A Comparative Study on Serial Decision Tree Classification Algorithms in Text Mining

Khaled M. Almunirawi, Ashraf Y. A. Maghari
Islamic University of Gaza, Gaza, Palestine

Abstract

Text mining refers to the process of deriving high quality information from text. It is used in search engine, digital libraries, fraud detection, and other applications that handles text data. Text mining tasks include text classification, text clustering, entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling.

Classification of objects into pre-defined categories based on their features is a widely studied problem. It aims to employ labeled training data set to build a classification model based on other attributes, such that the model can be used to classify new data according to their class labels.

The decision tree-based classification is one of the most practical and effective methods that uses inductive learning. It is implemented serially or in parallel, depending on data set size. Some of the classifiers such as SLIQ, SPRINT and Rainforest can be implemented serially or parallel. ID 3, CART and C4.5 are serial classifiers.

In this paper, we review various decision tree algorithms with their limitations, and conduct a comparative study to evaluate their performance regarding accuracy, learning time and tree size, using four sample datasets. We found out that Random Forest classifier is the most accurate one among other classifiers. However, the increase of the dataset size and its attributes, the more the learning time and tree size, and vise versa.

1. Introduction

Text classification has recently gained a prominent status in computer field; because the emerging number of documents is countless. All these documents need to be categorized in order to make the searching for those documents easy.

Text Classification is the task of classifying or assigning a document to a predefined category. For example, if d is a document of the entire set of documents D and c_1, c_2, c_3 …c_n are the set of all the categories, then text classification assigns one category c to a document d. In detail, it is all about detecting the type of the unlabeled document, which is passed as an input to the system. This document has no label, and expects the system to give a type to it. So the system scans that document and sends it to preprocessing unit. The document can be collected in various formats. All these documents could be fetched from different sources.

System is already trained for categorizing the document using some keywords. E.g., a system has some pre-defined categories, politics, movies, computers, etc. Each of these categories has decided their own keywords like for the type or category movies, names of actors, box office are all keywords. Similarly, each category has their specific keywords. With the help of these keywords, the system detects the type of the unlabeled document. If the inputted document has the keyword of politics type then the label of that document is decided as politics. After deciding the type, the label of the document is set and hence the document is classified. The system also learns the newly added document and trains itself to pick out some more keywords, and hence improving the efficiency of the system. Some technical terms in this paper include:

Entropy is a measure of the number of random ways in which a system may be arranged. For a data set S containing n records, the information entropy is defined as:

$$Entropy(S) = -\sum P_I \log_2 P_I$$

(P_I is the proportion of S belonging to class I)

Gain or the expected information gain is the expected reduction in information entropy caused by partitioning the examples according to a given attribute. The information gain of example set S on attribute A is defined as:

where *Value*(*A*) is the set of all possible values of attribute A, *Sv* is the subset of S for which attribute A

$$Gain(S,A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

has value v, | *Sv* | is the number of elements in *Sv* and |*S*| is the number of elements in S.

Gini index for a data set S is defined as:

$$gini(S) = 1 - \sum P_I^{\ 2}$$

and for a 2-split:

$$gini_{split}(S) = \frac{n_1}{n} gini(S_1) + \frac{n_2}{n} gini(S_2)$$

and so on for a k-split.

Recent study [1] pay attention at serial implementation of decision tree algorithms. Compared to parallel implementation of decision which is complex to implement, the serial implementation is memory resident, fast and easy to implement. The performance of the decision tree algorithms has been experimentally evaluated and analyzed using Statlog data sets. The study was limited to ID3, C4.5, CART and SPRINT algorithms, and showed that the execution time in building the tree model depends on the volume of data records. While, there is an indirect relationship between execution time in building the model and attribute size of the data sets.

Other most recent comparison study in educational field [2] applied three algorithms (ID3, C4.5, and CART) to predict student's performance in examination. Accuracy and time taken to build the tree were the main attributes to analyze the algorithms. The study observed that C4.5 was the best algorithm for small datasets with better accuracy and efficiency than the other algorithms, with disadvantage when the training data is large. The study solved this problem using SPRINT and SLIQ decision tree algorithm, and suggested that there is a need to develop effective algorithms for decision tree.

Both previous studies did not mention other important sequential decision tree algorithms such as Best First Tree and Random Forest.

In this paper, a comparative study between the classifiers is conducted, including previously studied algorithms, in addition to other algorithms, namely Best First and Random Forest algorithms.

The rest of this paper is organized as follows: section 2 reviews background of classification algorithms with three subsections about text classification process 2.1, text classification algorithms 2.2, and decision tree algorithms 2.3, section 3 explains experiment details, section 4 views results, and associated discusses, and ends with conclusion in section 5.

2. Background

The main tasks of text mining include text classification, text clustering, entity extraction and the production of granular taxonomies among other tasks.

2.1. Text Classification Process

The stages of text classification process are shown in **Figure 1. Document Classification Process** explained as follows:

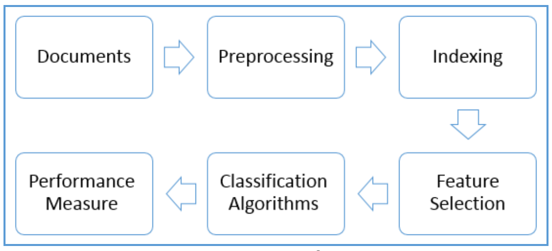

Figure 1. Document Classification Process

2.1.1. Document Collection

In this first stage of classification process, we collect the different types (format) of document like html, .pdf, .doc, web content etc.

2.1.2. Pre-Processing

In this stage, the text documents are represented in clear word format. The documents are prepared for the next step in text classification whereas the documents are represented by a great amount of features. Commonly the steps taken are:

a. Tokenization: A document is treated as a string, and then partitioned into a list of tokens.
b. Removing stop words: Stop words such as "the", "a", "and", etc. are frequently occurring, so the insignificant words need to be removed.
c. Stemming word: Converting the word form into similar canonical form by applying stemming algorithm.

2.1.3. Indexing

The documents representation is one of the pre-processing techniques that is used to reduce the complexity of the documents, and make them easier to handle. In this stage, the document is transformed from the full text version to a document vector, where the documents are represented by vectors of words.

2.1.4. Feature Selection.

It is the most important step after pre-processing and indexing, where a subset of features is selected from the original documents. A vector space is constructed to improve the scalability, efficiency and accuracy of a text classifier.

2.1.5. Classification

The documents can be automatically classified by three ways, unsupervised, supervised and semi-supervised methods. In the last few years, the task of automatic text classification have been extensively studied and rapid progress seems in this area, including the machine learning approaches such as

Bayesian classifier, Decision Tree, K-nearest neighbor (KNN), Support Vector Machines(SVMs), Neural Networks, and others.

2.1.6. Performance Evaluations

The text classifiers are evaluated experimentally rather than analytically. Usually, the experimental evaluation of classifiers concentrates on the evaluating the effectiveness of a classifier, i.e. its capability of taking the right categorization decisions, rather than concentrating on issues of Efficiency. One of the most important issues of Text categorization is how to measures the performance of the classifiers. Many measures like Precision and recall, and accuracy are usually used to evaluate the classifiers [3].

2.2. Text Classification Algorithms

Text Classification algorithms are categorized as follows [4]:

1. Classification using Decision Tree:
a. Serial Decision: A decision tree model consists of internal nodes and leaves. Each of the internal nodes has a decision associated with it and each of the leaves has a class label attached to it, (also called sequential).
b. Parallel Formulation: which uses three types (synchronous, partitioned, and hybrid parallel formulation) for constructing trees.
2. Classification using Neural Network
3. Classification using Genetic Algorithm, including Genetic Operators and End Condition.

2.3. Decision Tree Algorithms.

Decision tree is a popular approach for representing classifiers. It is considered one of the most popular data-mining techniques for knowledge discovery. Decision tree can systematically extract valuable rules and relationships from information contained in a large data source. These extracted rules are usually used for the purpose of classification/prediction. Compared to other text mining techniques, it is widely applied in various areas since it is robust to data scales or distributions [5].

Decision tree algorithm partitions a data set of records - recursively - using depth-first greedy approach [6] or breadth-first approach, until all the data items belong to a particular class are identified.
Decision tree algorithms are implementable in both serial and parallel form. Parallel implementation of decision tree algorithms is desirable in order to ensure fast generation of results especially with the classification/prediction of large data sets, it is also possible to exploit the underlying computer architecture. However, when small-medium data sets are involved, serial implementation of decision algorithms is desirable and easier to implement. Some algorithms are explained in the next sub-sections.

2.3.1. ID3

Iterative Dichotomized algorithm (ID3) basically built on the Concept Learning System (CLS) algorithm, the basic algorithm for decision tree learning. ID3 initially designed to improve CLS by adding a heuristic for attribute selection. ID3 is based on Hunt’s algorithm [7] and is implemented serially [1]. This algorithm recursively partitions the training dataset, using depth first greedy technique, until the record sets belong to the class label. In growth phase of the tree construction, this algorithm uses information gain, an entropy based measure, to select the best splitting attribute, and selects the attribute with the highest information gain as the splitting attribute. If the training set has a lot of noise or details, ID3 does not give accurate result. Previous experiments included serious pre-processing steps for the data before building a decision tree model with ID3 [1]. The main weakness point of ID3 is that the measure gain used be likely to to favor attributes with a large number of distinct values [8]. Besides, it accepts categorical attributes only when start building the tree model. This decision tree algorithm generates variable branches per node.

2.3.2. C4.5

It is an enhanced version of ID3, it uses Gain Ratio as a splitting criteria, instead of taking gain in ID3 [9] in tree growth phase. Hence C4.5 is an evolution of ID3 [1]. This algorithm handles both continuous and discrete attributes- In order to handle continuous attributes, C4.5 creates a threshold and then splits the list into two categories: one for the group with the value of the attribute is above the threshold, and the second for the group with the value of the attribute is less than or equal to it [10]. Similar to ID3, to determine the best splitting attribute, the data is sorted at every node of the tree.

The splitting ends when the number of instances to be split is below a predefined threshold. The main advantages of C4.5 is in the phase of building a decision tree, deal with the case of attributes with continuous domains by discretization, also it can deal with datasets that have patterns with unknown attribute values. C4.5 can deal with training data with attribute values by allowing attribute values to be marked as missing. Missing attribute values are simply not used in gain and entropy calculations. It has an enhanced method of tree pruning, which reduces misclassification errors results from noise or too much detail in the training data set.

2.3.3. CART

Classification and Regression Trees (CART) [11], is branded by the fact that it constructs binary trees, which means that each internal node has exactly two outgoing edges, unlike ID3, C4.5 algorithms, which generate the decision trees with variable branches per node. Unlike other Hunt's algorithms, CART can also be used for regression analysis with the help of regression trees. Given a set of predictor variables over a given period of time, the regression analysis feature is used in predicting a dependent variable (result). The CART decision tree is a binary recursive partitioning procedure, capable of processing continuous and nominal attributes both as targets and predictors. CART uses gini index for splitting procedure to a maximum size without using stopping rule, and then pruned back (mainly split by split) to the root via cost-complexity pruning. The CART mechanism is intended to produce not only one, but also a sequence of nested pruned trees, all of which are candidate optimal trees. The CART mechanism allows for cost-sensitive learning, dynamic feature construction, and probability tree estimation [12]. It contains automatic missing value handling, and automatic (optional) class balancing.

2.3.4. Best First Tree (BF Tree)

A standard algorithm for the top-down generation of decision trees expand nodes in depth-first order in each step using the divide-and-conquer strategy, such as ID3, C4.5 and CART. In Best-first decision trees, the selection of best split is based on boosting algorithms [13] which is used to expand nodes in best-first order instead of a fixed order. BF Tree algorithm uses both the gain and gini index in calculating the best node in the phase of tree grown of the decision tree so that the "best" split node is added to the tree in each step. The best node is the one that maximally reduces impurity among all nodes available for splitting (i. e. not labeled as terminal nodes). Although this results in the same fully-grown tree as standard depth-first expansion, it enables us to investigate new tree pruning methods that use cross-validation to select the number of expansions. Both pre-pruning and post-pruning can be performed in this way.

2.3.5. SLIQ

Supervised Learning In Quest (SLIQ) [14] was one of the first scalable algorithms for decision tree induction. It can be implemented in serial and parallel way. It partitions a training data set recursively using breadth-first greedy strategy that is integrated with pre-sorting technique during the tree building phase. SLIQ uses a vertical data format, meaning all values of an attribute were stored as a list, which was sorted at the start of the algorithm. This means that the attributes need not be sorted repeatedly at each node as the case was in existing algorithms like CART and C4.5. With the pre-sorting technique, sorting at decision tree nodes is eliminated and replaced with one-time sort with the use of list data structure for each attribute to determine the best split point. The calculation of gini index for each possible split point can be done efficiently by storing class distributions in histograms, one per class per node. However, SLIQ uses a memory-resident data structure called class list which stores the class labels of each record. This data structure limits the size of the datasets SLIQ can handle [15].

Numeric and categorical attributes both are handled in building a decision tree model of the SLIQ. One of the main drawbacks of SLIQ is that it uses a class list data structure that is memory resident, thereby imposing memory restrictions on the data. It uses Minimum Description length Principle(MDL) in pruning the tree after constructing it MDL is an inexpensive technique in tree pruning that uses the least amount of coding in producing tree that are small in size using bottom-up technique [1], [16].

The main advantage of SLIQ decision tree algorithm is that it produces accurate decision trees that are significantly smaller than the trees produced using C4.5 and CART. At the same time, SLIQ executes nearly an order of magnitude faster than CART [14].

2.3.6. SPRINT

Scalable Parallelizable Induction of decision Tree algorithm (SPRINT) present a more memory efficient version of SLIQ [17]. It associates the class label along with the record identifier for each value in the attribute lists. It is a fast, scalable decision tree classifier. It is not based on Hunt's algorithm in constructing the decision tree, rather it partitions the training data set recursively using breadth-first greedy technique until each partition belong to the same leaf node or class [1], [18]. SPRINT is considered as an enhancement of SLIQ as it can be implemented in both serial and parallel pattern for good data placement and load balancing [17]. In this paper we will focus on the serial implementation of SPRINT, Like SLIQ it uses one time sort of the data items and it has no restriction on the input data size. Unlike SLIQ it uses two data structures; attribute list and histogram which is not memory resident, this implementation makes SPRINT more suitable for large data set, thus it removes all the data memory restrictions on data. It handles both continuous and categorical attributes.

2.3.7. Random Forest

Random Forest [19] are ensemble of unpruned binary decision trees, unlike other decision tree classifiers, Random Forest grows multiple trees which creates a forest like classification. Each tree is grown on bootstrap sample of the training set (this help in over fitting). Ensemble learning method of Random Forest is very promising technique in terms of accuracy. It also provides a distributed aspect that can be adapted to distributed data mining [20]. In the tree grown phase of standard trees (ID3, C4.5, CART, SLIQ, SPRINT, BFTree) each node is split using the best split among all variables. In a random forest, each node is split using the best split among a subset of predictors randomly chosen at that node.

This somewhat counterintuitive strategy turns out to perform very well compared to many other classifiers, including discriminant analysis, support vector machines and neural networks, and is robust against over fitting [19]. The Random Forest produces has better performance when compared to that of single tree classifier [21]. This method is computationally effective, whereas it can be applied when the number of variables is much larger than the number of samples. It also does not over fit and is robust to noise. Random Forest includes a worthy way for estimating missing data, and maintains accuracy when a large proportion of the data are missing.

3. Experimental Setup

Currently, there are many available tools that can be employed in text mining. These tools allow execution of several tasks in text mining such as data preprocessing, classification, regression, clustering, association rules, features selection and visualization. All the above mention tasks are categorized under different algorithms and are available in applications or tools, or as plugins added to applications. In this paper, we choose WEKA 3.6 (The Waikato Environment for Knowledge Analysis) to conduct our experiments on the decision tree algorithms.

The decision tree algorithms have been implemented to assess their Accuracy, Learning Time and Tree Size. This experiments were carried out on four datasets including transportation (vehicle), planets (mushroom), medical area (chronic kidney disease), and phishing websites. The datasets were taken from the UCI Machine Learning Repository[1]. Table 1 shows datasets details.

Table 1. Datasets Details

Datasets	Attributes	Classes	Instances
Vehicle	19	4	846
Mushroom	23	2	8124
Chronic Kidney Disease	25	2	400
Phishing Websites	30	2	2456

[1] https://archive.ics.uci.edu/ml/

4. Results and Discussion

Based on the observations on our experimental results the comparison is as follows:

4.1. Accuracy

Accuracy indicates to the reliability of the decision tree algorithms, it is used in the comparison of different approaches. Accuracy is the percentage of test samples that are correctly classified.
Table 2 shows the accuracy values of the ensemble decision trees (Random Forest) is better than the accuracy of the other algorithms like C4.5, CART, BFTree, SLIQ, SPRINT. The accuracy of the presorting algorithms is better than the accuracy of the entropy based induction decision trees (C4.5, CART, and BFTree). **Figure 2. Accuracy Chart**.

Table 2. Accuracy of Different Algorithms

Datasets	C4.5	CART	BF Tree	SLIQ	SPRINT	Random Forest
Vehicle	71.01	70.83	72.56	79.05	79.62	83.61
Mushroom	100	99.93	100	100	99.96	100
Chronic Kidney Disease	99	97.5	97	97	96.96	99.75
Phishing Websites	95.87	95.79	95.69	95.7	95.58	97.34

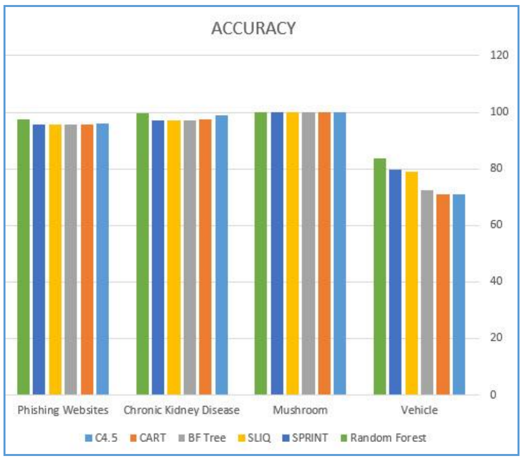

Figure 2. Accuracy Chart

4.2. Tree Size

Table 3 shows the tree sizes of different decision tree classifiers. The results show that when the size of the decision tree grows, then the number of operations, which has to be done for classification, increases accordingly. Therefore, the simplicity of this approach leads to reduction of the classification time. The Random Forest ensemble of 10 trees with 5 random features out of with different bag errors [22].

Table 3. The Tree Size Of Different Algorithms

Datasets	C4.5	CART	BF Tree	SLIQ	SPRINT	Random Forest
Vehicle	195	159	117	423	49	----
Mushroom	30	13	13	158	38	----
Chronic Kidney Disease	14	11	33	44	18	----
Phishing Websites	297	487	599	613	321	----

4.3. Learning Time

Learning Time is the time that is taken for learning and constructing decision trees, within several approaches. **Table 4** shows the learning times of different classification algorithms in seconds, also shows in chart as **Figure 3. Learning Time** explains. It is observed that the number of instances increases the time taken to construct the decision tree increases.

Table 4. Learning Time

Datasets	C4.5	CART	BF Tree	SLIQ	SPRINT	Random Forest
Vehicle	0.17	0.39	0.39	0.13	0.11	0.03
Mushroom	1.26	6.17	6.55	6.44	5.13	1.02
Chronic Kidney Disease	0.09	0.24	0.65	0.61	0.59	0.55
Phishing Websites	0.3	3.91	12.18	4.01	3.94	4.01

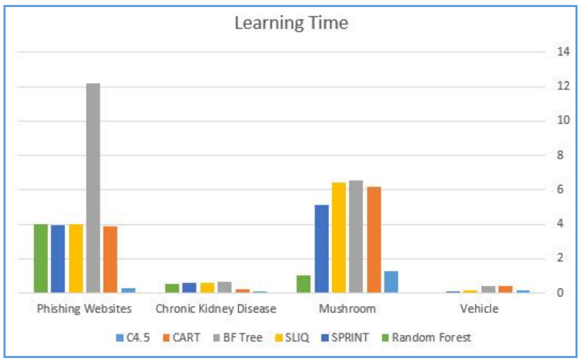

Figure 3. Learning Time

5. Conclusion and Future Work

Decision tree is one of the most popular classification techniques in text and data mining. Selection of a classifier for certain data set is a challenges task. However, if the basic features of these classifiers ware known, then it is quite easier to select the most relevant classifier that can provide better results.

In this paper, we focused our experiment on C4.5, CART, BF Tree, SLIQ, SPRINT and Random Forest serial decision tree algorithms, compared between them based on their accuracy, learning time and tree size. We observed that, the increase of the dataset size and its attributes, the more the learning time and tree size. In addition, we conclude that SPRINT and Random Forest algorithms have good classification accuracy over above compared algorithms, and noted that SLIQ and SPRINT are suitable for larger data sets whereas C4.5 is best suited for smaller data sets.

In future, we plan to conduct experiments on parallel decision tree algorithms, and compare them with the serial implementation of decision tree algorithms, in order to determine the best type, and the controlling conditions in the results.

10. References

[1] Anyanwu, M.N. and S.G. Shiva, Comparative analysis of serial decision tree classification algorithms. International Journal of Computer Science and Security, 2009. 3(3): p. 230-240.

[2] Priyama, A., et al., Comparative Analysis of Decision Tree Classification Algorithms. International Journal of Current Engineering and Technology, 2013. 3(2): p. 334-337.

[3] Korde, V. and C.N. Mahender, Text classification and classifiers: A survey. International Journal of Artificial Intelligence & Applications (IJAIA), 2012. 3(2): p. 85-99.

[4] Ghosh, S., S. Roy, and S. Bandyopadhyay, A tutorial review on Text Mining Algorithms. International Journal of Advanced Research in Computer and Communication Engineering, 2012. 1(4): p. 7.

[5] Osei-Bryson, K.-M., Post-pruning in decision tree induction using multiple performance measures. Computers & operations research, 2007. 34(11): p. 3331-3345.

[6] Lomax, S. and S. Vadera, A survey of cost-sensitive decision tree induction algorithms. ACM Computing Surveys (CSUR), 2013. 45(2): p. 16.

[7] Srivastava, A., et al., Parallel formulations of decision-tree classification algorithms. 2002: Springer.

[8] Yang, N., T. Li, and J. Song. Construction of decision trees based entropy and rough sets under tolerance relation. in International Conference on Intelligent Systems and Knowledge Engineering 2007. 2007. Atlantis Press.

[9] Kwok, S.W. and C. Carter, Multiple decision trees. arXiv preprint arXiv:1304.2363, 2013.

[10] Quinlan, J.R., C4. 5: programs for machine learning. 2014: Elsevier.

[11] Speybroeck, N., Classification and regression trees. International journal of public health, 2012. 57(1): p. 243-246.

[12] Wu, X., et al., Top 10 algorithms in data mining. Knowledge and Information Systems, 2008. 14(1): p. 1-37.

[13] Friedman, J., T. Hastie, and R. Tibshirani, Additive logistic regression: a statistical view of boosting (with

discussion and a rejoinder by the authors). The annals of statistics, 2000. 28(2): p. 337-407.

[14] Zhang, Y., Z. Zhang, and R. Gao, Study on Customer Relation Management Based on Data Mining SLIQ Algorithm, in LISS 2014. 2015, Springer. p. 475-481.

[15] Zeng, L., et al., Distributed data mining: a survey. Information Technology and Management, 2012. 13(4): p. 403-409.

[16] Anyanwu, M. and S. Shiva. Application of Enhanced Decision Tree Algorithm to Churn Analysis. in 2009 International Conference on Artificial Intelligence and Pattern Recognition (AIPR-09), Orlando Florida. 2009.

[17] Jian, L., et al., Parallel data mining techniques on graphics processing unit with compute unified device architecture (CUDA). The Journal of Supercomputing, 2013. 64(3): p. 942-967.

[18] Shafer, J., R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classi er for data mining. in Proc. 1996 Int. Conf. Very Large Data Bases. 1996. Citeseer.

[19] Criminisi, A., J. Shotton, and E. Konukoglu, Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. 2012: Now.

[20] Nojima, Y., S. Mihara, and H. Ishibuchi. Ensemble classifier design by parallel distributed implementation of genetic fuzzy rule selection for large data sets. in Evolutionary Computation (CEC), 2010 IEEE Congress on. 2010. IEEE.

[21] Ng, E.L. and Y. Abu Hasan, Evaluation Method in Random Forest as Applied to Microarray Data. Malaysian Journal of Mathematical Sciences, 2008. 2(2): p. 73-81.

[22]Verikas, A., A. Gelzinis, and M. Bacauskiene, Mining data with random forests: A survey and results of new tests. Pattern Recognition, 2011. 44(2): p. 330-349.