Induction of decision trees by looking to data sequentially and using error correction rule

NargesSadat Bathaeian
Computer engineering department
Bu-Ali Sina University
Hamedan, I.R. of Iran
bathaeian@basu.ac.ir

Muharram Mansoorizadeh
Computer engineering department
Bu-Ali Sina University
Hamedan, I.R. of Iran
mansoorm@basu.ac.ir

***Abstract*—Decision trees are common algorithms in machine learning. Traditionally, these algorithms make trees recursively and at each step, they inspect data to induce the part of the tree. However decision trees are famous for their instability and high variance in error. In this paper a solution which adds error correction rule to a traditional decision tree algorithm is examined. In fact an algorithm which we call it, ECD3 is introduced. Algorithm of ECD3 inspects data sequentially in an iterative manner and updates tree only when it finds an erroneous observation. This method was first proposed by Dr. Utgoff but not implemented. In this paper, the method is developed and several experiments are performed to evaluate the method. We found that in most cases, performance of ECD3 is comparable to its predecessors. However ECD3 has some benefits over them. First, sizes of its trees are significantly smaller. Second, on average, variance of error in ECD3 is lower. Furthermore, ECD3 automatically chooses part of data for induction of the tree and sets aside others. This capability can be exploited for prototype selection in various learning algorithms. To explain these observations, we use inductive bias and margin definitions in our theories. We introduce a new definition of margin in ordinary decision trees based on shape, size and splitting criteria in trees. We show that how ECD3 expands the margins and enhances precision over test data.**

Keywords— decision tree; sequential reading of data; error correction rule; injecting randomness; margin; entropy; induction bias

I. INTRODUCTION

Algorithms for decision trees, like CART and C4.5 [1], look through the entire data and find the best discriminator attribute. Then they break data based on the selected attribute and repeat that process recursively until reaching a point that the remained data can't be broken. In that situation, they decide based on the majority label of data. These algorithms act greedily to find the smallest tree.

On the other hand, there are some algorithms that don't look through the entire data but read and use them one by one. Algorithms which use error correction rule are in this group. Perceptron is a neural network that uses error correction rule in order to update its weights [2]. It reads individual input and calculates the output for that input. If there is a difference between the desired and real outputs, then it updates the weights. Perceptron acts iteratively. In each round, so called epoch, it looks to same data and decides on each of them individually.

At first glance, the property of decision trees, looking to whole data, is an advantage for them. It is believed that these algorithms exploit the statistical property of all the samples and they are less sensitive to errors in individual training examples [3]. Nevertheless, decision tree can be unstable because small variations in the data might result in a completely different tree being generated [4]. In other words, variance of error is high. Literatures propose different solutions for that problem. For example: Making a forest of trees by, among other procedures, bootstrapping or boosting and then deciding based on voting [5,29] might solve that problem. However, the forest of trees misses the most important advantage of an individual tree which is interpretability [6]. Some literatures consider splitting criteria [7]. Several solutions let the tree grow and then focus on pruning of tree [8]. Others stop the growing of the tree based on statistical measures [9]. Also recent studies show that in the case of number of classes<5, balanced data are more prone to instability of decision trees [10].

However as a new solution for that problem, this paper focuses on the method of reading and choosing data. We aim to find out the difference between a decision tree algorithm that looks through entire data to build the best tree and an iterative algorithm which looks to data one by one and improves its representation when an error occurs. We show that the new algorithm builds trees that function at least as well as the traditional algorithms. Nevertheless, it has some preferences over them. The experiments show that when the size of decision tree is a criterion or when attributes are numerical; it performs significantly better. It has less variance of errors, so it

978-1-5090-4335-4/16/$31.00 ©2016 IEEE

can be a solution for instability of trees. In addition, it can choose essence of dataset. That is an opportunity for some algorithms like k-nearest-neighbor.
Rest of the paper is organized as follows: the following section describes the proposed algorithm which we call it ECD3. Section three shows experimental results. Section four concludes the paper and gives some suggestions for improvement of implementation of the algorithm.

II. ECD3: Error Corrector Decision Tree

A. Brief Description and Algorithm

In order to briefly describe ECD3, one can say it assumes a decision tree based on entropy. Then it looks to each sample individually and decides whether to update the decision tree or not. It repeats that task until it reaches a consistent tree. To better explain ECD3, we use notations used in describing Perceptron:

Perceptron is a neural network's algorithm that incorporates error correction rule to find a discriminator line. Eventually, after enough repeats, it reaches to desired discriminator. Error correction rule can be described as (1):

$$\begin{cases} W_{new} = W_{old} & \text{no error} \\ W_{new} = W_{old} + \eta * error * input & \text{error} \end{cases} \quad (1)$$

Here we use almost the same rule to induce ECD3. Normally, a decision tree is made by looking among the whole data, but in ECD3, the algorithm looks to each individual input and based on the output of the current tree, decides to whether update the tree or not. Equation 2 shows the use of error correction rule in ECD3:

$$\begin{cases} D_{new} = D_{old} \ , T_{new} = T_{old} & \text{no error} \\ D_{new} = D_{old} \cup \{input\} \ , T_{new} = Tree(D_{old}) & \text{error} \end{cases} \quad (2)$$

Formula 2 expresses that if no error occurs then the dataset for inducing the tree remains unchanged and so the tree will remain unchanged too. However if by looking to a sample, tree couldn't recognize the true class for that, then dataset along with tree would be updated. Fig.1 shows the algorithm of ECD3. Presentation of the algorithm is based on the algorithmic framework which was described in [11].

```
Initialize flag to false
Set D1 as the input training dataset
Initialize datasets D2 and D3 to empty
Initialize decision tree T to empty
While (flag)
    flag ← false
    For each xi in D1
        If (T doesn't classify xi correctly)
            Add xi to D2
            T= TreeGrowing(D2, A, y)
            flag ← true
        Else
            Add xi to D3
    Clear D1
    Copy D3 to D1
    Clear D3
```

Figure 1. Algorithm of ECD3. Procedure *TreeGrowing*(*S; A; y*) makes the decision tree. The inputs are *S* (Training Set), *A* (Input Feature Set) and *y* (Target Feature).

Both Perceptron and ECD3 act iteratively to reach the target, but unlike Perceptron which investigates all data in each epoch; ECD3 uses data that have not produced errors in previous epoch. In other words, it investigates data which are not contributed in building the current tree. Therefore, there are three sets of data in ECD3: D_1, D_2 and D_3. Initially, D_1 is set to the whole input data. In each epoch, ECD3 looks to it and picks an input for evaluation. D_2 is the set of data that has produced errors. So if an input is incorrectly classified by current tree, it would be added to D_2 and the tree will be rebuilt. In contrast, if the current tree correctly classifies an input, that input would be added to D_3. At the end of each epoch, D_1 is replaced by D_3 and above process is repeated.

There is an issue must be considered here. There is a big difference between ECD3 and some well-known trees so called incremental trees. Incremental decision tree learning may be seen as some variation of ECD3. These algorithms assume data come in a stream [12]. However, there is no repetition in those algorithms and each sample would be processed once in a time.
Utgoff raised an idea of training in error correction mode with the pool regimen [13] which would be a work that can be compared to ECD3. However his idea has not implemented yet. So he didn't present the detail description and experimentations of that algorithm.

B. An Illustrative Example

In this example, table 1 gives a set of observed data and Fig.2 depicts the steps of ECD3 algorithm.

Continuing to the third epoch isn't necessary. Since the tree obtained from second epoch produces right answer for input 4. As the result, tree is based on the inputs 1, 2 and 3

TABLE 1. A CONTINGENCY TABLE WHICH SHOWS OBSERVED DATA

	A	B	C	Class
1	0	1	0	T
2	1	0	1	T
3	0	0	1	F
4	0	0	0	F

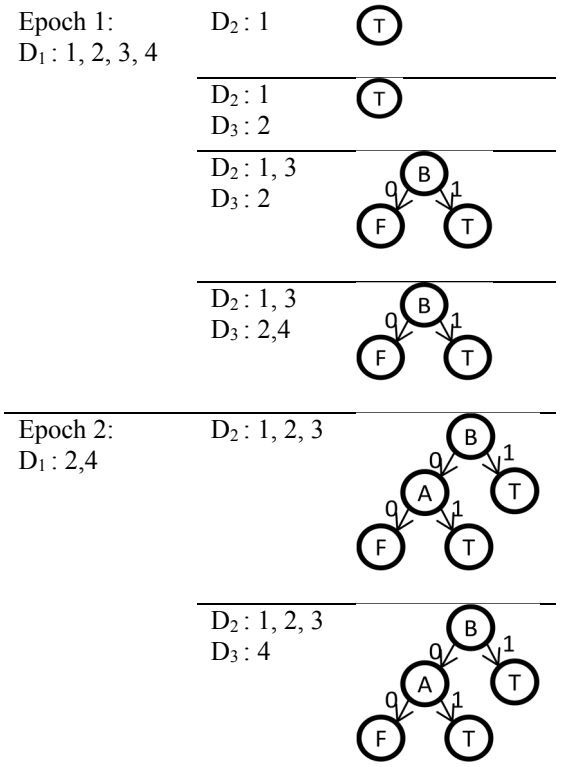

Figure 2. Steps of ECD3 algorithm for data set of table I

III. EMPIRICAL BEHAVIORE OF ECD3

Back to the earlier section, error correction which was proposed by Utgoff by some means has to do with ECD3. Utgoff [13] saw some preferences in error correction mode of inducing decision trees. He stated that error correction mode produces a tree which is often smaller and more accurate than a tree based on all the instances presented. He added that the reason for this phenomenon is still unknown [13]. Here, we want to report empirical behavior of the ECD3 more precisely. Three issues will be addressed here:

- First, sizes of the trees in both algorithms will be compared.
- Second, performance of ECD3 will be compared with its predecessors. It is done by calculating average and variance of errors at the test phase.
- Third, performance of ECD3 as a prototype selection method will be investigated.

In order to carry out those experiments, we developed a java programmed tool named *DecisionTrees*. Two algorithms; ID3 and ECD3; are implemented in the tool. ID3 is implemented as presented in Tom Mitchell's text book [3]. In the case of numeric attributes, we followed the C4.5 algorithm [11]. In other words, we used binary splits approach for numeric attributes. Missing attributes are considered as ignorance ones. In our implementation, ECD3 uses that algorithm for updating the tree too. *DecisionTrees* is available at http://profs.basu.ac.ir/bathaeian/index.php?L=free

We used 23 data sets, from UCI repository [14]. First column in table 2 shows the list of them.

TABLE 2. EXPERIMENTAL RESULTS. THE FIRST COLUMN SHOWS THE NAME OF DATASET. THE PLUS SIGNS IN COLUMN 2 AND 3 INDICATE THAT AVERAGE AND VARIANCE OF ERROR IN ECD3 ARE LESS. THE PLUS SIGNS IN FOURTH COLUMN SHOW THAT SIZE OF ECD3 IS SMALLER. THE LAST COLUMN SHOWS THAT WHAT PERCENT OF DATA IS USED FOR INDUCTION OF ECD3.

Data set	**Ave. error**	**Var. error**	**H_0 (tree size)**	**Sample used (%)**
audiology			+	53.43
breast_cancer			+	63.99
credit_a	-	+		55.39
diabetes	+		+	69.49
glass	+			64.97
heart_c			+	62.93
heart_h	+	+	+	51.66
heart_stat			+	65.02
hepatites	+	+		56.14
iris	+	+		22.22
kr_vs_kp		+	+	5.55
labor			+	51.35
led24_2000_10			+	71.90
lymph	-	+	+	60.30
primary_tumor	-	+	+	81.63
segment	+		+	21.78
sick_noTBG	+	+	+	8.06
solar		+	+	76.28
sonar		+		74.41
Tic-tac-toe	+	+	+	55.96
vehicle		+	+	65.67
vot	+	+	+	39.18
wine	+	+	+	25.34

A. Method

We used k-fold cross validation to get k opportunities for doing experiments. This approach is very common and we implemented it in our tool. Therefore we describe it briefly here. In that approach, first we partition the training set into k segments. Second we set aside a segment and build classifier based on the remaining ones. Then repeatedly, we replace the removed segment with another one and construct a new classifier. By this manner, we would have k classifiers built on slightly different training sets. In our experiments, we assumed 10 for k. So for each data set 10 times algorithms ID3 and ECD3 are run. On each run, we may use the remaining segment as the test set and figure out the differences between errors of both classifiers. Average and variances of

these outputs are feed to appropriate hypothesis tests to prove our claims. We also performed similar calculation to figure out the differences between sizes of constructed trees in both algorithms.

B. Comparision of Tree Sizes

Number of nodes in a decision tree can be accounted for its size. We claim that in the 95% confidence interval, average tree-size of ECD3 (s_2) is smaller than or equal to average tree-size of ID3 (s_1). So the H_0 hypothesis would be (3).

$$H_0: s_2 \leq s_1 \quad (3)$$

And alternative hypothesis would be (4).

$$H_1: s_2 > s_1 \quad (4)$$

We performed one-sided **t** test to prove our claim. We found H_0 is true for all data sets. In addition, for 18 of 23 data sets, tree-sizes in ECD3 are significantly smaller than tree-sizes in ID3. In table 2, these data sets are shown by plus sign in the fourth column.

C. Comparision of Errors

We claim that in the 95% confidence interval, average errors of ECD3 (e_2) is smaller than or equal to average errors of ID3 (e_1). So the H_0 hypothesis would be (5). Alternative hypothesis would be (6).

$$H_0: e_2 \leq e_1 \quad (5)$$

$$H_1: e_2 > e_1 \quad (6)$$

We performed one-sided **t** test to prove our claim. We found that for 20 of 23 data sets, H_0 is accepted. In 10 datasets, ECD3 performs significantly better than ID3. In table 1, these data sets are shown by plus sign in the second column. In only three of 23 datasets H_1 is accepted. These data sets are shown by minus sign in the third column of table 1.

We also claim that in the 95% confidence interval, variance of errors in ECD3 (v_2) is smaller than or equal to variance of errors in ID3 (v_1). So the H_0 hypothesis would be (7). Alternative hypothesis would be (8).

$$H_0: v_2 \leq v_1 \quad (7)$$

$$H_1: v_2 > v_1 \quad (8)$$

We performed one-sided **t** test to prove our claim. We found that for all data sets, H_0 is accepted. In 14 datasets, ECD3 performs significantly better than ID3. In table 1, these data sets are shown by plus sign in the third column.

So we can conclude that ECD3 performs significantly better than ID3 and variance of errors in ECD3 is significantly lower.

D. ECD3 as a Method for Prototype Selection

Prototype selection can be used in several applications such as compressing [15] or preprocessing of datasets. For example, it can be adopted in the preprocessing phase of K-Nearest Neighbors (KNN) which is one of the well-known classifiers [16]. KNN belongs to the family of lazy learners. Lazy learners first store training samples. Then they do the most of the task and computation when they are presented with a test sample. This method suffers from several drawbacks. Need for big storages to save patterns as well as huge computation in test or apply phase are parts of its weaknesses. To overcome those difficulties, some preprocessing algorithms are used. In fact, KNN isn't a completely lazy method! [17]. For example, finding the optimum K and selecting a prototype of the data set are types of solutions. Prototype selection means choosing subset of training samples instead of all of them [15, 16]. This subset must be a condensation of the whole original data set. The condensed data set of course must not decrease the generalization accuracy of classifier. As the second example, prototype selection is used in down-sampling of imbalanced data sets. Reference [13] introduced an algorithm for selection of a prototype of majority class.

Inspecting the algorithm of ECD3 carefully, we understand that generated tree is consistent with whole data but the samples used for construction of the tree make a subset of primary dataset. Assume D_1 as the input dataset and D_2 as a subset of D_1 which actually contributes in building the tree. If we take members of D_2 as random variables, then we get (9):

$$D_2 = \underset{\text{size of } D_{train}}{\arg\min} \; p\left(\frac{D_{train}}{D_1}\right), \quad \text{Tree}(D_2) \text{ is consistent with } D_1 \quad (9)$$

Above formula shows that ECD3 chooses subset of training data which has minimum size and creates the same classifier as one created by the whole data. Therefore D_2 is a prototype selected by ECD3. Column five of table 2 shows the average percentage of samples used in ECD3 learning. The percentage of reduction varies from 20% to 95%. On average ECD3 reduces number of samples to 55%.

E. Discussion

Just like ID3, ECD3 is a consistent algorithm. In fact, at the worst case, all samples are incorporated to induce the tree. So, considering the fact that both ECD3 and ID3 make decision trees as well as the fact that ECD3 uses ID3 algorithm to induce the tree, equivalence of these algorithms isn't surprising.

Nevertheless, there are some differences between the trees obtained by these algorithms. We believe that two reasons are accounting for those differences: inductive bias and chance of creating different margin.

Normally, inductive bias of ID3 is related to its search strategy [3]. It searches incompletely through

the hypothesis space, from simple decision trees to complex ones, until it finds a decision tree consistent with the data. ECD3 follows a similar approach except that it combines the strategy with gradient descent search approach. As the result, ECD3 can build a different and in most cases smaller decision tree than ID3.

Further, we see some differences between errors of both trees. Margin might explain those variations. In definition, margin is a quantity to measure the confidence [19]. Larger margins create much more reliance on the classifier. So margins have specific contribution in generalization errors. In the case of decision trees, there is no formal definition for margin. Some researches try to inject directly margin concept into the decision tree [20 and 21]. However, decision trees in our research are based on the traditional definition of ID3 and we can't use direct definition of margin in this area. Apparently, error bias and variance account for confidence of a learning algorithm. In addition, we know that decision trees with different sizes and nodes have different error bias and variance [22]. We showed that in most cases tree-size of ECD3 is shorter than tree-size of ID3. This can be meant that trees induced by ECD3 have different error variances and consequently different confidences.

Other explanation for this phenomenon is possible too. One can define margin as dissimilarities between rules generated by decision trees. For example in Fig.3 two decision trees and generated rules are shown.

In Fig.3-a, h1 and h3 have two distinctions but h2 and h3 have one of it. Provided that minimum margin is 2 then we can conclude that one irregularity is happened. However in Fig.3-b, two of these irregularities are seen: one between h2 and h3, other one between h1 and h4. One conclusion might be that more unbalanced trees produce fewer irregularities on the presumed margin. All trees produced by ECD3 have fewer nodes than those generated by ID3 but the height of both trees are equal. Therefore we can conclude that ECD3 generate more unbalanced trees compare to ID3.

In the case of numeric attributes, when we use binary splits, split position depends on the selected adjacent points belonging to different classes. In ID3, these adjacent points are constant in a data set but in ECD3, the inner loop of the algorithm makes alternatives of these points. For example in Fig.4, we have three points belonged to two different classes (circles and multiplication sign).

In this situation, we can use alternative definition of margin introduced in support vector networks [23]. According to that definition, margin is the minimum distance between training samples of two classes. In order to construct an optimal separator, one only has to take into account a small amount of the training data, so called support vectors, which determine this margin. C4.5 definitely chooses points 1 and 2 as adjacent points (support vectors) and then outputs line A as separator threshold. Here, margin of two classes would be the distance between points 1 and 2. On the other hand, ECD3 acts randomly for choosing the samples. So, there is a probability of 0.5 that it chooses point 3 instead of point 2 and consequently it reaches to line B as a splitting line. Because of larger distance between point 1 and 3, we would have greater margin than previous one. Nevertheless, both separators correctly distinguish between two classes in the phase of training. We expect that classifier with larger margin perform better in the testing phase.

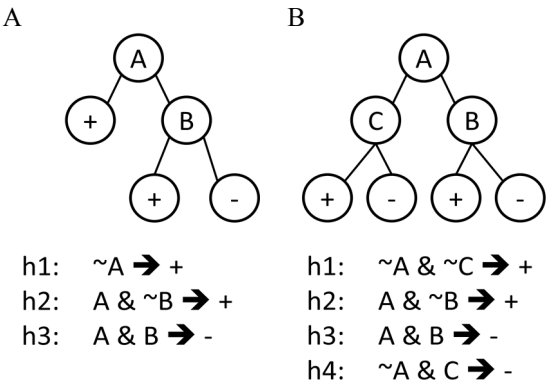

Figure 3. Generated rules of two decision trees

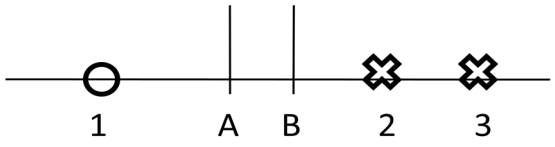

Figure 4. Two classes of data and their separators obtained from two algorithms (A for algorithm C4.5 and B for algorithm ECD3)

That fact can be clearly seen in our experiments. ECD3 induced larger tress for datasets of Iris, Glass and Hepatitis but surprisingly their performance in term of generalization error is much better than ones produced by ID3. It's because all attributes of those datasets are numeric.

IV. Conclusions and Future Works

In this paper a new algorithm, ECD3, was introduced for decision tree induction. ECD3 uses ID3 or C4.5 as its base classifier but it uses data differently for inducing the tree. In our experiments, in most cases, ECD3 produces smaller trees compared to ID3 and in some cases it performs better than ID3. Distinctions between inductive biases and margins somehow can explain the differences. Other explanations are possible too. For example we didn't consider the randomization property of ECD3 and its ability to choose variant support vectors. These are phenomenon might account for differences.

Several applications of ECD3 algorithm are possible. For example, an issue that was not addressed here is that some machine learning algorithms benefit from randomness in classifier. Specifically, those classifiers used in ensemble learning [24] would be better if they have diverse errors [25]. Due to sequentially entrance of data to ECD3, it may be dependent on the place of individual samples in the queue. By running one round of algorithm, we get a tree which is consistent with part of the data. This task can be done repeatedly to make variant trees; each one is consistent with different part of the data.

Although in this paper, ECD3 is introduced as a method for prototype selection, more studies are required to measure its performance comparing with other prototyping techniques.

We defined margin in decision trees based on the shape and size of tree. However considering [26 and 27] which introduce growing of decision tree as a boosting method; other definitions and more specific mathematical proofs might be possible too. That is the area which has more works to do.

ECD3 is slower than ID3. In our implementation, in each inner loop, if the required condition was hold, ECD3 is built from scratch. Algorithm of ITI is a state of the art in incremental learning of decision trees [13]. However more recent and optimal algorithms are presented that can be used in some way. For example [28] introduces an algorithm for efficient representation of incoming data.

REFERENCES

[1] S. B. Kotsiantis, (2013) Decision trees: a recent overview, Artif Intell Rev, 39:261–283, Springer.

[2] Alpaydin, E. (2004) Introduction to Machine Learning. Massachusetts Institute of Technology.

[3] Mitchell, T. (1997) Machine Learning. McGraw-Hill.

[4] Li RH, Belford GG (2002) Instability of decision tree classification algorithms. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, pp 570–575. ACM, New York, NY, USA.

[5] Lior Rokach , (2016) Decision forest: Twenty years of research, Information Fusion, Volume 27, Pages 111–125, Elsevier B.V

[6] Zurada, J., (2010) Could Decision Trees Improve the Classification Accuracy and Interpretability of Loan Granting Decisions?, 43rd Hawaii International Conference on System Sciences (HICSS), pp.1-9.

[7] Kweku-Muata Osei-Bryson, Kendall Giles, (2006) Splitting methods for decision tree induction: An exploration of the relative performance of two entropy-based families, Information Systems Frontiers, July, Volume 8, Issue 3, pp 195-209.

[8] Abdulaziz Alkhalid, Igor Chikalov, and Mikhail Moshkov, (2013) Comparison of Greedy Algorithms for Decision Tree Optimization, A. Skowron and Z. Suraj (Eds.): Rough Sets and Intelligent Systems, ISRL 43, pp. 21–40, Springer-Verlag Berlin Heidelberg

[9] Tomas Aluja-Banetl and Eduard Nafria2, (2003) Stability and scalability in decision trees, Computational Statistics , 18:505-520, Springer.

[10] [4-5] José Augusto Baranauskas , (2015) The number of classes as a source for instability of decision tree algorithms in high dimensional datasets, , Artif Intell Rev, 43:301–310, Springer.

[11] Rokach, L.; Maimon, O., (2005) Top-down induction of decision trees classifiers – a survey," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.35, no.4, pp.476-487.

[12] Gama, J., Fernandes, R., & Rocha, R. (2006) Decision trees for mining data streams. Intelligent Data Analysis, 10, pp. 23-45, IOS Press

[13] Utgoff, P.E. (1994) An improved algorithm for incremental induction of decision trees. Proceedings of the 11th international conference on machine learning, pp 318–325.

[14] Bache, K. and Lichman, M. (2013) UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[15] Schiilkop, B., Burgest, P., and Vapnik, V. (1995) Extracting support data for a given task, Proceeding of Knowledge Discovery and Data Mining, pp 252-257, AAAI Press.

[16] Garcia, S., Derrac, J., Cano, J., and Herrera, F. (2012) Prototype selection for nearest neighbor classification: taxonomy and empirical study. IEEE Trans. On Pattern Analysis And Machine Intelligence, 34, no. 3, 417-435.

[17] Garcia, S., Feldman, M.R., Gupta, and Srivastava, S. (2010) Completely lazy learning. IEEE Trans. Knowledge and Data Eng., 22, no. 9, 1274-1285.

[18] Han, F., Lei, M., Zhao, W., and Yang, J. (2012) A new support vector machine for imbalance data classification. Intelligent automation and soft computing, 18, no. 6, 679-686, Taylor & Francis.

[19] Schapire, R.E., Freund, Y., Bartlett, P., and Lee, W.S. (1997) Boosting the margin: a new explanation for the effectiveness of voting methods. Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc.

[20] Yıldız, O.T. (2012) Univariate decision tree Induction using maximum margin classification. The Computer Journal, 55, issue 3, 293-298, Oxford Press.

[21] Tibshirani, R. (2007) Margin trees for high-dimensional classification. Journal of Machine Learning Research, 8, 637-652, JMLR.org.

[22] Domingos, P. (2000) A unifeid bias-variance decomposition and its applications. Proceedings of the Seventeenth International Conference on Machine Learning, pp.231-238, Morgan Kaufmann Publishers Inc.

[23] Cortes, C. and Vapnik,V. (1995) Support-vector networks. Machine Learning, 20, no. 3, 273-297.

[24] Banfield, R.E., Hall, L.O., Bowyer, K.W., and Kegelmeyer, W.P. (2007) A comparison of decision tree ensemble creation techniques. IEEE Trans. On Pattern Analysis And Machine Intelligence, 29, no. 1.

[25] Kuncheva, L. and Whitaker, C. (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning, 51, no. 2, 181 -207, Springer.

[26] Kearns, M. and Mansour, Y. (1999) On the boosting ability of top_down decision tree learning algorithms. Journal of Computer and System Sciences, 58, 109-128, Elsevier BV.

[27] Takimoto, E. and Maruoka, A. (2003) Top-down decision tree learning as information based boosting. Theoretical Computer Science, 292, 447–464, Elsevier BV.

[28] Swere, E., Mulvaney, D., and Sillitoe, I. (2006) A fast memory-efficient incremental decision tree algorithm in its application to mobile robot navigation," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 645-650.

[29] Breiman, L., 2001. Random forest. Mach. Learn. 45, 5–32