Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 7, No. 4, August, 2023

Advanced Genetic Programming vs. State-of-the-Art AutoML in Imbalanced Binary Classification

Franz Frank [1*], Fernando Bacao [1]

[1] *NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal.*

Abstract

The objective of this article is to provide a comparative analysis of two novel genetic programming (GP) techniques, differentiable Cartesian genetic programming for artificial neural networks (DCGPANN) and geometric semantic genetic programming (GSGP), with state-of-the-art automated machine learning (AutoML) tools, namely Auto-Keras, Auto-PyTorch and Auto-Sklearn. While all these techniques are compared to several baseline algorithms upon their introduction, research still lacks direct comparisons between them, especially of the GP approaches with state-of-the-art AutoML. This study intends to fill this gap in order to analyze the true potential of GP for AutoML. The performances of the different tools are assessed by applying them to 20 benchmark datasets of the imbalanced binary classification field, thus an area that is a frequent and challenging problem. The tools are compared across the four categories average performance, maximum performance, standard deviation within performance, and generalization ability, whereby the metrics F1-score, G-mean, and AUC are used for evaluation. The analysis finds that the GP techniques, while unable to completely outperform state-of-the-art AutoML, are indeed already a very competitive alternative. Therefore, these advanced GP tools prove that they are able to provide a new and promising approach for practitioners developing machine learning (ML) models.

Keywords:

Genetic Programming;
Automated Machine Learning;
AutoML;
Imbalanced Binary Classification.

Article History:

Received:	06	February	2023
Revised:	11	April	2023
Accepted:	15	June	2023
Available online:	12	July	2023

1- Introduction

A hot topic within machine learning (ML) that has recently been spreading rapidly in both the research as well as industry domains is automated machine learning (AutoML). The partially or fully automated application of ML not only supports the work of domain experts but, by allowing the automation of entire pipelines, also addresses the problem of the lack of such experts, as it makes ML accessible to non-experts as well [1]. The range of tasks to which AutoML can be applied is very broad, ranging from processing simple baseline ML models up to complex artificial neural networks (ANNs) in deep learning (DL), with the number of upcoming AutoML methods and concepts continuing to rise [2]. Among the latest trends to be observed is the application of genetic programming (GP) as an optimization method in AutoML [3, 4].

Two novel advanced techniques from the field of genetic programming (GP) are differentiable Cartesian genetic programming of artificial neural networks, abbreviated as DCGPANN [5], and geometric semantic genetic programming, abbreviated as GSGP [6], which have both already been shown to be able to achieve promising results, for example when compared to a variety of traditional ML algorithms [7]. However, research lacks evaluations of these emerging GP approaches alongside state-of-the-art AutoML tools. While a rising number of studies have compared various AutoML tools with each other [2, 8, 9], there is a gap when it comes to direct comparisons of these with GP. Therefore, this paper intends to fill this gap by comparing the two aforementioned advanced GP techniques with three

* **CONTACT**: m20200618@novaims.unl.pt

DOI: http://dx.doi.org/10.28991/ESJ-2023-07-04-021

of the most popular state-of-the-art AutoML tools, Auto-Keras [10], Auto-PyTorch [11], or Auto-Sklearn [12], by applying them to 20 benchmark datasets and thereby investigating their respective capabilities.

The subfield of supervised learning on which this work focuses is imbalanced binary classification, i.e., learning from a dataset with two classes where the number of instances of one class is considerably higher than the number of instances of the other. This aspect tends to be a frequent and challenging problem when applying ML; thus, GP and AutoML approaches should be capable of overcoming it properly. Conventional ML approaches, in which no action is taken regarding the imbalance of the data, typically return a biased model when dealing with an imbalanced binary classification problem since the model is trained using far more instances of the majority class, which results in it favoring that class in predictions [13]. Therefore, the research community is continuously striving to discover and develop improved solutions to this problem [13–15].

Consequently, the central research question of this research paper is whether the two novel GP techniques, DCGPANN and GSGP, are able to compete with the three state-of-the-art AutoML tools, Auto-Keras, Auto-PyTorch, and Auto-Sklearn, when applied to imbalanced binary classification, and which of the five methods achieves the best overall performance on this task. To answer this question, we primarily evaluate the average as well as the maximum achieved scores in terms of three common metrics for imbalanced binary classification, F1-score, the geometric mean (G-mean), as well as the area under the receiver operating characteristics (ROC) curve (AUC), of the methods across 20 benchmark datasets. In addition, the individual techniques are investigated regarding the consistency of their performance as well as their generalization ability. Since the core mission of AutoML is to facilitate the use of ML [16], the tests are performed under simple setups without requiring additional knowledge of specific topics, such as hyperparameter optimization or sampling of imbalanced data, so that the results depend mainly on the capabilities of the five techniques themselves rather than on external factors.

The remainder of this paper is structured as follows: Section 2 refers to existing literature on the topic and provides the theoretical foundations. Subsequently, the research methodology in Section 3 outlines the study design and presents the benchmark datasets, evaluation metrics, and tool setups. Section 4 analyzes the results, followed by a debate of all findings and their meanings in the discussion. Finally, the article is wrapped up by the conclusion provided in Section 5.

2- Related Literature and Concepts

2-1- Automated Machine Learning

The term AutoML stands for the automated application of ML. AutoML has recently gained much popularity [17–19] because it allows the automation of certain parts of ML, thus eliminating the need for a human expert for those specific operations. However, human involvement is still required to a certain extent in order to successfully solve real-world tasks using AutoML [16]. As a result, AutoML can be considered a useful tool for data scientists rather than some sort of competitor, while it can also provide access to ML for non-experts.

The various AutoML tools can often be distinguished by two main characteristics: the search space used by a particular tool, for example, for finding a model and its corresponding best-performing hyperparameters, and the optimization method used to navigate through this search space [2]. The search space is closely related to the task of a specific tool; e.g., the search space of an AutoML tool designed for the optimization of ANNs typically consists of all possible architectures of eventual ANNs, usually within certain defined boundaries [10]. Therefore, there are huge differences in the respective search spaces, depending on the specific purpose of a particular tool. Likewise, there are various different optimization methods used by existing AutoML tools. Essentially any type of optimization can be employed, with some of the most common being grid and random search, gradient descent, reinforcement learning, surrogate model-based optimization, or evolutionary algorithms such as genetic algorithms (GAs) or GP [1]. The application areas of AutoML tools in ML are very diverse, with, among others, especially the usage of AutoML in DL having recently seen a rapid increase in interest [11].

Three of the most popular AutoML tools for regression as well as classification tasks are Auto-Keras [10], Auto-PyTorch [11], and Auto-Sklearn [12, 20]. Auto-Keras is able to automatically create and optimize ANNs using a Bayesian optimization approach [10]. Auto-PyTorch and Auto-Sklearn, on the other hand, draw from a pool of baseline ML models and exploit the benefits of meta-learning, Bayesian optimization, and ensemble construction in order to find the best possible final model [11, 12, 20]. However, the latest versions of Auto-PyTorch can also be used for ANN optimization, but for this work, the version that uses baseline models is chosen.

2-2- Genetic Programming

2-2-1- Standard Genetic Programming

Genetic Programming was first introduced by Koza [21]. He describes GP as “a way to find a computer program of unspecified size and shape to solve, or approximately solve, a problem” [21]. Similar to GAs, GP is an evolutionary method, and its basic idea is to evolve a set of solutions based on Darwin’s theory of evolution [22]. However, one of

the most important differences between GP and GAs is the representation of the solutions. While with the application of GAs, usually strings with a fixed length are evolved, in GP, these solutions are dynamic computer programs, typically represented as LISP-like trees [23]. To enable the GP algorithm to develop these solutions, a set of functions and a set of terminals have to be defined in advance, on the basis of which the individual solution trees can be expressed [21]. The set of functions can, for example, include, among others, mathematical functions, arithmetic operations, or conditional logical operations, while examples for the terminal symbols would be a variety of numerical constants [21, 23]. However, both the predefined set of functions as well as the set of terminals are highly dependent on the underlying problem. The solution space in GP consists of every possible composition of the possible functions and terminals [21].

Traditional GP has certain limitations. According to Vanneschi et al. [23], two of the main issues that come with GP are its heavy time consumption and the processes of mutation and crossover. The first problem of time consumption arises because each solution within the population must be evaluated, which depending on the problem at hand, often requires a highly extensive computation procedure. In this context, time consumption often correlates with certain parameters, such as tree size in particular, whereby an increasing value does not often lead to the corresponding desired better fitness [23]. The second problem addresses the fact that by applying the standard genetic operators for GP, only virtually blind syntax transformations occur in the respective solutions. This means that no knowledge about the resulting behavior of a solution after a transformation is taken into account [23, 24]. Thus, Traditional GP entirely ignores the meanings of the computer programs being evolved, the so-called semantics [24].

2-2-2- Geometric Semantic Genetic Programming

Addressing the aforementioned issue of the ignorance of semantics in traditional GP, Moraglio et al. [24] introduced an advanced version of GP called geometric semantic genetic programming (GSGP). They refer to the fact that the success of a computer program, and thus of a solution in the population of a GP algorithm, is dependent on the semantics of the solution, which is why these semantics should not be disregarded. There are numerous ways to formally define the semantics of a solution. They can be defined as simply the fitness of the solution, the canonical representation, the mathematical function of the program, or as a logical formalism describing the program's behavior [24]. Moreover, semantics in GSGP can, in this context, be defined as the vector of the output values of a possible solution within the population computed during the training phase [25]. According to this particular definition, a specific solution within the solution space associated with a GP algorithm is "a point in a multidimensional space called semantic space, where the dimensionality is equal to the number of observations in the training set" [26]. In accordance with this, Figure 1 shows an example of a representation of solutions on the left side in the genotypic space and the right side with the corresponding points in a simple exemplary two-dimensional semantic space.

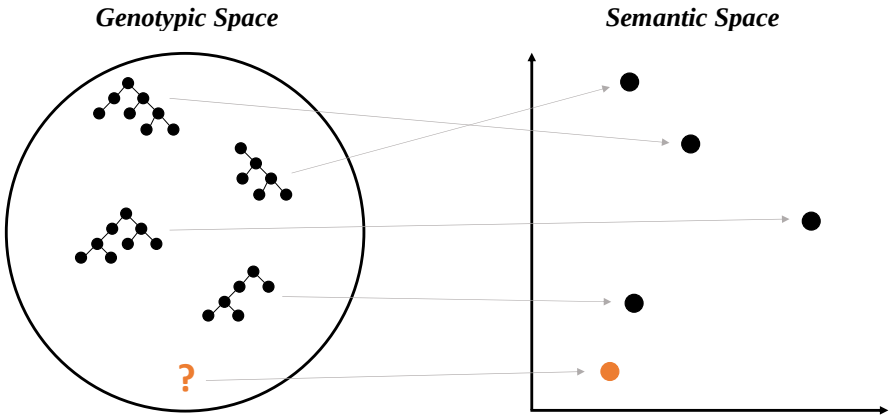


Figure 1. Representation of solutions in the genotypic as well as the semantic space

The introduction of the semantic space enables the global optimum to be approached in a stepwise manner during the execution of a GP algorithm while, contrary to traditional GP, taking the meaning of the individual solutions into account [24, 27]. For this purpose, however, new methods for crossover and mutation are necessary, the so-called geometric semantic operators, geometric semantic crossover, and geometric semantic mutation [24].

The aim of the introduction of the geometric semantic crossover is to perform a crossover of the genotypes of two solutions in such a way that it is possible to know the effect of the crossover on the offspring in the semantic space, and thus, on the phenotype, in order to approach the global optimum [24, 28]. More precisely, the introduced geometric semantic crossover is performed in a way that any possible resulting offspring must be located between its parents in the semantic space; thus, the semantics of the offspring are a linear combination of the semantic vectors of its parents [24, 27]. The consequence of this is that the fitness of the offspring must be at least as good as the worst fitness of its parents,

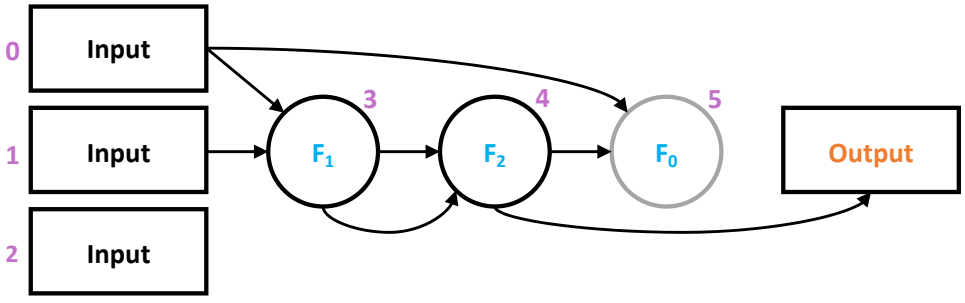
which means that there is no possibility for the crossover to produce worse solutions in terms of fitness, which in turn implies that the movement in the semantic space can never be away from the global optimum [24]. A formal definition of geometric semantic crossover is given by Moraglio et al. [24], who prove that a crossover based on the given definition always returns an offspring as a linear combination of its parents in the semantic space. In the actual process of creating the new offspring, the effect of the described crossover operator on a particular entry of the semantic vector will first be observed and then used to describe the associated genotypic change required to produce that particular semantic modification. This description will then be used as guidance for the production of the offspring, whereby it must be noted that this means that the syntax of the offspring must also contain both of its parents, which means that the size of the solutions in this form of GSGP increases exponentially with each generation [24].

In addition to geometric semantic crossover, a geometric semantic mutation method is also necessary for GSGP. Also, in this respect, the effect of the genetic operator in the semantic space is of essential importance. In this context, the aim of geometric semantic mutation is to create a unimodal fitness landscape [24, 27]. This is achieved by means of a box mutation, whereby the solution emerging after the mutation is only able to range within certain boundaries around the initial solution in the semantic space [27]. A formal explanation of geometric semantic mutation is given by Moraglio et al. [24], who prove that a mutation method based on their definition results in the desired box mutation within the semantic space. Vanneschi et al. [27] state that every element of the resulting semantic vector in this form of mutation represents a rather weak perturbation of the associated element of the semantic vector of the parent. While the magnitude of the mutation's effect can be changed by modifying the mutation step, the perturbation can be described as weak as the random expression it results from is centered around zero [27].

The initial design of GSGP could only be applied to regression tasks. Therefore, Bakurov et al. [6] have recently developed it further so that it can also be used for classification problems. The general concept is essentially the same, with the key modification being fairly straightforward: the classification task is treated as a regression problem, with the only possible target values being 0 and 1, which represent the two classes of the underlying problem [6]. To accomplish this, a logistic activation function is used to transform the actual output of the GSGP solution into one of the two binary numbers and the root mean square error is utilized as the fitness function in this case [6, 7].

2-2-3- Cartesian Genetic Programming

In GSGP, computer programs are evolved as tree-like structures, while Cartesian genetic programming (CGP) programs are encoded as graph-based structures [29]. Here, the name Cartesian GP comes from the fact that these structures are indexed typically by their respective Cartesian coordinates [30]. In this context, a chromosome in CGP is represented as an "encoding of a graph as a string of integers that represent the functions and connections between graph nodes and program inputs and outputs" [31]. Turner & Miller [29] thereby distinguish between three different types of chromosome genes: First, there are function genes, which define the functionality of a particular node within the graph. Then, there are genes that specify where the input for a particular node originates from, the so-called connection genes. Finally, the output genes indicate what is used as program output, which can be internal nodes as well as any input. The formal definition of the chromosome composed of these different types of genes is given by Turner & Miller [29] as follows: "A generic (one row) CGP chromosome is… [defined as] $F_0\ C_{0,0} \ldots C_{0,\alpha}\ F_1\ C_{1,0} \ldots C_{1,\alpha} \ldots F_n\ C_{n,0} \ldots C_{n,\alpha}\ O_0 \ldots O_m$ …where α is the arity of each node, n is the number of nodes and m is the number of program outputs", and F, C, and O stand for function, connection, and output genes respectively [29]. For illustration, Figure 2 provides an example of a single CGP program as both the graph and the corresponding numerical chromosome. It can be obtained that all three nodes of the CGP individual are connected to either inputs or previous nodes, or both. Furthermore, it can be seen that one input is not used at all while another one is used multiple times, which can be the case in this form of GP. Additionally, in this example, only two of the three nodes are actually contributing to the output, which is also possible to occur in CGP.



corresponding chromosome: 101 233 004 4

Figure 2. Example of a CGP program represented as a graph as well as a chromosome (adapted from [29])

Märtens & Izzo [5] proposed the concept of encoding and training ANNs using a differentiable form of CGP, which they refer to as DCGPANN. The term differentiable here implies that the information about the derivatives of the program output with regard to the respective input as well as the weights is used for learning weights and biases while training the network. This is equivalent to the backpropagation procedure in ANNs. During the training phase of DCGPANN, ANNs are modified by means of a mutation of the activation functions and the neural connections [5]. The DCGPANN module developed by Izzo et al. [32] can be applied to both classification as well as regression tasks.

3- Research Methodology

The performance of the two GP approaches and the three state-of-the-art AutoML modules is evaluated through a comparative analysis of the five techniques on 20 different ML benchmark datasets for binary classification. In order to put the results into perspective, the mentioned tasks are also solved using simple logistic regression, a typical baseline model for classification. First, the performance data of the individual tools has to be collected. For this purpose and in order to obtain statistically representative results, each technique is applied 30 independent times to each dataset. This number has also been used in similar studies [6, 7] to ensure statistical relevance. Every single execution is strictly limited to 300 seconds, as no improvement occurs for any of the tools after this period of time, which creates equal conditions. The performance is measured by three common metrics for imbalanced binary classification, namely the F1-score, the G-mean, and the AUC. Once the data collection is complete, the data is analyzed according to four different categories: average score, maximum score, standard deviation, and generalization ability. For the first category, the average of the metric values achieved after the 30 executions is crucial; for the second category, the respective maximum value achieved; and for the third category, the standard deviation of the 30 values. In the fourth category, the average generalization ability is analyzed after the 30 executions. For this purpose, the test score is divided by the train score for each execution and metric in order to see what percentage of the train score the respective model is able to achieve on the test data. Subsequently, for each category, the tools are ranked per dataset according to their scores, with rank 1 being the best and rank 6 consequently being the worst rank. The results are then statistically investigated using two different tests. First, the Friedman [33] test checks whether there are general differences in the performance of the six approaches, followed by pairwise Wilcoxon rank-sum tests [34] to determine whether there are differences between each of the respective pairs of tools. The findings regarding the average and maximum score categories are then used to identify which tools are better or worse suited for imbalanced binary classification, while the standard deviation and generalization ability categories aim to reveal further, more specific properties of the techniques.

3-1-Application to Benchmark Datasets

As usual, in these kinds of comparisons of various techniques, a variety of different datasets are needed in order to obtain meaningful results. Therefore, 20 different benchmark datasets are used for this work, which is a number similar to other related papers [7, 14]. All chosen datasets and their characteristics are presented in Table 1. They differ mainly in the categories' number of instances, number of features, and imbalance ratio, which is calculated by dividing the size of the majority class by the size of the minority class.

Table 1. Benchmark datasets and their characteristics

Name	Number of Features	Number of Instances	Majority Instances	Minority Instances	Imbalance Ratio
Arcene	1998	200	112	88	1.273
Audit	25	775	470	305	1.541
Banknote Authentication	4	1372	762	610	1.249
Breast Cancer	30	569	357	212	1.684
Cleveland	13	297	243	54	4.500
Dermatology	33	366	346	20	17.300
Ecoli	7	336	284	52	5.462
Eucalyptus	8	642	544	98	5.551
Haberman	3	306	225	81	2.778
Ionosphere	32	351	225	126	1.786
Led	7	500	455	45	10.111
Libras	90	360	336	24	14.000
Liver	6	345	200	145	1.380
Page Blocks	10	5473	4913	560	8.773
Parkinsons	22	195	147	48	3.063
Pima	8	768	500	268	1.866
Spambase	57	4601	2788	1813	1.538
Vehicle	18	846	647	199	3.251
Vowel	12	9961	8865	1096	8.089
Yeast	8	1484	1240	244	5.082

The datasets are downloaded using ML-Research [35], an open-source library for machine learning research, and represent several of the best-known binary classification benchmark datasets in the ML field. All datasets remain unmodified in terms of their number of features as well as their number of instances. Furthermore, no missing values exist across all 20 datasets. Thus, there is no need to apply any missing-value replacement techniques. However, one step that needs to be performed is the normalization of the variables, since they often vary widely in their respective magnitudes. The normalization method chosen for this purpose is Z-score normalization (Equation 1), which is defined, for example, by Bakurov et al. [6] as follows:

$$z_{ij} = \frac{xij - \mu i}{\sigma i} \tag{1}$$

Here, x_{ij} represents the i-th feature of the j-th instance, μ_i and σ_i describe the mean and the standard deviation, respectively, calculated for every feature f_i. Finally, the resulting z_{ij} is the desired normalized value and therefore replaces the corresponding real value x_{ij} in the dataset. After the normalization is completed, the datasets are ready to be processed by all classifiers in exactly this same format.

In order to be able to assess the performance of the various techniques on unseen data, they are evaluated using the relevant metrics on the test partitions of the respective datasets, i.e., the train partitions are irrelevant in this context and are only used for training the models. The train and test partitions are generated for each dataset with a fully random division into 75% train and 25% test sets. This split is constantly redone for each of the 30 independent executions so that the results do not depend on the specificities of a single partition.

3-2- Evaluation Metrics

The so-called confusion matrix is the basis of several evaluation metrics for binary classification [15]. It divides the predictions given by a model into four different categories. True positives (TP) are the correctly identified instances of one class, e.g., class 1, false positives (FP) are instances that are incorrectly assigned to that class, and true negatives (TN) are the correctly identified instances of the other class, e.g., class 0, while false negatives (FN) are instances that are incorrectly assigned to this other class.

Since the majority of the benchmark datasets used are highly imbalanced, using a more appropriate metric than accuracy, which simply calculates the percentage of correctly identified instances in total, is essential for this analysis. For this reason, in the course of this work, as in various other studies on imbalanced learning [7, 13, 14], the so-called F1-score (Equation 4), which is defined below, is used as a metric for assessing the success of the tools. The F1-score is the harmonic mean of precision (Equation 2) and recall (Equation 3), with precision being the share of all correctly predicted positives among the entire number of predicted positives and recall representing the percentage of correctly identified positives out of the total actual positives

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

In order to consolidate the results of the comparison instead of making them dependent on just one single metric, in addition to the F1-score, the individual techniques are also compared using two further measures. These are the G-mean, which is defined hereafter, as well as the AUC [15]. The G-mean (Equation 7) is defined as the square root of the product of sensitivity (Equation 5) and specificity (Equation 6). The sensitivity is equivalent to the recall described above, while the specificity describes the fraction of correctly predicted negatives out of the entire actual negatives. Therefore, the G-mean is a measure for evaluating "the degree of inductive bias in terms of a ratio of positive accuracy and negative accuracy" [15].

$$\text{Sensitivity} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{6}$$

$$\text{G-mean} = \sqrt{\text{Sensitivity} \cdot \text{Specificity}} \tag{7}$$

For calculating the AUC metric, it is necessary to first obtain the ROC curve. The ROC graph comprises both TP rate (Equation 8) and FP rate (Equation 9) axes, with the TP rate being identical to sensitivity or recall and the FP rate representing the ratio between incorrectly predicted positives and the actual negative instances. Typically, the TP rate

represents the y-axis of the ROC graph, while the FP rate corresponds to the x-axis. Commonly, the ROC curve of a classifier is composed of several points, usually characterized by different threshold values, i.e., the cutoff point of the predicted probability from which an instance is assigned to one class instead of the other. However, it is also possible that a classifier produces only a single ROC point, so only a single pair of TP rate and FP rate, which is the case with models that only output hard class labels instead of the corresponding probabilities [15]. The latter is the case for this study since it is not possible to directly obtain the probabilities with certain tools. Therefore, for reasons of comparability, the AUC score is calculated solely based on the predicted class labels throughout all techniques. While this makes a single classifier's ROC curve itself hardly informative, comparing the respective AUC value is nevertheless a useful measure to evaluate the tools' performances against each other.

$$\text{TP rate} = \text{Sensitivity} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

$$\text{FP rate} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{9}$$

For the FP rate, 0 is the best value since it implies that no FP exists among the predicted class labels, and 1 is the worst. Regarding all the other metrics, a value of 1 is the optimal score, while a classifier evaluated with a 0 is considered to be not useful at all. The resulting metric values are consistently rounded to three decimal places in the course of this work.

3-3- Setups of the Techniques under Analysis

Since the objective of the AutoML tools is to simplify and automate the application of ML models, the most appropriate action for this study is to use them without altering their setups. Furthermore, for the comparison with the established AutoML tools to be meaningful, the GP approaches also do not undergo a separate hyperparameter optimization. Instead, they are applied in the same setup to all datasets. Likewise, the logistic regression is applied in its default setup, as given by the ML library scikit-learn [36].

However, a major benefit of GSGP and the three established AutoML tools is that they allow the user to choose the metric according to which the models should be optimized. In order to take advantage of this feature, the default optimization metric of all four of them is replaced with the F1-score since the examined datasets are all imbalanced binary classification problems, and the techniques are subsequently evaluated mainly based on the achieved F1-score. In contrast, a drawback of DCGPANN is that it cannot be optimized according to the F1-score. Hence, the optimization in DCGPANN is conducted according to its default measure for classification, the cross-entropy loss. Apart from these changes in the optimization metrics, all five techniques are used in their default setups as given by their developers.

4- Results and Discussion

4-1- Average Performances

First, the different classification methods are examined based on their average performance on the 20 benchmark datasets. Thus, the average values of the metrics achieved by each approach after 30 repetitions per dataset are considered. Each technique is given a rank per dataset, which expresses its relative performance to the other five techniques, with the best average metric value corresponding to rank one and the worst consequently to rank six. The average ranks across all datasets achieved by each classification method per metric are shown in Figure 3.

In order to determine whether there is a significant difference between the average ranks of the methods in this case, the Friedman test is applied. For each of the three metrics, a separate Friedman test is performed, independent of the rankings associated with the respective other metrics. Thereby, the null hypothesis is the same for each test, namely that there is no significant difference in the mean ranks of the classification methods with regard to the specific metric. The null hypothesis is rejected from a significance level of 0.05 and lower, and the alternative hypothesis, which states that there are statistically significant differences, is accepted. The p-values resulting from the three Friedman tests are presented in Table 2. Since all three p-values are smaller than the significance level, the null hypothesis is rejected in all three cases. Thus, it can be concluded that there are significant differences in the average performances of the six methods with respect to all three metrics, F1-score, G-mean, and AUC.

Table 2. Results of the Friedman tests for the average performances on the benchmark datasets

Metric	P-value	Significance
F1-score	0.005	True
G-mean	0.002	True
AUC	0.003	True

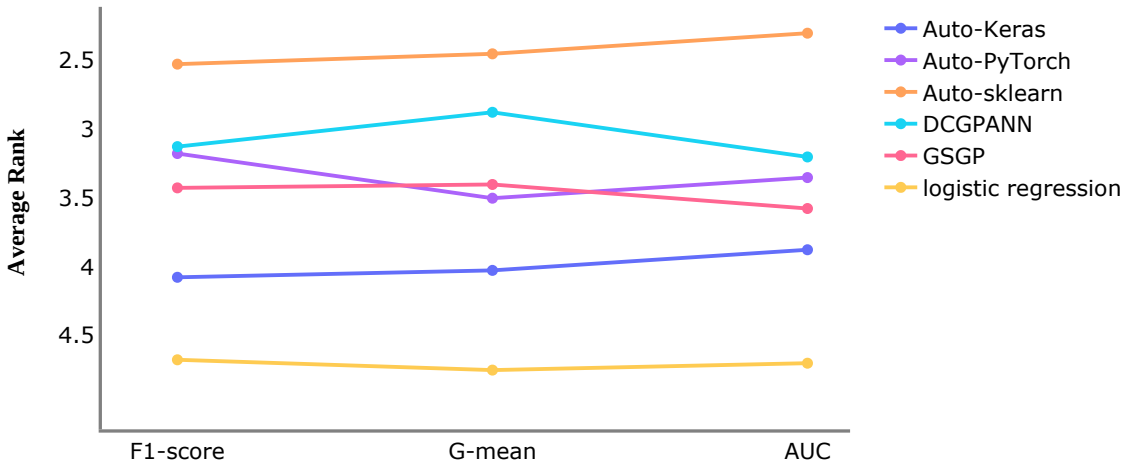


Figure 3. Average ranks achieved by the different approaches across all benchmark datasets in terms of the respective average achieved metric values per dataset after 30 repetitions

In the following paragraphs, the general differences between the individual techniques, as confirmed by the Friedman test, are investigated in more detail. For this purpose, the average metric values achieved by the individual methods are compared in a pairwise manner in order to draw conclusions about the performance differences between two methods. For each metric, 15 separate and independent Wilcoxon tests are performed, covering all 15 possible pairs of techniques. The null hypothesis throughout all the tests is that there is no statistically significant difference in the mean ranks of the two respective classification methods under consideration. At a significance level of 0.05 and lower, the null hypothesis is rejected, and the alternative hypothesis, which indicates that there is a statistically significant difference between two methods, is accepted. The p-values resulting from the pairwise Wilcoxon tests per metric are presented in Table 3, significant p-values are highlighted in bold.

Table 3. P-values resulting from pairwise Wilcoxon tests for the average performances on the benchmark datasets

Technique	Metric	Auto-Keras	Auto-PyTorch	Auto-Sklearn	DCGPANN	GSGP
Auto-PyTorch	F1-score	**0.042**	-	-	-	-
	G-mean	**0.050**	-	-	-	-
	AUC	0.107	-	-	-	-
Auto-Sklearn	F1-score	**0.003**	0.126	-	-	-
	G-mean	**0.002**	**0.022**	-	-	-
	AUC	**0.003**	**0.027**	-	-	-
DCGPANN	F1-score	**0.048**	0.911	0.126	-	-
	G-mean	**0.013**	0.235	0.173	-	-
	AUC	0.156	0.881	0.117	-	-
GSGP	F1-score	0.126	0.654	0.204	0.970	-
	G-mean	**0.037**	0.455	0.263	0.526	-
	AUC	0.191	0.867	0.083	0.823	-
Logistic regression	F1-score	0.681	**0.030**	**0.005**	**0.021**	0.191
	G-mean	0.658	**0.037**	**0.007**	**0.011**	0.076
	AUC	0.641	0.057	**0.008**	**0.024**	0.135

The Wilcoxon tests show that there are eight pairs of techniques for which the null hypothesis has to be rejected with respect to at least one metric, meaning that those pairs have statistically significant performance differences on one or more metrics. These eight pairs are shown in Table 4. The metrics columns show the exact pairwise results if there is a significant difference for a specific metric. The first figure represents the number of datasets in which the first-mentioned technique achieved a higher average value than the second one, the figure in the middle stands for the number of ties, and the last figure indicates the number of datasets in which the latter method achieved a higher value. Thereby, the highest figures, and consequently statistically significantly superior techniques with respect to at least one metric, are highlighted in bold.

Table 4. Pairs of techniques with significant differences concerning their average performance

Pair of Techniques	F1-score	G-mean	AUC
Auto-Keras & **Auto-PyTorch**	7-0-**13**	7-2-**11**	not significant
Auto-Keras & **Auto-Sklearn**	4-1-**15**	4-2-**14**	4-2-**14**
Auto-Keras & **DCGPANN**	7-0-**13**	6-0-**14**	not significant
Auto-Keras & **GSGP**	not significant	8-0-**12**	not significant
Auto-PyTorch & **Auto-Sklearn**	not significant	4-1-**15**	4-1-**15**
Auto-PyTorch & logistic regression	**16**-0-4	**16**-0-4	not significant
Auto-Sklearn & logistic regression	**17**-0-3	**16**-1-3	**17**-0-3
DCGPANN & logistic regression	**16**-0-4	**17**-0-3	**17**-0-3

Several observations can be made from these results. Auto-Sklearn is the only method that outperforms three others with respect to at least one metric, followed by Auto-PyTorch and DCGPANN, which outperform two others each, and GSGP, which is superior to one other method. In contrast, the most frequently significantly outperformed method on at least one metric is Auto-Keras, with a total of four times, followed by logistic regression with three times and Auto-PyTorch with one time. The only two methods that statistically significantly dominate another across all three metrics are Auto-Sklearn and DCGPANN, with the former even managing to do so twice. Only Auto-Keras and logistic regression are not able to significantly outperform other methods, while only Auto-Sklearn, DCGPANN, and GSGP are not significantly outperformed by any other technique.

4-2- Maximum Performances

In comparison to the previous section, the different classification methods are examined based on their best performance on the 20 benchmark datasets instead of the average performance in this section. Thus, the maximum values of the metrics achieved by each technique after 30 repetitions per dataset are considered. However, this is the only difference between this part and section 4-1, meaning that the procedures are identical, and are therefore not explained in detail again here. Please refer to the previous section for more information about the formal process. The average ranks across all datasets achieved by each method per metric, regarding their respective best score per dataset, are shown in Figure 4.

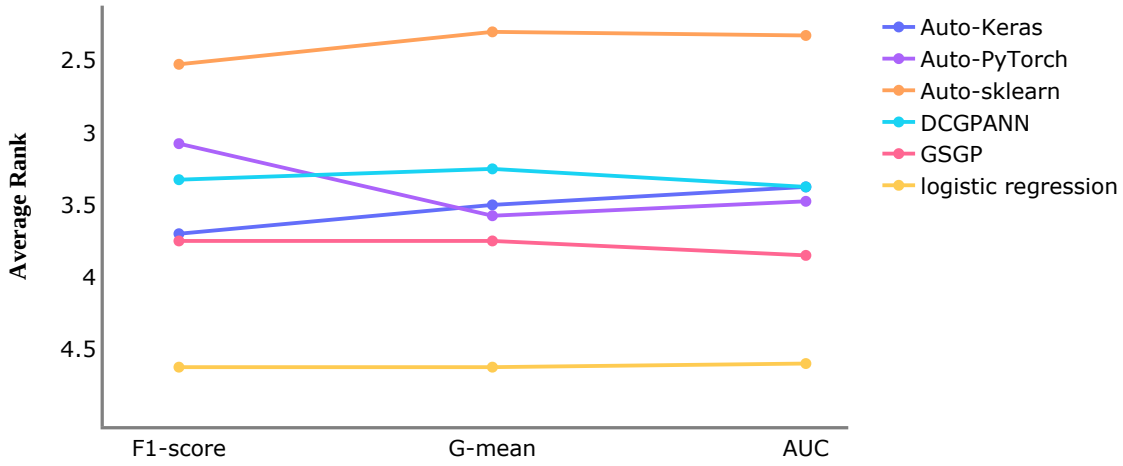


Figure 4. Average ranks achieved by the different approaches across all benchmark datasets in terms of the respective maximum achieved metric values per dataset after 30 repetitions

Friedman tests are also applied here to test for statistically significant differences. Table 5 presents the results of these tests, which show that for all three metrics, the null hypothesis is rejected, meaning that there are statistically significant differences between the average ranks of the methods in terms of the maximum score achieved throughout all metrics.

Table 5. Results of the Friedman tests for the maximum performances on the benchmark datasets

Metric	P-value	Significance
F1-score	0.013	True
G-mean	0.006	True
AUC	0.008	True

As in Section 4-1, Wilcoxon tests are conducted next. These compare the techniques in pairs and test the null hypothesis that there are no significant differences in the average ranks between two methods, meaning that their maximum reached metric scores across all datasets do not differ significantly. The p-values resulting from these Wilcoxon tests per metric are presented in Table 6, whereby p-values that indicate significance are highlighted in bold.

Table 6. P-values resulting from pairwise Wilcoxon tests for the best performances on the benchmark datasets

Technique	Metric	Auto-Keras	Auto-PyTorch	Auto-Sklearn	DCGPANN	GSGP
Auto-PyTorch	F1-score	0.494	-	-	-	-
	G-mean	0.872	-	-	-	-
	AUC	0.952	-	-	-	-
Auto-Sklearn	F1-score	**0.015**	0.085	-	-	-
	G-mean	**0.008**	**0.013**	-	-	-
	AUC	**0.013**	**0.011**	-	-	-
DCGPANN	F1-score	0.523	0.523	0.084	-	-
	G-mean	0.570	0.828	**0.015**	-	-
	AUC	0.619	0.586	**0.015**	-	-
GSGP	F1-score	0.936	0.276	**0.014**	0.616	-
	G-mean	0.601	0.811	**0.008**	0.845	-
	AUC	0.809	0.948	**0.007**	0.845	-
Logistic regression	F1-score	0.314	**0.013**	**0.001**	**0.023**	0.121
	G-mean	0.295	**0.010**	**0.002**	**0.021**	0.131
	AUC	0.305	**0.010**	**0.002**	**0.033**	0.136

This time, the Wilcoxon tests show that there are seven pairs of techniques for which the null hypothesis is rejected with respect to at least one metric, indicating that those pairs have statistically significant performance differences on one or more metrics with regard to their maximum score achieved. All these seven pairs are shown in Table 7, in the same fashion as in section 4-1.

Table 7. Pairs of techniques with significant differences concerning their best performance

Pair of Techniques	F1-score	G-mean	AUC
Auto-Keras & **Auto-Sklearn**	5-2-**13**	4-2-**14**	5-2-**13**
Auto-PyTorch & **Auto-Sklearn**	not significant	6-3-**11**	6-3-**11**
Auto-PyTorch & logistic regression	**14**-1-5	**15**-1-4	**15**-1-4
Auto-Sklearn & DCGPANN	not significant	**12**-3-5	**12**-3-5
Auto-Sklearn & GSGP	**14**-2-4	**15**-2-3	**16**-2-2
Auto-Sklearn & logistic regression	**16**-2-2	**16**-2-2	**16**-2-2
DCGPANN & logistic regression	**13**-4-3	**14**-2-4	**13**-4-3

It is noticeable that the comparison in terms of the maximum achieved scores is more consistent across the different metrics than in the case of the average achieved scores. Five of the seven significantly different pairs here are significantly different throughout all three metrics, whereas in the previous section, this was the case for only three out of eight. While Auto-Sklearn is already the dominant technique in terms of average scores, it is even more so in terms of maximum scores. All five other techniques are statistically significantly outperformed by Auto-Sklearn on at least two metrics. The logistic regression, in terms of average scores, is outperformed by three other methods, is again outperformed by the same three, namely Auto-PyTorch, Auto-Sklearn, and DCGPANN. The method that arguably represents the one with the greatest difference between the analysis in Section 4-1 and the one in this section is Auto-Keras. While Auto-Keras was significantly outperformed by four other techniques in at least one metric for average results, it was only outperformed once, by Auto-Sklearn, in terms of maximum results.

4-3- Standard Deviation

After analyzing the techniques based on their average scores as well as their maximum scores in the previous sections, the analysis in this chapter is performed based on the standard deviation. For this purpose, the standard deviation of the scores is calculated after the 30 executions per technique and dataset, respectively, for the three metrics, F1-score, G-mean, and AUC, based on the test partitions of the datasets. Thus, the analysis aims to provide information on how

consistently the individual methods perform. The average ranks across the 20 datasets in terms of standard deviation are shown in Figure 5, with rank one indicating the highest standard deviation and rank six the lowest. Therefore, a lower rank is desirable in this case, as it implies a more stable performance throughout the 30 repetitions.

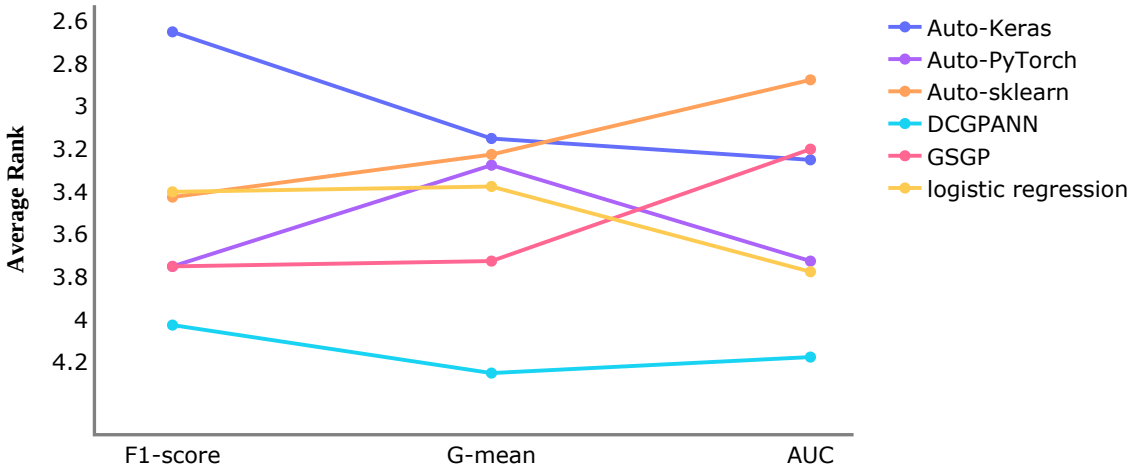


Figure 5. Average ranks achieved by the different approaches across all benchmark datasets in terms of standard deviation of the metric values per dataset after 30 repetitions

In this analysis, however, the null hypothesis of the Friedman tests could not be rejected for any of the three metrics, contrary to the two preceding analyses. This means that according to the Friedman tests, there is no statistically significant difference in the average rankings in terms of the standard deviation of the six techniques under investigation. However, in order to test whether there are any differences in the pairwise comparisons, in contrast to the non-existent differences across all six methods together, the Wilcoxon tests are nevertheless conducted. Only three pairs that differ statistically significantly can be identified, with none being significantly different across all three metrics. However, it is noteworthy that in each case, the three pairs are formed by one of the three state-of-the-art AutoML tools as well as DCGPANN consistently as a counterpart, with the latter always being the technique with the statistically significantly lower standard deviation with respect to at least one metric. Therefore, it can be concluded that DCGPANN provides more consistent results than the three established AutoML tools.

4-4- Generalization Ability

Ultimately, the analysis already familiar from sections 4-1 to 4-3 is performed a final time, specifically this time regarding the generalization ability of the classification methods. This ability is measured by the percentage of the training metrics that the methods are able to reach in the test partition. Thus, for example, if a technique achieves an F1-score of 0.8 at training and an F1-score of 0.7 on the test partition of an execution on one dataset, the associated generalization percentage is 0.7/0.8 = 0.875. For the following analysis, the average of these calculated percentage values after 30 executions is used in each case. The results of the pairwise Wilcoxon tests reveal that simple logistic regression outperforms all other approaches across all metrics in terms of generalization ability. A possible reason for this is that, since the test partition scores of the logistic regression are already rather poor in comparison, it is also comparatively worse at modeling the relationships during training. Therefore, this property is not necessarily a benefit of the logistic regression and is disregarded for the pairwise analysis in this case. So, the analysis of the pairs with statistically significant differences among the three state-of-the-art AutoML tools tested and the two GP approaches under examination reveals that in all cases, methods based on ANNs are outperformed, more precisely Auto-Keras once and DCGPANN three times. The tool that outperforms both in terms of generalization ability is Auto-Sklearn, while Auto-PyTorch, as well as GSGP, are also able to outperform DCGPANN.

4-5- Discussion

During the analysis based on 20 ML benchmark datasets (sections 4-1 to 4-4), five techniques for imbalanced binary classification are examined, and a default logistic regression is used as a reference. The five techniques include two approaches based on GP, DCGPANN, and GSGP. The other three are the established AutoML tools Auto-Keras, Auto-PyTorch, and Auto-Sklearn. Both GP methods accomplish the given classification tasks with only one attempt, i.e., only one initialization is performed per execution per dataset, followed by the execution of the algorithm until the time limit of 300 seconds is reached. In GSGP, a population of mathematical functions is evolved to fit the dataset as accurately as possible, whereas in DCGPANN, an ANN is trained over numerous epochs for this purpose. Unlike the two GP techniques, the three state-of-the-art AutoML tools handle the task with multiple attempts. Auto-Keras repeatedly initializes new ANNs with varying network architectures and trains them. Auto-PyTorch and Auto-Sklearn try models

from a pool of baseline ML algorithms, optimize them, and then build ensembles. Only Auto-PyTorch and Auto-Sklearn also include data pre-processing steps. All three of these AutoML tools return the respective best model found after the time has elapsed. All five techniques are used by default. This means that all the results obtained in this work were achieved without having to apply steps such as model selection or hyperparameter optimization. In addition, the datasets are not modified by any resampling techniques for adjusting their imbalance since it is intended to test how well the respective techniques can deal with the problem of imbalanced data. Therefore, the results are of particular interest to non-ML experts, as no profound knowledge is required for the application of the methods as used here, but only the data and the prebuilt modules, which take care of the whole process of creating a suitable model, are needed.

Analyzing the tools first with respect to the two categories, standard deviation and generalization ability, i.e., the categories that are supposed to reveal specific peculiarities, the following three observations can be made. First, DCGPANN has a significantly lower standard deviation within its performance than all three state-of-the-art AutoML tools; thus, it is the tool that provides the most consistent results. Then, Auto-Sklearn has a significantly higher generalization ability than Auto-Keras and DCGPANN and is, therefore, the tool with the best generalization ability. Lastly, DCGPANN has a significantly lower generalization ability than Auto-PyTorch, Auto-Sklearn, and GSGP; thus, it is the tool with the worst generalization ability. Additionally, it is noted that both techniques based on ANNs, namely Auto-Keras and DCGPANN, are outperformed by others in terms of generalization ability. Therefore, it can be argued that both are not capable of solving the overfitting problem of ANNs sufficiently.

Now, for the core part of this discussion, we assume that one tool only outperforms another if it is superior to it in at least one of the two categories of average or maximum performance since these two are the crucial ones in practice, while the categories of standard deviation and generalization ability are rather intended to provide further insights into the techniques' performances. Auto-Sklearn, a tool based on various baseline ML algorithms, emerged clearly as the best technique from the comparison, as it outperformed three out of five others in terms of average results and all five in terms of best results. Therefore, it can be concluded that Auto-Sklearn is the strongest-performing technique, with a great chance of achieving good results, and its use for imbalanced binary classification can be recommended. Auto-PyTorch, again an AutoML tool using multiple baseline ML models, outperformed two other techniques in terms of average performance and one other concerning maximum performance, while in both cases only being outperformed by Auto-Sklearn. Thus, Auto-PyTorch is a solid tool for imbalanced binary classification. The third state-of-the-art AutoML tool, Auto-Keras, was less successful than the other two during this study. It is noticeable that Auto-Keras, particularly, falls behind in terms of average performance, while it scores fairly average in terms of maximum performance. Therefore, it can be assumed that Auto-Keras strongly depends on the respective initialization. Hence, applying Auto-Keras only once bears the risk of getting a poor model, so if applied, it should be tried several times, as it seems to vary considerably from attempt to attempt.

DCGPANN arguably performed at a similar level as Auto-PyTorch, as it also outperformed two others at average scores and one other at maximum scores. It is remarkable that DCGPANN is superior to Auto-Keras, i.e., the DCGPANN approach seems more effective than the one of Auto-Keras, even though the latter generates multiple ANNs while DCGPANN only generates a single one per attempt. Furthermore, it is interesting that DCGPANN achieves significantly more consistent results than all three established AutoML tools. The other GP method, GSGP, could only outperform one technique, namely Auto-Keras, in terms of average performance. However, it has also itself been outperformed only once, and that was by Auto-Sklearn in terms of maximum scores. Therefore, it can be concluded that both DCGPANN and GSGP are also solid techniques to be used for imbalanced binary classification, being even able to keep up with certain state-of-the-art AutoML tools. Interpreting the results from the point of view of the advanced GP techniques, it is astonishing that neither of them is statistically significantly outperformed on average after 30 executions across all 20 benchmark datasets by any state-of-the-art AutoML tool, not even by the evidently best tool, Auto-Sklearn. While the two GP techniques have already proven to be superior to various fundamental ML algorithms in other studies [5, 7], this study reveals that they are even competitive in comparison to AutoML tools. This demonstrates the outstanding capabilities of DCGPANN and GSGP in the field of imbalanced binary classification and thus makes these two GP tools highly recommended for this task in practice.

5- Conclusion

To overcome the research gap described in the introduction regarding direct comparative analyses between advanced GP techniques and state-of-the-art AutoML, this study compared the two advanced GP methods, DCGPANN and GSGP, alongside the three state-of-the-art AutoML tools Auto-Keras, Auto-PyTorch, and Auto-Sklearn in terms of their performance in the domain of imbalanced binary classification across 20 benchmark datasets. One of the key results is that neither of the two GP techniques could be statistically significantly outperformed on average by any of the state-of-the-art tools and that they were both outperformed only by Auto-Sklearn in terms of best performance. Therefore, to answer the first part of the above-stated research question, it can be concluded that the GP concepts DCGPANN and GSGP are indeed competitive compared to the three state-of-the-art AutoML tools when applied to imbalanced binary classification, making their use in practice highly recommended. This is a significant and remarkable finding since the

two GP techniques are still rather novel and unexploited, while the three AutoML tools represent the current top standard in the field. Nevertheless, Auto-Sklearn emerged as the strongest performer out of all of the tools under analysis. Thus, the answer to the second part of the research question, which of the five methods is the most successful one overall, is conclusively Auto-Sklearn. As a consequence of the results of this study, it is advisable for practitioners dealing with imbalanced binary classification tasks to consider the advanced GP techniques DCGPANN and GSGP in addition to the AutoML tools or even as an alternative. For the academic community, the results shall emphasize the potential of GP and are intended to serve as motivation for continued exploration and development of advanced GP techniques in order to further improve their capabilities.

6- Declarations

6-1- Author Contributions

Conceptualization, F.F. and F.B.; methodology, F.F. and F.B.; software, F.F.; validation, F.F. and F.B.; formal analysis, F.F. and F.B.; investigation, F.F. and F.B.; resources, F.F. and F.B.; writing—original draft preparation, F.F.; writing—review and editing, F.F. and F.B.; visualization, F.F. and F.B.; supervision, F.B. All authors have read and agreed to the published version of the manuscript.

6-2- Data Availability Statement

Publicly available datasets were analyzed in this study. This data can be found here: *https://github.com/joaopfonseca/ml-research.*

6-3- Funding

This work was supported by a grant of the Portuguese Foundation for Science and Technology ("Fundação para a Ciência e a Tecnologia"), DSAIPA/DS/0116/2019, and project UIDB/04152/2020—Centro de Investigação em Gestão de Informação (MagIC).

6-4- Institutional Review Board Statement

Not applicable.

6-5- Informed Consent Statement

Not applicable.

6-6- Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancies have been completely observed by the authors.

7- References

[1] He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 212, 106622. doi:10.1016/j.knosys.2020.106622.

[2] Wever, M., Tornede, A., Mohr, F., & Hullermeier, E. (2021). AutoML for Multi-Label Classification: Overview and Empirical Evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(9), 3037–3054. doi:10.1109/TPAMI.2021.3051276.

[3] Le, T. T., Fu, W., & Moore, J. H. (2020). Scaling tree-based automated machine learning to biomedical big data with a feature set selector. Bioinformatics, 36(1), 250–256. doi:10.1093/bioinformatics/btz470.

[4] Castelli, M., Pinto, D. C., Shuqair, S., Montali, D., & Vanneschi, L. (2022). The Benefits of Automated Machine Learning in Hospitality: A Step-By-Step Guide and AutoML Tool. Emerging Science Journal, 6(6), 1237–1254. doi:10.28991/ESJ-2022-06-06-02.

[5] Märtens, M., & Izzo, D. (2019). Neural network architecture search with differentiable cartesian genetic programming for regression. Proceedings of the Genetic and Evolutionary Computation Conference Companion. doi:10.1145/3319619.3322003.

[6] Bakurov, I., Castelli, M., Fontanella, F., & Vanneschi, L. (2019). A Regression-like Classification System for Geometric Semantic Genetic Programming. Proceedings of the 11th International Joint Conference on Computational Intelligence, Vienna, Austria. doi:10.5220/0008052900400048.

[7] Bakurov, I., Castelli, M., Fontanella, F., Scotto di Freca, A., & Vanneschi, L. (2022). A novel binary classification approach based on geometric semantic genetic programming. Swarm and Evolutionary Computation, 69, 101028. doi:10.1016/j.swevo.2021.101028.

[8] Conrad, F., Mälzer, M., Schwarzenberger, M., Wiemer, H., & Ihlenfeldt, S. (2022). Benchmarking AutoML for regression tasks on small tabular data in materials design. Scientific Reports, 12(1), 19350. doi:10.1038/s41598-022-23327-1.

[9] Alsharef, A., Aggarwal, K., Sonia, Kumar, M., & Mishra, A. (2022). Review of ML and AutoML Solutions to Forecast Time-Series Data. Archives of Computational Methods in Engineering, 29(7), 5297–5311. doi:10.1007/s11831-022-09765-0.

[10] Jin, H., Song, Q., & Hu, X. (2019). Auto-Keras: An Efficient Neural Architecture Search System. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. doi:10.1145/3292500.3330648.

[11] Zimmer, L., Lindauer, M., & Hutter, F. (2021). Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(9), 3079–3090. doi:10.1109/TPAMI.2021.3067763.

[12] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., & Hutter, F. (2019). Auto-Sklearn: Efficient and Robust Automated Machine Learning. The Springer Series on Challenges in Machine Learning, 113–134. doi:10.1007/978-3-030-05318-5_6.

[13] Douzas, G., & Bacao, F. (2019). Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE. Information Sciences, 501, 118–135. doi:10.1016/j.ins.2019.06.007.

[14] Douzas, G., & Bacao, F. (2018). Effective data generation for imbalanced learning using conditional generative adversarial networks. Expert Systems with Applications, 91, 464–471. doi:10.1016/j.eswa.2017.09.030.

[15] He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284. doi:10.1109/TKDE.2008.239.

[16] Santu, S. K. K., Hassan, M. M., Smith, M. J., Xu, L., Zhai, C., & Veeramachaneni, K. (2022). AutoML to Date and Beyond: Challenges and Opportunities. ACM Computing Surveys, 54(8), 1–36. doi:10.1145/3470918.

[17] Bahri, M., Salutari, F., Putina, A., & Sozio, M. (2022). AutoML: state of the art with a focus on anomaly detection, challenges, and research directions. International Journal of Data Science and Analytics, 14(2), 113–126. doi:10.1007/s41060-022-00309-0.

[18] Cerrada, M., Trujillo, L., Hernández, D. E., Correa Zevallos, H. A., Macancela, J. C., Cabrera, D., & Vinicio Sánchez, R. (2022). AutoML for Feature Selection and Model Tuning Applied to Fault Severity Diagnosis in Spur Gearboxes. Mathematical and Computational Applications, 27(1), 6. doi:10.3390/mca27010006.

[19] Celik, B., Singh, P., & Vanschoren, J. (2022). Online AutoML: an adaptive AutoML framework for online learning. Machine Learning. doi:10.1007/s10994-022-06262-0.

[20] Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2022). Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning. Journal of Machine Learning Research, 23. doi:10.48550/arXiv.2007.04074.

[21] Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. Statistics and Computing, 4(2), 87–112. doi:10.1007/BF00175355.

[22] Tran, B., Xue, B., & Zhang, M. (2019). Genetic programming for multiple-feature construction on high-dimensional classification. Pattern Recognition, 93, 404–417. doi:10.1016/j.patcog.2019.05.006.

[23] Vanneschi, L., Castelli, M., Scott, K., & Trujillo, L. (2019). Alignment-based genetic programming for real life applications. Swarm and Evolutionary Computation, 44, 840–851. doi:10.1016/j.swevo.2018.09.006.

[24] Moraglio, A., Krawiec, K., & Johnson, C.G. (2012). Geometric Semantic Genetic Programming. Parallel Problem Solving from Nature - PPSN XII, Lecture Notes in Computer Science, volume 7491, Springer, Berlin, Germany. doi:10.1007/978-3-642-32937-1_3.

[25] Vanneschi, L., Castelli, M., & Silva, S. (2014). A survey of semantic methods in genetic programming. Genetic Programming and Evolvable Machines, 15(2), 195–214. doi:10.1007/s10710-013-9210-0.

[26] Bakurov, I., Castelli, M., Gau, O., Fontanella, F., & Vanneschi, L. (2021). Genetic programming for stacked generalization. Swarm and Evolutionary Computation, 65, 100913. doi:10.1016/j.swevo.2021.100913.

[27] Vanneschi, L., Castelli, M., Manzoni, L., & Silva, S. (2013). A New Implementation of Geometric Semantic GP and Its Application to Problems in Pharmacokinetics. Genetic Programming. EuroGP 2013. Lecture Notes in Computer Science, volume 7831, Springer, Berlin, Germany. doi:10.1007/978-3-642-37207-0_18.

[28] Krawiec, K., & Pawlak, T. (2013). Locally geometric semantic crossover: A study on the roles of semantics and homology in recombination operators. Genetic Programming and Evolvable Machines, 14(1), 31–63. doi:10.1007/s10710-012-9172-7.

[29] Turner, A. J., & Miller, J. F. (2015). Introducing a cross platform open source Cartesian Genetic Programming library. Genetic Programming and Evolvable Machines, 16(1), 83–91. doi:10.1007/s10710-014-9233-1.

[30] Turner, A. J., & Miller, J. F. (2017). Recurrent Cartesian Genetic Programming of Artificial Neural Networks. Genetic Programming and Evolvable Machines, 18(2), 185–212. doi:10.1007/s10710-016-9276-6.

[31] Miller, J. F., & Smith, S. L. (2006). Redundancy and computational efficiency in Cartesian genetic programming. IEEE Transactions on Evolutionary Computation, 10(2), 167–174. doi:10.1109/TEVC.2006.871253.

[32] Izzo, D., Biscani, F., & Mereta, A. (2017). Differentiable Genetic Programming. Genetic Programming. EuroGP 2017, Lecture Notes in Computer Science, vol 10196. Springer, Cham, Switzerland. doi:10.1007/978-3-319-55696-3_3.

[33] Guyon, I. (2003). Design of experiments of the NIPS 2003 variable selection benchmark. NIPS 2003 workshop on feature extraction and feature selection, 11-13 December, 2003, Whistler, United States.

[34] Haynes, W. (2013). Wilcoxon Rank Sum Test. Encyclopedia of Systems Biology, 2354–2355, Springer, New York, United States. doi:10.1007/978-1-4419-9863-7_1185.

[35] Fonseca, J., Douzas, G., & Bacao, F. (2021). Increasing the effectiveness of active learning: Introducing artificial data generation in active learning for land use/land cover classification. Remote Sensing, 13(13), 2619. doi:10.3390/rs13132619.

[36] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.