UNIVERSIDADE D
COIMBRA

Miguel André Lourenço Rabuge

BIO-INSPIRED AUTOMATIC FEATURE ENGINEERING FOR AUDIOLOGY

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, supervised by Professor Nuno António Marques Lourenço and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July of 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Miguel André Lourenço Rabuge

Bio-Inspired Automatic Feature Engineering for Audiology

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, supervised by Prof. Nuno António Marques Lourenço and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July 2023

This work was funded by project A4A: Audiology for All (CENTRO-01-0247-FEDER-047083) financed by the Operational Program for Competitiveness and Internationalisation of PORTUGAL 2020 through the European Regional Development Fund and by the FCT - Foundation for Science and Technology, I.P. / MCTES through national funds (PIDDAC), within the scope of CISUC RD Unit - UIDB/00326/2020 or project code UIDP/00326/2020.

Acknowledgements

I would like to express my gratitude to my supervisor, Nuno Lourenço, with whom I have learned a great deal, for allowing me to participate in research projects for the past 3 years. Thank you for sparking my interest in research and for all of your answers, criticism and reassurance. For my friends, Duarte Dias, Gabriel Fernandes and Pedro Rodrigues thank you for your constant availability to discuss my work, giving me helpful clues on what I could be doing wrong. I would also like to thank my family, for believing in me. Thank you for all of your support and willingness to listen to subjects you do not quite understand, but still striving to comprehend them. Lastly, I would like to thank Inês, my significant other, for all of our time together and for the trust, patience and support that you have given me throughout these years.

Abstract

Hearing Loss (HL) is becoming a concerning problem in modern society. In February 2023, the World Health Organization (WHO) stated that over 5% of the world's population (430 million people) requires rehabilitation to address HL [WHO - Hearing Loss]. As more medical data is being produced and stored digitally, data scientists and engineers are trying to help solve medical field-related problems in an automated manner. Therefore, a method to preemptively predict HL is of utter importance.

Given the current trend where information of varied types is being collected from multiple sources, a need to select and combine these pieces of information arises when aiming to maximize the prediction of HL.

Feature Engineering (FE) is a time-consuming and error-prone task as it is usually made by human experts. The framework examined in this work aims to automatically boost Machine Learning (ML) models' performance by enhancing original features through evolutionary Feature Selection (FS) and Feature Construction (FC) methods.

This work proposes FEDORA, an Evolutionary Automated Machine Learning (AutoML) framework for FE. The proposed framework will be compared with state-of-the-art FE techniques and analysed in terms of performance and of FE. The best result of the framework is 76.2% balanced accuracy, using 9 total features (5 Original, 2 Engineered and 2 Complex) out of the 60 original ones. FEDORA was able to engineer a transformation that achieved a score of 72,8% balanced accuracy with 1 single complex feature. Results point to FEDORA being able to reduce the dimensionality of the data while statistically maintaining performance.

Keywords

Automated Machine Learning, Evolutionary Computation, Grammatical Evolution, Feature Engineering, Hearing Loss.

Resumo

A perda de audição é um problema cada vez mais sério na sociedade. Em Fevereiro de 2023, a Organização Mundial da Saúde (OMS) afirmou que 5% da população mundial (430 milhões de pessoas) necessita de reabilitação para tratar da sua perda auditiva [WHO - Hearing Loss]. À medida que dados médicos vão sendo gerados e armazenados digitalmente, engenheiros e ciêntistas de dados procuram ajudar a resolver problemas na área da medicina, de forma automatizada. Deste modo, um método que consiga prever preemptivamente perda auditiva é de extrema importância.

Dada a tendência atual onde dados de vários tipos estão a ser obtidos de diversas fontes, surge a necessidade de selecionar e combinar esta informação com vista a maximizar a deteção de perda auditiva.

Feature Engineering é uma tarefa demorada e propensa a erros, dado que geralmente é realizada por especialistas humanos. A *framework* examinada neste trabalho tem como objetivo aumentar automaticamente o desempenho dos modelos de *Machine Learning*, aprimorando os atributos originais através de métodos evolucionários de seleção e construção de atributos.

Este trabalho propõe FEDORA, uma *framework* evolucionária de *Automated Machine Learning* para *Feature Engineering*. A *framework* proposta será comparada com técnicas estado-da-arte de *Feature Engineering* e analisada em termos de desempenho e de seleção e construção de atributos. O melhor resultado da *framework* é de 76,2% *Balanced Accuracy*, usando 9 atributos (5 originais, 2 *Engineered* e 2 complexos) dos 60 originais. A *framework* foi capaz de gerar uma transformação que alcançou uma pontuação de 72,8% *Balanced Accuracy* com 1 único atributo complexo. Os resultados apontam para a *framework* ser capaz de reduzir a dimensionalidade dos dados enquanto estatisticamente mantém *performance*.

Palavras-Chave

Aprendizagem Computacional Automática, Computação Evolucionária, Evolução Gramatical, Engenharia de Atributos, Perda Auditiva

Contents

Acronyms

A4A Audiology for All.

AE Autoencoder.

AI Artificial Intelligence.

ANN Artificial Neural Network.

ANOVA Analysis of Variance.

AST Abstract Syntax Tree.

AutoML Automated Machine Learning.

BNF Backus-Naur Form.

CASH Combined Algorithm Selection and Hyperparameter optimization problem.

CFG Context-Free Grammar.

CNN Convolutional Neural Network.

DT Decision Tree.

EA Evolutionary Algorithm.

EC Evolutionary Computation.

ER Entity-Relationship.

ETL Extract, Transform and Load.

FC Feature Construction.

FE Feature Engineering.

FMS Full Model Selection.

FS Feature Selection.

GA Genetic Algorithm.

GBGP Grammar-Based Genetic Programming.

GE Grammatical Evolution.

GP Genetic Programming.

HL Hearing Loss.

HPO Hyper-parameter Optimisation.

HyTEA Hybrid Tree Evolutionary Algorithm.

ML Machine Learning.

MLP Multi-Layer Perceptron.

NaN Not a Number.

PCA Principal Component Analysis.

RF Random Forest.

SGE Structured Grammatical Evolution.

SOM Self-Organizing Map.

UMAP Uniform Manifold Approximation and Projection.

WHO World Health Organization.

XGB Extreme Gradient Boosting.

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Over the years, large amounts of information have been produced and stored. We need methods to automatically extract knowledge to make good use of this data. Artificial Intelligence (AI), namely Machine Learning (ML), provides a pathway to this end by offering a wide range of methods that look forward to harnessing the knowledge in the data.

In particular, the medical field has generated a vast amount of information that is becoming ever more useful as the personalized medicine field is evolving, tailoring medical solutions to a specific set of patients. Currently, multi-disciplinary teams composed of medical staff, data scientists and engineers are working together to better understand medical problems by studying them from different perspectives, with different backgrounds. This problem-solving paradigm enables the discovery of creative and interdisciplinary solutions.

Audiology is the medical branch that studies hearing, balance and respective disorders. In February 2023, the World Health Organization (WHO) stated that over 5% of the world's population (430 million people) requires rehabilitation to address Hearing Loss (HL) [WHO - Hearing Loss]. The same report states that by 2050 nearly 2.5 billion people will suffer from HL to some degree, with 1 out of 10 people having disabling HL. Therefore, hearing health is of utter importance to society, since HL can negatively impact every person, leading to social interaction deficits, which can by itself jeopardize a person's professional and personal life.

To minimize the problems associated with HL, specialized technicians can conduct hearing screenings to assess the hearing health of people. During these procedures, several pieces of information regarding the patient are collected and stored in a database. Later, one can use this information to create ML models to help guide the screening towards people at risk.

A well-known ML model performance improvement technique is Feature Engineering (FE). This process selects and enhances the original data in a manner that ML models can use to achieve better results, especially those that cannot create a complex internal representation of the given data. Many methods are traditionally used for this task, such as Principal Component Analysis (PCA) or even manual selection by humans. Evolutionary FE methods also show them-

selves as successful, with Genetic Programming (GP) based methods being the most used [Ahmed et al., 2014; Tran et al., 2016a,b; Zhang et al., 2022]. FERMAT [Monteiro et al., 2021], the first Structured Grammatical Evolution (SGE) based FE framework enables the selection and construction of novel features through a Grammar-Based GP engine, given a supervised dataset for a regression task.

This work proposes FEDORA, an Automated Machine Learning (AutoML) framework for FE based on FERMAT, where the ML model used to evaluate the quality of the features being constructed can be specified by the user. The proposed framework can address classification or regression tasks.

The problem that this work addresses is HL detection, by predicting (classifying) if a person has HL based on contextual features, using ML models. The supervised dataset used for the experiment is built from audiology data provided by an audiology company and public HL related data, such as medical and demographic information. The label is a binary variable, classifying each entry as having or not having HL.

An experimental study will be conducted to assess the viability of the proposed framework for the above-mentioned problem, in multiple contexts. The study consists of running the framework with different parameters and proxy models, for a total of 5 experiments. The results will be analysed with regards to performance, Feature Construction (FC) and Feature Selection (FS), and compared with the baseline and other common FE methods. Also, three evolution biasing methods are tested, to address some of the behaviours of the proposed framework. Lastly, FEDORA was added as a pre-processor of the AutoML-DSGE framework [Assunção et al., 2020], which is capable of addressing the Combined Algorithm Selection and Hyperparameter optimization problem (CASH), to understand if their transformations can match the performance of common pre-processors.

1.1 Objectives

This work has 2 objectives. The first one aims to automate the first steps of the ML pipeline (Figure 1.1) by studying FE techniques, namely FS and FC. This objective is fulfilled with the study and development of FEDORA, an Evolutionary FE framework. The goal of the framework is to boost the performance of the models for the given HL detection problem. This is achieved through the selection and construction of novel enhanced features from the original dataset. The level of satisfaction of this objective defines the success criteria of this work and hence will be the primary focus.

The second objective is related to the model selection and corresponding hyperparameter optimization steps in the ML pipeline. This objective is fulfilled by completing the full pipeline with a method that can properly address the CASH and by studying the impact that the FEDORA transformations have in these steps, which aim to maximize classification performance. The method selected to address them is the AutoML-DSGE framework [Assunção et al., 2020], which can select and tune the models of a full ML pipeline.

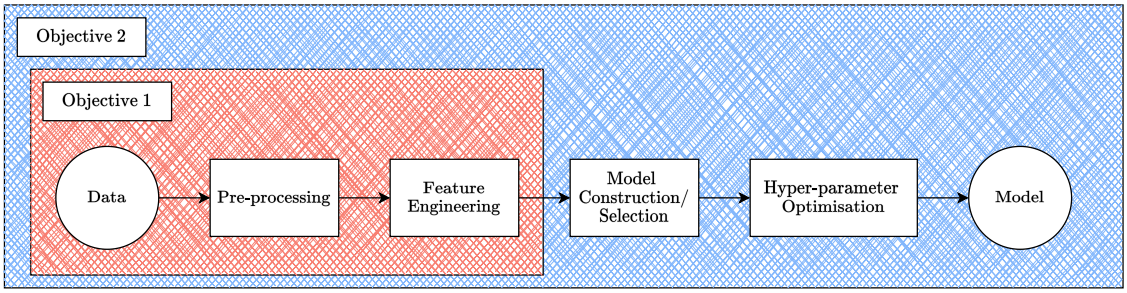


Figure 1.1: Objectives

1.2 Contributions

This work provides multiple contributions. Regarding the problem, this work is focused on detecting HL through the use of only personal, medical and demographic factors that contextualize each person in their corresponding environment, instead of using an audiology screening or similar test, as most works do.

Concerning the approach, the development of an AutoML Evolutionary FE framework contributes to the development of the Evolutionary Computation (EC) and AutoML fields, with regards to FE. Also, the framework is compared with common FE techniques, which allows pragmatic contextualising of the results obtained through this alternative method.

Results show that FEDORA can engineer new features that can improve the performance of the chosen model, with some degree of interpretability. Most of the framework's results are statistically better than commonly used FE techniques, such as PCA, Uniform Manifold Approximation and Projection (UMAP), Self-Organizing Maps (SOMs) and Autoencoders (AEs). Results also point to the choice of the proxy as a decisive factor in the behaviour of the framework, whether in terms of performance or FE.

The framework's best result obtained is 76.2% balanced accuracy using an individual from an Extreme Gradient Boosting (XGB) proxy experiment, with also XGB as the testing model, using 9 total features (5 Original, 2 Engineered and 2 Complex) out of the 60 original ones. When using the least amount of features, the best result is 72,8% balanced accuracy using an individual from a Decision Tree (DT) proxy experiment and a Random Forest (RF) algorithm as the testing model, using 1 single complex feature.

1.3 Organization

Chapter 2 describes the EC and AutoML concepts and methods that this work addresses, as well as the related work. Chapter 3 describes FEDORA, the proposed FE framework. Chapter 4 reports the results of the proposed framework regarding both objectives. Chapter 5 presents the conclusion of this document and points to future work.

Chapter 2

Background

This work aims to solve a Hearing Loss (HL) detection problem through an evolutionary Feature Engineering (FE) framework. Therefore, concepts and methods from Evolutionary Computation (EC) and Automated Machine Learning (AutoML) are required. This chapter overviews both fields and provides related work on this work's problem and approach.

Section 2.1 gives an overview of EC, namely Evolutionary Algorithms (EAs), Genetic Algorithms (GAs), Genetic Programming (GP), Grammatical Evolution (GE) and Structured Grammatical Evolution (SGE). AutoML is described in section 2.2, providing background in Hyper-parameter Optimisation (HPO), FE and Machine Learning (ML) Pipeline Optimization. At last, section 2.3 describes the progress of ML in HL detection and the state-of-the-art of evolutionary FE approaches.

2.1 Evolutionary Computation

EC is a field of Artificial Intelligence (AI) that draws inspiration from Nature. It comprises many optimization algorithms, drawing inspiration from Physics, Darwin's theory of evolution [Darwin, 1859] and Swarm Intelligence, as classified in [Zhang, 2020].

This field addresses the development of meta-heuristic algorithms that do not require much computational effort to reach near-optimal solutions when compared to precisely calculating the global optimum. [Eiben and Smith, 2015]

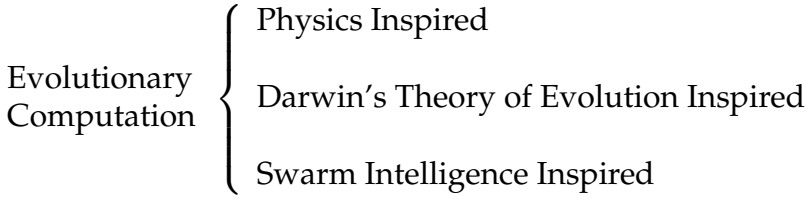


Figure 2.1: Evolutionary Computation Tree. Adapted from [Zhang, 2020]

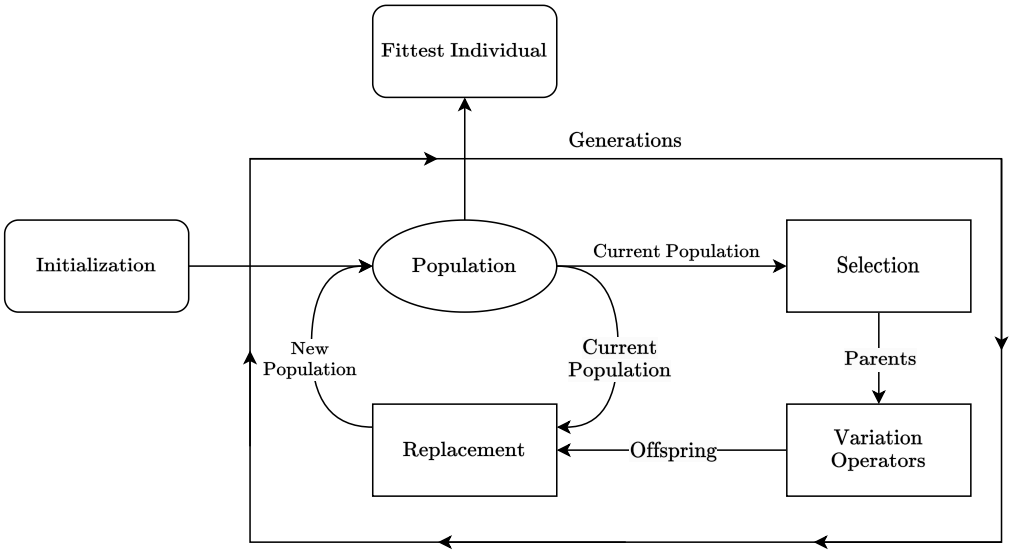


Figure 2.2: Evolutionary Algorithm Cycle. Adapted from [Brabazon et al., 2015]

2.1.1 Evolutionary Algorithms

EAs are a sub-field of EC. An EA is a population-based meta-heuristic, guided by an objective (fitness) function. It is inspired by Darwin's theory of evolution by natural selection - survival of the fittest - and by modern genetics.

In biological terms, EAs utilizes an initial population (set of candidate solutions) that will be modified throughout several iterations. The population will evolve through a given fitness-defined environment as it will be subject to evolution mechanisms, such as selection, variation operators and retention of fit forms [Brabazon et al., 2015], as shown in Figure 2.2.

2.1.2 Genetic Algorithms

GAs are a sub-field of EAs. The GAs contain a genotype-to-phenotype mapping concept, where each individual is evaluated and selected at the phenotypic level while being modified via recombination and mutation at the genetic level. This distinction aims to better simulate Nature's evolution knowledge by incorporating modern genetic discoveries. In modelling terms, it also allows for generic genotypic representations and operators, since the mapping and the fitness function are the only problem-dependent components. Also, the search space is different from the solution space. Algorithm 1 details a GA.

Resembling other EAs, GAs start by initializing a population, in a stochastic manner. This population will be iterated for many generations. In each generation, each individual in the population is evaluated by the fitness function, at the phenotypic level. Next, some will be selected to produce new solutions by a selection operator, such as tournament or roulette wheel. This selection process is stochastic but fitness-biased since the fittest individuals are more likely to be selected. Then, the chosen individuals reproduce, i.e. generate new solutions, through ge-

Algorithm 1 Genetic Algorithm

```
1: procedure GeneticAlgorithm(N_generations, S_population, P_crossover, P_mutation)
2:     population ← initialize_population(S_population)
3:     population ← evaluate(population)
4:     for i ← 1 to N_generations do
5:         parents ← select_parents(population)
6:         offspring ← crossover(parents, P_crossover)
7:         offspring ← mutation(offspring, P_mutation)
8:         offspring ← evaluate(offspring)
9:         population ← survive(population, offspring)
10:    end for
11:    return best(population)
12: end procedure
```

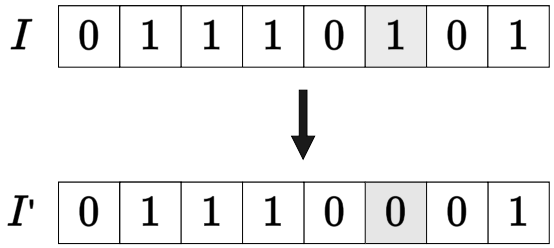


Figure 2.3: Binary Genetic Flip Mutation

netic crossover and mutation operators, producing offspring. Together, the population and the offspring will face a replacement operator that will define which individuals will pass on to the next generation. This operator is usually age-based or fitness-based [Eiben and Smith, 2015]. Figures 2.3 and 2.4 give an example of a mutation and crossover operator, for a binary string representation.

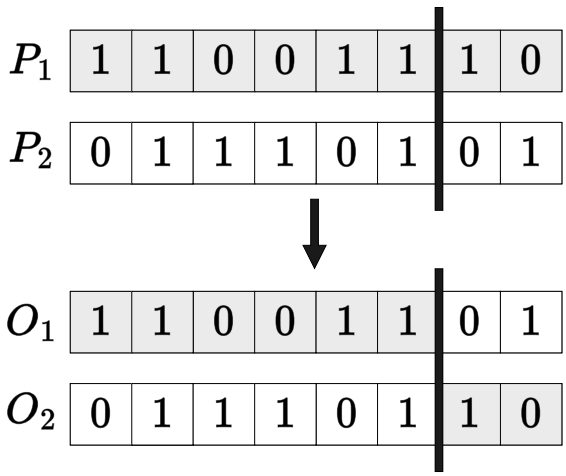


Figure 2.4: Binary Genetic One-point Crossover

The genotype usually consists of a binary string or an array of floats. The crossover and mutation are dependent on the genotype representation. The phenotype consists of a candidate solution for the given problem. Figure 2.5 illustrates the geno-

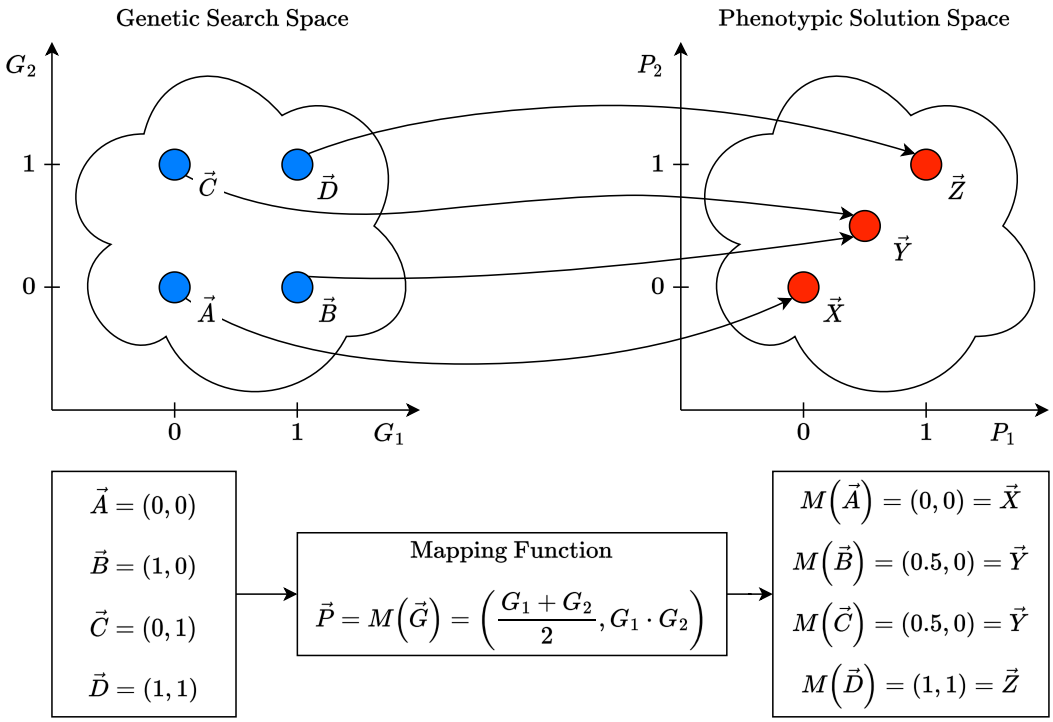


Figure 2.5: Genetic vs Phenotypic Spaces

type and phenotype differences, related to the search and solution space, with an example where two binary genes (G_1, G_2) map into two phenotypic characteristics (P_1, P_2) through the mapping function M.

2.1.3 Genetic Programming

GP [Koza, 1993] is another variant of EAs. The evolution paradigm in GP does not focus on evolving solutions but on evolving executable programs that can find a solution for the problem. Typically, the representation used for this type of EA is an Abstract Syntax Tree (AST), but there are alternatives. In GP the leaves of the AST are Terminals, given by the terminal set T, which comprises variables and constants. On the other hand, the nodes are functions, from the function set F, that must have the property of closure, i.e., must guarantee type consistency and evaluation safety [Poli et al., 2008]. Figure 2.6 illustrates a simple AST representation with $T = \{X_1, X_2, X_3, 0.5, 1\}$ and $F = \{min, max, cos, sen, +, -\}$

As the representation is given by an AST, specific initialization procedures are required. The grow and the full methods are the earliest ones described. Both of them require the definition of a maximum depth parameter for the AST. In the full method, the nodes are chosen from the function set F, until the maximum depth - 1 is reached in every branch. From that point forward, only terminal symbols can be chosen to complete the individual. This method guarantees that every individual will be a complete tree, up to the defined depth. On the other hand, the grow method randomly selects a terminal or a function as it generates the AST. When a branch reaches maximum depth - 1, only a terminal can be

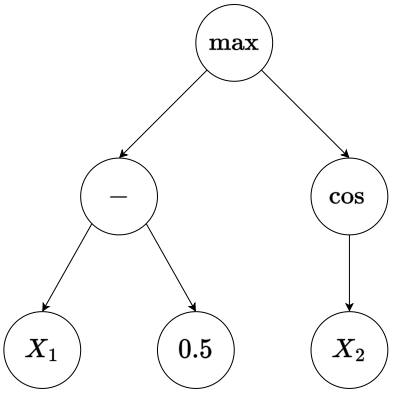


Figure 2.6: Abstract Syntax Tree representing $\max(X_1 - 0.5, \cos(X_2))$

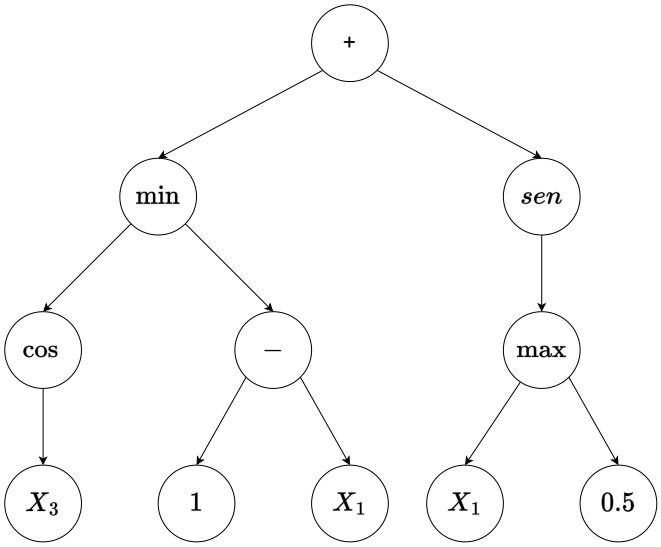


Figure 2.7: Full method

selected onward for that branch. This procedure diversifies the trees as they can be as short as a single terminal or as deep as the maximum depth parameter [Poli et al., 2008], [Brabazon et al., 2015].

Assuming that the AST root is at depth 0, the maximum depth is 3 and the terminal and function sets are the ones given above, an example of the full and the grow method are represented in Figures 2.7 and 2.8.

Ramping is also a technique used to increase the population's diversity. It consists of varying the maximum depth parameter on any given initialization method. Ramped-half-and-half is another initialization method, and the most commonly used [Eiben and Smith, 2015], which utilizes the ramping technique with both full and grow methods, each one initializing half of the individuals in the population. This strategy ensures diversity in terms of structure and content. [Brabazon et al., 2015]

As initialization, variation operators are also GP-specific. Figures 2.9 and 2.10 depict a sub-tree crossover and a sub-tree mutation, respectively.

One subject of intense study over the years is a phenomenon labelled as bloat

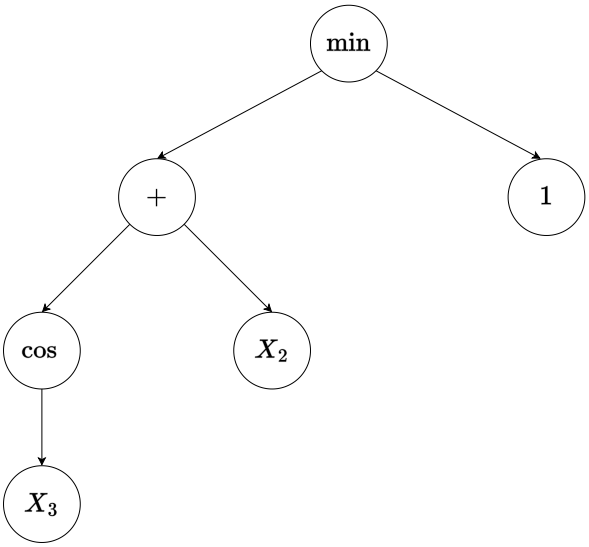


Figure 2.8: Grow method

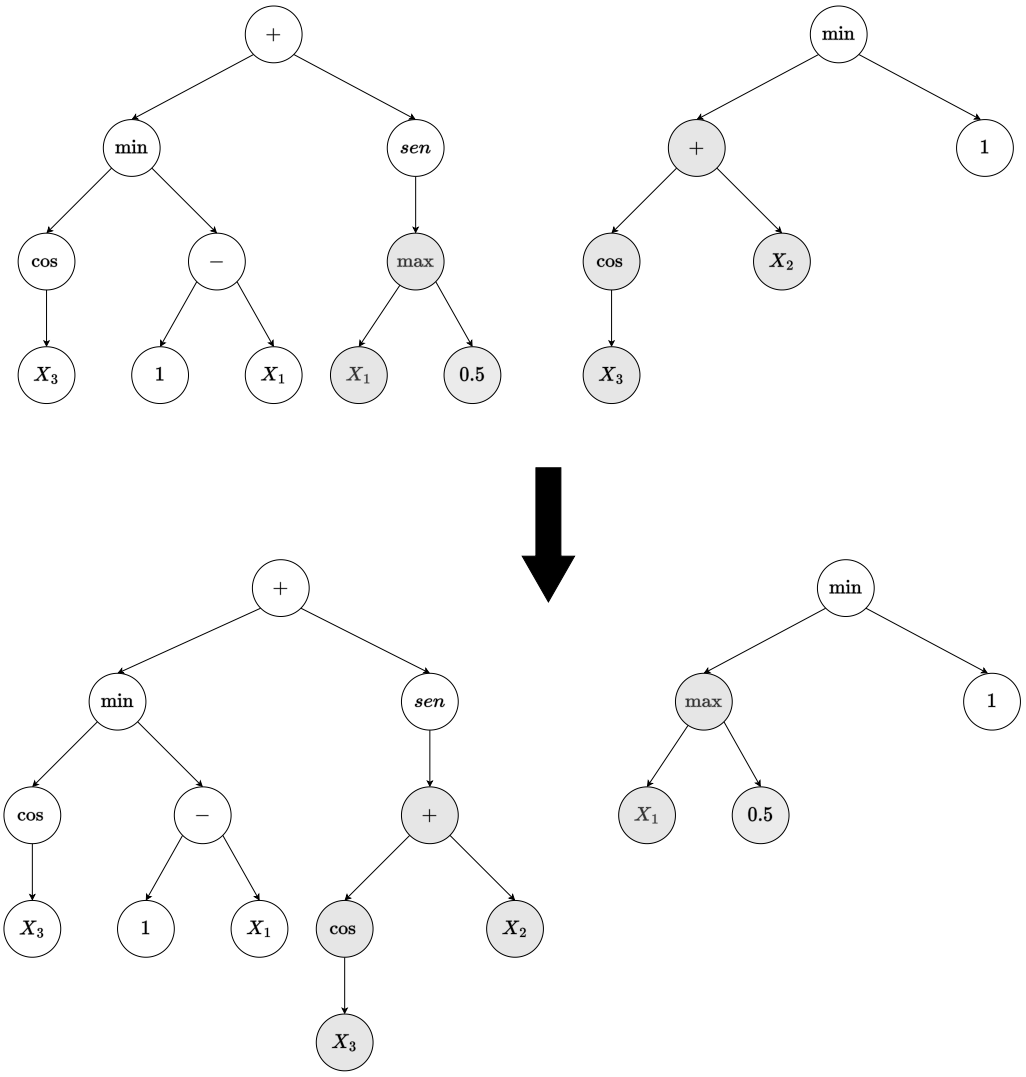


Figure 2.9: Sub-tree Crossover

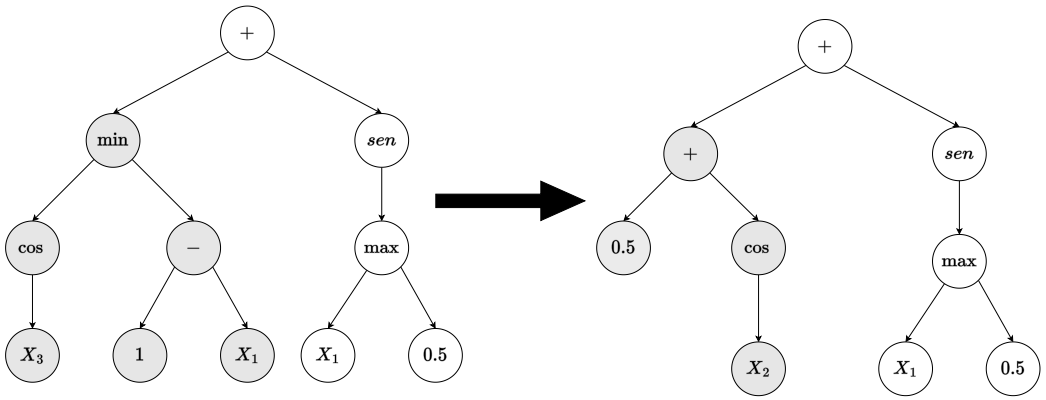


Figure 2.10: Sub-tree Mutation

[Poli et al., 2008]. Bloat happens when the average number of nodes in a population starts growing at a very fast pace, while not being accompanied by a corresponding fitness increase. As programs grow in size, more resources are needed to face up to the complex requirements. As such, these individuals become more computationally expensive to evolve and evaluate. There are several methods proposed to counteract bloat, but the most widely accepted is parsimony pressure, which introduces a term in the fitness function that decreases the fitness of larger individuals. [Eiben and Smith, 2015]

2.1.4 Grammatical Evolution

GE [O'Neill and Ryan, 2001] is a GP variant. Unlike standard GP, GE separates the genotype, a linear string of integers, from the phenotype, an executable program defined by an AST structure. As a Grammar-Based Genetic Programming (GBGP) approach, GE relies on a Context-Free Grammar (CFG) to define the genotype-to-phenotype mapping rules. A grammar can be defined by the tuple $G = (T, N, S, P)$ where T is the set of Terminals, N is the set of Non-Terminals, $S \in N$ acts as the start symbol, or axiom, and P is the set of production rules such that $\alpha \rightarrow \beta$, where $\alpha \in N$ and $\beta \in (T \cup N)$*. The language L defined by this grammar G is given by $L(G) = \{w : S \xrightarrow{*} w, w \in T^*\}$, which defines every possible word w that the grammar can produce, i.e. which programs can be structured by the algorithm. Therefore, GE addresses the closure requirement of GP by constraining how the programs are built through a CFG. Figure 2.11 illustrates a CFG in the Backus-Naur Form (BNF).

The mapping is performed by repeatedly getting an integer i from the linear genome (in the case of a binary string, 8 bits are used to generate a corresponding integer value). Each integer value is then used to select which production rule will be chosen from the grammar, by a modulo operation such as:

$$\text{i } \textbf{MOD } N_{rules}$$

The mapping ends when a word $w \in L(G)$ is built. If it runs out of integers in the genotype, it wraps around to the beginning of the genome. This method

$$N = \{< \text{start} >, < \text{ expr} >, < \text{op} >, < \text{var} >\}$$

$$T = \left\{+,\ -,\ \cdot\ , /, (\ ,\ )\ , X_1,\ \ldots\ ,\ X_{N_{features}}\right\}$$

$$S = < \text{start} >$$

P :

$$\begin{aligned}
< \text{start} > ::= &< \text{expr} >, \\
&\mid\ < \text{expr} >, < \text{expr} > \\
&\mid\ < \text{expr} >, < \text{expr} >, < \text{expr} > \\
&\mid\ \ldots \\
&\mid\ < \text{expr} >, \ldots, < \text{expr} > \\
< \text{expr} > ::= &< \text{expr} >< \text{op} >< \text{expr} > \\
&\mid (< \text{expr} >< \text{op} >< \text{expr} >) \\
&\mid\ < \text{var} > \\
< \text{op} > ::= &+ \mid\ - \mid\ \cdot\ \mid / \\
< \text{var} > ::= &\ X_1 \mid X_2 \mid X_3 \mid \ldots \mid X_{N_{features}}
\end{aligned}$$

Figure 2.11: Example of a Backus-Naur Form Grammar. Adapted from [Monteiro et al., 2021]

is known as wrapping. Figure 2.12 provides a simple mapping example for the grammar depicted in Figure 2.11, for $N_{features}$ = 4.

As in GAs, the separation of the genotype from the phenotype allows for generic genotypic representations and operators. Also, the initialization of GE is directly linked with the genetic representation. Instead of a simple random initialization, GE should be initialized by a method that properly ensures diversity. An alternative method could be sensible initialization, which uses the ramped-half-and-half approach of standard GP and applies it to GE. [Ryan et al., 2018]

In terms of performance, GE was compared to GP in a set of problems, outperforming GP in 2 out of 3. [O'Neill and Ryan, 2001]

As for drawbacks, GE has known redundancy and low locality issues [Rothlauf and Oetzel, 2006], [Lourenço et al., 2016]. Two genetically different individuals are labelled as redundant if they map into the same phenotype. This phenomenon can be witnessed in Figure 2.5. Although being a known Nature mechanism to ensure robustness, given that it neutralizes some mutations, too much redundancy in the population has practical consequences, since it slows down evolution processes and therefore declines the algorithm's efficiency. An individual has a low locality if a small genetic variation does not relate to a small phenotypic one. This is the most concerning problem since if it is not addressed, GE's performance will come near random search performance.

Linear Genotype	12	67	95	34	50	98	1	17

Non-Terminals	< start >	< expr >	< op >	< var >
Number of Rules	4	3	4	4

< start >	$12 \,\%\, 4 = 0$
< expr >	$67 \,\%\, 3 = 1$
(< expr >< op >< expr >)	$95 \,\%\, 3 = 2$
(< var >< op >< expr >)	$34 \,\%\, 4 = 2$
(X_3 < op >< expr >)	$50 \,\%\, 4 = 2$
($X_3 \cdot$ < expr >)	$98 \,\%\, 3 = 2$
($X_3 \cdot$ < var >)	$1 \,\%\, 4 = 1$
($X_3 \cdot X_2$)	*end*

Figure 2.12: Grammatical Evolution Mapping Process

2.1.5 Structured Grammatical Evolution

SGE proposed in [Lourenço et al., 2015] and later enhanced [Lourenço et al., 2018], is a GE variant that addresses the redundancy and low locality issues of GE. With a novel representation, where each gene is bound to a specific non-terminal of the grammar, with a list of integers assigned to each non-terminal, SGE ensures that a genetic modification does not affect the derivation path of other non-terminals, thereby narrowing the number of phenotypic changes that could have occurred otherwise. Formerly, the size of each list is given by computing the maximum number of possible expansions for each non-terminal. The maximum number of derivation choices for the associated non-terminal sets a limit on each gene's value. As an example, for a simple grammar (adapted from [Lourenço et al., 2015]):

$$< \text{start} > ::= < \text{char} > \mid < \text{char} >< \text{char} >< \text{char} >$$
$$< \text{char} > ::= \text{a} \mid \text{b} \mid \text{c}$$

The set of non-terminals N is $\{< \text{start} >, < \text{char} >\}$, hence the genotype comprises two lists, each one linked to a non-terminal symbol. The maximum number of possible expansions for <start> is one since it appears only as the axiom. On the other hand, three is the value for the <char> non-terminal, since there is a derivation path where this symbol appears three times. Therefore, the list linked to <start> is one gene long, while the <char> list has a length of three. Finally, the <start> symbol has two derivation alternatives, while the <char> has three. This implies that the corresponding lists' genes will take values from zero to two and from zero to three. A SGE genotype example is provided in Figure 2.13.

The genotype-to-phenotype mapping of Figure 2.13 is illustrated in Figure 2.14.

Non-Terminals	< start >	< char >
Genotype	[1]	[0, 1, 2]

Figure 2.13: Structured Grammatical Evolution genotype example

< start >	[[1], [0, 1, 2]]
< char >< char >< char >	[[], [0, 1, 2]]
a < char >< char >	[[], [1, 2]]
ab < char >	[[],[2]]
abc	[[],[]]

Figure 2.14: Structured Grammatical Evolution mapping

Concerning variation operators, SGE mutates the individuals by randomly replacing some non-terminal gene with a new integer. Crossover is performed using a binary mask with a length equal to the number of non-terminal symbols. Figure 2.15 depicts the crossover process.

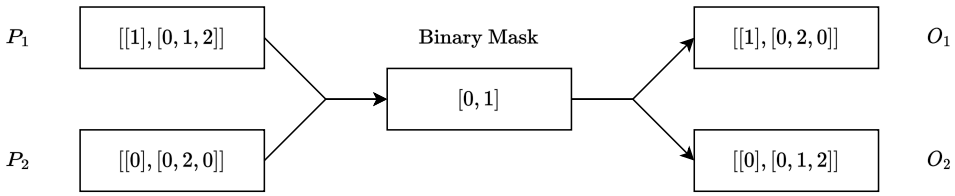


Figure 2.15: Structured Grammatical Evolution crossover

A problem arises when the grammars have recursive production rules, such as in the grammar of Figure 2.11, which was previously solved by defining a maximum recursion depth and pre-processing the grammar to remove the recursion. The current version of SGE [Lourenço et al., 2018] deals with this problem by generating the required genes on the fly, only requiring the definition of the maximum tree depth. This implies that the lists now have a dynamic size rather than a fixed one. It constrains the size of the trees by only allowing non-recursive derivation rules to be chosen for expanding the current non-terminal, once the trees reach the depth limit.

Performance-wise, SGE has reportedly outperformed GE [Lourenço et al., 2016] [Lourenço et al., 2017], as it can effectively reduce high redundancy levels and address low locality issues of it.

2.2 Automated Machine Learning

AutoML is a field of ML that aims to provide hands-free solutions by completely removing humans from the ML pipeline. This enables ML to be accessible to non-

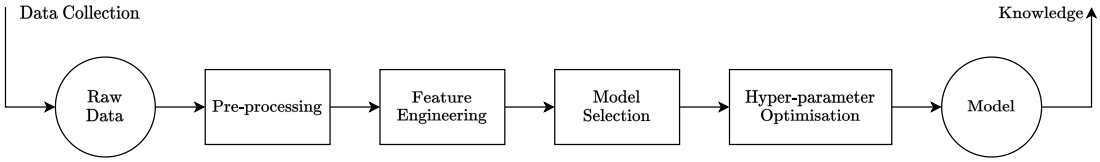


Figure 2.16: A Simple Machine Learning Pipeline. Adapted from [Assunção et al., 2020]

experts and to systematise its practice, which usually is an *ad hoc* process. Current AutoML methods can outperform human ML experts in some tasks [Hutter et al., 2019]. AutoML research revolve around the traditional ML Pipeline. The main ones can be summarized (Figure 2.16) as Data Collection, Data Clean-up, FE, Model Selection and Hyper-parameter optimization.

Besides the combinatorial complexity related to the problems, algorithms and parameters in this field to automatically extract some data-driven knowledge, explainability in AutoML also presents itself as a challenge. Explainability refers to how the AutoML method's choices relate to its results. It is essential because, by explaining the models, a human may be able to comprehend the fundamental decision criteria, advancing human knowledge discovery.

2.2.1 Hyper-parameter Optimization

HPO is one of the earliest and most basic, yet complex, tasks in AutoML. It aims to tune the parameters of a given method to its increase performance. Regarding the HPO methods, grid and random search are the most basic model-free black-box optimization methods. They rely on a grid and stochastic sampling of the search space, respectively. Other methods, such as Bayesian optimization and EC related, have also been applied successfully to HPO problem instances [Feurer and Hutter, 2019]. Figure 2.17 illustrates grid and random search approaches in a bi-dimensional search space.

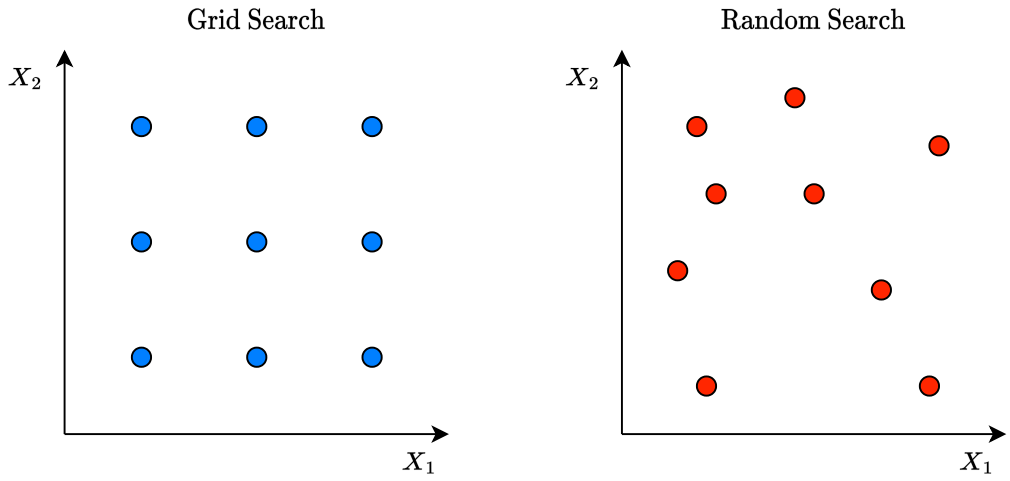


Figure 2.17: Grid Search vs Random Search

2.2.2 Feature Engineering

Feature Engineering (FE) is a step in the ML pipeline, as well as an AutoML topic of interest for this work, where a pre-processed data set undergoes selection and transformation processes. FE can be sub-divided into two fields: Feature Selection (FS) and/or Feature Construction (FC).

Both FS and FC can be divided into three categories: filter, wrapper and embedded methods [Cherrier et al., 2019]. Filter methods are model-free ranking methods, as they do not require a model to rank the features by relevance. Wrappers assess a model's performance to evaluate which set of features is relevant to the problem at hand. Due to the use of heavier techniques, wrapper methods usually outperform filter methods, although being much slower. At Last, embedded methods combine both virtues of filter and wrapper methods, as they perform FE while training the model.

The intent of FS is to reduce the dimensionality of the data set to remove redundant or misleading data that can negatively affect the models' performance. From a practical standpoint, it also reduces the training time, as less data is required. The resulting data set is a subset of the original one.

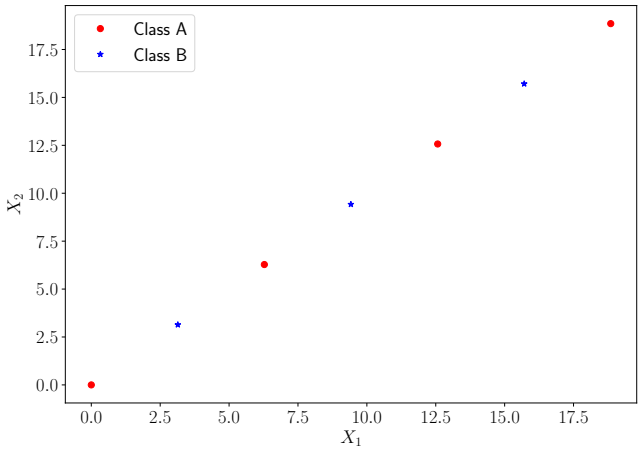
The goal of FC is to build novel features from the original data set D, building a new data set D'. The main goal of the newly generated features is to enhance the models' performance, namely those that suffer from a lack of complex internal representation of the data. Figures 2.18a and 2.18b show how a cosine transformation of one original feature distinguishes between both classes in the resulting data set. The data set is provided in table 2.1.

2.2.3 Machine Learning Pipeline Optimization

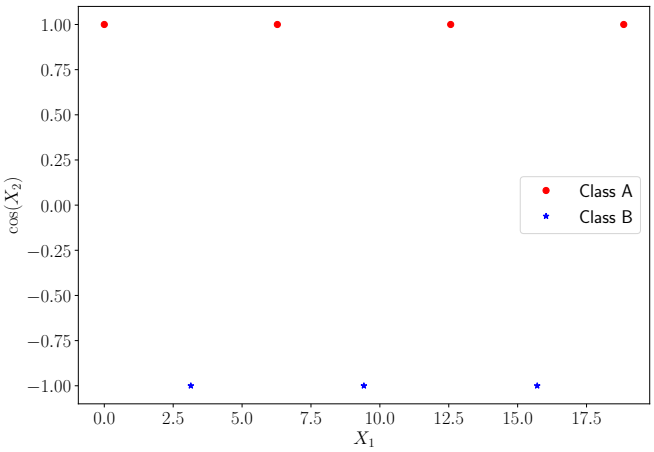
Machine Learning Pipeline Optimization is a research branch of AutoML that aims to automatically choose the best-performing algorithms (pre-processing, predictors) and the corresponding hyper-parameters for a given problem. The simplest form comprises selecting, training and testing a small range of predictors and picking the best one. However, this method becomes too expensive as more algorithms are added to the search space and when the hyper-parameter search space becomes too large. As such, more intelligent search methods are needed. Auto-WEKA [Kotthoff et al., 2019] addresses the Combined Algorithm Selection and Hyperparameter optimization problem (CASH), also known as Full Model Selection (FMS), by a Bayesian optimization technique. Auto-sklearn is another Bayesian optimization-based method [Feurer et al., 2019]. As for EC approaches, TPOT is a GP pipeline optimization technique [Olson and Moore, 2019]. Another EC example is AutoML-DSGE [Assunção et al., 2020], which also addresses this problem, this time through the use of SGE as the search engine.

Table 2.1: Data set

Class	X_1	X_2	$\cos(X_2)$
A	0	0	1
B	π	π	-1
A	2π	2π	1
B	3π	3π	-1
A	4π	4π	1
B	5π	5π	-1
A	6π	6π	1



(a) Original data set (X_1, X_2)



(b) Transformed data set $(X_1, \cos(X_2))$

Figure 2.18: Feature Construction Example

2.3 Related Work

For this work, there are two main topics worth covering in terms of related work: Advances in ML for HL detection and FE related tasks using EC.

2.3.1 Machine Learning in Hearing Loss Detection

In Miranda [2022] an overview of the current status of ML in HL detection is made. In terms of features, attributes such as age, gender, demographics, medical data and duration and intensity of noise exposure are the most insightful in the performed experiments. These attributes are also pointed out by the World Health Organization (WHO) as relevant HL causes. Images, such as magnetic resonance scans and otoscopy images [Byun et al., 2022], have also been used with Deep Learning to address HL detection.

From regular ML algorithms, such as K-Nearest Neighbors, Support Vector Machines and ensemble methods, applied to standard ML datasets to Convolutional Neural Networks (CNNs) applied to magnetic resonance scans, HL detection has been tried in many ways, successfully performing well above simple random classification. Performance metrics are hard to compare as the datasets and the metrics differ between works.

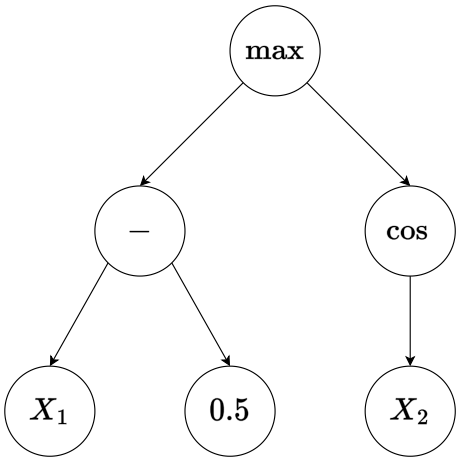
Despite the progress made in this research area, its focus has not been on detecting HL through the use of only personal, medical and demographic factors that contextualize each person in their corresponding environment. Instead, the focus of the above-mentioned studies is set on actively detecting HL, through the use of ML models that act upon audiology screening or similar tests. Often, HL detection studies address particular types or causes of HL, such as sensorineural or noise-induced, respectively, or places where people are prone to losing hearing, such as industrial environments [Tomiazzi et al., 2019].

2.3.2 Evolutionary Feature Engineering

Evolutionary FE has been a research topic over the years. Most works rely on GP-based methods to engineer novel features using different approaches in terms of representation such as single-tree or multiple-tree [Zhang et al., 2022].

In single-tree representation, each individual is represented by an AST where the leaves are the original features, while the nodes act as the transformation functions. This type of representation allows for single or multiple generated features, as the AST's sub-branches may be engineered features themselves. A random subset of the generated features set is chosen for the transformation, as shown in Figure 2.19.

Ahmed et al. [2014] use a single-tree representation to evolve multiple features. It generates engineered features from the ASTs' root and sub-trees and selects them based on a method to measure discriminating information between the classes,



(a) Abstract Syntax Tree

Possible Subtree Features
0.5
X_1
X_2
$X_1 - 0.5$
$\cos(X_2)$
$\max(X_1 - 0.5, \cos(X_2))$

Possible Transformation
$\{X_1, X_1 - 0.5\}$

(b) Possible subtree features

Figure 2.19: Single-tree feature engineering example

namely a fitness function based on the Fisher criterion and the one-way analysis of variance p-value.

In Tran et al. [2016a] each individual of the proposed method also has a single-tree representation. The novel features are generated by six different criteria: The first one uses the root of the tree to construct a feature. The second incorporates the first and adds the first n features, where n is given by the number of leaves (selected features). The third is the set of leaves (selected features). The fourth is a union of the first and the third. The fifth is given by every possible sub-tree, as well as the root. Finally, the sixth is a union of the third and fifth criteria. Evolution is guided by the average balanced accuracy metric of a classification made by the individual itself. Also mentioned in the above work, another way to generate engineered features may rely on using a specific function from which the resulting sub-tree is selected as an engineered feature.

On the other hand, in a multi-tree representation, each individual is represented as a list of ASTs. This design option allows for more FE choices but it has a computational cost trade-off. Figure 2.20 gives a feature engineering example of a multi-tree representation. It considers using each whole AST as a feature. From the 2 available trees, it selects a random subset of them to generate the transformation. In the given example, it selects both.

In MultGPFC [Tran et al., 2016b], each GP individual is encoded as a list of ASTs. Each constructed feature is related to one of those trees using the root node. To engineer m features, an individual representation must contain m trees. Then, by combining a filter and a wrapper FE method, the fitness function components are defined by the Czekanowski distance metric, as the filter, and the performance of a Decision Tree (DT), as the wrapper.

In Cherrier et al. [2019] work, both GP representations are presented. For an individual to engineer m features, its representation will be of m trees. This work also presents relevant explainability design choices, such as type-based grammar, ensuring dimensional consistency and a probabilistic transition matrix, which biases the choices on the grammar, introducing domain knowledge.

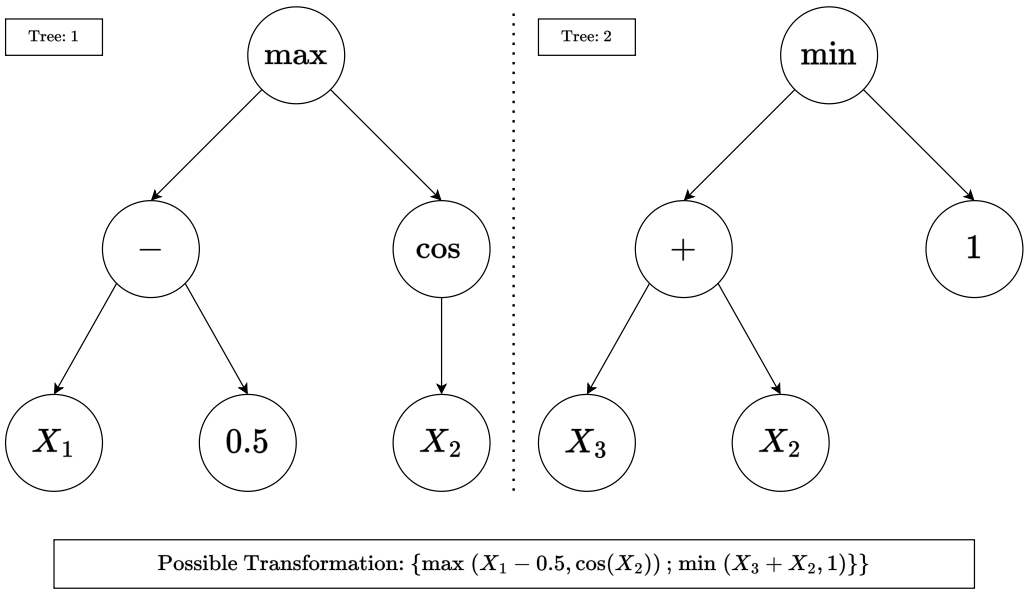


Figure 2.20: Multi-tree feature engineering example

The work of Zhang et al. [2022] uses a multi-tree representation for a regression problem, where each AST root represents an engineered feature. As such, each individual implicitly represents a set of engineered features. Each individual is evaluated by a 5-fold cross-validation scheme using a DT model. The resulting decision trees are oblique, i.e. they divide the original space by oblique hyperplanes since they are trained using the engineered features. The evaluation score is given by a fitness vector, using the absolute error metric for each train data entry. The selection is made by the lexicase selection operator. The best GP individuals are stored in an archive of a predetermined size. Once it reaches that size, the worst are replaced by ones that outperform them. At the end of the evolution, GP individuals are used to build DT models. The combination of these models forms the regression forest.

Some works use GE as the search algorithm. The work of Gavrilis et al. [2008] provides a GE-based wrapper FE framework where the evolution is guided by accuracy or negative mean squared error metrics, for classification and regression tasks, respectively. Each GE individual corresponds to k features. The genotype-to-phenotype mapping is done by splitting the genotype into k equal components and mapping each component into a function for a novel feature through the given grammar. Thereby, the phenotype is a function of the transformed data set.Tsoulos et al. [2022] applies a similar framework to predict COVID-19 mortality, this time with a Radial Basis Function acting as the wrapper's model. There is a second stage, where the best individual is applied to the data set, obtaining the transformed data set. Then a neural network is evolved by a GA using that data set.

Lastly, SGE has also been used to FE. FERMAT [Monteiro et al., 2021] is a wrapper FE framework that uses SGE as the search algorithm to engineer an enhanced data set, aiming to improve the performance of the test models. Each individual consists of a set of feature transformations. The population is evaluated by a DT that acts as a proxy for a Random Forest (RF), the testing model. FERMAT has

shown being able to create insightful novel features and at the same time perform FS, for 2 regression problems.

Chapter 3

The FEDORA Framework

FEDORA is a wrapper Feature Engineering (FE) framework based on FERMAT, where the Machine Learning (ML) model used to evaluate the quality of the features being constructed, i.e. the proxy model, can be specified by the user. Each phenotype of an individual consists of a set of selected or constructed features, i.e. a transformation of the original dataset.

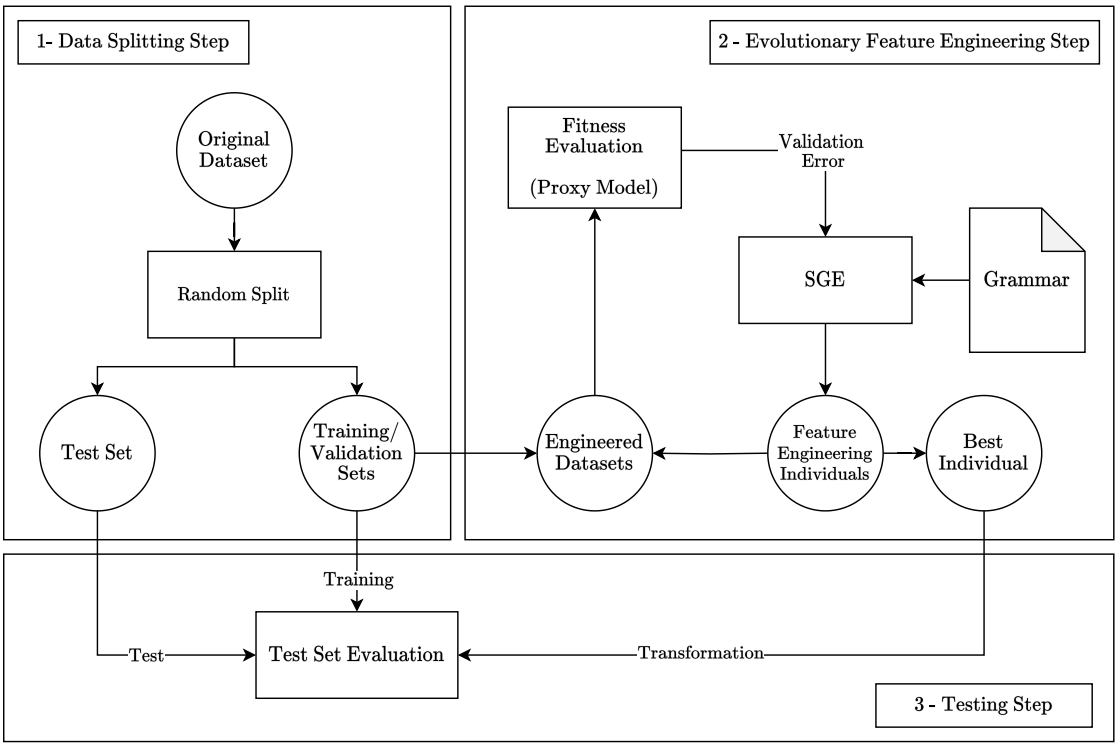


Figure 3.1: The FEDORA framework

The FEDORA framework is described in Figure 3.1. The first step (Data Splitting) consists of randomly splitting the original dataset into training, validation and test subsets. This split is made once in each run and is not modified throughout it.

The training and validation sets are passed to the second step, the Evolutionary

Feature Engineering step. In this step, Structured Grammatical Evolution (SGE) searches for the best individual's phenotype, i.e. the transformation that makes the proxy model provide the best validation performance, based on any selected metric. The search space is defined by the user through a problem-specific grammar that details which individuals the SGE algorithm can generate. Every transformation (individual) is applied to the training and validation sets, resulting in engineered training and validation sets. The proxy model fits the engineered training set and predicts the labels of the engineered validation set. These predictions are used to calculate the validation set error, based on any given error metric. The quality of each individual is given by this error value, which means that the individuals with the lowest validation error are the ones who have better (lower) fitness and are more likely to survive and pass their genes to the next generation. This process is repeated for the defined number of generations in the SGE parameters. At the end of this step, the individual that achieved the lowest validation error is passed to the third and last step, the testing step.

In the testing step, the transformation of the best individual is given as input to the test set evaluation method, alongside the training, validation and test subsets. Any method capable of performing the test set evaluation, given the mentioned inputs may be applied here. The simplest one is to apply the transformation to the whole dataset, resulting in engineered training, validation and test sets. The engineered training and validation sets are then used to fit a ML model, while the engineered test set is used to assess its generalization capability to unseen data. This step is not directly related to FE but it is needed to complete the ML pipeline.

3.1 Implementation Details

Although having outstanding results in regression problems [Monteiro et al., 2021], the FERMAT framework was not properly implemented to withstand parameterisation for classification problems, nor was its configuration simple and modular. As such, FERMAT does not have an Open-Source instance.

Since FEDORA is based on FERMAT, the first effort made to build towards the FE framework that this work proposes, was an adequate implementation of FERMAT and corresponding validation on the same problems. The architecture is similar to the original, although having some slight modifications. First, numerical attributes are bounded to a 32-bit float range, as some of Scikit-Learn's components were designed to work with this format. Hence, values below or above this range are capped at the corresponding boundary. Not a Number (NaN) values, obtained from some operator result, are mapped to 1. This assures that every individual is valid in the evaluation step, and there is no need to purge the invalid ones, by assigning them the worst possible fitness value. Also, the given run seed is applied to every component of the framework, such as the data splitting function, SGE and to the random state hyper-parameter of Scikit-Learn models.

The obtained results were consistent with the ones described in the FERMAT paper. Therefore, an experimental study was conducted to check the viability of the FEDORA framework, by allowing the user to specify different proxy and

testing models, on a classification problem. This study is conducted in Chapter 4. To allow for the use of scale-dependent proxy models, the fitness evaluation of an individual starts by standardizing the transformed dataset.

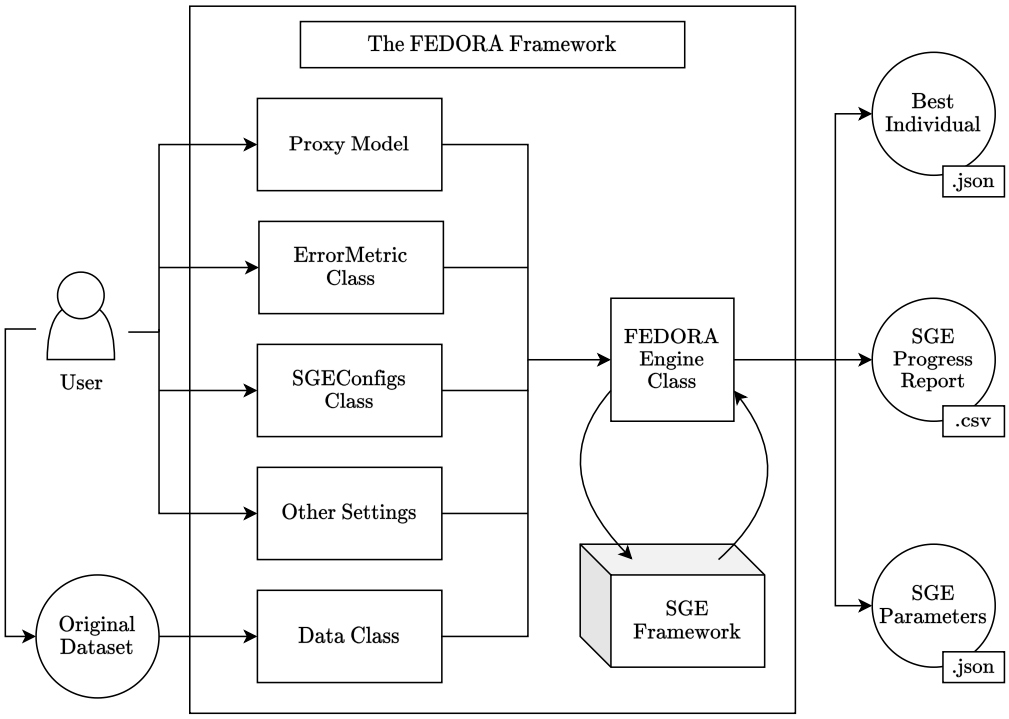


Figure 3.2: FEDORA Implementation Details

The current version of the FEDORA framework is illustrated in Figure 3.2. It takes as input the original dataset, the number of runs and corresponding seeds, SGE parameters (where the grammar path is included), the fitness error metric, the proxy model and an optional fitness penalty function that receives as argument the fitted proxy model. As output for each run, the framework gives out two JSON files, one containing the phenotype of the individual with the best validation score, while the other gives the SGE parameters, for logging purposes. The SGE progress report file is also provided, which includes the best, the mean and the standard deviation fitness of the population, and other generational attributes. The Python 3 code to run the framework is presented in Algorithm 2.

Algorithm 2 FEDORA Instantiation in Python 3

```
1: configs = {
2:         "proxy_model": DecisionTreeClassifier(),
3:         "error_metric": ErrorMetric.balanced_accuracy,
4:         "sge_configs": SGEConfigs.audiology,
5:         "seeds": range(30),
6:         "data": Data.audiology(),
7: }
8: fedora = Fedora(**configs).run(runs=30, name="experiment_name")
```

Chapter 4

Experimental Study

An experimental study was performed to compare the proposed framework in multiple contexts to address both objectives of this work. Firstly, Section 4.1 describes the problem and the full process to obtain the data needed to address it. Concerning the first objective, i.e. automating the first steps of the Machine Learning (ML) pipeline by studying Feature Engineering (FE) techniques, the FEDORA framework was compared with common FE methods, such as Principal Component Analysis (PCA), Uniform Manifold Approximation and Projection (UMAP), Self-Organizing Maps (SOMs) and Autoencoders (AEs), with regards to performance and FE, in Section 4.2. Evolution biasing ideas were also tested in this section. Regarding the second objective, to complete the full ML pipeline, the AutoML-DSGE framework was used to address the Combined Algorithm Selection and Hyperparameter optimization problem (CASH) in section 4.3. FEDORA was added as a possible pre-processor to understand if the transformations can match the performance of common pre-processors.

4.1 Hearing Loss Detection Dataset

The problem that this work addresses is Hearing Loss (HL) detection using contextual attributes. We will use an audiology dataset that was built using personal, demographic and medical data from Portugal. This data was collected from an audiology company and trusted databases, namely PORDATA [PORDATA] and the Portuguese National Health System (SNS) open-data initiative [SNS - Portal da Transparência]. The final dataset comprises 60 features and 25398 entries. The label is a binary variable, classifying each entry as having or not having HL, based on an audiology screening, where people that have at least one ear above the 40dB threshold were diagnosed with HL. Each sample is characterized by 16 boolean variables and 44 real values. The dataset labels can be considered balanced, due to 58% positive and 42% negative percentages.

The full process of building this dataset is documented in [Miranda, 2022]. In the former work, Hybrid Tree Evolutionary Algorithm (HyTEA) was proposed and this problem was addressed for the first time, using the above-described dataset,

achieving an accuracy of approximately 73%. It is the only available reference of this problem in the literature since the dataset is built from private sources and thus cannot be disclosed publicly. This section summarizes the process of identifying, obtaining, structuring and transforming the data to address this problem.

4.1.1 Data Collection

As in any supervised ML problem, the first step in the pipeline is data collection. In our case, for the HL detection problem, an initial study was made to understand which factors have an impact on HL in the scope of the Audiology for All (A4A) research project. From it, medical, demographic and personal data was highlighted as the more influential.

The following step consists in finding credible sources of data. As the data provided by the main stakeholder for the A4A research project was mostly from Portuguese people, and corresponding hearing loss screenings, the search for sources of data was therefore biased. [PORDATA], a certified Portuguese statistics database, and [SNS - Portal da Transparência], a Portuguese National Health System (SNS) open-data initiative, were hence selected as the sources for demographics and medical data, correspondingly.

Finally, the data needs to be properly stored and organized to be worked upon, namely through a database. Data related to ageing, diabetes, blood pressure, stroke fatality, ENT exams, education levels, wages, number and turnover of companies were associated with a specific date and geographical location. A Portuguese county (*Concelho*) was selected as the location granularity level. It is through this location granularity that data from the mentioned public sources and the main stakeholder are linked. This Extract, Transform and Load (ETL) process results in the Entity-Relationship (ER) diagram depicted in Figure 4.1.

A total of 25398 screenings were stored in the database. Screenings data was collected between January 2020 and June 2021. Each screening contains anonymized data from the patient, such as the birth date and address, and from the screening-related setup, such as the screening device, type of headphones and geographical place where the screening took place. It also contains data from the screening itself, such as the hearing level of both ears and the frequencies used.

4.1.2 Data Pre-Processing

The next step in the pipeline is Data Pre-processing. It consists of properly transforming the structured data into a format which can be given to a ML algorithm. This transformation process is fully detailed in [Miranda, 2022]. To link the public data to the stakeholder's data, the attributes of hypertension, diabetes, stroke and ENT exams tables were decomposed into 7 features each: mean, standard deviation, minimum, maximum and quartiles. This was made due to the multiple temporal measurements of each variable for the same place. This way, the resulting features overcome this issue, since time is no longer a dimension. For

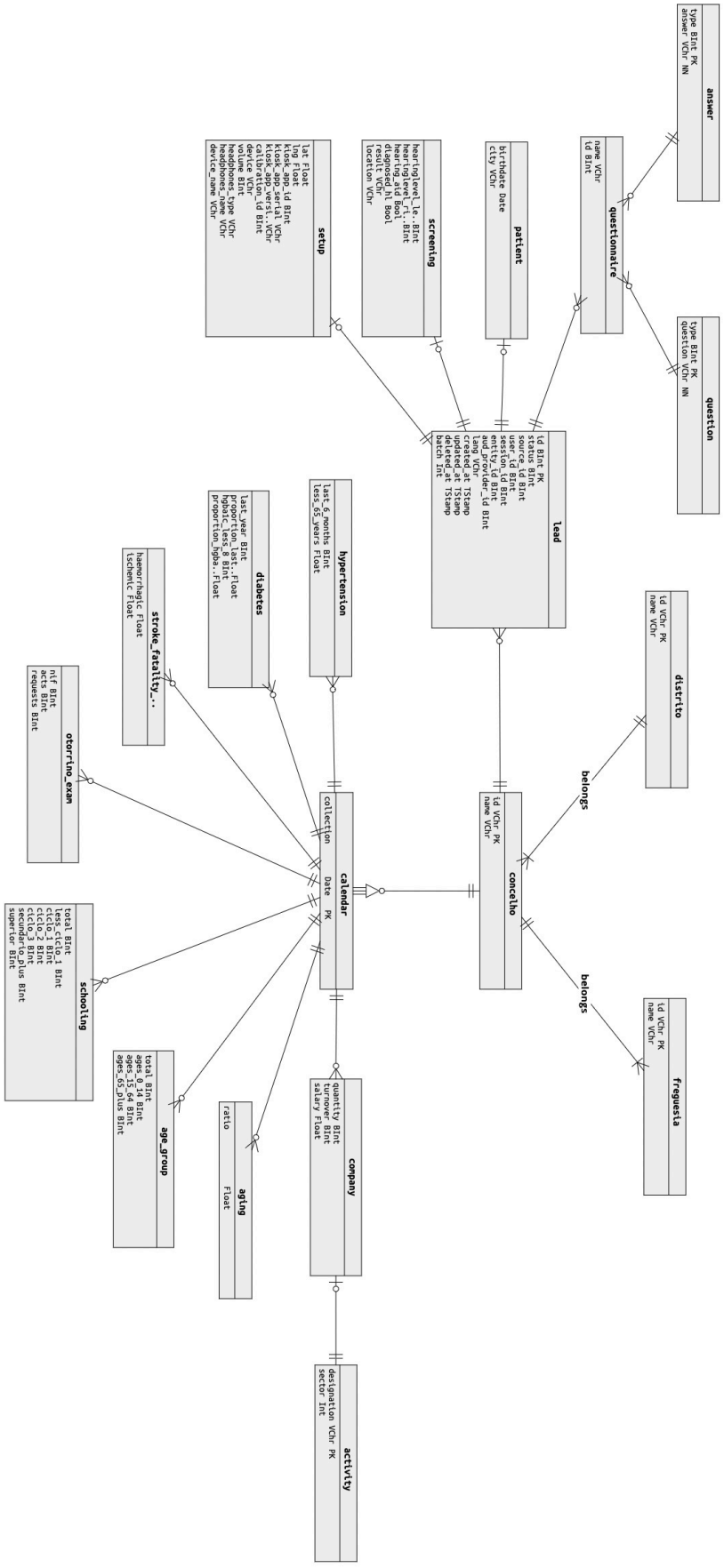


Figure 4.1: Entity-Relationship Diagram

schooling, age group, ageing and company tables, as the temporal measurements were at maximum four by table attribute, a single value was selected from each one. Company data was previously split by type of economic activity, hence each type provided a value, from each corresponding temporal measurement available. Questionnaire features were transformed by One-Hot Encoding and Ordinal Encoding methods. At last, missing values were replaced by the mean of the corresponding features.

The supervised labels were defined by a threshold of 40 dB. People that have at least one ear above the threshold were considered to have hearing loss. Hence, the screening audiometry data was removed since it positively biased the models to always correctly classify each data point, due to the definition of the label being derived from them.

From this process, 204 features were obtained. 42% of the labels (10667 entries) were defined as having HL. A study was then carried on to assess class discrimination and redundancy issues of the features. Concerning class discrimination, feature mean differences, of each class, were statistically tested by an independent t-test at a 95% confidence level. Features that had no statistically significant mean differences were therefore removed. Correlation metrics between the features and the labels were also considered. Pearson, Kendall and Spearman correlations were used and the most correlated features for each method were stored, resulting in sixteen features always appearing in the top 50. Also, Analysis of Variance (ANOVA) F-values were calculated and the features were sorted based on these values. Regarding feature redundancy, features were grouped in consonance with their original database table. A maximum correlation threshold between features of the same group was defined at 80%. Features that did not meet this requirement were removed sequentially, as features with higher ANOVA's F-value were preferred not to be removed. Finally, the 16 features that always appeared in the top 50 were added to the resulting data set in case of removal. This process reduced the previous data set with 204 features into 60 features. This is the data set that will be used throughout this work. As it happens, the data set was already manually feature-engineered by the processes described above and will be considered pre-processing steps for simplicity.

4.2 FEDORA for Feature Engineering

The first objective of this work is addressed by conducting an experimental study concerning the viability of FEDORA. This section compares the proposed framework with other common FE methods, namely PCA, UMAP, SOMs and AEs, regarding performance, Feature Construction (FC) and Feature Selection (FS). Three methods to bias the evolution process are also reported in this section.

4.2.1 Experimental Settings

Concerning the experimental settings, Table 4.1 summarizes the selected parameters of the framework for each one of the 5 experiments performed. Most settings are alike, only diverging in the proxy model, population or generations. All the models used the default Scikit-Learn package parameters, except for the RandomForest where the n_estimators and max_depth parameters were defined to 5. The grammar used is also common in all experiments and is depicted in Figure 2.11. The axiom of the grammar can be expanded to a maximum of 60 $< expr >$ non-terminals, to not bias the evolution since the total number of features in the original dataset is 60. The grammar defines the possible operations between original features through the $< op >$ non-terminal. Therefore, addition, subtraction, multiplication and division are the only accepted operators, effectively creating polynomial-like structures.

Table 4.1: FEDORA Experimental Settings

Name	rf-200-100	xgb-200-100	dt-200-100	dt-1000-50	mlp-100-50
Proxy Model	RandomForest	XGB	DecisionTree		MLP
Population	200			1000	100
Generations	100			50	
Runs	30				
Elitism	10%				
Crossover Rate	0.9				
Mutation Rate	0.1				
Min. Tree Depth	3				
Max. Tree Depth	10				
Selection	Tournament (size 3)				
Fitness	1 - Balanced Accuracy				

For each experiment, 30 runs were performed due to the stochastic nature of the framework. Each run uses a different seed to set the Structured Grammatical Evolution (SGE) algorithm, the dataset split, the model's random state and every other random number generator seed. The original dataset was split into 40% train, 40% validation and 20% test.

For each one of the 5 experiments, 30 individuals were selected, one for each run. Every single individual, i.e. transformation, had the best validation score on their respective run. Each one will be applied to the original dataset and tested with 4 different models. The testing models are the same as the defined proxies in Table 4.1, using the default Scikit-Learn package parameters. The results are compared to their respective baseline, i.e. not applying any transformation to the original dataset.

Other common FE methods, such as PCA, UMAP, SOMs and AEs, are also used to validate the results of the FEDORA framework. These methods were chosen due to belonging to different FE categories. PCA is the most popular linear dimensionality reduction technique and UMAP is a novel non-linear manifold dimension reduction method that has a much better time complexity than any other manifold technique while maintaining state-of-the-art performance. SOMs

and AEs are 2 different types of Artificial Neural Networks (ANNs) that can learn efficient representations by reducing the dimensionality of the data. The AE architecture is described in Figure 4.2. Its parameters are 50 neurons for the single hidden layers, using linear activation functions, and mean squared error as the error metric. It is trained using a batch size of 32, with 50 epochs and using Stochastic Gradient Descend as the optimizer. The remainder of the methods used their default package parameters. A grid search on the hyper-parameters of the UMAP, SOMs and AEs methods was also performed in Appendix A, showing that the selected parameters do not negatively bias the general performance of the algorithm.

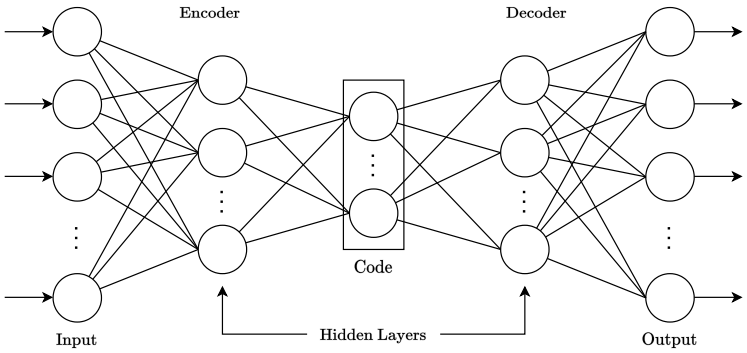


Figure 4.2: Autoencoder architecture

To make a fair comparison, each technique will use the same number of features of the FEDORA individual. As an example, if the FEDORA individual used 15 features, the number of PCA and UMAP components would be set to 15, the 2D SOM grid will have the dimensions of 15x1 and the code size of the AE will be 15. s.

The seed that generated its corresponding individual is once again used for all the testing and comparison tasks mentioned above.

4.2.2 Main Results and Discussion

Regarding results and corresponding discussion, a detailed analysis is performed for each one of the 5 experiments, regarding performance and FE. We will analyse the fitness evolution of the individuals in the Evolutionary Algorithm (EA) through the mean values of the 30 runs, for each experiment. In parallel, their number of features is also presented. Such metrics allow us to overview the evolution process, and to check for any relevant behaviours that must need to be addressed. Concerning the best FEDORA individuals, we will analyse them by counting the number of features that they use and classifying each feature into 1 of 3 categories: original, engineered or complex. This gives us insights from FC and FS standpoints.

The FEDORA transformations will be compared to common FE methods, by measuring the balanced accuracy given by ML classifiers, for every method. This

comparison assesses the practical viability of the proposed framework. To compare the performance results of the different experiments, we performed a statistical analysis, to check for meaningful differences.

Using Random Forests as Proxy

Figure 4.3 presents a set of plots with the results of the rf-200-100 experiment, from various points of view. Panel 4.3a shows how the mean of the populations and the mean of the bests of the 30 runs evolves throughout the generations. As the SGE algorithm considers the minimization of the fitness function, i.e. the validation error given by 1 - balanced accuracy, both lines decrease over the generations. Looking at the plot, one can see that the best line can achieve a lower error score than the population line, but both seem to stabilize at around the 20-generation mark, having small improvements in the remaining generations.

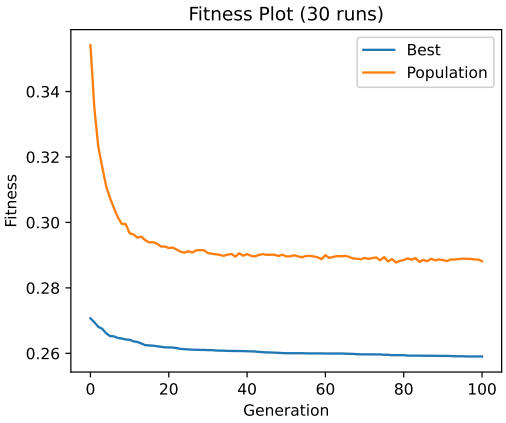
In Panel 4.3b it is possible to observe 4 lines, where each one represents the mean number of constructed or selected features by FEDORA for the overall population (population), the best individual (best) and the individuals with the least (minimum) and with the greatest (maximum) number of features. The plot shows that the minimum and maximum lines tend to be around both limits of the number of allowed features by the grammar. It also shows that the population and best fitness lines tend to grow up together until the 50-feature mark, with the latter being slightly above the former. Note that it is likely that the population line starts from the 30 features mark, due to the random uniform initialization of the SGE population in the given grammar.

Panel 4.3c shows feature ratios from the best individual of each run. To produce this chart, we assume that a feature produced by FEDORA is said to be original if it is solely selected from the original dataset (e.g. feature1), engineered if a single operator is merging 2 original features (e.g. feature1 + feature2), and complex if 2 or more operators are used (e.g. feature1 + feature2 - feature3). When interpreting this Panel, one should also take into consideration Panel 4.3d, which gives the total number of generated features by each best individual. This is relevant since the ratios are normalized by the total number of features of each individual, which implies that the sum of the 3 ratios is always equal to 1. Therefore, the ratios are given by the equations below:

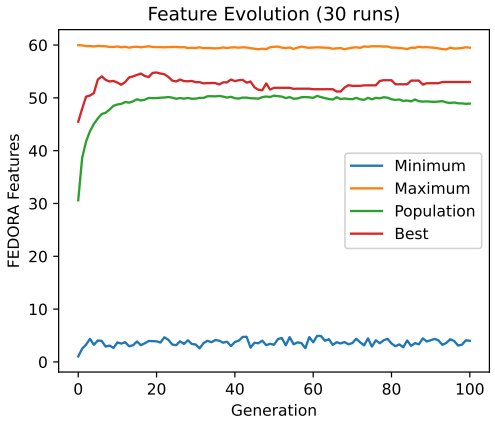
$$R_{original} = \frac{N_{Selected}}{N_{Total}} \qquad R_{engineered} = \frac{N_{Engineered}}{N_{Total}} \qquad R_{complex} = \frac{N_{complex}}{N_{Total}}$$

Hence, Panel 4.3c shows that roughly 70% of all the features present in the individual are original ones, 15% are engineered and 15% are complex. This is an interesting result that shows that FEDORA can select original features, since the ratio of the original features is not null, and construct novel features, due to the sum of the engineered and complex ratios not being null, for every best individual.

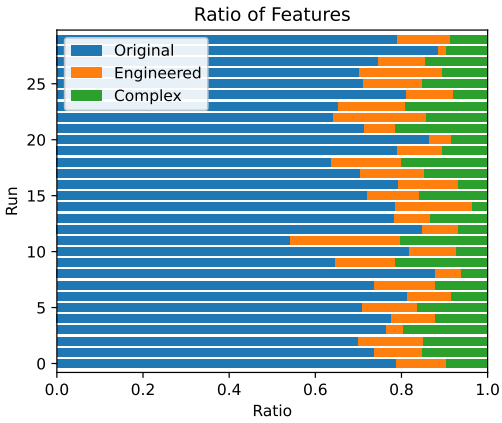
Looking at 4.3d, one can see the total number of features. It is possible to observe



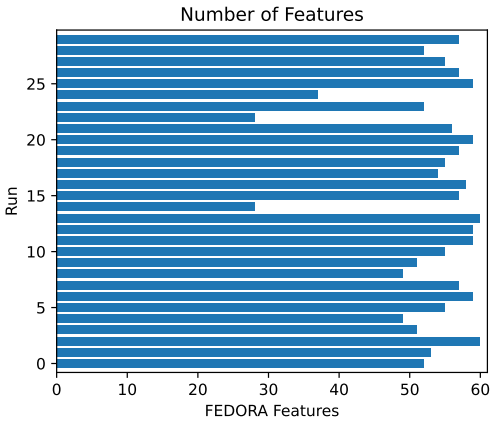
(a) Fitness Plot



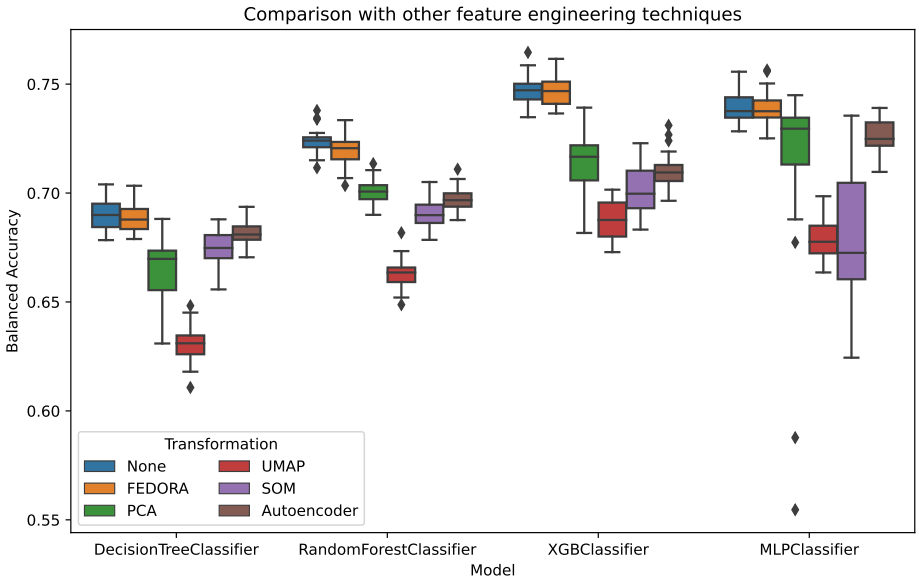
(b) Feature Evolution Plot



(c) Feature Complexity Ratios



(d) Number of features



(e) Feature Engineering Methods Comparison

Figure 4.3: Experiment rf-200-100

that the best individuals rarely used the maximum number of possible features, i.e. 60. Nevertheless, most of them still seem to use a considerable amount of features.

Regarding the comparison with the baseline and other common FE methods, Panel 4.3e shows a collection of 24 boxplots related to the testing results. Each boxplot contains 30 points, one for each seed. The value of each point corresponds to the balanced accuracy of the respective testing model. Results show that the FEDORA can maintain baseline performance, although using fewer features. Concerning the other FE techniques, a clear general baseline performance loss is observable. FEDORA can outperform them in every classifier. The best-performing individual was obtained in run 19, with a 76.2% balanced accuracy score, using the Extreme Gradient Boosting (XGB) classifier with 57 total features (45 Original, 6 Engineered and 6 Complex).

Using Extreme Gradient Boosting as Proxy

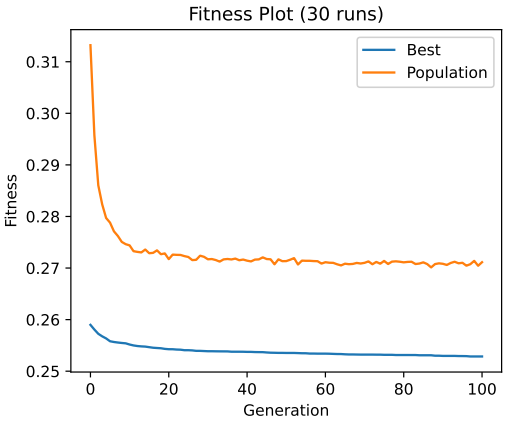
Figure 4.4 summarizes the obtained results of the xgb-200-100 experiment. Similarly to the rf-200-100 experiment, Panel 4.4a shows a clear effective minimization of the error, this time achieving even lower error scores with both lines stabilizing earlier, at the 10-generation mark.

The analysis made for the Panels 4.3b, 4.3c and 4.3d of the rf-200-100 experiment is directly applicable to the Panels 4.4b, 4.4c and 4.4d of this experiment, i.e. FEDORA is effectively able to perform FS and FC since the original ratio and the sum of the remaining ratios are positive, correspondingly. The ratios are also similar to the Random Forest (RF) experiment, containing roughly 70% original, 15% engineered and 15% complex features.

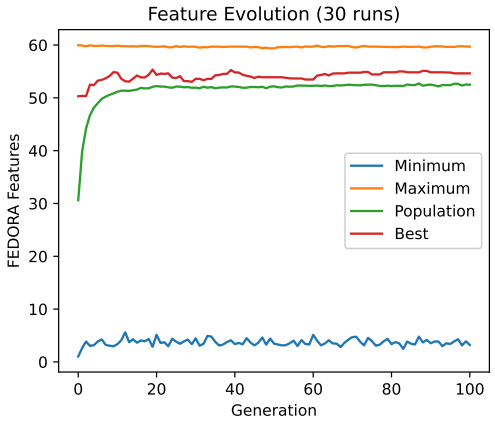
Concerning the comparison (Panel 4.4e), FEDORA is still able to maintain baseline performance across all classifiers. The difference between the FEDORA framework and the other FE methods is still quite noticeable, with FEDORA outperforming the common methods, especially when using the RF and XGB classifiers. It is possible to observe a slight improvement over the baseline with the XGB classifier when using the FEDORA individuals. The best-performing individual was obtained in run 19, with a 76% balanced accuracy score, using the XGB classifier with 58 total features (39 Original, 13 Engineered and 6 Complex).

Using Decision Trees as Proxy

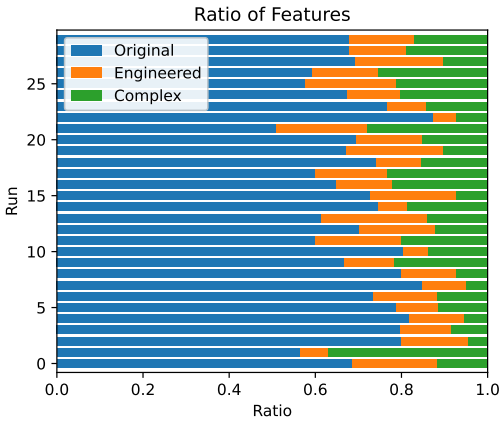
Concerning the dt-200-100 experiment, Figure 4.5 illustrates the obtained results. Unlike the previous experiments, the current one shows different characteristics. Panel 4.5a shows once again that SGE is effectively minimizing the error. Unlike the population line, the best line does not seem to stabilize. It linearly improves until the 29% error mark, while the population's line finishes around the 32% error mark. However, both lines achieve worse validation scores than the previous experiments.



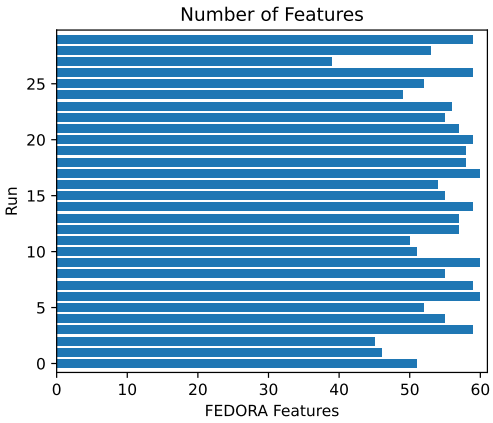
(a) Fitness Plot



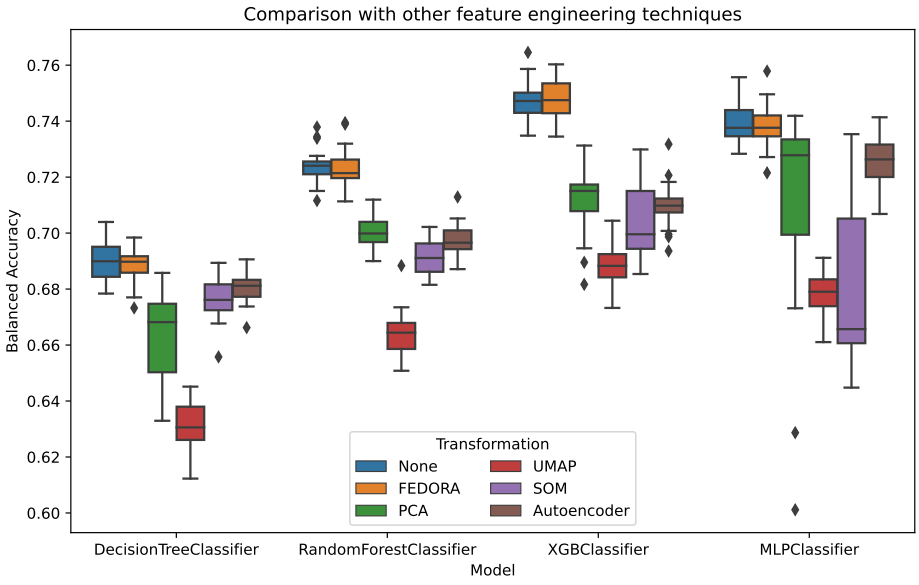
(b) Feature Evolution Plot



(c) Feature Complexity Ratios



(d) Number of features



(e) Feature Engineering Methods Comparison

Figure 4.4: Experiment xgb-200-100

The population and best lines of Panel 4.5b also show differences from the previous experiments, with both lines having a negative trend and not showing any signs of stabilizing. Also, it appears to exist a correlation between the best fitness and population lines. This suggests that the choice of the proxy model has a direct impact on the overall behaviour of the framework.

Panel 4.5c shows once again that FEDORA can construct and select features, for most individuals. This time, it is noticeable greater ratios of engineered and complex figures. This is due mainly to the fact that a lot of individuals have a low number of features, as observed in Panel 4.5d. Runs 6, 8 and 25 were able to generate individuals without any original features, being composed of engineered or complex features only.

Panel 4.5d illustrates the true FE impact that the choice of a Decision Tree (DT) as the proxy model has. The framework was able to greatly reduce the number of features in the individuals, down to 1 single feature, while maintaining baseline performance. In fact, in terms of having the best performance with the least amount of features, run 8 gives the best individual, achieving a 72.8% balanced accuracy score with a RF classifier, with only 1 complex feature.

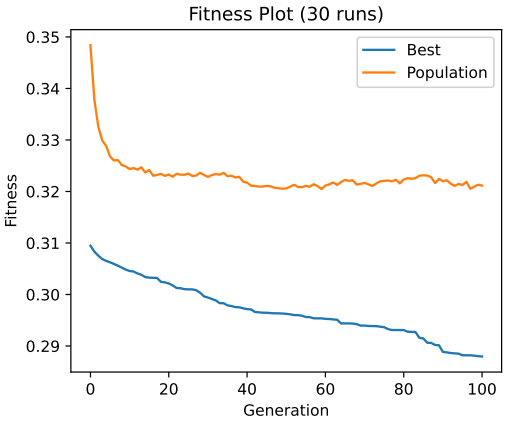
From Panel 4.5e, one can observe differences between the current and the past experiments. When using a DT as the testing model, the FEDORA boxplot improves the baseline performance. In the remaining models, it deteriorates baseline performance. Concerning the other FE techniques, most of them still lay below the FEDORA boxplots. Also, the FEDORA scores seem to be consistent across all the testing models, not showing as much of a difference when comparing to the rf-200-100 and xgb-200-100 experiments, although not quite achieving such high performances.

Resembling the dt-200-100 analysis, the dt-1000-50 experiment further extends its behaviour, since it has more overall fitness evaluations. The plots are illustrated in Figure 4.6. Panel 4.6a shows that both lines keep decreasing throughout the generations, with the best effectively stopping at the 28% error mark, showing signs of stabilizing. The mean of the population also achieves slightly lower error fitness scores.

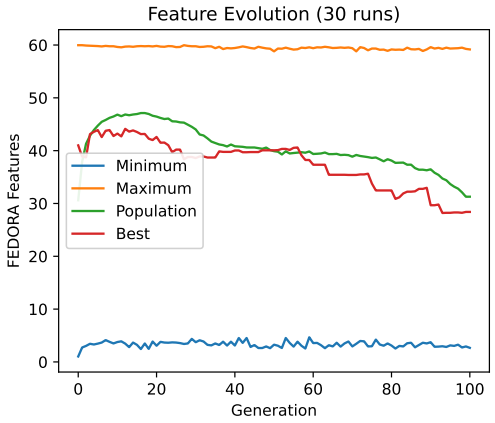
In Panel 4.6b, the best and population lines can further decrease the number of required features. This means that evolution is effectively generating individuals with fewer features while keeping on improving performance. This experiment's results also support the claim that the choice of the proxy model has a direct impact on the framework's behaviour. It is also noticeable an initial increase in the number of features of the population line for the first 10 generations, up to around 45 features. From then onwards, a clear gradual reduction of the number of features is observable.

Panel 4.6c also points to the ability of FEDORA to construct and select features from the original dataset. Engineered and complex ratios also have higher values, for the same reason as Panel 4.5c.

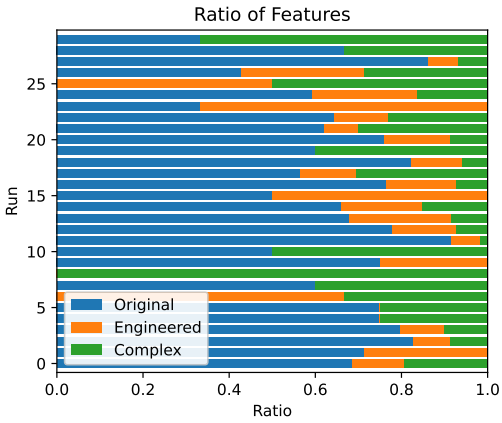
Panel 4.6d shows that most of the best individuals have a small number of features. This is excepted since the dt-200-100 experiment has already suggested that



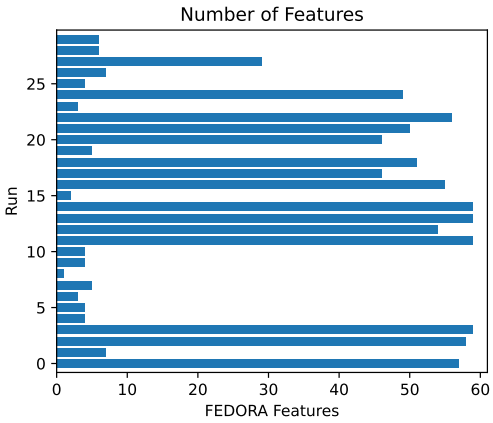
(a) Fitness Plot



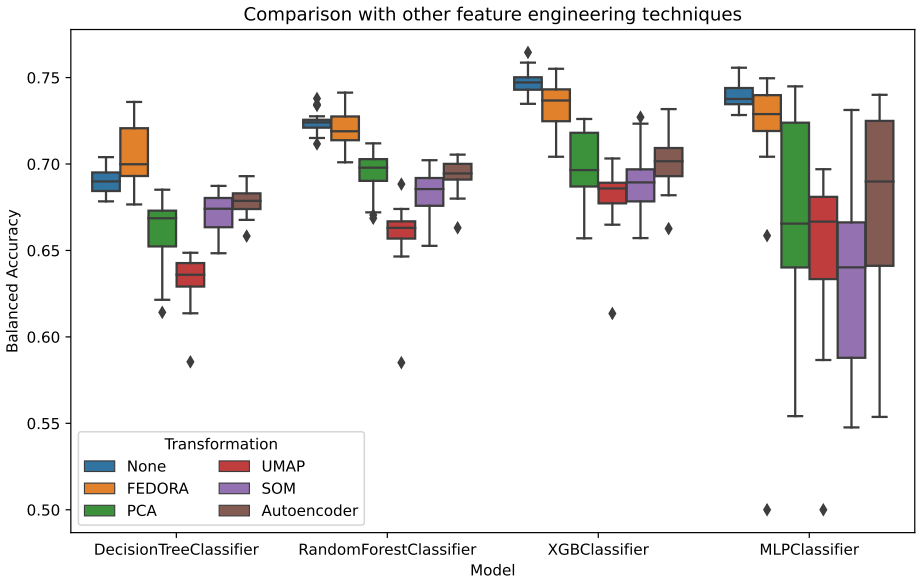
(b) Feature Evolution Plot



(c) Feature Complexity Ratios

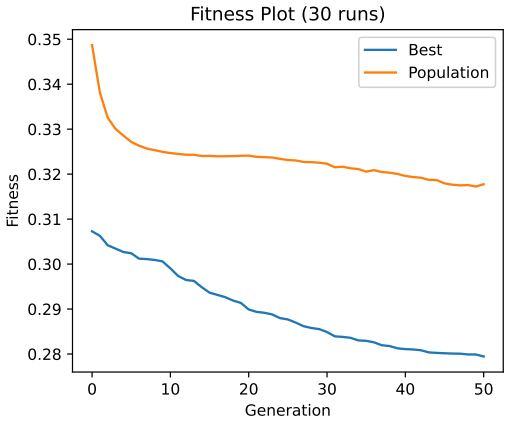


(d) Number of features

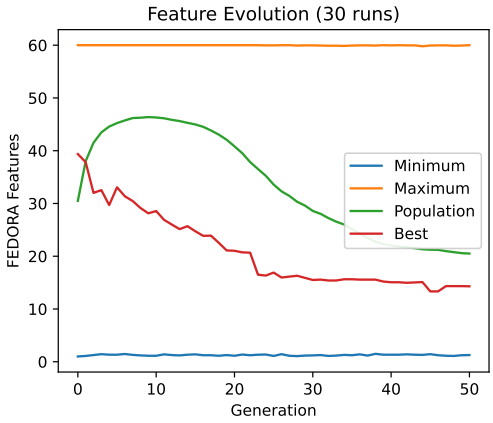


(e) Feature Engineering Methods Comparison

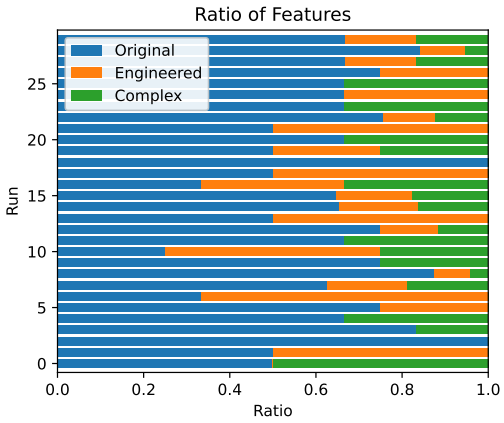
Figure 4.5: Experiment dt-200-100



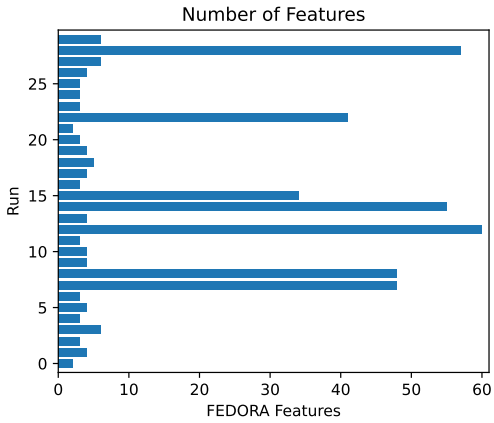
(a) Fitness Plot



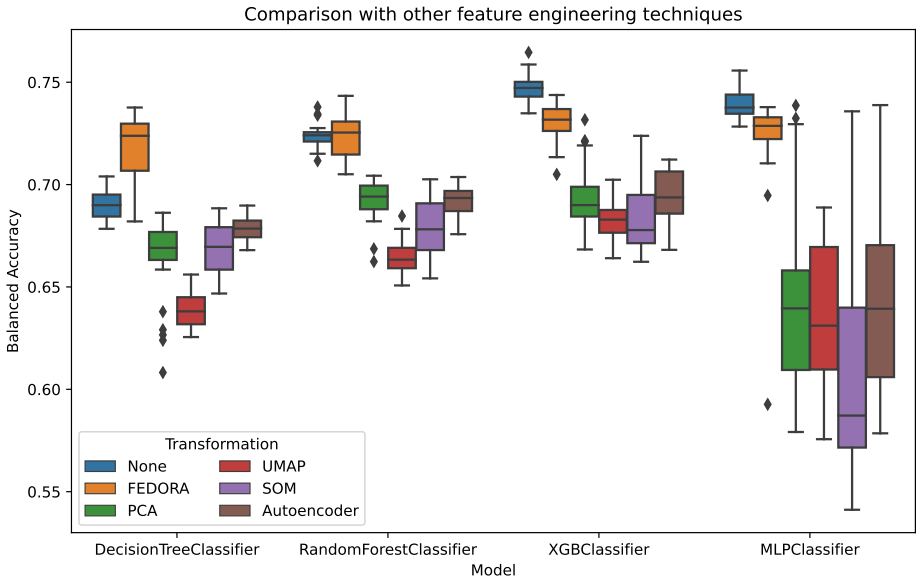
(b) Feature Evolution Plot



(c) Feature Complexity Ratios



(d) Number of features



(e) Feature Engineering Methods Comparison

Figure 4.6: Experiment dt-1000-50

this reduction is a direct consequence of the proxy choice. Therefore, with more evaluations, it is expected a greater reduction in such numbers.

Panel 4.6e shows that when using a DT or a RF as the testing model, FEDORA individuals median can improve the baseline one. In the DT case, a clear improvement is observable. Regarding the XGB and Multi-Layer Perceptron (MLP) classifiers, FEDORA individuals seem to worsen baseline performance. Comparing the results of the remaining FE methods, there is a clear difference between them and the framework scores, with FEDORA outperforming them. This effect is perhaps more noticeable in this experiment, due to the low number of features that each individual has, generally. This forces the common FE methods to also restrict themselves to that number of features, which might not be optimal for them. Nevertheless, the FE methods match the number of features that the FEDORA individuals use. Once again, the FEDORA scores are consistent across all classifiers, not showing remarkable differences.

Using Multi-Layer Perceptrons as Proxy

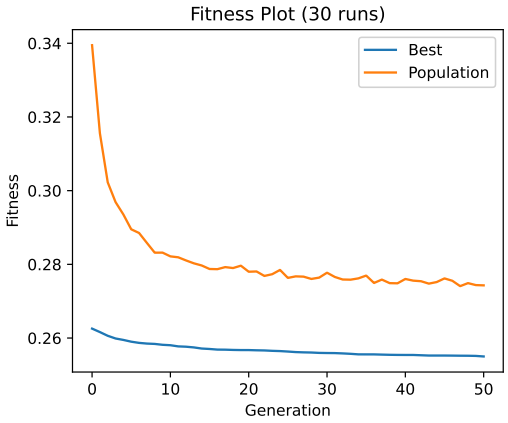
Lastly, Figure 4.7 describes the obtained results for the mlp-100-50 experiment. Panel 4.7a shows that the error decreases for both lines throughout the generations. The best line has a very small improvement, stopping at around the 26% error threshold. The population line has a bigger improvement, ending at an error of around 28%.

Panel 4.7b shows an already-seen behaviour in the rf-200-100 and xgb-200-100 experiments. The best and population lines seem to grow together until slightly below the 50 features mark. Also, the minimum and maximum lines seem to rather be growing towards one another, in a very faint manner. The trend of such lines is certainly different than in the other experiments. Therefore, it seems to exist a pattern relating the choice of the proxy model and the behaviour of FEDORA. Panel 4.7c shows once more that FEDORA creates and selects original features. The proportions of the ratios are similar to the ones observed in the rf-200-100 and xgb-200-100 experiments. In Panel 4.7d, one can observe that the number of features used by each best individual is somewhat high, although reducing the original number of features of the dataset.

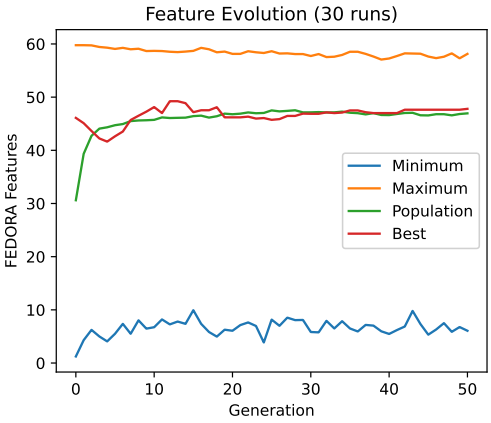
Regarding performance, Panel 4.7e shows that FEDORA can maintain performance, although having slightly worse results than the baseline for all classifiers. Concerning the remaining FE methods, the proposed framework still can visually outperform all of them. The best-performing individual was obtained in run 19, with a 76.1% balanced accuracy score, using the MLP classifier with 43 total features (33 Original, 5 Engineered and 5 Complex).

Statistical Analysis

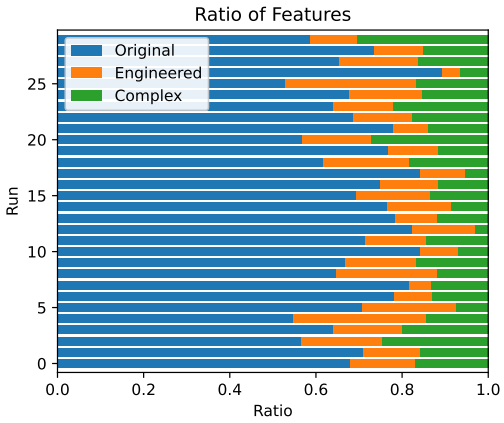
To compare the results of the different experiments, we performed a statistical analysis to check for any meaningful differences. The statistical tests were only applied to the FE methods of one single testing classifier for each experiment,



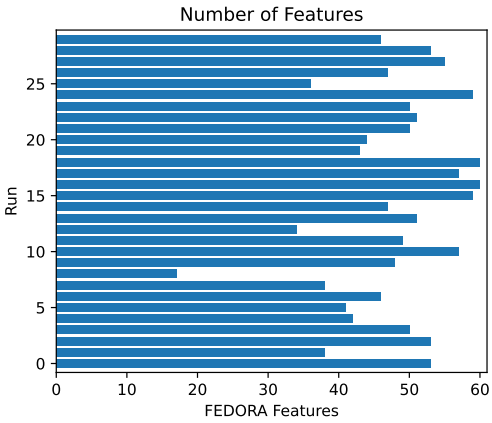
(a) Fitness Plot



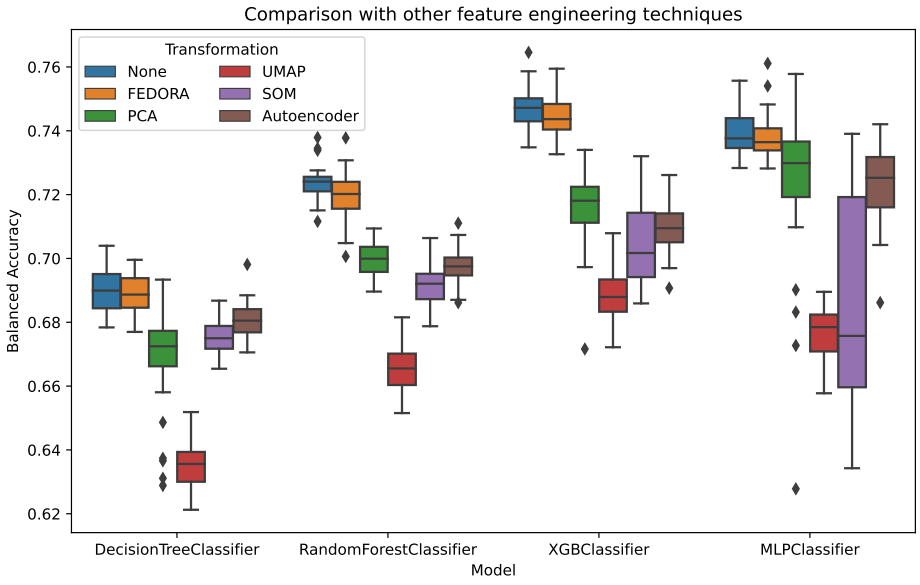
(b) Feature Evolution Plot



(c) Feature Complexity Ratios



(d) Number of features



(e) Feature Engineering Methods Comparison

Figure 4.7: Experiment mlp-100-50

for simplicity. The chosen testing model for the statistical test is the same as the proxy of the corresponding experiment.

Without making any parametric or paired assumptions, the Kruskal-Wallis non-parametric test was applied to compare the FE techniques, in each experiment, to check if the medians of all the groups are equal. A significance level of 0.05 was selected. Having the null hypothesis rejected, pair-wise comparisons were made using Dunn's posthoc test and correcting the resulting p-values with the Bonferroni correction. Cliff's δ was used to measure the effect size. The symbol "$\sim$" denotes a negligible effect size ($|\delta| < 0.147$), "+" denotes a small effect size ($0.147 \leq |\delta| < 0.33$), "++" a medium one ($0.33 \leq |\delta| < 0.474$) and "+++" a large one ($|\delta| \geq 0.474$).

Table 4.2: Kruskal-Wallis Test Results

Experiment	Model	Test Statistic	P-Value
rf-200-100	RandomForestClassifier	156.48	0
xgb-200-100	XGBClassifier	145.27	0
dt-200-100	DecisionTreeClassifier	143.45	0
dt-1000-50	DecisionTreeClassifier	143.65	0
mlp-100-50	MLPClassifier	116.58	0

Table 4.2 gives the Kruskal-Wallis test results for every experiment. As the p-value is 0 for all experiments, every experiment rejects the null hypothesis, i.e. there are differences in the medians of the groups. Therefore, a pairwise post hoc analysis is required for every pair of groups in each experiment.

Table 4.3 details the effect sizes for Dunn's posthoc analysis for the rf-200-100 experiment. It shows that there are statistically significant differences between FEDORA and the other FE methods, with a large effect size. There are also differences between the baseline and the common FE methods, with a large effect size. There is no evidence of differences between the baseline and the FEDORA groups, meaning that the framework can statistically maintain performance, for this experiment. Furthermore, there are statistically significant differences between the PCA and UMAP groups and between the AE and UMAP groups, with large effect sizes.

Table 4.4 gives the effect sizes for the xgb-200-100 experiment. The statistical analysis is the same as the one made for Table 4.3 since the tables are identical.

Table 4.5 provides the effect sizes for the dt-200-100 experiment. Once again, the baseline and the proposed framework have statistically significant differences with the common FE methods. Also, the baseline and FEDORA groups do not seem to have differences. There are statistically significant differences between the UMAP and the ANN based FE methods, i.e. the SOMs and the AEs, both with large effect sizes.

Table 4.6 reports the effect sizes for the dt-1000-50 experiment. The analysis is the same made for Table 4.5, except the comparison between the UMAP and PCA groups, which show statistically significant differences, with large effect size.

Table 4.3: Dunn's test effect sizes - rf-200-100

RandomForest	baseline	FEDORA	PCA	UMAP	SOM	Autoencoder
baseline						
FEDORA						
PCA	+++	+++				
UMAP	+++	+++	+++			
SOM	+++	+++				
Autoencoder	+++	+++		+++		

Table 4.4: Dunn's test effect sizes - xgb-200-100

XGB	baseline	FEDORA	PCA	UMAP	SOM	Autoencoder
baseline						
FEDORA						
PCA	+++	+++				
UMAP	+++	+++	+++			
SOM	+++	+++				
Autoencoder	+++	+++		+++		

Table 4.5: Dunn's test effect sizes - dt-200-100

DecisionTree	baseline	FEDORA	PCA	UMAP	SOM	Autoencoder
baseline						
FEDORA						
PCA	+++	+++				
UMAP	+++	+++				
SOM	+++	+++		+++		
Autoencoder	+++	+++		+++		

Table 4.6: Dunn's test effect sizes - dt-1000-50

DecisionTree	baseline	FEDORA	PCA	UMAP	SOM	Autoencoder
baseline						
FEDORA						
PCA	+++	+++				
UMAP	+++	+++	+++			
SOM	+++	+++		+++		
Autoencoder	+++	+++		+++		

Table 4.7: Dunn's test effect sizes - mlp-100-50

MLP	baseline	FEDORA	PCA	UMAP	SOM	Autoencoder
baseline						
FEDORA						
PCA	+++					
UMAP	+++	+++	+++			
SOM	+++	+++	+++			
Autoencoder	+++	+++		+++	+++	

Lastly, Table 4.7 shows the effect sizes for the mlp-100-50 experiment. The baseline still maintains statistically significant differences with the common FE methods. However, besides the proposed framework not having statistically significant differences with the baseline, it also does not have differences with the PCA technique, for this experiment. Concerning the common FE groups, there are significant differences for both the PCA and the AE groups with the UMAP and SOM groups, with large effect sizes.

Discussion

Concerning the evolution plots, all fitness plots show that individuals are gradually evolving throughout the generations. When using a DT model as the proxy, the lines appear to be the ones with greater evolution progress, although not quite reaching the other experiments' low error performances, on either the best or the population lines.

The feature evolution plots show a different angle of such evolution. The number of features in the DT experiments' best individuals is decreasing throughout the generations, alongside the population mean. Such an event is not noticeable in the other experiments. The exact opposite happens, i.e. the best and population lines tend to grow and stabilize, with the latter resembling a logarithmic function. By observing the number of features in the DT experiments, it is noticeable that its individuals' transformations can achieve a much lower feature dimensionality. These experiments also show a higher ratio of engineered and complex features, although having fewer features biasing them. For RF, XGB and MLP experiments, it is possible to observe that FEDORA can simultaneously select and construct novel features since the ratio of original features and the sum of engineered and complex features' ratios are positive.

Regarding the comparison with other common FE methods and the baseline, the comparison plots show that FEDORA is consistently above the PCA, UMAP, SOMs and AEs methods while statistically maintaining baseline performance. In the DT experiments, FEDORA is also able to improve past the baseline values when using a DT as the testing model, although such results not being statistically significant.

From the analysed experiments, a pattern emerges in the behaviour of FEDORA. The DT experiments can reduce the number of features to a degree that the other proxy models cannot. When comparing the inner workings of the models, the RF, XGB and MLP models all have one thing in common that the DT model does not: the ability to create a more complex internal representation of the given data or decision boundary, which generally translates into better performances. A DT can only make simple decisions with the provided data, which translates into axis-parallel hyper-planes decisions in the feature space, which might not properly address a complex dataset. As such, if the evolution transformations do not provide adequate features to this model, i.e. constructed features that allow for oblique hyper-plane decisions in the original feature space, the DT will most likely have worse performance than the remaining models, when facing a hard problem. Consequently, this encourages evolution to provide well-engineered

features, thus making the fitness function much more discriminant. On the other hand, the remaining models do not put this kind of pressure on the evolution process. Each model takes charge of either constructing its features internally or defining a more complex decision boundary. Therefore, evolution just gives it a solid amount of original features, so that the model can find what works best for itself, and a few suggestions in the form of engineered and complex features. Consequently, the best individuals tend to have a much higher number of features in the RF, XGB and MLP experiments. When using these models as the proxy, aiming for individuals with a low number of features becomes a problem. As such, ways to bias the evolution are required, namely reducing the number of features that a transformation can produce in the grammar, e.g. 1 to 10 instead of 1 to 60, or adding a fitness component that penalizes individuals with many features. The usage of different feature combining operators may also be of use. These modifications might prove themselves useful in such a task.

When comparing FEDORA with PCA by pinning variance thresholds and selecting the individuals with the closest number of features, FEDORA still overall outperforms PCA.

Given these results and considering that FEDORA and the other methods are usually working with fewer features, with their main purpose being a FE technique, effectively reducing the number of features and statistically maintaining the baseline performance are great results. From the methods used in this work, FEDORA is the only one that can almost always have this behaviour. Also, it is possible to understand a FEDORA individual's transformation logic to a certain degree, depending on the choice of the operators defined in the grammar.

4.2.3 Evolution Biasing

Other experiments were conducted to validate how some evolution biasing methods impact the performance and FE aspects of the proposed framework. These three ideas consist of using a problem-specific boolean grammar, reducing the number of possible features in the grammar or modifying the fitness function.

Boolean Grammar

The boolean grammar is depicted in Figure 4.8. This strategy consists in setting apart float and boolean attributes from one another, where they will have specific operators. The float attributes will maintain the addition, subtraction, multiplication and division, while the boolean ones will have the logical AND, the logical OR and the logical negation. These operators were selected due to the nature of the types in the dataset. Any user-custom operator might be defined in the grammar, provided that it makes sense in its corresponding problem-specific context.

Notice that the expansions of the $< expr >$ non-terminal ($< exprr >$ or $< exprb >$) are repeated a specific number of times. This is made to try no minimize the bias over the feature type since there are 44 real features and 16 boolean

$N = \{< \text{start} >, < \text{expr} >, < \text{exprr} >, < \text{exprb} >, < \text{opr} >, < \text{opb} >, < \text{real} >, < \text{bool} >\}$

$T = \{+, -, \cdot, /, (,), \neg, \text{and}, \text{or}, X_1^B, \ldots, X_{N_{\text{bool}}}^B, X_1^R, \ldots, X_{N_{real}}^R\}$

$S = < \text{start} >$

P :

$< \text{start} > ::= < \text{expr} >$,
$\mid < \text{expr} >, < \text{expr} >$
$\mid < \text{expr} >, < \text{expr} >, < \text{expr} >$
$\mid \ldots$
$\mid < \text{expr} >, \ldots, < \text{expr} >$

$< \text{expr} > ::= < \text{exprb} > \mid < \text{exprb} > \mid < \text{exprb} > \mid < \text{exprb} >$
$\mid < \text{exprr} > \mid < \text{exprr} > \mid < \text{exprr} > \mid < \text{exprr} >$
$\mid < \text{exprr} > \mid < \text{exprr} > \mid < \text{exprr} > \mid < \text{exprr} >$
$\mid < \text{exprr} > \mid < \text{exprr} > \mid < \text{exprr} >$

$< \text{exprr} > ::= < \text{exprr} >< \text{opr} >< \text{exprr} >$
$\mid (< \text{exprr} >< \text{opr} >< \text{exprr} >)$
$\mid < \text{real} >$

$< \text{exprb} > ::= < \text{exprb} >< \text{opb} >< \text{exprb} >$
$\mid (< \text{exprb} >< \text{opb} >< \text{exprb} >)$
$\mid < \text{bool} >$
$\mid \neg < \text{bool} >$

$< \text{opr} > ::= + \mid - \mid \cdot \mid /$

$< \text{opb} > ::= \text{and} \mid \text{or}$

$< \text{real} > ::= X_1^R \mid X_2^R \mid X_3^R \mid \ldots \mid X_{N_{real}}^R$

$< \text{bool} > ::= X_1^B \mid X_2^B \mid X_3^B \mid \ldots \mid X_{N_{\text{bool}}}^B$

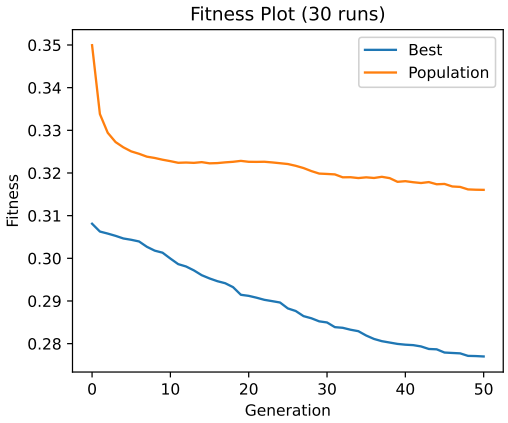
Figure 4.8: Boolean Grammar

ones. Therefore, there is a ratio of 4 booleans for every 11 real features, which corresponds to the number of repetitions.

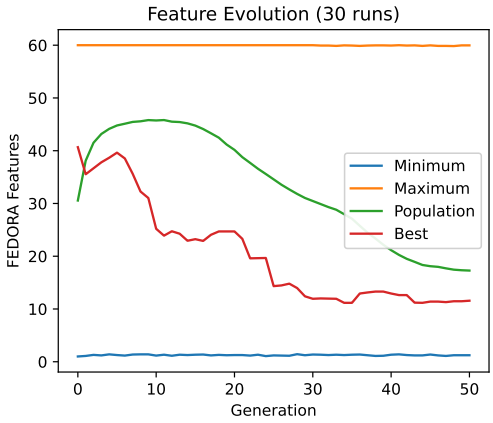
Contrary to the constructed float attributes, the boolean features can be read in an intelligible manner. Therefore, by choosing a DT as the proxy model with individuals generated by this grammar, the interpretability of the underlying decisions might increase. Figure 4.9 shows the results of the dt-1000-50 experiment, with the boolean grammar, which we will name dt-1000-50-bool for simplicity.

Panels 4.9a, 4.9b and 4.9d plot identical results to the dt-1000-50 experiment, with lines of the fitness and of the number of features decreasing similarly.

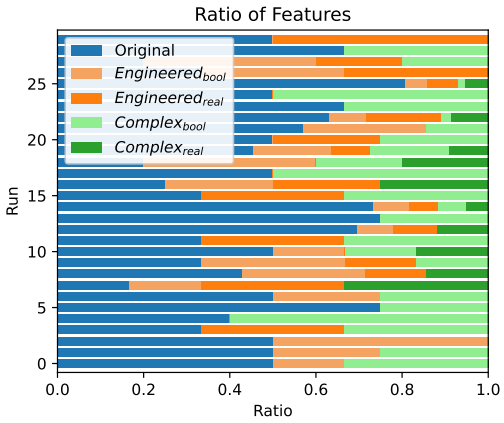
Panel 4.9c shows the feature ratios but splits the engineered and complex ratios by their type, i.e. boolean (brighter) or real (darker). The number of engineered and complex features increases relatively to the dt-1000-50 experiment, which may be explained due to the introduction of boolean expressions. The engineered and complex boolean features have greater ratios than the real ones in the best individuals, showing that the constructed boolean expressions are more relevant



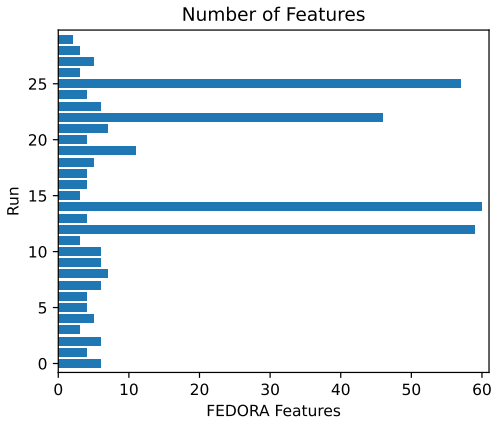
(a) Fitness Plot



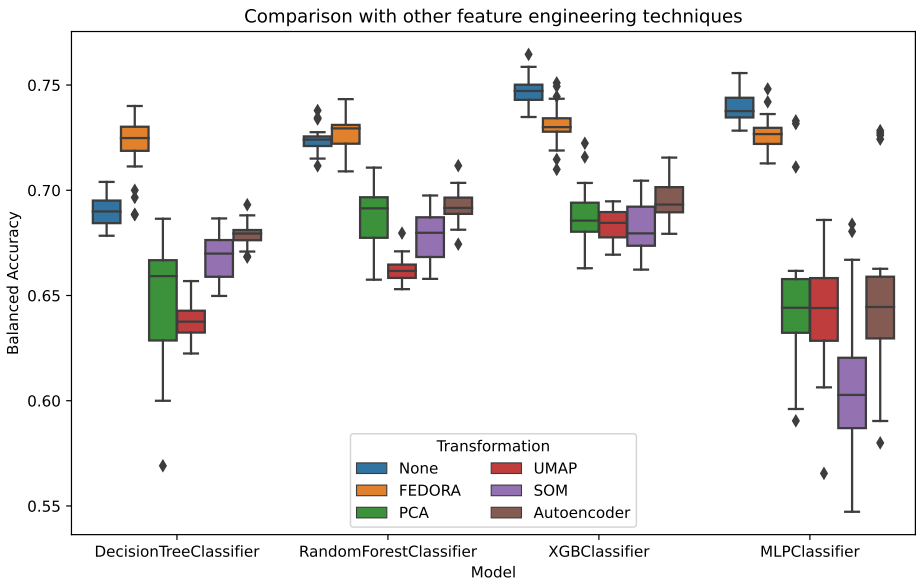
(b) Feature Evolution Plot



(c) Feature Complexity Ratios



(d) Number of features



(e) Feature Engineering Methods Comparison

Figure 4.9: Experiment dt-1000-50-bool

to achieve better performances than the real expressions.

Lastly, Panel 4.9e also shows different results than the dt-1000-50 experiment. Across all testing models, the standard deviation value of the FEDORA groups has decreased, showing more consistency in the obtained results. Also, in the DT testing model, the minimum value of the FEDORA group is above the best DT baseline result, when discarding outliers. Regarding other common feature engineering techniques, all of them are underperforming FEDORA. These results point to the relevance of problem-specific operators when aiming for good and consistent performance scores.

An example of a phenotype generated by this grammar is presented below. It comprises 1 original real feature (age), 2 original boolean features (question_2_1 and question_3) and a complex boolean feature, separated by semicolons. This transformation achieved 72.8% balanced accuracy with XGB as the testing model.

$$X^R_{age} \ ; \ X^B_{question\_2\_1} \ ; \ X^B_{question\_3} \ ; \ \neg X^B_{question\_1\_1} \ \& \ \neg X^B_{question\_3\_3}$$

Concerning statistical results, the same analysis of the previous experiments is applied here. The Kruskal-Wallis non-parametric test was applied to the DT testing model groups, which lead to the null hypothesis being rejected for a significance level of 0.05. Therefore, Table 4.8 summarizes the Cliff's δ effect sizes of the post hoc pairwise comparisons between the groups, using Dunn's test.

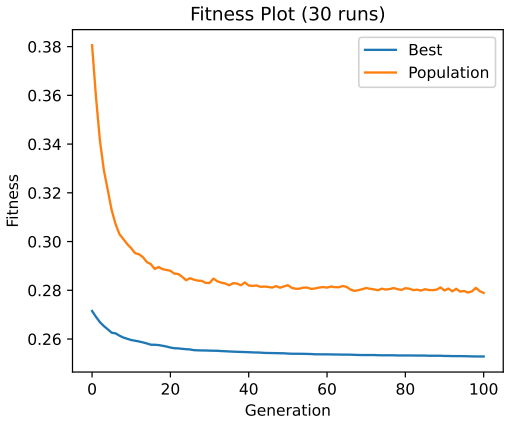
The baseline has statistically significant differences with the PCA, UMAP and SOMs methods, with large effect sizes, while having no differences with the FEDORA and AE groups, showing that FEDORA can maintain performance. The FEDORA framework has meaningful differences with all common FE techniques, with a large effect size. Lastly, the UMAP has large effect size differences with the ANN based methods, and the PCA group has differences with the AE group.

Table 4.8: Dunn's test effect sizes - dt-1000-50-bool

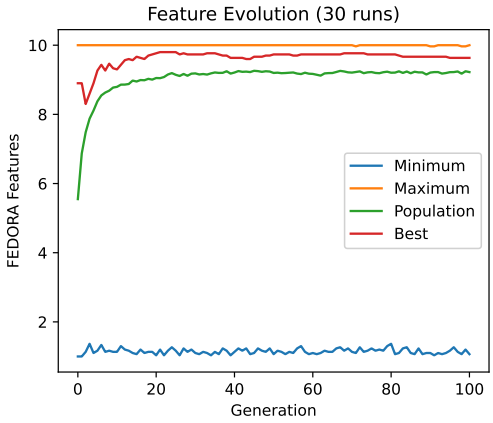
DecisionTree	baseline	FEDORA	PCA	UMAP	SOM	Autoencoder
baseline						
FEDORA						
PCA	+++	+++				
UMAP	+++	+++				
SOM	+++	+++		+++		
Autoencoder		+++	+++	+++		

Available Features

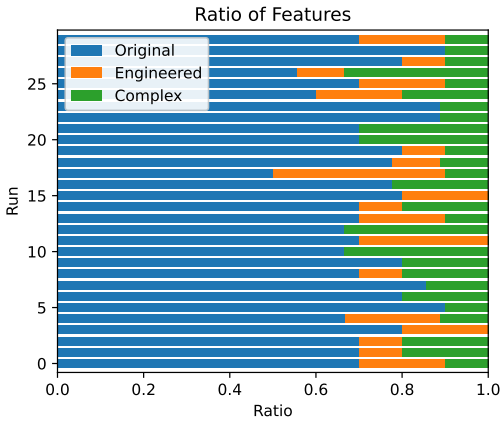
This idea consists of forcefully reducing the number of features that an individual can have in the grammar. By default, any individual can select between 1 to 60 features to not bias the evolution. Therefore, changing the maximum number of features to a lower value, i.e. 60 features into 10, is possible. From the 5 main experiments reported in Subsection 4.2.2, all but the DT experiments had similar



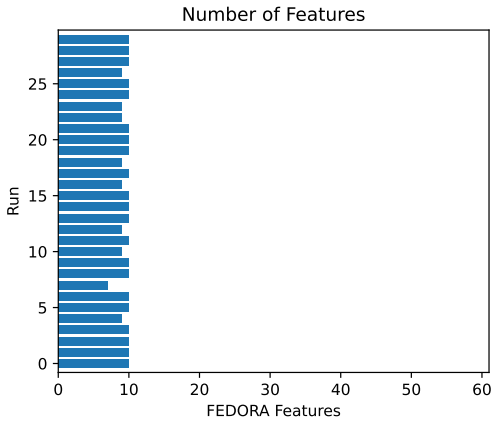
(a) Fitness Plot



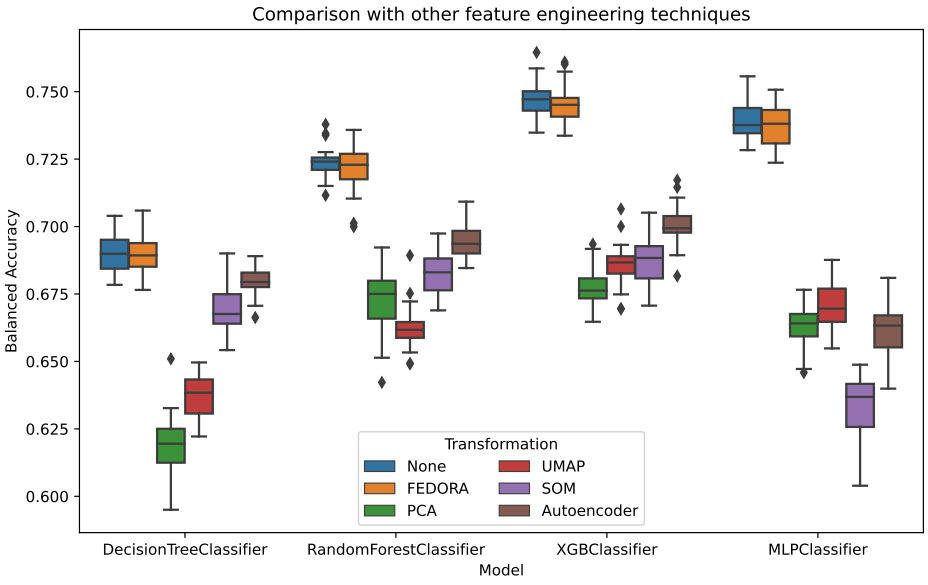
(b) Feature Evolution Plot



(c) Feature Complexity Ratios



(d) Number of features



(e) Feature Engineering Methods Comparison

Figure 4.10: Experiment xgb-200-100-low10

behaviours in terms of reducing the number of features. Therefore the XGB experiment was arbitrarily selected to apply this evolution biasing idea, which we will name xgb-200-100-low10.

Figure 4.10 presents the results for the xgb-200-100-low10 experiment. Panel 4.10a shows a similar evolution to the xgb-200-100 experiment, this time with both lines stabilizing around the 20th generation.

Panel 4.10b shows that this experiment maintains the same behaviour as the xgb-200-100 experiment, with the lines increasing and stabilizing over the generations, but this time the maximum allowed number of features is 10 instead of 60.

Panel 4.10c also presents a recurring distribution of the feature types, with around 70% original, 15% engineered and 15% complex. The last 2 analysed charts strongly point out that biasing the number of features allowed in the grammar has no impact on the behaviour of the evolution when using XGB as the proxy model, even though the number of features being forcefully reduced to a maximum of 10, as one can see in Panel 4.10d.

Panel 4.10e presents the testing results for all the FE methods. It is possible to observe that the FEDORA framework has negligible differences with the baseline across all testing models. The FEDORA results are slightly worse than in the xgb-200-100 experiment but use much fewer features. Concerning the remaining FE methods, FEDORA can achieve better performance results.

Regarding the statistical results, the Kruskal-Wallis test was rejected in the XGB testing models groups, for a significance level of 0.05. Table 4.9 reports the Cliff's δ effect sizes for Dunn's post hoc test.

The baseline and FEDORA groups have statistically significant differences, with a large effect size, with all common FE methods, but have no meaningful differences between one another. Also, the AE group has large effect sizes differences with the PCA and UMAP methods.

Table 4.9: Dunn's test effect sizes - xgb-200-100-low10

XGB	baseline	FEDORA	PCA	UMAP	SOM	Autoencoder
baseline						
FEDORA						
PCA	+++	+++				
UMAP	+++	+++				
SOM	+++	+++				
Autoencoder	+++	+++	+++	+++		

These results are very interesting since they show that FEDORA can maintain baseline performance, outperforming common FE techniques when using 10 features as the maximum allowed dimensionality. There is also evidence that forcefully reducing the number of features does not impact the framework's general strategy to evolve individuals when using an XGB model, since the number of features tends to grow throughout the generations and ends up achieving similar

ratios to the xgb-200-100 experiment, with a negligible loss of performance, i.e. less than 0.2% mean balanced accuracy across all testing models except for the DT, but with fewer total features.

Fitness Function Penalization

The last approach that we evaluated concerns the penalization of individuals that provide more features to the classifiers than the ones that the models are effectively using. For instance, an individual could provide 40 features to the classifier and it would only end up using 30. The penalty is given by the ratio of unused features by the model. In this example, the penalty would be calculated by counting the number of unused features ($40 - 30 = 10$) and dividing it by the total number of features ($10/40$), which would lead to a penalty of 0.25. The formula is described below, where I is the individual and M is the fitted model.

$$Penalty(I, M) = \frac{I_{features} - M_{used}}{I_{features}}$$

Therefore, the equation below gives the fitness function, which SGE looks to minimize. The higher the penalty, i.e. the number of unused features, the worse the fitness. Notice that both components of the fitness function are bounded to the same interval, i.e. $[0, 1]$, hence both have the same weight in the fitness of an individual.

$$Fitness(I) = Score(I) + Penalty(I, M)$$

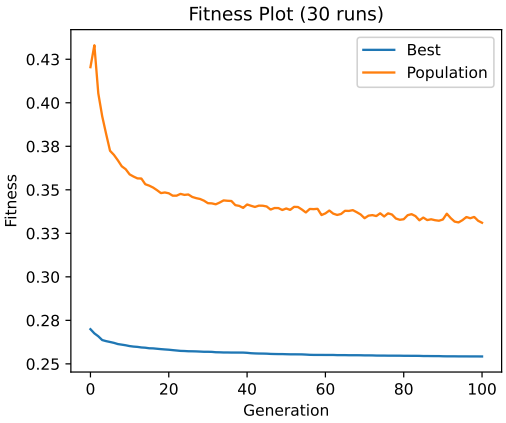
This evolution biasing method makes the fitness function consider the score and the number of unused features as equally relevant. This technique will be applied to the xgb-200-100 experiment, to check if it can change the underlying strategy to evolve individuals. Figure 4.11 shows the results of this experiment, which we will name xgb-200-100-penalty.

Panel 4.11a shows that both fitness lines decrease throughout the generations. These lines cannot be perceived as performance, but rather as a mean of the performance and the unused feature ratios.

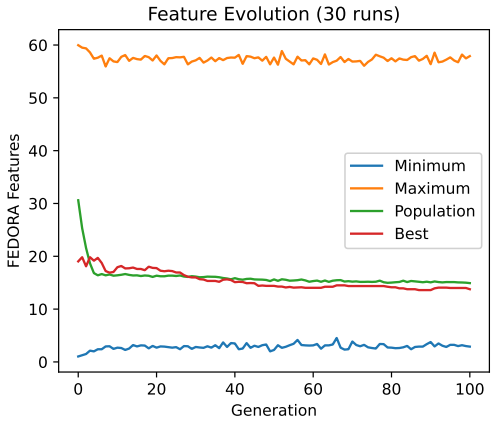
Panel 4.11b reports a completely different behaviour of the framework when using the XGB model as the proxy. The population and best lines tend to simultaneously decrease until reaching a plateau where both stabilize, around 18 total features.

Regarding the ratios of the features, Panel 4.11c shows the FEDORA is always able to select and construct features. The ratio of engineered and complex had a small increase when comparing it to the xgb-200-100 and xgb-200-100-low10 experiments.

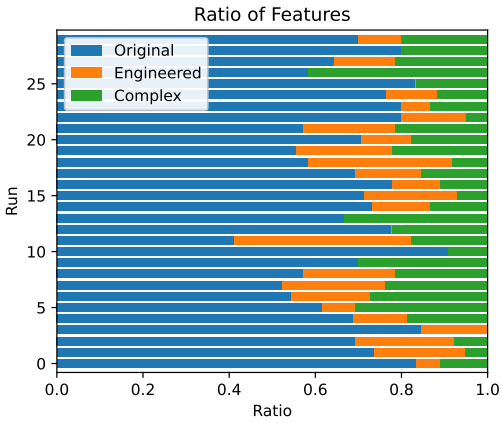
Panel 4.11d shows that the framework was able to reduce the number of features of the best individuals to around 15 total features, with some variance.



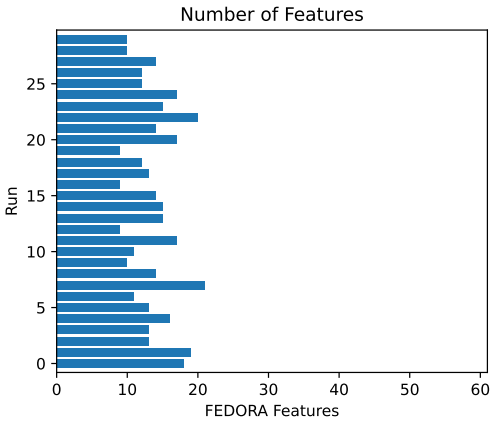
(a) Fitness Plot



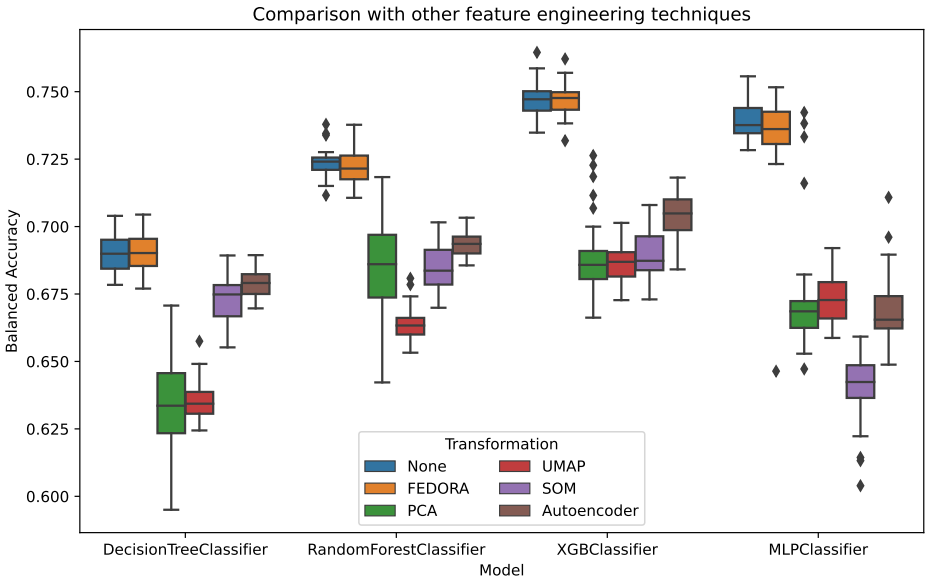
(b) Feature Evolution Plot



(c) Feature Complexity Ratios



(d) Number of features



(e) Feature Engineering Methods Comparison

Figure 4.11: Experiment xgb-200-100-penalty

Concerning results, Panel 4.11e shows that FEDORA can match baseline performance across all testing models while performing well above common FE methods. The best-performing individual was obtained in run 19, with a 76.2% balanced accuracy score, using the XGB classifier with 9 total features (5 Original, 2 Engineered and 2 Complex). This corresponds to the best-obtained result of this work. Figure 4.12 illustrates the phenotype of this individual using a simplified derivation tree.

Regarding the statistical comparison, the Kruskal-Wallis non-parametric test was applied to the groups of the XGB testing model, resulting in the rejection of the null hypothesis, for a significance level of 0.05. Table 4.10 reports the Cliff's δ effect sizes for Dunn's post hoc test.

Table 4.10: Dunn's test effect sizes - xgb-200-100-penalty

XGB	baseline	FEDORA	PCA	UMAP	SOM	Autoencoder
baseline						
FEDORA						
PCA	+++	+++				
UMAP	+++	+++				
SOM	+++	+++				
Autoencoder	+++	+++	+++	+++	+++	

Once again, the baseline and FEDORA groups have statistically significant differences, with a large effect size, with the common FE techniques, but have no meaningful differences between each other. The AE group has large effect sizes differences with the PCA and UMAP methods.

The results of this experiment show that by adding a penalization component to the fitness function one can guide the evolution strategy of the framework, to not only achieve individuals with fewer features but also to increase performance. Due to the being a component that directly affects the evolution fitness, this biasing method is highly likely to work with any other proxy model that tends not to decrease the number of features throughout the evolution, such as RFs or MLP models.

4.3 AutoML with FEDORA

This section addresses the second objective of this work, i.e. completing the full ML pipeline, by performing a comparison study with FEDORA and a method capable of addressing the CASH problem, the AutoML-DSGE framework [Assunção et al., 2020]. This framework has been applied to 10 different datasets with good results. The best FEDORA individuals are added to the list of pre-processors of the AutoML-DSGE framework to check if such transformations can match other common pre-processing methods.

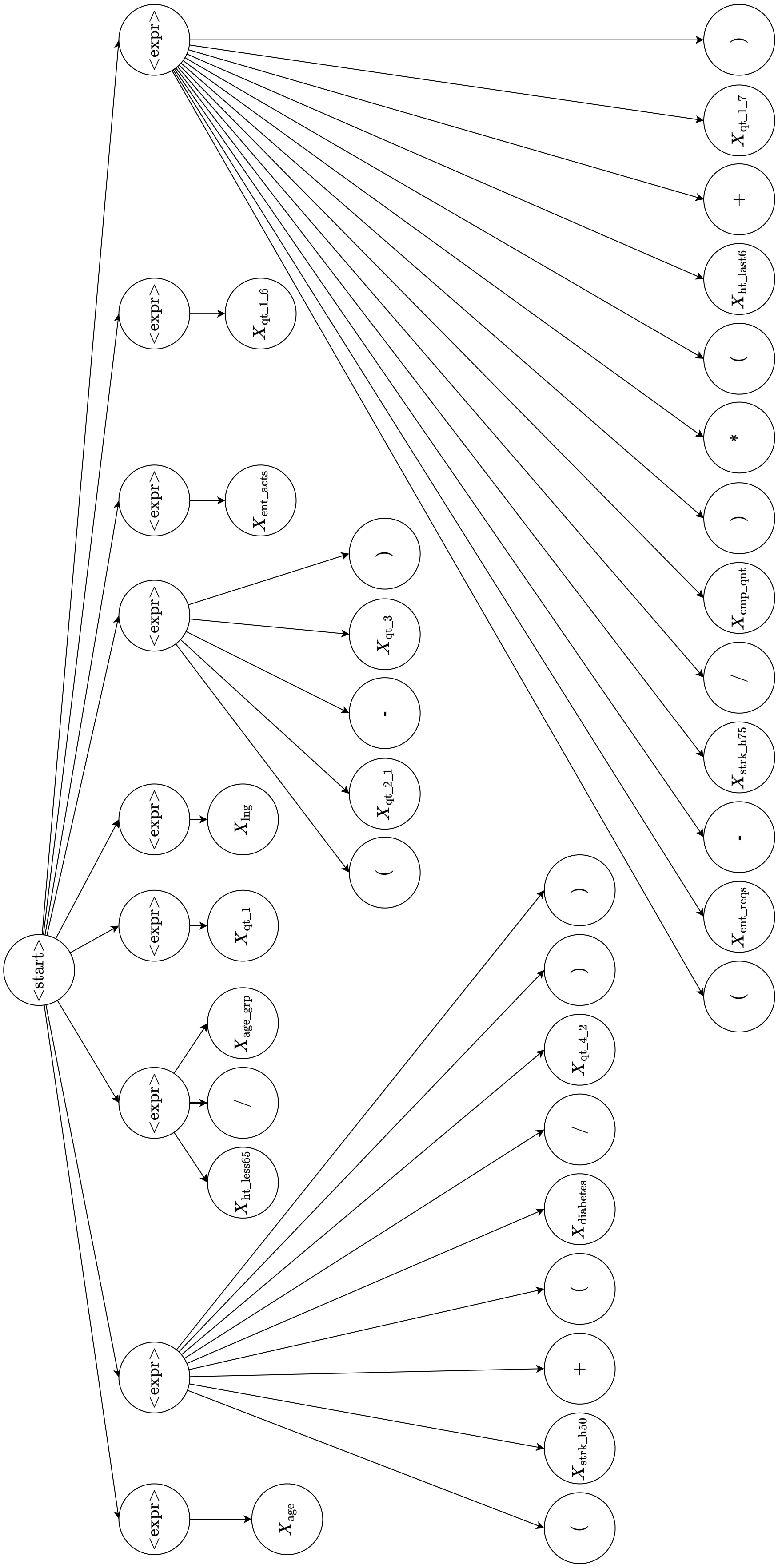


Figure 4.12: Simplified derivation tree of the best individual's phenotype

4.3.1 Experimental Settings

Figure 4.13 describes the AutoML-DSGE architecture and how it was merged with FEDORA for this study. Firstly, the Data Splitting step consists of randomly splitting the original dataset into training, validation and test subsets, for each run. In the Evolutionary Feature Engineering step, the best FEDORA individuals of the 5 experiments reported in Section 4.2 are selected and added to the list of pre-processors of the AutoML-DSGE framework.

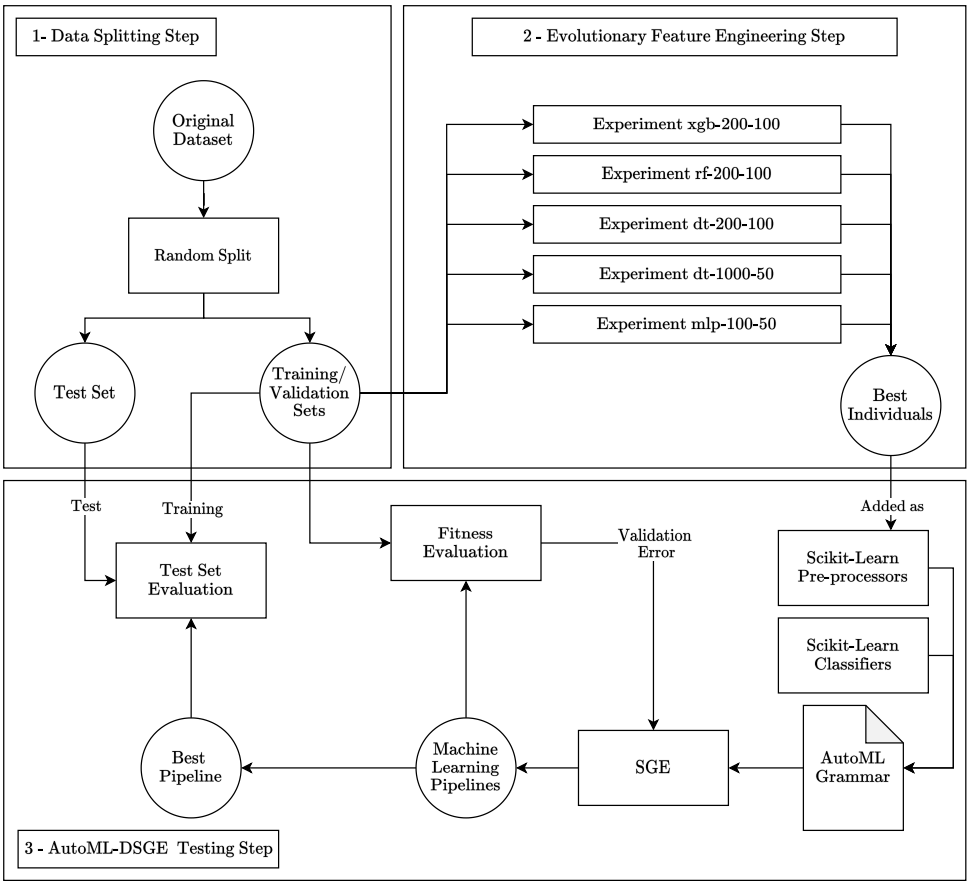


Figure 4.13: Experimental Architecture of FEDORA with AutoML-DSGE

In the final step (AutoML-DSGE Testing step), given a collection of pre-processing algorithms and classifiers defined in a Context-Free Grammar (CFG), the AutoML-DSGE framework searches for the optimal ML pipeline, i.e. the one that provides the lowest validation error, with SGE. At the end of the evolution, the ML pipeline with the lowest validation error is selected and is trained on the training and validation subsets to predict the test set.

The AutoML-DSGE framework was adapted from the original code, available in [AutoML-DSGE Github]. Some Scikit-learn components were not up-to-date with the current version of the package, namely some parameters and even algorithm class names. Therefore, the proper adjustments were performed.

Figure 4.14 describes the modifications made to the original grammar of AutoML-DSGE to include the FEDORA individuals. Notice that a FEDORA pre-processor

needs 2 parameters: the FEDORA experiment name that generated the individuals and their seed. For each run of *AutoML-Fedora*, 5 individuals are added as possible pre-processors, one for each experiment, matching the corresponding run seed. Therefore, the seed parameter is **None** in the grammar because it is redefined programmatically based on the seed of the current run.

<preprocessing> ::= <fedora> | <fedora> <bounding> | [...]

<fedora> ::= preprocessing:fedora seed:None experiment:<exps>

<exps> ::= rf-200-100 | xgb-200-100 | dt-200-100 | dt-1000-50 | mlp-100-50

Figure 4.14: AutoML-DSGE grammar modifications

This experimental study consists of comparing the *AutoML-FEDORA* and *AutoML-Baseline* groups that are defined by the performances of AutoML-DSGE on the original dataset with or without the possibility of adding the best FEDORA individuals as pre-processors, respectively. These experiments will allow us to have further insights on whether the FEDORA transformations are as good as common FE methods. If a FEDORA transformation is chosen by AutoML-DSGE as a preprocessor for a given run, FEDORA is effectively matching common FE techniques. Otherwise, FEDORA transformations might be more useful in a different scenario other than solely performance maximization, namely when there is a need to drastically reduce data dimensionality while maintaining performance. Both groups will be compared with one another by performing a statistical comparison.

Table 4.11: AutoML with FEDORA Experimental Settings

Parameters	**AutoML-Baseline**	**AutoML-FEDORA**
Population	100	
Generations	100	
Runs	30	
Elitism	5%	
Crossover Rate	0.9	
Mutation Rate	0.1	
Min. Tree Depth	5	
Max. Tree Depth	17	
Selection	Tournament (size 2)	
Fitness	1 - Balanced Accuracy	
Max. pipeline train time	5 minutes	
Stop criteria	5 generations without improvement	
Individuals as pre-processors	False	True

Table 4.11 presents the parameters chosen for both groups, adapted from [Assunção et al., 2020]. Notice that the inclusion of more pre-processing methods in the grammar increases the search space of AutoML-DSGE.

4.3.2 Results and Discussion

Concerning results, Figure 4.15 shows a boxplot comparison of both groups. The scores that were obtained using the FEDORA transformations are highlighted with a solid dot in the *AutoML-FEDORA* group. One can see that both groups achieve similar results over the 30 runs, with the median of *AutoML-FEDORA* being slightly higher. A single FEDORA transformation was selected, enabling a ML pipeline to obtain a testing score of 75.2% balanced accuracy, which is above the median values of both groups. The overall best pipeline achieved a score of 76.4% balanced accuracy on the *AutoML-FEDORA* group but did not use a FEDORA transformation.

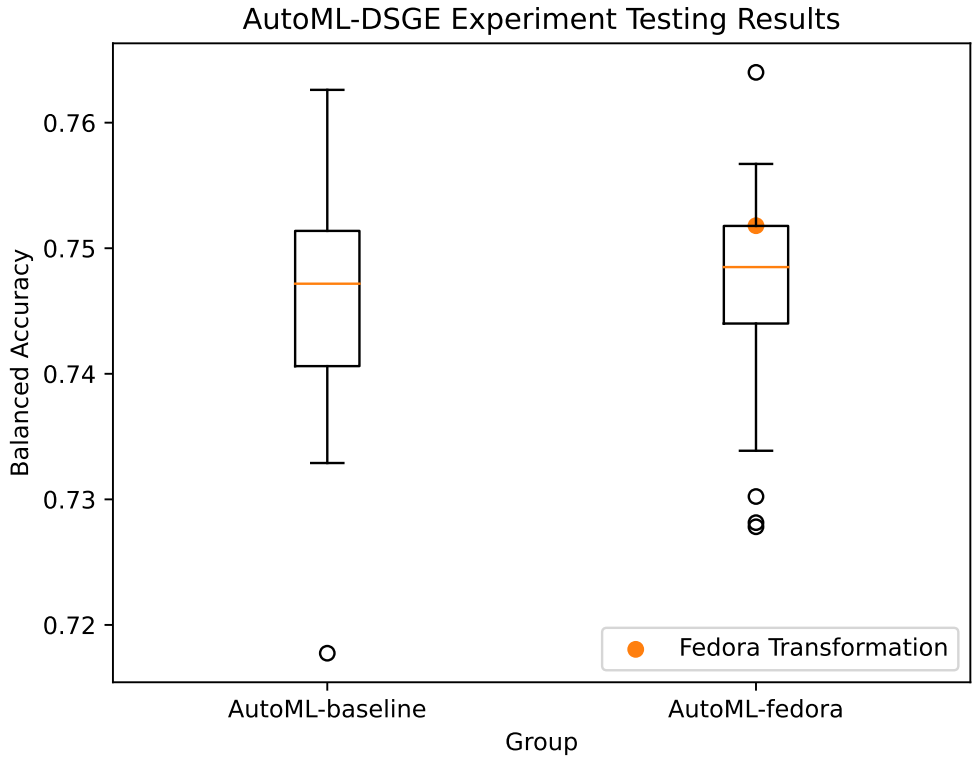
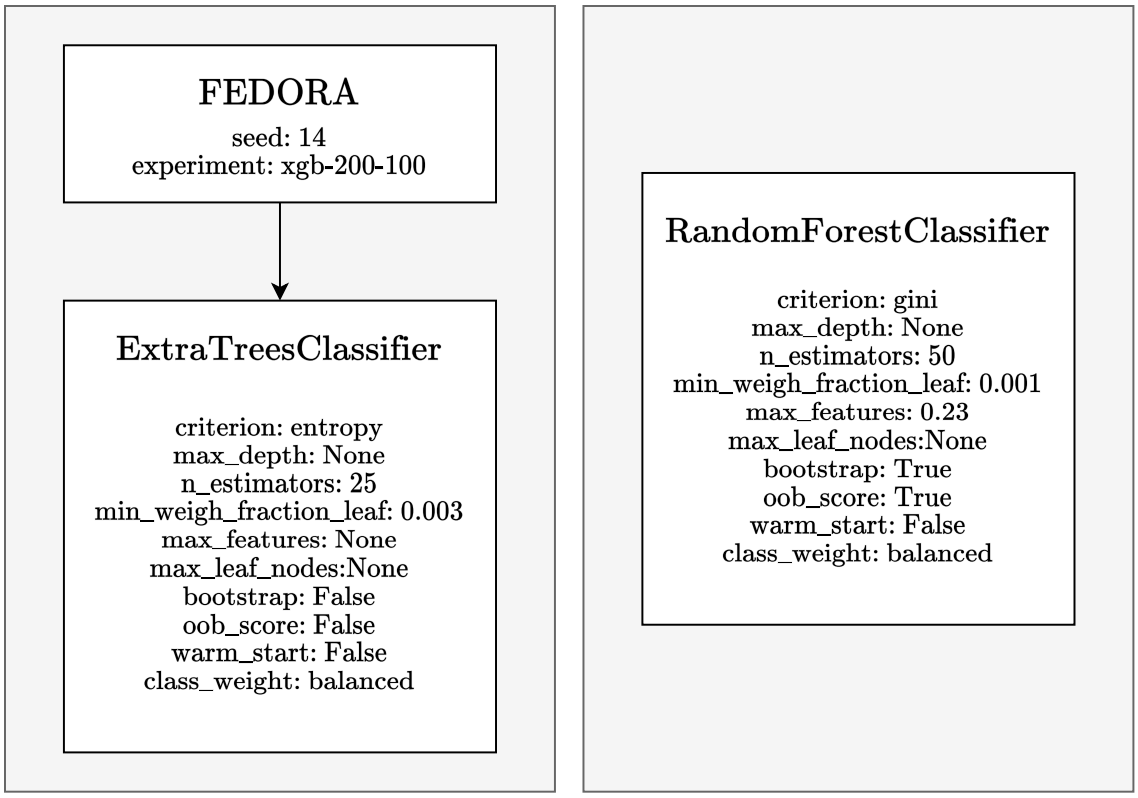


Figure 4.15: AutoML-DSGE Experiment Testing Results

Both pipelines are presented in Figure 4.16. The pipeline that uses FEDORA (Panel 4.16a), selects as the classifier an ExtraTreesClassifier, while the best pipeline (Panel 4.16b) does not require any pre-processing method and simply uses a RandomForestClassifer on the original dataset.

Figure 4.17 shows the components of the 30 best pipelines for each group, distinguishing between pre-processors and classifiers. Regarding the classifiers, both groups used the same ones, namely AdaBoost, GradientBoosting, ExtraTree and RandomForest, with similar ratios. As for pre-processing methods, a large number of pipelines did not use any pre-processing method while the remaining ones used a variety of them. The *AutoML-FEDORA* used a total of 8 pre-processing methods, with FEDORA being one of them, while the *AutoML-Basline* used 5.



(a) FEDORA Pipeline (b) Best Pipeline

Figure 4.16: AutoML-DSGE Machine Learning Pipelines

Despite only being selected once, these results point to the potential of FEDORA as an effective FE method to maximize the performance of ML models, since it was chosen by AutoML-DSGE, scoring above the median value. Concerning the statistical analysis, the Mann-Whitney U non-parametric test was selected to check for meaningful differences between both groups. For a significance level of 0.05, the null hypothesis failed to be rejected with a p-value of 0.48, meaning that there is no statistical evidence that both groups differ.

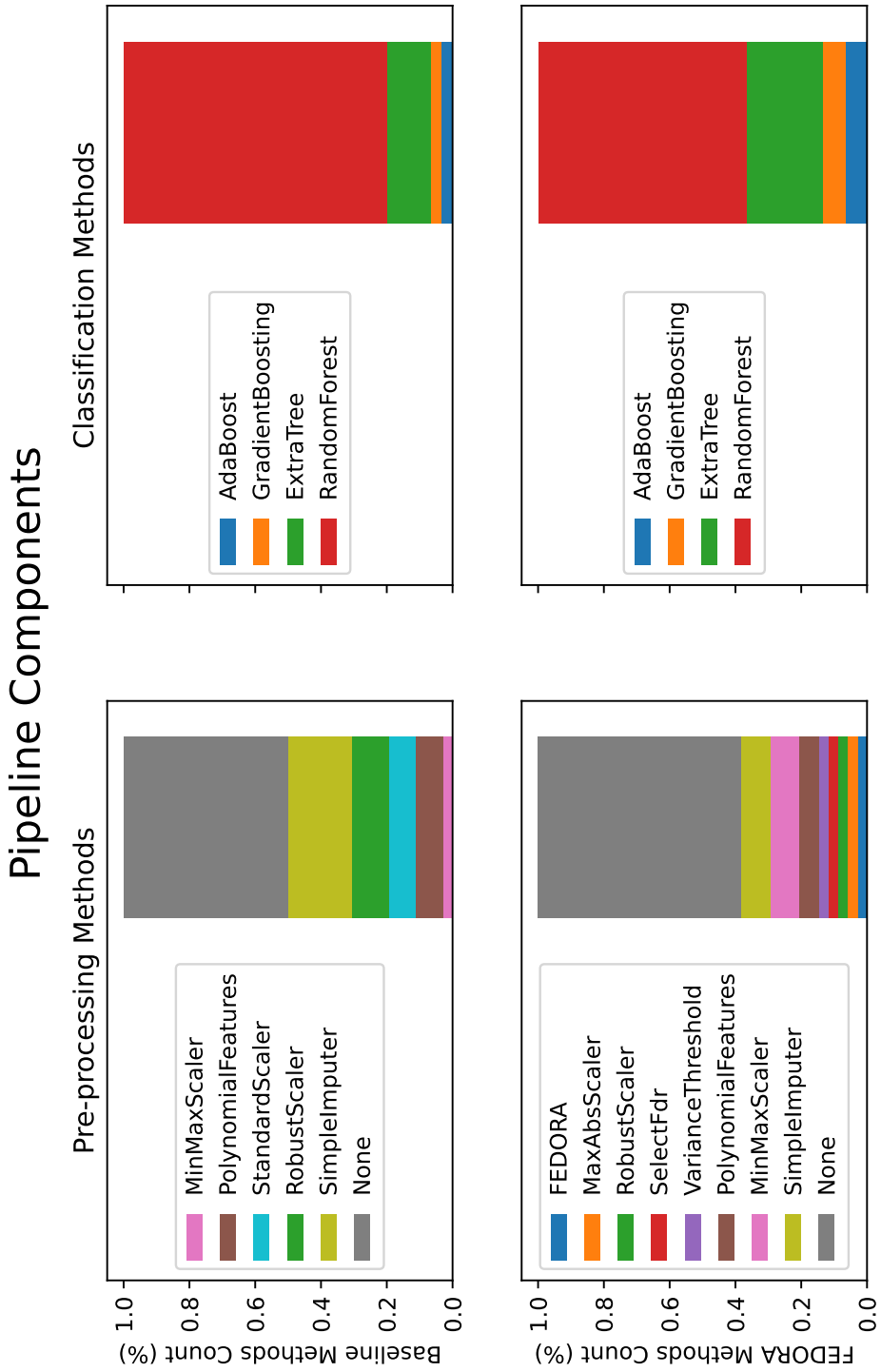


Figure 4.17: Pipeline Components

Chapter 5

Conclusion

Hearing Loss (HL) is becoming a concerning problem in modern society. A way to preemptively detect such loss is of utter importance since it allows for measures to be taken. As medical data is being produced, multi-disciplinary teams composed of medical staff, data scientists and engineers are working together to help solve medical field-related problems in an automated manner. Automated Machine Learning (AutoML) and Evolutionary Computation (EC) provide the necessary tools to such end.

The first objective of this work, i.e. automating the first steps of the Machine Learning (ML) pipeline by studying Feature Engineering (FE) techniques, was fulfilled by the proposal of an Evolutionary AutoML FE framework for a HL detection problem. The FEDORA framework aims to boost the performance of ML models by automatically enhancing original features through Feature Selection (FS) and Feature Construction (FC) evolutionary methods. Results show that the framework can evolve individuals that transform the original data set into a more predictive one, requiring fewer features.

Most of the framework's results are statistically better than commonly used FE techniques, such as Principal Component Analysis (PCA), Uniform Manifold Approximation and Projection (UMAP), Self-Organizing Maps (SOMs) and Autoencoders (AEs). Results point to the choice of the proxy as a decisive factor in the framework's behaviour, whether in terms of performance or FE.

By applying baseline classification algorithms, without any hyper-parameter tuning, the best result obtained is 76.2% balanced accuracy using an individual from the rf-200-100 instance, and a Extreme Gradient Boosting (XGB) as the testing model, using 57 total features (45 Original, 6 Engineered and 6 Complex) out of the 60 original ones. When using the least amount of features, the best result is 72,8% balanced accuracy using an individual from the dt-200-100 instance, and a Random Forest (RF) algorithm as the testing model, using 1 single complex feature.

By biasing the evolution, a better performance result was obtained when adding a penalty proportional to the number of features that the XGB algorithm was not using in the xgb-200-100 instance. A score of 76.2% balanced accuracy was

obtained by a XGB testing model, using 9 total features (5 Original, 2 Engineered and 2 Complex).

Regarding the second Objective, the full ML pipeline was completed by using the FEDORA transformations as possible pre-processors to the AutoML-DSGE framework [Assunção et al., 2020]. The results showed that a FEDORA preprocessor was chosen 1 out of 30 times. The corresponding pipeline was able to obtain a test score of 75.2% balanced accuracy, which places it above the average of the 30 runs. This means that FEDORA has the potential to effectively match common FE methods for performance improvement. The overall best pipeline obtained achieved a score of 76.4% balanced accuracy on the *AutoML-FEDORA* group but did not use a FEDORA transformation.

5.1 Future Work

Future work may consist of an in-depth study of how the framework behaves when using other models than the ones used in this experimental study, as well as applying it to other problems. Considering the evolution biasing work performed, which suggests potential performance and FE improvements on the proposed framework, further studying such techniques may prove helpful to achieve better results. Also, including a filter component in the fitness function, alongside the wrapper score, is a valid and powerful idea. Lastly, comparing the full ML pipeline to other full pipeline frameworks may be of use to contextualize and compare the framework on the AutoML and EC scenes.

References

Soha Ahmed, Mengjie Zhang, Lifeng Peng, and Bing Xue. Multiple feature construction for effective biomarker identification and classification using genetic programming. In Dirk V. Arnold, editor, *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014*, pages 249–256. ACM, 2014. doi: 10.1145/2576768.2598292. URL https://doi.org/10.1145/2576768.2598292.

Filipe Assunção, Nuno Lourenço, Bernardete Ribeiro, and Penousal Machado. Evolution of scikit-learn pipelines with dynamic structured grammatical evolution. *CoRR*, abs/2004.00307, 2020. URL https://arxiv.org/abs/2004.00307.

Anthony Brabazon, Michael O'Neill, and Seán McGarraghy. *Natural Computing Algorithms*. Natural Computing Series. Springer, 2015. ISBN 978-3-662-43630-1. doi: 10.1007/978-3-662-43631-8. URL https://doi.org/10.1007/978-3-662-43631-8.

Hayoung Byun, Chae Jung Park, Seong Je Oh, Myung Jin Chung, Baek Hwan Cho, and Yang-Sun Cho. Automatic prediction of conductive hearing loss using video pneumatic otoscopy and deep learning algorithm. *Ear and Hearing*, pages AUD–0000000000001217, 2022.

Noëlie Cherrier, Jean-Philippe Poli, Maxime Defurne, and Franck Sabatié. Consistent feature construction with constrained genetic programming for experimental physics. In *IEEE Congress on Evolutionary Computation, CEC 2019, Wellington, New Zealand, June 10-13, 2019*, pages 1650–1658. IEEE, 2019. doi: 10.1109/CEC.2019.8789937. URL https://doi.org/10.1109/CEC.2019.8789937.

Charles Darwin. *On the Origin of Species by Means of Natural Selection*. Murray, London, 1859. or the Preservation of Favored Races in the Struggle for Life.

A. E. Eiben and James E. Smith. *Introduction to Evolutionary Computing, Second Edition*. Natural Computing Series. Springer, 2015. ISBN 978-3-662-44873-1. doi: 10.1007/978-3-662-44874-8. URL https://doi.org/10.1007/978-3-662-44874-8.

Matthias Feurer and Frank Hutter. Hyperparameter optimization. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automated Machine Learning - Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 3–33. Springer, 2019. doi: 10.1007/978-3-030-05318-5_1. URL https://doi.org/10.1007/978-3-030-05318-5_1.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Auto-sklearn: Efficient and robust automated machine learning. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automated Machine Learning - Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 113–134. Springer, 2019. doi: 10.1007/978-3-030-05318-5_6. URL https://doi.org/10.1007/978-3-030-05318-5_6.

Dimitris Gavrilis, Ioannis G. Tsoulos, and Evangelos Dermatas. Selecting and constructing features using grammatical evolution. *Pattern Recognit. Lett.*, 29 (9):1358–1365, 2008. doi: 10.1016/j.patrec.2008.02.007. URL https://doi.org/10.1016/j.patrec.2008.02.007.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning - Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer, 2019. ISBN 978-3-030-05317-8. doi: 10.1007/978-3-030-05318-5. URL https://doi.org/10.1007/978-3-030-05318-5.

Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka: Automatic model selection and hyperparameter optimization in WEKA. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automated Machine Learning - Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 81–95. Springer, 2019. doi: 10.1007/978-3-030-05318-5_4. URL https://doi.org/10.1007/978-3-030-05318-5_4.

John R. Koza. *Genetic programming - on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press, 1993. ISBN 978-0-262-11170-6.

Nuno Lourenço, Francisco B. Pereira, and Ernesto Costa. SGE: A structured representation for grammatical evolution. In Stéphane Bonnevay, Pierrick Legrand, Nicolas Monmarché, Evelyne Lutton, and Marc Schoenauer, editors, *Artificial Evolution - 12th International Conference, Evolution Artificielle, EA 2015, Lyon, France, October 26-28, 2015. Revised Selected Papers*, volume 9554 of *Lecture Notes in Computer Science*, pages 136–148. Springer, 2015. doi: 10.1007/978-3-319-31471-6_11. URL https://doi.org/10.1007/978-3-319-31471-6_11.

Nuno Lourenço, Francisco B. Pereira, and Ernesto Costa. Unveiling the properties of structured grammatical evolution. *Genet. Program. Evolvable Mach.*, 17(3): 251–289, 2016. doi: 10.1007/s10710-015-9262-4. URL https://doi.org/10.1007/s10710-015-9262-4.

Nuno Lourenço, Joaquim Ferrer, Francisco Baptista Pereira, and Ernesto Costa. A comparative study of different grammar-based genetic programming approaches. In James McDermott, Mauro Castelli, Lukás Sekanina, Evert Haasdijk, and Pablo García-Sánchez, editors, *Genetic Programming - 20th European Conference, EuroGP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings*, volume 10196 of *Lecture Notes in Computer Science*, pages 311–325,

2017. doi: 10.1007/978-3-319-55696-3_20. URL https://doi.org/10.1007/978-3-319-55696-3_20.

Nuno Lourenço, Filipe Assunção, Francisco B. Pereira, Ernesto Costa, and Penousal Machado. Structured grammatical evolution: A dynamic approach. In Conor Ryan, Michael O'Neill, and J. J. Collins, editors, *Handbook of Grammatical Evolution*, pages 137–161. Springer, 2018. doi: 10.1007/978-3-319-78717-6_6. URL https://doi.org/10.1007/978-3-319-78717-6_6.

Francisco Paim de Bruges Rodrigues Miranda. *HYTEA-HYBRID TREE EVOLUTIONARY ALGORITHM FOR HEARING LOSS DIAGNOSIS*. PhD thesis, Universidade de Coimbra, 2022.

Mariana Monteiro, Nuno Lourenço, and Francisco B. Pereira. FERMAT: feature engineering with grammatical evolution. In Goreti Marreiros, Francisco S. Melo, Nuno Lau, Henrique Lopes Cardoso, and Luís Paulo Reis, editors, *Progress in Artificial Intelligence - 20th EPIA Conference on Artificial Intelligence, EPIA 2021, Virtual Event, September 7-9, 2021, Proceedings*, volume 12981 of *Lecture Notes in Computer Science*, pages 239–251. Springer, 2021. doi: 10.1007/978-3-030-86230-5_19. URL https://doi.org/10.1007/978-3-030-86230-5_19.

Randal S. Olson and Jason H. Moore. TPOT: A tree-based pipeline optimization tool for automating machine learning. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automated Machine Learning - Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 151–160. Springer, 2019. doi: 10.1007/978-3-030-05318-5_8. URL https://doi.org/10.1007/978-3-030-05318-5_8.

Michael O'Neill and Conor Ryan. Grammatical evolution. *IEEE Trans. Evol. Comput.*, 5(4):349–358, 2001. doi: 10.1109/4235.942529. URL https://doi.org/10.1109/4235.942529.

Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A Field Guide to Genetic Programming*. lulu.com, 2008. ISBN 978-1-4092-0073-4. URL http://www.gp-field-guide.org.uk/.

PORDATA. https://www.pordata.pt.

Franz Rothlauf and Marie Oetzel. On the locality of grammatical evolution. In Pierre Collet, Marco Tomassini, Marc Ebner, Steven M. Gustafson, and Anikó Ekárt, editors, *Genetic Programming, 9th European Conference, EuroGP 2006, Budapest, Hungary, April 10-12, 2006, Proceedings*, volume 3905 of *Lecture Notes in Computer Science*, pages 320–330. Springer, 2006. doi: 10.1007/11729976_29. URL https://doi.org/10.1007/11729976_29.

Conor Ryan, Michael O'Neill, and J. J. Collins. Introduction to 20 years of grammatical evolution. In Conor Ryan, Michael O'Neill, and J. J. Collins, editors, *Handbook of Grammatical Evolution*, pages 1–21. Springer, 2018. doi: 10.1007/978-3-319-78717-6_1. URL https://doi.org/10.1007/978-3-319-78717-6_1.

AutoML-DSGE Github. https://github.com/fillassuncao/automl-dsge/.

SNS - Portal da Transparência. https://transparencia.sns.gov.pt/.

WHO - Hearing Loss. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.

Jamile Silveira Tomiazzi, Danillo Roberto Pereira, Meire Aparecida Judai, Patrícia Alexandra Antunes, and Ana Paula Alves Favareto. Performance of machine-learning algorithms to pattern recognition and classification of hearing impairment in brazilian farmers exposed to pesticide and/or cigarette smoke. *Environmental Science and Pollution Research*, 26:6481–6491, 2019.

Binh Tran, Bing Xue, and Mengjie Zhang. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Comput.*, 8(1):3–15, 2016a. doi: 10.1007/s12293-015-0173-y. URL https://doi.org/10.1007/s12293-015-0173-y.

Binh Tran, Mengjie Zhang, and Bing Xue. Multiple feature construction in classification on high-dimensional data using GP. In *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016, Athens, Greece, December 6-9, 2016*, pages 1–8. IEEE, 2016b. doi: 10.1109/SSCI.2016.7850130. URL https://doi.org/10.1109/SSCI.2016.7850130.

Ioannis G. Tsoulos, Alexandros T. Tzallas, and Dimitrios G. Tsalikakis. Prediction of COVID-19 cases using constructed features by grammatical evolution. *Symmetry*, 14(10):2149, 2022. doi: 10.3390/sym14102149. URL https://doi.org/10.3390/sym14102149.

Hengzhe Zhang, Aimin Zhou, and Hu Zhang. An evolutionary forest for regression. *IEEE Trans. Evol. Comput.*, 26(4):735–749, 2022. doi: 10.1109/TEVC.2021.3136667. URL https://doi.org/10.1109/TEVC.2021.3136667.

Xian-Da Zhang. *A Matrix Algebra Approach to Artificial Intelligence*. Springer, 2020. ISBN 978-981-15-2769-2. doi: 10.1007/978-981-15-2770-8. URL https://doi.org/10.1007/978-981-15-2770-8.

Appendices

Appendix A

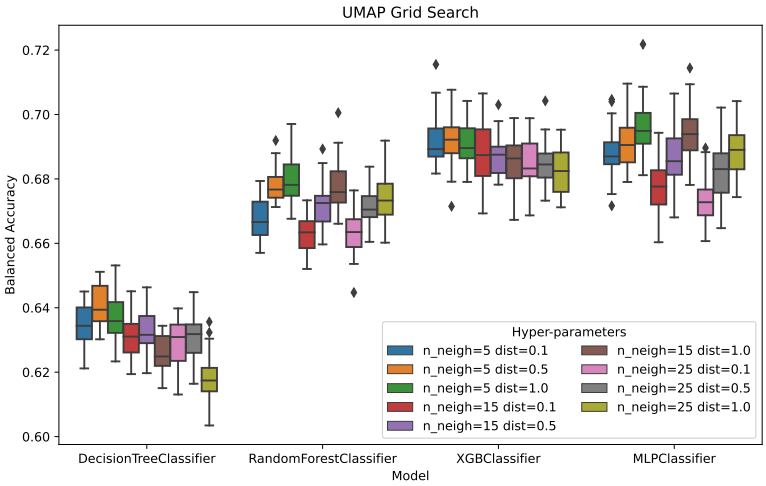
Hyper-parameter Search for Feature Engineering Methods

For each experiment in Section 4.2.2, some of the hyper-parameters of UMAP, SOMs and AEs were analysed concerning testing performance to check if the choice of the hyper-parameters was not negatively biasing the performance of these methods.

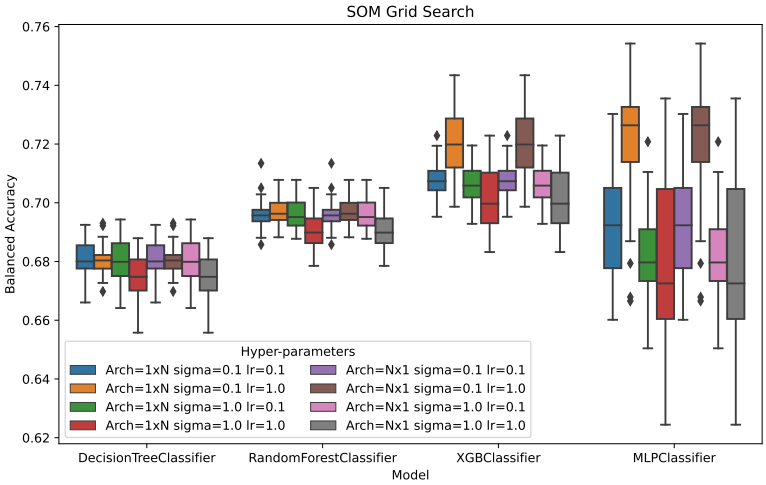
Regarding the UMAP algorithms, we performed a grid search for the n_neighbors and min_dist parameters, allowing them to take the values in $\{5, 15, 25\}$ and $\{0.1, 0.5, 1\}$ respectively. For the SOMs algorithms, a grid search was made for the sigma and learning rate, both allowed to take the values in $\{0.1, 1\}$, and for the architecture, i.e. the grid being Nx1 or 1xN. Concerning the AEs algorithms, we checked the activation functions, linear or ReLU, the number of epochs, 50 or 100, and the size of the hidden layer, with $\{20, 30, 40, 50\}$ as possible values. The number of components was set according to the current FEDORA individual of the given experiment.

Figures A.1, A.2, A.3, A.4 and A.5 display the results obtained for the rf-200-100, xgb-200-100, dt-200-100, dt-1000-50 and mlp-100-50 experiments, respectively.

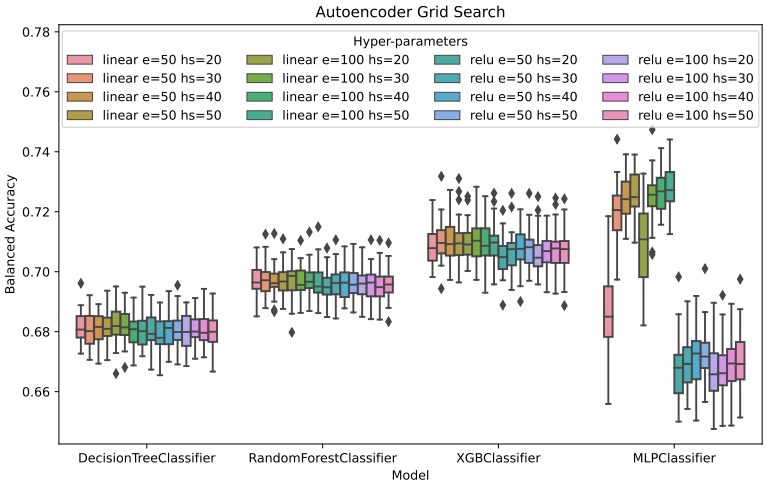
Notice that the default parameters used in Section 4.2.2 were 15 n_neighbors and 0.1 min_dist for the UMAP algorithm, Nx1 architecture, sigma of 1 and learning rate of 1 for the SOM algorithm and linear activation functions, with 50 epochs and 50 hidden size for the AE. The results show that the chosen hyper-parameters do not negatively bias the performance of the algorithms, since there are no meaningful differences across all testing models. The only considerable difference can be seen in the choice of linear or ReLU activation functions for the AE algorithm when using the Multi-Layer Perceptron (MLP) classifier as the testing model in Panels A.1c, A.2c and A.5c.



(a) UMAP Grid Search

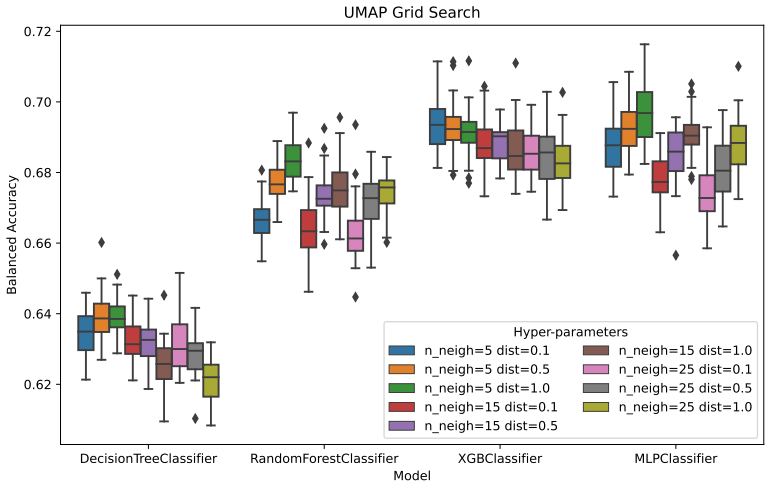


(b) Self-Organizing Maps Grid Search

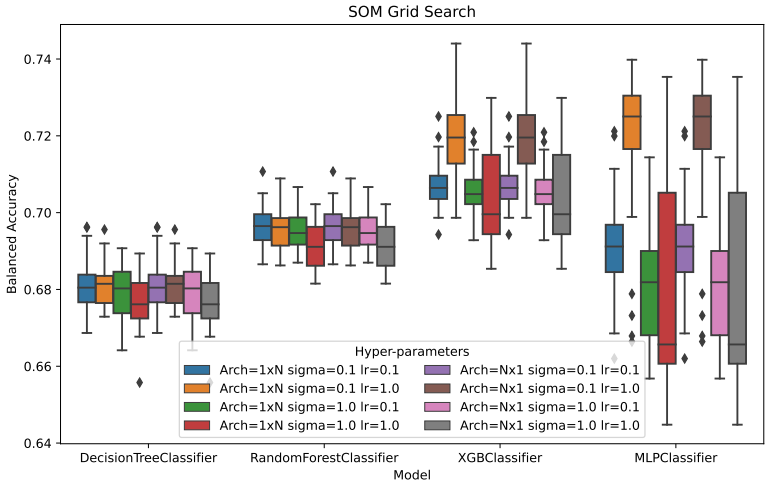


(c) Autoencoders Grid Search

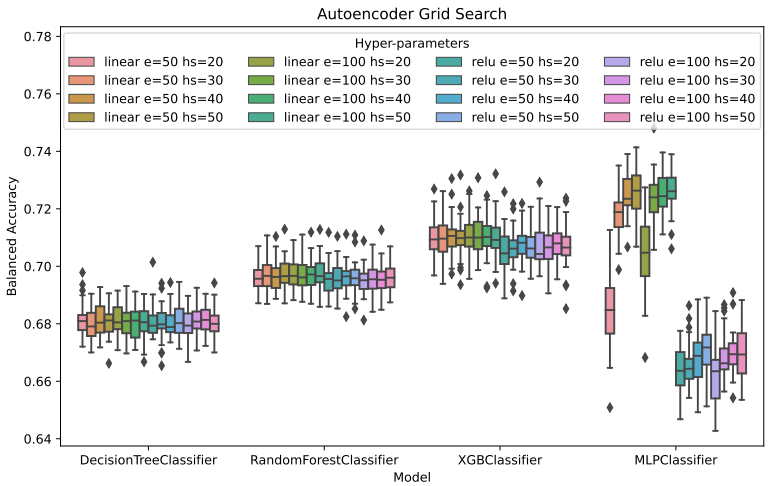
Figure A.1: Hyper-parameter Search for rf-200-100



(a) UMAP Grid Search

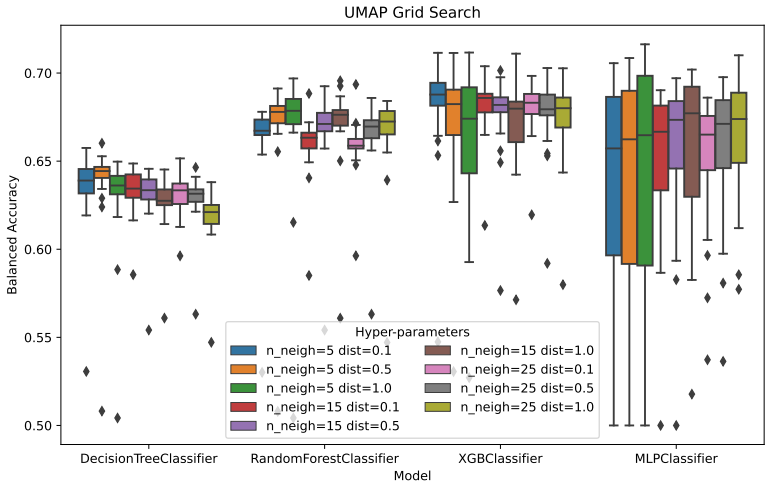


(b) Self-Organizing Maps Grid Search

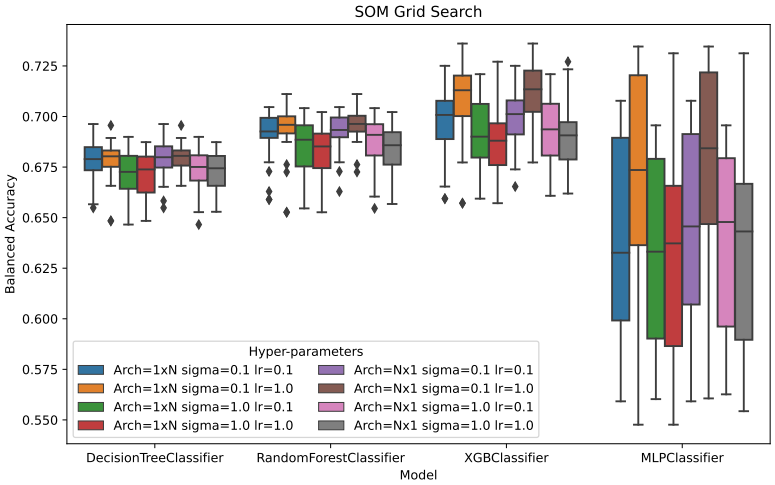


(c) Autoencoders Grid Search

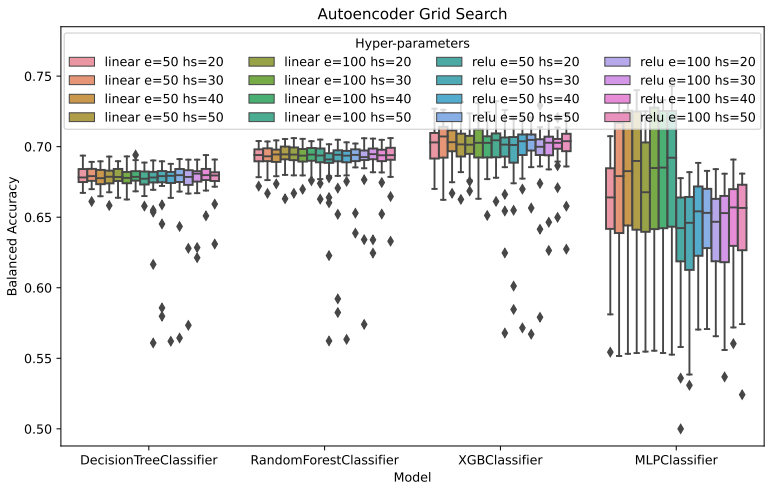
Figure A.2: Hyper-parameter Search for xgb-200-100



(a) UMAP Grid Search

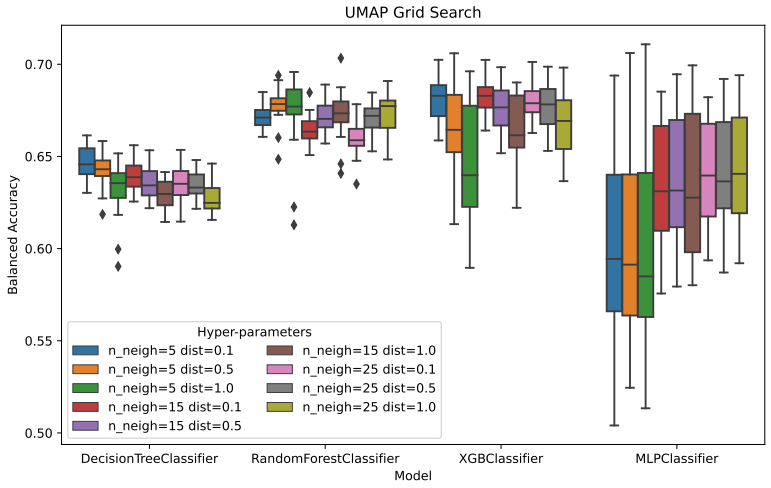


(b) Self-Organizing Maps Grid Search

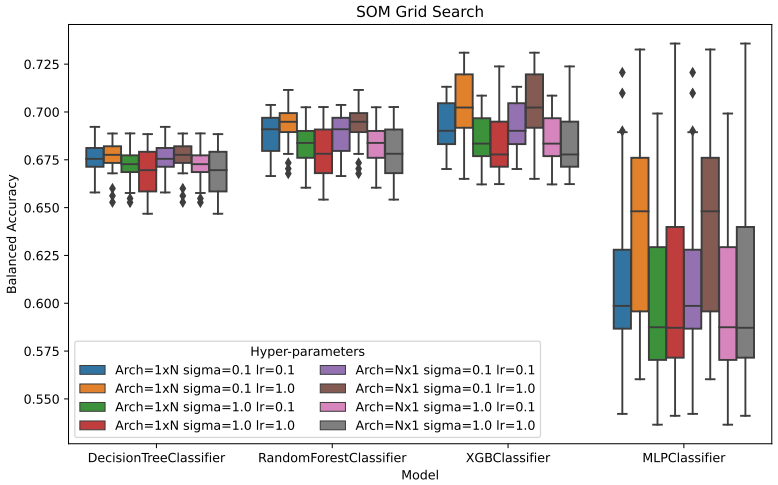


(c) Autoencoders Grid Search

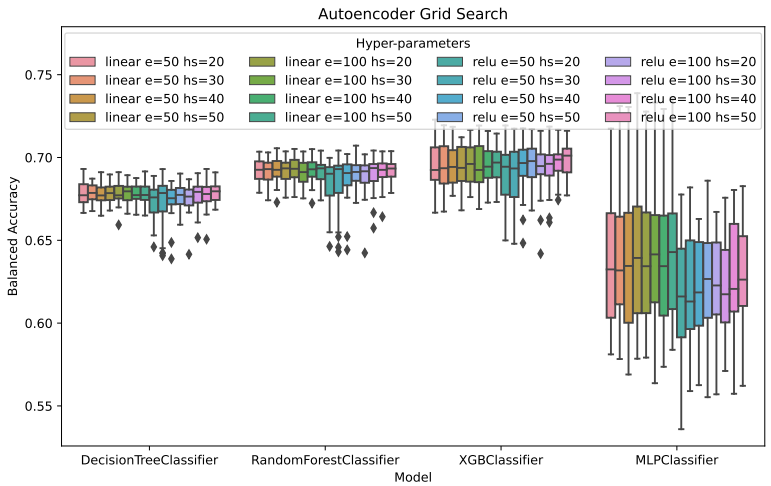
Figure A.3: Hyper-parameter Search for dt-200-100



(a) UMAP Grid Search

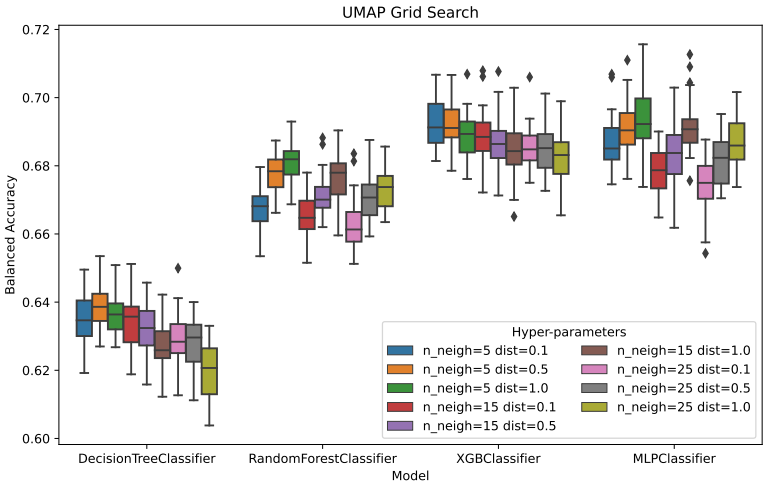


(b) Self-Organizing Maps Grid Search

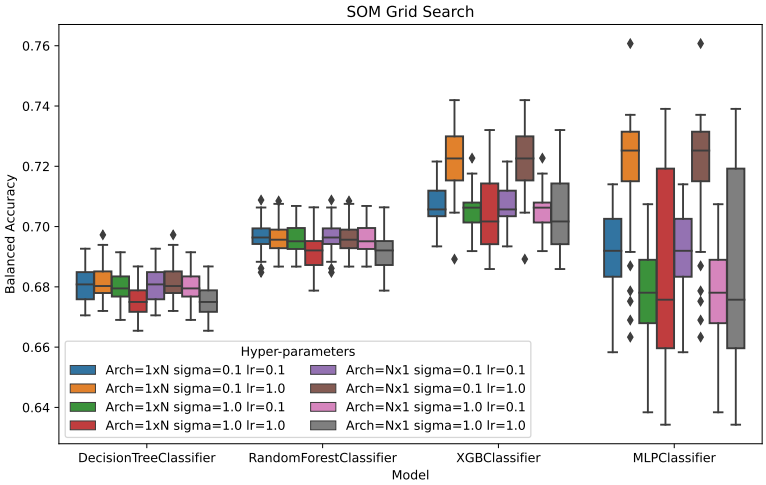


(c) Autoencoders Grid Search

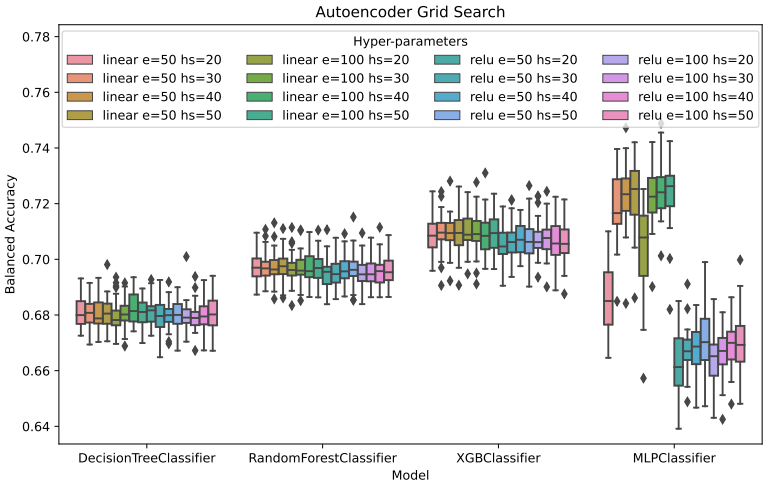
Figure A.4: Hyper-parameter Search for dt-1000-50



(a) UMAP Grid Search



(b) Self-Organizing Maps Grid Search



(c) Autoencoders Grid Search

Figure A.5: Hyper-parameter Search for mlp-100-50