Classification using Probabilistic Random Forest

Rajhans Gondane
System Science and Automation
Indian Institute of Science
Bangalore, India
rajhans.gondane@gmail.com

V. Susheela Devi
Computer Science and Automation
Indian Institute of Science
Bangalore, India
susheela@csa.iisc.ernet.in

***Abstract*—The Probabilistic random forest is a classification model which chooses a subset of features for each random forest depending on the F-score of the features. In other words, the probability of a feature being chosen in the feature subset increases as the F-score of the feature in the dataset. A larger F-score of feature indicates that feature is more discriminative. The features are drawn in a stochastic manner and the expectation is that features with higher F-score will be in the feature subset chosen. The class label of patterns is obtained by combining the decisions of all the decision trees by majority voting. Experimental results reported on a number of benchmark datasets demonstrate that the proposed probabilistic random forest is able to achieve better performance, compared to the random forest.**

I. Introduction

In recent times the data stored in databases is growing consistently. This growth of data needs some techniques to transform the data into meaningful information and knowledge. The decision tree classifier is a supervised learning approach, which selects a set of reducts from the feature space and generate decision rules based on known data. In decision tree the goal is to minimize the number of decision rules and classify the test data correctly with higher accuracy. An efficient ID3 algorithm was proposed in 1979, which decides the reducts based on information gain. The ID3 algorithm leads to inaccurate decision making when there is noise in the sample data and if the decision tree is binary then we need a long sequence of tests [1].

In random forest the concept of bagging is used to generate diverse ensemble classifiers [2], but the problem with applying only bagging is that the first splitting node in the decision tree remains the same [3] (even if we sample the data with replacement). So the random forest uses bagging as well as randomly selected inputs at each node to grow each tree [3]. This extra randomness improves the accuracy of random forest to be as good as Adaboost and sometimes even better [3].

The Roulette wheel selection strategy is based on roulette wheel mechanism to probabilistically select individuals based on some measure of their performance [4]. Roulette wheel selection method is stochastic sampling with replacement (SSR). So it gives zero bias but potentially unlimited spread [4]. In this, individuals are mapped one-to-one into continuous interval of rang $[0, 1]$. The possibility of an individual having large segment size in roulette wheel being selected is larger than the one having smallest segment size [4].

F-score is an efficient feature ranking strategy that does not depend on the class labels [5]. F-score of every feature is a fraction of the sum of the discrimination between the sets of different classes and the sum of the discrimination within each sets of classes. A larger F-score of any feature indicates that the feature is more discriminative [5].

The Probabilistic random forest (PRF) proposed by us is an ensemble learning model which is constructed using the concept of F-score and roulette wheel selection strategy. In PRF instead of choosing feature subset at random for every tree in the random forest, we choose a feature subset based on roulette wheel selection. In roulette wheel selection, the more discriminative feature have greater probability of being selected in the feature subset.

II. Background Theory

A. Decision Tree

The Decision tree classifier is a supervised learning method that is built from a set of training examples [6]. The Decision tree uses, recursive top-down partitioning process and divide and rule approaches for dividing the search space into several subsets in its construction.

In Decision tree, the best feature is selected as a split point so that the data in each descendent subset are purer than the data in the parent superset and finally it will classify into some classes. Each path from root to the leaf node is known as a *Decision rule* and the subset of features involove in the final decision tree are called *Reducts* of the decision tree.

The Decision tree (see Fig. 1) consist of splitting nodes for testing unknown data samples, edges which represent the outcome of the splitting and leaves which represent the class labels. Algorithms like ID3 and C4.5 uses Entropy ($eq.1$) and

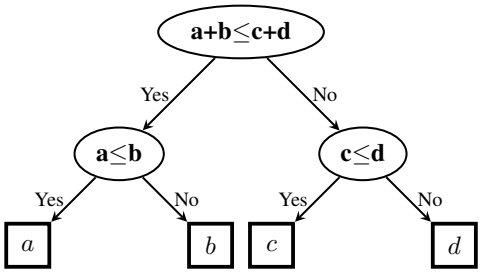


Fig. 1: Decision Tree

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.35

Information gain ($eq.2$) for choosing the best splitting point for the features based on the available samples [7], [8]. The main goal of the decision tree algorithm is to minimize the number of decision rules. The C4.5 is improved algorithm of ID3. In gain value calculation, it allows missing data by simply ignoring them and looks only at those records which have some values. The problem with Information gain is that, it favours the feature having high branching factor which means it is likely to choose the feature which will split the data into smaller fragments.

$$E(S) = -\sum_{i=1}^{c} P_i \log_2 P_i \tag{1}$$

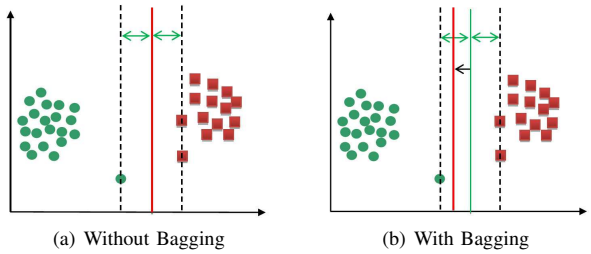
$$Gain(S, B) = E(S) - \sum_{j=1}^{n} f_S(S_j) E(S_j) \tag{2}$$

P_i : Proportion of Label i elements in set S
S_j : Subset of element having B_j value
$f_S(S_j)$: Frequency of subset S_j in Set S
$E(S)$: Entropy of Set S
c : Number of classes
n : Number of different values of conditional attribute $B = (B_1, B_2, \ldots, B_n)$

However, the decision tree also has some drawbacks. If the depth of the tree is very low, then it has to face the problem of *underfitting* and if the depth is more then it has to face the problem of *overfitting*. Mostly because of larger depth the decision tree becomes an over-complex classifier that does not generalize well [1]. Pruning [9] and Ensemble learning are the two techniques used to solve this problem.

Ensemble learning uses the fact that ensemble of classifiers can give more accurate results than the single classifier [2]. If the number of decision trees are trained based on the same training data without sampling then the ensemble model will work as a maximal margin classifier (like Support Vector Machine). As shown in Fig.2(a) just because of one outlier the position of decision boundary is affected. This issue can be solved by training a number of classifiers each with only a part of the training samples.

Bagging [10] is based on bootstrapping and aggregation. It generates many bootstrap samples from training data with (or without) replacement and trains classifiers with one bootstrap sample each and then aggregate the results of the classifiers. In Bagging the performace of the classifier trained with bootstrap samples in which the outlier is not present is better than the classifier trained based on complete training data [11]. If the presence of classifiers trained without outliers is in majority then it will improve the performance of ensemble model. Thus, the ensemble model with bagging sometimes gives better performance than an individual classifier [11]. As shown in Fig.2(b) when bagging is used in the classification model we can see that our decision boundary moves slightly towards the left. This small movement makes sense in terms of test data.



(a) Without Bagging (b) With Bagging

Fig. 2: Ensemble Learning

The Decision forest [12] is a classification paradigm which combines multiple decision trees for classification. There are two methods for constructing decision forest: Sample Subset Method (SSM) and Feature Subset Method (FSM). In SSM, it randomly selects several training samples from the original training data and then each of the subset of training samples is used to construct one decision tree of the forest. This method is preferable when patterns are large. In FSM, a subset of features are chosen at random from the original set of features (if there are d features in original data sample then r features are chosen at random, where $r < d$) and then each of the subset of features is used to train one decision tree of the forest. This is preferable in many-features cases. Due to SSM and FSM, the random decision forest is suitable for classification of large datasets and datasets with high dimensionality.

B. Random Forest

The Random forest is an ensemble learning technique developed by Breiman [3]. The Random forest was constructed using two concepts: Subset of samples and then Random subset of features. The main difference between the decision forest and the random forest is that, the decision forest uses either sample subset method or feature subset method in its construction whereas the random forest uses both.

The Random forest combines a number of decision trees, constructed using subset of training samples and chooses the best feature as split point among random subset of features at each split node. Suppose there are m decision trees (see Fig.3) then the sample data is divided into m subset of samples to train each decision tree. If there are d features in the dataset then the best split point is calculated from randomly selected $\sqrt{d}$ features at each split node of the tree. This extra randomness in the random forest increases the performance. After m trees are generated, the decision of the trees for the class label of a test pattern is decided by majority voting.

In 2001, Breiman [3] mathematically defines a random forest as a combination of m decision tree classifiers as $\{h_i(x, \theta_i), i = 1 \ldots m\}$, where θ_i is a random vector generated independent of past random vectors $\{\theta_1, \ldots, \theta_{i-1}\}$ from the same distribution and for each input x, each decision tree classifier gives a unit vote to find the most popular class. The margin function (**MF**) for the ensemble of m classifiers is defined as:

$$\mathbf{MF}(X, Y) = \mathbf{avg}_m \mathbf{I}(h_m(X) = Y) - \max_{i \neq Y} \mathbf{avg}_m \mathbf{I}(h_m(X) = i) \tag{3}$$

where, X and Y are the random vectors from which the training samples are drawn randomly and **I(.)** is the indicator function. The margin function (eq.3) defines the difference of the average number of votes at X, Y for correct decision class and the average vote for any other decision class. The

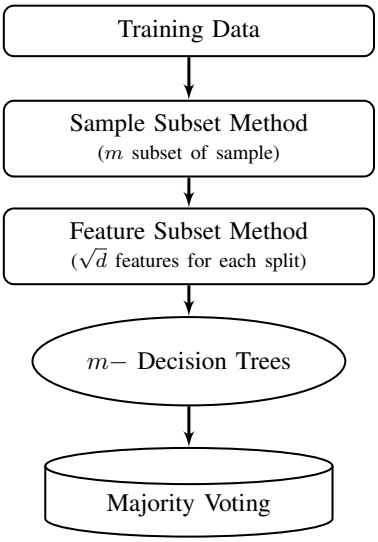


Fig. 3: Random Forest Algorithm

accuracy of random forest increases with increase of margin function [3]. The Generalization error ($\mathbf{G}_e$) of random forest can be defined as:

$$\mathbf{G}_e = P_{X,Y}(\mathbf{MF}(X,Y) < 0) \tag{4}$$

Hence, by Strong Law of Large Numbers we can say that as the number of decision trees increases in random forest, all sequences $\theta_1 \ldots$ converge to

$$P_{X,Y}(P_\theta(h(X,\theta) = Y) - \max_{i \neq Y} P_\theta(h(X,\theta) = i) < 0) \tag{5}$$

Eq.5 shows that it will always converge because of Strong law of large numbers, which shows that overfitting is not a problem in random forest [3].

In Random forest, an upper bound for the generalization error depends on the accuracy of individual decision trees and the diversity between the trees [3]. It is given by:

$$\mathbf{G}_e \leq \frac{\bar{\rho}(1-s^2)}{s^2} \tag{6}$$

s : Strength of set of decision trees
$\bar{\rho}$: Mean value of correlation between decision trees

Hence, Eq.6 says that we can improve the performance of random forest by improving the accuracy of individual decision trees and increasing the diversity between the trees.

In Random forest, bagging process improves the performance of the single decision tree by reducing the variance error without significantly increasing the bias [13] and no pruning is performed, so all decision trees of random forest are maximal trees [14].

The Random forest algorithm can be used for estimating the importance of features by looking at how much generalization error increases when that feature is permuted while all other features remain unchanged [14]. This can be calculated for every tree in the random forest. A complete discussion of the random forest can be found in Breiman [3].

In 2005, M.Hamaz and D.Larocque [15] concluded that the random forest is significantly better than bagging, boosting and single decision trees and it is more accurate and more robust to noise than the other methods

C. Roulette Wheel Selection

Roulette wheel selection technique of the Genetic Algorithm employs a roulette wheel mechanism [4]. Roulette wheel selection method uses fitness values of individual's for selecting the best. In roulette wheel selection, probability of selecting an individual is directly proportional to its fitness value i.e. an individual's selection corresponds to a portion it's covering in roulette wheel [16]. If $f_1, f_2, \ldots, f_N$ be fitness values of individual $X_1, X_2, \ldots, X_N$, then the selection probability p_i of individual X_i is defined as:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{7}$$

Obviously, those having larger fitness value have more probability of being selected. The fittest individual will occupy the larger space, whereas the least fittest have correspondingly smaller space within the roulette wheel [16]. In roulette wheel selection each time, a real random number r in the range of $[0, 1)$ is generated, and the individual X_k where k satisfies (eq. 8) is selected [17]. In roulette wheel selection, the segment size and selection probability remain same throughout the selection phase [4].

$$k = \min\{j \mid r \leq \sum_{i=1}^{j} p_i \ , \ j = 1, 2, \ldots, N\} \tag{8}$$

As shown in Fig. 4, individual X_3 is the fittest one (i.e. having largest fitness value f_3), so its probability p_3 of being selected is large and it occupies largest segment, whereas individual X_1 is least fittest (i.e. having smallest fitness value f_1), so its probability p_1 of being selected is correspondingly very small and it occupies smallest segment in the roulette wheel.

The main advantage of using roulette wheel selection is that it discards none of the individuals in the population and gives a chance to all the individuals to be selected. Therefore, diversity in population is preserved in roulette wheel selection [16].

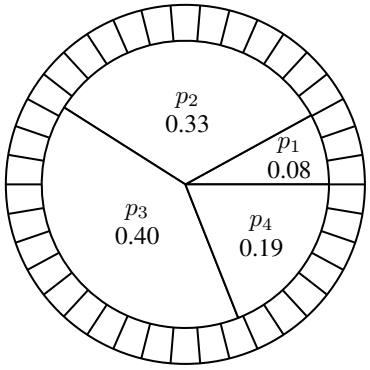


Fig. 4: Roulette Wheel Selection

However, roulette wheel selection has some drawbacks. If some population contains few very fit but not the best individuals and the rest of the population not correspondingly good, then those few individuals will dominates the whole population and prevent the population from exploring better individuals. On the other hand, if all individuals have similar probability of being selected, then it will be very difficult to select the better one among the fit and unfit individuals [16].

D. F-score

F-score (Fisher score) is a simple and efficient technique to carry out feature selection. The F-score measures the discrimination of sets of real numbers [18]. This score is based on statistical characteristics of pattern and it is independent of the class labels. Given the training vector $x_i,\ i = 1, 2, \ldots, d$, F-score of every feature is computed using information of training data. The F-score of the j^{th} feature is define as:

$$\mathbf{F}(j) = \frac{\sum_{k=1}^{c}(\bar{x}_j^k - \bar{x}_j)^2}{\sum_{k=1}^{c}\frac{1}{n_k-1}\sum_{i=1}^{n_k}(x_{i,j}^k - \bar{x}_j^k)^2} \tag{9}$$

$\bar{x}_j$: Average of j^{th} feature in whole dataset
$\bar{x}_j^k$: Average of all elements of j^{th} feature belong to k^{th} class
$x_{i,j}^k$: i^{th} element of j^{th} feature of k^{th} class
c : Number of classes in dataset
n_k : Number of elements of k^{th} class in dataset

The numerator of Eq. 9 indicates the inter class variance, while the denominator indicates the sum of variance within each class. A larger F-score indicates that, the feature is more discriminative. If $\mathbf{F}_i > \mathbf{F}_j$, it means that the i^{th} feature is more discriminative than j^{th} feature. A disadvantage of F-score is that it considers each feature separately and it does not reveal mutual information between features [5].

III. Methodology

In roulette wheel selection, individuals are selected with probability proportional to the segment of a roulette wheel they are occupying and gives a chance to all individuals (based on their probability) to be selected. The F-score measures the discrimination between sets using the information in the training data. A larger F-score indicates that the set is more discriminative. The overall idea of probabilistic random forest is based on these two facts.

The Probabilistic random forest proposed by us uses roulette wheel selection strategy and F-score in random forest. In probabilistic random forest instead of selecting $\sqrt{d}$ features randomly from d features (see Fig. 3) for deciding the best at each split, we are selecting $\sqrt{d}$ features for each split based on roulette wheel based feature selection algorithm (see Algorithm 1).

Algorithm 1 Roulette Wheel based Feature Selection

Input: Features $\{f_1, f_2, \ldots, f_q\}$ and their F-score $\{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_q\}$ where $q \leq d$
Output: Feature f_k where $k \in \{1, 2, \ldots, q\}$
1: Compute $fsum \leftarrow \sum_{i=1}^{q} \mathbf{F}_i$
2: $last \leftarrow 0$
3: **for** $i \leftarrow 1$ to q **do**
4: $\quad Rwheel(i) \leftarrow \mathbf{F}_i / fsum$
5: $\quad Rwheel(i) \leftarrow Rwheel(i) + last$
6: $\quad last \leftarrow Rwheel(i)$
7: **end for**
8: $rnum \leftarrow$ GENERATERANDOM(0,1)
9: **for** $k \leftarrow 1$ to q **do**
10: $\quad$ **if** $rnum \leq Rwheel(k)$ **then**
11: $\quad\quad$ **return** f_k;
12: $\quad$ **end if**
13: **end for**

In Algorithm 1 we take features available at that stage $\{f_1, f_2, \ldots, f_q\}$ and their F-scores $\{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_q\}$ where $(q \leq d)$ as input. F-scores of q features are used as a fitness values for roulette wheel selection method. $fsum$ is the summation of all q F-scores, used to calculate the selection probability of individual feature (see eq. 7). $Rwheel(i)$ contains the probability of i^{th} feature being selected. GENERATERANDOM(0,1) is a function, which returns a real random number between 0 and 1. Finally, Algorithm 1 returns the feature f_k satisfies the statement 10 of algorithm.

In Probabilistic random forest, if there are d features $\{f_1, f_2, \ldots, f_d\}$ then initially F-score of all features $\{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_d\}$ is computed (eq. 9) using information of

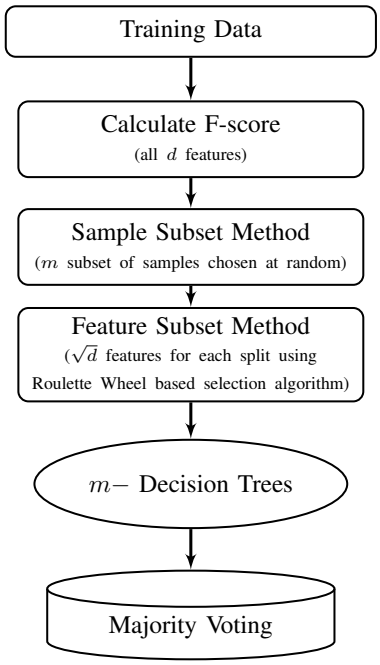


Fig. 5: Probabilistic Random Forest Algorithm

TABLE I: Dataset detail

Dataset	Samples	Training Samples	Test Samples	Features	Classes
wine	178	119	59	14	3
wdbc	596	398	198	31	2
digit	5620	3747	1873	65	10
libras	360	240	120	91	15
letter	20000	13334	6666	17	26
MiniBooNE	130064	86701	43363	51	2

TABLE II: Timebase comparison between Random Forest and Probabilistic Random Forest

Dataset	Random Forest		Probabilistic Random Forest	
	Accuracy $(percent)$	Time $(sec.)$	Accuracy $(percent)$	Time $(sec.)$
wine	91.97 ± 2	2.97	92.24 ± 4	3.67
wdbc	89.37 ± 2	14.71	93.22 ± 1	8.46
digit	53.26 ± 4	152.46	76.52 ± 2	311.27
libras	44.05 ± 4	24.73	46.75 ± 6	27.54
letter	63.30 ± 1	4123.62	65.96 ± 1	4074.17
MiniBooNE	71.94 ± 1	134.55	74.33 ± 2	162.66

training data. If there are m decision trees then as a process of bootstrapping, the training data is divided into m bootstrap samples. Then each decision tree i is trained using i^{th} bootstrap sample of training data. While training the decision tree, the best split feature is calculated among $\sqrt{d}$ features chosen based on *Roulette Wheel based Feature Selection Algorithm* (see Algorithm 1) from d features at each spilt, as shown in Fig.5. While testing, the test data is passed through each decision tree of probabilistic random forest and decision class is calculated based on each decision tree result. Then the final result of the probabilistic random forest is calculated based on which is the most popular class and that final class label calculated by majority voting is considered as class label of the test pattern by the probabilistic random forest.

Algorithm 2 Calculate Split Points Values

Input: Features column vectors $\{f_1, f_2, \ldots, f_d\}$
Output: Split Points $\{s_1, s_2, \ldots, s_d\}$

for $i \leftarrow 1$ to d **do**
  $min \leftarrow$ MIN(f_i)
  $max \leftarrow$ MAX(f_i)
  $check \leftarrow \infty$
  $inc \leftarrow (max - min)/100$
  for $j \leftarrow min$ to max **do**
    $l \leftarrow$NUMBEROFVALUES $(f_i \leq j)$
    $r \leftarrow$NUMBEROFVALUES $(f_i > j)$
    $temp \leftarrow$ ABSOLUTEVALUE$(1 - (l/r))$
    if $temp \leq check$ **then**
      $check \leftarrow temp$
      $s_i \leftarrow j$
    end if
    $j \leftarrow j + inc$
  end for
end for

IV. Implementation & Results

In Probabilistic random forest algorithm approximately $2/3$ of observations are considered as training set and the remaining $1/3$ of data is said to be Out of bag (OOB). This OOB data is used for testing each decision tree. Let there be m decision trees in the probabilistic random forest so the training data is divided into m samples each of $2/3$ size of training data. In probabilistic random forest we used only binary decision trees which will divide the set of elements into two parts based on split point value. The splitting value which will divide the samples approximately in equal parts is chosen as split points (using Algorithm 2) for continuous predictor variables in a decision tree.

Table I shows the details of some well known datasets taken from UCI-Machine Learning Repository [19]. In our experiments, we have covered different types of datasets having large (or small) number of samples, large (or small) number of features and binary (or multiple) classes. Due to the randomness involved in the random forest algorithm, we ran each experiment 10 times and the average and standard deviation of the accuracy obtained has been reported in the Table II & III.

Table II shows the performance and the timebase comparison of Breiman's random forest and probabilistic random forest having 50 trees in their decision making. It can be seen that for most of the datasets probabilistic random forest is taking more time in calculation than the random forest. However, the probabilistic random forest is giving higher accuracy when compare to random forest.

One of the parameter to be determined is the number of decision trees to use. Table III shows the comparison of random forest and probabilistic random forest for different number of trees. It can be seen that the performance of both random forest and probabilistic random forest is improving with increase in number of trees and probabilistic random forest is consistently giving better classification accuracy than the Breiman's random forest irrespective of number of trees. A tree size of 100 is giving the best result.

V. Conclusion

It can be seen that with slight increase of time, the probabilistic random forest gives better performance as compared to information gain based random forest. As in the case of random forest, the classification accuracy of probabilistic random forest consistently increases with increase in number of decision trees. The extra time required is only for the selection of features which are selected stochastically instead of randomly.

TABLE III: Classification accuracy for Random Forest (RF) and Probabilistic Random Forest (PRF)

Dataset	Number of Trees					
	20 Trees		50 Trees		100 Trees	
	RF	PRF	RF	PRF	RF	PRF
wine	90.17 ± 4	91.20 ± 3	91.97 ± 2	92.24 ± 4	91.26 ± 4	93.71 ± 4
wdbc	88.53 ± 1	91.73 ± 2	89.37 ± 2	93.22 ± 1	90.21 ± 2	93.43 ± 1
digit	51.46 ± 3	72.93 ± 4	53.26 ± 4	76.52 ± 2	56.85 ± 3	77.80 ± 2
libras	40.55 ± 5	42.50 ± 6	44.05 ± 4	46.75 ± 6	46.20 ± 5	47.67 ± 6
letter	62.19 ± 1	65.14 ± 1	63.30 ± 1	65.96 ± 1	63.77 ± 1	66.18 ± 1
MiniBooNE	70.63 ± 1	73.92 ± 1	71.94 ± 1	74.33 ± 2	72.14 ± 1	74.48 ± 2

REFERENCES

[1] R. Patel Brijain and K. K. Rana, "A survey on decision tree algorithm for classification," in *International Journal of Engineering Development and Research*, vol. 2. IJEDR, 2014.

[2] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, no. 2, pp. 139–157, Aug. 2000. [Online]. Available: http://dx.doi.org/10.1023/A:1007607513941

[3] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://dx.doi.org/10.1023/A%3A1010933404324

[4] A. Zalzala and P. Fleming, *Genetic Algorithms in Engineering Systems*, ser. Control Series. Institution of Electrical Engineers, 1997. [Online]. Available: https://books.google.co.in/books?id=Cwo0d1ymPIkC

[5] Y.-W. Chang and C.-J. Lin, "Feature ranking using linear svm," *Causation and Prediction Challenge Challenges in Machine Learning*, vol. 2, p. 47, 2008.

[6] L. Rokach and O. Maimon, *Data Mining with Decision Trees: Theroy and Applications*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2008.

[7] J. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986. [Online]. Available: http://dx.doi.org/10.1007/BF00116251

[8] Quinlan and J. Ross, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[9] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[10] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996. [Online]. Available: http://dx.doi.org/10.1023/A%3A1018054314350

[11] M. Skurichina and R. P. W. Duin, "Bagging, boosting and the random subspace method for linear classifiers," *Pattern Analysis and Applications*, vol. 5, no. 2, pp. 121–135, 2002. [Online]. Available: http://dx.doi.org/10.1007/s100440200011

[12] Q.-H. Hu, D.-R. Yu, and M.-Y. Wang, "Constructing rough decision forests," in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, ser. Lecture Notes in Computer Science, D. lzak, J. Yao, J. Peters, W. Ziarko, and X. Hu, Eds. Springer Berlin Heidelberg, 2005, vol. 3642, pp. 147–156.

[13] P. Bonissone, J. Cadenas, M. Garrido, and R. Dıaz-Valladares, "A fuzzy random forest: Fundamental for design and construction," in *Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU08)*, 2008, pp. 1231–1238.

[14] C.-C. Yeh, F. Lin, and C.-Y. Hsu, "A hybrid {KMV} model, random forests and rough set theory approach for credit rating," *Knowledge-Based Systems*, vol. 33, no. 0, pp. 166 – 172, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095070511200086X

[15] M. Hamza and D. Larocque, "An empirical comparison of ensemble methods based on classification trees," *Journal of Statistical Computation and Simulation*, vol. 75, no. 8, pp. 629–643, 2005.

[16] M. R. Noraini and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving tsp," *Proceedings of the World Congress on Engineering*, vol. II, 2011.

[17] L. Zhang, H. Chang, and R. Xu, "Equal-width partitioning roulette wheel selection in genetic algorithm," in *Technologies and Applications of Artificial Intelligence (TAAI), 2012 Conference on*, Nov 2012, pp. 62–67.

[18] Y.-W. Chen and C.-J. Lin, "Combining svms with various feature selection strategies," in *Feature Extraction*, ser. Studies in Fuzziness and Soft Computing, I. Guyon, M. Nikravesh, S. Gunn, and L. Zadeh, Eds. Springer Berlin Heidelberg, 2006, vol. 207, pp. 315–324. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-35488-8_13

[19] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml