UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA E MATEMATICA APPLICATA

Master's Degree in Information Engineering for Digital Medicine

Final project

Natural Computation

Prof. Antonio Della Cioppa
Prof. Angelo Marcelli

Francesco D'Auria 0623200057
Mario Apicella 0623200060

ACADEMIC YEAR 2024/2025

CONTENTS

INTRODUCTION

The overall aim of the project is to propose the development of a supervised classification system based on Natural Computation techniques. Specifically, it focuses on an Evolutionary Computation technique, namely Grammatical Evolution (GE). The scope is to monitor vital signs, which plays a crucial role in the early detection of patient deterioration and in the prevention or management of high-risk clinical diseases. The first chapter is a brief but detailed introduction regarding the utilized dataset, starting from the basics of the diseases that are referred to in its creation, arriving at dataset analysis and preprocessing. The second chapter will cover the development of the model, addressing not only parameter tuning but also grammar choice and implementation details. In the third chapter, model performances are presented along with a detailed explanation of the testing carried out and a comparison with traditional baseline classifiers. Finally, the fourth chapter is meant to provide some potential future additions to this project, along with well-reasoned conclusions and criticisms.

CHAPTER 1 DATASET

Coronary heart disease (CHD) is a major type of Cardiovascular disease. As its name implies, CHD more specifically involves malfunctioning of the heart itself. Nevertheless, the causes of the malfunctioning can often be traced elsewhere. For example, over time, a substance known as plaque can build up within the walls of the arteries leading to the heart. Eventually, these arteries can become blocked to such an extent that the heart can no longer function, resulting in what is known as a myocardial infarction (or heart attack).

The Framingham Heart Study dataset was developed as part of a long-term cardiovascular cohort study [1], initiated in 1948 in Framingham, Massachusetts. The dataset was created to identify common factors and characteristics that contribute to cardiovascular disease. It includes demographic, clinical, and laboratory data collected over several decades, aimed at predicting the risk of CHD within 10 years and related conditions.

1.1 Dataset details

The dataset collects precisely 4240 samples to which 16 features (including target) are associated. The target value is the variable `TenYearCHD`, which indicates whether a patient developed Coronary Heart Disease (CHD) within the next ten years. The population of interest is made up of individuals (30 to 74 years old) without overt CHD at the baseline examination. The target value is a binary variable where 0 means that the patient did not experience CHD within ten years, with 1 being the opposite. Demographic, clinical, and lifestyle features such as age, sex, blood pressure (systolic/diastolic), cholesterol levels, BMI, smoking habits, heart rate, and glucose levels are present. An important note is that the dataset is —highly —imbalanced, as the majority (85%) of samples belong to class 0 (no CHD), while the complementary smaller portion (15%) belongs to class 1 (CHD). In the following report, it is assumed that the Positive class is the minority class (1), as this is a common procedure especially in clinical contexts. This convention not only reflects the fact that the positive class indicates the presence of the disease (CHD), which is the condition of interest, but is also motivated by the clinical importance of minimizing false negatives. If the majority class (0) is considered as the positive class, false negatives would correspond to predicting the disease in individuals who would not actually develop it, which is generally less critical than cases where the predictor fails to identify individuals who will develop CHD, potentially delaying preventive measures or treatments.

1.2 Preprocessing steps

In this section, all preprocessing steps are presented and briefly explained in order of application, with the overall process leading to data ready to be passed to the appropriate models.

1.2.1 Initial analysis

High class imbalance is shown in 1.1; in addition, the Dataset presented some missing values and no duplicated samples. This has to be treated in order for better results through the evolutionary models. One of the first analyses was understanding the relationship between the Continuous variables and the Dependent variable, and this is done through violin plots. The width of each "violin" reflects data density at different values, revealing peaks, valleys, and multimodal patterns that box plots alone might miss, while the height represents the range of values for the data. Only the more interesting are reported in the following figures.

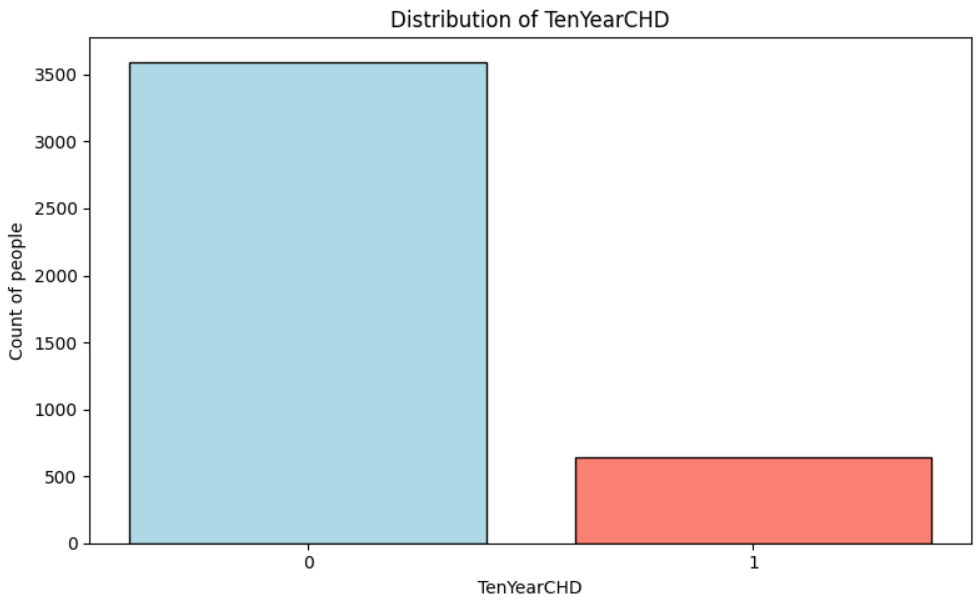


Figure 1.1: Class distribution

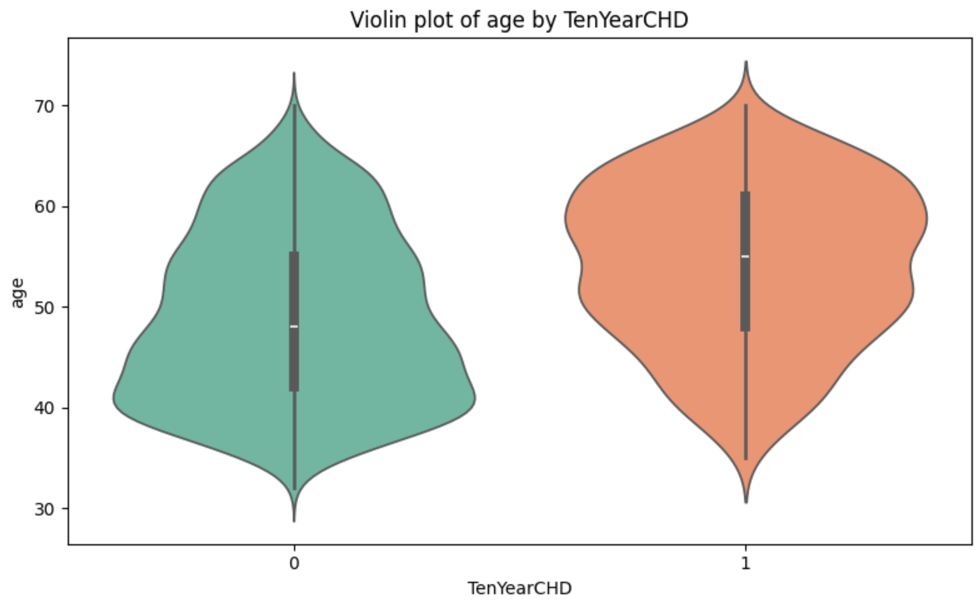


Figure 1.2: Age-TenYearCHD violin plot

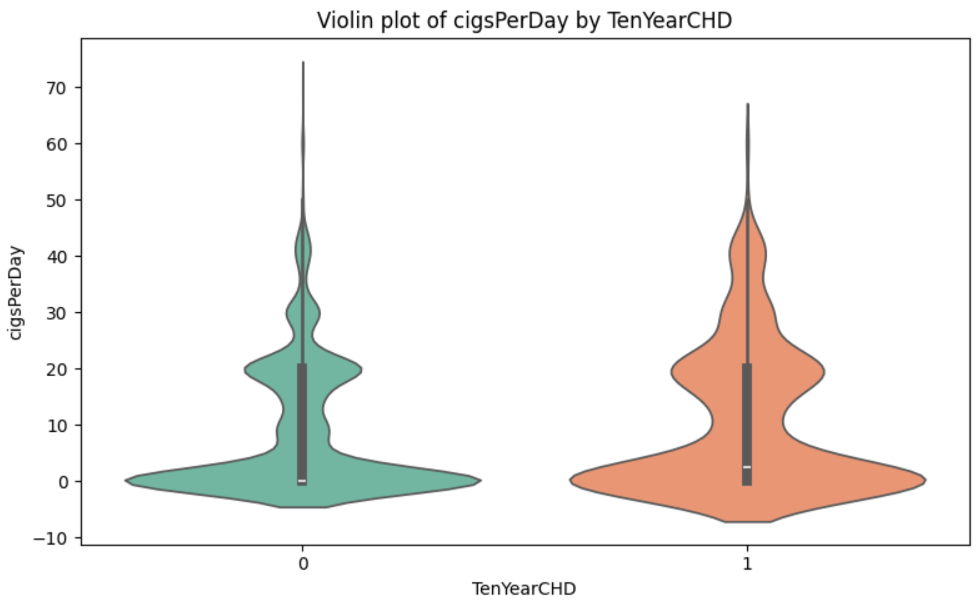


Figure 1.3: cigsPerDay-TenYearCHD violin plot

By just looking at the plots, it can be observed that age is a major factor contributing to risk of CVD, since more number of older patients have a risk of CVD than younger patients. More number of non-smokers are present among those who have no risk of CVD in next ten years.

In opposition, relationship between the Discrete variables and the Dependent variable is provided by both a count plot and a proportion plot, since the distribution is often not clear in the former. This was observed because, although only 40 individuals with diabetes are at risk of CVD compared to 604 individuals without diabetes, considering the proportion within each group reveals a contrast: 36.7% of people with diabetes face a CVD risk, whereas this figure drops to 14.6% among those without diabetes. Such a significant difference is not evident from a simple count plot alone.

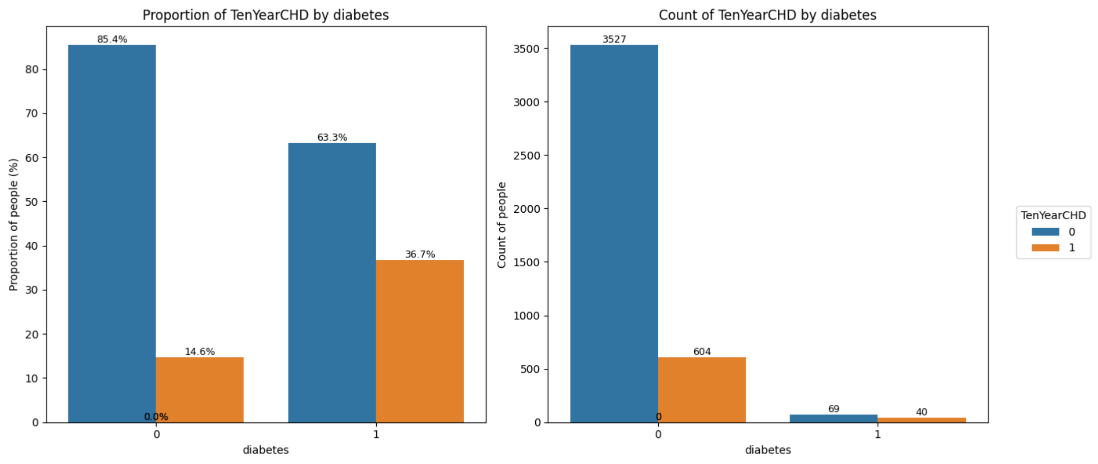


Figure 1.4: diabetes-TenYearCHD count and proportion plot

1.2.2 Hypothesis test

High correlation between some of the independent variables can be observed in the following plot 1.5, like between `sysBP` and `diaBP`, `prevalentHyp` and `sysBP`, and diabetes and glucose. The dimension of the problem can indeed be reduced by feature engineering, and this will be addressed in the upcoming sections.

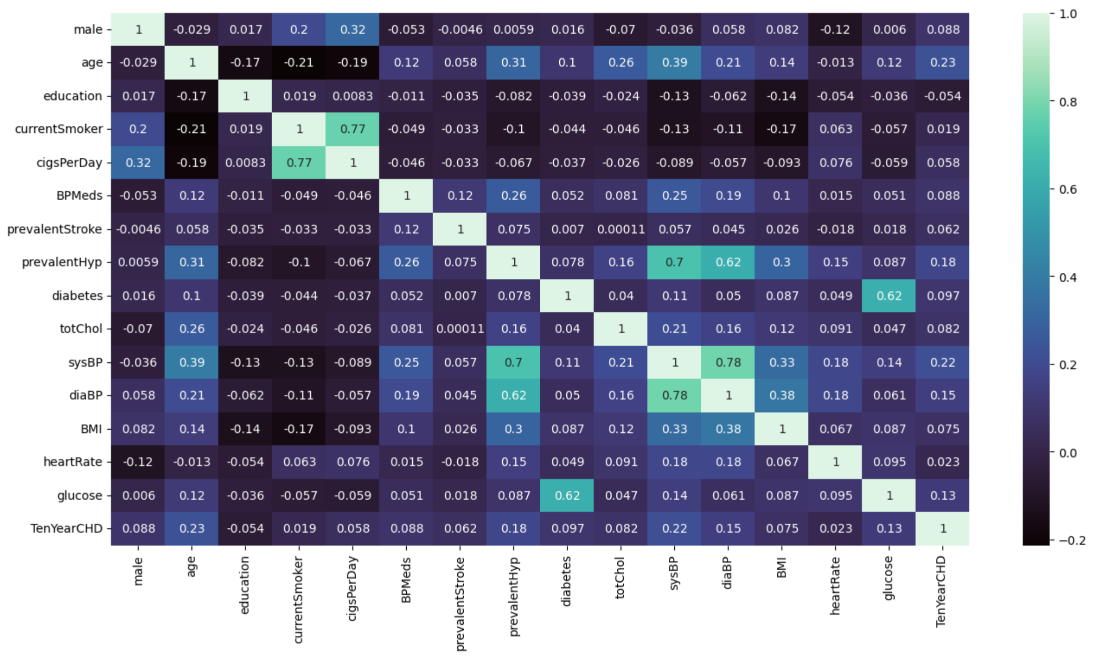


Figure 1.5: Correlation heatmap

The hypothesis test aims to assess whether each categorical variable in the dataset is associated with the target variable, TenYearCHD. This is the only hypothesis test conducted. The hypotheses are defined as:

- **Null Hypothesis (H_0):** Each categorical variable is independent of the TenYearCHD variable;
- **Alternative Hypothesis (H_1):** Each categorical variable is dependent on the TenYearCHD variable.

A chi-squared test of independence is used for this purpose, with a significance level of 0.01, and results are shown within Figure 1.6.

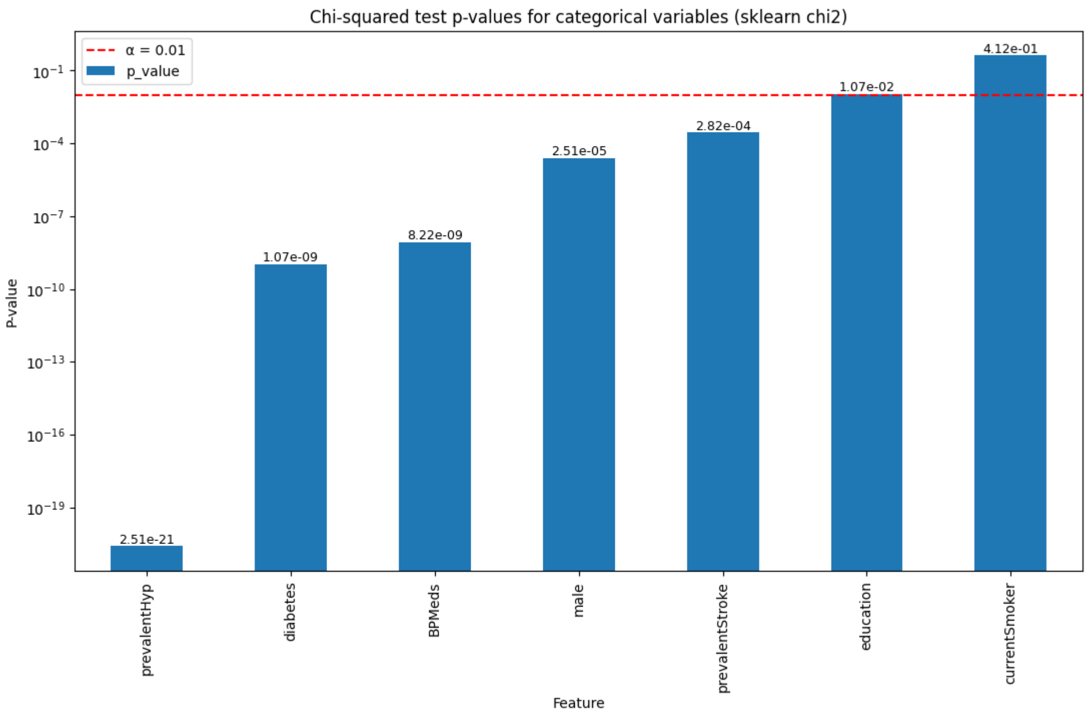


Figure 1.6: Chi-squared test results

For the variables `education` and `currentSmoker`, the p-values were greater than 0.01. This means that the null hypothesis of independence could not be rejected, indicating no significant dependency on the target variable `TenYearCHD`. Those features are going to be dropped from the study later on.

1.2.3 Data imputation

Beside some prior knowledge regarding feature distributions, a missing value imputation was performed since 582 rows (13.73%) contain at least one missing value, for a total of 645.

For the only categorical feature `BPMeds`, the imputation of missing values was performed based on clinical guidelines reported in [2]. Blood pressure medications are recommended for patients with systolic and diastolic blood pressure levels above 140/90 mmHg. For those who have diabetes or kidney disease, the threshold for treatment is lower, with blood pressure levels above 130/80 mmHg.

As for continuous variables, all individuals with missing values in `cigsPerDay` are current smokers. Additionally, non-smokers form the largest group in the dataset. Based on these observations, imputing missing values for `cigsPerDay` using the median of the entire dataset would be inappropriate; instead, the median calculated only from current smokers was utilized.

All the remaining missing values in each variable were imputed with the median of the respective variable. Figures 1.7 and 1.8 are a visual representation of the imputation process. In addition to this step, `prevalentStroke` was removed because of the high class imbalance (only 0.59% of the dataset having 1 as value of this feature).

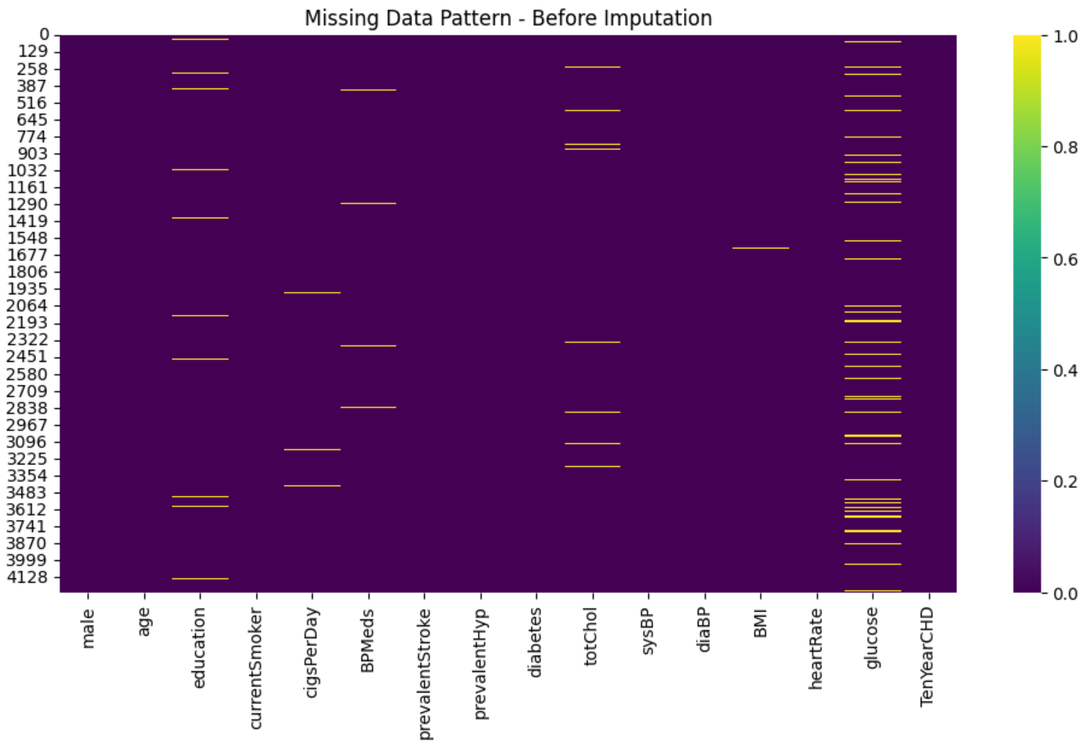


Figure 1.7: Heatmap of missing values before imputation

Missing Data Pattern - After Imputation
male
age
cigsPerDay
BPMeds
prevalentStroke
prevalentHyp
diabetes
totChol
sysBP
diaBP
BMI
heartRate
glucose
TenYearCHD

Figure 1.8: Heatmap of missing values after imputation

1.2.4 Feature Engineering: MAP and Diabetes Grade

To improve the interpretability and reduce multicollinearity within the dataset, two significant feature engineering operations were applied: the computation of Mean Arterial Pressure (MAP) from systolic and diastolic pressures, and the derivation of a categorical variable for glucose-related risk, named `diabetes_grade`.

Mean Arterial Pressure (MAP)

Given the high correlation observed between `sysBP` (systolic blood pressure) and `diaBP` (diastolic blood pressure), a new variable `MAP` was introduced using the clinical formula:

$$MAP = \frac{\texttt{sysBP} + 2 \cdot \texttt{diaBP}}{3} \tag{1.1}$$

MAP represents the average arterial pressure during a single cardiac cycle and is recognized as a more stable and meaningful indicator of perfusion than either systolic or diastolic pressure alone [3].

Further analysis revealed that over **90%** of individuals with a recorded history of hypertension (`prevalentHyp = 1`) exhibited a `MAP` greater than 100mmHg, suggesting a high degree of information redundancy.

Due to the strong overlap between `MAP` and `prevalentHyp`, the latter was deemed redundant and was therefore removed from the dataset.

Diabetes Grade

A similar consideration was applied to the `diabetes` variable. While binary in nature, this variable was found to be highly correlated with `glucose` levels, a continuous indicator more suitable for nuanced modeling.

To maintain interpretability and mitigate issues with outliers in the `glucose` distribution, glucose was converted into a categorical risk score named `diabetes_grade`, following guidelines from the American Diabetes Association [4, 5]:

1. `diabetes_grade = 1`: Hypoglycemia (`glucose` $<$ 70 mg/dL)

2. `diabetes_grade = 2`: Normal (70 mg/dL $\leq$ `glucose` $\leq$ 100 mg/dL)

3. `diabetes_grade = 3`: Pre-diabetes (100 mg/dL $<$ `glucose` ¡ 126 mg/dL)

4. `diabetes_grade = 4`: Diabetes (`glucose` $\geq$ 126 mg/dL)

A violin plot confirmed the discriminatory power of `glucose` across binary `diabetes` classes, with diabetic individuals typically showing much higher values. As a result, `glucose` was transformed and the original `diabetes` column was dropped.

1.2.5 Correlation Analysis

A correlation analysis was conducted and, as shown in Figure 1.9, most pairwise correlations are weak (absolute value below 0.3), indicating low collinearity and a generally independent contribution of variables.

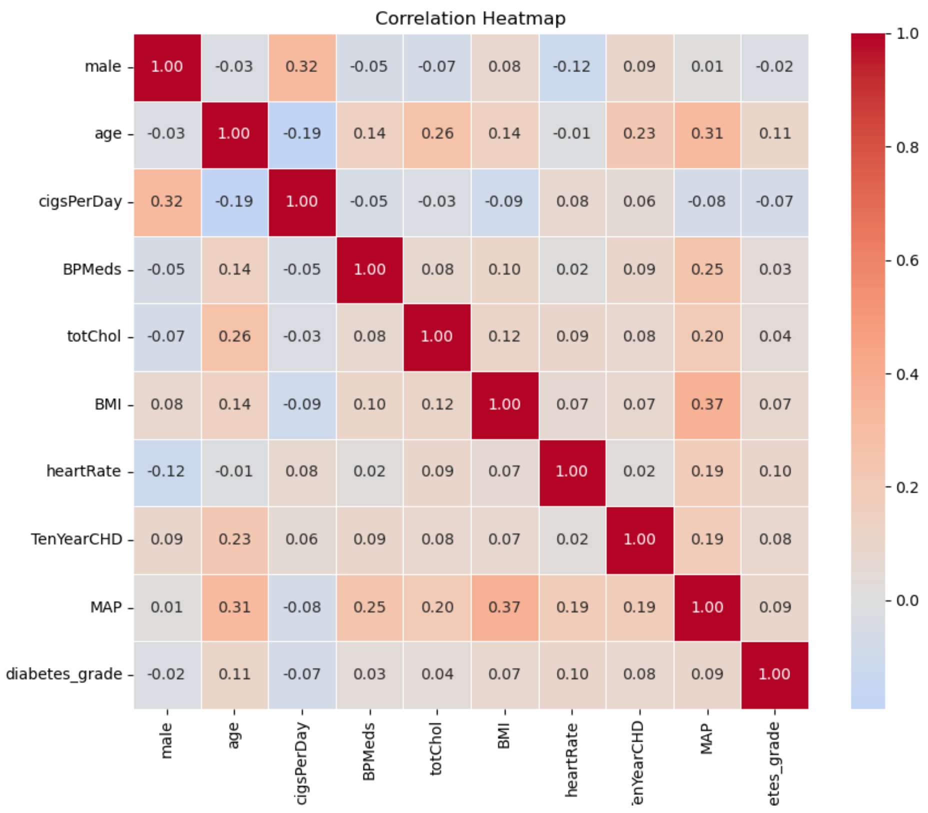


Figure 1.9: Correlation heatmap of final features after preprocessing

The highest observed correlation is between `MAP` and `BMI` ($r = 0.37$). This relationship is physiologically consistent, as individuals with elevated body mass index tend to exhibit increased blood pressure, which directly affects the mean arterial pressure. A moderate correlation ($r = 0.31$) also emerges between `MAP` and `age`, aligning with the expectation that arterial stiffness and pressure tend to increase with age.

Also noteworthy is the correlation between `MAP` and `BPMeds` ($r = 0.25$), supporting the hypothesis that medical treatment for blood pressure correlates with higher MAP values prior to pharmacological control.

Regarding the outcome variable `TenYearCHD`, the strongest correlation is found with `age` ($r = 0.23$), confirming the already stated hypothesis that age is a critical factor in cardiovascular disease.

1.2.6 Train-Test Split and Imbalance Correction

A `train-test split` was performed using a 70/30 proportion, with stratification based on the target label to ensure that both sets reflect the same class imbalance ratio.

The decision to adopt a 70/30 ratio is motivated by a balance between the need for sufficient training data and the necessity of reliable model evaluation—particularly in the context of imbalanced datasets. This is particularly important in contexts where even a small number of false negatives or false positives can have significant implications. Alternative strategies, such as 80/20, may further increase the training data but come at the cost of reduced reliability in test evaluation—especially when the absolute number of positive cases in the test set becomes too small. The previous statement was proved by a comparison between the outcomes of the models with both split proportions (70/30 had better performances than 80/20).

To mitigate the impact of this imbalance, a resampling strategy was adopted. Specifically, **BorderlineSMOTE** was applied to the training set only. This technique extends the Synthetic Minority Oversampling Technique (SMOTE) by focusing the generation of synthetic samples near the decision boundary, where misclassification is more likely. The `'borderline-1'` variant oversamples only those minority instances that are surrounded by majority-class examples.

As a result, the classifier is expected to achieve better generalization performance, particularly in detecting the minority class.

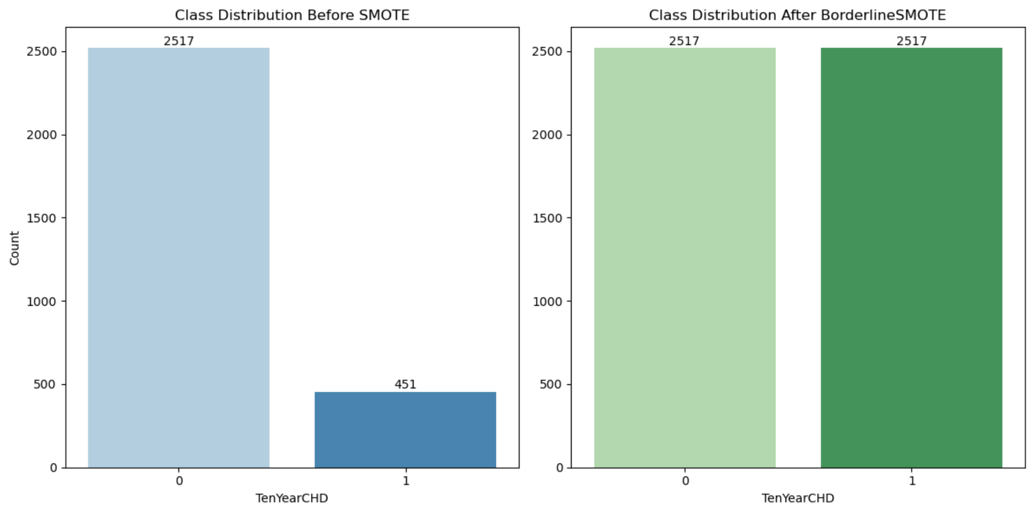


Figure 1.10: Class distribution before (left) and after (right) applying BorderlineSMOTE to the training set

As illustrated in Figure 1.10, the training data initially showed a pronounced imbalance, with only 451 instances of the positive class (`TenYearCHD = 1`) compared to 2,517 of the negative class. After applying BorderlineSMOTE, the two classes were exactly balanced, each containing 2,517 samples. It is important to clarify that these techniques were applied only to the training set. Applying them to the Test Set would artificially balance the class distribution, introducing data that does not reflect reality. In addition, if oversampling is done randomly, some samples could end up in both the Training and Test Sets, compromising the Test Set's role as unseen data.

1.3 A further preprocessing

Finally, another different preprocessing was applied, as there were uncertainties about which technique would be most suitable. This allowed for a comparison of results based on different data modifications. In this case, missing data were imputed using MICE (Multiple Imputation by Chained Equations) [6], implemented via `IterativeImputer`. This technique models each feature with missing values as a function of the other features and iteratively estimates the missing entries. It helps preserve the multivariate relationships among variables, so it was compared with the previous feature specific and median replacement. Even in this case, the features `education` and `currentSmoker` were dropped, but no further modification (i.e. feature engineering) was applied.

CHAPTER 2 MODEL IMPLEMENTATION

As previously stated, this project focuses on Grammatical Evolution (GE), an Evolutionary Computing methodology that evolves a population of individuals through a grammar-based genotype-to-phenotype mapping. In GE, each individual's genotype is represented as a linear string of integers (codons), which is then interpreted through a formal grammar to generate the corresponding phenotype—typically a program or decision rule.

The evolutionary process proceeds over generations through the application of crossover and mutation operators on the genotypes. Selection is driven by the evaluation of the resulting phenotypes according to a predefined fitness function. Depending on the replacement strategy adopted, part or all of the population is substituted at each generation. This iterative process continues until a termination criterion is met, such as reaching a maximum number of generations or achieving a satisfactory fitness level.

2.1 GE Development

PonyGE2 was employed in this project, which is a powerful open source implementation of GE in Python, developed at UCD's Natural Computing Research and Applications group [7]. In addition to providing the characteristic genotype-to-phenotype mapping of GE, this implementation encompasses many features in terms of usable metrics, parameter settings, and flexibility in defining grammars and evolutionary strategies.

2.1.1 Parameters

A notable component of the development process was the careful tuning of GE parameters to guide the evolutionary process and improve model performance. The fitness function adopted was the **weighted F1-score**, particularly suitable for imbalanced classification tasks like ours. This metric ensures that both precision and recall are balanced across classes while accounting for class frequencies, thus avoiding bias toward the majority class.

Before selecting the final configuration, a series of exploratory runs was conducted in which different combinations of parameters (such as population size, tournament size, crossover and mutation probabilities) were tested. The aim was to evaluate their effect on convergence behavior, model interpretability, and classification performance. The configuration used in this project reflects this trade-off between exploration and convergence stability. The complete list can be found in Appendix B.

The population size was set to `1000`, a value large enough to maintain genetic diversity and prevent premature convergence. The number of generations was fixed at `400`, which provided sufficient evolutionary cycles for improvements to occur without excessive runtime. The initial choice was to run for 1000 generations, but as can be seen from Figure 2.1, there was little improvement beyond generation 400. Therefore, for time efficiency, this trade-off was accepted.

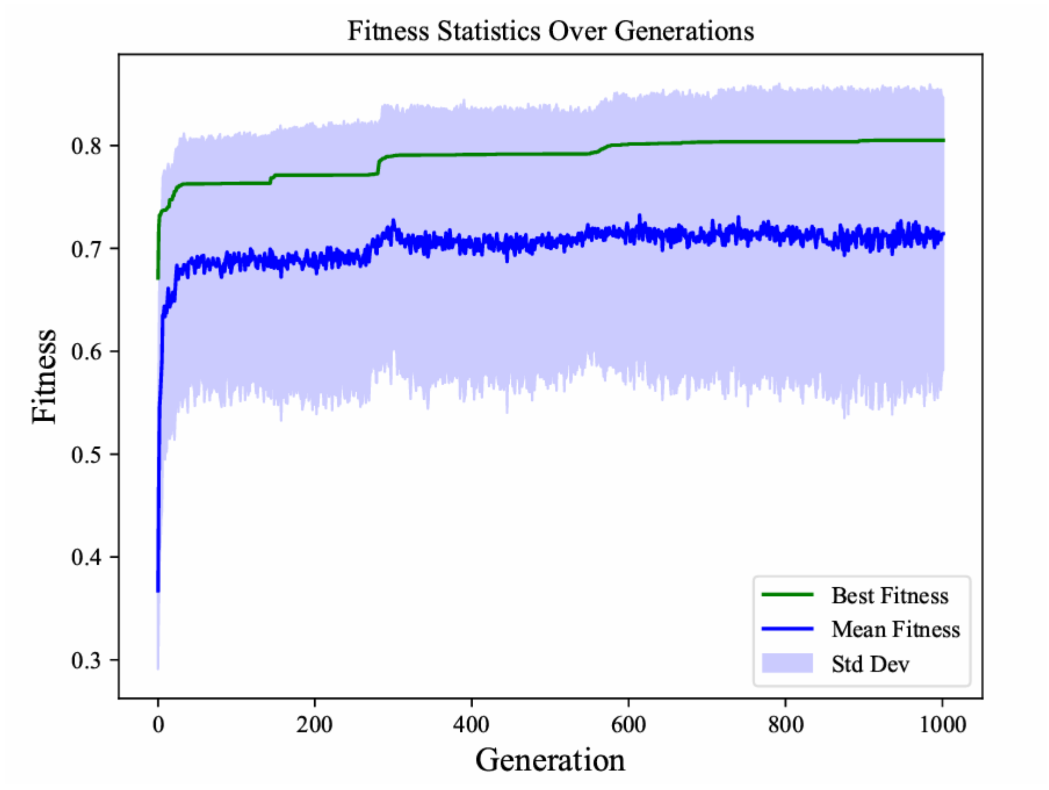


Figure 2.1: Fitness statistics over 1000 Generations

Tree complexity was constrained via `MAX_TREE_DEPTH` (set to 20) to maintain interpretability and avoid code bloat, while other constraints such as `MAX_GENOME_LENGTH` and `CODON_SIZE` were calibrated to allow expressive phenotypes without exceeding memory or runtime limits.

Selection was implemented via **Tournament Selection** with a `TOURNAMENT_SIZE` of 20, corresponding to **2%** of the population. This relatively small tournament size maintains selection pressure while preserving genetic diversity. Larger tournaments increase exploitation but reduce diversity, which may cause premature convergence.

As genetic operators, **subtree crossover** and **integer flip mutation** were adopted: `subtree crossover` was selected because it respects the tree-based structure of phenotypes, enabling the exchange of syntactically valid subtrees between parents and preserving grammatical constraints. `int_flip_per_ind` mutation was chosen for its simplicity and compatibility with codon-based genotypes; one mutation event per individual was allowed, which ensures gradual exploration without excessive disruption.

The replacement strategy was generational, with an `ELITE_SIZE` of 1 to ensure elitism: the best individual in each generation is preserved into the next, avoiding regression in fitness.
All parameters were declared in the `parameters.py` file within the `\src\algorithm\` directory and can be adapted for future runs with minimal modification. For reproducibility and documentation purposes, the complete set of parameters is also reported in Appendix B.

2.1.2 Grammar Design

One of the key advantages of Grammatical Evolution lies in its ability to generate interpretable and executable expressions using a custom-defined grammar. For this project, a domain-specific grammar was designed in `BNF` (Backus–Naur Form), tailored to produce symbolic classification functions for the `TenYearCHD` prediction task. The defined grammar supports recursive construction of `if-then-else` blocks. This enables the evolution of both shallow and nested decision rules, allowing expressive logical structures to emerge from the evolutionary process. These conditional structures are central to the classifier's logic and permit a natural representation of decision boundaries based on combinations of clinical variables.

- **Binary conditions**, such as `x[0] == 1`, corresponding to binary variables (i.e. sex or smoking status);
- **Categorical conditions**, involving only `diabetes_grade`, which encodes glucose-related risk classes: Hypoglycemia (1), Normal (2), Pre-Diabetic (3), and Diabetic (4);
- **Continuous comparisons**, such `x[1] > 130`, allowing thresholds on features like cholesterol or glucose.

Each individual's logic block can express conditions based on the above structures. Logical operators such as conjunction and disjunction allow for composite expressions, while the recursive definition of conditions supports arbitrarily nested decision paths. .

During the early stages of development, alternative versions of the grammar were explored that introduced more sophisticated mathematical constructs. Specifically, they included arithmetic operations such as addition, multiplication, logarithmic and exponential transformations, and protected division. The rationale behind these additions was to extend the expressive power of the search space and to discover more nuanced relationships between variables. While this theoretically increased the capacity of the model to capture complex patterns, it became evident through empirical observation that such complexity came at a significant cost in terms of interpretability.

The grammar structure used in the final version of this project constrained the search space in a way that naturally guided evolution toward functionally valid, interpretable, and potentially actionable rules. The complete grammar is reported in Appendix A, and was loaded by the GE engine to perform the genotype-to-phenotype mapping during evolution.

Example of Evolved Individual

To illustrate the type of solutions produced by the Grammatical Evolution process, an example of a fully evolved individual expressed in Python code is provided. This function was automatically generated using the domain-specific grammar described in Appendix A, and selected based on its weighted F1-score during the evolutionary search.

```python
def CHD(x):
    if x[1] < 57:
        if ( x[2] < 5.15 or x[1] < 48 ):
            if x[3] == 0:
                return False
            else:
                if ( x[6] < 5.0 or x[7] > 9.67 ):
                    if x[3] == 1:
                        if x[1] > 45:
                            return True
                        else:
                            return False
                    else:
                        if ( x[4] < 86 or x[7] < 97 ):
                            return True
                        else:
                            return True
                else:
                    return True
        else:
            return True
    return True
```

Listing 2.1: Example of an evolved classifier function

Despite the syntactic complexity, the classifier remains interpretable thanks to its strict reliance on threshold-based logic. Each decision is made by comparing input variables against explicit numeric constants.

Although the function contains multiple levels of nesting, each condition can be semantically interpreted in clinical terms. This form of structured logic —although verbose —is significantly more interpretable than opaque black-box models, and allows domain experts to validate or challenge the reasoning behind the predictions.

2.1.3 Fitness Evaluation and Cross-Validation

The evaluation of individuals was performed through a custom fitness class named `TenYearsCHD`, specifically developed for this project. This class extends the `base_ff` structure provided by PonyGE2 and is responsible for compiling, executing, and validating each evolved individual. The goal is to determine how effectively the individual can classify patients as being at risk of developing coronary heart disease (CHD) within ten years.

Each individual's phenotype is first transformed from the internal PonyGE2 representation, using custom markers (`{:` `:}`, `::`, etc.), into valid Python code through a dedicated converter function (`markers_to_python`). The resulting code is expected to define a function with the following structure:

```
def CHD(x):
    if ...:
        return True
    else:
        return False
```

To ensure safety and robustness during execution, if the individual's code causes runtime errors (e.g., division by zero, undefined behavior), it is safely intercepted and penalized.

The fitness value is computed using **5-fold Stratified Cross-Validation**, ensuring that the class distribution is preserved across folds. For each fold, the `CHD(x)` function is applied to all test samples, and predictions are generated.

The fitness metric adopted is the **weighted F1-score**, which balances precision and recall while accounting for class imbalance. This choice was motivated by the strong imbalance observed in the dataset, where positive cases (i.e., patients who developed CHD) are significantly underrepresented. The weighted variant ensures that performance on the minority class contributes proportionally to the final score, reducing the bias toward majority class predictions and encouraging fairer classifiers.

Invalid individuals (e.g., those predicting only one class, failing execution, or lacking a `return` statement) are assigned a fitness of 0. Furthermore, utility functions such as `safe_div` and `log_safe` were defined to avoid computational issues during runtime and are made available in the execution environment.

CHAPTER 3 TESTS AND RESULTS

3.1 Experimental Setup

To assess the robustness, consistency, and impact of preprocessing choices on the evolutionary process, two distinct sets of independent experiments were conducted using Grammatical Evolution. Each set consisted of **5 independent runs**, initialized with different random seeds. The first set was executed on the dataset obtained through the standard preprocessing pipeline (feature-specific and median-based imputation, engineered features), while the second set used the alternative preprocessing version based on MICE imputation and no feature engineering.

Each run employed the same grammar, fitness function, and parameter configuration (as detailed in the previous sections); the only varying factor was the random seed used to control the stochastic components of the algorithm—namely, population initialization, and crossover.

Only one shared seed was used in both groups to allow for direct comparison under controlled randomness. This setup was designed to disentangle the effect of data preparation from that of evolutionary variability. By replicating each scenario across multiple seeds, aggregated statistical indicators such as mean fitness, standard deviation, and convergence profiles were compared.

The results are presented and discussed in the next section.

3.1.1 Results with Primary Preprocessing

Each run was initialized with a distinct random seed, except for one case (seed `330711`) which was deliberately kept identical to allow for a direct comparison.

The table below summarizes the key performance metrics obtained across five runs:

Seed	**Accuracy**	**w. F1-Score**	**Precision**	**Recall**	**AUC**
330711	0.6580	0.7038	0.2370	0.5648	0.6197
570061	0.6533	0.6968	0.2019	0.4352	0.5638
724856	0.6616	0.7050	0.2436	**0.5855**	0.6300
262107	0.6171	0.6693	0.1925	0.4767	0.5595
104356	0.6887	0.7266	0.2469	0.5130	0.6165

Table 3.1: Performance metrics obtained from GE runs using updated preprocessing.

Overall, the results show improved sensitivity (recall) in several runs compared to the secondary preprocessing, especially in seeds 330711 and 724856. These runs achieved recall values above 0.56, while preserving moderate precision. This highlights that the new preprocessing pipeline better supports the identification of high-risk individuals (positive class), which is particularly relevant in the clinical context of CHD prediction. The average F1-score across all runs also remains consistently high, indicating balanced predictive performance even with the revised input features.

From a clinical standpoint, the increase in recall observed with the updated preprocessing may be interpreted as a positive outcome, given that identifying more at-risk individuals (even at the expense of a higher false positive rate) is preferable in preventive cardiology.

In this set of runs, the best individual achieved an **F1-score of 0.705**, the highest recall across all updated runs (58.55%), and an AUC of 0.630, decently detecting high-risk CHD cases while maintaining balanced performance overall.

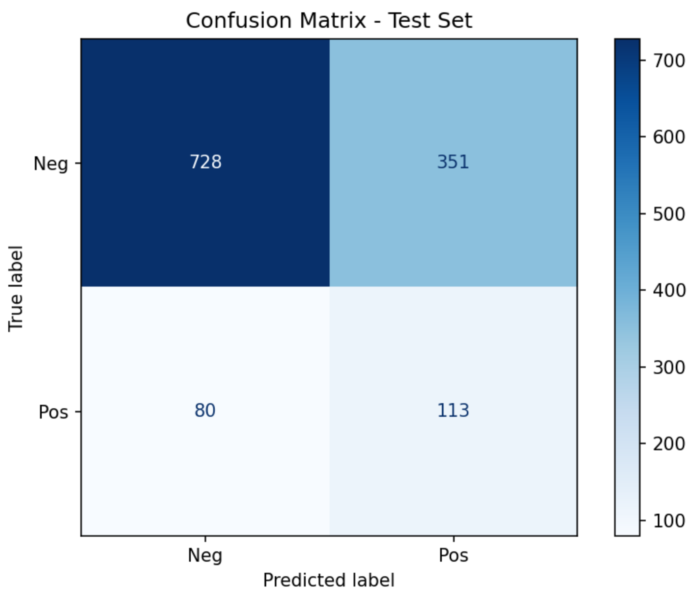


Figure 3.1: Confusion matrix on Test Set for the best-performing individual

As shown in Figure 3.1, the classifier correctly predicted **113 out of 193** positive samples and **728 out of 1079** negative samples. Compared to other configurations, this result highlights a significant improvement in recall while keeping false positives at a clinically acceptable level.

The evolution of the best fitness across generations is reported in Figure 3.2. The fitness increased steadily throughout the training, reaching a final value of approximately **0.705**.

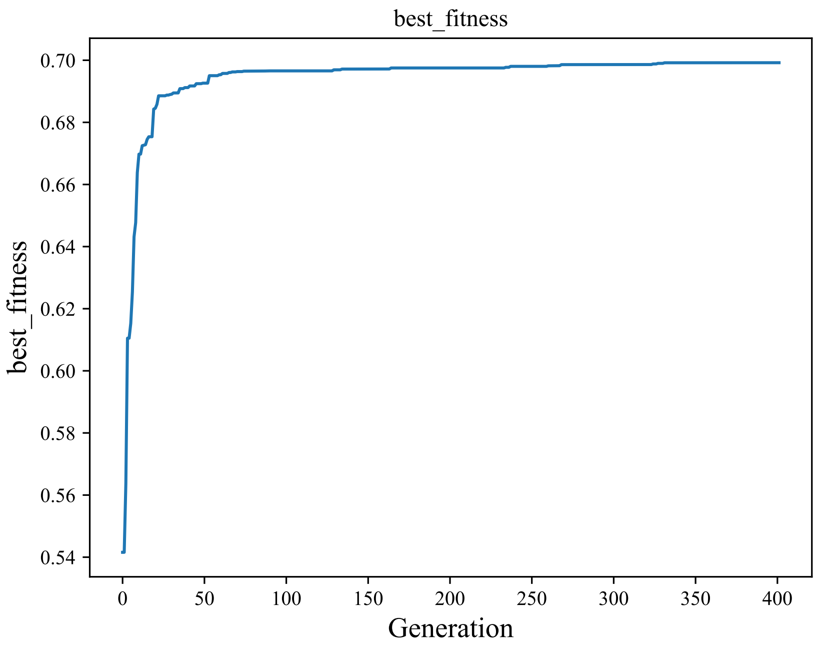


Figure 3.2: Best fitness value per generation for seed 724856

Phenotype of the Best Individual

Below is the Python representation of the best evolved phenotype in the run. The structure remains interpretable and consistent with rule-based decision-making, relying on conditional checks over clinical variables:

```python
def CHD(x):
if ((x[3] == 0 and ((x[0] == 0 or x[1] < 32) or x[5] < 37.0))
or (x[3] == 1 and (x[3] == 1 or x[3] == 1))):
    if (((x[0] == 0 or x[0] != 1) or x[3] == 0) or x[3] == 0):
        if ((x[0] == 0 and x[0] == 0) and x[4] > 317):
            return True
        else:
            if ((x[1] < 50 or x[7] < 94.7) and x[1] < 62):
                if ((x[3] == 0 or x[0] == 0) and x[7] > 125):
                    return True
                else:
                    if ((x[1] < 53 or x[7] < 87) and x[5] < 39.7):
                        return False
                    else:
                        return False
return True
```

Listing 3.1: Best evolved classifier (seed 724856)

This function features interpretable boolean logic and threshold-based comparisons across multiple features (e.g., `x[1]`: age, `x[7]`: MAP, `x[4]`: totChol).

3.2 Results with Secondary Preprocessing

This section presents the results obtained from the five best-performing runs executed on the dataset processed using the secondary preprocessing pipeline. Each run was performed with a different random seed (with 330711 being the common one with the primary), ensuring variability in population initialization, crossover, and mutation operations.

Seed	Accuracy	w. F1-Score	Precision	Recall	AUC
330711	0.7358	0.7604	0.2773	0.4611	**0.6231**
953039	**0.8137**	**0.7814**	0.1579	0.0690	0.5037
19152	0.7846	0.7729	0.2282	0.1762	0.5348
104356	0.8129	0.7938	0.3109	0.1917	0.5579
226691	0.7626	0.7719	0.2681	0.3264	0.5835

Table 3.2: Performance metrics for 5 runs using the standard preprocessing pipeline

As shown in Table 3.2, the evolutionary algorithm exhibits consistent performance across multiple seeds in terms of **F1-score**, which ranges from 0.7604 to 0.7938. This suggests a generally stable ability to balance precision and recall across runs, despite the stochastic nature of the system.

Among all conducted experiments with secondary preprocessing, the run initialized with **random seed 953039** produced the most effective and interpretable classifier. This individual achieved the highest F1-score and accuracy (see Table 3.2), demonstrating a strong ability to generalize on the test set.

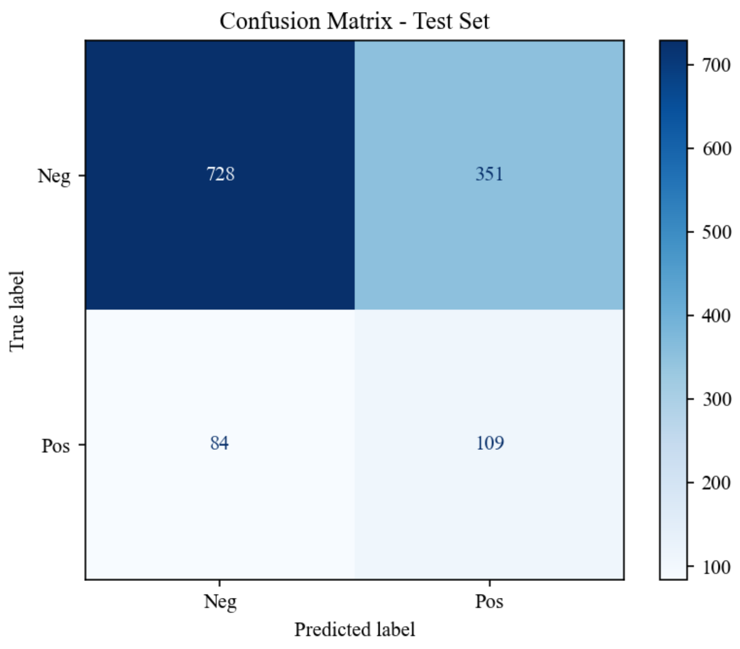


Figure 3.3: Confusion matrix on the test set for the best-performing individual (seed 953039)

As shown in Figure 3.3, the model correctly classified **975 out of 1079 negative samples** and **12 out of 193 positive samples**. Although the number of true positives is low, the model achieved a higher precision compared to other runs, demonstrating cautious and interpretable decision boundaries with fewer false positives.

The evolutionary trajectory of this solution is illustrated in Figure 3.4, which displays the progression of the best fitness across 400 generations.

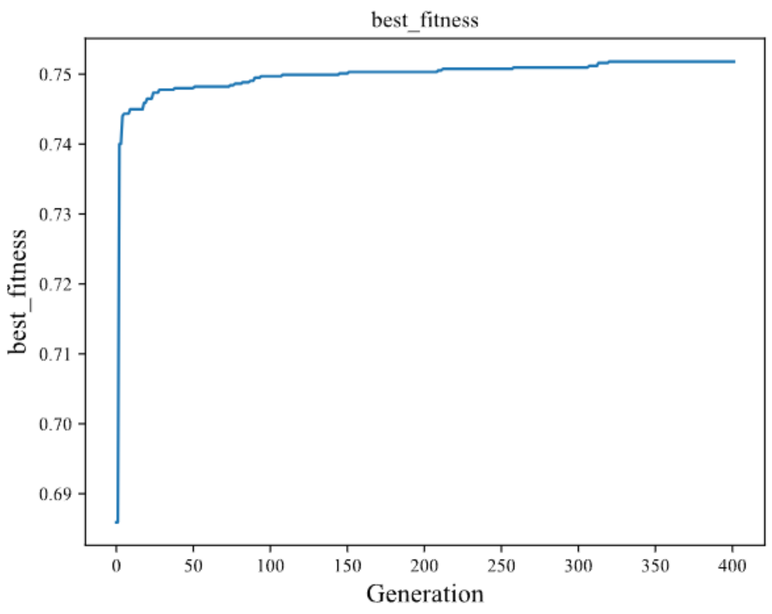


Figure 3.4: Best fitness value per generation

The plot confirms that a consistent improvement was achieved throughout the evolutionary process, with fitness converging steadily toward a plateau (approached **0.75**).

Phenotype of the Best Individual

Below is presented the phenotype (Python code) of the best evolved individual. It uses recursive conditional logic to infer risk of CHD from the input vector `x`, where each index corresponds to a specific clinical feature (e.g., age, MAP, diabetes grade):

```python
def CHD(x):
    if x[5] == 0:
        if (x[5] == 1 or x[0] != 0) or x[5] == 1:
            if x[0] == 1:
                if (x[4] == 1 or x[4] == 0):
                    if (x[11] < 64 or (
                        (x[8] < 154.8 and x[6] == 0) and
                        ((x[3] == 0 or x[5]==0) and x[3]==0)
                    )):
                        return False
                    else:
                        if (x[8]>156.8 or (x[12]<145.8 and x[6]==1)):
                            return False
                        else:
                            if x[7] < 198.5:
                                if x[8] < 139.7:
                                    return False
        else:
            if (x[6] == 0 or x[6] == 1):
                if (x[11] < 64 or x[11] > 95):
                    if x[6] == 0:
                        if (x[8] < 198.5 and (x[5] == 0 and x[5]!=1)):
                            return False
                else:
                    return False
    return True
```

Listing 3.2: Best evolved classifier (seed 104356)

Despite some redundancy in its conditional structure (e.g., repeated checks on the same variable), the logic remains fully interpretable and compatible with clinical reasoning. Features such as `x[5]` (previous Hypertension), `x[6]` (previous diabete) play a central role in the decision boundaries.

3.3 Comparison Between Preprocessing Strategies

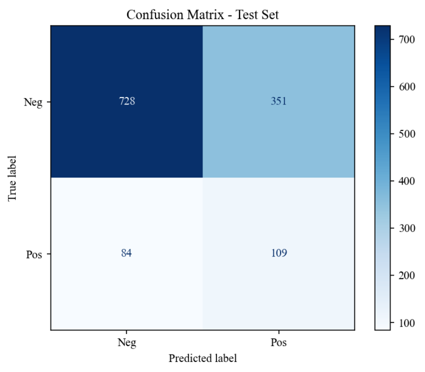
To assess the impact of preprocessing techniques on model performance, we compared the results obtained using the same random seed (`330711`) for both the classic and MICE-imputed datasets. This direct comparison isolates the influence of data imputation and cleaning strategies from stochastic variability. The table below summarizes the performance metrics obtained under the two conditions:

Table 3.3: Performance comparison for seed 330711 under different preprocessing strategies

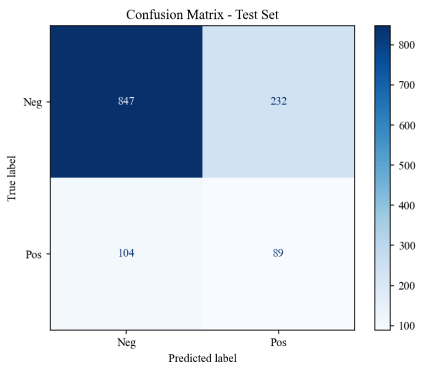
Metric	**Primary Preprocessing**	**Secondary Preprocessing**
Accuracy	0.6580	**0.7358**
F1-Score (Weighted)	0.7038	**0.7604**
Precision	0.2370	**0.2773**
Recall	**0.5648**	0.4611
AUC	0.6197	**0.6231**

Confusion Matrix Comparison

To provide a more intuitive and detailed comparison between the two preprocessing strategies, the respective confusion matrices obtained with the same random seed (`330711`) are shown in Figure 3.5. Both runs used the same Grammatical Evolution configuration, grammar, and fitness function, allowing for a fair evaluation.



(a) Primary Preprocessing



(b) Secondary Preprocessing

Figure 3.5: Confusion matrix comparison

The results show that the primary preprocessing pipeline leads to a significant improvement in the identification of true positives (from 89 to 109), albeit with an increase in false positives (from 232 to 351), with respect to the secondary preprocessing pipeline.

From a clinical perspective, this trade-off is critical; in the context of Coronary Heart Disease (CHD) risk estimation, minimizing false negatives (i.e., failing to identify at-risk patients) is often more important than minimizing false positives, and this is also coherent with what was initially stated in this report. In this scenario, models with higher **recall** are preferred, even if they slightly compromise on precision or accuracy. Therefore, although the secondary preprocessing strategy leads to more balanced metrics, the primary preprocessing is arguably more desirable for clinical deployment, as it improves the model's sensitivity to high-risk patients.

In addition to those conclusion, by looking at the average of those metrics for the two models, it can be observed that by applying the primary preprocessing, recall increases from 0,24488 (secondary processing) to 0.5150 doubling in its value. In opposition to this, accuracy falls from 0.78 (secondary processing) to 0.655 when using the primary.

Nonetheless, if the system is intended to assist clinicians by proving a better response to false negative (as preferred in this project), the primary approach may still be justified; otherwise, the secondary approach will perform better in the opposite scenario. The choice ultimately depends on whether the priority is to maximize **case detection** or to optimize for **decision support efficiency**.

3.4 Comparison with baselines

Finally, a comparison in terms of both interpretability and performances against traditional baseline classificators is presented (only regarding class 1). As a note, only the model designed on the primary preprocessing will be compared. The chosen models are Decision Tree, Random Forest and SVM (RBF kernel), and they can all be visualized under the `\datasets\CHD\baseline` project folder. Specifically, the best seed (724856) is going to participate.

Model	Accuracy	**w. F1-Score**	**Precision**	**Recall**	**AUC**
GE	0.6616	0.7050	0.2436	0.5855	**0.6300**
DT	0.61	0.67	0.22	**0.60**	0.606
RF	**0.78**	**0.78**	0.26	0.25	0.561
SVM-RBF	0.73	0.73	0.13	0.14	0.4883

Table 3.4: Performance metrics for 5 runs using the standard preprocessing pipeline

Based on the results obtained, it can be observed that the Grammatical Evolution (GE) model achieved the best performance in terms of AUC. This indicates a good overall ability to distinguish between the two classes. The Decision Tree (DT), while not matching GE in terms of AUC, stands out with the highest recall value (0.60) among the models considered, showing good sensitivity in detecting positive cases. The Random Forest (RF) achieved the highest accuracy (0.78) and weighted F1-score (0.78), but its relatively low recall (0.25) makes it less suitable if the priority is to identify as many positive cases as possible. Overall, although the Decision Tree offers the best recall, the superior performance of GE on the other metrics, combined with its greater interpretability compared to more complex models like Random Forest or SVM, suggests that GE may represent the most balanced and suitable choice for supporting clinical decision-making.

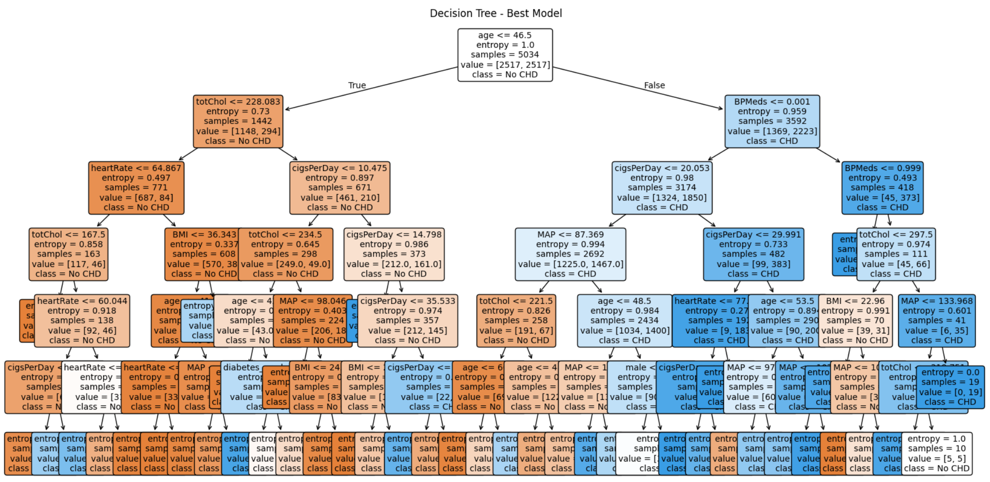


Figure 3.6: Provided Decision Tree

CHAPTER 4 CONCLUSIONS

In this project, we explored the use of Natural Computation techniques, specifically Grammatical Evolution (GE), for the development of an interpretable and effective system for patient risk stratification. Our experiments, conducted on the Framingham Heart Study dataset, demonstrated that GE is capable of generating classification models with good predictive performance, achieving the highest AUC among the tested approaches. This indicates a strong ability to distinguish between classes, an essential characteristic in clinical decision-making.

While the Decision Tree (DT) stood out for its higher recall, making it valuable in contexts where sensitivity is a priority, GE offered a more balanced solution. In particular, it combined competitive accuracy and F1-score with superior interpretability compared to more complex models such as Random Forest or SVM. The predefined grammar used by GE enabled the creation of syntactically correct and, maybe, medical plausible rules, facilitating their understanding and application by healthcare professionals.

Overall, our findings suggest that Grammatical Evolution represents a promising approach for building decision models that are not only accurate but also interpretable, thus supporting real-world clinical workflows. Future work will focus on further validating these results on larger and more diverse datasets, that could also be less imbalanced, and to perform a more finer parameter tuning, that was not performed because of time necessities. The overall model could likely perform better with different parameters than those applied. Different preprocessing techniques also could be tried and validated, in order to better prepare Data for the models.

BIBLIOGRAPHY

[1] *Available FHS Data – Framingham Heart Study*. https://www.framinghamheartstudy.org/fhs-for-researchers/data-available-overview/. Accessed: 2025-06-13. Framingham Heart Study, Boston University & NHLBI, 2025.

[2] "Things you need to know about blood pressure and hypertension". In: *Canadian Journal of Cardiology* 22.7 (May 2006), pp. 601–602.

[3] D. DeMers and D. Wachs. "Physiology, Mean Arterial Pressure". English. In: *StatPearls* (2023). [Updated 2023 Apr 10].

[4] American Diabetes Association. *Diagnosis*. English. https://www.diabetes.org/about-diabetes/diagnosis. Accessed June 15, 2025. 2025.

[5] American Diabetes Association. *Low Blood Glucose (Hypoglycemia)*. English. https://diabetes.org/living-with-diabetes/hypoglycemia-low-blood-glucose. Accessed June 15, 2025. n.d.

[6] Jesper N. Wulff and Linda Ejlskov Jeppesen. "Multiple imputation by chained equations in praxis: Guidelines and review". English. In: *Electronic Journal of Business Research Methods* 15.1 (2017), pp. 41–56. ISSN: 1477-7029.

[7] Michael Fenton et al. "PonyGE2: grammatical evolution in Python". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '17. Berlin, Germany: Association for Computing

Machinery, 2017, pp. 1194–1201. ISBN: 9781450349390. DOI: `10.1145/3067695.3082469`. URL: `https://doi.org/10.1145/3067695.3082469`.

LIST OF FIGURES

APPENDIX A GRAMMATICAL EVOLUTION - CUSTOM GRAMMAR

```
<CHD>          ::= <defp>{::}

<defp>         ::= def CHD(x):{:<if-else>{::}return <TF>:}

<if-else>      ::= if <cond>:{:<if-else>:}
                 | if <cond>:{:<if-else>:}else:{:<if-else>:}
                 | return <TF>

<cond>         ::= <simple_cond>
                | ( <cond> <logic_op> <cond> )

<simple_cond> ::= <bin_cond> | <cont_cond>

<logic_op>     ::= and | or

<bin_cond>     ::= x[<var_bin>] <op_eq> <bit>
<var_bin>      ::= 0 | 3
<op_eq>        ::= == | !=
<bit>          ::= 0 | 1


```

```
<cat_cond>    ::= x[<var_cat>] <op_eq2> <c_value>
<var_cat>     ::= 8
<op_eq2>      ::= == | !=
<c_value>     ::= 1 | 2 | 3 | 4

<cont_cond>   ::= x[<var_cont>] <op_ord> <num>
<var_cont>    ::= 1 | 2 | 4 | 5 | 6 | 7
<op_ar>       ::= + | - | *
<op_ord>      ::= > | <
<power>       ::= 2 | 3 | 4 | 5

<num>         ::= <int> | <dec>
<int>         ::= <digit_no_zero><digit><digit>
                | <digit_no_zero><digit>
                | <digit>
<dec>         ::= <digit_no_zero>.<digit>
                | <digit_no_zero><digit>.<digit>
                | <digit_no_zero>.<digit><digit>
                | <digit_no_zero><digit><digit>.<digit>
<digit>          ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<digit_no_zero> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<TF>          ::= True | False
```

Listing A.1: BNF Grammar for the TenYearCHD Classifier

APPENDIX B EVOLUTIONARY ALGORITHM CONFIGURATION

This appendix reports the full list of evolutionary parameters used in the Grammatical Evolution (GE) experiments conducted with PonyGE2. These values were defined in the `parameters.py` file located in the `\src\algorithm\` directory of the project repository.

Parameter	Value
POPULATION_SIZE	1000
GENERATIONS	400
ERROR_METRIC	f1_score (weighted)
MAX_TREE_DEPTH	20
MAX_TREE_NODES	None
CODON_SIZE	10000
MAX_GENOME_LENGTH	300
INITIALISATION	RHH (Ramped Half-and-Half)
INIT_GENOME_LENGTH	10
MAX_INIT_TREE_DEPTH	15
MIN_INIT_TREE_DEPTH	5
SELECTION	Tournament
TOURNAMENT_SIZE	20 (2% of population)
SELECTION_PROPORTION	0.5
CROSSOVER	operators.crossover.subtree
CROSSOVER_PROBABILITY	0.7
MUTATION	int_flip_per_ind
MUTATION_PROBABILITY	None
MUTATION_EVENTS	1
REPLACEMENT	Generational
ELITE_SIZE	1

Table B.1: Complete list of GE parameters used in the experiments.