International Journal of Engineering Innovation & Research
Volume 3, Issue 6, ISSN: 2277 – 5668

Data Classification According to the Genetic Binary Tree Based on the Nearest Neighbor

Marzieh Hakami
Department of Computer Engineering and Information Technology, Payame Noor University,
PO Box 19395-3697 Tehran, Iran

Ahmad Faraahi
Assistant Professor, Department of Computer Engineering and Information Technology, Payame Noor University,
PO Box 19395-3697 Tehran, IRAN

***Abstract* – Classification is one of the matters discussed in the field of data mining, being of high importance in knowledge discovery. Many data mining techniques have been applied to deal with classification problems. Decision trees are one of the oldest and most well-known methods to construct classification models, in a way that the model implements itself.**

In this paper, a method which is the combination of the powerful and evolutionary algorithm of GP and the Instance-based learning algorithm of KNN is presented, acting efficiently in separating classes and data. This method is proposed in order to check and improve the presented method of MGP (Multi-age Genetic Programming) and to construct decision trees. The simulation of the proposed method has been done using the software MATLAB on the data collection of UCI. The results indicate an improvement in classification accuracy and a reduction in the runtime of the algorithm compared to the MGP and other methods.

***Keywords* – Decision Tree, Classification, KNN, MGP, GP.**

I. INTRODUCTION

Generally, the learning methods of data mining fall into two supervised and unsupervised groups [1]. Classification could be applied as a supervised learning algorithm in the learning process of the machine, and it could allocate the class titles to data purposes according to class's previous knowledge to which data items belonged [2].

Classification consists of the predicting classification's (class's) specifications values based on the values of other specifications (specifications prediction). In other words, it predicts class or category specifications which include two or more groups known as classes based on the values of other specifications. The training set is a data set which is used to create a classifier, whereas the test set is a data set which is used to measure the classification quality [1, 3].
Classification could be used to extract and explain data models in important data classes and to predict future data procedures. Data classification is firstly used to find the similar specifications of one group of items among many random data; then, it classifies them into different categories and classes (according to the classification model), finally leading to certain and special rules by which future data are predicted. The analyses and predictions of these data could offer a good support of decision in different industries, also having a broad and important impact on technology development [4].

Many of different methods, including decision trees and evolutionary algorithms, presented in the papers are used to solve classification problems [3, 5].

Decision trees are one of the most widely used forms of displaying the classifiers [1]. The decision trees techniques have a range of broad application in constructing classifying methods, for these models are closely similar to the human reasoning and are easily fathomable [6].

The evolutionary pattern is a general term to indicate a number of computational strategies which are based on evolution principles. The production and test algorithms are used according to the population. The evolutionary algorithms are probably repetitive programs which are used in optimization problems.

The evolutionary algorithm pattern is based on the use of possible searching algorithms inspired by the certain points in Darwin theory [1]. Genetic programming is a soft computing search technique which is used to conclude a structured program as a tree through its fitness value. The distinct specifications of GP make it easier for classification, and one of its advantages is flexibility which allows the algorithm to be compatible with the certain requirements of each problem [7]. Using GP is a well-known method to develop and evolve the decision trees [1].

Another approach which is used to classify samples based on proximity to the training samples in the feature space in recognition pattern is the K algorithm of the nearest neighbor [8, 9]. This algorithm is a type of instance-based learning in which a sample with the majority of votes is classified in its neighbors and the most common class out of K nearest neighbors is allocated to that sample. However, implementing the nearest neighbor algorithm for classification is easy and powerful to search the problem space, and it needs a few parameters like the inter space scale and K value for configuration [1, 9, 10].

II. SOFT COMPUTING SYSTEMS

The term soft computing indicates the methods used to find approximate answers to the real world problems including different types of inaccuracy and uncertainty. The guiding principle of soft computing is to develop an acceptable error tolerance for imprecision, uncertainty, partial truth, approximation to achieve flexibility, stability, and the low cost of response. The major soft computing patterns are neural computations, fuzzy logic computations, and evolutionary computations. The systems based on such patterns are Artificial Neural Networks (ANN), Fuzzy Systems (FS), and Evolutionary Algorithms (EA). The evolutionary pattern is a general term to indicate some of the computing strategies which

are based on evolutionary principles. The production and test algorithms are used according to the population. The evolutionary algorithms are possibly repetitive programs which are applied in optimization programs [11].

III. Decision Trees

Decision trees are one of the most widely used forms of display for the classifiers [1]. Decision tree techniques have a broad range of application in constructing classification models because these models are very similar to human reasoning and are easily understandable [6].

The decision tree algorithm is for a data mining technique which repeatedly analyzes and divides a set of data including items by using depth-first greedy method or breadth-first approach. The decision trees produced by these methods have good classification accuracy [2]. However, they are often faced with many forms of complexity. Since constructing an optimized decision tree is a NP-full problem [12], finding an efficient exploratory approach to construct approximately optimized decision trees is necessary.

IV. Genetic Programming

The genetic programming is an evolutionary training method [3] which was first presented by Koza (1992) [5].GP necessarily refers to a type of genetic algorithms which uses a complicated display language to encrypt people of specimen. The most common and applicable display scheme is based on trees, although there are other alternatives. The primary objective of GP was to evolve computer programs, and it is to extract knowledge and information nowadays. GP samples are usually seen as parse trees in which the leaves correspond to the set of terminals (variables and constants), while the internal nodes correspond to non-terminals (operators and functions). Using GP in classification presents interesting advantages whose main one is the flexibility enabling this technique to be compatible with the requirements of each certain problem. GP could be applied to construct classifiers by using different types of display forms like decision trees [13].

V. KNN Algorithm

KNN algorithm is an instance-based training method in which a set of data is used as a reference to classify new samples with the help of an appropriate interspace. To classify a new data sample, its KNNs are to be found, and the number of samples in each class and category are counted K times for its subsets. Also, an example which has to be classified is allocated to the class (category) with a counter plus one.

VI. Multi-Age Programming

Many GP algorithms are introduced to construct a decision tree for classification in data mining. However, when the optimized solution is a very full, narrow-width, or very fragmented tree, obtaining a satisfactory and efficient result is very hard for the algorithm. All in all, the convergence rate caused by choosing operator's load is much greater than that of cutting and mutation operators. Therefore, it always leads to a maximum local solution, but not a general solution [7].

In MGP algorithm, an evolutionary hybrid parallel-serial process has been presented. Decision trees are applied as individuals of different ages (sizes) in this process. In every generation, people are divided into different groups according to age. Their qualifications get more and more enhanced as the age goes up. Also, the competency values are used to choose good individuals in the same group, and the competition between different individuals is considered only inside the same group. Therefore, the choice pressure is limited to a particular area (group), and cutting and mutation operations don't eliminate evolution continuity [7].

VII. The Proposed Method

The first step is the random initialization a population in which each individual is a decision tree:

$$P_t = p_{1,t} + p_{2,t} + \cdots + p_{M,t}. (t = 0) \quad (1)$$

Learning the decision tree C4.5 (Quinlan 1993) is one the most widely used and practical methods for deduction to produce decision trees [4]. Decision trees produced by the algorithm C4.5 could be used for classification. The algorithm C4.5 uses the ratio and gain as the rules and principles of choosing the specification, picking one specification with the highest information as the test specification for a given set each time.

Considering [4], the possibility that one sample randomly chosen from the set of training samples T belongs to the class C_i (i= 1, 2, …, k) is as follows:

$$p_i = \frac{|C_i|}{|T|} \quad (2)$$

$|C_i|$ is the number of samples belonging to the class C_i. Generally, if we have a possibility distribution like $P = (p_1 = \frac{|C_1|}{|T|}, p_2 = \frac{|C_2|}{|T|}, \dots, p_k = \frac{|C_k|}{|T|})$, then the information which is contained in this distribution, which we call entropy P, is defined as follows:

$$Info(T) = I(P) = -\sum_{i=1}^{k} p_i * \log(p_i) \quad (3)$$

I(P) measures the mean of required information to specify a sample's class out of the training set T (weighted set). The more the uniformity of possibility distribution is, the more its information is.

After the training set T is divided according to n outputs of one test attribute X, the informational requirements could be considered as a weighted set on the subsets:

In a way which $T_1, T_2, \dots, T_m$ are the partitions of the set T obtained by the value X.

$$Info(X,T) = \sum_i^n \frac{|T_i|}{|T|} Info(T_i) \quad (4)$$

Also, the value of Gain(X, T) is calculated like the following:

$$Gain(X,T) = Info(T) - Info(X,T) \quad (5)$$

This equation indicates the difference between the required information to identify an entry out of the set T and the required information to identify an entry out of the set T after using the value of attribute X. it, in fact, shows the data gain as a result of the attribute X.

The concept of introduced gain is likely to test the specifications which have a lot of outputs (n). For instance, if one attribute of X has a separate value for each record, then Info(X, T) is equal to zero; therefore, Gain(X, T) is maximized. To make this up, Quinlan suggested using gain ratio instead of gain:

$$GainRatio(X,T) = \frac{Gain\ (X,T)}{SplitInfo\ (X,T)} \quad (6)$$

$SplitInfo(X,T)$ would calculate the information obtained by partitioning the training set T into n subsets on the test attribute X.

$$SplitInfo(X,T) = -\sum_i^n \frac{|T_i|}{|T|} \log_2 \frac{|T_i|}{|T|} \quad (7)$$

So $GainRatio(X,T)$ is the ratio of information, produced through separation, which is useful for classification. We could use the concept of gain ratio to rank the attributes and to construct decision tree in a way that the attribute with the highest gain ratio is placed on the route from root to the leave in each node.

We could act like the following to deal with the case in which attributes have contiguous values. When the attribute X has a contiguous domain, we sort out its values in a training set like $A_1, A_2, \ldots, A_m$ in the ascending order. Then for each A_j, j= 1, 2, …, m, we split the records in two sets. The first set includes the values which are equal to or smaller than A_j, and the second set contains the values which are greater than A_j. For each of these m parts, we calculate *GainRatio(X(j),T), j= 1, 2, …, m*, and we choose a part which maximizes the gain. If all the attributes are contiguous, we will have a binary tree.

This initialization for the first population leads to the fact that pretty good trees are produced from the very beginning. We use the trees produced in the first generation induct better trees.

The second step is to construct a tree by genetic programming. To obtain higher accuracy and to improve the classification with the use of genetic programming, we should produce trees as many as the initial population and substitute decision trees with those trees bearing higher accuracy than decision trees do.

The set of possible constructions in genetic programming include the set of possible combinations of functions which could be recursively derived from the possible set of N functions out of the functions set $F= \{f_1, f_2, \ldots, f_N\}$ and the possible set of M terminals out of the terminals set $T=\{a_1, a_2, \ldots, a_M\}$ [14].

The initial structures of genetic programming include entries in the second generation of the program which is obtained by constructing a tree on a set of functions and terminals at random. This begins with choosing a function out of functions set as tree's root. Choosing a function as tree's root is because we want to create a hierarchical structure not one made up of only one root node.

The act of creating trees includes creating a number of certain trees in each generation with a depth between two and a certain value. The ramped half-and-half method [15] is used to construct trees as follows. Half of the trees with specified width are produced as full trees, and this means the rout length is maximized between each terminal till the root. In fact, trees are obtained by limiting nodes choice to a width smaller than the maximum value than in the functions set and nodes choice at maximum width out of the terminal set. Half of the trees with specified width are trees with a variable width, which means that the length of route between each terminal till the root shouldn't be greater than the maximum value. In fact, trees are obtained by choosing the nodes to a depth smaller than the maximum value at random out of the set of functions and terminals and by choosing nodes with maximum width out of the set of terminals. This initialization process produces various and mass trees which are widely used, and they act successfully. To prevent the production of similar trees, we compare each new tree with the set of previously-produced trees and delete it in case they are the same. So there won't be any repetitive member in the population.

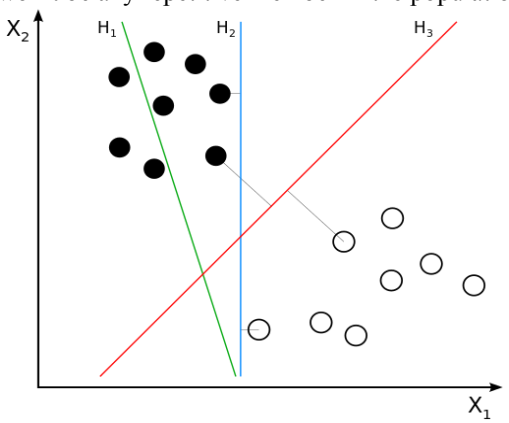

Fig.1. H_1 doesn't separate two classes, but the lines H_2 and H_3 separates two classes well

In the third step, we use the KNN algorithm to enhance the multi-age genetic programming algorithm to improve and construct decision trees. Using the nearest neighbor algorithm in evaluating the ratio function, we can find the optimized answer faster and achieve higher classification accuracy in comparison with multi-age genetic programming algorithm.

The purpose of KNN is to find a matrix, coefficient, or divider to separate classes and also data well. The better this matrix or dividing line is, the higher the classification accuracy will be. An instance of this classification is shown in the following figure. The dividing lines H_2 and H_3 separates these two classes well.

Since the initial trees bear better accuracy rather than random trees, the coefficient matrix is derived from the threshold values of these trees. To do so, we create a matrix in the size of M*M containing these threshold values, with M being the number of attributes existing in

the training data. Then we put all of threshold values obtained from the decision tree in this matrix. After that, we fill the empty elements of this matrix with the values from the training data set at random.

In the second generation, we obtain GP trees from the training data set, too, and compare their accuracy with decision trees. If their accuracy is better than that of the decision trees, we substitute decision trees' values with GP trees' values. In this case, the way of deriving the matrix out of the trees is different from the previous one. At first, we obtain an M*M matrix out of the training data set in a way that M is the number of attributes existing in the training data. Then we put all values of this matrix into the phrase obtained from the tree.

After extracting the matrix, we multiply this matrix (a) into data of both test and training sets like the following to make test data closer to training data and to have a correct data mapping:

$A = Train*a$

$B = Train*a$

Using the nearest neighbor algorithm and the similarity between training data and test data, then we find the best class for test data set by predicting and estimating.

The problem rising while using KNN is that this algorithm is so time-consuming for large data sets because each sample of training set gets processed during classifying a new data, and this needs more classification time [16]; however, this problem cannot be a big deal for some application cases, although time becomes of high importance like classification accuracy while being used in cases like medical diagnosis.

The initial choice of the proper value for k and measuring the interspace could determine the functionality of the classifier KNN. We use the Euclidean correlation to calculate the distance.

After we calculate the inter-set distance between two sets of A and B, we choose K ones of the nearest distances. We consider the value of K to be equal to 4 values [5, 6, 7, 8] in this method. At first, we consider 5 of the nearest distances and specify the values of their classes. Then we keep the class with the greatest frequency and repeat this action for the other values of K. Finally, we allocate the class with the greatest frequency in all values of K to the result class of the desired test data. In this method, we use the similarity of test data with training data to achieve a better estimation of the result class for test data, so the classification accuracy increases noticeably in comparison with other methods, and also the algorithm converges to the optimal answer so faster; therefore, the runtime is reduced.

The fourth step is to choose people from the population P_t in order to form different groups. Individuals have the same age in one group. The age is the size of tree. In MGP, Δage is the age span between two consecutive groups. When Δage is set to a higher value, individuals with different ages are allowed to compete in the same group. If Δage has a lesser value, the algorithm converges to the optimal solution faster. Also, the convergence rate caused by the pressure of choosing operator is greater than the convergence rate caused by cutting and mutation operators. Therefore, the algorithm doesn't find the optimal solution with a greater Δage and leads to a maximum local solution [7].

In the population P_t, we choose individuals to form different groups. So after creating the initial population including M individuals or trees, each tree is categorized according to the terminals existing in it (the tree's age). Every tree with age=k is put into the group k (k=1, 2, …, N). As a result, we will have N same groups in which N is the maximum number of terminals in the initial population, and all the individuals existing in one group are the same age. The population of age=k will be like $p_{k,1}, p_{k,2}, \dots, p_{k,mk}$.

In the fifth step, the fitness values of individuals are calculated in different groups after the classification of population is done. Evaluating the fitness is done according to the equation 8 [7]. The fitness criteria lead the evolutionary process toward approximately optimal solutions. This function is a hybrid of classification accuracy and age of a tree. The fitness of individuals becomes more and more as the age goes up. Therefore, greater values will be accorded to smaller trees by placing the age in the denominator. The fitness values are used to choose good people in one group. The size of the tree remains small due to using a complicated fitness function which combines the accuracy and size of the tree.

$$Fitness(i) = N_{\text{correct (i)}}/(\text{LOG}(age_{\text{i}}) \times N_{\text{total}}) \qquad (8)$$

The individuals are chosen through a certain choosing mechanism. Choosing the roulette wheel [16] is implemented in a way that a range of values is considered, and the individual who is fitter contain a bigger range. Then a random number is produced in the general range. The respective individual is chosen whatever sub-range the number is in. The tournament selection [16] is done in a way that a competition is held among some individuals selected from the population at random. The winner or the best member is chosen as the competition result. The competition size determines how many random members should be chosen for the competition. A big size presents more appropriate solutions for selection. The cutting operator is done by selecting two parents out of the population. Two random sub-trees are chosen out of each parent and exchanged to create children which are added to the population. To implement this algorithm, this operator is applied to 60% of the population, and it is the most important operator because it is the only source to create new individuals and finally fitter individuals and to spread good blocks among generations. The mutation operator [13] selects an individual out of the population and returns it to the population by making changes to it. In this method, a single node is selected out of the parent tree, and it is replaced by a random node of the same type. Selection, cutting, mutation, and reproduction operators act in a parallel way in each group, so they lead to runtime reduction, and individuals with higher fitness will exist in the next generations as a result of reproduction. Then we gather every individual from different groups and repeat this action for the next generations:

$$p_{t+1} = p_{1,t+1} + p_{2,t+1} + \cdots + p_{M,t+1} \quad (9)$$

In the sixth step, the termination condition is checked. The termination condition is the classification accuracy in case it is greater than a constant threshold or reaching the maximum generation. So if the termination condition is established, the algorithm is stopped.

In the final step, we act like the followings to build the decision tree:

By reaching the final generation and finding better trees and better separating lines as a result, we could identify test samples class with higher accuracy with the use of the nearest neighbor algorithm and separate the samples well. Therefore, to find the best attribute which plays the role of separating the samples, we could act like the followings:

1. Finding the distance of each sample from the separating line
2. Selecting K closest samples to the separating line, because separating the classes of samples which are near this separating line is very dependent on the linear separators according which we created this line. These samples are considered to be borderline samples and are of high importance.
3. Selecting an attribute which has the smallest value and closest distance from the line.

The chosen attribute is selected as the root node because it bears high importance to separate samples, and we act like mentioned for the other nodes.

Table 1: Configurable parameters by the user

Parameter's name	**Description**
Population size	The number of trees
Functions set	To create mathematical expressions used to fit the samples of certain finite data
Terminals set	Information to be processed by the mathematical expression
The maximum number of generations	A pre-determined value for algorithm's termination condition to choose the best individual
Cutting possibility	The possibility that an internal node is replaced in each parent as the cutting node.
Mutation possibility	The possibility that an internal node is replaced as the root of a sub-tree.
The maximum depth	Constructing trees to a maximum and constant determined length
The value of K	The number of the nearest neighbors
Δage	The age range between two consecutive groups

VIII. EVALUATION

In this part, we deal with checking the results of implementations we have conducted on some data sets to improve the decision tree by using the hybrid method of serial-parallel genetic programming and the nearest neighbor algorithm. To implement the proposed algorithms, the programming language MATLAB and the software MATLAB 2013 have been used. The results of tests on data sets of UCI indicate that we have achieved a good improvement in classification accuracy and also runtime in comparison with previous methods.

The proposed method has been implemented with the programming language MATLAB and the software MATLAB 2013. Since different results were obtained at each runtime, we ran the algorithm 10 times and reported the mean of obtained results as the final result. To test the efficiency of the proposed method, the method 10-fold CV have been used to avoid the surrounding problem. Consequently, the mean of obtained results is reported as the final result.

One of the most important criteria which have been considered to fit the efficiency of the algorithm is the classification rate which shows the number of samples which have been classified correctly [17]. The following equation is used to find the classification rate:

$$\text{Classification Rate} = \frac{N_{\text{correct}}}{N_{\text{total}}} \quad (10)$$

Table 2: The results of the proposed algorithm in comparison with other methods

Data collection	**Accuracy of GP**	**Accuracy of DTiGP (2010)**	**Accuracy of MGP (2011)**	**Accuracy of KNN-MGP**
Lung cancer	55.8±12.6	58.8±11.8	72.4±3.4	100.0±0.0
Zoo	92.6±4.7	93.3±3.8	93.9±4.9	94.8±3.2
Iris	94.9±4.8	95.1±3.8	96.4±2.6	100.0±0.0
Wine	92.6±5.6	93.4±4.2	95.8±1.8	96.3±3.7
Glass	65.0±7.6	68.0±4.7	71.6±6.8	88.23±5.2
Heart disease	78.8±6.8	78.2±4.8	79.9±5.2	91.4±4.2
Ionosphere	91.2±3.5	92.8±2.8	95.4±3.4	96.37±3.26
Balance scale	98.6±0.8	98.8±0.6	100.0±0.0	96.23±2.72
Breast cancer	95.9±6.8	97.2±1.0	97.8±1.2	98.6±1.4
Pima Indians	70.2±4.2	70.0±3.8	68.9±3.4	79.4±3.8
Car	91.0±1.5	93.0±0.6	94.6±1.7	97.6±1.2

Table 3: Comparing the proposed algorithm's runtime with other algorithms in seconds

Data Collection	GP Time	DTiGP Time	MGP Time	KNN-MGP Time
Lung cancer	8	9	8	3
Zoo	26	29	27	26
Iris	55	63	56	3
Wine	84	91	85	55
Glass	206	246	194	72
Heart disease	103	116	94	92
Ionosphere	155	181	138	248
Balance scale	288	326	204	174
Breast cancer	326	384	263	243
Pima Indians	299	346	243	262
Car	624	719	552	680

As it is observed in table 1, evaluating the proposed approach has had a noticeable improvement rather than other approaches in terms of accuracy. Table 2 indicates the proposed algorithm's runtime in comparison with other algorithms, in a way that this algorithm acts better than other algorithms except for 3 data sets.

IX. Conclusion

The classifying models of data are created in order to convert the learning studies into the learning of the machine. The evolutionary programming, decision trees, and the nearest neighbor searching algorithm are some examples of these models. In this paper, an approach has been presented to improve and construct decision trees in order to classify data with the maximum accuracy and reduce the runtime by using the classifying models like GP and KNN. A number of data set from UCI bank has been used to check the functionality of the proposed method. The results acquired by simulating with the software MATLAB indicate the improvement in accuracy and runtime of the proposed algorithm rather than other approaches in the comparison.

References

[1] P.G Espejo, S Ventura, and Herrera F, "A Survey on the Application of Genetic Programming to Classification," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions, vol. 40, no. 2, pp. 121-144, 2010.

[2] M Anyanwu and S Shiva, "Comparative Analysis of Serial Decision Tree Classi," International Journal of Computer Science and Security, (IJCSS), vol. 3, no. 3, pp. 230-240, 2009.

[3] Pu Wang, Ke Tang, Edward P.K. Tsang, and Xin Yao, "A Memetic Genetic Programming with Decision Tree-based Local Search for Classification Problems," Evolutionary Computation (CEC), IEEE Congress , pp. 917-927, 2011.

[4] Xie Niuniu and Liu Yuxun, "Review of Decision Trees," In: Proceedings of the 2010 3rd IEEE International Conference on Computer Science and Information [5] Technology (ICCSIT), vol. 5, pp. 105-109, 2010.

[5] Chan-Sheng Kuo, Tzung-Pei Hong, and Chuen-Lung Chen, "Applying genetic programming technique in classification trees," Springer-Verlag, vol. 11, no. 12, pp. 1165-1172, 2007.

[6] S. B Kotsiantis, "Decision trees: a recent overview," Journal Artificial Intelligence Review, vol. 39, no. 4, pp. 261-283, 2013.

[7] Li Yi and Kang Wanli, "A new genetic programming algorithm for Building decision tree," vol. 15, pp. 3658-3662, 2011.

[8] J.M. Keller, M.R. Gray, and J.A. Givens, "A fuzzy K-nearest neighbor algorithm," IEEE Transactions on Systems, Man and Cybernetics, vol. 15, no. 4, pp. 580 - 585, 2012.

[9] N Suguna and K Thanushkodi, "An Improved decision tree algorithm with the nearest neighbor," International Journal of Computer Science (IJCSI), vol. 7, no. 4, pp. 18-21, 2010.

[10] M Aci and M Avci, "K nearest neighbor reinforced expectation maximization method," 2012 Third International Conference on Computer and Communication Technology, vol. 38, no. 10, pp. 12585-12591, 2011.

[11] G. Castellano, C. Castiello, A.M. Fanelli, and L Jain, Evolutionary neuro-fuzzy systems and applications.: Springer, 2007.

[12] Shiueng-Bien Yang and Shen-I Yang, "New Decision Tree Based on Genetic Algorithm," Computer Communication Control and Automation (3CA), International Symposium, IEEE Conference Publications, vol. 1, pp. 115-118, 2010.

[13] P.G Espejo, S Ventura, and F Herrera, "A Survey on the Application of Genetic Programming to Classification," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions, vol. 40, no. 2, pp. 121-144, 2010.

[14] T Broughton, A Tan, and P Coates, "The use of genetic programing in exploring 3D design worlds," Junge, R. (ed.) CAAD Futures 1997, Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures 4-6 August 1997, Kluwer Academic Publishers, Munchen, Germany, pp. 885-917, 1997.

[15] J. R Koza., 1992.

[16] NArtem Sokolov and Darrell Whitley, "Unbiased Tournament Selection," Proceedings of the 2005 conference on Genetic and evolutionary computation Washington DC, USA, pp. 1131-1138, 2005.

[17] Vincent Labatut and Hocine Cherifi, "Accuracy Measures for the Comparison of Classifiers," The 5th International Conference on Information Technology, amman : Jordanie (2011), 2012.