EAI Endorsed Transactions
on Energy Web

Research Article EAI.EU

# A Genetic Programming Approach to Binary Classification Problem

Leo Willyanto Santoso[1,*], Bhopendra Singh[2], S. Suman Rajest[3], R. Regin[4], Karrar Hameed Kadhim[5]

[1]Petra Christian University, 121-131 Siwalankerto Rd, Surabaya, East Java, Indonesia.
[2]Associate Professor, Amity University, Dubai.
[3]Vels Institute of Science, Technology & Advanced Studies (VISTAS), Tamil Nadu, India.
[4]Assistant Professor, Department of Information Technology, Adhiyamaan College of Engineering, India.
[5]AL-Musaib Technical College, AL-Furat Al-Awsat Technical University, Iraq.

## Abstract

The Binary classification is the most challenging problem in machine learning. One of the most promising technique to solve this problem is by implementing genetic programming (GP). GP is one of Evolutionary Algorithm (EA) that used to solve problems that humans do not know how to solve it directly. The objectives of this research is to demonstrate the use of genetic programming in this type of problems; that is, other types of techniques are typically used, e.g., regression, artificial neural networks. Genetic programming presents an advantage compared to those techniques, which is that it does not need an a priori definition of its structure. The algorithm evolves automatically until finding a model that best fits a set of training data. Feature engineering was considered to improve the accuracy. In this research, feature transformation and feature creation were implemented. Thus, genetic programming can be considered as an alternative option for the development of intelligent systems mainly in the pattern recognition field.

**Keywords:** binary classification, evolutionary algorithms, genetic programming, machine learning

Received on 26 May 2020, accepted on 10 June 2020, published on 17 July 2020



doi: 10.4108/eai.13-7-2018.165523

*Corresponding author mail: leow@petra.ac.id

## 1. Introduction

Today it is more common to find computer programs that implement artificial intelligence in their operations. It could generate as an application that aim to facilitate the daily life of the human being through the use of ICTs (Information and Communications Technology) as facilitating tools [6], [20], ie, smartphones, security systems, etc. According to [19], it can be said that virtualization, "is an abstraction of technological resources where you can get to use a server or many servers being invisible to the end user." One of these operations is the recognition of patterns, such as the voice. Voice recognition that is integrated almost in all modern smartphones, coupled with finger and facial recognition. In industry, the recognition of patterns has allowed the development of automated systems that are increasingly used and demanded.

Within the stages of pattern recognition, classification is a primordial phase, which can be defined as the assignment of a value to something according to its qualities or characteristics. The classification has been used in different problems of engineering, medicine, and finance [18], [7],[11],[9]. The binary classification is that where the output can take only two possible values, true or false, {0, 1}. Some examples are for the development of diagnostic systems, E.G. (evolutionary algorithms) for the detection of diseases such as diabetes, one suffers or does not suffer. In the financial field, for the development of models to predict whether a business has financial risk, that is, whether it is feasible to invest in the business or not [9].

Another application is in the agricultural sector, through the use of images to classify fields and determine if this field is a potential field for cultivation [12].

Different algorithms like machine learnings have been developed to deal with classification problems, for example: regression algorithms, artificial neural networks, fuzzy logic, and Support Vector Machines (SVM). An alternative use to these machine learning algorithms is genetic programming (GP). The GP is a machine learning technique that is part of the evolutionary algorithms and, which is inspired by evolutionary processes which employ Darwinian principles of survival and reproduction of the fittest.

One of the advantages of GP, is that it does not require a priori definition of its structure, i.e., in neural networks, it is required to define the number of layers and neurons before applying any training algorithm. Therefore, this research focuses on the implementation of the GP in binary classification problems. It is important to develop evidence of the implementation of this technique so that it can be applied mainly in real life problems. The objective of this research is to demonstrate the use of genetic programming to solve binary classification problem. The remaining part of this paper is organized as follows. Section 2 presents the background and the related work. Sections 3 presents the methodology of this research. Section 4 presents the discussion and analysis. Finally, the conclusions are drawn in Section 5.

## 2. Genetic Programming

Genetic programming is an evolutionary algorithm that automatically evolves programs, functions, or any other type of symbolic expression, that carries out some type of calculation, to solve a specific task or problem [1], [3], [8]. Generally, these programs are represented as variable length structures such as a syntax tree.

The most common application of GP, perhaps, is the symbolic regression which attempts to find a mathematical expression that best fits a given set of training data (supervised learning) [3], [8]. So then, the goal is to find that expression K ^ O: R ^ n → R that fits a set T = {($x_1$, $y_1$), ..., ($x_p$, $y_p$)} of $p$ input/output data with xERn and yER and E {0,1} for a binary classification problem.

The general problem of symbolic regression can be defined in (1):

$$K^0 \leftarrow \arg\min_{K \in G} f(K(x,\beta),\gamma) \quad (1)$$

Where G is the search space defined by the primitive set of functions and terminals, P = F U T; f is the objective function which can be the difference between the program output K (x, β) and the desired output y.

Like any other evolutionary algorithm in GP, the initial population is generated randomly. Likewise, genetic operators such as selection, cross and mutation are applied. Although the crossing and mutation operators differ from other evolutionary algorithms, the basic idea is the same.

GP begins with the choice of an initial node or root, then, as for each category of the grammar there is a list, then a draw is made to choose one of them. Again, an element of that list is chosen at random, which may be, depending on the choice, an internal vertex, or a terminal vertex (leaf). If the chosen element is a binary operation, then a binary subtree will be formed that will always put the left son in the first choice and the right son in the second. If the element chosen at random is a function, the vertex only displays a child who can become a binary or unary operator depending on the draw. If in the election, the result is a digit or a variable, this node will become a sheet.

It should be noted that in the process of choosing each node, the options chosen are discarded in the next draw, this in order to give the algorithm a finite and varied process. Finally, the tree is built when the options of the lists are exhausted or when the tree reaches a maximum depth of length 10. The process of construction of trees ends, when the size of the initial population is reached.

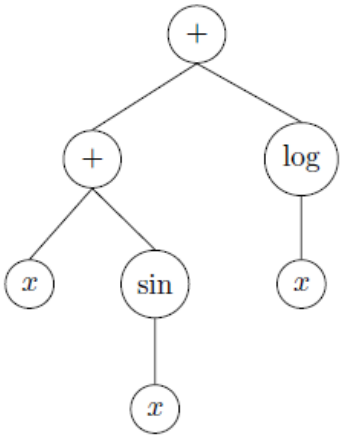


**Figure 1.** Example of a coded individual

Figure 1 shows an individual produced through the previous process. The tree in the previous figure produces the following formula in fixed notation: ((x + sin(x)) + (log(x)). To solve a differential equation, the GP algorithm comprises the following phases:

1. Initialization
2. Adaptation or adjustment function
3. Evaluation of derivatives
4. Genetic operators
5. Completion of the algorithm

## 2.1 Initialization

GP begins with the random generation of individuals (trees) until obtaining a population of size *M*, which is chosen for each problem in a particular way. In this phase, the values for the replication rate and mutation are also established. The replication rate denotes the fraction of the number of trees that would go without any change to the next generation. This is where *tr* is the response rate. The mutation rate controls the average number of changes within a tree.

$$Probability\ of\ crossover = 1 - tr \quad (2)$$

## 2.2 Adaptation or Adjustment Function (fitness)

In the case of ordinary differential equations (ODE). We express the equation in the following way:

$$f\left(x, y, y^{(1)}, \dots, y^{(n-1)}, y^{(n)}\right) = 0, x \in [a, b] \quad (3)$$

where *y (n)* denotes the derivative of order *n* of *y*. The initial or boundary conditions are given by:

$$g_i\left(x, y, y^{(1)}, \dots, y^{(n-1)}\right)|_{x=t_1} = 0, i = 1, \dots, n \quad (4)$$

where you are one of the ends of the domain interval. To calculate the population adaptation function, the following steps must be performed:

1. Choose *N* points, not necessarily equidistant, ($x_0$, $x_1$, ..., $x_{N-1}$) in the domain [a, b].
2. For each tree *i*
   a) Construct and evaluate a formula in fixed notation $M_i(x)$, expressed in the grammar described above, this formula is evaluated at points $x_j$.
   b) Calculate and evaluate at points $x_j$, the *n* derived from $M_i(x)$ necessary to evaluate the differential equation.
   c) Calculate the quantity
   d) Calculate an associated penalty function $P(M_i)$, which depends on the initial or border conditions and has the form:

$$P(M_i) = \lambda \sum_{k=1}^{n} g^2 k\left(x, M_i, {M_i}^{(1)}, \dots, {M_i}^{(n-1)}\right)|_{x=t_k} \quad (6)$$

where λ is a positive number.

   e) Calculate the adaptation value of the tree as:

$$v_i = E(M_i) + P(M_i) \quad (7)$$

Consider the ODE

$$y'' + 100y = 0, x \in [0, 1]$$

With the boundary conditions $y(0) = 0$ and $y'(0) = 10$. Taking in the range [0,1] $N = 10$ equidistant points $x_0$, ..., $x_9$. Suppose we obtained the tree g which is equivalent to the expression $M_g(x) = \exp(x) + \sin(x)$. It can be seen in Figure 2.

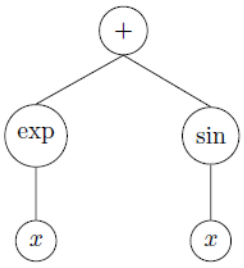


**Figure 2.** Tree that produces the expression $M_g(x) = exp(x) + sin(x)$

The first and second order derivatives for this expression are ${M_g}^1(x) = \exp(x) + cos(x)$ and ${M_g}^2(x) = \exp(x) - \sin(x)$ respectively.
The symbolic derivation of the above expressions is described in detail below. Using the above equation with a weight $w_j = 1$, we have:

$$E(M_g) = \sum_{i=0}^{9} \left({M_g}^{(2)}(x_i) + 100 M_g(x_i)\right)^2$$
$$\sum_{i=0}^{9} \left(101 exp(x_i) + 99 sin(x_i)\right)^2$$
$$= 4849322.3$$

The penalty function P ($M_g$) is calculated with as follows:

$$P(M_g) = \lambda(\left(M_g(0) - y(0)\right)^2 + \left({M_g}^{(1)}(0) - y'(0)\right)^2)$$
$$= \lambda((\exp(0) + \sin(0) - 0)^2 + (\exp(0) + \cos(0) - 10)^2)$$
$$= \lambda((1 + 0 - 0)^2 + (1 + 1 - 10)^2)$$
$$= 63\lambda$$

Thus, the adaptation value $v_g$ associated with this tree is given by:

$$v_g = E(M_g) + P(M_g)$$
$$= 4849322.3 + 63\lambda$$

In this way, all the trees in the population are evaluated and sorted according to their adaptation value. As a cconsequence, when applying genetic operators, a new population is created and the process is repeated until the stopping criterion is met.

## 2.3 Evaluation of the derivatives

The evaluation of the derivatives is done by any method of Automatic Differentiation and the use of the basic rules of differentiation. In our case we have adopted the Python SymPy library to symbolically derive. Basically, what this library does is take the tree resulting from the grammar, transform it into a chain and then evaluate it to obtain the derivative in symbolic form. SymPy is a python library specifically designed to perform symbolic calculation in python in a simple and extensible way. The system is fully written in python and does not require external libraries.

## 2.4 Genetic operations

The genetic operators that apply to the genetic population are: initialization, replication, crossover, and mutation. Initialization is applied only once on the first generation. The initial trees are chosen randomly with a maximum of 10 levels deep. The replication chosen is 10% of the initial population in each generation. The crossing is applied in each generation to create new trees from existing ones, which would replace the worst individuals in the population. In this operation, each pair of parents is selected, then a random node is chosen from each parent for the cut, and subsequently the sub-exchange is made tree or genetic material. Parents are selected by tournament selection, for example:

- First, a group of individuals $K \geq 2$ is created, randomly selected from the current population.
- Individuals with the best adaptation functions in the group are selected, others are discarded. This percentage of selection varies depending on each problem.

Finally, the last genetic operator used is the mutation, where a tree node (listed previously) is randomly changed to another randomly selected node. This operator is applied in each generation to a specific percentage of the population. In each generation the following steps are performed:

1. Trees are ordered with respect to their adaptation functions, so that the best individual is at the top of the list and the worst individual at the end of it.
2. Of the 100% of the initial population, 10% of the best individuals are automatically selected for the next generation (replica). The remaining 90% is selected as follows:
    a. From the originally ordered list of individuals, a percentage of the best individuals is taken, in our case 40%, with which selection is made per tournament (two individuals are taken randomly and the best one is chosen) to form the first 45% of a list that we will call intermediate population.
    b. From this intermediate population, two parents are chosen randomly, to perform the crossing operator, and thus form the other 45% remaining. The new individuals would replace the worst in the population at the end of the crossing operator.
3. The mutation operator is applied to a percentage of the new population, except those individuals who were selected in the replication operation. The importance of this operator is that it generates new individuals and, in turn, new search spaces in those cases in which the population tends to be very homogeneous.

## 2.5 Completion of the algorithm

The genetic operators are applied to the population by creating new generations, until reaching a maximum number of generations or until obtaining the best tree of the population according to the pre-established tolerance.

## 2.6 Related research

This subsection presents some works that have been developed in recent years to solve classification problems, both binary and multiclass. Different learning algorithms have been used in various fields of application. To mention, [15] they developed a binary news classification system through the use of support vector machines (SVMs). Authors such as [17] present a new classification approach by combining logistic regression with decision-theoretic rough sets (DTRS) to solve binary and multiclass problems. [4] propose the incorporation of a numerical optimizer in the nominal GP algorithm using artificial neural networks (ANNs) to classify cardiac signals in order to demonstrate whether an electrocardiogram was normal or could present a problem. Authors such as [9] used fuzzy logic and ANNs for the classification of crop fields. Likewise [13] presents a method based on multinomial logistic regression to solve a multi-class problem of hyperspectral classification. In 2016, they make a comparison between RNAs and SVMs to classify flaws in roller production, obtaining as a result that SMVs provide better results. Similarly, [16] and use SVMs to classify skin diseases [15] and present a deep belief network (DBN) to solve a classification problem about sleep quality [22-23].

Although the works published in different areas are more extensive and through the use of different algorithms, it can be noted that RNAs and MSVs are the most reported in the literature. However, they require an a priori definition of their structure, i.e. In RNAs you have to establish the number of layers and in SVMs the kernel function, before training the algorithm. GP is an evolutionary algorithm that automatically transforms program populations into new populations through the use of genetic search operators. Therefore, it does not require the establishment of an a priori structure. Although there are certain parameters that must be set at the beginning of the algorithm, the final structure is obtained automatically, that is, a classification model generation after generation is generated autonomously.

## 3. Methodology

For binary classification problems, GP was treated as a supervised algorithm where a set of training patterns, x ERn, is used to map a function $g$ (x) E Rn → R. Then, a unit step function (2) is inserted.

$$G(f(x)) = \begin{cases} 0, & f(x) < 0 \\ 1, & f(x) \geq 0 \end{cases} \quad (8)$$

In such a way that G (x): R → {0, 1}, as shown in Figure 3.

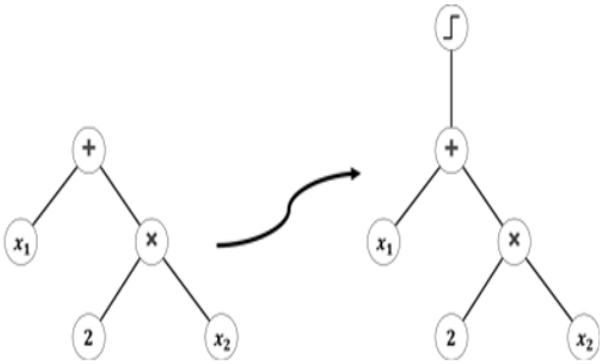


**Figure 3.** Transformation of tree to binary output

In classification problems, accuracy can be used as a metric to evaluate the performance of the models. Accuracy can be defined as the fraction of correct predictions over the total number of cases. Specifically, for binary classification problems it can be calculated in terms of positive and negative predictions (9).

$$\text{accuracy} = \frac{VP + VN}{VP + VN + FP + FN} \quad (9)$$

where VP are the true positives, VN the true negatives, FP are the false positives and FN the false negatives [10], [5]. In this research the objective function (10) was used as a metric to evaluate the performance of the GP in binary classification problems.

$$\text{Obj. Function} = 1 - \text{accuracy} \quad (10)$$

where it is desired to find values equal or close to zero. The standard GP algorithm was implemented using the GPLAB1 Matlab tool. The set of problems to be solved covers a series of binary classification reference problems taken from the repository of the machine learning center and intelligent UCI systems [2]. Table 1 shows a summary of the problems evaluated.

Table 1. Summary of the binary classification problems used to evaluate the GP algorithm

| *Name* | *# of attributes* | *# of instances* |
|---|---|---|
| XOR | 2 | 4 |
| Globe | 8 | 20 |
| Cryotherapy | 6 | 91 |
| Trains | 32 | 10 |
| Tic Tac | 9 | 958 |

The outputs of interest reported are the objective function in (10) and the size of the generated program, used to evaluate the performance of each model in terms of the resources consumed. The developed model is a real time application [14], [21]. The configuration used for each of the problems is shown in Table 2.

Table 2. Parameters of the GP algorithm

| *Parameter* | *Value* |
|---|---|
| Run | 10 |
| Population | 200 |
| Generations | 150 |
| Training Set | 70% |
| Test Set | 30% |
| Tree initialization | Ramped half-and-half, max. Depth 6 |
| Set of functions | +, -, *, sin, cos, log, sqrt, tan, tanh |
| Set of terminals | entry attributes x, constants, |
| Selection | Tournament, size 2 |
| Crossing Operator | Standard sub-tree crossover, 0.8 prob. |
| Mutation Operator | Mutation probability per node 0.15 |
| Elitism | Major individual survives |

## 4. Discussion and Analysis

**Figures 4-7** summarize the results of the GP implementation in selected binary classification problems 3 and 5. The results show the convergence graphs of the objective function and the size of the program with respect to the number of generations, showing the average of the runs. The first problem is the XOR gate, considered a relatively simple problem. Since the number of samples is small, the accuracy in test data is not evaluated. The results of this problem are not plotted since in the generation of the initial population it was possible to find a model with perfect adjustment, that is, an accuracy of 1 with a value of the objective function equal to 0. The same happened with the problem of the balloons, when generating the initial population, a model with perfect adjustment was found in all the runs.

For the problem of cryotherapy, the results were a bit different, Figures 4-5 show the results obtained for this problem. Given that the number of instances is greater than the previous problems, we consider test data. The results show that on average a perfect accuracy was found before generation 30 for the training data set; while with a median of 0.80 in the accuracy for the test data.

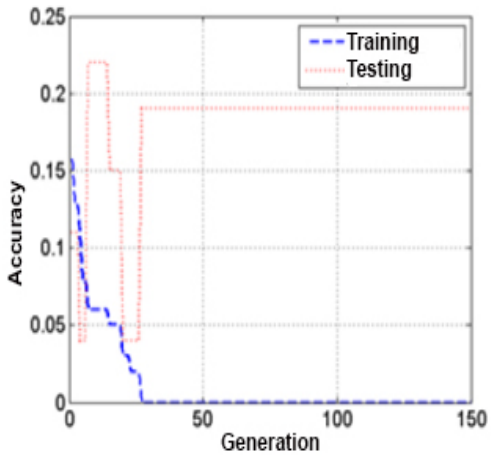


**Figure 4.** Result of the objective function cryotherapy problem

For the train problem, the results show that, on average before generation 5, a model with perfect accuracy was found for the 10 runs, so we consider as well as problems 1 and 2, it is not necessary to show the obtained graphs. Finally, for the problem of ticking, which is considered the most complex due to the number of attributes and instances, the results obtained for this problem. Given that the number of instances is greater than the previous problems, we consider test data. The results show that on average a perfect accuracy was found before generation 30 for the training data set; while with a median of 0.80 in the accuracy for the test data.

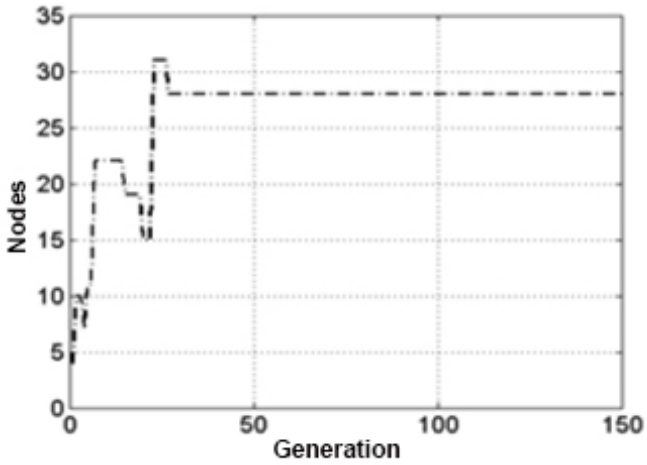


**Figure 5.** Model size cryotherapy problem

For the train problem, the results show that, on average before generation 5, a model with perfect accuracy was found for the 10 runs, so we consider as well as problems 1 and 2, it is not necessary to show the obtained graphs. Finally, for the ticking problem, which is considered the most complex due to the number of attributes and instances, Figures 5-6 show the convergence graphs of the objective function for both the training data and the test data, as well as the median of the program size. It can be seen then that the algorithm is able to find a classification model with an accuracy greater than 95%, on average, for this problem.

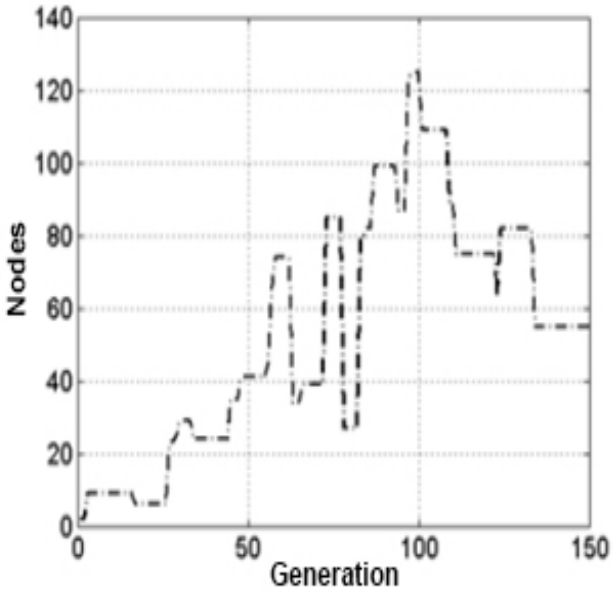


**Figure 6.** Result of the Tic Tac problem objective function

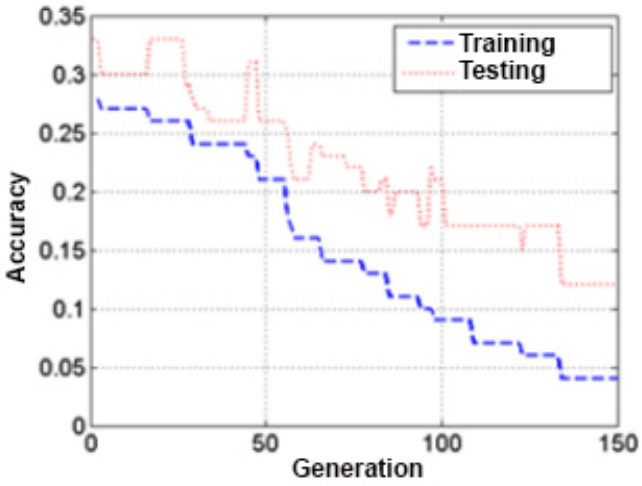


**Figure 7.**Problem model size Tic Tac

## 5. Conclusion

In this study, the implementation of genetic programming to solve binary classification problems was addressed. A unit step function was added at the tip of the tree in such a way that the result will be limited to providing a false or a true, {0,1}. It was tested in simple problems (considered so by the number of instances and attributes) where immediate results were obtained generating models with perfect fit. While the problems are relatively simple, the results demonstrate the power of PG to find models quickly and efficiently. Problem 5 was the most complex given the number of instances, however, it can be said that satisfactory results were obtained since a model with an accuracy above 95% on average was found. Although the results are good in general, there are several points that can be specifically addressed and improved; for example, the size of the models. In more complicated problems the size of the generated model can be increased too much, and it is a matter that must be dealt with since it is directly linked to the use of resources (in some real application the speed of evaluation of the model can be critical). It is recommended in the future the incorporation of some numerical method of local search, to optimize the numerical terminals and thus find trees of smaller size. Similarly, implement genetic programming in real-life problems such as a vision system with a focus on quality in a production system.

## References

[1] Santoso, L.W. (2011) 'Classifier combination for telegraphese restoration', Proceedings of the 2011 International Conference on Uncertainty Reasoning and Knowledge Engineering (URKE), Vol. 1, pp. 79-82.

[2] Banzhaf, W., Spector, L. and Sheneman, L. (2018) 'Genetic programming theory and practice XVI (Springer International Publishing, Cham).

[3] Dua, D. and Taniskidou, E. UCI Machine Learning Repository. Internet: http:// archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

[4] Espejo, P.G., Ventura, S. and Herrera, F. (2009) 'A Survey on the application of genetic programming to classification', IEEE Transactions of Systems, Man., and Cybernetics, Vol. 40, pp. 121-144.

[5] Flores, E.Z., Trujillo, L., Schütze, O. and Legrand, P. (2014) 'Evaluating the effects of local search in genetic programming', EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionay Computation V, pp. 213-228.

[6] Flores, E.Z., Trujillo, L., Schutze, O. and Legrand, P. (2015) 'A local search approach to genetic programming for binary classification', Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 1151-1158.

[7] Ghavifekr, S., Kunjappan, T., Ramasamy, L. and Anthony, A. (2016) 'Teaching and learning with ICT tools:issues and challenges from teachers' perceptions', Malaysian Online Journal of Educational Technology, Vol. 4, No. 2. pp. 38-57.

[8] Holts, A., Riquelme, C. and Alfaro, R. (2010) 'Automated text binary classification using machine learning approach', Proceedings of the 2010 XXIX International Conference of the Chilean Computer Science Society, pp. 212-217.

[9] Koza, J. (1992) 'Genetic programming: on the programming of computers by means of natural evolution', Cambridge: MIT Press, 1992.

[10] Leo, M., Sharma S., and Maddulety K. (2019) 'Machine learning in banking risk management: a literature review', Risks, Vol. 7, No. 1, pp. 29.

[11] Liu, D., Li, T. and Liang, D. (2014) 'Incorporating logistic regression to decision-theoretic rough sets for classifications' International Journal of Approximate Reasoning, Vol. 55, pp. 197-210.

[12] Mohamed, B., Issam, A., Mohamed, A. and Abdellatif, B. (2015) 'ECG image classification in real time based on the Haar-like features and artificial neural networks', Procedia Computer Science, Vol. 73, pp. 32-39.

[13] Murmu S. and Biswas, S. (2015) 'Application of fuzzy logic and neural network in crop classification', Aquatic Procedia, Vol. 4, pp. 1203-1210.

[14] Nidhin-Prabhakar, T.V.N., Xavier, G., Geetha, P. and Soman, K.P. (2015) 'Spatial preprocessing based multiomial logistic regression for hyperspectral image classification', Procedia Computer Science, Vol. 46, pp. 1817-1826.

[15] Palarivattom, S., and Kochunni, K. (2015) 'Real-time, interposable communication for web services', International Research Journal of Management, IT and Social Sciences, Vol. 2, No. 5, pp. 26-32.

[16] Parikh, K. and Shah, T.P. (2016) 'Support vector machine - a large margin classifier to diagnose skin illness', Procedia Technology, Vol. 23, pp. 369-375.

[17] Patel, J.P. and Upadhyay, S.H. (2016) 'Comparison between artificial neural network and support vector method for a fault diagnosis in rolling element bearings', Procedia Engineering, Vol. 144, pp. 390-397.

[18] Poli, R., Langdon, W.B. and McPhee, N.F. (2008) 'A field guide to genetic programming', San Francisco: Lulu Enterprises.

[19] Santoso, L.W. (2011) 'Classifier combination for telegraphese restoration', Proceedings of the 2011 International Conference on Uncertainty Reasoning and

Knowledge Engineering (URKE), Vol. 1, pp. 79-82.
[20] Santoso, L.W. (2019) 'Cloud technology: opportunities for cybercriminals and security issues', Proceedings of the 2019 12th International Conference on Ubi-Media Computing.
[21] Santoso, L.W. and Yulia. (2014) 'Analysis of the impact of information technology investments-a survey of Indonesian universities' ARPN Journal of Engineering and Applied Sciences, Vol. 9, No. 12, pp. 2404-2410.
[22] Xavier, I.M.D.D.G. (2015) 'Email issue for working at information technology field', International Research Journal of Management, IT and Social Sciences, Vol. 2, No. 5, pp. 1-5.
[23] Yulita, I.N., Fanany, M.I. and Arymuthy, A.M. (2017) 'Bi-directional long short-term memory using quantized data of deep belief networks for sleep stage classification', Procedia Computer Science. Vol. 116, pp. 530-538.