COS 314 Assignment 3

Tayla Orsmond u21467456

Comparing the Effectiveness of Different Machine Learning Models in Classifying Breast Cancer

Introduction

The goal of this assignment is to design different machine learning models to perform classification of breast cancer. This assignment makes use of a dataset from the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/breast+cancer. The effectiveness of each model is then assessed according to different statistical significance tests.

Environment (Specs)

This assignment was completed on a Dell Inspiron 7490 laptop with the following specs:

Processor: Intel(R) Core (TM) i7-10510U CPU @ 1.80GHz 2.30 GHz

RAM: 16.0 GB (15.8 GB usable)

Development Environment: Visual Studio Code, JDK version 17.0.6

Artificial Neural Network (ANN)

An Artificial Neural Network (ANN) is a mathematical model of the brain that aims to predict patterns within data. ANNs are designed to model how humans learn. In this case, the ANN performs binary classification, i.e., it assigns one of two labels (classes) to a particular instance based on the pattern the ANN recognizes. The classes the ANN can assign in this case are **recurrence-events** or **no-recurrence-events**.

Recurrence-events refers to the patient having cancer and is represented by **a class of 1** in this instance. Similarly, **no-recurrence-events** refers to the patient not having cancer and is given **a class of 0**.

The rest of the dataset was pre-processed for use in the ANN as follows:

Data Transformation and Pre-Processing

This breast cancer domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data.

Domain	# Attributes	# Nominal	# Interval-scaled	# Integer	# Instances
Breast cancer	9 (+ class)	5 (+ class)	3	1	286

Table 1: Summary of Dataset

This dataset includes 286 instances in total, with 201 instances of no-recurrence-events and 85 instances of recurrence-events. The instances are described by 9 attributes, some of which are interval-scaled, and some are nominal. There are four instances with missing values, denoted by "?".

Attribute	Possible Values
class	no-recurrence-events, recurrence-events
age	10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99
menopause	lt40, ge40, premeno

tumor-size	0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59
inv-nodes	0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39
node-caps	yes, no
deg-malig	1, 2, 3
breast	left, right
breast-quad	left-up, left-low, right-up, right-low, central
irradiat	yes, no

Table 2: Attribute Information

As was previously stated, **"recurrence-events" were represented by the class 1, and "no-recurrence-events" were represented by the class 0**. Instances that had **values missing**, had those values **replaced by 0**, in order that they have no effect on the outcome of the ANN's decision making. This was done because these missing values belonged to instances of class "recurrence-events", which already make up a smaller portion of the data, so it was important that these instances were preserved.

The rest of the data was encoded through **Hot-one encoding** (Brownlee, 2020). This type of encoding involves assigning each value of an attribute a unique binary string indicating its position in the sequence. For example, for the node-caps attribute, the value of "yes" is denoted by the string 10, while "no" is denoted by 01. This was done to avoid nominal variables of higher values accidentally having a higher impact on the outcome of the ANN's decision making process (like what might be the case with integer encoding). For example, the age attribute has value ranges from 10-99. If these values were encoded using integer encoding (i.e., 10-19 was given the value 1 and 90-99 was given the value 9), then 90-99 would have a higher impact on results than 10-99 etc. This could otherwise lead to erroneous results or poor performance.

The downside of Hot-one encoding is that there are many inputs to the ANN, which could lead to overfitting as the dimensionality of the data is very large. This is addressed again later.

The data was then **split into a training and testing set by a 70-30% split** respectively. The instances were shuffled and randomly selected to be in either set in order that a good portion of the differences in the domain are represented.

Design Components and Parameters

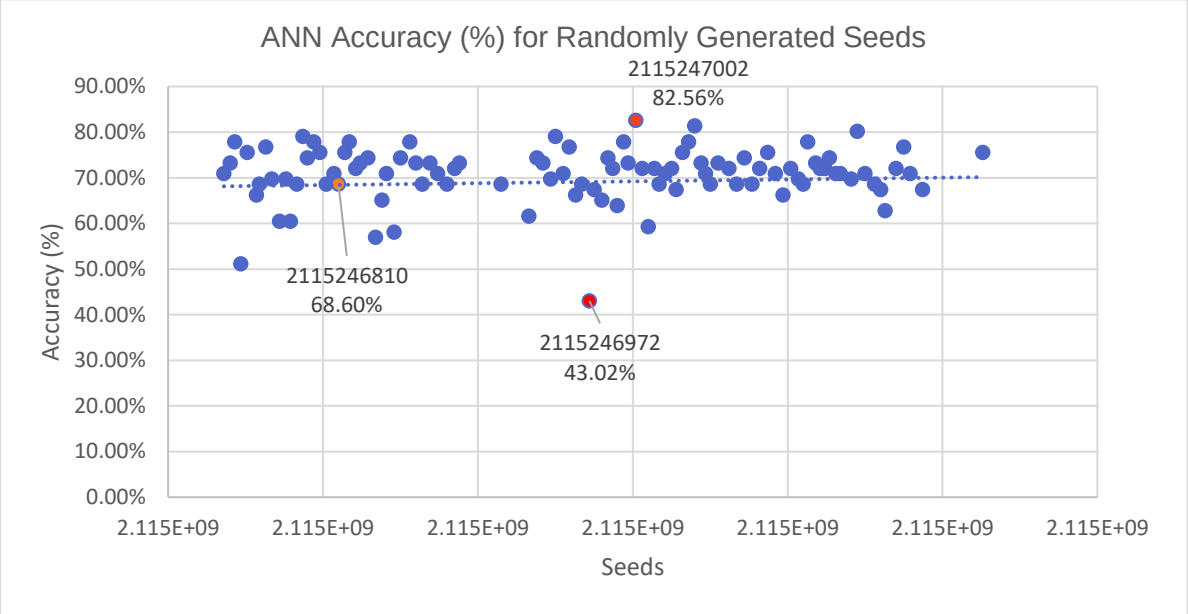
Parameter Description	Initial Value(s)	Final Value
Maximum Epochs for Training	10, 25, 50, 100, 200, 500, 1000	25
Max No-Improvement Epochs	1, 2, 5, 10, 20, 50, 100	10
Learning Rate	0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 0.7	0.05
Error Tolerance	0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 0.7	0.05
Number of Hidden Neurons	1, 2, 5, 9, 15, 20, 51	5
Number of Inputs	51	51
Number of Outputs	1	1
Weights	Small random values	Small random values
Best Seed	**2115247002 (Test set Accuracy 82.56%)**	

Table 3: ANN Design Components and Parameters Before and After Tuning

Above is a table of parameters before and after fine-tuning. The fine-tuning process happened offline, whereby each parameter was incrementally adjusted until there was no more improvement in the ANN's performance. An "improvement" is considered an increase in the average accuracy (for simplicity only accuracy is considered) of the algorithm over 100 runs using randomly generated seeds. The randomly generated seeds ensured that the contents of the test and training sets would be random, and thus helped avoid overfitting the network.

A statistical representation of these parameters during tuning is available in Appendix A

The ANN was then run another 100 times with the tuned parameters and randomly generated seeds. The best of these seeds (i.e., the seed that produced the best average accuracy) was then used as the seed for the final ANN. This can be seen in the graph below:



It can be seen from the graph above that the **average accuracy range** for the ANN is **60-80%** with the average accuracy being **~68.60%** (light blue dot). This is a good average accuracy, and it can thus be concluded that the ANN has been successfully tuned without overfitting the algorithm since the performance remained relatively stable across different randomly generated training and test sets. The **best** accuracy was **82.56%**, with **seed 2115247002**, so that was chosen as the seed for the algorithm.

The **Rectified Linear (ReL)** Activation function was used as the activation function for the Hidden layer due to (unlike sigmoid and tanh functions) its ability to avoid the vanishing gradient problem that is common in networks that make use of backpropagation to train (Brownlee, 2020). The vanishing gradient problem refers to large positive and negative values saturating around the positive and negative ends of a nonlinear activation function, making it difficult for the network to detect changes that are not between 0 and 1. This means that ANNs using ReL function are generally easier to train and perform better.

The **Binary Sigmoid** activation function was used as the activation function for the output layer to scope the output between 0 and 1 for easy binary classification of instances (Brownlee, 2020). The data was binary encoded during pre-processing, which also makes this function a good fit in this case. **Outputs over 0.5 were considered class 1, and outputs below 0.5 were considered class 0.**

To **avoid overfitting** (due to, as previously mentioned, the large number of inputs into the ANN), **early stopping and thinning** of the layers was implemented. The number of hidden nodes was thinned to 5 (< the number of classes) from 51 (equal to the number of inputs). A fixed cap on the maximum number of epochs of the training set was also implemented (25), as well as a stopping condition that indicated if the ANN's output did not change by a tolerance margin of 0.05 for 10 consecutive iterations, then the algorithm was considered to have converged, and the training was stopped. This was done to hopefully allow the algorithm to generalize better and thus perform better on the test set.

Genetic Programming (GP)

The Genetic Programming metaheuristic uses Darwin's principle of natural selection to produce programs which provide optimal solutions to problems. The algorithm begins by generating an initial population of individuals (programs) and, through several generations, modifies and updates the population to reflect the evolving individuals.

In this case, the GP is a classification algorithm that is used to evolve decision trees (classifiers) that classify breast cancer. The classes the trees can assign in this case are also **recurrence-events** or **no-recurrence-events**.

Data Transformation and Pre-Processing

The Decision Trees were capable of learning directly from the nominal data and so **no transformation was necessary**.

The **exact same training set and test set instances** (just with the data in its raw form) were used in the training and testing of the decision trees as the ANN to ensure these methods were comparable.

Individual Representation

Individuals in the algorithm are represented as decision trees in which each node can either be a functional or terminal node.

The functional and terminal sets for this case are shown below:

Set Description	Values
Functional Set: *with the output of each attribute (function) being one of the values of said attribute.*	age | menopause | tumor-size | inv-nodes | node-caps | deg-malig | breast | breast-quad | irradiat
Terminal Set:	recurrence-events | no-recurrence-events

Design Components and Parameters

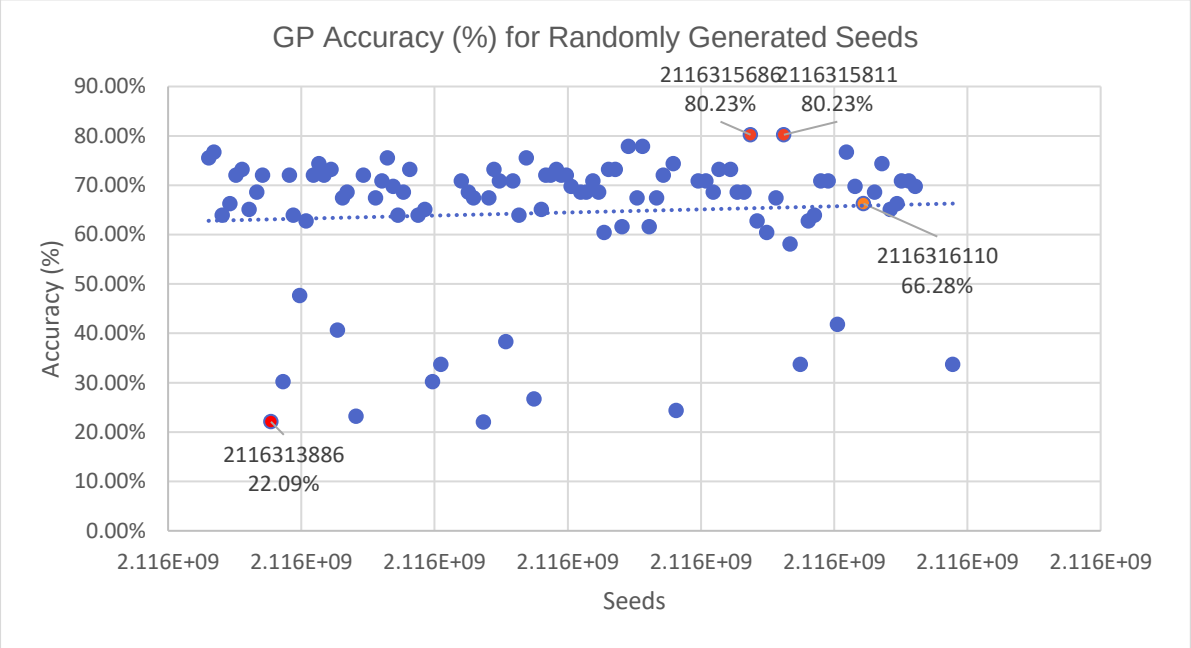
Parameter Description	Initial Value(s)	Final Value
Representation	Decision trees	
Tree Generation	Grow	
Elitism?	Yes (1 Individual)	
Maximum Generations	50	50
Population size	100	100
Min Tree Depth	2	2
Max Tree Depth	3, 4, 5, 6, 7, 8, 9	4
Max Offspring Tree Depth	3, 4, 5, 6, 7, 8, 9	4
Selection Method	Tournament Selection	
Tournament Size	2, 5, 10, 15, 20, 50, 70	50
Max No-Improvement Gens	1, 2, 5, 10, 20, 35, 50	5
Error Tolerance	0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 0.7	0.01
Crossover Rate *(mutation = 100-crossover)*	0.01, 0.1, 0.2, 0.5, 0.7, 0.9, 1.0	0.9
Mutation Type	Grow and Shrink Mutation (50% chance)	
Max Mutation Depth	3, 4, 5, 6, 7, 8, 9	4
Best Seed	**2116315811 (Test set Accuracy 80.23%)**	

Table 3: GP Design Components and Parameters Before and After Tuning

Above is a table of parameters before and after fine-tuning. The fine-tuning process happened in the same manner as was done with the ANN. An "improvement" is again considered an increase in the average accuracy of the algorithm over 100 runs using randomly generated seeds. The randomly generated seeds ensured that the contents of the test and training sets would be random, and thus helped avoid overfitting the algorithm.

A statistical representation of these parameters during tuning is shown in Appendix B.

The GP algorithm was then run another 100 times with the tuned parameters and randomly generated seeds. The best of these seeds (i.e., the seed that produced the best average accuracy) was then used as the seed for the final GP. This can be seen in the graph below:



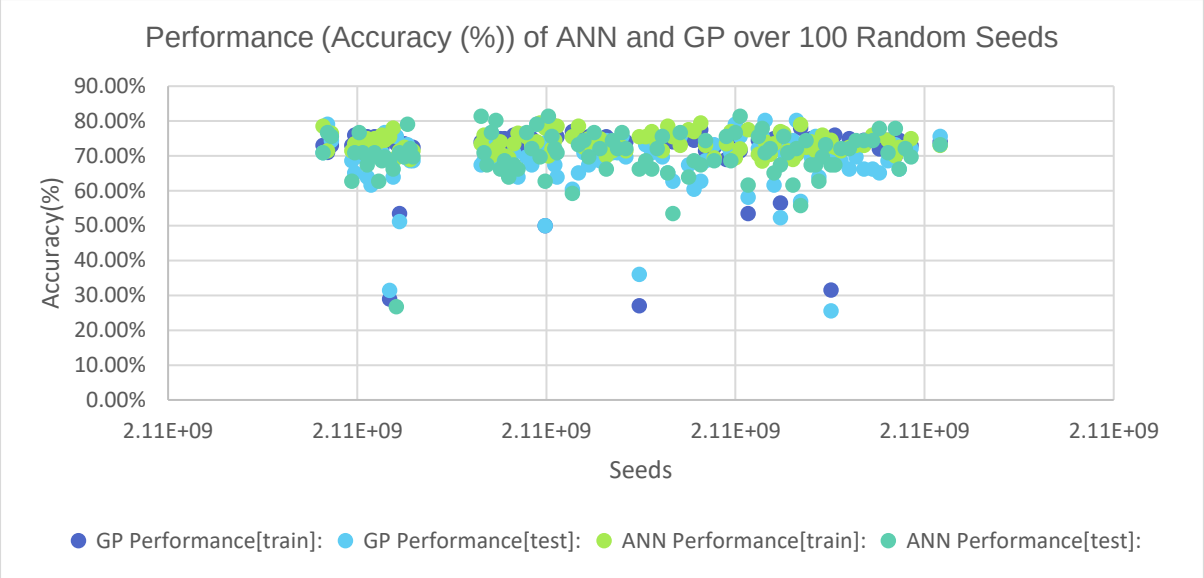
It can be seen from the graph above that the **accuracy range for GP is very large** and this could be due to the **stochastic** nature of the algorithm, with the average accuracy being **~66.28%** (light blue dot). However, **majority of the accuracy values lie above 60%**, meaning that the algorithm is relatively stable. It can thus be concluded that the algorithm has been successfully tuned. The **best** accuracy was **80.23%**, with **seed 2116315811**, so that was chosen as the seed for the algorithm.

Weka C4.5 Decision Tree

This Machine Learning (ML) tool was used as a control to measure the standard for the other two algorithms used in this assignment. The parameter values were kept as default and so the algorithm was not tuned for this dataset. The data was not pre-processed but given in its raw form as an .arff file to the ML tool for classification. There was thus no way to ensure the training and test instances would be the same for this tool, but one can assume that there would be significant overlap in the instances in each set due to the inclusion of only one small dataset.

Results

To ensure both algorithms are comparable, it is imperative that they run on the same testing and training data. To ensure this, both algorithms were run on 100 random seeds to determine seeds that would produce good results for the ANN and GP decision tree.



Seed:	GP Accuracy [train]:	GP Accuracy [test]:	ANN Accuracy [train]:	ANN Accuracy [test]:
2109628050	71.50%	75.58%	72.00%	81.40%
2109626023	70.50%	74.42%	70.00%	81.40%
2109628645	70.50%	80.23%	70.50%	72.09%
2109628314	72.00%	80.23%	74.00%	70.93%
2109625898	73.50%	79.07%	74.00%	79.07%

Table 4: Table Comparing the Performance (Accuracy) of ANN and GP for Different Seeds

The algorithms were then run a total of 5 times (using the 5 different seeds presented above) and the results are available in the tables below:

Seed	Dataset	# Instances	# Correct	Accuracy	Precision	Recall	F-Measure
2109628050	Train	200	144	72.00%	0.559	0.317	0.404
2109628050	Test	86	70	81.40%	0.765	0.520	0.619
2109626023	Train	200	140	70.00%	0.578	0.388	0.464
2109626023	Test	86	70	81.40%	0.600	0.333	0.429
2109628645	Train	200	141	70.50%	0.568	0.385	0.459
2109628645	Test	86	62	72.09%	0.433	0.650	0.520
2109628314	Train	200	148	74.00%	0.660	0.485	0.559
2109628314	Test	86	61	70.93%	0.375	0.706	0.490
2109625898	Train	200	148	74.00%	0.622	0.444	0.519
2109625898	Test	86	68	79.07%	0.750	0.273	0.400

Table 5: Table Comparing Different Statistical Measures of the Performance of the ANN on the Breast Cancer Dataset for Different Seeds

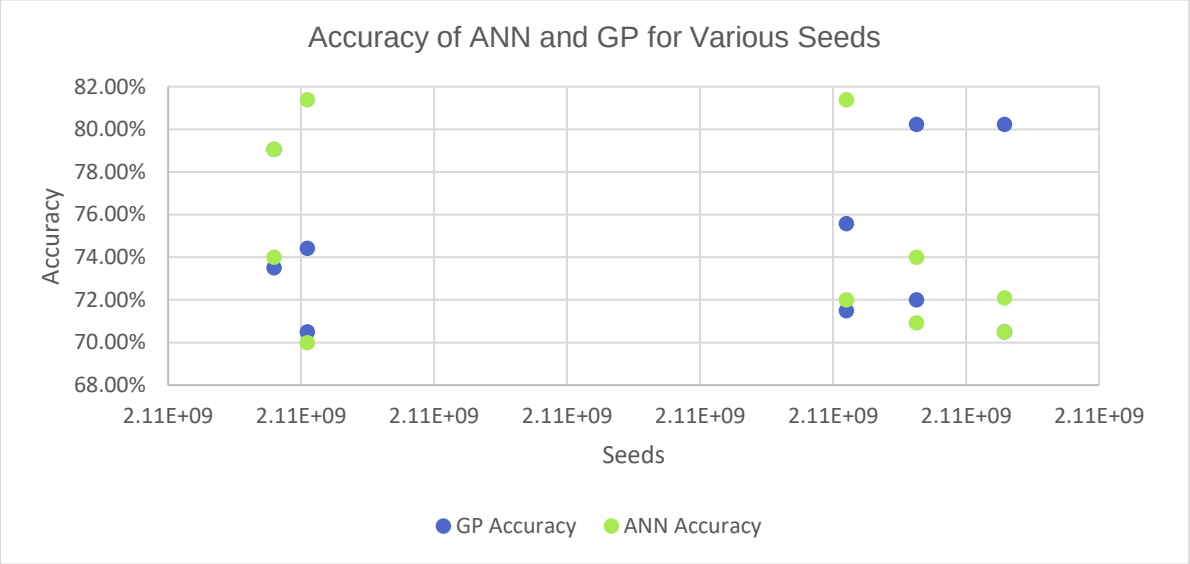
Seed	Dataset	# Instances	# Correct	Accuracy	Precision	Recall	F-Measure
2109628050	Train	200	143	71.50%	0.714	0.083	0.149
2109628050	Test	86	65	75.58%	0.833	0.200	0.323

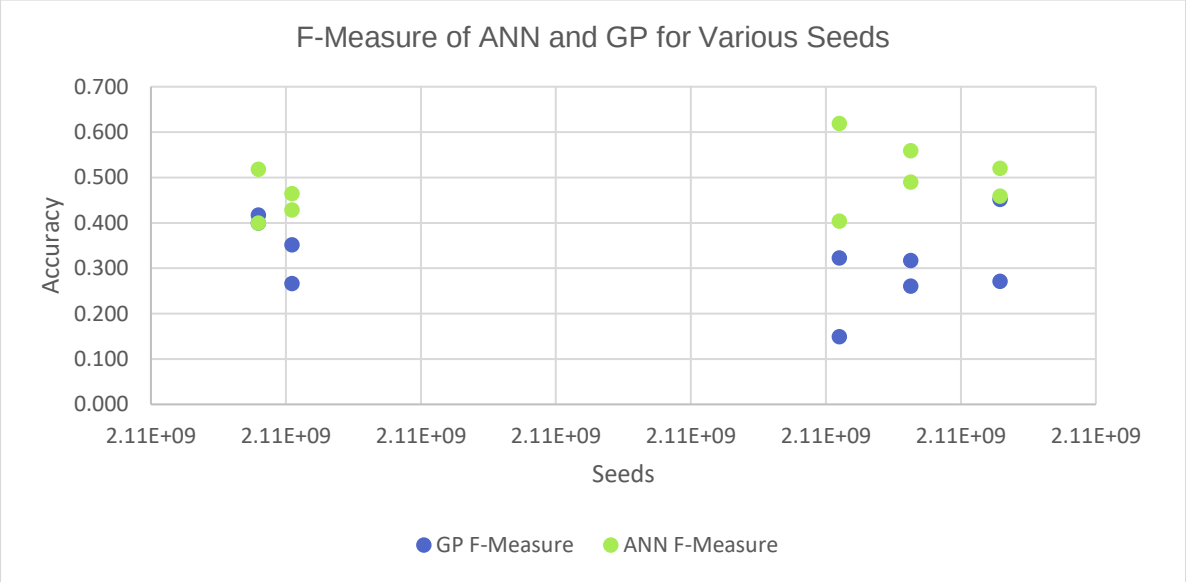
2109626023	Train	200	141	70.50%	0.667	0.239	0.352
2109626023	Test	86	64	74.42%	0.333	0.222	0.267
2109628645	Train	200	141	70.50%	0.688	0.169	0.272
2109628645	Test	86	69	80.23%	0.636	0.350	0.452
2109628314	Train	200	144	72.00%	0.929	0.191	0.317
2109628314	Test	86	69	80.23%	0.500	0.176	0.261
2109625898	Train	200	147	73.50%	0.679	0.302	0.418
2109625898	Test	86	68	79.07%	0.750	0.273	0.400

Table 6: Table Comparing Different Statistical Measures of the Performance of GP on the Breast Cancer Dataset for Different Seeds

Seed	Dataset	# Instances	# Correct	Accuracy	Precision	Recall	F-Measure
N/A	Total	286	219	76.57%	0.513	0.294	0.374

Table 7: Table Comparing Different Statistical Measures of the Performance of the Weka ML tool on the Breast Cancer Dataset for Different Seeds





Statistical Analysis

Kruskal Wallis Test

The Kruskal Wallis test (or H-test) is a common non-parametric statistical analysis test that allows the difference between 3 or more independent groups to be analysed, particularly when the data is not normally distributed, or one cannot assume equal variance. That makes this test a good fit for the analysis of these three independent ML algorithms.

It is assumed for the Kruskal Wallis test that the data sampled contains nominal or ordinal values with more than two expressions, and a metric variable. The data here satisfies both assumptions.

The initial step is to formulate a null hypothesis. This null hypothesis assumes no significant difference between the independent groups. In this case, the **null hypothesis** is: "**there is no significant difference between the different models' performance in classifying breast cancer**".

Then rank assignment is performed, whereby the different statistical data for each performance metric from each model are combined and ranked according to the highest-lowest values (lowest value receiving rank 1). Ties are ignored. The two metrics that will be assessed are Accuracy and F-measure. Because Weka does not provide statistical values for the training sets, only the test-set values will be considered for this test.

The rank assignment table for each metric is shown below:

Rank Assignment for **Accuracy**:

Model	Value	Rank
ANN	70.93%	1
ANN	72.09%	2
ANN	79.07%	6
ANN	81.40%	10
ANN	81.40%	11
GP	74.42%	3
GP	75.58%	4
GP	79.07%	7
GP	80.23%	8
GP	80.23%	9
WEKA	76.57%	5

Rank Assignment for **F-measure**:

Model	Value	Rank
ANN	0.400	5
ANN	0.429	7
ANN	0.490	9
ANN	0.520	10
ANN	0.619	11
GP	0.261	1
GP	0.267	2
GP	0.323	3
GP	0.400	6
GP	0.452	8
WEKA	0.374	4

The next step is to compute the rank sum and the mean rank sum for each model. The results of this are seen in the table below:

Accuracy:

Model	Rank Sum (R_i)	Mean Rank Sum ($\bar{R}_i$)
ANN	30	30/5 = 6.0
GP	31	31/5 = 6.2
WEKA	5	5/1 = 5

F-measure:

Model	Rank Sum (R_i)	Mean Rank Sum ($\bar{R}_i$)
ANN	42	42/5 = 8.4
GP	20	20/5 = 4.0
WEKA	4	4/1 = 4.0

Expected value for rank sums (if no difference in the groups): $E_R = \frac{n+1}{2} = \frac{11+1}{2}$ **= 6.0**

Number of Cases (n): 11 Number of models(k): 3 Degrees of Freedom (k-1): 2

Variance of Ranks: $\frac{n^2-1}{12} = \frac{11^2-1}{12} = 10$

After all the rank sum values are calculated, the test value H (or the p-value) is calculated for each metric. This value corresponds to the Chi-square value which follows a chi-square distribution with (k - 1) degrees of freedom, where k is the number of models.

Test Value H (p-value) for Accuracy:

$$H = \frac{n-1}{12} * \sum_{i=1}^{k} \frac{n_i(R_i - E_R)^2}{\sigma_R^2} = \frac{11-1}{12} * \frac{5(6.0-6.0)^2+5(6.2-6.0)^2+1(5-6.0)^2}{10} =$$ **0.1**

Test Value H (p-value) for F-Measure:

$$H = \frac{n-1}{12} * \sum_{i=1}^{k} \frac{n_i(R_i - E_R)^2}{\sigma_R^2} = \frac{11-1}{12} * \frac{5(8.4-6.0)^2+5(4.0-6.0)^2+1(4.0-6.0)^2}{10} =$$ **4.4**

Once the H value is calculated, this is compared to the corresponding critical chi-square value in the chi-square (below). If this value is below the critical chi-square value, the null hypothesis can be accepted. If not, it must be rejected as there are significant differences among the models.

Chi-square Distribution Table

d.f.	.995	.99	.975	.95	.9	.1	.05	.025	.01
1	0.00	0.00	0.00	0.00	0.02	2.71	3.84	5.02	6.63
2	0.01	0.02	0.05	0.10	0.21	4.61	5.99	7.38	9.21
3	0.07	0.11	0.22	0.35	0.58	6.25	7.81	9.35	11.34
4	0.21	0.30	0.48	0.71	1.06	7.78	9.49	11.14	13.28
5	0.41	0.55	0.83	1.15	1.61	9.24	11.07	12.83	15.09
6	0.68	0.87	1.24	1.64	2.20	10.64	12.59	14.45	16.81
7	0.99	1.24	1.69	2.17	2.83	12.02	14.07	16.01	18.48
8	1.34	1.65	2.18	2.73	3.49	13.36	15.51	17.53	20.09
9	1.73	2.09	2.70	3.33	4.17	14.68	16.92	19.02	21.67
10	2.16	2.56	3.25	3.94	4.87	15.99	18.31	20.48	23.21

The **degrees of freedom** in this case are 2. Since these algorithms are used to classify medical data, the **cost of error is significant** and thus the **significance level** (alpha) used here is **0.01**. It can therefore be concluded that the **critical chi-square value** is **9.21** in this case.

Looking at the H values for accuracy and f-measure, both values are smaller than the critical chi-square value, thus the **null hypothesis is accepted**.

Conclusion

As can be seen from the results above, the null hypothesis is accepted, and it can be concluded that there are no statistical differences between the different models' performance. This means that any of these models can be used to classify breast cancer, with similar results. This is supported by the fact that both ANN and GP models perform similarly (in terms of the accuracy and F-measure metrics listed in **Tables 5-7**) to each other and to the WEKA algorithm. However, it is worth pointing out that because of the small amount of data used in this classification assignment, these models may not be best suited to classifying breast cancer, as they would need to be trained on more data and more varied data.

It is also worth pointing out some of the differences between the ANN and the GP algorithms. GP is by nature stochastic and so may perform worse on average compared to the ANN or may produce varied results. This can be seen in some of the outlier accuracy values found in the GP **Accuracy Graphs** above and in **Appendix B**. Thus, GP may take longer to produce accurate results due to the algorithm needing to be rerun again and again. ANN is in general more stable and so may be better in this case.

However, ANNs in general have the problem of easily being overfit to certain instances, and so tend to generalize worse than a Decision Tree. In this case, it can be assumed from the accuracy values in **Table 5** that ANN performs better on the test set than on the training set, and so this is not the case. However, looking at the precision and recall values, it is revealed that, because the dataset had so few recurrence-events compared with no-recurrence events, that the ANN would either overfit to no-recurrence or recurrence depending on how many recurrence-events were in the training set, resulting in (in general) poor recall. A model that produces no false negatives has a recall of 1.0, meaning that the ANN above produces many false negatives, which could be detrimental in a medical environment. GP Decision Trees in this case have even worse recall, as can be seen in **Table 6**, but in general may generalize better and avoid overfitting due to their exploration via random generation and mutation.

Each of these algorithms could be improved to increase their likelihood of producing accurate and reliable results. The ANN could be tested on more sets of data with more even distributions of positive and negative cases, which would assist it in generalizing better and thus avoid the problem of overfitting. The GP algorithm could make use of a sensible construction heuristic such as the ID3 algorithm that WEKA uses (with each tree starting with a random attribute as its root), to produce stronger trees and less sporadic results.

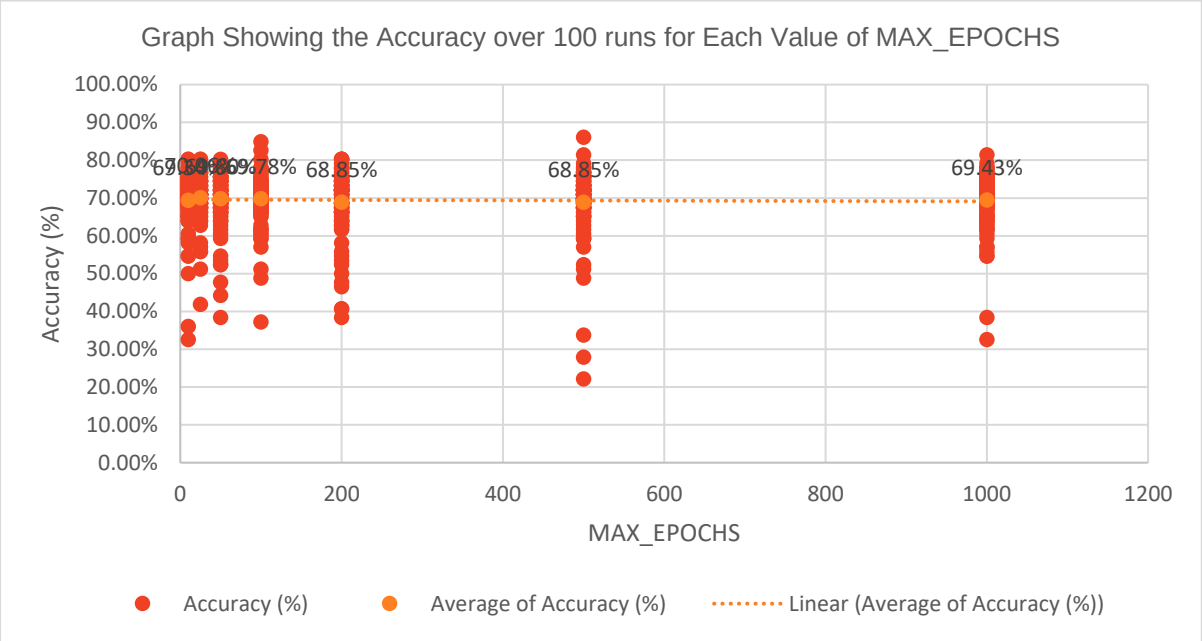
In conclusion, both ANN and GP algorithms may be used to classify breast cancer. However, there are specific strengths and weaknesses of each algorithm, and thus careful consideration of each should be considered. Overall, both algorithms would benefit from being trained on more data with more equal distribution of both positive and negative cases.

References

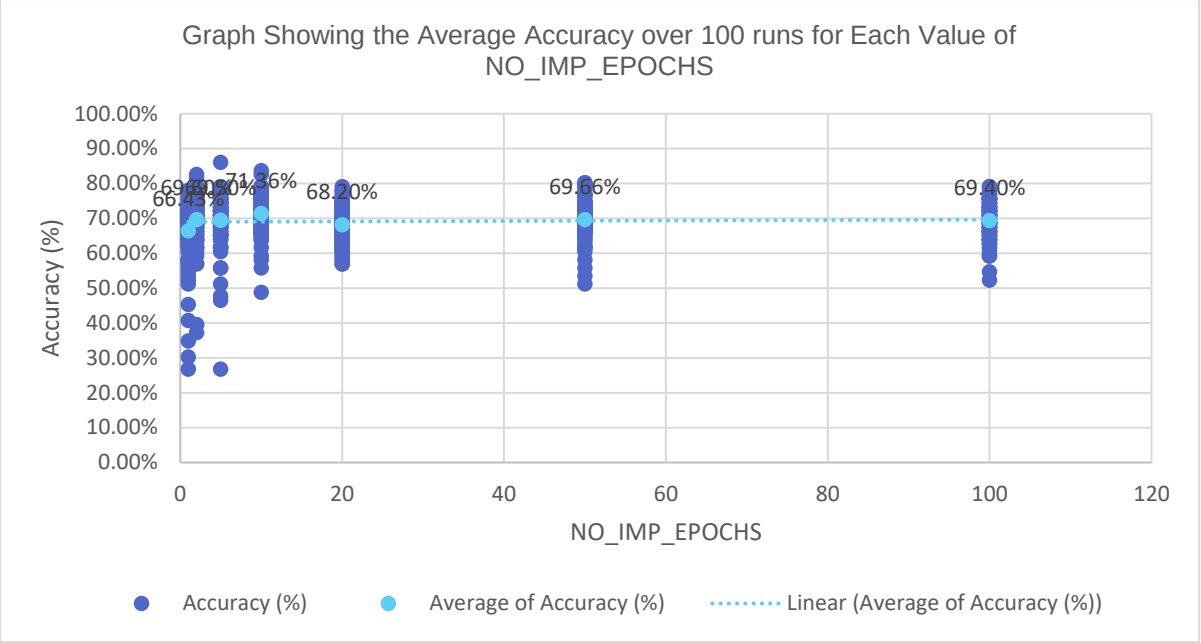
1. Brownlee, J. (2020) A gentle introduction to the rectified linear unit (ReLU), MachineLearningMastery.com. Available at: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ (Accessed: 20 May 2023).
2. Brownlee, J. (2020) Why one-hot encode data in machine learning? MachineLearningMastery.com. Available at: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/ (Accessed: 18 May 2023).

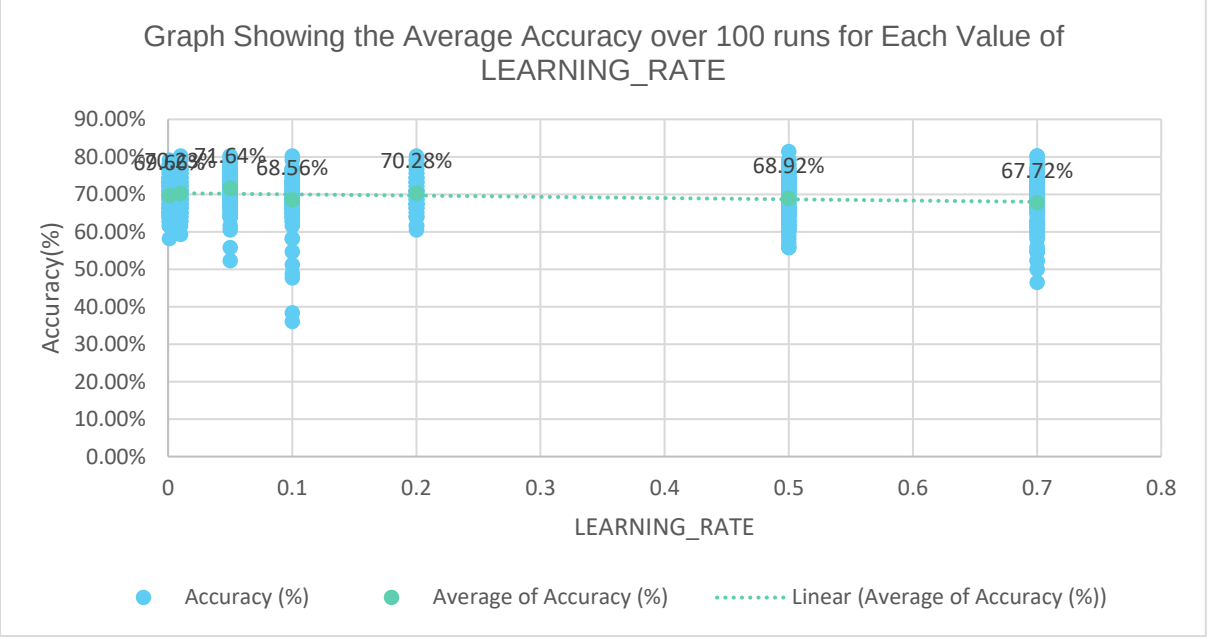
3. Classification: Precision and recall; machine learning; google for developers (2023) Google. Available at: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall (Accessed: 01 June 2023).
4. Wong, C.W. et al. (2021) Analysis of gut microbiome using explainable machine learning predicts risk of diarrhea associated with tyrosine kinase inhibitor NERATINIB: A pilot study, Frontiers. Available at: https://www.frontiersin.org/articles/10.3389/fonc.2021.604584/full (Accessed: 25 May 2023).

Appendix A – Tuning ANN Parameters

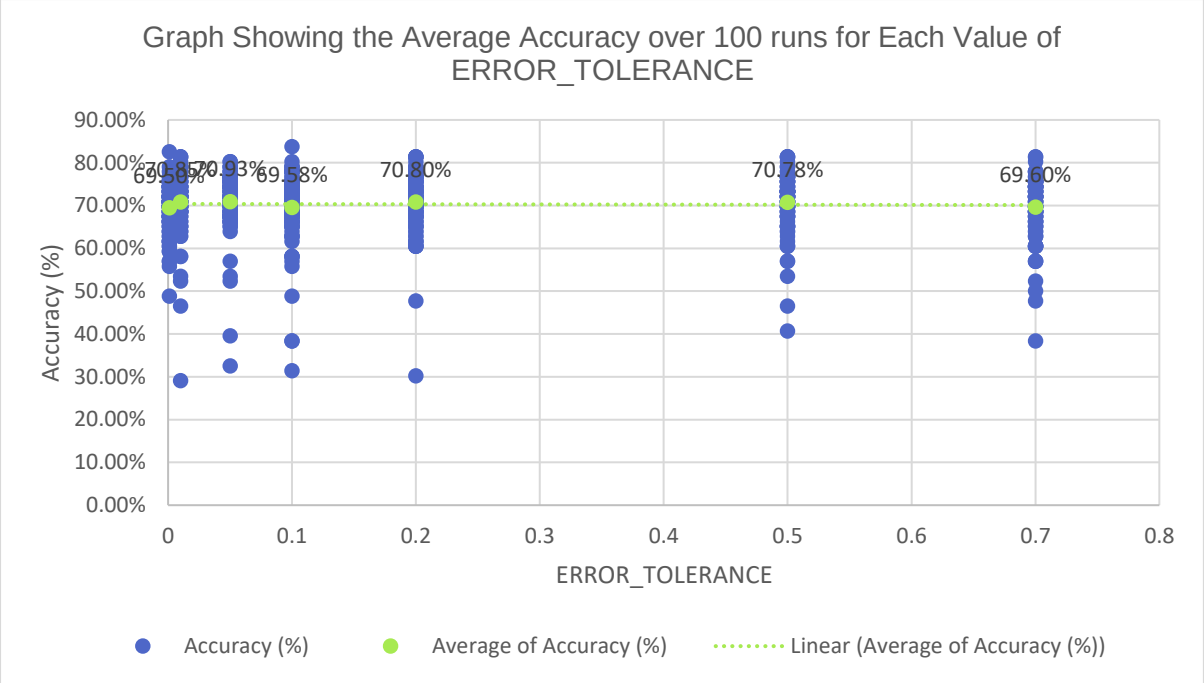


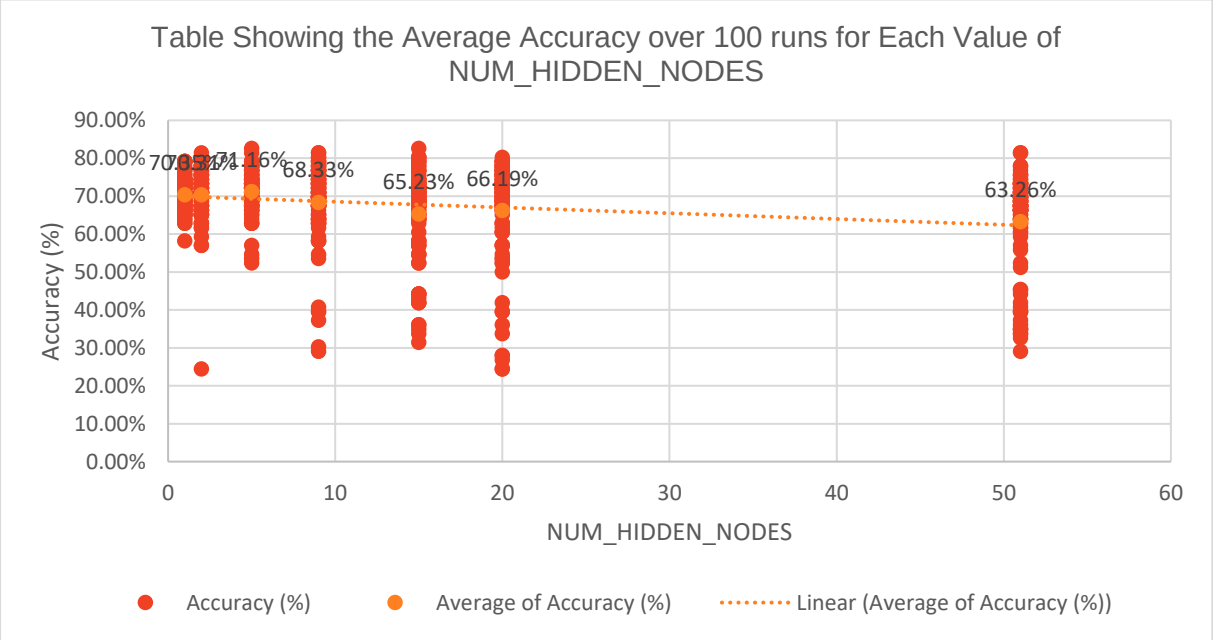
MAX_EPOCHS	Average of Accuracy (%)	NO_IMP_EPOCHS	Average of Accuracy (%)
10	69.34%	1	66.43%
25	**70.06%**	2	69.60%
50	69.80%	5	69.50%
100	69.78%	**10**	**71.36%**
200	68.85%	20	68.20%
500	68.85%	50	69.66%
1000	69.43%	100	69.40%
Total Average	**69.44%**	**Total Average**	**69.16%**





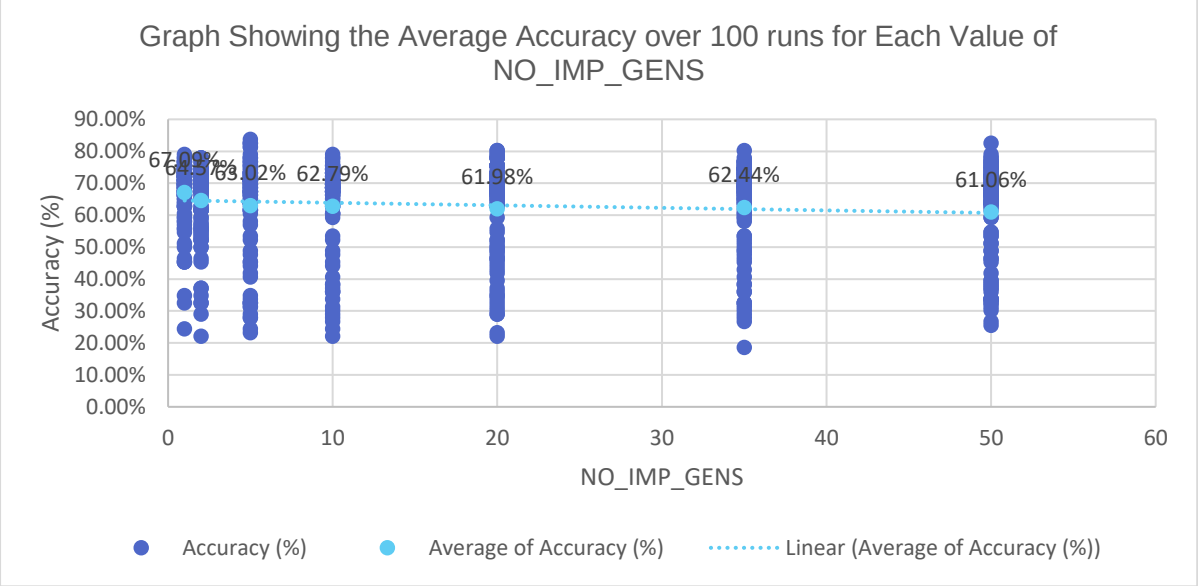
LEARNING_RATE	Average of Accuracy (%)	ERROR_TOLERANCE	Average of Accuracy (%)
0.001	69.66%	0.001	69.50%
0.01	70.23%	0.01	70.85%
0.05	**71.64%**	**0.05**	**70.93%**
0.1	68.56%	0.1	69.58%
0.2	70.28%	0.2	70.80%
0.5	68.92%	0.5	70.78%
0.7	67.72%	0.7	69.60%
Total Average	**69.57%**	**Total Average**	**70.29%**





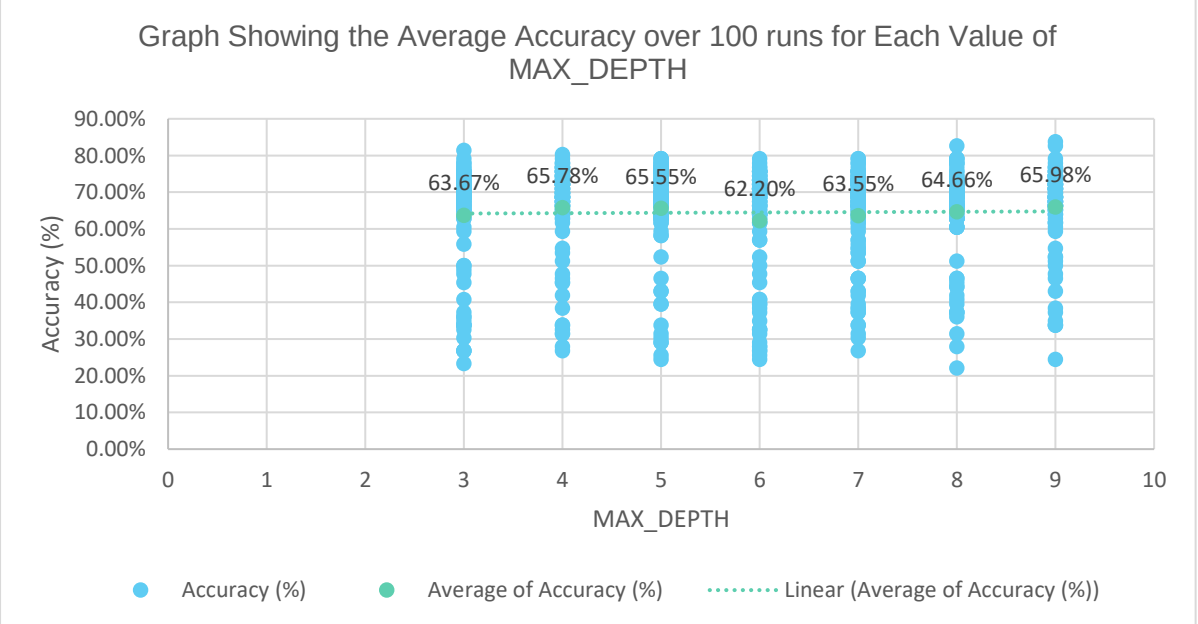
NUM_HIDDEN_NODES	Average of Accuracy (%)
1	70.35%
2	70.31%
5	71.16%
9	68.33%
15	65.23%
20	66.19%
51	63.26%
Total Average	**67.83%**

Appendix B – Tuning GP Parameters



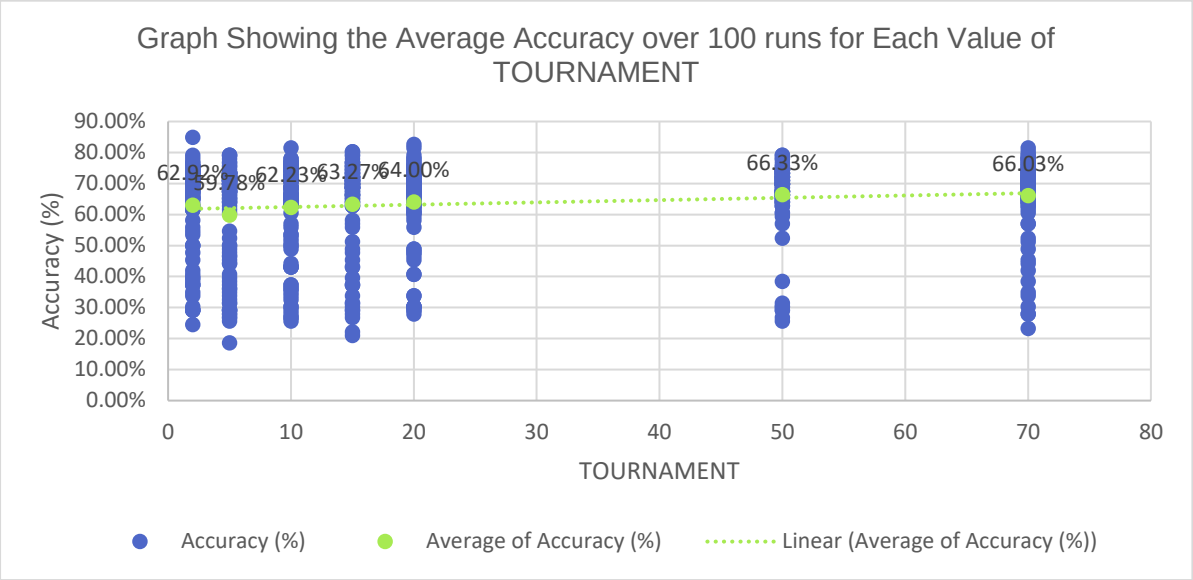
NO_IMP_GENS	Average of Accuracy (%)	MAX_DEPTH	Average of Accuracy (%)
1	67.09%	3	63.67%
2	64.57%	**4**	**65.78%**
5	**63.02%**	5	65.55%
10	62.79%	6	62.20%
20	61.98%	7	63.55%
35	62.44%	8	64.66%
50	61.06%	9	65.98%
Grand Total	**63.28%**	**Grand Total**	**64.48%**

As can be seen from the graph above, **Adding Early dropout** (NO_IMP_GENS i.e., no. of consecutive generations without improvement) seemed to have a **positive impact** on the algorithm's performance as allowing the algorithm to end early greatly improved performance. To allow for exploration and to avoid overfitting, the value 5 will be used.

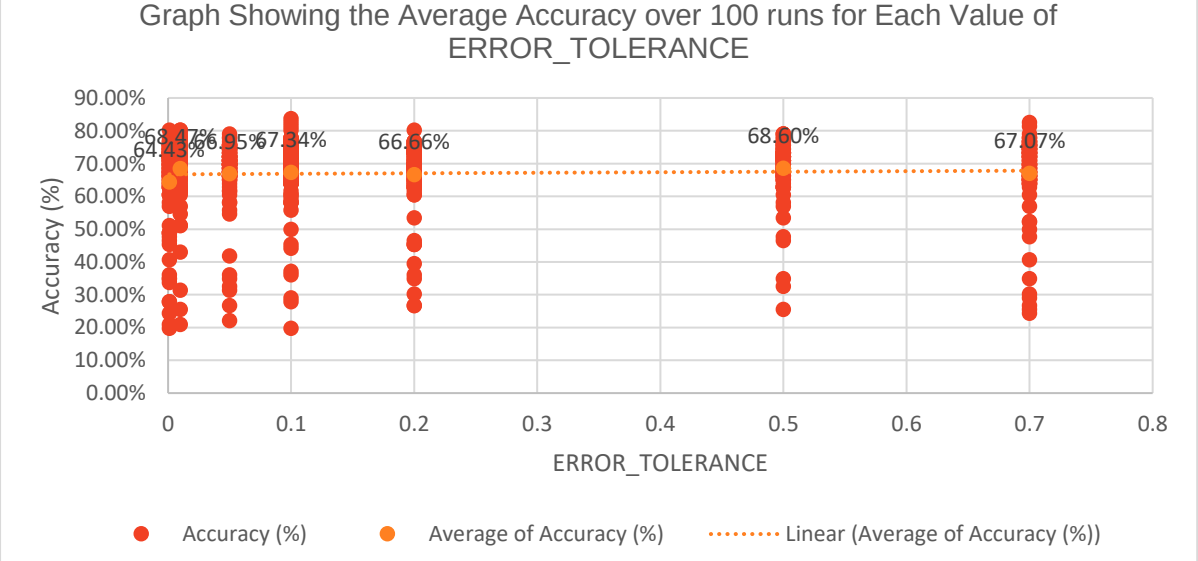


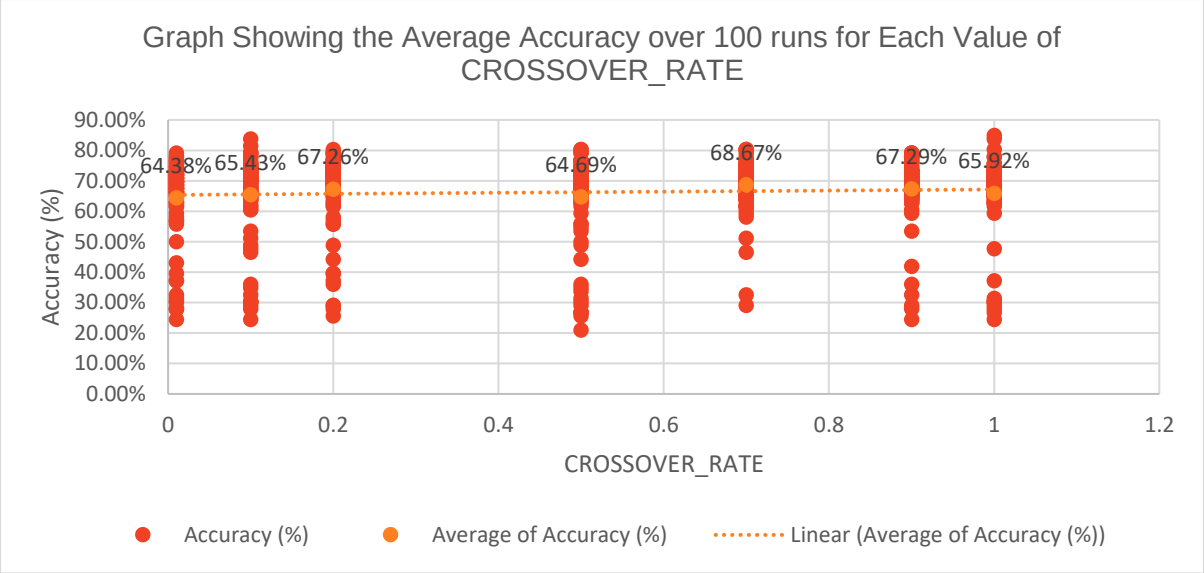
It can be seen in the graph above that **smaller trees produce, on average, results that are just as good as larger trees**. Since there were two values that were <0.5% apart from each other, to avoid having unnecessarily large, redundant trees, it was decided to use value 4.

The graph below shows that **larger tournament sizes produce better results** on average. This can be due to the **higher selection pressure**, meaning that fitter individuals are exploited more which balances out the high exploration in the random generation of individuals.



TOURNAMENT	Average of Accuracy (%)	ERROR_TOLERANCE	Average of Accuracy (%)
2	62.92%	0.001	64.43%
5	59.78%	**0.01**	**68.47%**
10	62.23%	0.05	66.95%
15	63.27%	0.1	67.34%
20	64.00%	0.2	66.66%
50	**66.33%**	0.5	68.60%
70	66.03%	0.7	67.07%
Grand Total	**63.51%**	**Grand Total**	**67.07%**





CROSSOVER_RATE	Average of Accuracy (%)
0.01	64.38%
0.1	65.43%
0.2	67.26%
0.5	64.69%
0.7	68.67%
0.9	**67.29%**
1	65.92%
Grand Total	**66.23%**

As can be seen from the graph above, a very **high crossover** produces slightly better results. This could be due to the **stochastic generation** of individuals that are balanced out by more exploitation (crossover), thus producing better results.