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**. 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** | **1994978327 (Accuracy 83.72%)** | |

*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 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 **~72.09%** (dark green 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 **83.72%**, with **seed** **1994978327**, 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. 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. 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 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 | 20 |
| Error Tolerance | 0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 0.7 | 0.05 |
| Crossover Rate  *(mutation = 100-crossover)* | 0.01, 0.1, 0.2, 0.5, 0.7, 0.9, 1.0 | 1.0 |
| Mutation Type | Grow and Shrink Mutation (50% chance) | |
| Max Mutation Depth | 3, 4, 5, 6, 7, 8, 9 | 4 |
| **Best Seed** | **2019712204 (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 **average** **accuracy** **range** for GP is very large and this could be due to the **stochastic** nature of the algorithm. This is further supported by the average accuracy being **~50.00%** (dark blue dot). The **best** accuracy was **80.23%**, with **seed** **2019712204**, 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 due to the inclusion of only one small dataset.

## Results

After much trial and error and random generation of seeds, there were some seeds found that produce good results for both ANN and GP:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Seed: | GP Accuracy [train]: | GP Accuracy [test]: | ANN Accuracy [train]: | ANN Accuracy [test]: |
| 2069687265 | 68.00% | 81.40% | 19.50% | 83.72% |
| 2069569824 | 65.00% | 72.09% | 76.50% | 77.91% |
| 2069407346 | 54.50% | 58.14% | 86.50% | 77.91% |
| 2068090817 | 66.50% | 79.07% | 75.50% | 76.74% |
| 2069571189 | 70.50% | 73.26% | 81.00% | 76.74% |
| 2068092770 | 28.00% | 26.74% | 82.50% | 82.56% |
| 2069406591 | 65.50% | 60.47% | 88.00% | 76.74% |
| 2069565546 | 69.00% | 73.26% | 77.50% | 75.58% |
| 2069405030 | **71.50%** | **72.09%** | **84.50%** | **74.42%** |
| 2068087822 | 71.00% | 70.93% | 82.00% | 74.42% |
| 2069570475 | 52.50% | 66.28% | 85.50% | 74.42% |
| 2069690202 | 71.00% | 73.26% | 81.00% | 72.09% |
| 2068093112 | 74.00% | 70.93% | 80.50% | 72.09% |
| 2069570728 | 45.50% | 41.86% | 70.00% | 81.40% |
| 2069688877 | 74.50% | 74.42% | 83.50% | 70.93% |

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

It was decided to use one of these seeds () to ensure both algorithms perform on the same portions of testing and training data in order that they are comparable.

Seed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset | # Instances | # Correct Instances | Accuracy | F-measure |  |
| Training set | 200 (70%) |  |  |  |  |
| Test set | 86 (30%) |  |  |  |  |
| Breast cancer (total) | 286 |  |  |  |  |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset | # Instances | # Correct Instances | Accuracy | F-measure |  |
| Training set | 200 (70%) |  |  |  |  |
| Test set | 86 (30%) |  |  |  |  |
| Breast cancer (total) | 286 |  |  |  |  |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset | # Instances | # Correct Instances | Accuracy | F-measure |  |
| Training set | 200 (70%) |  |  |  |  |
| Test set | 86 (30%) |  |  |  |  |
| Breast cancer (total) | 286 |  |  |  |  |

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

[graphs]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]

## Conclusion

As can be seen from the results above, the \_\_\_

Performing a [STAT TEST HERE] statistical analysis of the data, we find that the \_\_\_\_ has ….

## 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

|  |  |  |  |
| --- | --- | --- | --- |
| CROSSOVER\_RATE | Average of Accuracy (%) | MAX\_DEPTH | Average of Accuracy (%) |
| 0.01 | 61.71% | **3** | **66.52%** |
| 0.1 | 63.10% | **4** | **66.35%** |
| 0.2 | 62.37% | 5 | 62.73% |
| 0.5 | 61.80% | 6 | 65.26% |
| 0.7 | 63.65% | 7 | 63.88% |
| 0.9 | 63.58% | 8 | 65.90% |
| 1 | **65.41%** | 9 | 64.81% |
| Total Average | **63.09%** | **Total Average** | **65.06%** |

As can be seen from the graph above, an increase in the rate of mutation was found to decrease the fitness of individuals generated over time. Thus, **it was decided that mutation would be removed from the algorithm**, as the random generation of individuals provided enough diversity in the program space that mutation was unnecessary and resulting in the search becoming a random walk.

It can be seen in the graph above that **smaller trees produce, on average, better results**. This could be due to the trees being less redundant. Since there were two values that were <0.5% apart from each other, it was decided to take the larger of the two values to provide more diversity for the grow tree generation and crossover methods.

|  |  |  |  |
| --- | --- | --- | --- |
| Row Labels | Average of Accuracy (%) | NO\_IMP\_GENS | Average of Accuracy (%) |
| 2 | 60.74% | 1 | 69.02% |
| 5 | 60.43% | 2 | 68.97% |
| 10 | 66.35% | 5 | 67.70% |
| 15 | 65.27% | 10 | 66.84% |
| 20 | 66.00% | **20** | **69.86%** |
| 50 | **67.17%** | 35 | 67.50% |
| 70 | **67.45%** | 50 | 69.35% |
| Total Average | **64.77%** | **Total Average** | **68.46%** |

As can be seen from the graph above and below, the **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. **Adding Early dropout** (NO\_IMP\_GENS i.e., no. of generations without improvement) seemed to have the **biggest positive impact** on the algorithm’s performance as allowing the algorithm to end early greatly improved performance.

|  |  |
| --- | --- |
| ERROR\_TOLERANCE | Average of Accuracy (%) |
| 0.001 | 49.44% |
| 0.01 | 50.12% |
| 0.05 | **52.76%** |
| 0.1 | 49.56% |
| 0.2 | 49.44% |
| 0.5 | 49.03% |
| 0.7 | 48.73% |
| Total Average | **49.87%** |