Genetic Program to evolve a classifier for Diabetes diagnosis.

Jaimen Govender (u20464348)

University of Pretoria

COS 710 : AI 1

Assignment 1

1.Preface

This report examines Genetic Programming (GP) and how it can be used to evolve a model that can accurately predict whether an individual has Diabetes or not. I demonstrate the effectiveness of GP in improving the predictive performance of my model on the Diabetes dataset. This report aims to provide insight into how I went about using GP to tackle binary classification problems.

2.Data set and data preprocessing

The dataset consists of attributes related to individuals' health, such as pregnancies, glucose levels, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age. The outcome variable indicates whether an individual has diabetes or not. Using this information, we can use the dataset to investigate different features contributing to diabetes and develop predictive model for diabetes diagnosis.

2.1 Handling Missing Values:

Missing values are represented by 0 in the dataset except for pregnancy as individuals are allowed to have not had children before and male individuals cannot get pregnant. The missing values are replaced by the mean value of that attribute in the dataset. The records with 0 were excluded in the mean calculation used to fill in the 0 values.

2.2 Feature Extraction/Selection:

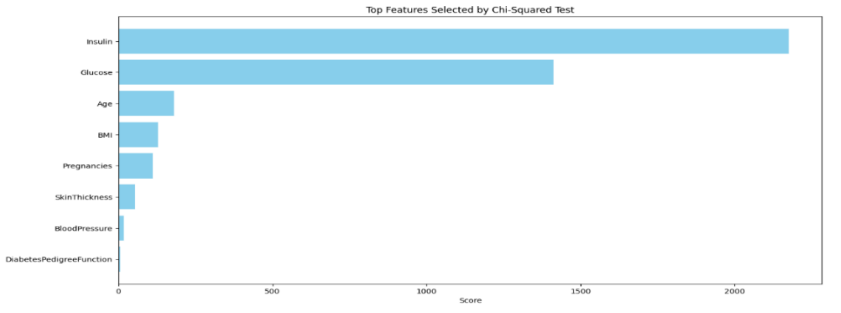

Figure 1: The top features that are most likely to be related to the outcome variable (diabetes) based on their chi-squared scored.

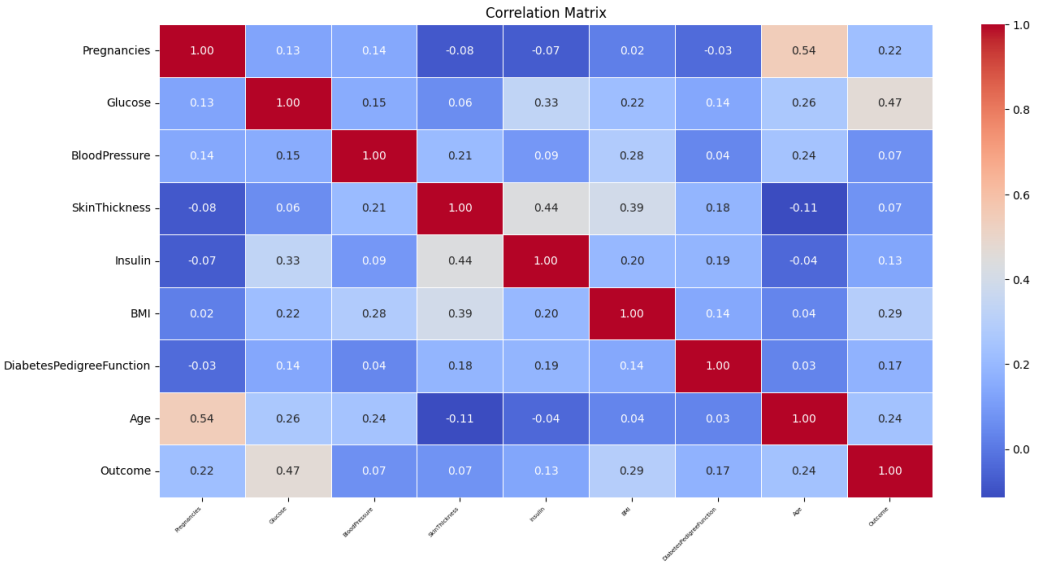

Figure 2: correlation coefficients between pairs of features in the diabetes dataset.

2.3 Exploratory Data Analysis:

	Mean (Non-Diabetic)	Mean (Diabetic)	Standard Deviation (Non-diabetic)	Standard Deviation (Diabetic)	Range (Non-diabetic)	Range (Diabetic)
Pregnancy	3.298	4.866	3.017	3.741	(0.281,6.315)	(1.124,8.607)
Glucose	109.98	141.257	26.141	31.9396	(83.839,136.121)	(109.318,173.197)
Blood Pressure	68.184	70.825	18.063	21.492	(50.121,86.247)	(49.333,92.316)
Skin Thickness	19.664	22.164	14.8899	17.6797	(4.774,34.554)	(4.485, 39.844)
Insulin	68.792	100.36	98.865	138.689	(-30.073,167.657)	(-38.353,239.025)
BMI	30.304	35.143	7.6898	7.263	(22.614,37.994)	(27.8795, 42.406)
Diabetes Pedigree Function	0.4297	0.5505	0.2990	0.3724	(0.131,0.729)	(0.178,0.923)
Age	31.19	37.067	11.6676	10.968	(19.522,42.8576)	(26.0989,48.035)

Figure 3: Exploratory data analysis involved calculating average values for diabetic and non-diabetic individuals, as well as grouping data by the outcome variable to determine mean, standard deviation, and percentile ranges for each attribute.

2.4 Data Splitting:

I made a 70/30 split in the dataset, where 70% is used to train the model and the other 30% is used to test the final model.

3.Tree Structure

The **Decision Tree** structure consists of nodes representing a feature threshold for decision making. At each node, a feature is evaluated against a threshold value, leading to a split (True or False). The tree continuously divides the feature space into smaller subsets until it reaches leaf nodes, which determines the predicted class labels (0 or 1) based on the values of the features.

3.1 Terminal Set

The terminal set includes the final decision points in the tree. They determine whether a patient is predicted to have diabetes (1) or not (0) based on the features observed.

3.2 Function Set

The function set consists of Internal nodes. These internal nodes can be seen as a decision point within the decision tree. These decision points evaluate features from the dataset to determine the path the tree follows. Internal nodes consist of a feature and a threshold value.

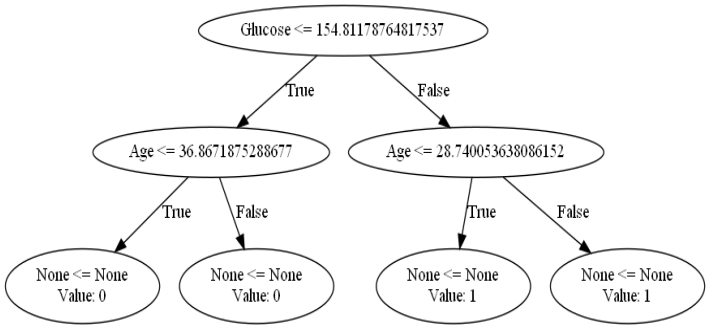

Figure 4: Example Tree with Internal Nodes containing a feature and threshold value and Leaf Nodes with predicted outcome diabetes (1) or not (0).

3.3 Tree Generation

3.3.1 Feature Selection: From the data preprocessing and the correlation matrix above, I have identified the most important factors contributing to diabetes: Glucose, Insulin, BMI, and Skin Thickness. These features stand out as the main influencers in determining an individual's likelihood of being diabetic.

Feature Set = { "Glucose" , "Insulin" , "BMI" , "SkinThickness"}

3.3.2 Threshold Value Calculation: Once a feature is selected, I randomly generate a splitting threshold value between a predefined ranges specific to each feature, calculated in the data preprocessing. The ranges are carefully chosen for each feature to ensure meaningful splits that capture the underlying patterns in the data.

3.3.3 Random Values: To create diversity in my decision trees, I randomly generate values within a certain range for each feature. These random values act as splitting thresholds at each node. This ensures that my decision tree captures various aspects of the data, ensuring a more diverse split of data points.

4.Initial Population Generation

I implemented a method that generates random decision trees with a specified maximum depth constraint, the maximum depth of the trees in my population is 4 (counting from 0). The population generation process involves two methods: "grow" and "full", which are alternated based on the population size (population size = 100). This method ensures a diverse initial population of decision trees. This diversity ensures that the algorithm explores a wide range of potential solutions from the start. With diverse tree structures in the initial population, the algorithm can better understand the

patterns in the data. Although, I was getting similar results with a smaller max depth than 4, I realized that my GP was generalizing the data rather than learning the underlying pattern of the dataset. And a larger depth tends to overfit the training set.

4.1 Full Method:

In the 'full' method, the tree strictly follows a maximum depth limit at each step. Random feature selection and splitting thresholds are still done, but tree growth stops once the maximum depth is reached. All “Full” trees have the **same structure**, as they reach the specified maximum depth without further branching.

4.1 Grow Method:

The "Grow" method is used to create individual decision trees with **varying tree structures**. This method expands the tree until it reaches a maximum depth. At each step of tree expansion, a decision is made whether to continue growing based on the current depth and a random probability condition.
If the current depth exceeds the maximum depth constraint or reaches a depth of at least 3 with a 70% (Grow Rate = 0.7) probability, the tree stops growing, and a leaf node with a randomly assigned value of either 0 or 1 is created. Otherwise, the tree continues to grow by randomly selecting a feature from feature set and determining a random splitting threshold within predefined ranges specific to each feature.

5.Fitness

5.1 Fitness Case: The fitness case is the predicted outcome of the individual in comparison with the actual outcome in the dataset based on the various diabetic attributes for each entry in the dataset.

5.2 Fitness Function: The fitness function evaluates how accurately an individual's predictions match the actual outcomes (diabetic or non-diabetic) in the training dataset.

The fitness case and fitness function play an important role. The fitness case compares predicted outcomes with actual outcomes, ensuring the models can learn in the GP run. The fitness function, accuracy, provides a straightforward measure of predictive performance. This aligns with the goal of correctly identifying diabetic and non-diabetic cases.

6.Selection Methods

6.1 Tournament Selection: Tournament Selection was used as a method for parent selection in my GP. This process involves randomly selecting a group of 8 individuals from the population, evaluating their fitness score and the fittest 2 individuals serve as parents for the next generation. The tournament size parameter determines the number of individuals competing in each tournament (tournament size = 8). If the tournament size is too small, the selection of parents for the next generation is more random leading to premature convergence. If the tournament size is too large, the selection pressure might decrease, resulting in slower convergence or insufficient exploration of the solution space.

7.Genetic Operators

7.1 Reproduction: Reproduction involves selecting 2 individuals from the previous generation as parents through tournament selection to produce offspring. The idea is that by selecting the fittest individuals through tournament selection, we increase the chances of even fitter offspring being produced. This process operates on the principle that crossing over or combining genetic material from these fit parents can produce offspring with a better fitness, enhancing the overall performance of the population over many generations. Reproduction aims to harness the genetic diversity in the population to improve the quality of solutions over time.

7.2 Crossover (Single Point Crossover): Crossover serves as a way for generating offspring by combining genetic material from parent individuals chosen in the tournament selection. This process introduces variation into the population, in hopes that the offspring is produced with improved fitness compared to their parents.

7.2.1 Crossover Process: A random crossover point is selected in both parent trees. These points determine where genetic material will be exchanged between the parents. Subtrees rooted at these crossover points are then swapped between the parents, ensuring an exchange of genetic information. The root nodes and leaf nodes are excluded from selection as crossover points, to preserves the structural integrity of the trees, preventing significant changes in in the overall structure of the tree which could lead to invalid solutions.

I enforce a maximum depth constraint, if this constraint is violated, I prune the resulting trees. This ensures that the offspring trees do not exceed a specified maximum depth, preventing overfitting and overly complex trees. Pruning involves removing unnecessary branches and nodes from the trees, reducing complexity. This process results in more efficient decision trees.

Crossover is used promote diversity within the population and promotes the exploration. This diversity ensures that the population does not converge prematurely to suboptimal solutions and increases the algorithm's ability to explore different regions of the search space.

7.3 Mutation: Mutation is a way of introducing genetic diversity within the population. Every node in the tree can be selected for mutation.

7.3.1 Controlled Mutation Rate: The mutation rate controls the likelihood of mutation occurring. Mutation Rate = 0.8. A higher mutation rate promotes better exploration of the solution space, helps the GP escape local optima, and promotes genetic diversity within the population.

7.3.1 Leaf Node Mutation: If a leaf node is selected for mutation, I invert the value of the leaf node (0 or 1). This mutation allows exploration of different predictions without affecting the tree's structure.

7.3.2 Internal Node Mutation: If an Internal Node is selected for mutation, attributes such as feature, and threshold are mutated. This involves randomly selecting a new feature and threshold, by shifting decision boundaries and improving performance.

7.4 Replacement:

After generating offspring through crossover and mutation, we evaluate their fitness by assessing the offspring's performance on the training data. By comparing the fitness of these offspring with that of the weakest individuals in the current population, we identify candidates for replacement. If the offspring has a higher fitness than the weakest individuals in the current population, I replace the offspring with the weak individuals in the population. This selective replacement refines the population, driving the algorithm towards better solutions while preserving genetic diversity.

8.Parameter Tuning

8.1 Grow Tree Rate: Grow Rate = 0.7. A high grow rate was selected to create many variable tree structures used in the "Grow Method" to create a diverse initial population.

8.2 Number of Generations: Number of Generations = 200. I wanted to give my GP sufficient time to refine the population and converge to a better solution. This larger number of generations ensures exploration of the solution space and increases the chances of finding stronger decision trees.

8.3 Population Size: Population Size = 100. A larger population size enables greater genetic diversity and exploration of a wider range of possible solutions.

8.4 Tournament Size: Tournament Size = 8. If the tournament size is too small, there's a risk of not enough diversity, potentially leading to premature convergence. If the tournament size is too large, the selection pressure might decrease, resulting in slower convergence or insufficient exploration of the solution space.

8.5 Mutation Rate: Mutation Rate = 0.8. A high mutation rate encourages exploration of new genetic material and helps prevent premature convergence by introducing something new into the population.

8.6 Maximum Depth: Maximum Depth of the trees in my population is 4 counting from 0.

9.Experimental Setup

The system used to run the program was an Intel(R) Core (TM) i5-10310U CPU @ 1.70GHz, 2208 MHz, 4 Core(s), 8 Logical Processor(s) with 16 GB RAM, OS 64-bit Windows 11.

A total of 10 experimental trials were conducted with different value seeds. All other experimental parameters have been mentioned above in the report.

10.Results

To evaluate the how accurate my model is, I have employed several performance metrics, such Accuracy, Precision, Recall, and F1-Score.

10.1 Accuracy: Accuracy is the ratio of correctly predicted instances to the total number of instances in the dataset. $Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Instances}$

10.2 Precision: Precision quantifies the ability of the model to correctly identify positive instances among all instances predicted as positive. Precision is calculated as the ratio of true positive instances to the total number of instances predicted as positive. $Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$

10.3 Recall: Recall measures the model's ability to identify all positive instances correctly.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

10.4 F1-Score: It provides a balance between Precision and Recall, and it is useful when the dataset is imbalanced. $F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

GP Run	Seed	Accuracy	Precision	Recall	F1-Score
1	88	74.02597402597402 %	0.64	0.56	0.60
2	100	72.72727272727273 %	0.60	0.65	0.62
3	110	74.02597402597402 %	0.66	0.50	0.57
4	119	73.59307359307358 %	0.65	0.51	0.57
5	129	75.32467532467533 %	0.69	0.51	0.59
6	139	71.86147186147186 %	0.70	0.33	0.44
7	149	71.86147186147186 %	0.60	0.58	0.59
8	160	73.16017316017316 %	0.68	0.43	0.52
9	169	73.16017316017316 %	0.64	0.53	0.58
10	179	72.2943722943723 %	0.63	0.48	0.54
Average		73.39%	0.65	0.51	0.57

11.Runtime

GP Run	Time in Seconds	Time in Minutes
1	331.0637435913086	5.52
2	341.5784766674042	5.69
3	342.3825135231018	5.71
4	318.8273251056671	5.31
5	328.3363080024719	5.47
6	380.4826295375824	6.34
7	365.4826865375824	6.09
8	348.14525747299194	5.80
9	379.7119209766388	6.33
10	365.7168809766388	6.10
Average	348.135	5.87

12.Comparison with previous papers

	My GP	Paper 1	Paper 2	Paper 3
Decision Tree Accuracy	75.32% (run 5)	96.62%	74.78%	65.08%
Decision Tree Precision	0.69 (run 5)	94.02%	70.86%	65%
Decision Tree Recall	0.51 (run 5)	95.45%	88.43%	65%
Decision Tree F1-Score	0.44 (run 5)	94.72%	78.68	65%

My final Decision Tree on average performs similar or even better than the Decision Trees in Paper 2 and Paper 3 in terms of Accuracy, however my Decision Tree is less accurate in than the decision tree used in Paper 1. My Decision Tree also falls short in comparison to the other method used in Paper's 1,2,3 such as Neural Networks, Random Forests.

My Decision Tree model struggles because it doesn't handle different insulin levels well. Low Insulin level or High Insulin level can both cause diabetes (Type 1 or Type 2 Diabetes). My decision Tree fails to identify the patterns correctly between High and Low Insulin Levels resulting in my tree prediction some Insulin cases incorrectly. My Decision Tree fails to capture the patterns that differentiate between these two scenarios. This is the reason for the inaccuracies in my model's, especially in cases where insulin levels deviate from the norm.

However, My Decision Tree gradually refines its solutions over successive generations. Despite its initial shortcomings in accurately involving insulin levels, the model shows signs of progress. With each iteration, it learns from its mistakes and adapts.

12.References and Libraries

12.1 References:

Dufourq, E. (2015). *Data Classification using.* Pietermaritzburg: University of KwaZulu-Natal.

12.2 Libraries:

12.2.1 Pandas(pd): Utilized for data manipulation and analysis. I used pandas for reading, cleaning, and preprocessing the dataset.

12.2.2 Time: Standard Python library used for measuring and reporting the execution time.

12.2.3 OS: Standard Python library used for interacting with the operating system. File Management.

12.3.4 Random: Standard Python library providing functions for generating random numbers and performing random selections.

12.3.5 Graphviz: External library used for visualizing decision trees.