are composed to the Hadoop Distributed File System (HDFS) and the following MapReduce job which needs this data as inputs peruses these messages from HDFS. Clearly, this operation doesn't have the advantages that the caching system can get for the in-memory computation. Additionally, attributable to data replication, disk I/O, and serialization, the methodology for making loops inside the original version of Hadoop causes enormous overheads. The time spent in this procedure may some of the time possess a noteworthy part in the aggregate execution time.

Presently, there are some methodologies [9–14] which manage the issue of lacking iterations in MapReduce. To effectively handle such large scale data, faster processing and optimization is turning out to be more critical. Subsequently, it has become vital to develop new algorithms that are more reasonable for parallel models. One straightforward methodology could be to convey a single inherently parallelizable data mining program to multiple data (SPMD) on numerous personal computers. In any case, for algorithms that are not inherently parallelizable in nature, upgrading to accomplish parallelization is the option.

In this paper, the proposed parallel algorithms, which accomplish parallelization in both time and space. Parallelization in space is additionally vital as a result of the restricting variable postured by the memory size. Large data sets that cannot fit into the main memory are regularly expected to swap between the main memory and the secondary storage, presenting latency cost which sometimes may even decrease the speedup picked up by parallelization in time. The proposed algorithms are intended to work in cloud environment where every hub in the computing cloud works just on a subset of the entire data. The joined impact of all the parallel working hubs is a supported classifier model prompted much speedier and with a fabulous speculation capacity.

The demonstration of the algorithm keeps up a competitive test exactness, which accomplishes significant speedup contrasted with big data analytics using Map-Reduce (BDAM), which is equipped for being fitted in a parallel design; and demonstrated that Parallel Computing using Map-Reduce (PCM) algorithm performs better both in terms of prediction accuracy and speedup. For the implementation, Map-Reduce [15] framework has been utilized, which a straightforward model for distributed cloud computing.

This paper is organized as follows: section "Related work" introduces work that has previously been proposed for solving the problem in Hadoop MapReduce; section "Proposed Algorithm" presents a new parallel framework PCM

algorithm that accomplishes parallelization in both time and space; section "Experimental Result" demonstrate experimentally the predominance of the proposed algorithm over BDAM as far as prediction accuracy and speedup; section "Conclusion" concludes the paper.

2. RELATED WORK

ADABOOST is one of the most popular boosting algorithm proposed in the mid-1990s [16]. Its straightforward natural algorithmic flow joined with its sensational change in the speculation execution makes it a standout among the most intense ensemble techniques. A clear hypothetical clarification of its execution is all around portrayed in [17], where boosting in a two class setting is seen as an additive logistic regression model.

LOGITBOOST is another generally utilized boosting algorithm which is proposed using additive modeling and is appeared to show more powerful execution particularly in the presence of noisy data.

FILTERBOOST [18] is a recent algorithm of the same kind, based on a modification of ADABOOST intended to minimize the logistic loss. FILTERBOOST expect an oracle that can deliver boundless number of labeled samples and in each boosting iteration, the oracle creates sample points that the base learner can either accept or reject. A small subset are utilized to prepare the base learner.

Escudero et al. [19] proposed LAZYBOOST for accelerating ADABOOST, which utilizes several feature selection and ranking techniques. In each boosting iteration, it picks a fixed-size arbitrary subset of features and the base learner is prepared just on this subset. Another fast boosting algorithm in this category was proposed by Busa-Fekete and Ke´gl [20], which uses multiple-armed bandits (MAB). In the MAB-based methodology, every arm speaks to a subset of the base classifier set. One of these subsets is chosen inevery iteration and after that the boosting algorithm seeks just this subset as opposed to optimizing the base classifier over the whole space.However, none of these works portrayed so far investigate accelerating boosting in a parallel or distributed setting and in this way their execution is restricted by the resources of a solitary machine.

Wu et al. [21] proposed an ensemble of C4.5 classifiers taking into account MapReduce called MReC4.5. By giving a progression of serialization operations at the model level, the classifiers based on a cluster of computers or in a cloud environment could be utilized as a part of different situations. PLANET [22] is another as of late proposed system for learning classification and regression trees on enormous data sets utilizing MapReduce. These methodologies are particular to the weak learners, (for example, tree models) and consequently don't show up as a general system for ensemble techniques such as boosting.

In spite of these endeavors, there has not been any huge examination to parallelize the boosting algorithm itself. Prior forms of parallelized boosting [23] were basically intended for tightly coupled shared memory frameworks and henceforth is not appropriate in a distributed cloud environment. Fan et al. [24] proposed boosting for adaptable also, distributed learning, where every classifier was prepared utilizing just a small portion of the training set. In this distributed adaptation, the classifiers were prepared either from irregular samples (r-sampling) or from disjoint partitions of the data set (d-sampling). This work fundamentally centered on parallelization in space however not in time. Henceforth, despite the fact that this methodology can deal with substantial data by distributing among the hubs, the objective of faster preparing time is not accomplished by this methodology.

Gambs et al. [25] proposed MULTBOOST algorithm which permits participation of two or more working hubs to develop a boosting classifier in a security safeguarding setting. In spite of the fact that initially intended for saving protection of algorithm, MULTBOOST's algorithmic design can fit into a parallel setting. It can accomplish parallelism both in space and time by requiring the hubs to have separate data and by empowering the hubs to process without thinking about other specialists' data.

However, the primary issue of these aforementioned methodologies is that they are reasonable for low latency intercomputer communication environments, for example, conventional shared memory architecture or single machine multiple processors frameworks and are not appropriate for a distributed cloud environment where for the most part the correspondence expense is higher. A huge part of the time is used for imparting data between the registering hubs instead of the real computation. In this proposed methodology, the constraint by making the hubs computation independent from each other thus minimizing these communications has been overcome.

3. PROPOSED ALGORITHM

In this section, the proposed PCM algorithm has been described. Before that, big data analytics using MapReduce framework has been described, in brief. The pseudocode for big data analytics using Map-Reduce (BDAM) is described in Algorithm-1. Let the data set D_n=*{(x_1, y_1), (x_2, y_2), ... ,(x_m, y_m)} with label classification* y_i ∈ *{*Recurrence (R), *Non-Recurrence (NR)};* x_i ∈ X is the object or instance; The algorithm initialize all the records with weight, so that$D_1(i) = \frac{1}{m}$for all the examples in D_m, where $t \in [1, T]$ and T is the total number of iterations. Before starting the first iteration these weights are uniformly initialized (line 1) and they are updated in every consecutive iteration. At each iteration, a weak learner function is applied to the weighted version of the data which then returns an optimal weak hypothesis h_t (line 5). This weak hypothesis minimizes the weighted error. At each iteration, a weight is assigned to the weak classifier (line 7). At the end of T iterations, the algorithm returns the final classifier H which is a weighted average of all the weak classifiers. The sign of H is used for the final prediction.

Algorithm-1: BDAM (D_n, T)

Input: Consider SEER dataset of n records (x_1,y_1), ……, (x_n, y_n) with label classifications $y_i \in$ Y = {Recurrence (R), Non-Recurrence (NR)}*;* x_i ∈ X is the object or instance; Base learner *B; and* Number of iterations *T*

Output: The final classifier $H_{final}(x)$

1. Initialize all the records with weight, so that $D_1(i) = \frac{1}{n}$
2. for t ← 1 to T do
3. Create distribution D_t on {1,…..,n} from the selected training subset S_t
4. Call base learner B, train B with S_t
5. Select weak classifier with smallest error rate (ε_t) on D_t

$$\varepsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i]$$
$$h_t : x \rightarrow \{R, NR\}$$

6. if $\varepsilon_t > 0.5$, then set T = t – 1 and exit from loop.
7. Update distribution $D_{t+1}(i) = \frac{D_t(i)}{Z_t}\ C(x)$
   C(x) = $\frac{\varepsilon_t}{1-\varepsilon_t}: y_i = h_t(x_i)$
   $1 : y_i \neq h_t(x_i)$
   $\alpha_t = \log\frac{1-\varepsilon_t}{\varepsilon_t} > 0$
   Z_t → Normalization constant ≤ 1
8. Output: The final classifier $H_{final}(x) = argmax_{yi \in Y} \sum_{t:h_t(x)=y} \alpha_t$

Computational Complexity of BDAM depends on the weak learner algorithm in line 4. Rest of the operations can be performed in Θ(*n*). Let's consider decision trees with only two leaf nodes as weak learners. Then the cost is Θ(*dn*) if the data examples are sorted in each attribute. Sorting all the attributes will take Θ(*dn* log *n*) time and this has to be done only once before starting the first iteration. So, the overall cost of the *T* iterations is Θ(*dn*(*T* + log *n*)).

The pseudocode for PCM is depicted as a part of Algorithm-2, let $\boldsymbol{D^p_n}$is the data set for the p[th] hub. The hubs compute classifier H_p by finishing all the T iteration of Algorithm-1 on their particular data sets (line 2). *H_p is defined as follows:*

$\{(h_p^1, \alpha_{p(1)}), (h_p^2, \alpha_{p(2)}), \ldots (h_p^T, \alpha_{p(T)})\}$

where h_p^t *is the weak classifier of* p[th] hub at t[th] iteration and $\alpha_{p(t)}$ is the corresponding weight of that weak classifier. The worker than reorders the weak classifier, h_p^t, *with increasing order of* $\alpha_{p(t)}$ (line 3). Thus, reordering H_p^* *communicated as follows:*

$\{(h_p^{*1}, \alpha_p^*{}_{(1)}), (h_p^{*2}, \alpha_p^*{}_{(2)}), \ldots (h_p^{*T}, \alpha_p^*{}_{(T)})\}$.

If, $\alpha_{p(k)} = \min\{\alpha_{p(t)}) \mid t \in \{1, 2, \ldots, T\}\}$ then $\alpha_p^*{}_{(1)} = \alpha_{p(k)}$ and $h_p^{*1} = h_p^k$. *Now, the reordered* h_p^{*t} *s are considered for converging in the T number of iterations for the final classifier.* $h_t(x)$ *is formed by converging* $\{(h_1(t), ..., h_M(t)\}$ (line 6) where, these weak classifiers does not necessarily come from the t[th]iteration of the hubs. This converged classifier, $h_t(x)$ is a *ternary classifier*, a variant of weak classifier proposed by Schapire and Singer [26] which alongside "+1" and "−1" may likewise return "0" as a method for going without replying. It takes a straightforward dominant part vote among the worker's weak classifiers. The ternary classifier will reply "0" if equal number of positive and negative expectations is made by the worker's weak classifiers. Else, it will answer the majority expectation. In line 7, the weights of the comparing classifiers are averaged to get the weight of the ternary classifier. After all the ternary classifiers for T rounds are created, the algorithm gives back their weighted combination as the final classifier.

Algorithm-2: PCM ($\boldsymbol{D^1_n}$, … , $\boldsymbol{D^M_n}$, T)

Input: The training set of M hubs $\boldsymbol{D^1_n}$, … , $\boldsymbol{D^M_n}$ *and* Number of iterations *T*
Output: The final classifier $\{H_{final}(x) = \sum_t \alpha_t h_t(x))$

1.for *p* ← 1 to *M* do
2. Hubs compute classifier H_p *by completing all the T iteration of A*lgorithm_1 ($\boldsymbol{D^p_n}$, *T*)
3. H_p^* ← *Select* weak classifiers of p[th] hub sorted corresponding to $\alpha_{p(t)}$ of t[th] iteration.
4. end for
5. for *t* ← 1 to *T* do
6. Hypothesis *h*(*t*) is formed by converging $\{(h_1(t), ..., h_M(t)\}$
7. $\alpha_t = \frac{1}{M}\sum_{p=1}^{M}\alpha_p\ (t)$
8. end for
9. Output: The final classifier $\{H_{final}(x) = \sum_t \alpha_t h_t(x))$

In an parallel computing environment, where M hubs participates parallelly and the data is distributed uniformly among the hubs, the Computational Complexity of Algorithm-2 is $\Theta(\frac{dn}{M} log\frac{n}{M} + \frac{Tdn}{M})$ which relies on upon the number of iterations T, the number of instances n, the number of attributes D and number of hubs M. The sorting of the *T* weak classifiers (line 3) will have an additional cost of Θ(*T* log *T*) time, which becomes a constant term if T is fixed.

4. EXPERIMENTAL RESULT

In this segment, the proposed algorithms exhibit in terms of certain performance metrics, for example, classification accuracy and speedup. The algorithm outcomes contrasted with big data analytics using MapReduce framework. All the tests were performed on Amazon EC2 distributed computing environment and the computing hubs used were of type M3 instance designed with Latest Intel Xeon Processor and SSD-backed instance storage that conveys higher I/O execution.

The algorithm applied stratified examining on SEER breast cancer dataset keeping in mind the end goal to form training, validation and test segments. The accuracy results are 10-fold cross validation results. In the analyses, the quantity of mappers in the training procedure is dictated by the quantity of splits of the training data. To uniformly appropriate the classes, the training data is split equally among the mappers utilizing the stratification method. The base learning algorithm is utilized as a part of Algoritm-1 is decision trees with standout non-leaf node as weak learners.

For the accuracy experiments, 2,20,811 instances and 17 attributes of the SEER breast cancer data set are utilized. The details of the 17 attributes can be found in Table 1. The error rate of parallel computing utilizing MapReduce (PCM) framework when the number of computing hubs changes from 1 to 20 are shown in Table 2 andgraphical representation in Figure 1. It can be seen from this table contrasted with the one mapper case the PCM calculation has lower or equal error rates.

Table 1: Variables Used For Breast Cancer Recurrance Modeling

Sl. No.	Variable Name
1.	Race
2.	Marital Status
3.	Primary site code
4.	Histological type
5.	Behavior code
6.	Grade
7.	Extension of Tumor
8.	Lymph node involvement
9.	Site specific surgery code
10.	Radiation
11.	Stage of cancer
12.	Age

13.	Tumor size
14.	Number of positive nodes
15.	Number of nodes
16.	Number of primaries
17.	Menopause

Table 2: Pcm Error Rates For Different Number Of Hubs

	Number of Computing Hubs for SEER dataset in PCM				
No. of Hubs	1	5	10	15	20
Error rates	0.1140	0.1102	0.1016	0.0976	0.0860

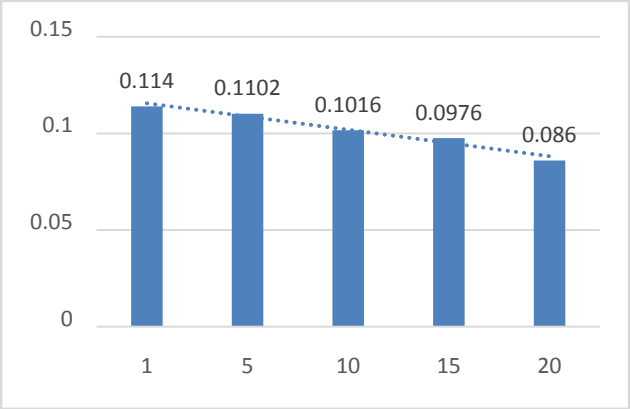

Figure 1: Graphical Representation Of Pcm Error Rates For Different Number Of Hubs

Further, we compared the error rates acquired by theparallel computing utilizing MapReduce with Big data analytics utilizing MapReduce framework algorithm. The correlation results are appeared in Table 3 and graphical representation in Figure 2. It can be seen that the PCM algorithm has the most reduced error rates in SEER breast cancer datasets.

Table 3: Error Rates Comparison Between Pcm And Data Analytics Using Mapreduce Framework Algorithm

BDAM	PCM (10 hubs)	PCM (20 hubs)
0.1146	0.1016	0.086

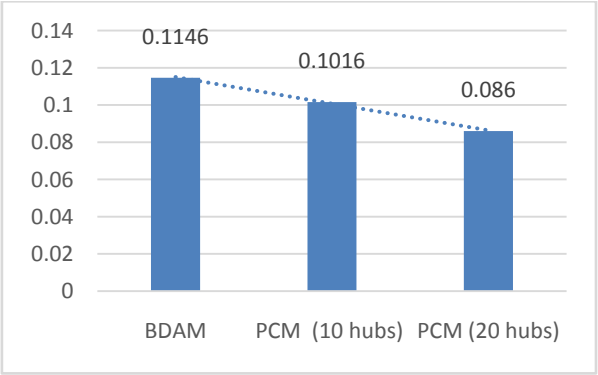

Figure 2: Graphical Representation Of Error Rates Comparison Between Pcm And Data Analytics Using Mapreduce Framework Algorithm

To show the adequacy speedup execution of the PCM with Big data analytics utilizing MapReduce framework algorithm, we ascertain speedup as the proportion of the training time for a single computing node over that of a number of computing hubs handling in parallel (we vary this number from 5, 10, 15 to 20). The itemized aftereffects of speedup are appeared in Table 4 and graphical representation in Figure 3. As can be seen from this table, the number of computing hubs increases, the higher speed up the algorithm accomplishes.

Table 4: Speedup Results For Different Computing Hubs

	Number of Computing Hubs for SEER dataset in PCM			
No. of Hubs	5	10	15	20
Speedup	4.9508	8.4430	11.8850	13.2450

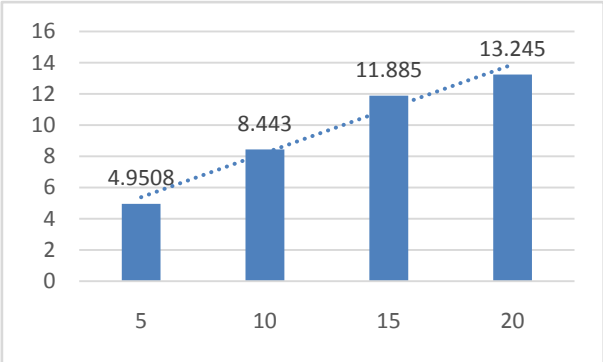

Figure 3: Graphical Representation Of Speedup Results For Different Computing Hubs

5. CONCLUSION AND FUTURE WORK

We proposed our parallel algorithms executed with MapReduce that have fantastic speculation execution. Because of the algorithms' parallel structure, the models can be prompted much quicker. We contrasted the execution of our algorithm with the big data analytics utilizing MapReduce framework as a part of a parallel distributed setting. The tests were performed with the Map-Reduce framework. Our outcomes show that the prediction accuracy of our algorithm is aggressive to the respective baseline and is far and away superior sometimes. We gain significant speedup while building exact models in a parallel domain. The scale up execution of our algorithms demonstrates that they can proficiently use extraresources when the problem size is scaled up.

For the PCM algorithm, since the base learners which handle part of the original datasets work in one single machine successively, in the following step, we plan to parallelize this progression and distribute the computation to additional computing hubs for expanding the computational efficiency. Also, we utilized the same algorithm: BDAM for all the computing hubs in this work. We plan to utilize distinctive algorithms on various computing hubs to build the accuracy further. The reason is that PCM algorithm belongs to the ensemble learning paradigm of machine learning and the more various the base learners are, the higher accuracy could be expected. Further, for the trails, our present adaptation of the algorithms isolates the data using random stratification. We plan to investigate other data partitioning algorithms that can enhance the classification execution significantly further.

6. REFERENCES

[1] Bacardit J, Llorà X (2013) Large-scale data mining using genetics-based machine learning. Wiley Interdiscip Rev Data Min Knowl Disc 3(1):37–61.

[2] Chang EY, Bai H, Zhu K (2009) Parallel algorithms for mining large-scale rich-media data. In: Proceedings of the 17th ACM International Conference on Multimedia. ACM, New York, NY, USA. pp 917–918.

[3] Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. Commun ACM 51(1):107–113.

[4] White T (2012) Hadoop: The Definitive Guide. " O'Reilly Media, Inc.", California.

[5] Venner J, Cyrus S (2009) Pro Hadoop. vol. 1. Springer, New York.

[6] Lam C (2010) Hadoop in Action. Manning Publications Co., New York.

[7] Chu C, Kim SK, Lin YA, Yu Y, Bradski G, Ng AY, Olukotun K (2007) Map-reduce for machine learning on multicore. Advance neural Info processing systems 19:281.

[8] Kearns M (1998) efficient noise-tolerant learning from statistical queries. J ACM (JACM) 45(6):983–1006.

[9] Malewicz G, Austern MH, Bik AJ, Dehnert JC, Horn I, Leiser N, Czajkowski G (2010) Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. ACM, New York, NY, USA. pp 135–146.

[10] Bu Y, Howe B, Balazinska M, Ernst MD (2010) Haloop: Efficient iterative data processing on large clusters. Proc of the VLDB Endowment 3(1-2):285–296.

[11] Ekanayake J, Li H, Zhang B, Gunarathne T, Bae SH, Qiu J, Fox G (2010) Twister: a runtime for iterative mapreduce. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ACM, New York, NY, USA. pp 810–818.

[12] Agarwal A, Chapelle O, Dudík M, Langford J (2014) A reliable effective terascale linear learning system. J Mach Learn Res 15:1111–1133.

[13] Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I (2012) 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, Berkeley, CA, USA. pp 2–2.

[14] Rosen J, Polyzotis N, Borkar V, Bu Y, Carey MJ, Weimer M, Condie T, Ramakrishnan R (2013) Iterative mapreduce for large scale machine learning. arXiv preprint arXiv:1303.3517.

[15] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[16] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," J. Computer and System Science, vol. 55, no. 1, pp. 119-139, 1997.

[17] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," The Annals of Statistics, vol. 38, no. 2, pp. 337-407, 2000.

[18] J. K. Bradley and R. E. Schapire, "Filterboost: Regressionand classification on large datasets," in *NIPS*, 2007.

[19] G. Escudero, L. M`arquez, and G. Rigau, "Boosting applied toe word sense disambiguation," in *ECML*, 2000, pp. 129–141.

[20] R. Busa-Fekete and B. K´egl, "Bandit-aided boosting," in *Proceedings of 2nd NIPS Workshop on Optimization for Machine Learning*, 2009.

[21] G. Wu, H. Li, X. Hu, Y. Bi, J. Zhang, and X. Wu, "Mrec4.5: C4.5 ensemble classification with map-reduce," in *ChinaGrid, Annual Conference*, 2009, pp. 249–255.

[22] B. Panda, J. Herbach, S. Basu, and R. J. Bayardo, "Planet: Massively parallel learning of tree ensembles with mapreduce," *PVLDB*, vol. 2, no. 2, pp. 1426–1437, 2009.

[23] A. Lazarevic and Z. Obradovic, "Boosting algorithms for parallel and distributed learning," *Distributed and Parallel Databases*, vol. 11, no. 2, pp. 203–229, 2002.

[24] W. Fan, S. J. Stolfo, and J. Zhang, "The application of adaboost for distributed, scalable and on-line learning," in *KDD*, 1999, pp. 362–366.

[25] S. Gambs, B. K´egl, and E. A¨ımeur, "Privacy-preserving boosting," *Data Min. Knowl. Discov.*, vol. 14, no. 1, pp. 131–170, 2007.

[26] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

7. AUTHOR PROFILE

Umesh D R completed his Engineering from PES College of Engineering Mandya, Masters from NIE Mysore, presently pursuing Ph.D. from University of Mysore, Mysore. Working in PES College of Engineering Mandya from 2005.

Dr.B.Ramachandra working as Professor and Head in Department of Electrical & Electronics, PES College of Engineering Mandya. He had his Ph.D. From Indian Institute of Science, Bangalore, Master's from Indian Institute of Technology, Bombay.