AEGADT: Application and Evaluation of Genetic Algorithm on Decision Trees

Francesca La Manna & Gioacchino Caliendo

Academic Year 2021/2022

Abstract. The project at the centre of this paper aims to apply and evaluate the evolution of Decision Trees by means of a Genetic Algorithm. The objectives of the project emphasise the desire to compare the results of different runs of equivalent models, obtained through a batch methodology, to the chosen Genetic Algorithm. Identifying the rate of improvement of decision trees and the number of generations required to obtain the maximum number of correct predictions is an interesting step that this project aims to achieve. It is also important to make observations on the application of the two techniques, considering various factors on the results obtained in the various runs.

---

1 Overview

1.1 Machine Learning

Machine Learning (ML) is a branch within the wider world of Artificial Intelligence and aims to make machines learn, in an automatic way, activities carried out by us humans. The term Machine Learning was first coined in 1959 by Arthur Samuel and later taken up by Tom Mitchell who gave it a formal and current definition [1]:

> *A program is said to learn from some experience E with respect to a class of tasks T by achieving a performance P, if its performance in accomplishing tasks T, as measured by performance P, improves with experience E."*

Machine Learning has the objective of creating models that allow the construction of learning algorithms to solve a specific problem. The learning model indicates the purpose of the analysis, i.e. how you want the algorithm to learn. There are various models of learning but we considered in this study only those related with *Supervised Learning* that is the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. Learning stops when the algorithm reaches an acceptable level of performance. Supervised learning is when you have input data (X) and output data (Y) and use an algorithm that learns the function that generates the output from the input. The objective is to approximate the function so that when a new input data (X) is available, the algorithm can predict the generated output value (Y) for that data. In this context, one of Supervised learning problems is called *Classification* that is the process in which a machine is able to recognise and categorise dimensional objects from a dataset.

1.2 Decision Tree

Regarding what has been explained in the section 1.1 of the document, one of the most widely used algorithms is called Decision Tree. A Decision Tree is a machine learning algorithm that partitions the data into subsets. The partitioning process starts with a binary split and continues until no further splits can be made. Various branches of variable length are formed. The goal of a decision tree is to encapsulate the training data in the smallest possible tree. The rationale for minimizing the tree size is the logical rule that the simplest possible explanation for a set of phenomena is preferred over other explanations. Also, small trees produce decisions faster than large trees, and they are much easier to look at and understand. There are various methods and techniques to control the depth, or prune, of the tree [2].

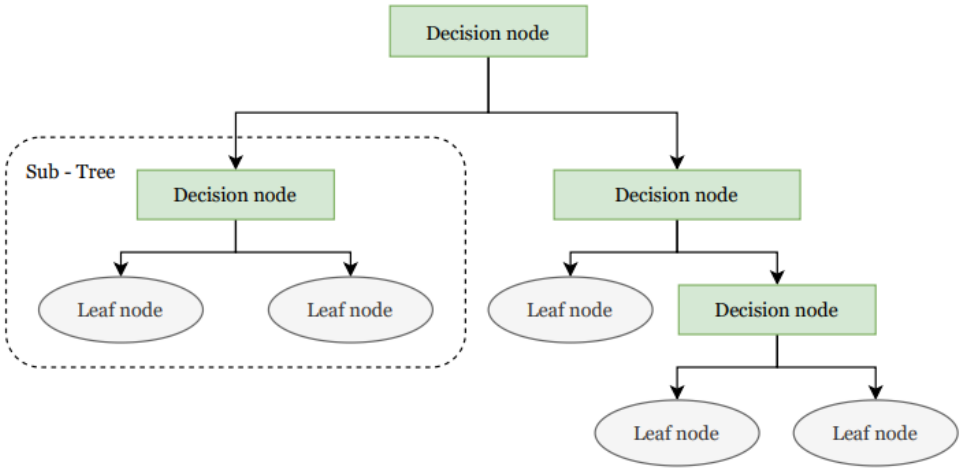

Figure 1: Decision Tree Classification Algorithm scheme

1.3 Genetic Algorithms

Any Traditional Machine Learning Algorithm learn from the data, where choice of algorithm and features (inputs) to be fed into algorithm are made by subject matter experts. Traditional ML models expects all inputs to in the format of structured data like numbers and can be used to solve classification, regression, clustering, dimensionality reduction problems [3]. In contrast to what is described for Traditional Machine Learning Algorithms, there are algorithms called Genetic Algorithms, that refer a search-based algorithms used for solving optimization problems in machine learning. This class of algorithms is important because it solves difficult problems that would take a long time to solve. A genetic algorithm (GA) is a meta-heuristic search algorithm used to solve search and optimization problems. This class of algorithms is a subset of Evolutionary Algorithms, which are used in computation. Genetic Algorithms employ the concept of genetics and natural selection to provide solutions to problems. These algorithms have better intelligence than random search algorithms because they use historical data to take the search to the best performing region within the solution space [4]. In this context, a *Population* is a subset of all the probable solutions that can solve the given problem. A *Chromosome* is one of the solutions in the population and a *Gene* is an element in a chromosome. An *Allele* is the value given to a gene in a specific chromosome. Moreover, a *Fitness function* is a function is based on the objective function to be optimised (minimised or maximised) and give the probability that a certain individual will survive in the next generation. In addition, in *Genetic Operators* the best individuals mate to reproduce an offspring that is better than the parents. Genetic operators are used for changing the genetic composition of this next generation.

1.3.1 Genetic Algorithms Workflow

Genetic algorithms use the evolutionary generational cycle to produce high-quality solutions. They use various operations that increase or replace the population to provide an improved fit solution [4]. Genetic algorithms follow the following phases to solve complex optimization problems:

- *Initialization*: The genetic algorithm starts by generating an initial population. This initial population consists of some random possible solutions to the given problem.

- *Fitness assignment*: The fitness function helps in establishing the fitness of all individuals in the population. It assigns a fitness score to every individual, which further determines the probability of being chosen for reproduction. The higher the fitness score, the higher the chances of being chosen for reproduction.

- *Selection*: In this phase, individuals are selected for the reproduction of offspring. The selected individuals are then arranged in pairs of two to enhance reproduction. These individuals pass on their genes to the next generation. The main objective of this phase is to establish the region with high chances of generating the best solution to the problem (better than the previous generation).

- *Reproduction*: This phase involves the creation of a child population. The algorithm employs variation operators that are applied to the parent population. The two main operators in this phase include crossover and mutation.

    1. *Crossover*: This operator swaps the genetic information of two parents to reproduce an offspring. It is performed on parent pairs that are selected randomly to generate a child population of equal size as the parent population.
    2. *Mutation*: This operator adds new genetic information to the new child population. This is achieved by flipping some genes in the chromosome.

- *Replacement*: Generational replacement takes place in this phase, which is a replacement of the old population with the new child population. The new population consists of higher fitness scores than the old population, which is an indication that an improved solution has been generated.

- *Termination*: After replacement has been done, a stopping criterion is used to provide the basis for termination. The algorithm will terminate when reaches the stopping criterion chosen.

1.4 CART: Classification And Regression Tree

It was decided to use the *Classification And Regression Tree* (CART) Algorithm for the training of the Tree [8]. This class of Algorithms is a classification algorithm for building a Decision Tree based on Gini's Impurity index as splitting criterion. CART is a binary tree build by splitting node into two child nodes repeatedly. The Algorithm works repeatedly in three steps:

1. Find each feature's best split. For each feature with K different values there exist K-1 possible splits. Find the split, which maximizes the splitting criterion. The resulting set of splits contains best splits (one for each feature).

2. Find the node's best split. Among the best splits from Step i find the one, which maximizes the splitting criterion.

3. Split the node using best node split from Step ii and repeat from Step i until stopping criterion is satisfied.

As splitting criterion we used Gini's impurity index, which is defined for *node* t as:

$$i(t) = \sum_{i,j} C(i|j)p(i|t)p(j|t)$$

where $C(i|j)$ is a cost of misclassifying a class j case as a class i case (in our case $C(i|j) = 1$ if $i \neq j$ and $C(i|j) = 0$ if $i = j$), $p(i|t)$ ($p(j|t)$ respectively) is is probability of case in class $i(j)$ given that falls into *node* t.
The Gini Impurity criterion is type of decrease of impurity, which is defined as:

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R),$$

where $\Delta i(s,t)$ is decrease of impurity at *node* t with split s, p_L (p_R) are probabilities of sending case to the left (right) child node t_L (t_R) and $i(t_L)$ $i(t_R)$ is Gini Impurity measure for left (right) child node.

2 Objectives

The focus of this document is on the application and evaluation of Genetic Algorithms on Decision Trees. In order to achieve this goal, we propose:

1. The implementation of a Genetic Algorithm capable of operating on Decision Trees;
2. The application of the Genetic Algorithm using different configurations;
3. The construction of Decision Trees by means of batch learning techniques based on the same configurations;
4. The comparison of the results obtained between equivalent models.

In this document we propose to evaluate the functioning of a Genetic Algorithm in training populations of Decision Trees in order to analyse the results obtained by having one Decision Tree constructed using a classical training technique called CART (Classification And Regression Trees). In this regard, it becomes interesting to empirically identify the rate of improvement of Decision Trees and the number of generations required to reach the highest number of correct predictions made. In order to assess the impact of the data on the construction of Decision Trees, it is proposed to repeat the experiment on two Datasets.

3 Proposed Methodology

In order to produce a Genetic Algorithm (discussed in section 1.3) capable of working on Decision Trees (discussed in section 1.2), it was necessary to construct a structure capable of representing them.

3.1 Individual structure

The individual is essentially a complete binary Decision Tree. Each node in the tree is given by the combination of feature, value and class. In addition, it was decided to implement the tree using an array because of the simplicity on representing binary tree through this structure. In order to prevent the tree from growing too large, a value has been introduced to limit its depth. Each node of the tree (consisting of feature and value if it is an internal node or class if it is an leaf node) will therefore have 2 children or 0 if it is a leaf. At this point, if a *node* i is not a leaf, then it will have a left child detectable at position $2i+1$ and a right child detectable at position $2i+2$.

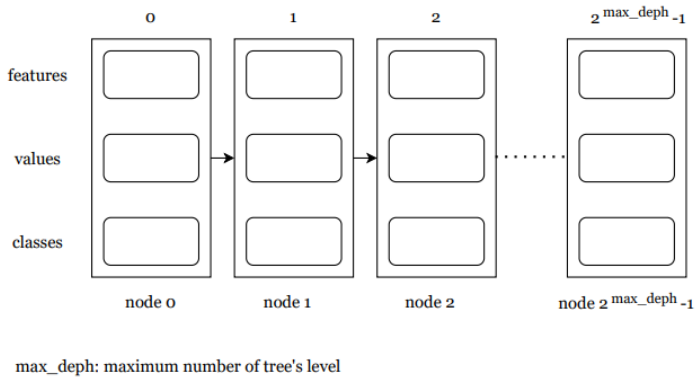

Figure 2: Structure of Decision Tree Classifier

Basically:

- the features will be composed of integers indicating the number of the column of the dataset to be considered. Features are selected randomly within the domain representing the available features.

- the values will be composed of values selected causally in the feature domain of the corresponding node. The relational operator used to split nodes is always "<=".

- the classes will be composed of values taken from the column indicating the classes of the dataset. The selection of classes is done causally within the domain of the available classes. The classes will be present only on leaf nodes, there will be no features and values but only the class.

3.2 Implementation

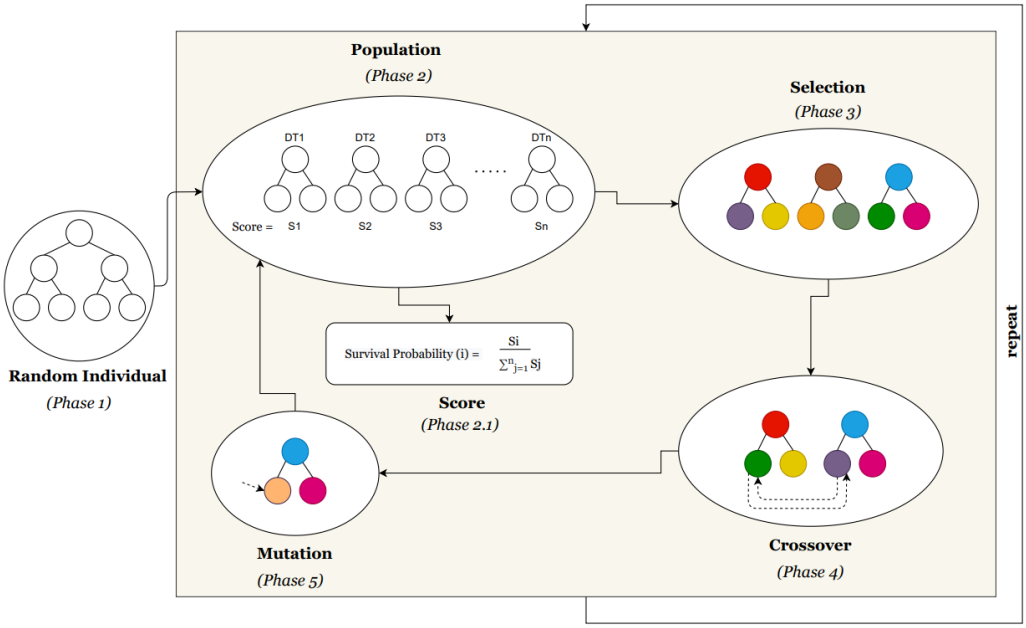

Figure 3: Schematic representation of Genetic Algorithm

Initially we create $k = 100$ decision trees as **Random Individuals** *(Phase 1)*. This individuals represent an initial **Population** *(Phase 2)*. Then, we proceed to **Score** *(Phase 2.1)* all the single individuals using the following metrics:

- *Precision*: The precision is the ratio [6]:

$$tp/(tp + fp)$$

where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The best value is 1 and the worst value is 0.

- *Recall*: The recall is the ratio [7]:

$$tp/(tp + fn)$$

where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples. The best value is 1 and the worst value is 0.

- *F-measure*: it also known as balanced F-score or F-measure [5]. The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$F1 = 2 \cdot (precision \cdot recall)/(precision + recall)$$

The evaluation of the population of individuals is calculated through the f-measure calculated for each individual assigning to each of them a probability of survival directly proportional to the f-measure. Having reached this point it becomes important to make a **Selection** *(Phase 3)* of the individuals capable of giving birth to the next generation. In this phase the *Roulette Wheel Selection* is used. This method is a stochastic selection method, where the probability for selection of an individual is proportional to its fitness. The method is inspired by real-world roulettes but possesses important distinctions from them. As we know from the movies on casinos and gamblings, roulettes always have slots with the same size. That means, however, that all slots have the same probability of being selected. Instead, we can implement a weighted version of the roulette. With it, the larger the fitness of an individual is, the more likely is its selection. In a population with n individuals, for each chromosome x with a corresponding fitness value f_x, we compute

the corresponding probability p_x of selection as:

$$p_x = \frac{f_x}{\sum_{i=1}^{n} f_i}$$

We can then simply treat the probability distribution

$$P = \{p_1, p_2, ..., p_n\}$$

as the one from which we sample the parents we choose for recombination.

In order to be able to produce the next generation, we have to make a **Crossover** *(Phase 4)* between the couple of individuals that have been selected. It was decided to proceed with the use of the *One Point Crossover*, thanks to which it is possible to breed individuals. In this method one arbitrary combination point is selected for both parents' chromosomes. The chromosomic section after these combination points are swapped with each other, giving birth to two new offspring. It was decided to implement the One Point Crossover method as follows:

- Of the surviving population, two trees are randomly selected;
- For each of them, the tree built on the left son of the root is selected;
- Then, the selected subtrees are swapped with each other.

In order to maintain diversity in the population the key idea is to produce a **Mutation** *(Phase 5)* into the offspring by using *Random Resetting*. In this method, a random value from the set of permissible values is assigned to a randomly chosen gene. The Random Resetting method is composed by different steps:

- Definition of the probability of chromosome mutation: At this phase a probability $p = 0.15$ of developing a mutation was chosen for each chromosome. Individuals selected at this stage will proceed to the next step.
- Definition of the probability of mutation of genes: For each chromosome that has reached this phase, a mutation probability is generated for each of its genes. A mutation probability of $p = 0.15$ was again chosen for each gene. Genes selected at this phase will move on to the next step.

- Mutation: At this point, if the gene that was able to pass both previous steps corresponds to a leaf in the decision tree, then a new random label is generated from among those in the dataset. Otherwise, a new feature is selected randomly and a new value for that node obtained in the space of possible values for the feature in question.

The steps between the evaluation of the population of individuals and the use of Random Resetting method are repeated until the stop criterion is reached. In this case, the stopping criterion is represented by the number of generations. At this point, it is substantial to analyse the performance of the best individuals obtained in all the generations performed in order to compare them with their own equivalent batch model.

3.3 Configurations

Various configurations on the execution of the genetic algorithm were applied to the proposed methodology (discussed in section 3). In order to evaluate the functioning of the Genetic Algorithm, it was decided to reapply the chosen configurations on more than one dataset. The selected datasets are: *Wine Data Set* [9] and *Iris Data Set* [10]. For each of them, the configurations made are based on the following parameters: *max_depth*[1], and *individuals_selected* [2]. The number of individuals per generation and the number of tree populations that the algorithm will generate before terminating have remained unchanged for all configurations because our intention was to identify the upward trend (if it exists).

In particular:

Configurations	*max_depth*	*individuals_selected*
1	4	20
2	5	20
3	7	20
4	10	20
5	4	10

Table 1: Configurations used for all Datasets chosen

[1] *max_depth*: the maximum length of a path connecting the root to a tree leaf

[2] *individuals_selected*: the number of survival individuals who will give birth to the next generation

4 Comparison of results obtained

4.1 Results for the Genetic training Algorithm

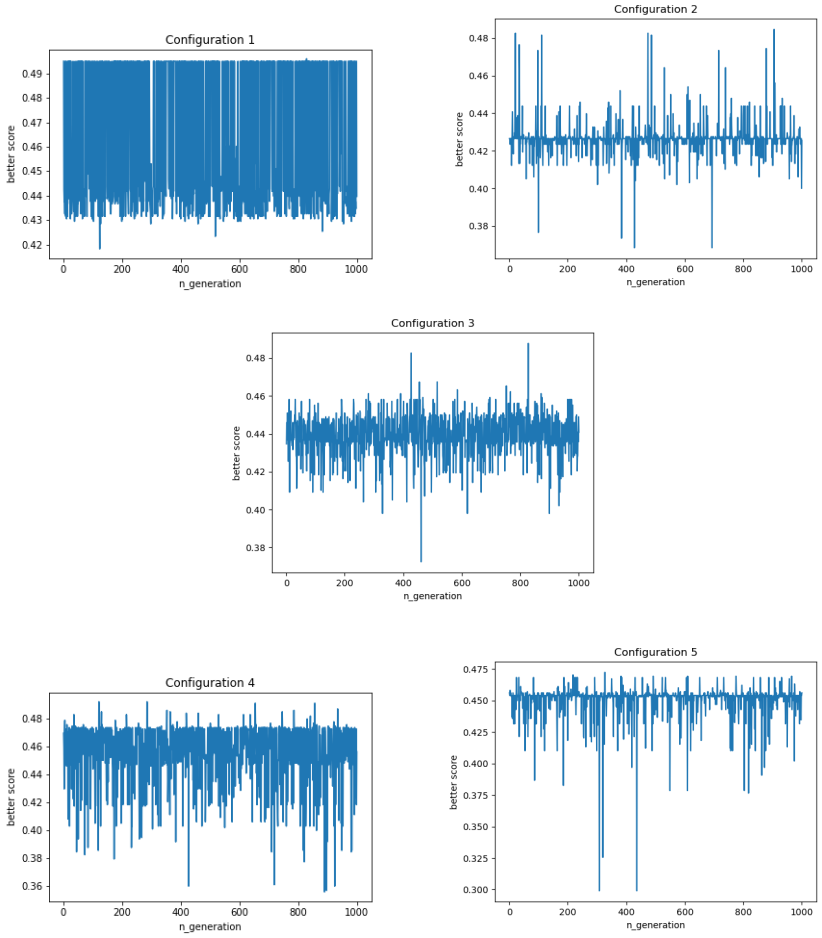

Figure 4: Results of Genetic Algorithm on Wine Data Set [9]

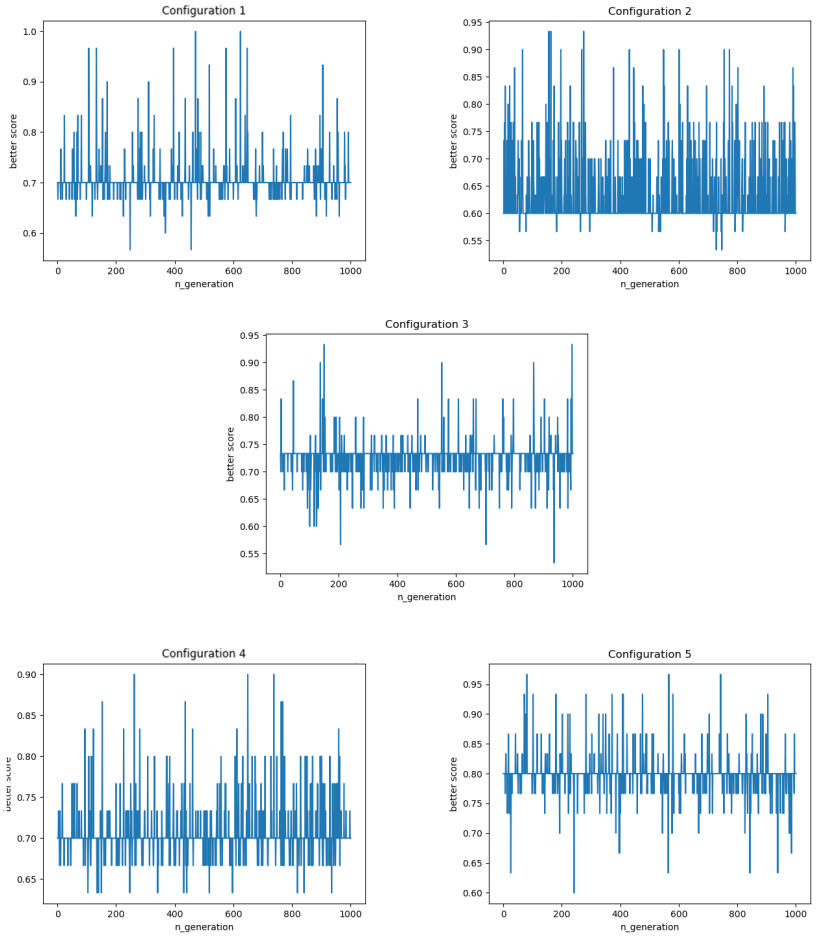

Figure 5: Results of Genetic Algorithm on Iris Data Set [10]

The graphs in Figure 4 and Figure 5 represent the best results obtained through generations on all configurations (listed in detail in the section 3.3). In detail, on the abscissa are represented the generations while on the ordinates are represented the scores.

4.2 Results for the Traditional training Algorithm

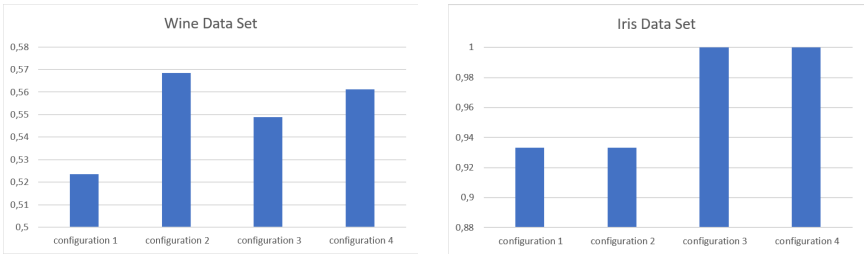

Figure 6: Results of the Training Algorithm on Datasets chosen

The procedure used for the evaluation of the CART Algorithm (batch methodology) consists of two steps:

1. Generation and evaluation of 1000 individuals;
2. Selection of the best individual.

The configurations (discussed in section 3.3) used for the CART Algorithm are equivalent to those used for the Genetic Algorithm. However, Configuration 1 and Configuration 5 of the Genetic Algorithm were both compared with Configuration 1 for the CART Algorithm because Configuration 1 and Configuration 5 differ in the parameter *individuals_selected*, which is a value not present in the concept of Decision Tree generation with batch algorithms.

5 Conclusions

As shown in section 4.1 and in section 4.2, contrary to expectations, there was no real upward trend in Figure 4 and Figure 5. However, for each of them, an almost linear trend was found from the earliest generations with sporadic and apparently random peaks of growth or decrease. On the other hand, the results obtained in Figure 6 using Traditional Training are greater in the best case and equivalent in the worst case than the results obtained using Genetic Algorithm traning. It is also important to consider the time taken to achieve the results described for both training methods. In this regard, the execution of the Genetic Algorithm is slower than the execution of the Traditional Algorithm without providing better results in return.

References

[1] Article of Humanativa Group,*Machine Learning: come scegliere il modello migliore*.

[2] WebFOCUS 8 Technical Library,*Explanation of the Decision Tree Model*.

[3] Sai,*Traditional and Representational Machine Learning*, May 18, 2020.

[4] Arthur Muthee,*The Basics of Genetic Algorithms in Machine Learning*, May 26, 2021.

[5] Scikit-learn, *sklearn.metrics.f1_score*.

[6] Scikit-learn, *sklearn.metrics.precision_score*.

[7] Scikit-learn, *sklearn.metrics.recall_score*.

[8] Breiman L, *Classification and regression trees*, The Wadsworth and Brooks-Cole statisticsprobability series.

[9] Machine Learning Repository, *Wine Data Set*.

[10] Machine Learning Repository, *Iris Data Set*.

The entire project is available at the following link:
https://github.com/LamFra/AEGADT.git