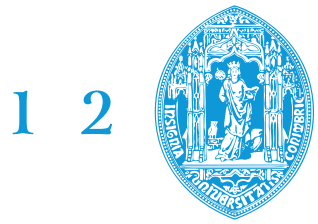
UNIVERSIDADE D
COIMBRA

Francisco Paim de Bruges Rodrigues Miranda

HyTEA - Hybrid Tree Evolutionary Algorithm for Hearing Loss Diagnosis

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

September 2022



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Francisco Paim de Bruges Rodrigues Miranda

HyTEA - Hybrid Tree Evolutionary Algorithm for Hearing Loss Diagnosis

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

September 2022

Acknowledgements

I would like to thank my counselor Nuno Lourenço for the opportunity to work with him, the constant availability, feedback, reassurance and genuine care. My co-counselor Evgheni Polisciuc for the precious inputs and for only barely mocking my design skills. My parents for the continuous support over all my life which led me here and for making me company over the phone wherever I have to drive alone. My grandparents for always being more excited than me for completing another step even if they don't understand why we need artificial intelligence, isn't natural intelligence enough? My brother for always having a sarcastic comment at hand. My dog Beckas for always going nuts when I show her a Tennis ball, that somehow makes it easier to go through the toughest days. To all my friends who contributed to the process of getting here with long talks or drinks, I'm sorry am not the most present person, but I do not forget and am grateful for all of you. Finally, to my girlfriend who had a critical role on this journey, for making it feel easy, for always lifting me up, reminding me to get some sleep and telling me everything will be ok whenever I'm despairing.

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 R&D Unit - UIDB/00326/2020 or project code UIDP/00326/2020.

Abstract

Hearing Loss affects an ever-growing number of people of all ages. It can occur due to a multitude of sources such as genetics, diseases, ageing, or noise exposure. If not treated properly and timely it may lead to socioeconomic difficulties such as poor job performance, hardship in finding a job, and social isolation.

In this work we propose HyTEA (Hybrid Tree Evolutionary Algorithm), a framework based on Evolutionary Computation to create Decision Tree like models to identify people that are likely to be diagnosed with hearing loss, so they can be called for screening by a health professional. To achieve this, we will use historic data about patients who have been tested for hearing problems and complement it with publicly available socioeconomic information. The models created should provide some understanding about the reason a decision is being made since this is key for the health professionals.

To build Decision Trees we usually rely on greedy induction algorithms which may result in overfitting of the training data. To counter this problem, HyTEA uses a combination of two Evolutionary Algorithms, namely Structured Grammatical Evolution and Differential Evolution to build the models. Additionally we propose variants of this method that allow evolving Gradient Boosted and Random Forest ensembles and present visualization tools to aid identifying the patients that are being wrongly classified.

The results show that HyTEA is capable of consistently modeling the problem space and predicting hearing loss with an accuracy of 73.8% and F1 of 74.1%, which was significantly higher than the results obtained with traditional Decision Trees. HyTEABoost proved capable of further improving the recall metric of single Decision Trees by up to 4.4%, the accuracy by 0.4% and the F1 by 1%.

The main contributions of this work are methods to generate Decision Trees, Random Forests and Gradient Boost ensembles with Evolutionary Computation, methods of predicting general hearing loss and if a screening should be performed and visualization tools to assist the decision of health professionals.

Keywords

Hearing Loss, Machine Learning, Evolutionary Computation, Structured Grammatical Evolution, Differential Evolution, Decision Tree, Gradient Boost, Random Forest.

Resumo

A perda de audição afeta um número cada vez maior de pessoas de todas as idades. Pode ter uma multitude de origens tais como genética, doenças, idade e exposição a ruído e, se não tratada devidamente, constitui um problema de saúde, social e económico. Se não compensada com aparelho auditivo, pode levar a dificuldades socioeconómicas como por exemplo má performance no trabalho, dificuldade em encontrar emprego e/ou isolamento social.

Neste trabalho propomos o HyTEA (Hybrid Tree Evolutionary Algorithm), uma framework baseada em Computação Evolucionária para criar modelos baseados em Árvores de Decisão para identificar pessoas prováveis de ser diagnosticadas com perda auditiva, para que sejam chamadas para um exame por um profissional de saúde. Para conseguir isto iremos usar dados históricos de pacientes testados para perda auditiva e dados publicamente disponíveis sobre informação socioeconómica. Os modelos criados devem permitir entender o porquê de uma decisão ser tomada, já que isto é crítico para a utilização por um profissional de saúde.

Para construir Árvores de decisão normalmente recorre-se a algoritmos de indução gulosos o que pode resultar em overfitting dos dados. Para contrariar este problema o HyTEA utiliza uma combinação de dois Algoritmos Evolucionários, nomeadamente o Structured Grammatical Evolution e a Evolução Diferencial para criar os modelos. Além disto, propomos variantes deste método que permitem evoluir modelos Gradient Boost e Random Forest e apresentamos ferramentas de visualização que ajudam na identificação de pacientes incorretamente identificados.

Os resultados mostram que o HyTEA é capaz de consistentemente modelar o espaço do problema e prever um diagnóstico de perda auditiva com uma accuracy de 73.8% e F1 de 74.1% o que é significativamente superior aos resultados obtidos com Árvores de Decisão tradicionais. HyTEABoost mostrou-se capaz de melhorar a performance da métrica de Recall de uma Árvore de Decisão individual por até 4.4%, a accuracy por 0.4% e o F1 por 1%.

As principais contribuições deste trabalho serão métodos para gerar modelos de Árvores de Decisão, Random Forest e Gradient Boost com Computação Evolucionária, métodos para previsão de se deve ser feito um exame auditivo e ferramentas de visualização para auxiliar a decisão dos profissionais de saúde.

Palavras-Chave

Perda Auditiva, Aprendizagem Computacional, Computação Evolucionária, Structured Grammatical Evolution, Evolução Diferencial, Árvore de Decisão, Random Forest, Gradient Boost.

Contents

Acronyms

ANN Artificial Neural Network.

AUC Area Under ROC Curve.

CD Cervical Dilation.

DBN Deep Belief Network.

DE Differential Evolution.

DSGE Dynamic Structured Grammatical Evolution.

DT Decision Tree.

EC Evolutionary Computation.

FN False Negative.

FP False Positive.

FPR False Positive Rate.

GA Genetic Algorithm.

GANN Genetic Algorithm Neural Network.

GB Gradient Boost.

GE Grammatical Evolution.

GP Genetic Programming.

HyTEA Hybrid Tree Evolutionary Algorithm.

KNN K-Nearest Neighbors.

LDA Linear Discriminant Analysis.

LR Logistic Regression.

MAE Mean Absolute Error.

ML Machine Learning.

MLP Multilayer Perceptron.

MSE Mean Squared Error.

NIHL Noise-Induced Hearing Loss.

NR Newton-Raphson.

PCA Principal Component Analysis.

PTS Permanent Threshold Shift.

R^2 Coefficient of Determination.

RF Random Forest.

RMSE Root Mean Squared Error.

ROC Receiver Operating Characteristics.

SGE Structured Grammatical Evolution.

SNE Stochastic Neighbor Embedding.

STS Significant Hearing Threshold Shift.

SVM Support Vector Machines.

t-SNE t-Distributed Stochastic Neighbor Embedding.

TN True Negative.

TP True Positive.

TPR True Positive Rate.

UDysRS Unified Dyskinesia Rating Scale.

UMAP Uniform Manifold Approximation and Projection.

List of Figures

List of Tables

Chapter 1

Introduction

According to the World Health Organization hearing loss affects around 466 million people. By 2050 it is expected for this number to double to around 900 million people. Of the people aged over 65, 30% are estimated to have hearing loss greater than 40dB. Developing and bringing mechanisms to lower the impact of hearing loss to the lives of the general population is therefore of ever growing importance.

The presented work, developed in the context of a scholarship in the research project "A4A - Audiology for All", aims to mitigate the impact of hearing loss in society by using Machine Learning to predict an hearing loss diagnosis ahead of a screening. This will allow health professionals to more accurately select potential patients to call for an official diagnosis reducing the negative effects that the hearing impediment might bring.

Given that understanding why the model is making a certain prediction is key for the medical professionals, our framework relies on models that can be understandable, namely Decision Trees (DTs). Usually, to build DTs, we rely on greedy induction algorithms which might be sub-optimal, resulting in models that might become overfitted to the training data. To overcome this issue we propose the usage of a hybrid Evolutionary Computation (EC) approach based on Structured Grammatical Evolution (SGE) [Lourenço et al., 2018] and Differential Evolution (DE) [Storn and Price, 1997]. The SGE algorithm will use a grammar to specify the syntactic restrictions of the DTs, and it will be responsible for evolving their macro structure. Then, the DE algorithm will optimize the numeric parameters of each model according to the real data.

Over the years, several approaches have been proposed aiming at using EC to build Decision Trees [Barros et al., 2012; Saremi and Yaghmaee, 2014], most of them using Genetic Programming (GP). However, the results show that, during the evolutionary process, the population tends to be plagued with invalid individuals, which slows down the evolutionary process, compromising the overall results. To tackle this, and eliminate the occurrence of invalid individuals, we rely on a Context-Free Grammar to limit the search space to a valid solution by specifying the syntax restrictions that should be followed to create DTs.

Besides proposing a method for evolving simple Decision Trees using Structured Grammatical Evolution and Differential Evolution, we also propose methodological variants to evolve both Random Forest and Gradient Boosted ensembles and present visualization tools to facilitate the interpretability of generated models. Finally we do a complete analysis on performance obtained with the different methods in the hearing loss prediction problem that we set out to solve.

Using traditional Decision Tree models we obtained on average an accuracy of 71.1% and an F1 of 55.7%, with HyTEA we improved these values to 71.9% and 72.1% respectively. Then our Evolutionary Gradient Boosting approach allowed us to further significantly improve our Recall metric from 65.0% to 72.2% with a slight cost for precision (-2.0%) and interpretability. Finally, our Evolutionary Random Forest approach got an accuracy of 70.4%, F1 of 72.9%, precision of 67.9% and recall of 79.4%, making it the best performing model when considering the F1 and recall metrics.

1.1 Objectives

The objectives of this work are:

1. To develop Decision Trees using an Hybrid Evolutionary Computation algorithm with Structured Grammatical Evolution and Differential Evolution.
2. To optimize the predictive performance of developed Decision Trees so that they can be considered powerful tools that health professionals can trust and rely on.
3. To obtain the simplest possible Decision Trees without undermining performance given that interpretability is key for health professionals to understand decisions.
4. To evolve other tree based classification models and evaluate the tradeoff between performance and interpretability.
5. To analyze the ability of the evolved classifiers to model the problem space.
6. To develop concepts of visual tools to assist the interpretation of the models by health professionals.

1.2 Contributions

According to the established objectives, this work has given multiple contributions, namely:

1. Problem agnostic Evolutionary Decision Tree and Random Forest induction techniques.

2. Model agnostic Evolutionary Gradient Boosting techniques to improve any probabilistic classification model with a Decision Tree ensemble.
3. Interpretable models to predict a hearing loss diagnosis based on demographic indicators and an health questionnaire.
4. Tools to utilize, visualize, interpret and evaluate the reliability of Decision Tree models.

1.3 Document Structure

In Chapter 2 we will give an introduction to all Machine Learning and Evolutionary Computation concepts required to understand the work at hand and will showcase relevant works on the fields of Medicine and Audiology in particular. In Chapter 3 we will detail the architecture and implementation of our general approach to evolve Decision Trees Random Forests and Gradient Boosting ensembles as well as discuss the preliminary experiments that lead to our final models. In Chapter 4 we detail the experiments ran with our solutions and compare the results to those obtained while experimenting with Sklearn. Finally in chapter 5 we visually analyze the problem space and how our best evolved classifiers model it and showcase tools that help health professionals understand the decisions made by our models.

Chapter 2

Background

In this Chapter the background knowledge on Machine Learning and Evolutionary Computation required to understand the developed work is discriminated. Besides the general literature knowledge, the final section of this Chapter presents relevant Machine Learning work in Audiology and Evolutionary Computation work in Medicine, showing both that Machine Learning is useful for Audiology problems and that Evolutionary Computation has been used before to solve general Medicine problems although no previous work was found of Evolutionary Computation in Audiology.

2.1 Machine Learning Background

In this section we'll discuss basic Machine Learning (ML) concepts with particular interest in Supervised Learning and Decision Trees. We'll cover basic models, evaluation metrics and pre-processing techniques, that is, all knowledge required to prepare data, build and evaluate models and choose optimal techniques.

Machine Learning is the usage of statistics and algorithms to program computers capable of optimizing a performance criterion based on observed data [Ayodele, 2010]. In Supervised Learning the result is a model with predictive capabilities, that is, the model learns what the expected value for a variable is based on a set of features observed in training data and is capable to predict what that value should be in new unseen data.

2.1.1 Types of Machine Learning

ML algorithms can be categorized according to their training data and goals into *supervised*, *unsupervised*, *reinforcement* and *semi-supervised* learning.

Image recognition tasks like predicting who is present in an image, or value prediction tasks like predicting the value of a house based on it's characteristics are examples of Supervised Learning since training data includes both features (the image in the first problem and the house characteristics in the second) and a tar-

get prediction value (the person depicted in the first problem and the real price in the second one). The former is an example of a classification problem since the output is a person, out of a finite set of possible people, which are known as classes, while the latter is an example of a regression problem since the output is a value in a continuous spectrum. [Bishop, 2006]

Unsupervised Learning consists of tasks on which training data does not have a target prediction value. These tasks can be clustering where similar groups of patterns are grouped, density estimation where we determine the distribution of data or projection of data into lower dimensions for visualization. [Bishop, 2006]

Reinforcement Learning happens when actions must be mapped to situations with the goal to maximize a reward function with the actions influencing not only the immediate reward but the reward in subsequent situations as well therefore these algorithms must learn by trial and error and consider delayed rewards. An example application of these type of learning is the development of a cleaning robot that must find the most trash before he needs to recharge while making sure he has enough battery to make it back to his charging station, and must make decisions based on his current battery level and previous experiences in finding his way back to the station. [Sutton and Barto, 1998]

Finally, Semi-supervised Learning is a a mix between Supervised and Unsupervised Learning as it deals with both labelled and unlabelled data. It's goals are the same as Supervised Learning however it tries to make the most out of the unlabelled data to improve predictions by application of techniques that either assume labels for the unlabelled data or request the labelling of key data. It is particularly helpful to deal with Supervised Learning situations where obtaining labelled data is expensive for example due to the requirement of human annotators. [Chapelle et al., 2006]

2.1.2 Common techniques

Some of the most common ML techniques are Logistic Regression (LR), Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), Decision Tree (DT), Random Forest (RF) and Gradient Boost (GB).

Logistic Regression

LR is a binary classification model that fits a logistic function to data and outputs the probability of an instance belonging to a certain class given a feature. The probability of a feature vector ϕ corresponding to class c_1 or c_2 is defined in Equation 2.1 where w is the vector of weights correspondent to each feature. Training of the LR is therefore the optimization problem of finding the vector w that maximizes the probability that the output of the model, given any known feature vector ϕ, corresponds to the correct known classes. [Bishop, 2006]

$$p(c_1|\phi) = 1 - p(c_2|\phi) = \sigma(w^T\phi)$$
$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

Support Vector Machines

SVMs are Maximum Margin Classifiers and work in binary classification problems. The model obtained with a SVM is a linear model in the form of Equation 2.2 where w and b are parameters to be determined, x is the input vector and $\phi(x)$ is a chosen transformation of the input vector. SVMs assume the 2 classes to be either -1 or 1, and therefore if y(x) < 0 we assume class 1 and if y(x) > 0 we assume class 1. [Bishop, 2006]

$$y(x) = w^T\phi(x) + b \tag{2.2}$$

Assuming a linearly separable problem, the SVM will find the decision boundary (the values of w and b), that maximizes the margin, which is defined as the smallest distance between the decision boundary and any of the samples. [Bishop, 2006]

The transformation $\phi(x)$ allows to mold the feature space into a linearly separable problem, however this is not possible to all problems or may lead to generalization problems. To solve this situation a slack penalty is given to every sample in the wrong side of the margin and therefore the optimization problem becomes to find w and b that minimize the penalties while maximizing the margin. [Bishop, 2006]

It becomes important to control a parameter C, which is a slack penalty multiplier, and therefore controls the relative importance of minimizing the penalties and maximizing the margin. This parameter must be experimentally tuned and sets a trade-off between minimizing training errors and controlling model complexity, since giving too much importance to slack errors will lead to overfitting and ignoring them will lead to underfitting. [Bishop, 2006]

K-Nearest Neighbors

KNN is multiclass lazy learning algorithm. The method is called lazy since, instead of learning a distribution function, it utilizes training instances to directly predict the class of new unseen instances by considering the K instances most similar to the prediction target and counting the number of occurrences of each class followed by a majority voting prediction.

For example, given a classification problem with a single feature and 4 training instances, 2 of each class, where instances of class c_1 have a feature value of 2 and 4 while instances of class c_2 have a feature value of 8 and 9. An instance of an unknown class with feature value 7 would be at distances of 5 and 3 from

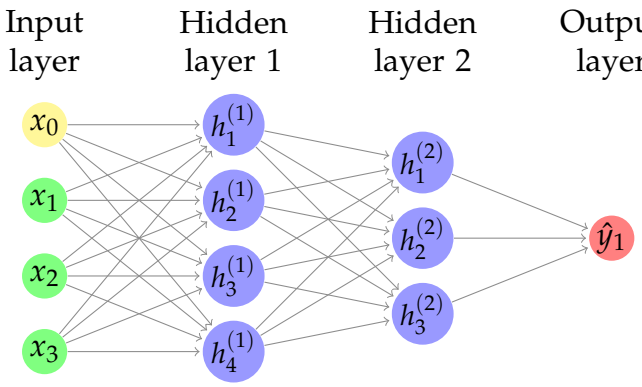


Figure 2.1: Example representation of a neural network with 4 inputs, 2 hidden layers and 1 output. x_i is i^{th} input, $h_j^{(k)}$ is the j^{th} neuron of the k^{th} hidden layer and $\hat{y}_1$ is the output value.

instances of class c_1 and distances of 1 and 2 from class c_2. If we set K = 3 then we would consider the 3 closest instances to be the ones with feature values 4, 8 and 9. Since the first one is of class c_1 and the other 2 belong to class c_2 then, by majority voting, the outputted prediction would be class c_2.

The KNN algorithm also supports probabilistic predictions by dividing the number of instances of a given class K_{c_i} among the K nearest neighbors by the total K according to Equation 2.3. [Bishop, 2006]

$$p(c_i) = \frac{K_{c_i}}{K} \tag{2.3}$$

In the example above we would say that $p(c_1) = \frac{K_{c_1}}{K} = \frac{1}{3}$ and $p(c_2) = \frac{K_{c_2}}{K} = \frac{2}{3}$.

Artificial Neural Network

ANNs are assembles of connected neurons where each neuron receives an input, performs some arithmetic operation and outputs a value. Typical ANN topologies have 3 or more layers. The input layer, that receives the feature values for an instance, the hidden layers that perform some calculations and the output layer that outputs a prediction based on the values outputted by the last hidden layer.

Figure 2.1 shows an example representation of a neural network with 4 inputs, 2 hidden layers and 1 output. x_i is i^{th} input, $h_j^{(k)}$ is the j^{th} neuron of the k^{th} hidden layer and $\hat{y}_1$ is the output value. Furthermore, each connection between 2 neurons has an associated weight w_h.

The output of a neuron is given by Equation 2.4, that is a function of the weighed sum of the inputs. [Bishop, 2006] This function is known as an activation function and can be any function that outputs a single value although common activation functions are the linear, relu and sigmoidal functions which are defined according

to Equations 2.5, 2.6 and 2.7 respectively.

$$o(x, w) = f(\sum_{i=1}^{n} w_j x_j) \tag{2.4}$$

$$f(x) = x \tag{2.5}$$

$$f(x) = max(0, x) \tag{2.6}$$

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

Building an ANN model requires therefore to first define the topology of the network and activation functions for each node and then follow a training procedure to find the optimal weights, that is, the weights that minimize an error function. Any optimization method works such as Evolutionary Computation techniques [M.B et al., 1997], however the most common and effective method is to take advantage of the error gradient by using the backpropagation algorithm [Bishop, 2006].

2.1.3 Tree Based Classifiers

Decision Tree is a ML technique that is of particular interest for the work at hand due to its explainability which allows human users to understand why the algorithm chooses to predict a certain class by following a series of "if then else" decisions.

DTs are composed of nodes in which decisions are made by doing a test over some feature in the format $X_j < x$ where X_j is a feature and x is the test value. If the value of X_j is lower than x then we proceed to the left child of that node, otherwise we proceed to the right child of that node. In the children 2 options are possible, either a new test over another parameter is done or a class prediction happens.

Figure 2.2 shows an example DT where t_i are nodes, X_j are features and c_k are classes. In this example an instance where X_1 is larger than 0.7 would be classified as class c_2 according to node t_2. However, if X_1 is smaller or equal to 0.7, then we would test if X_2 is smaller or equal than 0.5 according to node t_1 which if true would lead to a classification as class c_2 according to node t_3 and otherwise would lead to a classification as class c_1 according to node t_4.

Using DTs implies deciding on which parameters to test, in which order and what test values should be used. An usual approach to deciding on this is using the C4.5 algorithm [Quinlan, 1993], a greedy method in which at each level of the tree the best possible split is calculated from the attributes not previously used. The split is decided by first determining the best test value for each attribute and

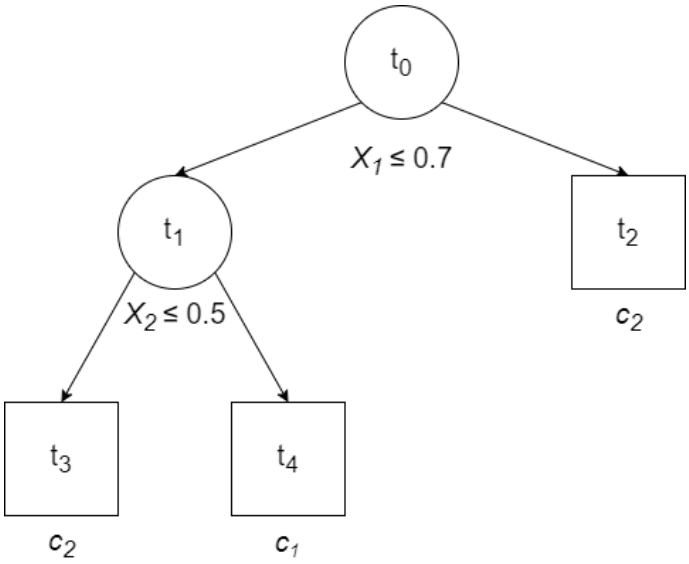


Figure 2.2: Example DT adapted from [Louppe, 2014]. t_i are nodes, X_j are features and c_k are classes.

then deciding on the best attribute. This decision is usually done by selecting the option that maximizes Information Gain or Gini Index. [Raileanu and Stoffel, 2004]

Random Forest

RF is an ensemble method to build more complex DT models without loss of generalization accuracy since building more complex DT models usually leads to overfitting. The method randomly selects subspaces of the feature space and builds a DT for each subspace using well known methods, then classification is done by allowing each DT to make a prediction and choosing the most common prediction as the final prediction of the RF ensemble. [Ho, 1995]

Gradient Boost

In theory, if we can predict by how much a model fails to predict the correct class, we can compensate for that error and obtain the correct classification. Gradient Boost (GB) is another tree based ensemble method which builds on that idea.

While in RF each tree has an equal contribution to the final prediction of the ensemble, in GB tree are built sequentially and each tree is based on the error of the previous ones. Trees are inducted with techniques such as those described in section 2.1.3, however from the second tree onward, the trees are inducted with the goal of predicting the residuals of the ensemble rather than directly predicting the classification.

For this to work however it is required that the output of the models is continuous, therefore in classification problems we allow the output to be a probability. In a binary problem, we predict the class 0 if the probability is lower than 0.5

Known Class	0	0	1	1
Tree 1	0.2	0.6	0.8	0.1
Ensemble Output	0.2	0.6	0.8	0.1
Ensemble Classification	0	1	1	0
Ensemble Residuals	-0.2	-0.6	0.2	0.9
Tree 2	-0.1	-0.2	0.2	0.3
Ensemble Output	0.1	0.4	1	0.4
Ensemble Classification	0	0	1	0
Ensemble Residuals	-0.1	-0.4	0	0.6
Tree 3	-0.05	-0.1	0	0.15
Ensemble Output	0.05	0.3	1	0.55
Ensemble Classification	0	0	1	1
Ensemble Residuals	-0.05	-0.3	0	0.45

Table 2.1: Example Gradient Boost ensemble building with 3 DTs in a binary problem assuming a learning rate of 1.

and class 1 otherwise. The prediction of the ensemble is therefore the rounded sum of the outputs of each individual Decision Tree. It is also common to apply a learning rate to the output of trees after the first one. [Hastie, 2009]

In Table 2.1 we show an example of the building of a Gradient Boost ensemble with 3 DTs assuming a learning rate of 1 for a binary problem with 4 instances. The known classes for each instance are 0, 0, 1 and 1. The first inducted DT outputs the values 0.2, 0.6, 0.8 and 0.1, leading to a classification of 0, 1, 1 and 0. The residuals of the ensemble are therefore the difference between the known class and the output of the ensemble, in this case -0.2, -0.6, 0.2 and 0.9.

The second DT will now be built to predict these residuals and outputs the values -0.1, -0.2, 0.2 and 0.3. Since the learning rate is 1, the output of the ensemble is now the sum of the outputs of the first 2 trees which is 0.1, 0.4, 1 and 0.4 and leads to the classifications 0, 0, 1 and 0. Following the same logic as before the ensemble residuals now are -0.1, -0.4, 0 and 0.6. Again the third DT will predict these residuals. The output of this third tree summed with the output of the previous 2 trees gives the final ensemble output of 0.05, 0.3, 1 and 0.55 which in turn leads to a perfect classification of 0, 0, 1 and 1 although the residuals are not yet 0.

2.1.4 Evaluation Metrics

Evaluating the performance of ML algorithms is of critical importance to appropriately select models and as such, choosing appropriate metrics is an essential part of ML. [Tharwat, 2018] Classification and regression problems use different metrics.

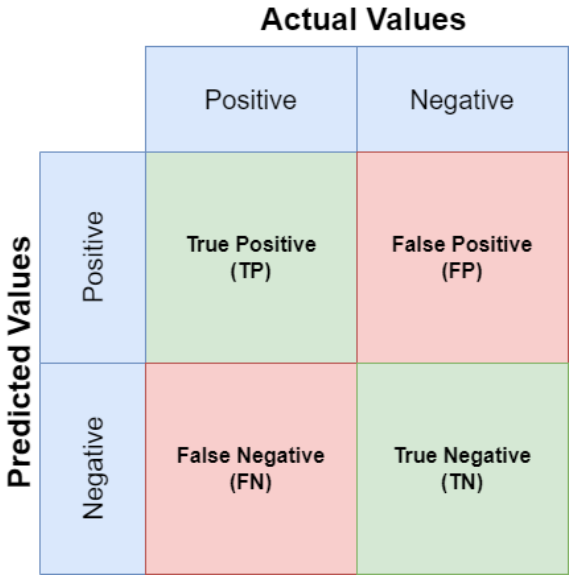


Figure 2.3: Example Confusion Matrix layout for a binary problem adapted from [Bhandari, 2020].

Confusion Matrix

A tool useful for diagnosing an algorithm is the confusion matrix which shows the predictions and the correct class so that we can get an overview of what is getting misclassified and what class is being selected instead of the correct one. In Figure 2.3 an example confusion matrix for a binary problem is shown where TP, FP, FN and TN correspond to True Positive, False Positive, False Negative and True Negative which can be shown as an absolute count or normalized by row or columns therefore showing a percentage of predictions or a percentage of real values respectively. The shown example can easily be extended to multiclass problems by simply adding a row and a column for each extra class.

Single Value Metrics in Classification

Classification uses mostly metrics based on how many instances are correctly classified such as precision, recall, specificity, accuracy and F1. In binary classification problems these metrics consider one class to be the positive class while the other one is the negative class and count the number of correct and incorrect predictions of each class, labelling instances as True Positive (TP) or True Negative (TN) when a correct prediction occurs and False Negative (FN) or False Positive (FP) when an incorrect prediction occurs. Each of these metrics is then calculated according to [Tharwat, 2018]:

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.9}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.10}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.11}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} = 2 * \frac{\frac{TP}{TP+FP} * \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} \tag{2.12}$$

As can be observed from the formulas, Precision answers the question "What percentage of positive predictions correspond to actually positive instances?" while Recall answers the question "What percentage of positive instances were predicted correctly?". Specificity works analogously to Recall, answering the question "What percentage of negative instances were predicted correctly?".

Accuracy measures how many predictions were correct overall. This metric however is problematic in unbalanced datasets where one of the classes has a much higher number of instances than the other. Let's say a given dataset has 990 instances of the positive class and only 10 instances of the negative class, then an algorithm that outputs a positive prediction regardless of input would obtain an accuracy of 99% which is not at all indicative of the algorithm's capacity to discriminate between classes.

Due to the problem described in the previous paragraph, it is common to use a combination of precision and recall or specificity to correctly evaluate an algorithm's performance. The goal of the F1 scoring metric is to get precision and recall values into one single metric so that interpretability is facilitated.

ROC Curve

Receiver Operating Characteristics (ROC) Curve is a graphical performance indicator. This method defines a set of thresholds above which the algorithm should decide on the positive class and for each threshold checks what the predictions would be and calculates the True Positive Rate (TPR) and False Positive Rate (FPR) which are then plotted as exemplified in Figure 2.4 where the yellow line is the ROC curve and the blue line is the reference ROC curve for a random classifier.

A perfect classifier would have TPR of 1 and FPR of 0 at all thresholds. Having high TPR and low FPR values leads to bigger areas under the ROC curve and therefore we can assume, the larger the area under the ROC curve, the better the classifier's performance. This is why we can use the Area Under ROC Curve (AUC) metric, which can vary between 0 and 1 and summarizes the information visible on the ROC plot. [Tharwat, 2018]

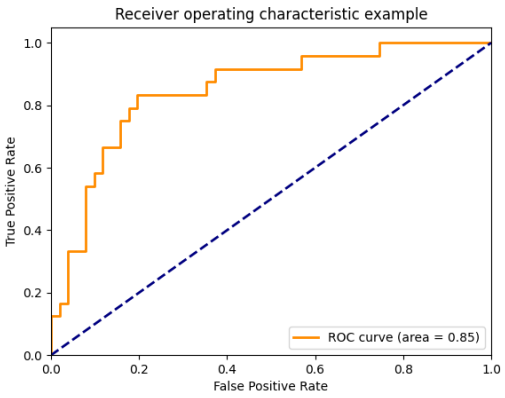


Figure 2.4: Receiver Operating Characteristics curve example adapted from [Sklearn documentation].

Regression Metrics

As mentioned before, regression problems utilize different evaluation metrics. Common examples of these metrics are the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2), which are given by Equations 2.13, 2.14, 2.15 and 2.16 respectively where n is the number of samples, p_i is a predicted value, a_i is the actual value, $\bar{p}$ is the mean predictions and $\bar{a}$ is the mean of actual values. [Witten et al., 2011]

$$MSE = \frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n} \tag{2.13}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}} \tag{2.14}$$

$$MAE = \frac{\sum_{i=1}^{n}|p_i - a_i|}{n} \tag{2.15}$$

$$\begin{aligned} R^2 &= \left(\frac{S_{PA}}{\sqrt{S_p S_a}}\right)^2 \\ S_{PA} &= \frac{\sum_{i=1}^{n}(p_i - \bar{p})(a_i - \bar{a})}{n-1} \\ S_P &= \frac{\sum_{i=1}^{n}(p_i - \bar{p})^2}{n-1} \\ S_A &= \frac{\sum_{i=1}^{n}(a_i - \bar{a})^2}{n-1} \end{aligned} \tag{2.16}$$

MSE is the most commonly used metric while RMSE is is often used so the error takes the same dimensions as the predicted values. MAE is an alternative to

RMSE since error will have the same dimensions as the predicted values. MSE tends to exaggerate the error due to outliers while MAE mitigates this problem. [Witten et al., 2011]

R^2 is a metric of the correlation between the prediction and the real values which can be 1 if all changes in the prediction are explained by changes in the real values, 0 if predictions are constant or anything below 0 if the regressor is arbitrarily worse than a constant prediction.

2.1.5 Pre-processing

Pre-processing data is a common requirement in ML due to most algorithms requiring numerical data, not tolerating missing values and being very sensible to scaling differences between variables. Major tasks in pre-processing are therefore engineering features, encoding categorical data, reducing dimensionality, solving missing values and scaling data.

Encoding Categorical Data

Categorical features are any feature whose values belong to a prespecified set of possibilities. [Witten et al., 2011] Encoding is the process of mapping the original categorical data into numeric data that can be used in ML. Two common techniques are used for this: Ordinal Encoding and One Hot Encoding. [Potdar et al., 2017]

In Ordinal Encoding, each possible categorical value is assigned an integer value. This technique can be a valuable choice if the categorical value can be ordered somehow, for example a difficulty feature can have the values "Low", "Medium" and "High" that can be mapped to 1, 2 and 3, enabling the logical order to be used by the ML solution.

In One-Hot Encoding a categorical feature with N possible values is transformed into N boolean features where each of these features is simply a boolean value indicating if a particular possible value was the value in that instance. Using the previous example, the difficulty feature would be mapped into three features: *isDifficultyLow*, *isDifficultyMedium* and *isDifficultyHigh* and a sample with the "Medium" value would have the feature *isDifficultyMedium* set to 1 while the two other features would be set to 0. This technique is valuable when values of a categorical feature have no logical order and therefore using Ordinal Encoding may induce ML algorithms in error.

Dimensionality Reduction

Irrelevant attributes have a negative effect on most machine learning models, furthermore redundant features are unnecessary and may lead to reinforcement of bias, therefore deciding on a optimal reduced subset of features is often an important step of the ML pipeline. Furthermore, models using less variables are more

compact and easier to interpret. Having domain knowledge on what attributes are and what they mean or imply in a specific problem allows manually selecting relevant attributes, however, that is not always possible and may lead to ignoring powerful previously unknown predictors. [Witten et al., 2011]

There are 2 main methods of pre-selecting features, filter and wrapped methods. Filter methods make an assessment on the general characteristics of the data, this assessment usually outputs an heuristic of the feature's quality which allows directly evaluating it's quality and comparing to others. This heuristics can be statistical inference tests like t-Student, ANOVA and Kruskal-Wallis which are applied to a feature and the associated class, outputting a test statistic that shows how likely the feature is to come from the same distribution as the target class. [Marques de Sá, c2001.] Another option is using correlation metrics such as Pearson, Kendall or Spearman correlations. This metrics also help removing features that are redundant by selecting features that are highly correlated to the class variable but are not correlated among each other. [Witten et al., 2011]

Wrapper methods utilize a learning algorithm in the feature selection step. An example would be to train a DT model, select the subset of most important features according to that model (the one's that are chosen in earlier splits) and then proceed to train a KNN on that subset. Another option would be to train a linear model such as an SVM and use the weights of each feature as an estimator of the feature quality.

Missing Values

Missing values correspond to data that was not collected, either because collection failed or because it did not make sense to collect a specific value in a specific case. Nevertheless, many ML algorithms are not tolerant to missing values and require them to be solved.

Solutions to this problem include, disregarding instances where values are missing and replacing missing values by some constant like the mean, median or mode of that value in other instances. Table 2.2 exemplifies the three mentioned methods. In categorical data it is often a solution to make missing values a new possible category.

A more complex solution is to use a strategy based on the KNN algorithm. Instead of considering all instances, the K most similar instances to the target (instance with a missing value) are calculated based on non missing values. Once selected, a weighed mean of those K instances is used to replace the missing value. The contribution of each instance to the weighed mean is based on some measure of similarity to the target instance. [Troyanskaya et al., 2001]

Scaling Data

Algorithms are very sensible to differences in scales between features, tending to give more importance to features whose values have higher variability even

Original Values	Solved w/ Mean	Solved w/ Median	Solved w/ Mode
7	7	7	7
Missing	5.6	6	7
3	3	3	3
7	7	7	7
Missing	5.6	6	7
5	5	5	5
6	6	6	6
Missing	5.6	6	7

Table 2.2: 8 samples of a given feature with 3 missing values. Missing values were solved using the mean, median and mode that have the values of 5.6, 6 and 7 respectively for the non missing values.

though they are not more discriminative. Pre-scaling features is therefore necessary to ensure the performance of the ML solution.

Scaling is usually done feature by feature with the goal to move all features value range into [-1; 1], [0; 1] or normalized spaces. Common techniques are Min-Max Scaling which translates and scales data into a defined [min, max] range, Standardization which normalizes features by subtracting the mean and scaling to unit variance and Maximum Absolute Scaling, which simply divides every value by the maximum absolute value in the original value set, ensuring a final range of values in [-1; 1]. Equations 2.17, 2.18 and 2.19 show how the calculations are done for each of these scaling techniques, with X being a feature vector and min, max, $mean$ and std being the functions that calculate the minimum, maximum, average and standard deviation of the vector respectively.

$$X_{MinMax} = \frac{X - min(X)}{max(X) - min(X)} \tag{2.17}$$

$$X_{MaxAbs} = \frac{X}{|max(X)|} \tag{2.18}$$

$$X_{Standardization} = \frac{X - mean(X)}{std(X)} \tag{2.19}$$

Table 2.3 exemplifies what happens to a dataset of 2 features, A and B, when applying each of the mentioned scaling techniques.

2.1.6 Visualization

Visualization of high dimensionality data is important to Machine Learning since it allows finding relationships between similar points and understanding where developed models may fail to get accurate predictions. Typically this visualization is done by projecting and plotting the high dimensionality data into 2 dimensions. There are many techniques capable of doing this projection such as Princi-

Original		Min-Max		Max Abs		Standard	
A	B	A	B	A	B	A	B
89.00	1.00	0.91	0.62	0.89	0.14	1.47	0.00
-20.00	3.00	0.00	0.77	-0.20	0.43	-1.18	0.53
15.00	-7.00	0.29	0.00	0.15	-1.00	-0.33	-2.14
3.00	4.00	0.19	0.85	0.03	0.57	-0.62	0.80
30.00	-2.00	0.42	0.38	0.30	-0.29	0.03	-0.80
-11.00	6.00	0.07	1.00	-0.11	0.86	-0.96	1.34
100.00	1.00	1.00	0.62	1.00	0.14	1.73	0.00
23.00	2.00	0.36	0.69	0.23	0.29	-0.14	0.27

Table 2.3: Example scaling of features A and B with Min-Max Scaling, Maximum Absolute Scaling and Standardization.

pal Component Analysis (PCA) or Linear Discriminant Analysis (LDA), however not all techniques are suitable for visualization.

Data projection techniques suitable for visualization focus on clustering similar points together and keeping dissimilar points apart so that these relationships are evident when data is plotted.

Stochastic Neighbor Embedding (SNE), Uniform Manifold Approximation and Projection (UMAP) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are examples of dimensionality reduction techniques designed for visualization. In this work we focus on SNE and it's variant t-SNE which was first proposed in [van der Maaten and Hinton, 2008].

Stochastic Neighbor Embedding

SNE is a technique that obtains an optimal projection by first creating a random projection and then slightly adjusting each projected point over multiple iterations.

Algorithm 2.1 Pseudocode for Stochastic Neighbor Embedding.

1: $X \leftarrow$ HIGHDIMENSIONALDATA
2: $n \leftarrow$ SIZE(X)
3: $p_{[n,n]} \leftarrow$ GETSIMILARITIES(X)
4: $Y \leftarrow$ RANDOMPROJECTION(X)
5: **repeat**
6: $\quad q_{[n,n]} \leftarrow$ GETSIMILARITIESOFPROJECTION(Y)
7: $\quad C \leftarrow \sum_i \sum_j p_{[j,i]} \log \frac{p_{[j,i]}}{q_{[j,i]}}$
8: $\quad Y \leftarrow$ ADJUSTPROJECTION(Y, Cost)
9: **until** C can't be further reduced
10: **return** Y

As seen in algorithm 2.1, SNE starts by calculating a similarity matrix for the original data points. This similarity is calculated based on euclidean distances and according to Equation 2.20 where σ_i is proportional to the density of points

around point i. The goal of this parameter is to have points in sparse clusters be equally similar between them as points in dense clusters.

$$p_{[j,i]} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \tag{2.20}$$

The next step in SNE is to create a random projection of the data into a lower dimensional space (typically 2D). Then, similarities are calculated between the points in the projection according to Equation 2.20 however, here σ_i is set to the constant $\frac{1}{\sqrt{2}}$. Other constants could be used, however using this value allows simplification of the calculations since the Equation can be rewritten as Equation 2.21.

$$q_{[j,i]} = \frac{\exp(-\|x_i - x_j\|^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2)} \tag{2.21}$$

Finally SNE will iteratively adjust each point in the projection using a gradient descent method that minimizes a cost function C which is defined as the sum of Kullback-Leibler divergences over all data points as shown in Equation 2.22. [van der Maaten and Hinton, 2008]

$$C = \sum_i \sum_j p_{[j,i]} \log \frac{p_{[j,i]}}{q_{[j,i]}} \tag{2.22}$$

t-Distributed Stochastic Neighbor Embedding

t-SNE is a variant of SNE that aims to solve 2 problems. First canonical SNE is computationally intensive due to the complicated gradient calculation. To tackle this t-SNE changes Equation 2.21 into Equation 2.23 and after calculating all similarities in the higher dimensional space defines that $p_{[j,i]}$ and $p_{[i,j]}$ should be set to $\frac{p_{[j,i]} + p_{[i,j]}}{2n}$ where n is the number of data points.

$$q_{[j,i]} = \frac{\exp(-\|x_i - x_j\|^2)}{\sum_{k \neq l} \exp(-\|x_k - x_l\|^2)} \tag{2.23}$$

Finally, t-SNE aims to solve the problem of overcrowded visualizations where points are on top of each other although they have no association. For this it changes the normal distribution for a Student t-distribution with one degree of freedom leading to Equation 2.23 being once again transformed into Equation 2.24. [van der Maaten and Hinton, 2008]

$$q_{[j,i]} = \frac{(1 + \|x_i - x_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|x_k - x_l\|^2)^{-1}} \tag{2.24}$$

2.2 Evolutionary Computation Background

In this section we'll cover relevant aspects of Evolutionary Computation (EC) starting with the basics of the Genetic Algorithm, covering Genetic Programming, Grammatical Evolution, Structured Grammatical Evolution, Differential Evolution and finally analyzing works on usage of Evolutionary Computation for Decision Tree induction.

EC is a set of nature inspired stochastic optimization algorithms that try to emulate the process of evolution in nature. [Brabazon et al., 2015] While many different techniques exist such as Genetic Programming (GP) [Langdon and Qureshi, 1995] and Grammatical Evolution (GE) [Ryan et al., 1998], most are derived from the Genetic Algorithm that will be explained in the next section.

2.2.1 Evolutionary Algorithms

Evolutionary Algorithms iterate on a population of individuals. Each individual has a genotype which represents a possible solution in the optimization problem and is presented as an array. Algorithm 2.2 shows the pseudocode for the GA with elitism and each component of the algorithm is further explained in the next paragraphs.

The algorithm runs for multiple generations and stops when either a predefined number of generations has been run or a stopping condition such as the optimum solution being found occurs. At every generation each individual's quality is evaluated by running a problem specific fitness function.

Once all individuals have been evaluated there is a selection process where individuals are picked to generate children using a crossover method. Selection is usually done by tournament, where N individuals are randomly picked to compete in a tournament, being the best of those selected for crossover. An alternative to tournament is fitness proportional selection where any individual can be selected with a probability proportional to the individual's quality, this method can however lead to great loss of diversity in situations where the fitness difference between the best individuals and the rest of the population is too drastic.

Crossover is the process of mixing individuals, typically 2, to generate new ones. There are many ways to perform crossover, one of the most common one being the two-point crossover where 2 indexes of the genotype's array are randomly selected and then the content of each individual's genotype between those 2 indexes is swapped. Figure 2.5 exemplifies the generation of 2 children using this crossover method.

Once crossover is done new individuals are mutated, usually by randomly changing one or more of the elements on the genotype. Finally, Survival Selection occurs, to choose the individuals that will represent the next generation where the whole process will be repeated. Survival Selection can be the complete replacement of the generation by the generated children and elite individuals, or can

Algorithm 2.2 Pseudocode for the Genetic Algorithm with Elitism adapted from [Luke, 2013].

```
popsize ← desired population size
n ← desired number of elite individuals

P ← {}
for popsize times do
    P ← P ⋃ {new random individual}
end for
Best ← NULL
repeat
    for each individual P_i ∈ P do
        AssessFitness(P_i)
        if Best = NULL or Fitness(P_i) > Fitness(Best) then
            Best ← P_i
        end if
    end for
    Q ← {the n fittest individuals in P, breaking ties at random}
    for (popsize − n)/2 times do
        Parent P_a ← SelectWithReplacement(P)
        Parent P_b ← SelectWithReplacement(P)
        Children C_a, C_b ← Crossover(P_a, P_b)
        Q ← Q ⋃ {Mutate(C_a), Mutate(C_b)}
    end for
    P ← SurvivalSelection(P, Q)
until Best is the ideal solution or we have run out of time
return Best
```

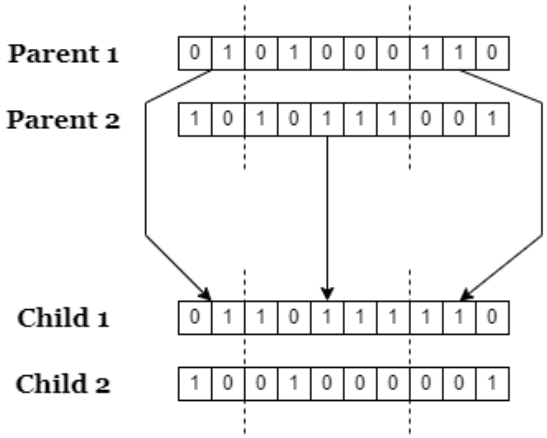


Figure 2.5: Generation of 2 children with two-point crossover. Image adapted from [Brabazon et al., 2015].

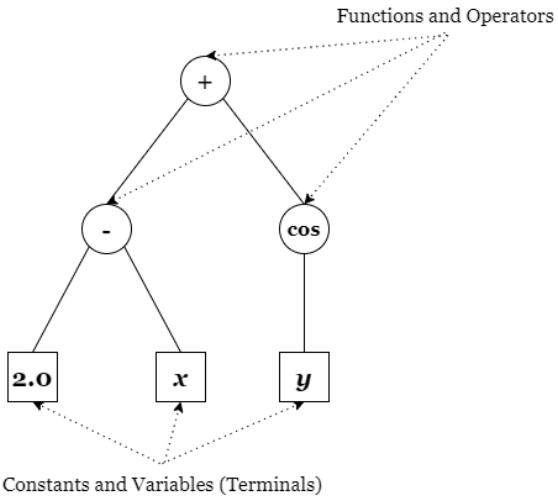


Figure 2.6: Example Genetic Programming individual representing the program (2 - x) + cos(y). Image taken from [Brabazon et al., 2015].

include a mix of the fittest individuals among the current generation and the generated children.

It should be noted that the crossover and mutation processes are done probabilistically, with crossover usually taking high probability values, above 0.5 and mutation taking low probability values, usually under 0.1. When crossover does not occur, the 2 parent individuals simply become children. Mutation may or may not occur probabilistically for each element of the genotype. It is also possible to use elitism which is the process of sending a certain percentage of the best individuals of the previous generation directly into the next one. Once the stopping criterion is met, a final evaluation of the individuals is done and the best one is returned by the algorithm as the optimal solution.

An alternative to the above presented generational approach is the steady state approach in which generated individuals get immediately inserted into the population while selection decides which individuals should die. This alternative allows for faster adaptation to environment changes since newly created children will be available to generate children of their own earlier. [Brabazon et al., 2015]

2.2.2 Genetic Programming

Genetic Programming is a new technique where instead of evolving solutions we evolve programs to calculate the solution. Here the typical array representation of the genotype is replaced by a tree structure where each node is an operator or function, and the leaves are constants or variables. Figure 2.6 exemplifies this representation.

The algorithm is mostly the same as the GA, there are however some modifications regarding some components. Firstly, the usual mutation and crossover operators no longer can be applied since the representation is a tree instead of an array. Here the common crossover operator is to swap random subtrees between

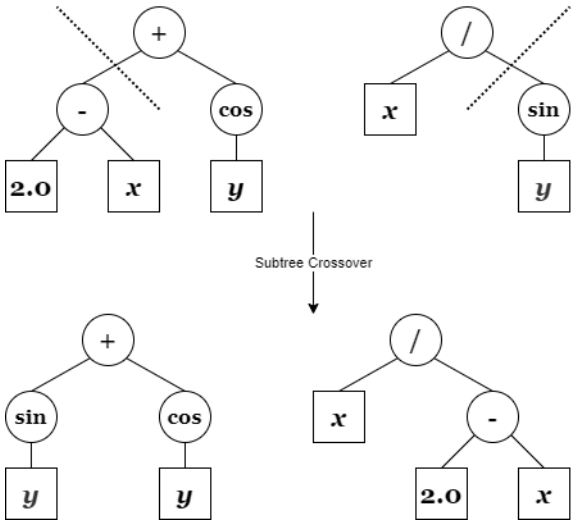


Figure 2.7: Example crossover in Genetic Programming. Image adapted from [Brabazon et al., 2015].

2 individuals while the mutation operator can be to randomly replace a subtree by a randomly generated one. Figure 2.7 exemplifies how crossover is done in GP.

The initialization method also becomes a problem since the representation here does not have a fixed size or structure. Two different methods are of common usage, *grow* or *full*. In *full*, method trees are generated with a maximum predefined size and making sure every possible node exists. For this, only non-terminal symbols (operators and functions) are selected on internal nodes, and once maximum depth is reached only terminal symbols such as constants and variables are selected. In the *grow* method there are no limitations to what symbols are selected at each node, except when maximum depth is reached, and once again only terminal symbols can be selected. Figure 2.8 exemplifies the trees generated by these methods. It is also common to use the *ramped-half-and-half* method where half the population is initialized with the *grow* method and the other half with the *full* method, effectively generating the most diversity of both structure and content. [Brabazon et al., 2015]

Another issue with this approach is bloat or overfitting which happens when trees keep growing without providing better performance or overfitting the data so that performance is improved in the training dataset while likely undermining the generalization capability. Bloat is also capable of drastically slowing down progress because memory is limited and parsing and testing the trees becomes much more computationally expensive. [Luke and Panait, 2006]

There are many methods of bloat control, in this document we expose mostly ad-hoc methods as in that they will only limit the propagation of bloated individuals rather than stopping individuals from bloating however other more advanced techniques exist such as Operator Equalisation [Dignum and Poli, 2008] and Dynamic Limits [Silva and Costa, 2004]. [Luke and Panait, 2006] compares many of these methods and concludes most of them to have effective problem independent settings and that they should always be paired with a predefined tree size

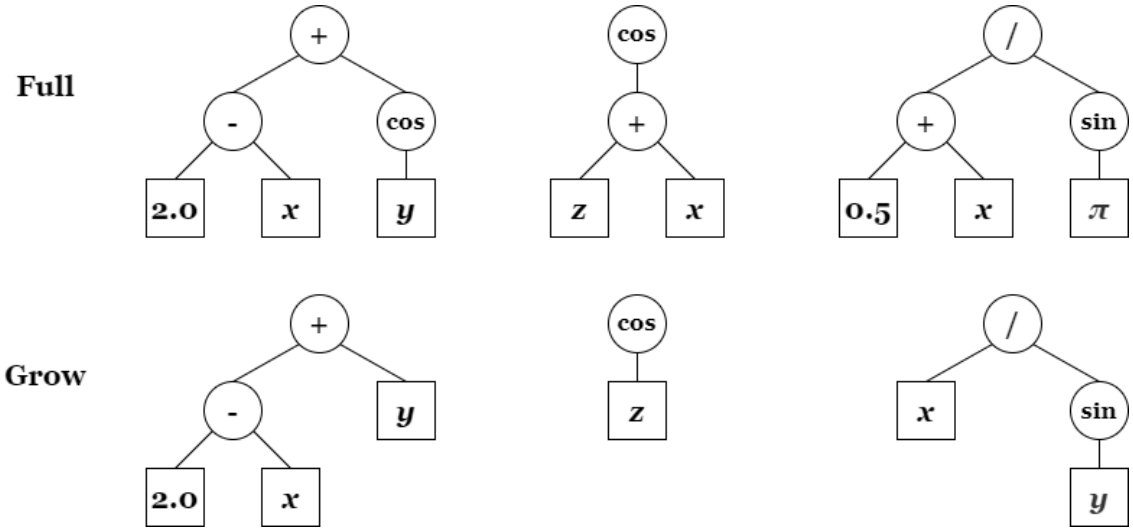


Figure 2.8: Example trees generated with *full* and *grow* methods for a maximum depth of 3. Image adapted from [Brabazon et al., 2015].

limit.

One of those methods is the Tarpeian method where a probability is defined for which trees having a size above population average are automatically evaluated with the worst possible fitness and can therefore only be selected in case a tournament happens to only have this type of individuals. [Luke and Panait, 2006]

Parametric parsimony pressure is a method that penalizes tree size in the fitness function by subtracting the size times some constant parameter from the raw fitness value. This method works as a multi-objective optimization problem with a predefined objective importance. It is also possible to implement a multi-objective optimization based on Pareto dominance, where only solutions that are non-dominated are kept, this however tends to lead to small size low performance and big size high performance solutions which is far from the desired outcome. [Luke and Panait, 2006]

Double Tournament and Proportional Tournament are techniques that use two consecutive tournaments during selection. In Double Tournament the first tournament selects by fitness, then the winners of the tournament are directed to a second tournament that selects by size with only those winning both tournaments getting selected. Proportional Tournament is similar however each tournament can be a fitness or size selector with the choice being done at random each time. [Luke and Panait, 2006]

2.2.3 Grammatical Evolution

Grammatical Evolution is a variant of GP that allows defining a set of rules which the structure of programs must follow using a context-free grammar. Instead of genotypes being represented as trees, there is a clear distinction between genotype and phenotype, with genotypes being a linear structure from which the phenotype tree structure is derived. This has the advantage of allowing the applica-

N = $\{< start >, < expr >, < term >, < op >\}$
T = $\{+, -, /, *, (,), x_1, 0.5\}$
S = $\{< start >\}$

Production set:

<start> ::= <expr><op><expr> (0)
| <expr> (1)

<expr> ::= <value><op><value> (0)
| (<value><op><value>) (1)

<op> ::= + (0)
| - (1)
| / (2)
| * (3)

<value> ::= x_1 (0)
| 0.5 (1)

Figure 2.9: Example of a Context-Free-Grammar in the Backus-Naur form. N is the set of non-terminal symbols, T is the set of terminal symbols and S is the grammar's axiom. Adapted from [Lourenço et al., 2019].

tion of generic search strategies and crossover operators to GP problems while guaranteeing no nonviable individuals are generated. [Brabazon et al., 2015]

One of the main drawbacks of GE approaches is the high redundancy and low locality that hinders the effectiveness of the search process. With the goal of avoiding these drawbacks while keeping GE's advantages, Structured Grammatical Evolution was created. [Lourenço et al., 2016]

2.2.4 Structured Grammatical Evolution

In Structured Grammatical Evolution (SGE) the genotype is composed by an array of arrays where each subarray corresponds to a single non-terminal symbol and is composed by integers that decide which grammatical production should be used next. The length of each subarray is determined by the maximum number of expansions of each non-terminal, which means recursive productions in the grammar must be rewritten in a non-recursive format. Figure 2.9 shows an example grammar which is used to derive an expression based on an SGE individual in Table 2.4. The right side of the Table shows the unused genotype's integers, the subarrays correspond to the non terminals $< start >$, $< expr >$, $< op >$ and $< value >$ respectively.

The crossover operator for SGE is to randomly choose and swap complete corresponding genes (subarrays) between the 2 individuals as exemplified in Table 2.5. The mutation operator is to randomly change the integer values inside the

Table 2.4: Extracting the phenotype of a SGE individual based on the genotype and on the grammar present in Figure 2.9.

Derivation step	Integers left
<start>	[[0], [1, 0], [2, 0, 3], [1, 1, 0, 0]]
<expr> <op> <expr>	[[], [1, 0], [2, 0, 3], [1, 1, 0, 0]]
(<value> <op> <value>) <op> <expr>	[[], [0], [2, 0, 3], [1, 1, 0, 0]]
(0.5 <op> <value>) <op> <expr>	[[], [0], [2, 0, 3], [1, 0, 0]]
(0.5 / <value>) <op> <expr>	[[], [0], [0, 3], [1, 0, 0]]
(0.5 / 0.5) <op> <expr>	[[], [0], [0, 3], [0, 0]]
(0.5 / 0.5) + <expr>	[[], [0], [3], [0, 0]]
(0.5 / 0.5) + <value> <op> <value>	[[], [], [3], [0, 0]]
(0.5 / 0.5) + x_1 <op> <value>	[[], [], [3], [0]]
(0.5 / 0.5) + $x_1 *$ <value>	[[], [], [], [0]]
(0.5 / 0.5) + $x_1 * x_1$	[[], [], [], []]

Parents	Children
[[0], [1, 0], [2, 0, 3], [1, 1, 0, 0]]	[[1], [1, 0], [0, 3, 1], [1, 1, 0, 0]]
[[1], [0, 0], [0, 3, 1], [1, 0, 0, 1]]	[[0], [0, 0], [2, 0, 3], [1, 0, 0, 1]]

Table 2.5: Example SGE crossover based on the individuals for the grammar in Figure 2.9. In this case the genes corresponding to non-terminals $< start >$ and $< op >$ were swapped to generate 2 distinct children.

subarrays to another valid option as exemplified in Table 2.6.

This algorithm still has some drawbacks. Grammars must be pre-processed to remove recursion and does not allow limiting tree size directly which makes it hard to transfer from work with other GP approaches. Dynamic Structured Grammatical Evolution (DSGE) is a new version of SGE which solves this problem. [Lourenço et al., 2018]

In DSGE tree depth limit is predefined and tree size is controlled by ensuring only terminal symbols are selected once maximum tree depth has been reached. Furthermore, instead of each gene having a calculated maximum size, gene size becomes variable. When generating the phenotype, if a gene does not have enough productions specified, new integers are randomly added to the gene until the needs are satisfied. This allows recursive productions to be used unlimitedly until maximum tree depth is reached.

Original	Mutated
[[0], [1, 0], [2, 0, 3], [1, 1, 0, 0]]	[[1], [1, 0], [2, 0, 3], [1, 1, 0, 0]]
[[1], [0, 0], [0, 3, 1], [1, 0, 0, 1]]	[[1], [0, 0], [0, 3, 1], [1, 1, 0, 1]]

Table 2.6: Example SGE mutation based on the individuals for the grammar in Figure 2.9. In this case the first individual was mutated so that production 1 will be used instead of production 0 for the $< start >$ non-terminal while the second individual was mutated so that the second time the non-terminal $< value >$ shows up production 1 will be used instead of production 0.

[**?**] showed SGE to be more effective than classic GE solutions, [Lourenço et al., 2016] shows it to have better locality and less redundancy than GE and [Lourenço et al., 2018] concludes DSGE to be never inferior to SGE while being superior in many benchmark tests. These results have been validated in [Lourenço et al., 2019] by creating robust models to predict glucose levels in diabetic patients, [Lourenço et al., 2020] that evolves models of energy demand using SGE and Differential Evolution and [Assunção et al., 2017] that evolves multi-layered neural networks.

2.2.5 Differential Evolution

Differential Evolution (DE) is a steady state Evolutionary Algorithm with analogous mutation and crossover mechanisms. [Storn and Price, 1997] Mutation and crossover always happen for every individual and in a sequence. Mutation does not directly affect an individual, it is done by randomly selecting 3 vectors x_{r1}, x_{r2}, x_{r3} and calculating a mutant vector according to:

$$v = x_{r1} + F \cdot (x_{r2} - x_{r3}) \tag{2.25}$$

where v is the mutant vector and F is a constant parameter which controls the amplification of the variation. This mutant vector is then crossed over using uniform crossover to generate one single child. [Storn and Price, 1997] specifies that the version of uniform crossover used should ensure at least one gene is swapped. Once a child is generated the selection step ensues. Here the generated child will compete directly with its parent, replacing him in the population if his fitness is better.

2.2.6 Evolving Decision Trees

[Barros et al., 2012] does a survey on works using EC to create DTs and makes a summary of choices taken for each phase of the evolutionary process. The vast majority of surveyed works rely on Genetic Programming and the survey concludes most works use a random initialization of trees with test values being constrained to guarantee the logic validity of the tests. Most works use the full or the *ramped-half-and-half* methods, while only 1 of the reviewed works uses the *grow* method.

Some works use a parametric parsimony pressure approach to counter overfitting. Works using this approach argue that the balance between parsimony and accuracy is critical for efficient evolution and that not finding the correct balance will make parsimony pressure a disadvantage. Works not using parsimony pressure usually do not defend this choice either since the evolution is slowed down due to larger trees and the bloat leads to overfitted DTs which do not perform well on test data.

Since most works use GP, the classical GP crossover operator is the most common. Mutation usually happens by replacing a subtree by a randomly generated one or by changing the values associated with tests. Tuning crossover and

mutation probabilities is usually done via trial and error in preliminary runs. The study also suggests tournament sizes should be no higher than 5, since that would lead to great loss of diversity.

Many of the reviewed works claim to get similar performance to classic DT induction approaches while getting smaller tree sizes due to parsimony pressure, although training takes longer. It is often suggested to parallelize the implementation to speed up the process however not many works actually do it and there are no comparisons on time gain.

[Saremi and Yaghmaee, 2014] makes a point that many invalid trees are created after applying the crossover and mutation operators and that one attribute may be examined more than once along the same path from root to leaf and doing a consistency check followed by a fix might facilitate exploration of the valid search space. [Saremi and Yaghmaee, 2014] solves this problem by pruning subtrees where a nominal attribute test is repeated or with invalid continuous tests for example, if a node tests $x < 10$, no subnode should test if $x < y$ where $y >= 10$ since passing the first test always ensures passing the second one in those conditions.

2.3 Related Work

In this section we present some of the Machine Learning work that has been developed on the field of Audiology and Evolutionary Computation work developed in Medicine. The presented Machine Learning works in Audiology show that the problem of identifying individuals with hearing loss is viable to solve via ML. Furthermore, the works on applications of Evolutionary Computation in Medicine showcase that EC, and specifically Structured Grammatical Evolution, is a viable approach to solve problems in the field as it has been successfully applied to many other problems in the field before although to the best of our knowledge no previous attempts have been made at predicting hearing loss with EC techniques.

2.3.1 Machine Learning in Audiology

There are many successful predictive machine learning studies in the field of Audiology. Most of these studies are focused on predicting a specific type of hearing loss such as noise induced [ElahiShirvan et al., 2020], sensorineural (deficiency of neural signal transfer from the cochlea to the auditory cortex) [Chen et al., 2021] and idiopathic sudden sensorineural [Park et al., 2020].

[Cao et al., 2021] does a review on the contributions and limitations of 8 papers [Aliabadi et al., 2014; ElahiShirvan et al., 2020; Farhadian et al., 2015; Greenwell et al., 2018; Zare et al., 2018, 2019; Zhao et al., 2018, 2019] using machine learning to predict Noise-Induced Hearing Loss (NIHL). This paper identifies exposure to noise above 85 dBA for over 8 hours and exposure to noises over 3 kHz as the most important risk factors for NIHL. Other identified factors that affect individ-

uals susceptibility to NIHL such as demography, hearing protection usage and mutations to genes that alter the K^+ concentration in endolymph. It should be noted that the mentioned conclusions in this paragraph were not obtained using ML.

Most of the referred studies use features such as age, gender, duration of noise exposure, smoking habits, working experience in years and hearing thresholds at multiple frequencies. These studies had unbalanced datasets, with only something between 1 third and 1 tenth of individuals in the datasets suffering from NIHL which is usually defined by patients having an hearing threshold above 25 dB. 5 of these studies have low sample sizes, equal to or under 210, while the remaining have sample sizes of 1113, 2110 and 10567.

[Zhao et al., 2018] with 1113 samples tried to predict noise induced hearing loss diagnosis and reported AUC scores of 0.81 for an SVM, 0.71 for a Multilayer Perceptron and 0.66 for both a RF and an AdaBoost classifier. [Zhao et al., 2019] with 2110 samples compares performance of an SVM to the ISO-1999 norm which describes a method of calculating the statistically expected Permanent Threshold Shift due to noise according to [Michel and Liedtke, 2021] and manages to obtain an F1 score of 0.715 compared to the 0.594 obtained using the norm described in ISO-1999.

[Greenwell et al., 2018], a study with 10567 samples, tried to predict a Significant Hearing Threshold Shift (STS) in hearing levels and if their hearing threshold is above 25 dB. To predict a STS, the study requires individuals to have a baseline audiogram obtained when ingressing in the air force and found both age and time since obtaining the baseline audiogram as the most important factors according to developed RF models, followed by gender and Air Force Specialty Tags that indicate the specific nature of their jobs. 1 fifth of individuals had STS while 1 tenth of individuals had a hearing threshold above 25 dB, and the RF models were cross validated obtaining accuracies of 0.801 and 0.793 respectively.

[ElahiShirvan et al., 2020] uses a SVM model for prediction of the the hearing threshold of 150 factory workers, outputting one of the following categories: "<25dB", "25-40dB", "41-60dB" or "61-80dB". The model obtains an accuracy of 94% however, this value may be misleading due to the highly unbalanced dataset (106 of the workers belong to the category "<25dB"). The model predicted 25.71% of the workers belonging to the category "25-40dB" to belong to the "<25dB" category, although all other predictions were correct. The study finally weighed the factors that can cause hearing loss by considered the SVM weights for each feature, concluding most important factors to be exposure to noise of frequencies of higher frequencies (8KHz and 4 KHz).

[Chen et al., 2021] detects sensorineural hearing loss by applying deep convolutional networks to magnetic resonance images achieving accuracy of 96% however, the small sample size of only 60 participants makes it difficult to make significant conclusions.

[Park et al., 2020] predicts if Unilateral Idiopathic Sudden Sensorineural hearing loss patients will recover based on their medical records with a total of 31 features. They developed five models, KNN, SVM, RF, AdaBoost and MLP, obtaining F1

scores of 0.64, 0.74, 0.74, 0.73 and 0.71 respectively. Best results were achieved using only 15 variables scaled to a space $[0; 1]$. From the RF models the study extracted feature importance, concluding most important features to be the hearing thresholds at the time of diagnostic, the blood urea nitrogen, the creatinine level, the age on onset, and the time from onset to treatment.

[Bing et al., 2018] uses demographics, medical records, medications, pure tone audiometry and laboratory tests results for a total of 149 variables to predict a Sensorineural Hearing Loss diagnosis. They trained a Deep Belief Network (DBN), a LR, a SVM and a MLP with subsets of 3, 11, 18, 21, 47 and 149 variables with best results obtaining F-scores of 0.79 with LR using 11 variables or SVM 3 variables, 0.71 with a MLP 1using 3 variables and 0.84 with a DBN using 149 variables.

The 11 variables subset included audiometric features, concurrent symptoms and comorbidity, the 18 and 21 variables subsets added demographic features, the 47 variables subset included medications, serum and urine indicators related to cardiovascular risk. The 3 variables subset consisted of the initial hearing threshold, initial audiogram configuration and time duration before study entry.

2.3.2 Evolutionary Computation in Medicine

Evolutionary Computation has been used in the medical field for multiple purposes. [Lourenço et al., 2019] and [Hidalgo et al., 2014] use Structured Grammatical Evolution and Grammatical Evolution algorithms in a symbolic regression problem to predict the glucose level in diabetic patients. [Bozcuk et al., 2004] uses a Genetic Algorithm to predict the prognosis of non-terminal cancer patients obtaining competitive results when compared to classic Machine Learning approaches on the same dataset. [Khalil et al., 2006] uses a GA for Vascular Soft tissue Elasticity Estimation, [Hoh et al., 2012] uses it to estimate the parameters on Friedman's labor curve, [Latkowski and Osowski, 2014] uses GA to select relevant genes for autism detection with an SVM classifier ensemble. [M.B et al., 1997] trains a Artificial Neural Network (ANN) with a GA to predict the prognosis of patients after non-small cell lung carcinoma surgery, achieving better results than a LR on the same dataset and [Lones et al., 2017] uses Genetic Programming to predict Dyskinesia levels in Parkinson patients. The interested reader can find a review of many more applications in [Ghaheri et al., 2015].

[Lones et al., 2017] uses a tri-axial accelerometer and gyroscope, 100Hz sampling rate, fitted to legs, arms torso head and trunk with adjustable bands associated with infrared video to allow 3 trained clinicians to label dyskinesia periods according to the Unified Dyskinesia Rating Scale (UDysRS), which is in a integer scale from 0 to 4 of intensity. Data was collect from 23 patients, 6 of which were evaluated for 6 hours and 17 of which were evaluated for 2 hours, for a total of 70 hours of data or around 25 million measurements for each of the axis of each gadget. A moving average of each axis in windows of 0.32s was calculated as well as several spectral characteristics. Individual classification models were then created for the temporal features and the spectral features using Implicit Context Representation Cartesian Genetic Programming. The best generated model has

been integrated into a monitoring tool for clinical use.

Predictions were easier the highest the level of dyskinesia, with the model obtaining AUC scores of 0.93 for the highest level and 0.56 for the lowest level. The study concludes more robust models to be built when not using classes 1 and 2 during training. It also compares performance between a sitting and a walking patient, obtaining a AUC of 0.92 for the former and of 0.73 for the latter. Spectral classifiers achieved best results on the walking patient, although achieving only an AUC of 0.58.

[Latkowski and Osowski, 2014] proposes a classifier system to diagnose autism. For this he had access to a database of gene expression of 54613 genes of 146 male individuals of which 82 have been diagnosed with autism and proposes an ensemble of 8 SVMs with the Gaussian Kernel where the output of each SVM is considered a feature for a RF classifier. The large number of genes makes feature selection a challenge since both determining good predictive genes and amount of genes to use is a difficult task. As such, the authors propose a two phased feature selection method where each of the SVMs will obtain a set of 100 features selected by one of the following methods:

- Fisher discriminant analysis
- ReliefF algorithm
- Two sample t-test
- Kolmogorov–Smirnov test
- Kruskal–Wallis test
- Stepwise regression method
- Feature correlation with a class
- SVM recursive feature elimination

and then proceeds to further narrow down the used features by usage of a GA with binary representation. Here fitness was defined as the error of 10-fold cross-validation of the SVM classifier. The whole process was repeated 10 times and obtained an ensemble accuracy of $86.07 \pm 2.79\%$, superior to the reference of 81.8%. The best individual SVM obtained an accuracy of only $79.84 \pm 5.66\%$. The reference obtained an accuracy of 81.8%, sensitivity of 91% and specificity of 61%, while this solution obtained an accuracy of 86.1%, sensitivity of 96.3% and specificity of 83%.

[M.B et al., 1997] predicts the prognosis of 620 patients with non-small cell lung carcinoma of which 372 (60%) were alive after 24 months of surgery using a Genetic Algorithm Neural Network (GANN). The study used variables such as UICC state, gender, TNM T score, TNM N score, Cell type, Differentiation Class, age and 4 different approximate measures of tumor volume. The leave-one-out method of cross-validation was employed, tests were performed with data collected at 6, 12, 18 and 24 months after surgery and results were compared to a LR

model. The GANN obtained accuracies of 0.94, 0.86, 0.87 and 0.82 respectively while LR obtained accuracies of 0.90, 0.78, 0.74 and 0.72. (Accuracy was reported only in a chart, and therefore mentioned accuracy results are approximations of the observed chart values.)

[Hoh et al., 2012] obtained data from 594 nulliparous women at 38 to 42 weeks of gestational labor, with spontaneous onset of labor, vertex presentation at admission, Cervical Dilation (CD) below 7 cm at admission and duration of labor from admission to delivery above 3 hours. Cases involving cesarean delivery, labor induction or epidural anesthesia were excluded. A CD measurement was taken at least once per hour. The study uses a real value representation GA, with population size 100, a maximum of 20,000 iterations, roulette wheel selection (fitness proportional), arithmetical crossover with weight 0.8 and probability 0.2 and mutation probability of 0.01. GA was used to estimate parameters for the Friedmans Labor curve and was compared against the Newton-Raphson (NR) method, obtaining on average a RMSE of 0.39 versus the 0.67 obtained by NR and average Mean Absolute Errors of 0.50 versus 0.92.

[Lourenço et al., 2019] compares performance of Structured Grammatical Evolution and standard Grammatical Evolution in the prediction of glucose levels, which should ideally be between 70 and 180 mg/dl, in diabetic patients based on past glucose levels, insulin injections, and amount of carbohydrate ingested with fitness being calculated as the RMSE. The models were trained and tested 30 times for each of the 10 different patients with SGE always significantly outperforming GE, obtaining on the best patient average RMSEs of 36.85 and 38.12 respectively. The study also does a Clarke Error Grid analysis which allows checking on the severity of wrong predictions, concluding less than 2 percent of predictions being in the most dangerous zone of the grid and on average only 10% of predictions presenting any risk.

2.4 Summary

Throughout this Chapter we have seen all the relevant background to the work that we will now develop. We have presented key Machine Learning concepts such as evaluation metrics and discussed common techniques. Then we focused on Tree Based Classifiers such as Decision Trees, Random Forests and Gradient Boost ensembles which will be crucial to the next stages of this work due to their interpretability.

We also saw key concepts on Evolutionary Computation, with particular interest in Structured Grammatical Evolution and Differential Evolution, explained the key advantages of this set of techniques and the drawbacks of Genetic Programming and Grammatical Evolution in particular. We then saw how we can use Evolutionary Computation to induct Decision Trees, which led to our decision of using SGE as a better alternative to GE that allows us to create such models without having to deal with invalids. We also saw DE which is known to be very effective in numerical optimization and will therefore also be of importance to this work.

Finally we did a review on Machine Learning works on the field of audiology which gave us an overview of what has been done, what is there still to be done and what kind of results we should expect at the end of this work. Since we found no works in audiology using Evolutionary Computation we also present a brief section on other works in the field of medicine that make use of it so the reader can rest assured that applying EC to audiology is not unrealistic and show the novelty behind this work.

Chapter 3

Approach

3.1 Hybrid Tree Evolutionary Algorithm

The goal of the proposed approach is to design Decision Trees (DT) to predict if a person is likely to have hearing problems. While aiming at maximizing the predictive power of the model, we also need to balance its complexity, keeping a simple structure making them easier to interpret.

For this we propose Hybrid Tree Evolutionary Algorithm (HyTEA), an Evolutionary Algorithm that relies on Structured Grammatical Evolution (SGE) [Lourenço et al., 2016, 2018] and Differential Evolution (DE) [Storn and Price, 1997]. The former is responsible for evolving the macro structure of each DT, such as deciding the number of nodes and which features, or combination of features, should be used at each node. Each model is then passed to the DE algorithm that parses the tree and optimizes the numeric parameters that will be used at each node to perform the splits. Figure 3.1 presents an overview of the proposed architecture.

In the first step, we prepare our dataset by performing several pre-processing operations such addressing the problem of missing values and performing feature normalization. Additionally, we split our dataset into 3 subsets: i) the Training set which will be used by the Differential Evolution component; ii) the Validation set which will be used by the SGE for fitness assignment; iii) the Test set which will be used to validate the quality and generalization ability of the best individuals found by our solution, in the end of the evolutionary search.

In the second step, we generate DTs using HyTEA. Firstly, SGE will search for the macro structure of each model, using a grammar that defines the necessary syntax restrictions using "if-then-else" constructs as shown in Figure 3.2. The symbol "%f" is a placeholder for a real number that will later be searched and optimized by the DE algorithm. Using this grammar we can create DTs where a node is a leaf when the terminal symbol "is_positive(%f)" is selected to replace the non-terminal symbol <node>. Otherwise, the node will correspond to a split. In the split, a decision is done based on a condition of the form "<expr> <= %f" where "%f" is the split value of the feature calculated in <expr>. <expr>can be replaced by a numeric constant, a feature from the original dataset (represent

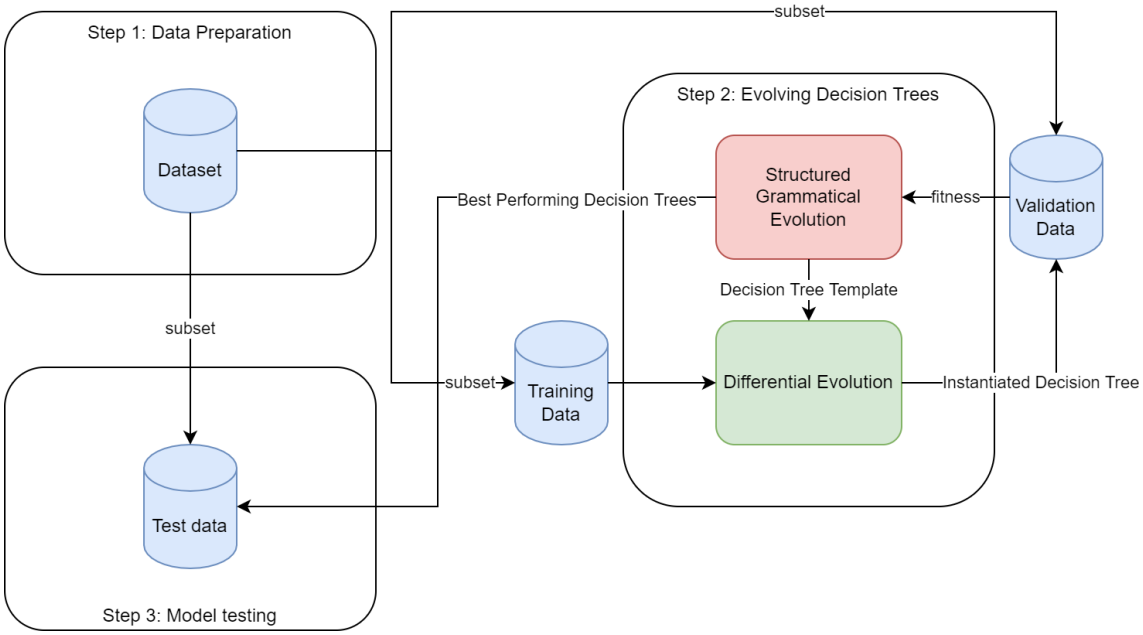


Figure 3.1: Overview of the proposed hybrid architecture.

```
<start> ::= <node>
<node> ::= is_positive(%f) | (<node>) if (<condition>) else (<node>)
<condition> ::= <expr><signal>%f
<signal> ::= <=
<expr> ::= <op>(<expr>,<expr>) | <var>
<op> ::= _add_ | _sub_ | _mul_ | protdiv
<var> ::= x[0] | x[1] | ... | x[59] | 1.0
```

Figure 3.2: Grammar used by SGE in HyTEA

by the x array) or a combination of features through the application of an addition, subtraction, multiplication or protected division. After having the macro structure of the DT, it is passed to the DE algorithm which will search for the numeric values of the "%f" placeholders that maximize the prediction accuracy of the model in the Training set. Lastly, the model is evaluated in the Validation set, and it is the quality obtained in this set that will be used as fitness in SGE.

Finally, in the third and final step the best performing models found are used in the Test set. This step is performed when the evolutionary run is finished, to assess the generalization ability of the best DT found. This step is paramount since it measures the extent to which the best DTs are robust and generalize for situations beyond the training data.

3.1.1 Fitness Assignment

Initially, the dataset is divided into three parts: 60% of the samples are used for Training, 20% are used for Validation, and the remainder 20% are used for Testing. As described in section 3.1, the training data is used by the DE algorithm to

optimize the parameters of the model. To reduce the training time, we randomly select a balanced subset of 1000 samples from the training set to be used by DE at each generation. This allows us to use all the available data for training during the evolutionary process, balancing the computational effort needed to train the model without compromising its predictive performance.

Once the individual has been optimized by the DE, we use it to classify the samples in the validation set. After all the samples are classified, we measure the accuracy of the model and use it as the fitness of the individual in the SGE algorithm.

During the parent selection stage if individuals have a validation accuracy difference lower than 2% we consider that they have the same fitness, i.e., we consider them to be tied. To resolve the ties, we take into account the individual's size measured as the number of internal nodes of the DT, i.e, individuals with less nodes are considered better. With this mechanism we introduce pressure towards parsimony, leading to simpler and easier to interpret models.

3.2 HyTEABoost - Evolutionary Gradient Boosting

Machine Learning algorithms often make classification mistakes, to be confident on their decisions we tend to strive to get the best possible performance out of our solutions. However, more complex models, which have the potential to be more accurate, tend to also be less interpretable.

Gradient Boosting is an ensemble solution where one major prediction is performed by an initial model, while subsequent models try to predict the previous ones' errors and compensate them. When used with Decision Trees, Gradient Boosting allows an initial Decision Tree to make the biggest impact on the final classification, keeping the interpretability, while subsequent Decision Trees compensate for the error. Subsequent Decision Trees can also be interpreted, however interpreting a error correction is not as easy as interpreting a classification, and when the number of trees in the ensemble grows interpretability gets progressively harder.

As an option to improve the accuracy of our model at a slight cost to interpretability, we developed an extension to HyTEA where we propose a mechanism similar to the traditional Gradient Boosting. The architecture of the solution is the same as presented in Figure 3.1, with main differences being found on the internals of SGE, the used grammar and the fitness function.

3.2.1 Updated Grammar

Gradient Boosting develops Decision Trees sequentially, where each DT tries to correct the output of the ensemble that preceded it. This iterative and gradual improvement requires that the output of the models is continuous, therefore we change our grammar to apply the sigmoid function to the output of tree leaves

```
<start> ::= (<node0>) if (<condition>) else (<node0>)
<node0> ::= sigmoid(%f) | (<node1>) if (<condition>) else (<node1>)
<node1> ::= sigmoid(%f) | (<node2>) if (<condition>) else (<node2>)
<node2> ::= sigmoid(%f)
<condition> ::= <expr><signal>%f
<signal> ::= <=
<expr> ::= <op>(<expr>,<expr>) | <var>
<op> ::= _add_ | _sub_ | _mul_ | protdiv
<var> ::= x[0] | x[1] | ... | x[59] | 1.0
```

Figure 3.3: Grammar used by SGE in HyTEABoost.

instead of using the is_positive function proposed in HyTEA, as can be seen in Figure 3.3. This function, defined according to Equation 3.1, transforms the parameter, a real value, into a probability, which then leads to a classification of class 0 (No Hearing Loss) if the probability is lower than 0.5 and class 1 (Has Hearing Loss) otherwise.

Furthermore, our initial experimentation with this technique showed great levels of tree bloat, we therefore also limited tree size in the grammar to 4 levels by replacing the <node> rule by 3 rules, <node0>, <node1> and <node2>.

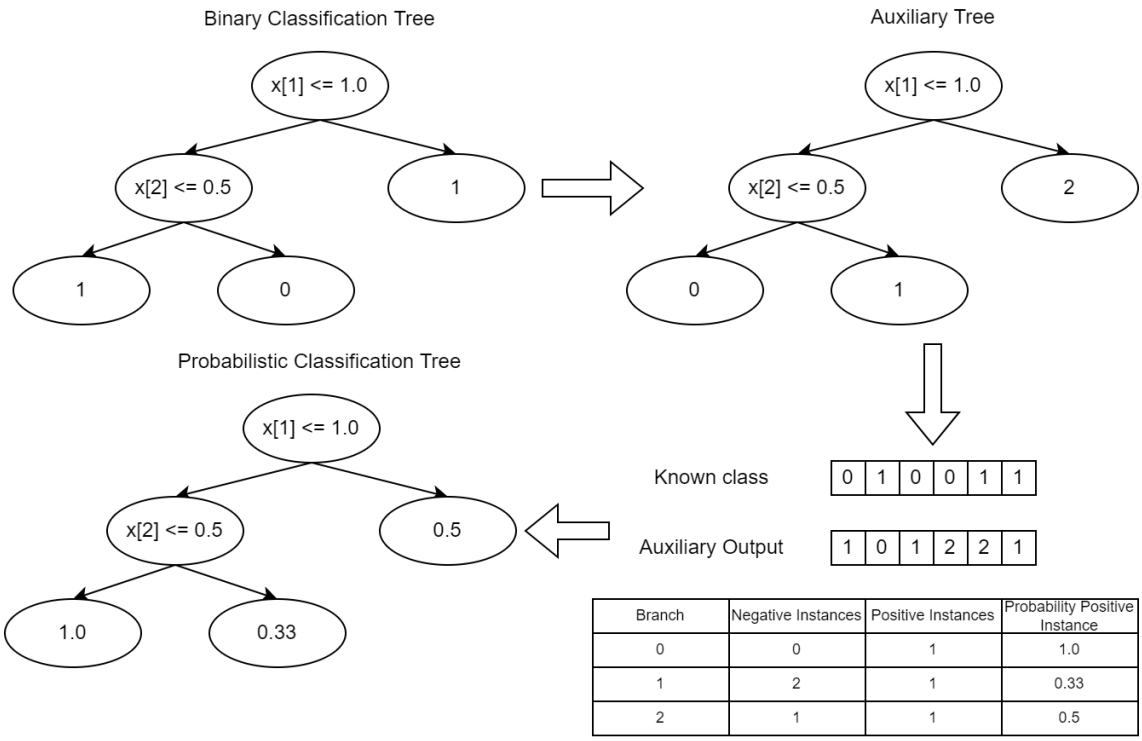
$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

3.2.2 Finding an Initial Tree

Gradient Boosting requires an initial Decision Tree that makes a prediction which will be improved by subsequent trees with the only requirement being that this Decision Tree outputs a probability instead of a simple binary value. Since we have already built Decision Trees with HyTEA we can improve the HyTEABoost execution time by providing a pre-built Decision Tree. The Decision Trees we built with HyTEA however, output a single binary value.

To make use of the trees built by HyTEA, at the beginning of the HyTEABoost run, we replace the predictions of the initial tree by a unique identifier, so that each terminal node gets a different id. Then we apply this Decision Tree to the joint Training and Validation sets. From there we can count how many samples were predicted by which branch, which we use to calculate the probability of a patient classified by a specific branch of the tree to have hearing loss. If a given branch did not classify any instances we consider it's probability to be 0.5 so that subsequently evolved trees can make the decision on which category it should fall. Now, given the probabilities of each branch, we once again replace the unique identifiers by the corresponding probabilities so that our initial tree outputs a probabilistic prediction instead of a binary decision.

This process allows us to reuse good pre-built Decision Trees, lowering the prob-



Branch	Negative Instances	Positive Instances	Probability Positive Instance
0	0	1	1.0
1	2	1	0.33
2	1	1	0.5

Figure 3.4: Example transformation of a Binary Classification Tree to a Probabilistic Classification Tree.

ability of evolving a bad Gradient Boosting ensemble while also speeding up the initial part of the evolutionary process since one less tree has to be evolved. An example of this transformation is shown in Figure 3.4 where a Decision Tree has 3 leave nodes, which are tagged with 0, 1 and 2, creating an auxiliary tree. Based on the known classes of 6 instances and the output of the auxiliary tree we determine the probability of each tag being associated with a positive class. We replace the tags with these probabilities therefore obtaining a probabilistic Decision Tree.

3.2.3 Evolving Subsequent Trees

Evolving the first Decision Tree of the HyTEABoost ensemble works exactly as on HyTEA. The subsequent trees however, should not predict the class of the instance, but rather the error of the preceding DT's prediction. Given this, the internals of SGE were updated as to receive two new parameters, the learning rate α, and the number of generations per tree N.

With these changes, every N generations the best tree in the population is chosen to join the ensemble. The residuals of the ensemble are calculated and become the new target output for the next N generations. The output of the ensemble is calculated by summing the learning rate α times the outputs of each tree chosen to join the ensemble with the exception of the initial tree whose full output is

taken, as described in Equation 3.2.

$$HyTEABoost(X) = InitialDT(X) + \sum_{DT \in HyTEABoost}^{DT \neq InitialDT} (\alpha \times DT(X)) \tag{3.2}$$

Evolving these subsequent trees requires a new fitness function. Since outputting any value lower than 0.5 leads to a classification of 0 and any value higher than 0.5 leads to a classification of 1, using metrics such as F1 or accuracy as fitness functions leads to there being no difference between any of these values and therefore gradual improving is not possible. With this in mind, our new fitness becomes the minimization of the sum of the absolute residuals of the classification, as given by Equation 3.3 where y_i and $\hat{y}_i$ are the correct class and the probabilistic prediction respectively, of a given instance i.

$$fitness = \sum_{i=0}^{length(y)} |y_i - \hat{y}_i| \tag{3.3}$$

This fitness function should lead to an evolutionary behavior where every small adjustment a tree makes to the ensemble's output is considered, therefore leading to gradual improving.

3.3 HyTEAForest

The last variant of HyTEA that we present, proposes to evolve complete Random Forest ensembles. We propose to evolve three different types of Random Forests according to the importance given to the outputs of each tree in the ensemble, with the type being chosen stochastically.

In majority voting, the output of each tree has the same weight and therefore the output of the ensemble is the class outputted by the majority of the trees in the ensemble. In weighted, each tree has a different weight in the decision, and in branch weighted each tree has a different weight depending on which branch of the tree made the decision. In both weighted versions the weights are parameters that will be determined by Differential Evolution.

HyTEAForest uses the concept of templates in the grammar to allow parameterizing the number of trees in the forest. The template used by the grammar can be seen in Figure 3.5 where <mdtrees>, <wdtrees>, <bwdtrees> and <ndtrees> will be replaced according to the number of trees chosen for the ensemble. In Figure 3.6 we can see a ready to use grammar for an ensemble with two trees.

Each of the referred types is encoded in the grammar according to the rules <majority>, <weighted> and <bweighted>. In majority voting, the output of the ensemble is the rounded mean of the outputs of each tree. Since the individual outputs of each tree are either 0 or 1, the mean of these values will be the probability that the ensemble output is 1 and therefore rounding it will obtain the ensemble output.

```
<start> ::= <majority>|<weighted>|<bweighted>
<majority> ::= round(sum([ind(x) for ind in <mdtrees>]) / <ndtrees>)
<weighted> ::= is_positive(sum([ind(x) for ind in <wdtrees>]))
<bweighted> ::= is_positive(sum([ind(x) for ind in <bwdtrees>]))
<node> ::= is_positive(%f) | (<node>) if (<condition>) else (<node>)
<bwnode> ::= (-%f*(-1)**is_positive(%f))|(<bwnode>) if (<condition>) else
(<bwnode>)
<condition> ::= <expr><signal>%f
<signal> ::= <=
<expr> ::= <op>(<expr>,<expr>) | <var>
<op> ::= _add_ | _sub_ | _mul_ | protdiv
<var> ::= x[0] | x[1] | ... | x[59] | 1.0
```

Figure 3.5: Grammar template used by SGE in HyTEAForest.

```
<start> ::= <majority>|<weighted>|<bweighted>
<majority> ::= round(sum([ind(x) for ind in [lambda x: <node>, lambda x:
<node>]]) / 2)
<weighted> ::= is_positive(sum([ind(x) for ind in [lambda x: -%f*(-1)**(<node>),
lambda x: -%f*(-1)**(<node>)]]))
<bweighted> ::= is_positive(sum([ind(x) for ind in [lambda x: <bwnode>,
lambda x: <bwnode>]]))
<node> ::= is_positive(%f) | (<node>) if (<condition>) else (<node>)
<bwnode> ::= (-%f*(-1)**is_positive(%f))|(<bwnode>) if (<condition>) else
(<bwnode>)
<condition> ::= <expr><signal>%f
<signal> ::= <=
<expr> ::= <op>(<expr>,<expr>) | <var>
<op> ::= _add_ | _sub_ | _mul_ | protdiv
<var> ::= x[0] | x[1] | ... | x[59] | 1.0
```

Figure 3.6: Example grammar used by SGE for HyTEAForest ensemble with 2 Decision Trees.

To obtain the output of the weighted solution a more complex mechanism was required. First we transform the output of a tree according to Equation 3.4, which leads to a value of -1 if the output was 0 and 1 if the output was 1. Then we multiply each of these values by a parameter which will be found by DE and obtain the sum of these values for each tree in the ensemble. Finally if this sum is positive we output 1, outputting 0 otherwise.

$$\hat{y}' = -(-1)^{\hat{y}} \tag{3.4}$$

Finally, in the branch weighted solution, each leaf node should have a different weight. We therefore use the same mechanism as in the weighted solution however it is applied directly to the leaf nodes instead of to the global tree output.

3.4 How did we get here?

The main goal when the work at hand started was to evolve Decision Trees for hearing loss prediction. Once this goal was attained we had 4 main issues to tackle: how to extend the algorithm for multiclass classification problems, how to improve the performance of generated models, how to speed up the evolutionary process and how to help health professionals interpret the output of these models. In this section we'll discuss the experimentation we carried on to tackle the first 3 issues, however these were preliminary experiments that prelude the solutions presented in the previous sections of this Chapter and we therefore do not have results of systematized experiments to discuss. The interpretability issue will be addressed in Chapter 5.

3.4.1 Multiclass Classification

Extending HyTEA to multiclass problems turned out to require only a few adjustments. Our grammar stopped using the "is_positive" function to use "get_class" since "is_positive" had a binary output. The new grammar can be found on Figure 3.7. This new function knows the values to which the output of Differential Evolution is bound and maps regular intervals of the possible values into the possible classes. For example, if the output of DE was bound to [-1.5; 1.5] and there are 3 possible classes, the function would map the intervals [-1.5; -0.5], [-0.5; 0.5] and [0.5; 1.5] to the classes 0, 1 and 2 respectively. The used implementation of DE however makes it unlikely for values to leave the defined bound, but not impossible, therefore instead of defining the intervals [-1.5; -0.5], [-0.5; 0.5] and [0.5; 1.5] we define the intervals [-∞; -0.5], [-0.5; 0.5] and [0.5; +∞] instead.

3.4.2 Improving Model Performance

To improve model performance we tried 3 different strategies: parameter optimization, parsimony pressure and ensembling.

```
<start> ::= <node>
<node> ::= get_class(%f) | (<node>) if (<condition>) else (<node>)
<condition> ::= <expr><signal>%f
<signal> ::= <=
<expr> ::= <op>(<expr>,<expr>) | <var>
<op> ::= _add_ | _sub_ | _mul_ | protdiv
<var> ::= x[0] | x[1] | ... | x[59] | 1.0
```

Figure 3.7: Grammar used by SGE for multiclass problems.

Parameter Optimization and Parsimony Pressure

The initial tries with parameter optimization were mostly aimed at trying to remove the early convergence detected with the first version of HyTEA and took place by increasing the values of mutation and decreasing the values of elitism. This experimentation lead to the early convergence to indeed disappear however, the apparent gain in performance was lost to overfitting as could be verified by the performance on the test set.

Since we were now dealing with overfitting we moved on to the second strategy of applying parsimony pressure accompanied by a soft limitation of the derivation tree size in SGE's parameters. An initial attempt was made by applying parametric parsimony pressure where the fitness function became that described by Equation 3.5 where γ is a parameter that defines the strength of the parsimony pressure and "n_nodes" is the number of internal nodes of the Decision Tree, therefore estimating it's complexity. Choosing the best value for γ proved to be difficult however, and we therefore changed our parsimony pressure strategy to the one previously presented in section 3.1.1.

$$fitness = accuracy - \gamma * n\_nodes \tag{3.5}$$

Ensembling

Finally we tested the viability of improving performance via ensembling. As a first attempt we created Random Forest models using the best trees generated by HyTEA. We then supplemented the created Random Forests with trees created by Sklearn, obtaining an ensemble with mixed evolutionary and greedily inducted Decision Trees. We tried 2 different ensembling methods, majority voting and weighted. In the weighted method the weight of each tree's decision was proportional to their accuracy in the training set. Here we thought that not all parts of an evolutionary Decision Tree are equally good due to it's stochastically built nature, and therefore the weight of a decision differing depending on which part of the tree made the decision could be beneficial for the ensemble. This lead to the 3 possible schemes for Evolutionary Random Forests which we imbued into our grammar in HyTEAForest as described in section 3.3.

Another attempt at improving model performance by ensembling was made by using Sklearn models such as Support Vector Machines, KNN, Multilayer Perceptrons and Logistic Regression to predict which instances would be incorrectly predicted by HyTEA Decision Trees, therefore correcting the classification. While this experiment lead to no viable results, it was what lead to the idea of evolving new models to improve the accuracy of the Decision Tree post hoc as described in section 3.2.

3.4.3 Speeding up Evolution

One of the main concerns with Evolutionary Computation is the time spent by the evolutionary process. With our approach this problem was evident as the evolutionary process could take days to converge when evolving simple Decision Trees, up to a month when evolving the Gradient Boosted ensemble or multiple months when evolving Random Forest ensembles. This problem is particularly accentuated when new data frequently arrives requiring retraining of models or when machines crash during the lengthy evolutionary process.

Our approach to speeding up evolution was therefore focused on the initialization of the population. If we have good pre-evolved models for old data, they can be used to fully or partially initialize the population therefore requiring less time for good models to be evolved for the new data. However diversity is a requirement for a well behaved evolutionary process so simply throwing in the old models to constitute the initial population of the evolutionary process is not the best solution as it will lead the algorithm to converge before any significant improvements on the initial population are observed.

We therefore created methods to populate the initial population from 3 sources. The first source was a pool of diversely selected Decision Trees from the pool of every Decision Tree ever generated by HyTEA. The second source was a pool of highly mutated Decision Trees based on the first source. And the final source was Decision Trees generated by Sklearn, extracted from either the DecisionTreeClassifier or the RandomForestClassifier models, which were translated into "if-then-else" statements (in the format used by HyTEA) and from which the genotype was algorithmically deduced.

Finally, during experimentation we often lost days or even weeks of progress due to the machines crashing either by lack of storage or power outages. To avoid further losses we created methods to recover the evolutionary process from the logs created at the end of each generation.

3.5 Summary

In this Chapter we described the foundations of HyTEA, how we create Decision Tree structures with Structured Grammatical Evolution and then populate the numeric fields of these structures using Differential Evolution. We also described the flow of our data, how we divide it into training, validation and testing sets

and use each to perform a different task, namely training data to find the best parameters for a given DT structure with DE, validation data to assign a fitness to each DT during SGE and test data to evaluate the performance of our models at the end of the algorithm's run.

Then we proposed HyTEABoost an extension to HyTEA that allows creating a gradient boosted ensemble from Decision Trees Generated by canonic HyTEA. As a final proposal we presented a variant to HyTEA that allows us to evolve both Random Forest ensembles.

Finally, we presented the main issues we had to solve after accomplishing the goal of evolving Decision Trees, described how we tried to tackle them and how that process eventually led to the tie breaking mechanism in HyTEA and the 2 evolutionary ensemble methods we propose.

Chapter 4

Experimental Study

4.1 Dataset

While the classification methods we developed where generic and could be used for any classification problem, the aim of this work has been to predict hearing loss. In this chapter we will discuss the data we have available, how we turned it into a dataset and how we subsequently validated our approach via an experimental study.

4.1.1 Database

In order to develop models capable of predicting a hearing loss Diagnosis we built a Database with records on 25,398 Portuguese patients, their counties of origin, birth dates, results and setup of their audiometry tests, their hearing aid usage and their responses to a Hearing Health questionnaire. 14,731 (58%) patients had a negative hearing loss diagnosis while 10,667 (42%) had a positive one. Patient data, audiometry tests and setup and questionnaires were provided by Evollu. Questionnaire answers were simplified to a range of 7 possible answers since questions were open. Demographics, companies and schooling data came from [PORDATA]. Health indicators were extracted from [Portal de Transparência do SNS].

Besides data on the patients we had data on demographic, health and economic indicators for each county in the country. Those were, the population, ageing index, number of companies, salary and turnover per type of economic activity as well as per economic sector, the numbers of diabetes diagnosis, the number of patients with Hemoglobin A1c lower than 8%, the number of hypertension diagnosis, the number of otorhinolaryngology acts and requests, the number of schools of each teaching level and the fatality rates for hemorrhagic and ischemic strokes.

Table 4.1 provides a summary of this data. Figure 4.1 shows the Entity Relationship Diagram for the Database. Table 4.2 shows all available patient data and

Patients	Total: 25,398 No Hearing Loss: 14,731 (58%) Hearing Loss: 10,667 (42%)
Patient data	County of Origin Audiometry Tests and Setup Hearing Aid Usage Hearing Health Questionnaire
Demographics (per county)	Population Aging Index
Economic Indicators (per county and per type of economic activity)	Number of Companies Salary Turnover Number of schools per teaching level
Health Indicators (per county)	Hemoglobin A1c lower than 8% Number of hypertension diagnosis Number of Otorhinolaryngology acts and requests Fatality rates for Hemorrhagic and Ischemic strokes

Table 4.1: Summary of available data during the development of the work at hand.

specifies format and/or possible values and Table 4.3 shows all metrics for these indicators and the periodicity with which measurements were taken.

4.1.2 Feature Engineering

Using all available information to perform a Machine Learning task required being able to associate all indicators to a single patient which was not immediately available by performing simple joins since the data is only correlated to the patient through his location and there are many measurements for the same indicators in the same location. Feature Engineering was therefore required to build a dataset to plug into a ML solution.

Patient, lead, audiometry and questionnaire data were straightforward to extract since there was only one lead and one answer to each question per patient. Since audiometry data would immediately lead to a correct prediction only the hearing loss diagnosis feature was considered which will be used later as a target prediction and was calculated as a boolean value set to true if either left or right hearing thresholds are equal to or larger than 40dB.

Hypertension, diabetes, stroke and otorhinolaryngology data was transformed into features by aggregating all measurements done over time per location by calculating the mean, standard deviation, minimum, maximum and quartiles for each of the metrics presented in the database, resulting in 7 additional features being created for variable. For otorhinolaryngology data, count was also used as

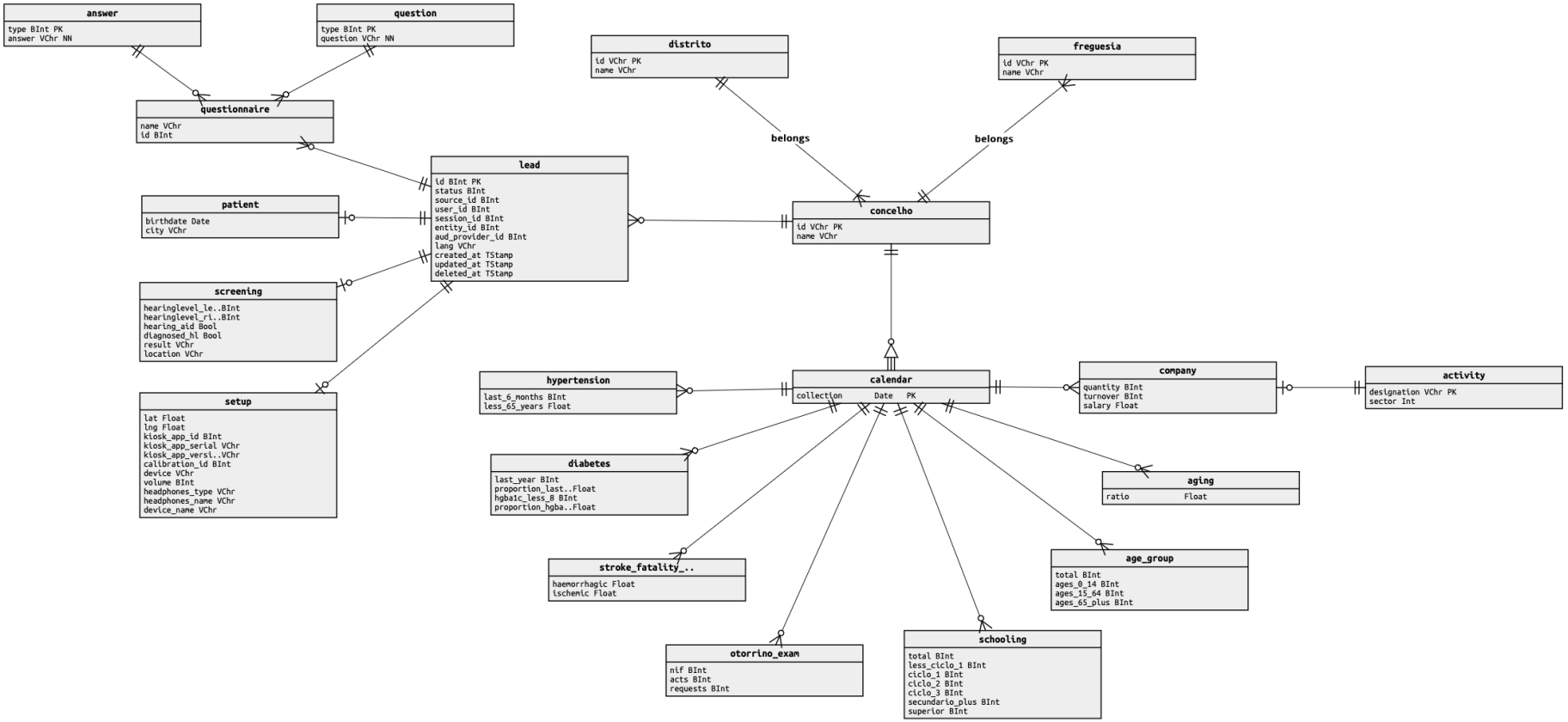


Figure 4.1: Entity Relationship Diagram for our Database. "Freguesia", "Concelho" and "Distrito" can be translated to "Parish", "County" and "District".

Table 4.2: Enumeration of available patient data followed by a specification of format and/or possible values.

Data	**Specification**
Birth Date	yyyy-MM-dd
City of Origin	String
Source of lead	kiosk OR platform OR web-questionnaire OR Promoter App - Android OR web-form OR Promoter App - IOS OR web-test OR App Evollu - IOS OR App Evollu - Android OR App Hearing Training - iOS OR App Hearing Training - Android OR Web Generic OR Facebook OR Google AdWords
Date of lead creation	yyyy-MM-dd hh:mm:ss.ffffff
Latitude & Longitude	Float value for each
Audiometry Setup	**Strings:** app serial, app version, app name, headphones type and headphones name **Numeric:** app id, calibration and volume
Audiometry Result	In the format below, where the first row specifies the tested frequencies and the question marks are replaced by integer values specifying the hearing threshold at each frequency. Second row show values for the right hear, and third row for the left hear. F 1000 2000 4000 6000 R ? ? ? ? L ? ? ? ?
Hearing level	Average of the hearing thresholds for each hear.
Hearing Aid Usage	true OR false
Question: Do you feel hearing difficulties?	Never OR No OR Sometimes OR Not sure OR In noisy places OR Yes OR Always
Question: Do you use an hearing aid?	Never OR No OR Sometimes OR Not sure OR In noisy places OR Yes OR Always
Question: Do you have behaviors that show hearing loss?	Never OR No OR Sometimes OR Not sure OR In noisy places OR Yes OR Always
Question: Do third parties denote your hearing loss?	Never OR No OR Sometimes OR Not sure OR In noisy places OR Yes OR Always

Table 4.3: Demographic, health and economic indicators for each county, the calculated metrics for this indicator and the frequency with which measurements were made.

Indicator	**Calculated Metrics**	**Frequency of Measurement**
Population	Total Population Population under 15 years old Population between 15 and 64 years old Population over 65 years old	1960 and 2011
Aging Index	Number of residents over 65 years old per 100 residents under 15 years old	2001 and 2019
Companies	Number of companies Turnover in thousands of Euro Average Salary in Euro (Measured per each kind of activity and economic sector)	1985, 2009, 2018 and 2019
Schooling	Number of schools at every educational stage from preschool to college	1985 and 2018
Diabetes	Number of patients that did a feet exam in the last year Percentage of patients with a feet exam in the last year Number of patients with Hemoglobin A1c level lower than 8% Percentage of patients with Hemoglobin A1c level lower than 8%	Monthly since January 2014
Hypertension	Number of patients with an hypertension diagnosis Percentage of patients with an hypertension diagnosis which are under the age of 65	Monthly since January 2014
Otorhinolaryngology	Number of requested exams Number of effectuated exams	Monthly since January 2020
Strokes	Percentage of hemorrhagic strokes that lead to death Percentage of ischemic strokes that lead to death	Monthly since January 2013

an aggregator since it is a metric of the number of otorhinolaryngology institutions in the area.

Since schooling, population, aging index and company data was only measured a maximum of 4 times, a single value was extracted per metric and per measurement. Company data was subdivided per type of activity and therefore a feature was extracted for each date of measurement and for each type of activity, resulting in 20 features per measurement per variable or 80 features per variable since there were 4 measurements.

Age was calculated from patient's birth date and question features were encoded with both One-Hot Encoding and Ordinal Encoding techniques. Finally missing values were solved by replacing missing values by the mean of the corresponding feature. In total 204 features were extracted or engineered, a few examples are shown below, and the full list can be consulted in appendix A.

- lat - Latitude
- lng - Longitude
- question_1 - Label Encoded answer to Question 1
- question_2 - Label Encoded answer to Question 1
- question_3 - Label Encoded answer to Question 1
- question_4 - Label Encoded answer to Question 1
- hypertension_last_6_months_mean
- hypertension_last_6_months_std

4.1.3 Feature Selection

Mean and standard deviation of each feature for each class were calculated and it was statistically verified if there was a significant difference in means between classes. This verification was done via an Independent t-test at a significance level $\alpha = 0.05$ with normality being assumed due to the Central Limit Theorem given the sample size bigger than 10,000. Results are shown in the appendix in Table A.1.

Besides this, 3 absolute correlation metrics were calculated that evaluate the correlation between a feature and the class: Pearson, Kendall and Spearman. The 50 best Pearson correlations varied between 0.302 and 0.046, the 50 best Kendall correlations varied between 0.339 and 0.112 and the 50 best Spearman correlations varied between 0.405 and 0.132. Features that always made top 50 included:

1. company_quantity_0_2009
2. company_quantity_0_2019

3. company_quantity_2_2009
4. company_quantity_2_2019
5. company_turnover_10_2009
6. company_turnover_11_2009
7. company_turnover_11_2019
8. company_turnover_2_2009
9. company_turnover_4_2019
10. question_1
11. question_1_1
12. question_1_6
13. question_1_7
14. question_2
15. question_2_1
16. question_2_2

Furthermore, ANOVA F-values between class and feature were calculated for each feature and features were sorted by this value, with higher values being higher on the list. After removing features with no significant difference in means, features were grouped according to their origin as *no_group*, *age_group*, *question*, *hypertension*, *diabetes*, *stroke*, *otorhinolaryngology*, *schooling*, *aging* and *company*.

Finally, a maximum correlation threshold between features of the same group of 80% was set, and features violating this condition were discarded sequentially, prioritizing low F-value features. Once this process was done, features belonging to the set of 16 features that always made top 50 in correlation scores were added back in case they had been removed. This resulted in a reduced set of 60 features instead of the original 204 features.

As a selection validation step, a default Sklearn Random Forest Classifier was trained with 30 splits of cross-validation with test size equal to 30% of the dataset. At each fold feature importance was extracted and the averages were obtained for the 10 most important features as shown in Table 4.4.

Since all the features in this top 10 made it into the reduced dataset, feature selection was considered successful.

Additionally we wanted to decide if generating new features as combinations of existing ones could be relevant. For this we built a total of $204^2 = 41616$ polynomial features of degree 2 and analyzed the correlation metrics for each of the new features. Examples of features that had an higher correlation to the target variable after combination are the multiplications of *question_2* by *age*, *question_2_2* by *age*,

Feature	Average Importance
age	0.291539
lng	0.180651
lat	0.162166
question_1	0.038634
question_3	0.019978
question_2	0.020277
question_2_1	0.018503
question_1_1	0.014186
question_2_2	0.014512
question_3_1	0.013429

Table 4.4: Average Importance of the 10 most important features according to a default Sklearn Random Forest Classifier after 30 splits of cross-validation.

question_1 by *age*, *question_1* by *age*, *question_1_1* by *question_2_1*, *question_3_1* by *question_2_1* and *lng* by *question_2_1*.

The best correlations obtained by combined features were of 0.385, 0.412 and 0.491 versus the correlations of 0.302, 0.339 and 0.405 obtained by simple features in the Pearson, Kendall and Spearman correlation metrics respectively. Since many of these combined features showed to have higher correlation with the target variable than the isolated features it was decided that further feature engineering could be beneficial.

4.2 Experimentation with traditional models

While the focus of this work is to obtain evolutionary classification models, it is important to also experiment with traditional models so that we have a comparison baseline, can identify shortcomings of these models and can analyze how our approach solves these shortcomings. With that in mind we started by studying the performance of Sklearn models in our dataset.

4.2.1 Experimental Setup

Multiple Sklearn models were tried with the features in the reduced dataset as a means to obtain a performance reference. Since we propose to develop Decision Trees, Gradient Boosting and Random Forest models, these are the same models we use here for comparison. Default Sklearn Decision Trees tended to overfit data, we therefore decided to tune the maximum depth of the tree. Since most our HyTEA generated Decision Trees ended up having a depth of 3, we created Sklearn trees using a maximum depth equal to 3, but also tried the values of 5 and 10 so that different behaviors could emerge.

Mean and standard deviation of accuracy, F1, precision and recall were calculated for each classifier after 30 splits of cross-validation with test size equal to 30% of

	Accuracy		F1		Precision		Recall	
Model	μ	σ	μ	σ	μ	σ	μ	σ
DecisionTree(max_depth=3)	0.711	0.070	0.557	0.131	**0.826**	0.149	0.436	0.134
DecisionTree(max_depth=5)	0.721	0.080	0.654	0.104	0.707	0.102	0.621	0.135
DecisionTree(max_depth=10)	**0.733**	0.058	**0.669**	0.089	0.723	0.080	**0.638**	0.135
RandomForest(max_depth=3)	0.721	0.072	0.612	0.124	**0.760**	0.123	0.528	0.151
RandomForest(max_depth=5)	0.737	0.072	0.659	0.106	0.742	0.096	0.603	0.136
RandomForest(max_depth=10)	**0.749**	0.081	**0.681**	0.119	0.751	0.111	**0.636**	0.150
GradientBoosting(max_depth=3)	**0.770**	0.069	**0.718**	0.089	**0.773**	0.109	**0.679**	0.107
GradientBoosting(max_depth=5)	0.751	0.073	0.686	0.107	0.755	0.107	0.644	0.132
GradientBoosting(max_depth=10)	0.737	0.070	0.678	0.086	0.727	0.100	0.644	0.102

Table 4.5: Mean and Standard deviation of accuracy, F1, precision and recall for each Sklearn model after 30 splits of cross-validation with test size equal to 30% of the dataset using Standard Scaling.

	Accuracy	F1	Precision	Recall
DecisionTree(max_depth=3)	[0.685; 0.738]	[0.508; 0.607]	**[0.769; 0.883]**	[0.385; 0.487]
DecisionTree(max_depth=5)	[0.691; 0.752]	[0.615; 0.694]	[0.668; 0.745]	[0.569; 0.672]
DecisionTree(max_depth=10)	**[0.711; 0.756]**	**[0.635; 0.703]**	[0.693; 0.753]	**[0.587; 0.690]**
RandomForest(max_depth=3)	[0.694; 0.748]	[0.565; 0.659]	**[0.713; 0.806]**	[0.471; 0.586]
RandomForest(max_depth=5)	[0.709; 0.764]	[0.619; 0.699]	[0.706; 0.778]	[0.551; 0.654]
RandomForest(max_depth=10)	**[0.718; 0.780]**	**[0.635; 0.726]**	[0.709; 0.794]	**[0.579; 0.693]**
GradientBoosting(max_depth=3)	**[0.744; 0.796]**	**[0.684; 0.752]**	**[0.731; 0.814]**	**[0.639; 0.720]**
GradientBoosting(max_depth=5)	[0.723; 0.779]	[0.646; 0.727]	[0.714; 0.796]	[0.593; 0.694]
GradientBoosting(max_depth=10)	[0.710; 0.763]	[0.646; 0.711]	[0.689; 0.765]	[0.605; 0.682]

Table 4.6: 95% confidence interval of accuracy, F1, precision and recall for each Sklearn model after 30 splits of cross-validation with test size equal to 30% of the dataset using Standard Scaling.

the dataset. The process was repeated for 3 different scaling techniques: Min-Max Scaling, Maximum Absolute Scaling and Standardization.

4.2.2 Results

As mentioned in section 4.2.1 we tried 3 different scaling techniques. Since the Standardization strategy undoubtedly led to the best results across all models, in this section we present the results for this strategy and on the next sections where experimentation is slower and we could not try all scaling strategies we proceed using this strategy as well.

The mean and standard deviation of the accuracy, F1, precision and recall metrics can be found in Table 4.5 and the 95% confidence intervals for the same metrics can be found in Table 4.6.

As can be observed, Gradient Boosting obtained the best performances overall across all metrics. Decision Tree and Random Forest models ended up having similar performances accuracy wise, however Random Forest tended to obtain better F1, precision and recall. The simplest Decision Trees, with a maximum depth of 3, ended up having a recall always inferior to 0.5, meaning more than half of the hearing loss patients were not detected.

Parameter	SGE	DE
Population	200	15
Generations	100	20
Parent Selection	Tournament with size 3	N/A
Elitism	10%	N/A
Crossover Rate	0.9	0.7
Mutation Rate	0.1	Between 0.01 and 0.2
Minimum Tree Depth	3	N/A
Maximum Tree Depth	10	N/A

Table 4.7: Parameters used in the experimental study for each method.

4.3 Experimentation with Canonic HyTEA

Now that we have a baseline study using traditional models we can start experimenting with evolutionary models and be sure of how well behaved these models are.

4.3.1 Experimental Setup

To obtain statistically reliable results we repeated each experiment 30 times using different seeds. HyTEA is an hybrid of 2 evolutionary algorithms, Structured Grammatical Evolution and Differential Evolution. To define the experimental setup for HyTEA we must then specify the parameters for both of these algorithms as explained in the following paragraphs.

The parameters used to configure each algorithm are summarized in Table 4.7. The settings used by the SGE were defined following the recommendations proposed in [Lourenço et al., 2017, 2018]: {Number of Runs: 30; Population Size: 200; Generations: 100; Crossover Rate: 0.9; Mutation Rate: 0.1; Elitism: 10%; Tournament Selection with size 3; Minimum Tree Depth:3, Maximum Tree Depth: 10}. It should be noted that Maximum Tree Depth is the depth of the derivation tree, not the DT, and that once this depth is reached only terminal derivations are prioritized.

For the DE algorithm we use 15 individuals, and allow the algorithm to run for 20 generations. The mutation rate is variable between 0.01 and 0.2, and we use the `best/1/bin` DE strategy.

Besides these standard parameters, other parameters can be tuned regarding how data is split into Training, Validation and Test sets, how we subset the training data used at each generation and the fitness difference threshold at which we consider competing individuals to be tied. While we considered these parameters to be fixed during conceptualization of HyTEA and therefore pointed them out while explaining our approach in section 3.1.1, these values can indeed changed so we point them out here again for the sake of completeness.

We split the data into 3 sets, 60% of the data represents the training set, 20% of

	Fitness	Accuracy	F1	Precision	Recall	Int. Nodes	Depth Est.
μ	0.722	0.719	0.721	0.716	0.727	5.667	3.262
σ	0.005	0.007	0.012	0.014	0.030	5.384	0.702
min	0.714	0.708	0.690	0.698	0.638	2.000	2.000
25%	0.717	0.714	0.714	0.704	0.711	4.000	3.000
50%	0.721	0.716	0.720	0.716	0.732	5.000	3.322
75%	0.726	0.722	0.731	0.723	0.743	5.750	3.519
max	0.735	0.737	0.741	0.752	0.779	33.000	6.044
95% CI	[0.720; 0.724]	[0.716; 0.722]	[0.717; 0.725]	[0.711; 0.721]	[0.716; 0.738]	[3.66; 7.68]	[3.00; 3.52]

Table 4.8: 95% confidence interval of fitness, accuracy, F1, precision and recall for the 30 runs of the canonical HyTEA experiment.

the data is used for validation the remaining 20% are used for testing. At every generation 1000 samples of the training set (approximately 6.67% of the training set) are selected for usage with DE while the rest of the training set is ignored for that generation. Finally if the fitness of competing individuals, which is measured as accuracy in the validation set, differs in less than 2%, they are considered to be tied and the winning individual is that with smaller size.

4.3.2 Results

As a result of these experiments we collected data on the performance metrics for the best individuals, measured in validation set fitness, at each run. The validation set fitness, testing accuracy, F1, precision and recall, as well as the number of internal nodes and estimates of tree depth calculated as log_2(InternalNodes) + 1 are summarized in Table 4.8.

We can see that all performance metrics are very consistent across experiments, leading to very small Confidence Intervals, and are around the 72% mark. The evolved Decision Trees are also very small, with average depth being in the interval [3.00; 3.52] with 95% confidence.

When comparing to traditional models we can see that accuracy on average is very close, with the independent t-test indicating there to be no statistical difference. F1 tends to be much higher in HyTEA while Precision is somewhat lower when compared to the simplest Sklearn Decision Tree. As the reader may recall when analyzing traditional Decision Trees we had a concern with a low recall metric which meant we were failing to find positive hearing loss diagnosis. Here this recall metric is much improved, with traditional models obtaining values only as high as 0.690 with 95% confidence when using trees with maximum depth of 10, while HyTEA obtains recall values in the range [0.716; 0.738] with 95% confidence using Decision Trees of an average depth of only 3.262.

4.4 Experimentation with HyTEABoost

4.4.1 Experimental Setup

To obtain statistically reliable results with repeated each experiment 12 times using different seeds. The goal was to repeat the experiment 30 times, however technical difficulties, the naturally longer processing time and the shortness of time due to this particular setup being experimented late in the project, we only managed to achieve 12 experiments by the time of writing. Never the less, the results were consistent across experiments, which speaks for their reliability.

Just like HyTEA, HyTEABoost is an hybrid of 2 evolutionary algorithms, Structured Grammatical Evolution and Differential Evolution, with selected parameters for both algorithms being the same as those described in section 4.3.1 with exception to the number of generations which was increased to 200. However, two additional parameters were required for this solution. The first one is the number of generations each individual is allowed to run for before trying to evolve the next one. We set this parameter to 20 generations, since there is a total of 200 generations, 10 individuals will be evolved as the final result of each experiment, iterating on top of the best Decision Tree we generated during experimentation with HyTEA. The second parameter is the learning rate which modifies the impact each subsequent tree will have on the final prediction, and we set this parameter to 0.1, following the default of Sklearn.

4.4.2 Results

To analyze the benefit of improving the initial Decision Tree with the HyTEABoost method, we tracked the performance metrics for the initial Decision Tree and the best performing corrective tree at each generation for each experiment. Since the testing set is not the same across all experiments, the average of the performance across experiments is considered.

The average accuracy of the individual initial Decision Tree was of 0.716 over the 12 experiments, while the average F1 was 0.696. The average achieved accuracy was of 0.727 while the average achieved F1 was 0.725, making it an improvement of 1.1% in accuracy and 2.9% in F1.

Meanwhile, precision was negatively impacted since we got a decrease of 0.02 from an initial value of 0.749 to 0.729. The Recall metric however, had the biggest benefit from using additional trees to correct the initial prediction, starting at the value of 0.650 and ending up with a value of 0.722, making it a 7.2% increase.

The evolution of these metrics over the generations can be visualized in Figure 4.2. It can also be seen that there is a peak at each of the metrics every 20 generations. This peak is due to the reset in the population which starts working on a new individual. After 120 generations, that is, 6 added individuals to the ensemble, the performance gain seems to stagnate.

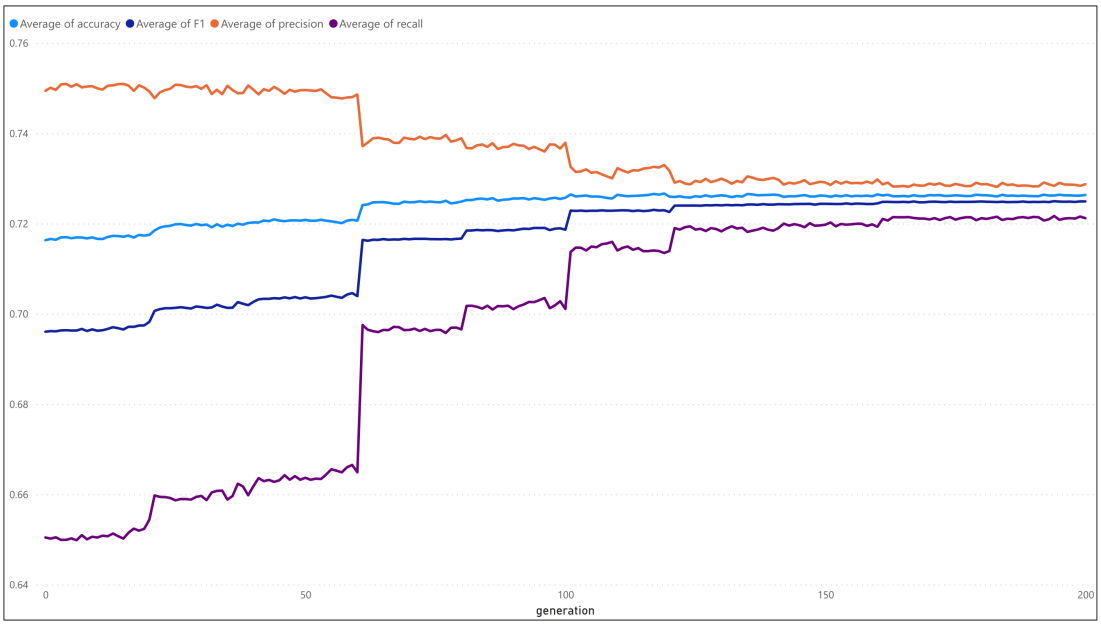


Figure 4.2: Progress of average performance metrics over the generations for the best individuals at each experiment with HyTEABoost.

When compared to the traditional gradient boosting with tree depth limited to 3, our solution got worse accuracy with 0.727 against 0.770 and worse precision, with 0.729 against 0.773. However it obtained better F1 with 0.725 against 0.718 and better recall, with 0.722 against 0.679. It seems each solution could be optimal depending on which metrics matters most for each use case. While at first glance, the results are not impressive, we did find significant improvements on the recall metric which is critical for the work at hand since it shows the percentage of patients with hearing loss we end up finding.

4.5 Experimentation with HyTEAForest

4.5.1 Experimental Setup

Following the same guidelines as the previously detailed experiments, we repeated each experiment 12 times using different seeds, not managing to achieve the 30 runs due to the heavy processing load and lack of time.

The general parameters for both SGE and DE were the same as those described in section 4.3.1, however this technique turned out to require a lot more computational power, and due to the lack of time we had to tune down some parameters so that we could have timely results. Specifically we decreased population size and the maximum number of generations to 100 and 50 down from 200 and 100 respectively.

Besides this, we had to select the number of Decision Trees to be considered on the ensemble and again, due to lack of time and processing power, we limited the

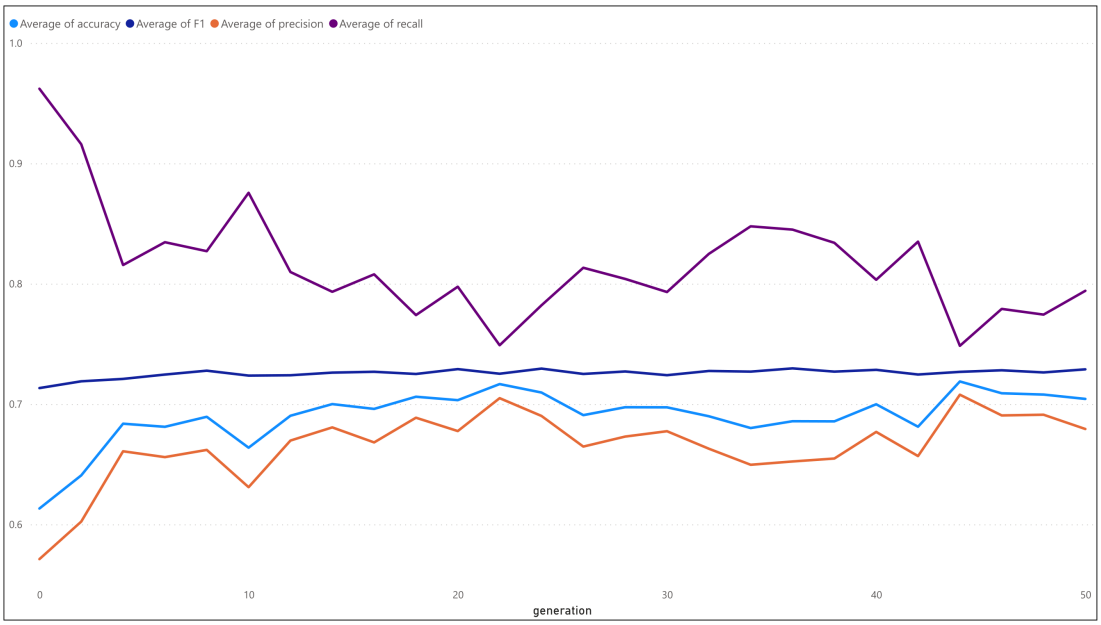


Figure 4.3: Progress of average performance metrics over the generations for the best individuals at each experiment with HyTEAForest.

number of trees to just enough to validate the concept, which is 2.

4.5.2 Results

For this solution, we track the results of the best performing individuals at each experiment and generation. Since the testing set is not the same across all experiments, the average of the performance across experiments is considered.

By the end of the experiment, the best evolved individuals had an accuracy of 0.704, F1 of 0.729, precision of 0.679 and recall of 0.794.

The evolution of these metrics over the generations can be visualized in Figure 4.3. We can see in the evolution that it is converges very early in the first 5 generations and then stops improving. This is probably due to a lack of diversity resulting from the low number of individuals used during this experiment.

When compared to the traditional Random Forest our solution had worse accuracy, with 0.704 against 0.749 and worse precision with 0.679 against 0.760. However, our solution did have better F1 with 0.729 against 0.681 and better recall with 0.794 against 0.638. Given very few resources were allocated to run this experiment, it has shown great potential as an alternative both when compared to traditional models and our other evolutionary solutions. As we have previously discussed, recall is an essential metric in this project as it ensures we're not missing positive hearing loss diagnosis and this model provided the best recall of all our alternatives although at a loss for accuracy and precision.

4.6 Discussion

In our scenario failing to diagnose a patient with hearing loss is more harmful than calling a patient for a screening which ends up not being diagnosed with hearing loss, therefore a low recall metric is a big concern and would make it unviable to use a model. The simplest Decision Tree we evolved with Sklearn, with a maximum depth of 3, had a recall value in the range [0.385; 0.487] with 95% confidence. More complex Sklearn Decision Trees (with maximum depths of 5 and 10) offered better recall values however their complexity makes interpretability harder. This early observation using traditional Decision Tree induction algorithms shows a gap which HyTEA as the potential to fill by evolving Decision Trees that have an increased performance, not only overall but specifically for the critical recall metric, while keeping the simplicity of a low depth tree.

When analyzing the results obtained by canonical HyTEA we find very consistent performance metrics, showcasing the robustness of the algorithm. We can find accuracy values comparable to those obtained with traditional models although much more consistent and we find significant improvements in both F1 and recall without sacrificing interpretability since obtained tree depths were on average lower than 4, making for very simple Decision Trees.

HyTEABoost was used to improve on the Decision Trees generated by canonical HyTEA. While the improvements in accuracy were not significant and precision did decrease, the F1, and most importantly, the recall metrics were significantly improved. Has we have already mentioned, recall is the most important performance metric in this particular use case and therefore the trade-off of interpretability for better recall offered by this approach is well worth considering.

HyTEAForest was capable of outperforming traditional models in the F1 and recall metrics, although losing in accuracy and precision. The lack of time to allocate larger resources to the algorithm could be to blame for the lack of performance in other areas and will require further experimentation to assess the full potential of this approach. It did however already prove to be a viable alternative especially if we want to favor a high recall model which ensures we avoid the most false negatives possible.

4.7 Summary

In this Chapter we described the data we had available for a total 25,398 patients of which 10,667 had a positive diagnosis for hearing loss. We showed to have data on the patient's county of origin and answers to a hearing health questionnaire as well as a series of demographic, economic and health indicators from each patient's county of origin.

Still on the subject of data, we explained how we transformed our data to form a coherent dataset, how we dealt with categorical data and with missing values and how we selected which features to keep on the dataset. Finally, we did some

experimentation with polynomial feature building and measured the discriminative power of these features in terms of correlation to the target class, concluding that combining features can be relevant to improve model performance and therefore this capability should be imbued in the grammar of HyTEA.

Once we discussed our problem's dataset we moved on to experimentation. We first obtained a performance baseline with traditional Sklearn models and identified issues with these obtained models. Then we performed experimentation with canonical HyTEA and found it to be more consistent than traditional methods, leading to similar or better performance depending on which performance metrics we are analyzing while keeping Decision Trees compact therefore conserving interpretability.

Then we saw that HyTEABoost manages to significantly increase performance on the recall and F1 metrics at a slight cost for precision and interpretability and finally saw that HyTEAForest is a viable alternative that favors the recall metric even more than HyTEABoost and given more resources has the potential to outperform all other models.

Chapter 5

Visualization

Visualization has a key role in this work for multiple reasons. Firstly we, as Machine Learning developers, must know if our classifiers are modeling the problem space correctly. Secondly, as we have often emphasized, interpretability is key in our Machine Learning solutions since health professionals must understand decisions. And lastly, these same health professionals must know how reliable decisions are.

With this in mind we first generated and analyzed t-Distributed Stochastic Neighbor Embedding visualizations to validate the modeling of the problem space and then we created a visualization tool for health professionals to visualize Decision Trees, perform classification and check the reliability of the generated decision.

5.1 Analyzing the Best Models using t-SNE

In order to verify if our models were correctly modeling the problem space we created t-SNE visualizations for every instance in the dataset, drawing them with different colors depending on their class.

In Figure 5.1 we can view a t-SNE visualization where the colors represent the known class. In this visualization there are many visible clusters, most of which have a mix of both classes, representing sets of instances which should be hard to accurately predict. There is however a region that noticeably represents mostly patients with no hearing loss, which starts at the central cluster and extends towards the upper right diagonal of the plot.

A well behaved classification model should follow similar patterns to those identifiable in this visualization of the original data. When changing the colors of the visualization to represent the output of our best HyTEA generated Decision Tree we obtain Figure 5.2. When comparing this visualization to that of the known classes we can see that our Decision Tree captures very well the pattern of the central cluster extending towards the upper right diagonal. However, there are many clusters that, while having only a few negative instances in the real data, show here as approximately half of the instances being negative. This means that

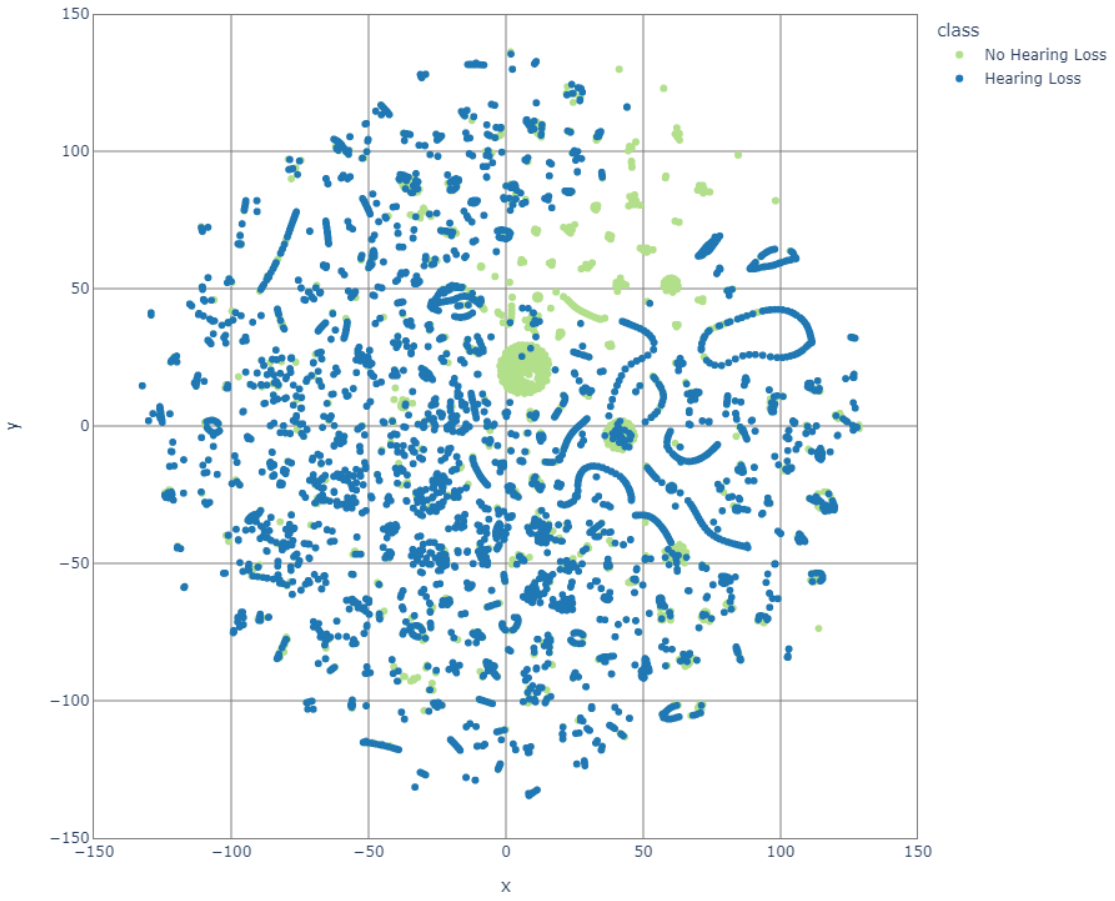


Figure 5.1: t-SNE visualization of the dataset with the colors representing the known class for each instance.

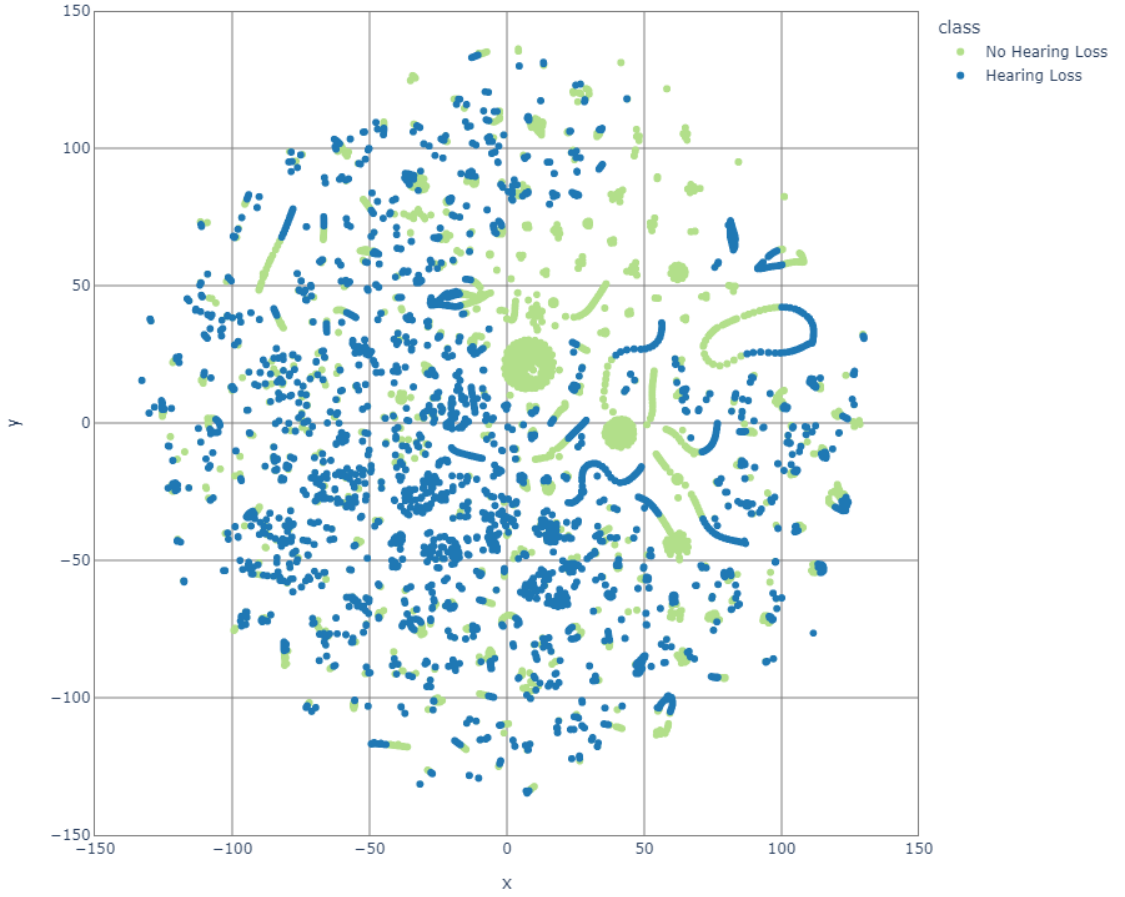


Figure 5.2: t-SNE visualization of the dataset with the colors representing the outputted class by our best HyTEA generated Decision Tree for each instance.

there likely are often positive instances being classified as negative instances.

This visualization however, especially in the original data, does not allow viewing overlaid instances which may make us believe that a cluster with a balanced amount of both positive and negative instances is made of mostly positive instances simply because the blue dots were drawn over the green. To allow a more reliable analysis we created a new visualization where green and blue dots represent the known negative or positive classes on instances that were correctly classified while a third red cross marker is used to indicate instances that were not correctly classified by our Decision Tree.

This visualization can be seen in Figure 5.3 and it shows that clusters that apparently were mostly made up of positive instances actually were a mix of both classes and that the algorithm while incapable of fully separating instances in these mixed clusters, was capable of dividing them, usually in 2 different sections, where the concentration of instances of each class tends to one class or the other.

When comparing the visualization on Figure 5.2 with the similar visualizations

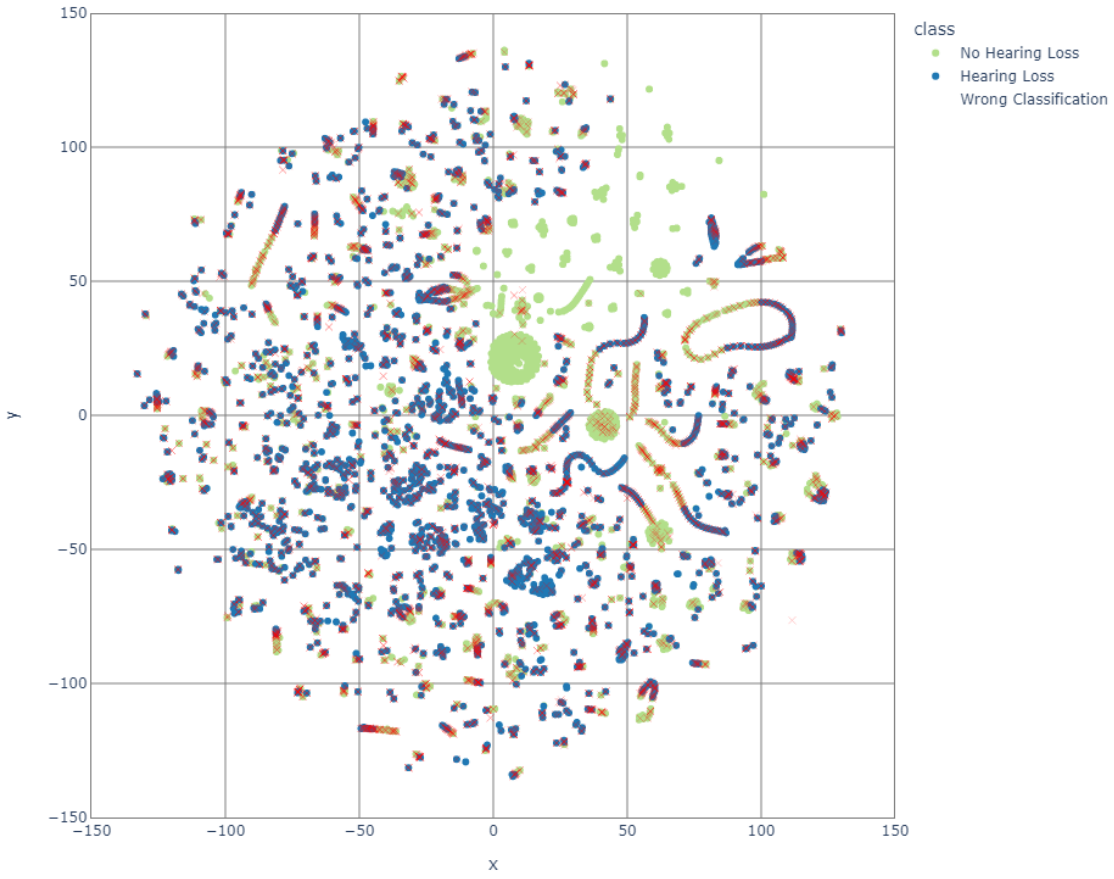


Figure 5.3: t-SNE visualization of the dataset with red markers showing the instances incorrectly classified by our best HyTEA generated Decision Tree.

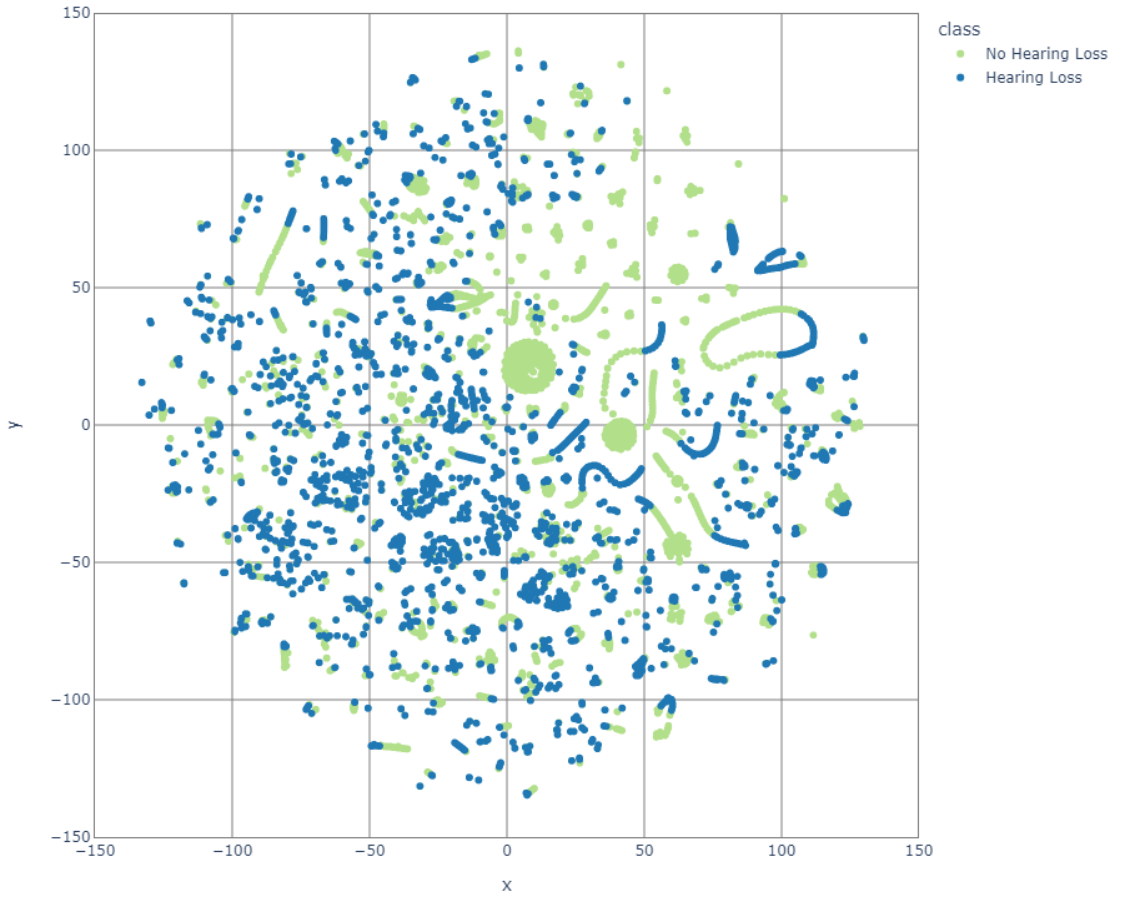


Figure 5.4: t-SNE visualization of the dataset with the colors representing the outputted class by a Sklearn Decision Tree for each instance.

for Sklearn Decision Trees and Random Forests, as observed in Figures 5.4 and 5.5 respectively, we see that the modeling behavior is very similar, with the differences being mostly on the thresholds at which the mixed clusters are split. It should be noted that for a fair comparison and since most of our Decision Trees ended up having depth equal to 3, we set the Sklearn trees to have a maximum depth of 3 as well.

To further analyze the behavior of our Decision Tree when modeling the problem space, we decided to create a visualization where we can see which branches of the tree are responsible for modeling each cluster. To do this we first name and color each of the terminal nodes as seen in Figure 5.6 and then use these names and colors to fill the t-SNE visualization as seen in Figure 5.7 while also calculating the accuracy of each branch in particular. Terminal nodes that did not classify any instance on the dataset are colored black in the Decision Tree.

In this new visualization of Figure 5.7 we can see that the cluster we identified earlier, starting in the center and extending towards the upper right, is classified by one single branch of the Decision Tree which also is the most accurate one with 95.5% accuracy. Meanwhile the blue, yellow and green branches divide the

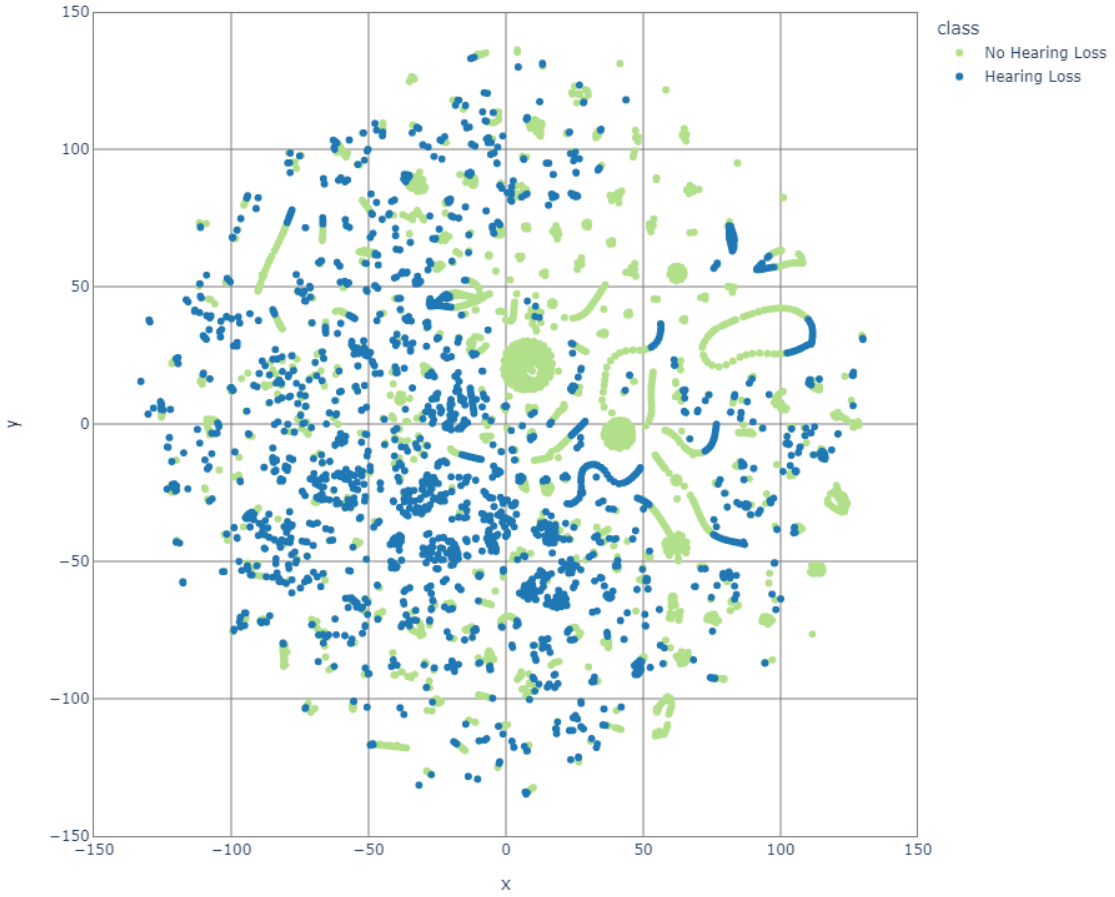


Figure 5.5: t-SNE visualization of the dataset with the colors representing the outputted class by a Sklearn Random Forest for each instance.

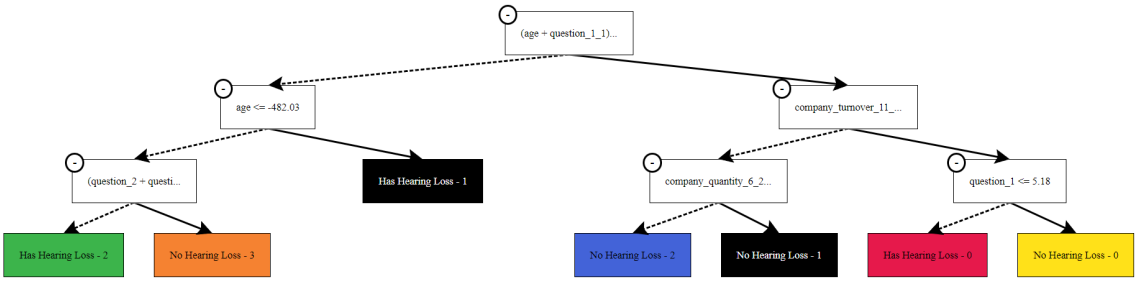


Figure 5.6: Best HyTEA generated Decision Tree with terminal nodes colored and tagged to be distinguished in the t-SNE visualization.

clusters into regions depending on the concentrations of points in each of those regions obtaining fairly accurate predictions with an accuracy between 68.7% and 77.1%. Finally, the red branch classifies regions of very high uncertainty, obtaining only 51.1% accuracy.

This visualization can help a health professional to assess the reliability of a prediction made by the Decision Tree. If the Decision Tree predicted the patient to not have hearing loss with the orange branch, there is a 95.5% chance that it is correct, while if the prediction was made with the red branch they could know that they should rely on other methods to make the decision since the prediction by the Decision Tree is a coin flip.

5.2 Visualization Tool

As previously mentioned, we developed a tool for health professionals to obtain predictions and analyze the reliability of these predictions. In short the tool allows selecting and visualizing from a set of pre-evolved Decision Trees, obtaining predictions for patients with specific characteristics, viewing which branch of the tree made the prediction and which points are modeled by that branch in a t-SNE visualization. These features are further detailed in section 5.2.2.

An overview of the developed tool can be seen in Figure 5.8.

5.2.1 Architecture

The tool is composed of 2 components. A REST API backend built with python and flask which provides data on Decision Trees and performs classification, and a Vue.js web app as a frontend.

Backend

The REST API has access to a SQLite database which contains every Decision Tree evolved with HyTEA and their metadata, such as genotype, size, used features and performance metrics. When initialized the API gathers the 50 best Decision

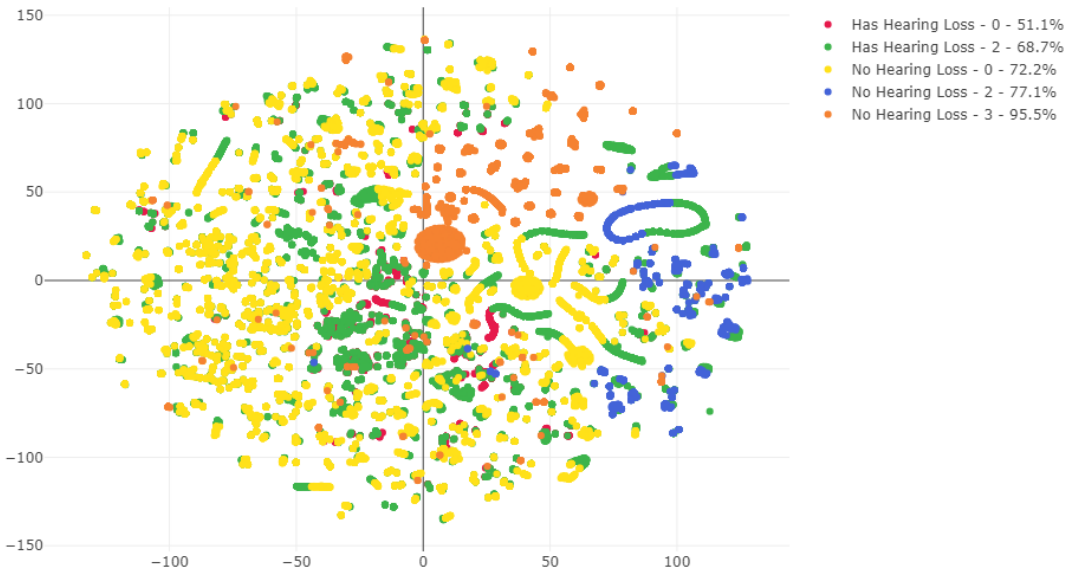


Figure 5.7: t-SNE visualization of the dataset with the colors representing the terminal nodes of our best generated Decision Tree which made the prediction for each instance of the dataset.

Trees in the database, disregarding any duplicates (same structure, possibly different parameters). Then it calculates a t-SNE projection for every known point in our dataset and for each Decision Tree stores information on which branch of the tree classifies that point which in turn will allow the t-SNE visualizations to show which parts of the Decision Tree are modeling which parts of the problem.

Selecting the trees and obtaining the mapping of each point in the dataset to branches of every tree is a lengthy task which makes initialization slow. As an alternative we could use lazy loading with the mappings being calculated to each tree only when necessary, however that would make usage slower when selecting other trees. We opted to store the result of this initialization in a pickle (python object file), therefore the initialization only has to occur when new data is added and the remaining times we simply read it directly from the pickle.

This backend exposes 2 data retrieving endpoints which allow obtaining meta data on every tree or obtaining detailed information of a single tree such as the python code that represents the tree, it's macro-structure in a Binary Tree like object and the t-SNE point mappings to each branch of the tree. Besides that it exposes 2 endpoints for classification which allow obtaining classification of a single instance as well as the branch of the tree that made that prediction and another endpoint which allows batch prediction so that many predictions can be obtained quickly without dealing with the latency of many consecutive HTTP requests.

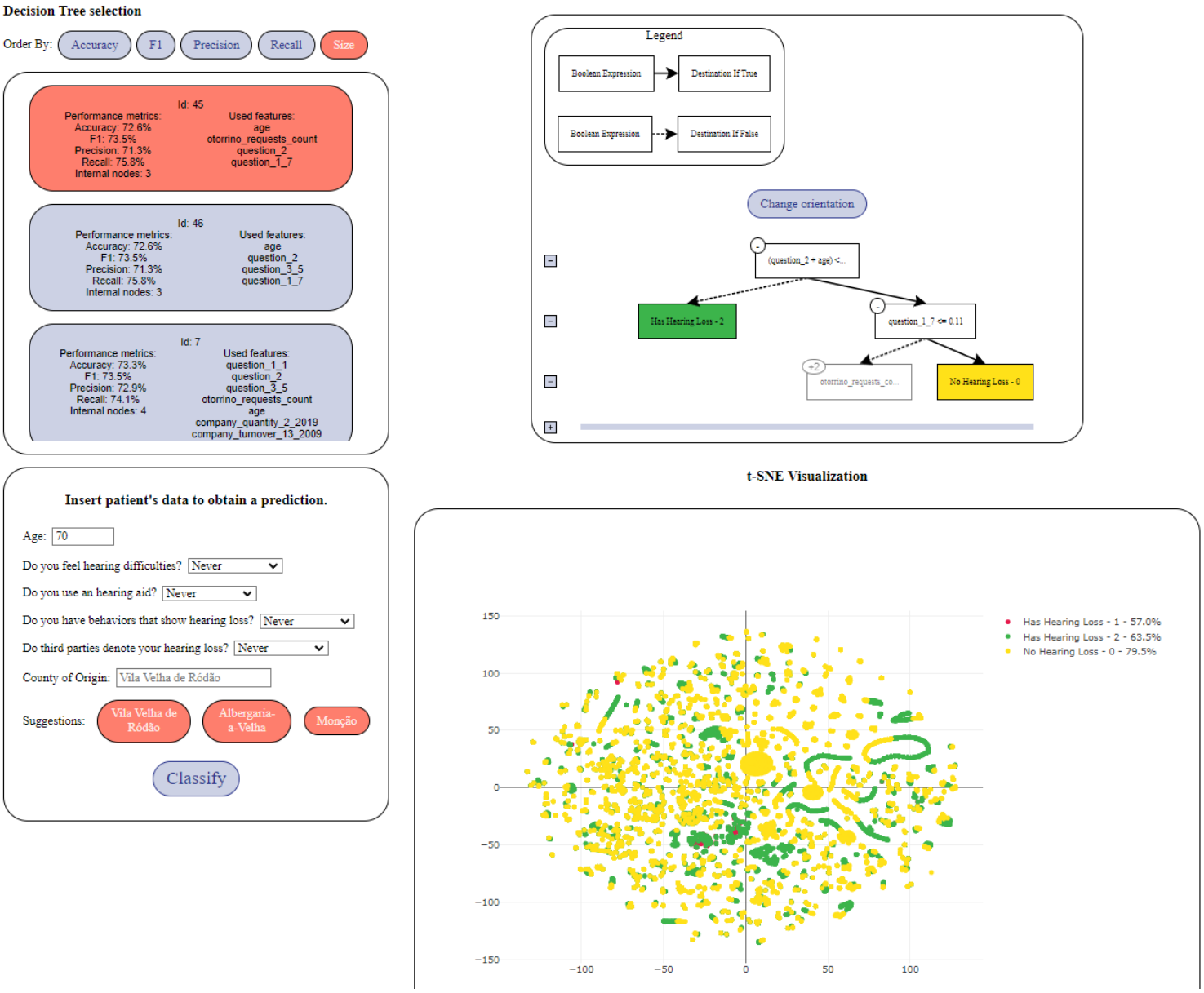


Figure 5.8: Overview of the developed Visualization and Classification tool.

Frontend

The frontend was built as a Vue.js web app. We took a component based approach to build the interface and every control. The Decision Tree visualization was built using Vue.js components and a SVG building library. Finally, the t-SNE visualization made use of plotly.js. All presented data is retrieved from the backend.

Deployment

To ensure easy deployment and usage of this tool we made use of Docker. Each component was made ready for deployment as a standalone container, by writing a Dockerfile for each of them. While the frontend component does not have any use without communication with the backend, this allows for the backend to also be used standalone for example for integration with other systems. To also make deployment of the tool easier we wrote a Docker Compose file which boots up both containers and provides all required configurations such as environment variables and port publishing.

Besides deploying the tool it is also important for the client company to be able to evolve new Decision Trees from any new data they have. For that purpose we also dockerized the HyTEA solution which evolves Decision Trees and appends them to the SQLite database so that they are made available to the tool.

5.2.2 Features

While this tool consists of a single page, it has implemented many features to make sure end users can take full advantage of the visualization potential. For a starter, the tool displays a list of the available Decision Trees displaying their performance metrics (accuracy, F1, precision and recall), their size and the features it uses. The tool allows sorting this list by any of the performance metrics or by size so that the user can find an appropriate tree faster.

When a tree is selected, it is displayed on the screen, with each terminal node being colored differently. The colors of these terminal nodes are then used on the t-SNE visualization so that we can see which branch of the selected Decision Tree classifies which points, and what is the accuracy of each branch. When a node does not lead to the classification of any of the points in the dataset we color it black. For example, in Figure 5.10 we can see that the orange branch of the Decision Tree represented in Figure 5.9 classifies most points in the central cluster of the t-SNE projection and that it has an accuracy of 95.5%.

Visualizing large Decision Trees can be difficult, therefore we also implemented features that allow compacting and visualizing only parts of the tree at a time. Namely we allow collapsing nodes, hiding levels and changing the orientation of the tree. We also shorten the text on long nodes which can be seen on mouse hover in a tooltip. Collapsing nodes hides every descendant node starting at

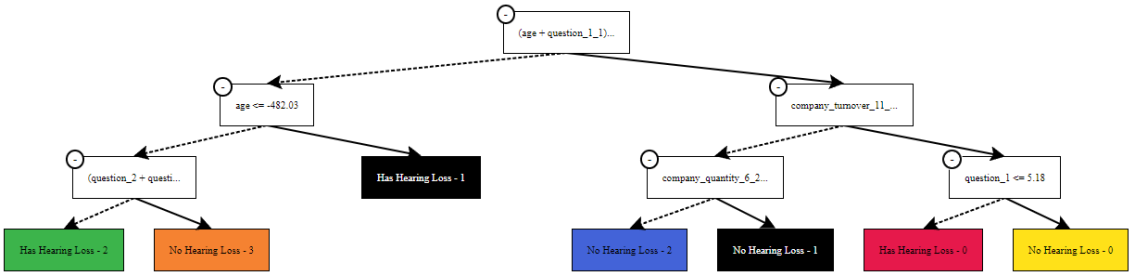


Figure 5.9: Example visualization of a Decision Tree with our tool.

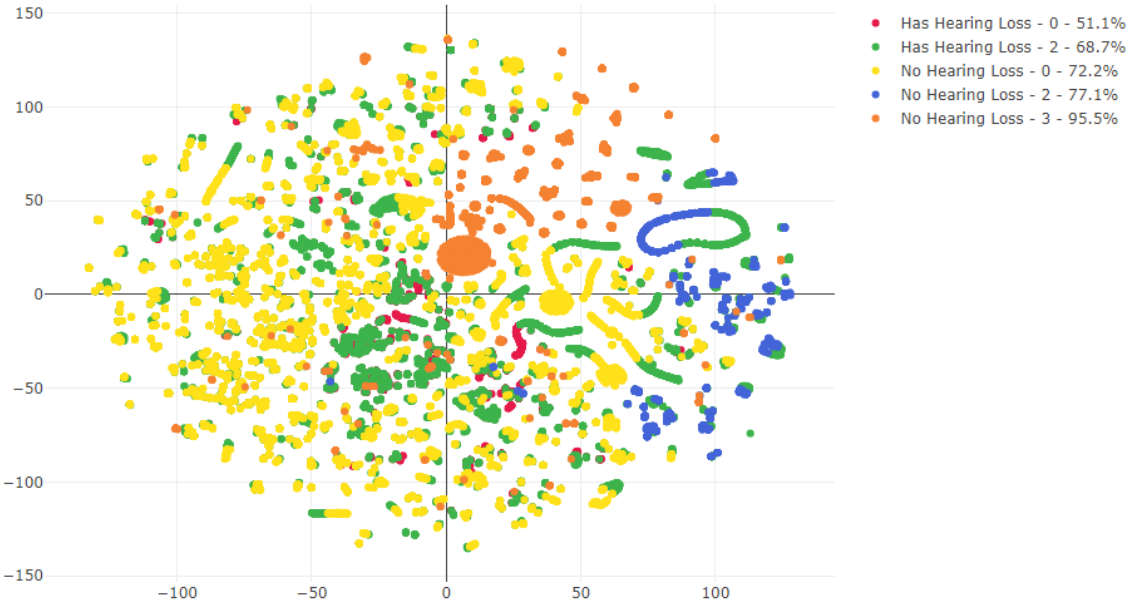


Figure 5.10: Example t-SNE visualization with our tool.

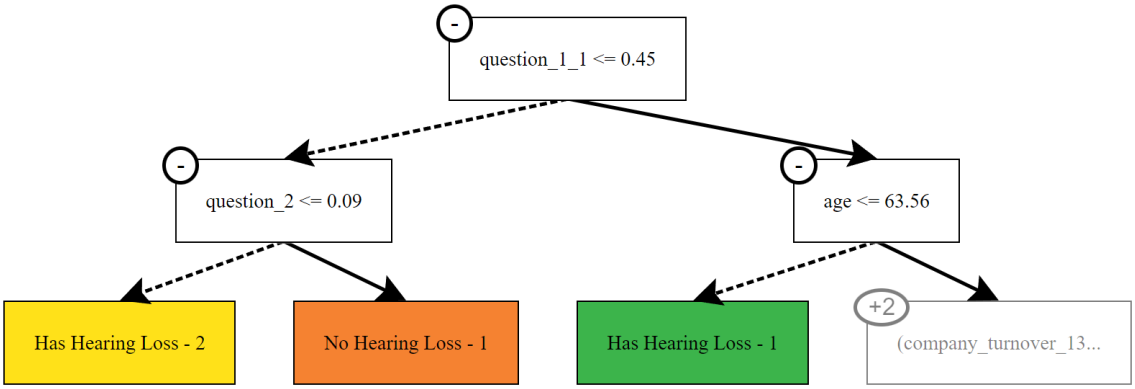


Figure 5.11: Example visualization of a Decision Tree with the last node of the 3rd level collapsed. The "+2" indicates there are 2 descendants to that node.

a selected node. If trees are large enough every node at the 3rd level will be collapsed by default. Hiding levels allows omitting every node in that level and can be useful for example to view only the top most levels of very deep Decision Trees. Examples of all these 4 features can be found in Figures 5.11 to 5.14.

Besides visualizing the trees and the respective t-SNE, the tool allows performing classification and showing the branch of the Decision Tree which is responsible for it. To do this, the user must insert the answer to 4 questions, the age of the patient and the name of the patient's county of origin. This data will be sent to the backend to calculate every other required feature, scale the values and perform the classification. It is important for the name of the county to be written correctly, however searching a list of the 308 Portuguese counties can be cumbersome. To solve this we allow the user to write the name of the county manually and give him auto complete suggestions to avoid mistakes. When making a classification we inform the user on the output and highlight the responsible tree branch as exemplified in Figure 5.15.

5.3 Summary

In this chapter we described how we use visualization to both validate our models and assist health professionals' decision making. We saw multiple t-SNE visualizations of the feature space with coloring by known class and predicted class and saw that our generated Decision Trees capture the patterns in original data. Then we saw the built visualization tool, it's components, architecture and features, and how it can be used to assist health professionals on assessing the reliability of a given prediction from our HyTEA generated Decision Trees.

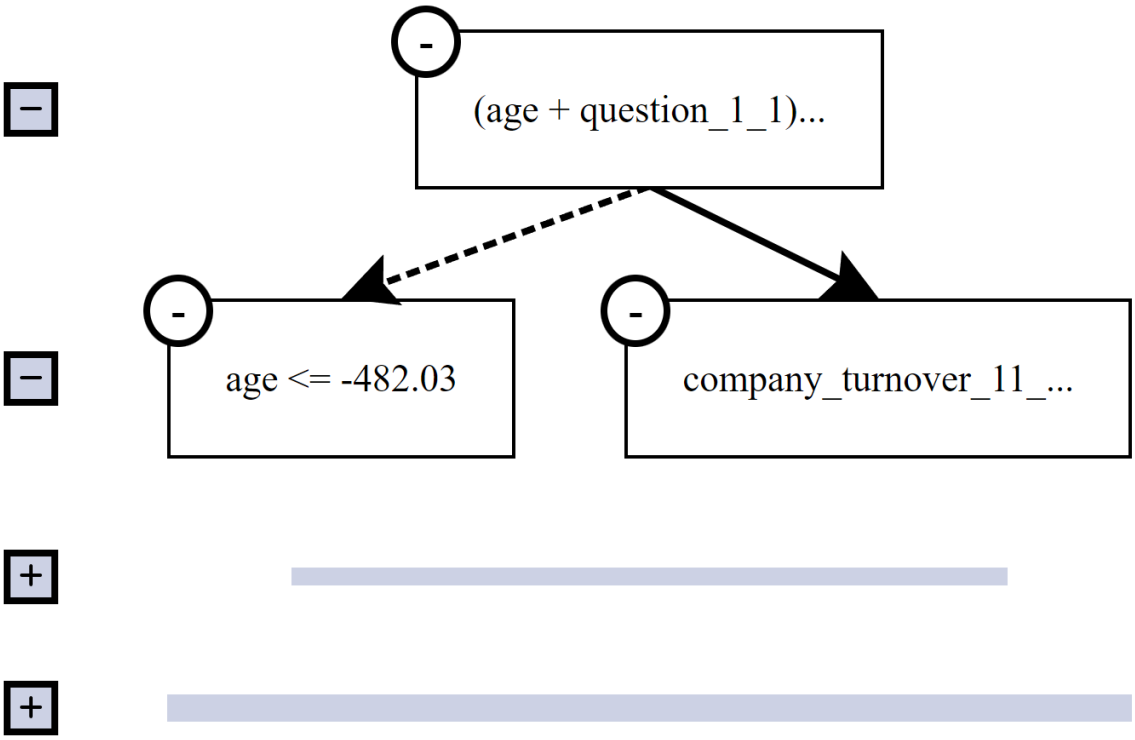


Figure 5.12: Example visualization of a Decision Tree with the last 2 levels hidden. The length and width of the gray lines are indicative of the amount of hidden nodes on that level.

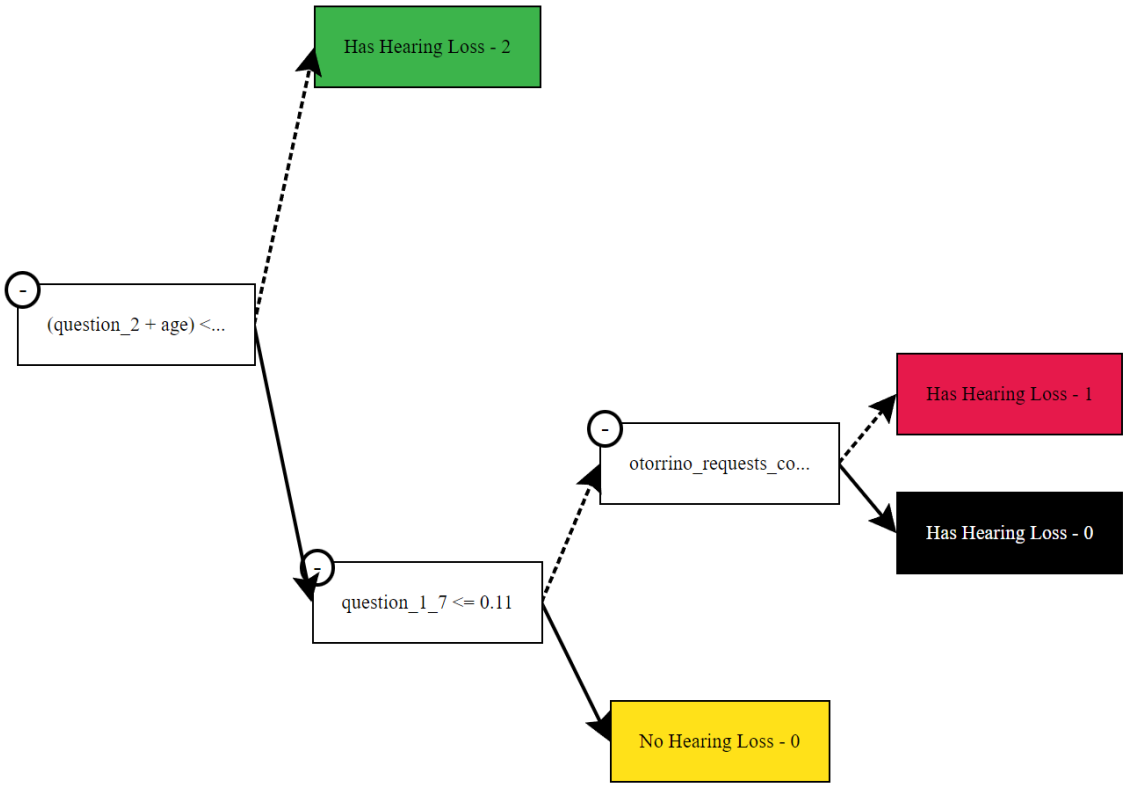


Figure 5.13: Example visualization of a Decision Tree rotated to the horizontal.

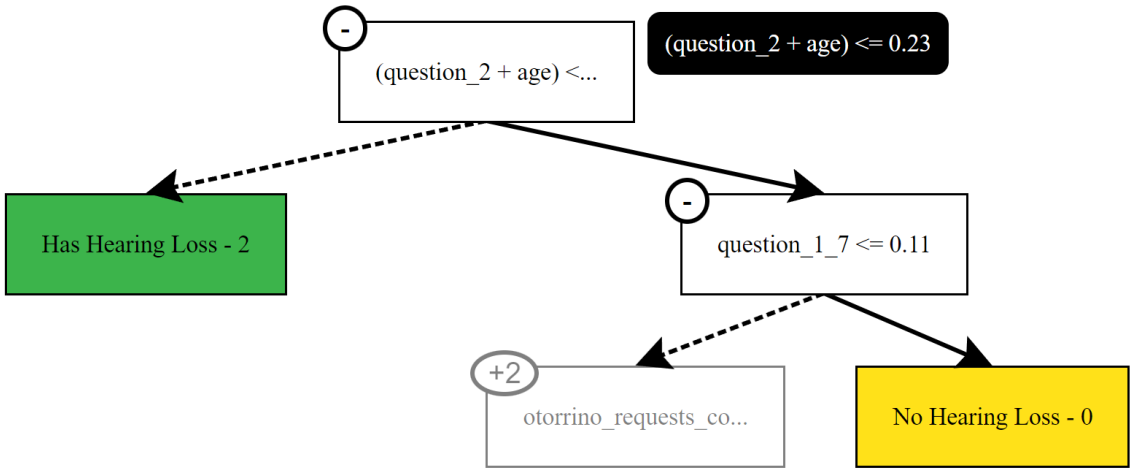


Figure 5.14: Example visualization of a Decision Tree with the tooltip showing the rest of the node's shortened text.

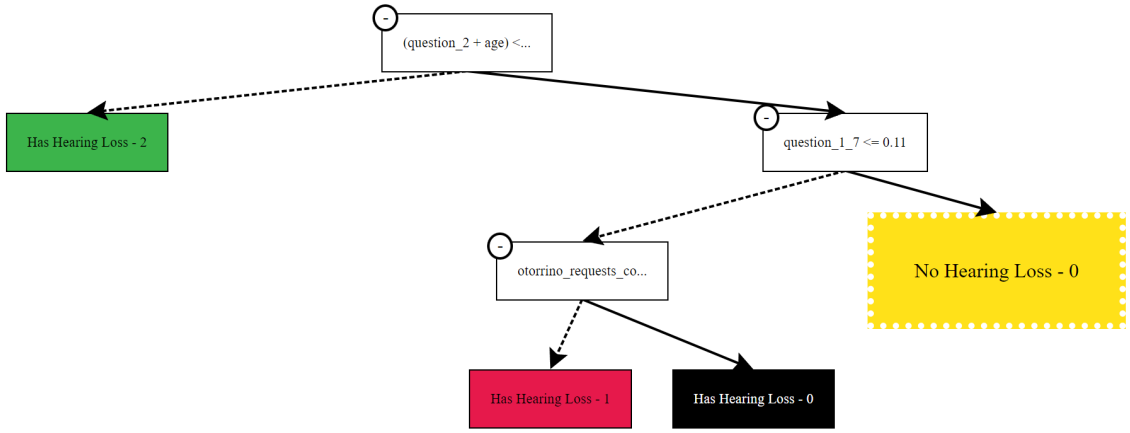


Figure 5.15: Example visualization of a Decision Tree after classification where a patient was classified as not having hearing loss by the highlighted yellow "No Hearing Loss - 0" branch.

Chapter 6

Conclusion

In this dissertation we tackle the problem of developing interpretable classifiers for hearing loss diagnosis. We favor an evolutionary approach due to it's global optimization properties and propose HyTEA, an hybrid evolutionary algorithm with Structured Grammatical Evolution to develop Decision Tree structures and select features and Differential Evolution to select parameters for the SGE generated structures. We also propose HyTEABoost and HyTEAForest, variants of HyTEA that allow evolving Gradient Boosting and Random Forest ensembles.

We experimented solving the hearing loss diagnosis problem with both our approaches and default Sklearn models. In the traditional Sklearn models we saw that Recall performance was low, which is particularly problematic in our domain, since that means we're failing to find patients with hearing loss. We saw that HyTEA managed to significantly improve performance, consistently developing good Decision Trees and with better Recall than those generated by traditional models. We also saw that HyTEABoost is capable of further increasing this Recall metric, although at a slight cost for precision and interpretability. Then we saw that Random Forests can also be evolved, with a small experiment evolving ensembles of just 2 Decision Trees, we achieved the best F1 and Recall metrics across all alternatives, evolutionary or traditional.

A big part of this project was interpretability, since health professionals must understand why a given decision is made. With this in mind we developed a web based tool to help visualize Decision Trees and the paths followed to reach a classification for any particular case. This same tool allows for visualization of the t-SNE projection of every point in the dataset and which particular branches of the Decision Tree classify each zone of this projection, allowing for health professionals to have a sense of the uncertainty associated to each classification.

Therefore the main achievements of this work were the development of problem agnostic algorithms to evolve Decision Trees, Gradient boosting ensembles and Random Forest ensembles, the development of Decision Tree and Gradient Boosting models for predicting hearing loss diagnoses, with emphasis on avoiding false negatives and the development of a visualization tool that integrates the developed Decision Tree models and facilitates usage and interpretation by health professionals.

As future work we intend to work on integrating Gradient Boosting and Random Forest models on our visualization tool, creating mechanisms to accelerate the process of retraining models as new data arrives and investigate possible sources of erroneous data such as the model of the headset used for the audiometry exams. It could also be beneficial to increase the amount of data we collect during questionnaires since for example, it was shown that economic factors were to some point indicative of the propensity to have hearing loss.

References

Mohsen Aliabadi, Maryam Farhadian, and Ebrahim Darvishi. Prediction of hearing loss among the noise exposed workers in a steel factory using artificial intelligence approach. *International Archives of Occupational and Environmental Health*, 88, 11 2014. doi: 10.1007/s00420-014-1004-z.

Filipe Assunção, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. Towards the evolution of multi-layered neural networks: A dynamic structured grammatical evolution approach. pages 393–400, 07 2017. doi: 10.1145/3071178.3071286.

Taiwo Oladipupo Ayodele. Introduction to machine learning. In Yagang Zhang, editor, *New Advances in Machine Learning*, chapter 1. IntechOpen, Rijeka, 2010. doi: 10.5772/9394. URL `https://doi.org/10.5772/9394`.

Rodrigo Barros, Márcio Basgalupp, Andre de Carvalho, and Alex Freitas. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42:291–312, 01 2012. doi: 10.1109/TSMCC.2011.2157494.

Aniruddha Bhandari. Everything you should know about confusion matrix for machine learning. `https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning`, 2020. [Online; accessed 09-December-2021].

Dan Bing, Jun Ying, Jun Miao, Lan Lan, Dayong Wang, Lidong Zhao, Zifang Yin, Lan Yu, Jing Guan, and Wang Qiuju. Predicting the hearing outcome in sudden sensorineural hearing loss via machine learning models. *Clinical Otolaryngology*, 43, 01 2018. doi: 10.1111/coa.13068.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Hakan Bozcuk, Uğur Bilge, Emine Koyuncu, and Kemal Gülkesen. An application of a genetic algorithm in conjunction with other data mining methods for estimating outcome after hospitalization in cancer patients. *Medical science monitor : international medical journal of experimental and clinical research*, 10:CR246–51, 06 2004.

Anthony Brabazon, Michael O'Neill, and Sen. McGarraghy. *Essentials of Metaheuristics*. Springer Publishing Company, Incorporated, 2015.

Zuwei Cao, Fei Zhao, Feifan Chen, and Emad Grais. Contributions and limitations of using machine learning to predict noise-induced hearing loss. *International Archives of Occupational and Environmental Health*, 07 2021. doi: 10.1007/s00420-020-01648-w.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006. ISBN 9780262033589. URL `http://dblp.uni-trier.de/db/books/collections/CSZ2006.html`.

Xi Chen, Qinghua Zhou, Rushi Lan, Shuihua Wang, Yu-Dong Zhang, and Xiaonan Luo. Sensorineural hearing loss classification via deep-hlnet and few-shot learning. *Multimedia Tools and Applications*, 80:1–14, 01 2021. doi: 10.1007/s11042-020-09702-y.

Stephen Dignum and Riccardo Poli. Operator equalisation and bloat free gp. In Michael O'Neill, Leonardo Vanneschi, Steven Gustafson, Anna Isabel Esparcia Alcázar, Ivanoe De Falco, Antonio Della Cioppa, and Ernesto Tarantino, editors, *Genetic Programming*, pages 110–121, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-78671-9.

Hossein ElahiShirvan, MohammadReza Ghotbi-Ravandi, Sajad Zare, and Mostafa Ahsaee. Using audiometric data to weigh and prioritize factors that affect workers' hearing loss through support vector machine (svm) algorithm. *Sound&Vibration*, 54:99–112, 01 2020. doi: 10.32604/sv.2020.08839.

Maryam Farhadian, Mohsen Aliabadi, and Ebrahim Darvishi. Empirical estimation of the grades of hearing impairment among industrial workers based on new artificial neural networks and classical regression methods. *Indian journal of occupational and environmental medicine*, 19, 05 2015. doi: 10.4103/0019-5278.165337.

Ali Ghaheri, Saeed Shoar, Mohammad Naderan, and Sayed shahabuddin Hoseini. The applications of genetic algorithms in medicine. *Oman Medical Journal*, 30:406–416, 11 2015. doi: 10.5001/omj.2015.82.

Brandon Greenwell, Anthony Tvaryanas, and Genny Maupin. Risk factors for hearing decrement among u.s. air force aviation-related personnel. *Aerospace Medicine and Human Performance*, 89:80–86, 02 2018. doi: 10.3357/AMHP.4988.2018.

Trevor Hastie. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 01 2009. ISBN 9780387848570. doi: 10.1007/978-0-387-84858-7.

Ignacio Hidalgo, J. Colmenar, José Risco-Martin, Alfredo Cuesta-Infante, Esther Maqueda, Marta Botella, and José Rubio. Modeling glycemia in humans by means of grammatical evolution. *Applied Soft Computing*, 20:40–53, 07 2014. doi: 10.1016/j.asoc.2013.11.006.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995. doi: 10.1109/ICDAR.1995.598994.

Jeong-Kyu Hoh, Kyung Cha, Moon-Il Park, Mei-Ling Lee, and Young-sun Park. Estimating time to full uterine cervical dilation using genetic algorithm. *The Kaohsiung journal of medical sciences*, 28:423–8, 08 2012. doi: 10.1016/j.kjms.2012.02.012.

Ahmad Khalil, Brett Bouma, and Mohammad Mofrad. A combined fem/genetic algorithm for vascular soft tissue elasticity estimation. *Cardiovascular engineering (Dordrecht, Netherlands)*, 6:93–102, 10 2006. doi: 10.1007/s10558-006-9013-5.

William Langdon and Adil Qureshi. Genetic programming — computers using "natural selection" to generate programs. 11 1995. doi: 10.1007/978-1-4615-5731-9_2.

Tomasz Latkowski and Stanisław Osowski. Computerized system for recognition of autism on the basis of gene expression microarray data. *Computers in Biology and Medicine*, 11 2014. doi: 10.1016/j.compbiomed.2014.11.004.

Michael Lones, Jane Alty, Jeremy Cosgrove, Philippa Duggan-Carter, Stuart Jamieson, Rebecca Naylor, Andrew Turner, and Stephen Smith. A new evolutionary algorithm-based home monitoring device for parkinson's dyskinesia. *Journal of Medical Systems*, 41, 09 2017. doi: 10.1007/s10916-017-0811-7.

Gilles Louppe. *Understanding Random Forests: From Theory to Practice*. PhD thesis, 10 2014.

Nuno Lourenço, Francisco Pereira, and Ernesto Costa. Unveiling the properties of structured grammatical evolution. *Genetic Programming and Evolvable Machines*, 17, 09 2016. doi: 10.1007/s10710-015-9262-4.

Nuno Lourenço, Joaquim Ferrer, Francisco Pereira, and Ernesto Costa. A comparative study of different grammar-based genetic programming approaches. pages 311–325, 03 2017. ISBN 978-3-319-55695-6. doi: 10.1007/978-3-319-55696-3_20.

Nuno Lourenço, Filipe Assunção, Francisco Pereira, Ernesto Costa, and Penousal Machado. *Structured grammatical evolution: A dynamic approach*, pages 137–161. 01 2018. doi: 10.1007/978-3-319-78717-6_6.

Nuno Lourenço, J. Colmenar, Ignacio Hidalgo, and Oscar Garnica. Structured grammatical evolution for glucose prediction in diabetic patients. pages 1250–1257, 07 2019. doi: 10.1145/3321707.3321782.

Nuno Lourenço, J. Colmenar, Ignacio Hidalgo, and Sancho Salcedo-Sanz. Evolving energy demand estimation models over macroeconomic indicators. pages 1143–1149, 06 2020. doi: 10.1145/3377930.3390153.

Sean Luke. *Essentials of Metaheuristics*. 2013.

Sean Luke and Liviu Panait. A comparison of bloat control methods for genetic programming. *Evolutionary computation*, 14:309–44, 02 2006. doi: 10.1162/evco.2006.14.3.309.

J. P. Marques de Sá. *Pattern recognition :*. Springer-Verlag Berlin Heidelberg,, New York :, c2001. Includes index.

Ch.B. M.B, Neil Pendleton, Sam Ph.D, and Michael Horan. Comparison of a genetic algorithm neural network with logistic regression for predicting outcome after surgery for patients with nonsmall cell lung carcinoma. *Cancer*, 79:1338 – 1342, 04 1997. doi: 10.1002/(SICI)1097-0142(19970401)79:7<1338::AID-CNCR10>3.0.CO;2-0.

Olaf Michel and M. Liedtke. Iso 1999:2013 teil 1iso 1999:2013 part 1: Überarbeitetes wahrscheinlichkeitsmodell zur berechnung des lärmbedingten hörverlustsrevised probability model for calculating noise-induced hearing loss. *HNO*, 69, 02 2021. doi: 10.1007/s00106-021-00999-1.

Keon Vin Park, Oh Kyoung Ho, Yong Jeong, Jihye Rhee, Mun Han, Sung Han, and June Choi. Machine learning models for predicting hearing prognosis in unilateral idiopathic sudden sensorineural hearing loss. *Clinical and experimental otorhinolaryngology*, 13, 03 2020. doi: 10.21053/ceo.2019.01858.

PORDATA. `https://www.pordata.pt/`. [Online; accessed 19-December-2021].

Portal de Transparência do SNS. `https://www.sns.gov.pt/transparencia/`. [Online; accessed 19-December-2021].

Kedar Potdar, Taher Pardawala, and Chinmay Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*, 175:7–9, 10 2017. doi: 10.5120/ijca2017915495.

Ross Quinlan. *C4.5: Programs for Machine Learning*, volume 1. 01 1993. ISBN 1-55860-238-0.

Laura Raileanu and Kilian Stoffel. Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41: 77–93, 05 2004. doi: 10.1023/B:AMAI.0000018580.96245.c6.

Conor Ryan, J. J. Collins, and Michael O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In Wolfgang Banzhaf, Riccardo Poli, Marc Schoenauer, and Terence C. Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391 of *LNCS*, pages 83–96, Paris, 14-15 April 1998. Springer-Verlag. ISBN 3-540-64360-5. doi: doi: 10.1007/BFb0055930. URL `http://www.lania.mx/~ccoello/eurogp98.ps.gz`.

Mehrin Saremi and Farzin Yaghmaee. Evolutionary decision tree induction with multi-interval discretization. pages 1–6, 02 2014. ISBN 978-1-4799-3351-8. doi: 10.1109/IranianCIS.2014.6802543.

Sara Silva and Ernesto Costa. Dynamic limits for bloat control. pages 666–677, 06 2004. ISBN 978-3-540-22343-6. doi: 10.1007/978-3-540-24855-2_74.

Sklearn documentation. Receiver operating characteristic (roc). `https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html`. [Online; accessed 09-December-2021].

Rainer Storn and K. Price. De-a simple and efficient heuristic for global optimization over continuous space. *Joumal of Global Optimization*, 114:341–359, 01 1997.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 978-0-262-19398-6. URL `https://www.worldcat.org/oclc/37293240`.

Alaa Tharwat. Classification assessment methods: a detailed tutorial. 08 2018. doi: 10.1016/j.aci.2018.08.003.

Olga Troyanskaya, Mike Cantor, Gavin Sherlock, Trevor Hastie, Rob Tibshirani, David Botstein, and Russ Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17:520–525, 07 2001. doi: 10.1093/bioinformatics/17.6.520.

Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, Amsterdam, 3 edition, 2011. ISBN 978-0-12-374856-0. URL `http://www.sciencedirect.com/science/book/9780123748560`.

Sajad Zare, N. Hasheminejad, H.E. Shirvan, Davoud Hasanvand, Rasoul Hemmatjo, and Saeid Ahmadi. Assessing individual and environmental sound pressure level and sound mapping in iranian safety shoes factory. *Romanian Journal of Acoustics and Vibration*, 15:20–25, 01 2018.

Sajad Zare, Mohammad Reza Ghotbi Ravandi, Hossein ElahiShirvan, Mostafa Ahsaee, and Mina Rostami. Predicting and weighting the factors affecting workers' hearing loss based on audiometric data using c5 algorithm. *Annals of Global Health*, 85, 06 2019. doi: 10.5334/aogh.2522.

Yanxia Zhao, Jingsong Li, Meibian Zhang, Yao Lu, Hongwei Xie, Yu Tian, and Wei Qiu. Machine learning models for the hearing impairment prediction in workers exposed to complex industrial noise: A pilot study. *Ear and Hearing*, 40:1, 08 2018. doi: 10.1097/AUD.0000000000000649.

Yanxia Zhao, Yu Tian, Meibian Zhang, Jingsong Li, and Wei Qiu. Development of an automatic classifier for the prediction of hearing impairment from industrial noise exposure. *The Journal of the Acoustical Society of America*, 145:2388–2400, 04 2019. doi: 10.1121/1.5096643.

Appendices

Appendix A

Extracted and Engineered Features

Question features are the answers to the following:

1. Do you feel hearing difficulties?
2. Do you use an hearing aid?
3. Do you have behaviors that show hearing loss?
4. Do third parties denote your hearing loss?

Answers to questions are the following:

1. Never
2. No
3. Sometimes
4. Not sure
5. In noisy places
6. Yes
7. Always

The number between 1 and 20 on the company features represents the activity type and corresponds to the following:

1. Information and Communication Activities
2. Administrative and Support Service Activities
3. Artistic, Entertainment, Sporting and Recreational Activities
4. Consulting, Scientific and Technical Activities

5. Human Health and Social Support Activities
6. Real Estate Activities
7. Agriculture, Animal Production, Hunting, Forestry and Fishing Activities
8. Agriculture, Animal Production, Hunting, Forestry and Fishing Activities
9. Lodging, Catering and similar
10. Capture, Treatment and Distribution of Water
11. Retail
12. Construction
13. Education
14. Electricity, Gas, Steam, Hot Water and Air Conditioning
15. Industry, Construction, Energy and Water
16. Extracting Industries
17. Transforming Industries
18. Other Activities and Services
19. Total
20. Transport and Storage

The complete list of extracted and engineered features is as follows (bold features were kept after feature selection):

1. **lat - Latitude**
2. **lng - Longitude**
3. **question_1 - Label Encoded answer to Question 1**
4. **question_2 - Label Encoded answer to Question 1**
5. **question_3 - Label Encoded answer to Question 1**
6. question_4 - Label Encoded answer to Question 1
7. hypertension_last_6_months_mean
8. hypertension_last_6_months_std
9. hypertension_last_6_months_min
10. hypertension_last_6_months_25%
11. hypertension_last_6_months_50%

12. hypertension_last_6_months_75%
13. **hypertension_last_6_months_max**
14. hypertension_less_65_years_mean
15. hypertension_less_65_years_std
16. **hypertension_less_65_years_min**
17. hypertension_less_65_years_25%
18. hypertension_less_65_years_50%
19. hypertension_less_65_years_75%
20. **hypertension_less_65_years_max**
21. diabetes_last_year_mean
22. diabetes_last_year_std
23. diabetes_last_year_min
24. diabetes_last_year_25%
25. diabetes_last_year_50%
26. diabetes_last_year_75%
27. diabetes_last_year_max
28. diabetes_proportion_last_year_mean
29. **diabetes_proportion_last_year_std**
30. diabetes_proportion_last_year_min
31. diabetes_proportion_last_year_25%
32. diabetes_proportion_last_year_50%
33. diabetes_proportion_last_year_75%
34. diabetes_proportion_last_year_max
35. diabetes_hgba1c_less_8_mean
36. diabetes_hgba1c_less_8_std
37. diabetes_hgba1c_less_8_min
38. diabetes_hgba1c_less_8_25%
39. diabetes_hgba1c_less_8_50%
40. diabetes_hgba1c_less_8_75%

41. **diabetes_hgba1c_less_8_max**
42. diabetes_proportion_hgba1c_less_8_mean
43. diabetes_proportion_hgba1c_less_8_std
44. **diabetes_proportion_hgba1c_less_8_min**
45. diabetes_proportion_hgba1c_less_8_25%
46. diabetes_proportion_hgba1c_less_8_50%
47. diabetes_proportion_hgba1c_less_8_75%
48. diabetes_proportion_hgba1c_less_8_max
49. stroke_fatality_rate_haemorrhagic_mean
50. stroke_fatality_rate_haemorrhagic_std
51. **stroke_fatality_rate_haemorrhagic_min**
52. stroke_fatality_rate_haemorrhagic_25%
53. **stroke_fatality_rate_haemorrhagic_50%**
54. **stroke_fatality_rate_haemorrhagic_75%**
55. stroke_fatality_rate_haemorrhagic_max
56. stroke_fatality_rate_ischemic_mean
57. stroke_fatality_rate_ischemic_std
58. stroke_fatality_rate_ischemic_min
59. stroke_fatality_rate_ischemic_25%
60. stroke_fatality_rate_ischemic_50%
61. stroke_fatality_rate_ischemic_75%
62. **stroke_fatality_rate_ischemic_max**
63. **otorrino_acts_count**
64. otorrino_acts_mean
65. otorrino_acts_std
66. otorrino_acts_min
67. otorrino_acts_25%
68. **otorrino_acts_50%**
69. otorrino_acts_75%

70. otorrino_acts_max
71. otorrino_requests_count
72. otorrino_requests_mean
73. otorrino_requests_std
74. otorrino_requests_min
75. otorrino_requests_25%
76. otorrino_requests_50%
77. otorrino_requests_75%
78. otorrino_requests_max
79. schoooling_less_ciclo_1_1985
80. schoooling_ciclo_1_1985
81. **schoooling_ciclo_2_1985**
82. schoooling_ciclo_3_1985
83. schoooling_secundario_plus_1985
84. schoooling_superior_1985
85. **schoooling_less_ciclo_1_2018**
86. schoooling_ciclo_1_2018
87. **schoooling_ciclo_2_2018**
88. schoooling_ciclo_3_2018
89. schoooling_secundario_plus_2018
90. schoooling_superior_2018
91. age_group_total_1960
92. age_group_ages_0_14_1960
93. age_group_ages_15_64_1960
94. **age_group_ages_65_plus_1960**
95. age_group_total_2011
96. **age_group_ages_0_14_2011**
97. age_group_ages_15_64_2011
98. age_group_ages_65_plus_2011

99. aging_ratio_2001
100. **aging_ratio_2019**
101. company_salary_7_1985
102. **company_salary_11_1985**
103. **company_salary_14_1985**
104. company_salary_16_1985
105. **company_salary_18_1985**
106. company_salary_19_1985
107. **company_quantity_0_2009**
108. company_turnover_0_2009
109. company_quantity_1_2009
110. company_turnover_1_2009
111. company_quantity_2_2009
112. **company_turnover_2_2009**
113. company_quantity_3_2009
114. company_turnover_3_2009
115. company_quantity_4_2009
116. company_turnover_4_2009
117. company_quantity_5_2009
118. company_turnover_5_2009
119. company_quantity_6_2009
120. **company_turnover_6_2009**
121. company_quantity_8_2009
122. company_turnover_8_2009
123. company_quantity_9_2009
124. company_turnover_9_2009
125. company_quantity_10_2009
126. **company_turnover_10_2009**
127. company_quantity_11_2009

128. **company_turnover_11_2009**
129. company_quantity_12_2009
130. company_turnover_12_2009
131. company_quantity_13_2009
132. **company_turnover_13_2009**
133. company_quantity_15_2009
134. company_turnover_15_2009
135. company_quantity_16_2009
136. company_turnover_16_2009
137. company_quantity_17_2009
138. company_turnover_17_2009
139. company_quantity_19_2009
140. company_turnover_19_2009
141. company_quantity_20_2009
142. company_turnover_20_2009
143. company_salary_7_2018
144. **company_salary_11_2018**
145. company_salary_14_2018
146. **company_salary_16_2018**
147. company_salary_18_2018
148. company_salary_19_2018
149. **company_quantity_0_2019**
150. company_turnover_0_2019
151. company_quantity_1_2019
152. company_turnover_1_2019
153. **company_quantity_2_2019**
154. company_turnover_2_2019
155. company_quantity_3_2019
156. company_turnover_3_2019

157. company_quantity_4_2019
158. **company_turnover_4_2019**
159. company_quantity_5_2019
160. company_turnover_5_2019
161. **company_quantity_6_2019**
162. company_turnover_6_2019
163. company_quantity_8_2019
164. company_turnover_8_2019
165. company_quantity_9_2019
166. company_turnover_9_2019
167. company_quantity_10_2019
168. company_turnover_10_2019
169. **company_quantity_11_2019**
170. **company_turnover_11_2019**
171. company_quantity_12_2019
172. company_turnover_12_2019
173. company_quantity_13_2019
174. company_turnover_13_2019
175. **company_quantity_15_2019**
176. company_turnover_15_2019
177. **company_quantity_16_2019**
178. company_turnover_16_2019
179. company_quantity_17_2019
180. company_turnover_17_2019
181. company_quantity_19_2019
182. company_turnover_19_2019
183. company_quantity_20_2019
184. company_turnover_20_2019
185. **age**

186. **question_1_1 - One-Hot Encoded answer 1 to Question 1**
187. **question_1_2 - One-Hot Encoded answer 2 to Question 1**
188. **question_1_3 - One-Hot Encoded answer 3 to Question 1**
189. **question_1_4 - One-Hot Encoded answer 4 to Question 1**
190. **question_1_5 - One-Hot Encoded answer 5 to Question 1**
191. **question_1_6 - One-Hot Encoded answer 6 to Question 1**
192. **question_1_7 - One-Hot Encoded answer 7 to Question 1**
193. **question_2_1 - One-Hot Encoded answer 1 to Question 2**
194. **question_2_2 - One-Hot Encoded answer 2 to Question 2**
195. **question_2_3 - One-Hot Encoded answer 3 to Question 2**
196. question_3_1 - One-Hot Encoded answer 1 to Question 3
197. question_3_2 - One-Hot Encoded answer 2 to Question 3
198. **question_3_3 - One-Hot Encoded answer 3 to Question 3**
199. **question_3_4 - One-Hot Encoded answer 4 to Question 3**
200. **question_3_5 - One-Hot Encoded answer 5 to Question 3**
201. **question_3_6 - One-Hot Encoded answer 6 to Question 3**
202. question_4_1 - One-Hot Encoded answer 1 to Question 4
203. **question_4_2 - One-Hot Encoded answer 2 to Question 4**
204. **question_4_3 - One-Hot Encoded answer 3 to Question 4**

Table A.1: Mean and Standard Deviation of each feature for each class and result of Independent t-test verification of significant difference between classes.

	No Hearing Loss		Hearing Loss		
Feature	**Mean**	**STD**	**Mean**	**STD**	**Stat Diff**
lat	3.98e+01	1.37e+00	4.00e+01	1.5e+00	True
lng	-8.86e+00	1.22e+00	-8.74e+00	1.0e+00	True
question_1	2.28e+00	2.04e+00	3.50e+00	2.3e+00	True
question_2	1.58e+00	9.44e-01	2.12e+00	8.1e-01	True
question_3	1.32e+00	1.56e+00	1.62e+00	2.0e+00	True
question_4	2.43e-01	9.22e-01	2.31e-01	9.7e-01	False
hypertension_last _6_months_mean	5.00e+03	1.28e+03	5.06e+03	1.4e+03	True
hypertension_last _6_months_std	2.29e+03	5.61e+02	2.31e+03	6.0e+02	True

hypertension_last _6_months_min	8.35e+02	3.13e+02	8.46e+02	3.4e+02	True
hypertension_last _6_months_25%	3.19e+03	8.70e+02	3.23e+03	9.2e+02	True
hypertension_last _6_months_50%	4.99e+03	1.28e+03	5.05e+03	1.4e+03	True
hypertension_last _6_months_75%	6.94e+03	1.80e+03	7.02e+03	1.9e+03	True
hypertension_last _6_months_max	9.28e+03	2.14e+03	9.37e+03	2.3e+03	True
hypertension_less _65_years_mean	3.17e+01	5.54e+00	3.21e+01	5.9e+00	True
hypertension_less _65_years_std	1.40e+01	2.32e+00	1.41e+01	2.5e+00	True
hypertension_less _65_years_min	5.63e+00	1.78e+00	5.71e+00	1.9e+00	True
hypertension_less _65_years_25%	2.08e+01	3.65e+00	2.10e+01	3.9e+00	True
hypertension_less _65_years_50%	3.18e+01	5.52e+00	3.22e+01	5.9e+00	True
hypertension_less _65_years_75%	4.42e+01	7.98e+00	4.47e+01	8.5e+00	True
hypertension_less _65_years_max	5.58e+01	8.34e+00	5.63e+01	9.1e+00	True
diabetes_last _year_mean	6.79e+03	1.96e+03	6.92e+03	2.2e+03	True
diabetes_last _year_std	2.92e+03	7.90e+02	2.96e+03	8.9e+02	True
diabetes_last _year_min	8.85e+02	3.96e+02	9.16e+02	4.5e+02	True
diabetes_last _year_25%	4.80e+03	1.45e+03	4.91e+03	1.6e+03	True
diabetes_last _year_50%	7.22e+03	2.14e+03	7.38e+03	2.3e+03	True
diabetes_last _year_75%	9.02e+03	2.62e+03	9.19e+03	2.9e+03	True
diabetes_last _year_max	1.21e+04	3.12e+03	1.23e+04	3.5e+03	True
diabetes_proportion _last_year_mean	4.49e+01	8.46e+00	4.56e+01	8.9e+00	True
diabetes_proportion _last_year_std	1.84e+01	2.59e+00	1.87e+01	2.8e+00	True
diabetes_proportion _last_year_min	5.67e+00	1.88e+00	5.81e+00	2.0e+00	True
diabetes_proportion _last_year_25%	3.20e+01	6.88e+00	3.27e+01	7.2e+00	True

diabetes_proportion _last_year_50%	4.91e+01	1.03e+01	5.01e+01	1.1e+01	True
diabetes_proportion _last_year_75%	5.94e+01	1.08e+01	6.04e+01	1.1e+01	True
diabetes_proportion _last_year_max	7.36e+01	8.96e+00	7.44e+01	9.6e+00	True
diabetes_hgba1c_less_8_mean	5.20e+03	1.65e+03	5.29e+03	1.9e+03	True
diabetes_hgba1c_less_8_std	2.63e+03	7.15e+02	2.66e+03	8.1e+02	True
diabetes_hgba1c_less_8_min	1.03e+03	5.11e+02	1.06e+03	5.8e+02	True
diabetes_hgba1c_less_8_25%	2.85e+03	1.01e+03	2.92e+03	1.2e+03	True
diabetes_hgba1c_less_8_50%	5.25e+03	1.80e+03	5.35e+03	2.1e+03	True
diabetes_hgba1c_less_8_75%	7.42e+03	2.23e+03	7.55e+03	2.5e+03	True
diabetes_hgba1c_less_8_max	1.02e+04	2.77e+03	1.03e+04	3.2e+03	True
diabetes_proportion_hgba1c _less_8_mean	3.35e+01	4.92e+00	3.40e+01	5.2e+00	True
diabetes_proportion_hgba1c _less_8_std	1.69e+01	2.03e+00	1.71e+01	2.2e+00	True
diabetes_proportion_hgba1c _less_8_min	6.13e+00	2.04e+00	6.26e+00	2.2e+00	True
diabetes_proportion_hgba1c _less_8_25%	1.86e+01	3.20e+00	1.89e+01	3.4e+00	True
diabetes_proportion_hgba1c _less_8_50%	3.36e+01	5.18e+00	3.41e+01	5.5e+00	True
diabetes_proportion_hgba1c _less_8_75%	4.88e+01	7.24e+00	4.94e+01	7.7e+00	True
diabetes_proportion_hgba1c _less_8_max	6.19e+01	6.53e+00	6.24e+01	7.1e+00	True
stroke_fatality_rate _haemorrhagic_mean	2.61e+01	3.15e+00	2.63e+01	3.5e+00	True
stroke_fatality_rate _haemorrhagic_std	1.24e+01	3.89e+00	1.23e+01	4.3e+00	False
stroke_fatality_rate _haemorrhagic_min	2.58e+00	2.91e+00	2.67e+00	3.4e+00	True
stroke_fatality_rate _haemorrhagic_25%	1.92e+01	2.82e+00	1.94e+01	3.1e+00	True
stroke_fatality_rate _haemorrhagic_50%	2.53e+01	3.63e+00	2.55e+01	4.1e+00	True
stroke_fatality_rate _haemorrhagic_75%	3.17e+01	4.41e+00	3.20e+01	5.0e+00	True
stroke_fatality_rate _haemorrhagic_max	6.84e+01	1.64e+01	6.85e+01	1.8e+01	False
stroke_fatality_rate _ischemic_mean	1.28e+01	1.63e+00	1.28e+01	1.7e+00	False
stroke_fatality_rate _ischemic_std	6.98e+00	3.46e+00	6.86e+00	3.6e+00	True

stroke_fatality_rate _ischemic_min	3.01e+00	2.16e+00	3.01e+00	2.3e+00	False
stroke_fatality_rate _ischemic_25%	9.25e+00	1.68e+00	9.25e+00	1.7e+00	False
stroke_fatality_rate _ischemic_50%	1.18e+01	1.48e+00	1.18e+01	1.5e+00	False
stroke_fatality_rate _ischemic_75%	1.45e+01	1.69e+00	1.45e+01	1.8e+00	False
stroke_fatality_rate _ischemic_max	4.27e+01	1.91e+01	4.19e+01	2.0e+01	True
otorrino_acts_count	1.52e+02	6.97e+01	1.49e+02	7.2e+01	True
otorrino_acts_mean	9.65e+01	4.24e+01	9.51e+01	4.3e+01	True
otorrino_acts_std	1.16e+02	5.91e+01	1.14e+02	6.1e+01	True
otorrino_acts_min	2.88e+00	1.92e+00	2.91e+00	2.4e+00	False
otorrino_acts_25%	1.43e+01	5.25e+00	1.42e+01	5.9e+00	False
otorrino_acts_50%	4.99e+01	2.32e+01	4.93e+01	2.4e+01	True
otorrino_acts_75%	1.17e+02	5.11e+01	1.16e+02	5.6e+01	False
otorrino_acts_max	5.72e+02	2.74e+02	5.66e+02	3.0e+02	False
otorrino_requests_count	1.52e+02	6.97e+01	1.49e+02	7.2e+01	True
otorrino_requests_mean	5.80e+01	2.33e+01	5.71e+01	2.3e+01	True
otorrino_requests_std	6.82e+01	3.33e+01	6.70e+01	3.4e+01	True
otorrino_requests_min	2.31e+00	1.57e+00	2.35e+00	2.0e+00	False
otorrino_requests_25%	1.03e+01	3.83e+00	1.02e+01	4.4e+00	False
otorrino_requests_50%	3.03e+01	1.31e+01	3.00e+01	1.3e+01	False
otorrino_requests_75%	7.03e+01	2.95e+01	7.01e+01	3.0e+01	False
otorrino_requests_max	3.27e+02	1.53e+02	3.23e+02	1.7e+02	False
schoooling_less_ciclo_1_1985	2.86e+01	4.21e+01	2.57e+01	4.2e+01	True
schoooling_ciclo_1_1985	1.11e+03	1.63e+03	9.96e+02	1.6e+03	True
schoooling_ciclo_2_1985	1.87e+02	2.87e+02	1.66e+02	2.9e+02	True
schoooling_ciclo_3_1985	4.05e+02	7.03e+02	3.55e+02	7.0e+02	True
schoooling_secundario _plus_1985	1.63e+02	3.01e+02	1.42e+02	3.0e+02	True
schoooling_superior_1985	2.02e+02	3.96e+02	1.74e+02	3.9e+02	True
schoooling_less_ciclo_1_2018	2.30e-01	7.59e-01	1.91e-01	7.6e-01	True
schoooling_ciclo_1_2018	1.90e+02	1.60e+02	1.80e+02	1.7e+02	True
schoooling_ciclo_2_2018	2.36e+02	1.68e+02	2.28e+02	1.8e+02	True
schoooling_ciclo_3_2018	5.16e+02	3.72e+02	4.85e+02	4.0e+02	True
schoooling_secundario _plus_2018	7.34e+02	6.69e+02	6.68e+02	6.9e+02	True
schoooling_superior_2018	9.70e+02	1.26e+03	8.58e+02	1.3e+03	True
age_group_total_1960	1.06e+05	1.41e+05	9.59e+04	1.4e+05	True
age_group_ages_0_14_1960	2.64e+04	2.65e+04	2.45e+04	2.7e+04	True
age_group_ages_15_64_1960	7.14e+04	1.02e+05	6.39e+04	1.0e+05	True
age_group_ages_65_plus_1960	8.39e+03	1.31e+04	7.55e+03	1.3e+04	True
age_group_total_2011	1.39e+05	9.89e+04	1.29e+05	1.1e+05	True
age_group_ages_0_14_2011	2.02e+04	1.36e+04	1.89e+04	1.5e+04	True

age_group_ages_15_64_2011	9.25e+04	6.38e+04	8.63e+04	6.9e+04	True
age_group_ages_65_plus_2011	2.65e+04	2.26e+04	2.43e+04	2.3e+04	True
aging_ratio_2001	1.06e+02	3.41e+01	1.06e+02	3.9e+01	False
aging_ratio_2019	1.69e+02	3.70e+01	1.74e+02	4.4e+01	True
company_salary_7_1985	1.01e+02	3.31e+01	1.01e+02	3.7e+01	False
company_salary_11_1985	1.19e+02	2.05e+01	1.17e+02	2.4e+01	True
company_salary_14_1985	1.36e+02	2.54e+01	1.34e+02	2.8e+01	True
company_salary_16_1985	1.33e+02	2.73e+01	1.31e+02	3.1e+01	True
company_salary_18_1985	1.43e+02	2.32e+01	1.43e+02	2.5e+01	True
company_salary_19_1985	1.38e+02	2.46e+01	1.37e+02	2.7e+01	True
company_quantity_0_2009	3.67e+02	5.37e+02	3.13e+02	5.3e+02	True
company_turnover_0_2009	5.61e+05	1.71e+06	4.47e+05	1.7e+06	True
company_quantity_1_2009	2.87e+03	2.95e+03	2.58e+03	3.1e+03	True
company_turnover_1_2009	3.29e+05	7.01e+05	2.68e+05	6.9e+05	True
company_quantity_2_2009	6.43e+02	8.71e+02	5.55e+02	8.7e+02	True
company_turnover_2_2009	5.17e+04	1.01e+05	4.18e+04	1.0e+05	True
company_quantity_3_2009	2.75e+03	3.62e+03	2.40e+03	3.6e+03	True
company_turnover_3_2009	3.89e+05	9.35e+05	3.17e+05	9.2e+05	True
company_quantity_4_2009	1.72e+03	1.84e+03	1.54e+03	1.9e+03	True
company_turnover_4_2009	1.61e+05	2.57e+05	1.39e+05	2.6e+05	True
company_quantity_5_2009	6.34e+02	9.59e+02	5.49e+02	9.6e+02	True
company_turnover_5_2009	1.59e+05	3.15e+05	1.35e+05	3.1e+05	True
company_quantity_6_2009	2.65e+02	2.04e+02	2.47e+02	2.1e+02	True
company_turnover_6_2009	1.41e+04	1.67e+04	1.46e+04	1.9e+04	True
company_quantity_8_2009	1.18e+03	1.10e+03	1.08e+03	1.1e+03	True
company_turnover_8_2009	2.09e+05	3.54e+05	1.77e+05	3.5e+05	True
company_quantity_9_2009	1.35e+01	1.04e+01	1.22e+01	1.1e+01	True
company_turnover_9_2009	4.51e+04	5.83e+04	3.90e+04	5.9e+04	True
company_quantity_10_2009	3.74e+03	3.15e+03	3.46e+03	3.3e+03	True
company_turnover_10_2009	2.85e+06	4.60e+06	2.40e+06	4.5e+06	True
company_quantity_11_2009	1.27e+03	8.49e+02	1.20e+03	9.3e+02	True
company_turnover_11_2009	7.55e+05	8.94e+05	6.55e+05	9.1e+05	True
company_quantity_12_2009	1.05e+03	8.30e+02	9.69e+02	8.8e+02	True
company_turnover_12_2009	4.46e+04	7.52e+04	3.82e+04	7.5e+04	True
company_quantity_13_2009	1.70e+01	3.29e+01	1.41e+01	3.3e+01	True
company_turnover_13_2009	6.11e+05	2.19e+06	4.92e+05	2.1e+06	True
company_quantity_15_2009	8.06e+00	8.55e+00	7.66e+00	9.2e+00	True
company_turnover_15_2009	2.48e+03	5.64e+03	2.46e+03	6.0e+03	False
company_quantity_16_2009	8.65e+02	5.73e+02	8.35e+02	6.4e+02	True
company_turnover_16_2009	1.08e+06	1.43e+06	9.82e+05	1.4e+06	True
company_quantity_17_2009	1.06e+03	1.09e+03	9.51e+02	1.1e+03	True
company_turnover_17_2009	3.47e+04	5.10e+04	3.04e+04	5.1e+04	True
company_quantity_19_2009	1.88e+04	1.84e+04	1.70e+04	1.9e+04	True
company_turnover_19_2009	7.76e+06	1.41e+07	6.56e+06	1.4e+07	True
company_quantity_20_2009	3.50e+02	4.36e+02	3.18e+02	4.4e+02	True
company_turnover_20_2009	4.31e+05	1.00e+06	3.52e+05	9.8e+05	True

company_salary_7_2018	7.46e+02	1.53e+02	7.49e+02	1.6e+02	False
company_salary_11_2018	8.50e+02	1.39e+02	8.34e+02	1.4e+02	True
company_salary_14_2018	9.87e+02	1.74e+02	9.70e+02	1.8e+02	True
company_salary_16_2018	1.01e+03	2.07e+02	9.91e+02	2.2e+02	True
company_salary_18_2018	9.36e+02	1.53e+02	9.17e+02	1.6e+02	True
company_salary_19_2018	9.51e+02	1.49e+02	9.34e+02	1.5e+02	True
company_quantity_0_2019	5.27e+02	7.78e+02	4.50e+02	7.8e+02	True
company_turnover_0_2019	5.50e+05	1.49e+06	4.45e+05	1.5e+06	True
company_quantity_1_2019	3.36e+03	3.47e+03	3.03e+03	3.6e+03	True
company_turnover_1_2019	4.34e+05	8.59e+05	3.59e+05	8.5e+05	True
company_quantity_2_2019	7.76e+02	1.03e+03	6.77e+02	1.0e+03	True
company_turnover_2_2019	9.38e+04	1.83e+05	7.80e+04	1.8e+05	True
company_quantity_3_2019	2.85e+03	3.81e+03	2.50e+03	3.8e+03	True
company_turnover_3_2019	4.62e+05	1.02e+06	3.81e+05	1.0e+06	True
company_quantity_4_2019	1.95e+03	1.99e+03	1.76e+03	2.1e+03	True
company_turnover_4_2019	2.28e+05	3.40e+05	1.93e+05	3.4e+05	True
company_quantity_5_2019	1.16e+03	1.84e+03	9.98e+02	1.8e+03	True
company_turnover_5_2019	2.63e+05	5.59e+05	2.19e+05	5.5e+05	True
company_quantity_6_2019	5.88e+02	3.80e+02	5.68e+02	4.0e+02	True
company_turnover_6_2019	2.31e+04	3.14e+04	2.35e+04	3.4e+04	False
company_quantity_8_2019	1.82e+03	2.45e+03	1.60e+03	2.5e+03	True
company_turnover_8_2019	3.69e+05	6.61e+05	3.11e+05	6.6e+05	True
company_quantity_9_2019	1.61e+01	1.41e+01	1.48e+01	1.5e+01	True
company_turnover_9_2019	6.21e+04	7.32e+04	5.47e+04	7.5e+04	True
company_quantity_10_2019	2.98e+03	2.46e+03	2.76e+03	2.6e+03	True
company_turnover_10_2019	3.30e+06	5.26e+06	2.82e+06	5.1e+06	True
company_quantity_11_2019	9.98e+02	6.83e+02	9.39e+02	7.4e+02	True
company_turnover_11_2019	4.44e+05	4.89e+05	3.87e+05	5.0e+05	True
company_quantity_12_2019	9.41e+02	7.72e+02	8.63e+02	8.1e+02	True
company_turnover_12_2019	4.85e+04	9.00e+04	4.07e+04	8.9e+04	True
company_quantity_13_2019	5.97e+01	6.10e+01	5.47e+01	6.1e+01	True
company_turnover_13_2019	8.85e+05	2.92e+06	7.06e+05	2.9e+06	True
company_quantity_15_2019	6.23e+00	6.49e+00	5.86e+00	7.0e+00	True
company_turnover_15_2019	1.94e+03	5.15e+03	1.89e+03	5.5e+03	False
company_quantity_16_2019	7.32e+02	4.92e+02	7.06e+02	5.5e+02	True
company_turnover_16_2019	1.44e+06	1.95e+06	1.29e+06	1.9e+06	True
company_quantity_17_2019	9.38e+02	7.34e+02	8.62e+02	7.8e+02	True
company_turnover_17_2019	3.65e+04	5.68e+04	3.22e+04	5.7e+04	True
company_quantity_19_2019	2.03e+04	2.09e+04	1.83e+04	2.1e+04	True
company_turnover_19_2019	9.27e+06	1.69e+07	7.86e+06	1.7e+07	True
company_quantity_20_2019	5.60e+02	6.49e+02	5.01e+02	6.6e+02	True
company_turnover_20_2019	6.09e+05	1.45e+06	4.99e+05	1.4e+06	True
age	4.89e+01	2.98e+02	6.99e+01	1.1e+01	True
question_1_1	2.41e-01	4.28e-01	7.08e-02	2.6e-01	True
question_1_2	1.92e-01	3.94e-01	2.14e-01	4.1e-01	True
question_1_3	2.47e-01	4.31e-01	1.69e-01	3.7e-01	True

question_1_4	1.03e-01	3.04e-01	1.28e-01	3.3e-01	True
question_1_5	1.31e-01	3.37e-01	1.51e-01	3.6e-01	True
question_1_6	7.23e-02	2.59e-01	1.92e-01	3.9e-01	True
question_1_7	1.35e-02	1.15e-01	7.52e-02	2.6e-01	True
question_2_1	2.30e-01	4.21e-01	1.75e-02	1.3e-01	True
question_2_2	7.60e-01	4.27e-01	9.45e-01	2.3e-01	True
question_2_3	9.37e-03	9.63e-02	3.79e-02	1.9e-01	True
question_3_1	4.10e-01	4.92e-01	4.20e-01	4.9e-01	False
question_3_2	1.63e-01	3.69e-01	1.55e-01	3.6e-01	False
question_3_3	3.40e-01	4.74e-01	2.66e-01	4.4e-01	True
question_3_4	4.38e-02	2.05e-01	3.66e-02	1.9e-01	True
question_3_5	4.27e-02	2.02e-01	1.12e-01	3.2e-01	True
question_3_6	6.11e-04	2.47e-02	1.10e-02	1.0e-01	True
question_4_1	9.13e-01	2.81e-01	9.28e-01	2.6e-01	True
question_4_2	6.90e-02	2.53e-01	4.98e-02	2.2e-01	True
question_4_3	1.76e-02	1.31e-01	2.19e-02	1.5e-01	True