<u>Bazi Mathangeni (19236213)</u>

<u>COS710: Artificial Intelligence</u>

<u>Assignment 2: Genetic Programming for Classification</u>

<u>Description of source and target problems:</u>

The source problem is a binary classification based on predicting if a female individual is diabetic or not *(Outcome: Class variable (0 non-diabetic or 1 diabetic))* based on 8 dependent variables:

- Pregnancies: Number of time pregnant (ranging from 0 to 17)
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test (ranging from 0 to 199)
- BloodPressure: Diastolic blood pressure (mm Hg) (ranging from 0 to 122)
- SkinThickness: Triceps skin fold thickness (mm) (ranging from 0 to 99)
- Insulin: 2-Hour serum insulin (mu U/ml) (ranging from) (ranging from 0 to 846)
- BMI: Body Mass Index (weight in kg/height in m)^2) (ranging from 0 to 67.1)
- DiabetesPedigreeFunction: Diabetes pedigree function (ranging from 0.08 to 2.42)
- Age: Age (years) (ranging from 21 to 81)

The dataset for this problem, (*diabetes.csv*) consists of 768 data elements.

The target problem is also a binary classification problem based on predicting if an individual (male of female) is diabetic or not *(diabetes (0 non-diabetic or 1 diabetic))* based on 8 dependent variables:

- Gender: (male or female)
- Age: (ranging from 0.08 to 80)
- Hypertension: (0 without hypertension or 1 with hypertension)
- Heart_disease: (0 without heart disease or 1 with heart disease)
- Smoking_history: (No Info, never, former, current)
- BMI: (ranging from 10 to 95.7)
- HbA1c_level: (ranging from 3.5 to 9)
- Blood_glucose_level: (ranging from 80 to 300)

The dataset for this problem, *(diabetes_prediction_dataset.csv)* consists of 100 000 valid data elements.

<u>Description of transfer learning employed:</u>

After the initial population computation and 10 genetic algorithm runs of the source problem, the last generations search space area is transferred to supplement the forming of the initial population in the target problem. This is achieved in my solution by passing the functional portion from the last generation of my source problems search space area to the target problem such that, the target problem then only makes the necessary modifications needed on the transferred search space area before evaluating the solution. These necessary modifications include, modifying the terminal set variables such that they match the target problems terminal set values. Furthermore, accounting for the categorical variables that define the difference between the source and target dataset.

A description of the target GP algorithm:

The target problems GP algorithm has the following description:

- Functional Set: (+ , - , x , <, with (+ , - , x) having equal chances of selection in the function nodes of the and (<) being at the root node).
- Terminal Set: (X1 gender, X2 age, X3 hypertension, X4 heart_disease, X5 smoking_history, X6 BMI, X7 HbA1c_level, X8 blood_glucose_level)
- Initial Population: the full method.
- Selection Methods: Fitness Proportionate Selection.
- Genetic Operators: crossover and mutation.
- Number of Generation: 10
- Fitness Functions: Raw Fitness and Accuracy Score

A description of the experimental setup for both the source and target GP algorithm:

The setup of my experiment was as follows:

- The general procedure to run simulations in my GP Algorithm simulations project is running this command from the terminal ***java Main <<Full Tree Height>> <<Number of Files>> <<File Name 1>> <<File Name 2>>***. With ***<<Full Tree Height>>*** being the full tree height parameter, ***<<Number of File>>*** being the number of input files, ***1*** if transfer learning is not employed and only one input file is used, ***2*** will be input in the event where event where the transfer learning is employed with ***<<File Name 1>>*** being the source file name and ***<<File Name 2>>*** being the target file name.

- Ran simulations of the source problem on the source data (***diabetes.csv***) residing on the ***./file*** directory of my submission by running this line ***java Main 5 1 diabetes.csv*** on the terminal from the ***./bin*** directory of my submission.

- Recorded the output results of from the above simulation, i.e.: Raw Fitness, Accuracy Score and Generations execution time over all generations.

- Ran simulations of the source problem on the source data (***diabetes_prediction_dataset.csv***) residing on the ***./file*** directory of my submission by running this line ***java Main 5 1 diabetes_prediction_dataset.csv*** on the terminal from the ***./bin*** directory of my submission.

- Recorded the output results of from the above simulation, i.e.: Raw Fitness, Accuracy Score and Generations execution time over all generations.

- Ran transfer learning implementing simulations with (***diabetes.csv***) being the source problem and (***diabetes_prediction_dataset.csv***) being the target problem by running this line ***java Main 5 2 diabetes.csv diabetes_prediction_dataset.csv*** on the terminal from the ***./bin*** directory of my submission.

- Recorded the output results of from the above simulation, i.e.: Raw Fitness, Accuracy Score and Generations execution time over all generations.

- I ran all these simulations on ***Visual Studio Code*** ide on a 12th Gen Intel(R) Core(TM) i5-1235U 1.30 GHz x64-based processor, 8,00 GB (7,68 GB usable) RAM with 256GB SSD.

The results and a discussion of the results:

Gen		Raw Fitness Average	Raw Fitness Best Value	Accuracy Average	Accuracy Best Value
Init	Source (with and without transfer learning)	45.75	59	39.5	56
	Target (without transfer learning)	68.5	91	66.25	92
	Target (with transfer learning)	49.5	91	48.25	92
1	Source (with and without transfer learning)	51.25	59	46.25	56
	Target (without transfer learning)	49	91	46.75	92
	Target (with transfer learning)	62.25	91	57.5	91
2	Source (with and without transfer learning)	50.25	59	44.75	56
	Target (without transfer learning)	49	91	46.75	92
	Target (with transfer learning)	66.25	91	63.5	92
3	Source (with and without transfer learning)	51.25	58	45.5	55
	Target (without transfer learning)	68.5	91	66.25	92
	Target (with transfer learning)	55.25	91	51.75	92
4	Source (with and without transfer learning)	52.25	58	47.25	55
	Target (without transfer learning)	69.75	91	68.25	92
	Target (with transfer learning)	55.25	91	51.75	92
5	Source (with and without transfer learning)	49.75	58	44	55
	Target (without transfer learning)	49.25	91	46.75	92
	Target (with transfer learning)	50.75	91	49.25	92

6	**Source (with and without transfer learning)**	**49.75**	**58**	**44**	**55**
	Target (without transfer learning)	**61**	**91**	**56**	**92**
	Target (with transfer learning)	**50.5**	**91**	**49**	**92**
7	**Source (with and without transfer learning)**	**53.25**	**58**	**48.25**	**55**
	Target (without transfer learning)	**49**	**91**	**46.75**	**92**
	Target (with transfer learning)	**74.25**	**91**	**71.25**	**92**
8	**Source (with and without transfer learning)**	**52.25**	**63**	**48**	**63**
	Target (without transfer learning)	**69.75**	**91**	**68.5**	**92**
	Target (with transfer learning)	**74.25**	**91**	**71.25**	**92**
9	**Source (with and without transfer learning)**	**54.25**	**64**	**50.75**	**65**
	Target (without transfer learning)	**69.75**	**91**	**68.5**	**92**
	Target (with transfer learning)	**64.75**	**91**	**61.5**	**92**
10	**Source (with and without transfer learning)**	**56.5**	**61**	**52.5**	**59**
	Target (without transfer learning)	**69.75**	**91**	**68.25**	**92**
	Target (with transfer learning)	**51.5**	**91**	**49.5**	**92**
Avg	**Source (with and without transfer learning)**	**51.5**	**59.5**	**46.4**	**57.2**
	Target (without transfer learning)	**54.2**	**91**	**58**	**92**
	Target (with transfer learning)	**59.5**	**91**	**56.77**	**91.91**

<u>Runtime of source and target GP algorithms</u>

Generation	**Source (with and without transfer learning)**	**Target (without transfer learning)**	**Target (with transfer learning)**
Init	172	282	122
1	38	296	151
2	31	276	157
3	27	285	149
4	20	235	121
5	15	216	114
6	13	224	126
7	14	225	121
8	10	169	91
9	13	176	94
10	15	170	95
Average	33.45	232.18	121.91

A comparison in terms of performance, runtimes and computational cost of GP with and without transfer learning:

In terms of performance, runtimes and computational cost, the source problem has proved to perform better, with and without transfer learning than the target problem, without and without transfer learning for all generations other than the initial population. This is a result of the low computational cost the source problem is subject to, 768 data elements than the 100 000 data elements the target problem is subject to. Implementing transfer learning to the target problem results in a better initial population runtime than the source problem regardless of the more than 100 times greater computational cost the target problem is subject to. Thus, this means, even with increased computational cost, transfer learning results in improved runtime and performance. Furthermore, the target problem with transfer learning has proven to outperform the target problem without transfer learning for all generations in the genetic evolution, including the initial population. In terms of raw fitness, on average the target problem with transfer learning performs better than the target problem without transfer learning, by a difference of (59.5-54.2) = 5.3. However, the target problem with transfer learning is outperformed by that without transfer learning in terms of accuracy on average by a difference of (58-56.77) = 1.23. Both these results are better than that of the source problem with and without transfer learning.

References:

java.io.File. (n.d.). Java Platform Standard Edition 8 API Specification. Oracle. https://docs.oracle.com/javase/8/docs/api/java/io/File.html

java.io.FileNotFoundException. (n.d.). Java Platform Standard Edition 8 API Specification. Oracle. https://docs.oracle.com/javase/8/docs/api/java/io/FileNotFoundException.html

java.text.DecimalFormat. (n.d.). Java Platform Standard Edition 8 API Specification. Oracle. https://docs.oracle.com/javase/8/docs/api/java/text/DecimalFormat.html

java.util.Scanner. (n.d.). Java Platform Standard Edition 8 API Specification. Oracle. https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

java.util.Random. (n.d.). Java Platform Standard Edition 8 API Specification. Oracle. https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

java.util.ArrayList. (n.d.). Java Platform Standard Edition 8 API Specification. Oracle. https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html