Heriot-Watt University

Master’s Thesis

Using cellular automata for non-binary classification

Author: Filip Wieslaw Bartoszewski
Supervisor: Dr. Michael A. Lones
H00254774
Second reader: Dr. Hind Zantout

School of Mathematical and Computer Sciences

August 2017
A statement of non-plagiarism

I, Filip Wieslaw Bartoszewski
confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: FwBartoszewski
Date: 17.08.2017
Abstract

Cellular Automata (CA) have been recognized as a powerful tool for a long time. They have been used for a myriad of tasks ranging from modelling and simulation to image processing and even as a random number generator in Mathematica. Even simple CAs, such as Conway’s Game of Life, are known to be computationally universal, and display a range of fascinating behaviours that mimic natural systems. Despite these many applications, there is still one possibility that remains barely investigated: using CAs for classification problems. Recently there has been some research in this area however. It was demonstrated that, using an evolutionary algorithm (EA), one is able to devise CA rules capable of solving a simplified binary classification problem. Given the huge variety of CAs and EA approaches, it seems highly likely that CAs have the potential to solve much more difficult predictive modelling problems. This is the case particularly for image data, where the grid structure of a CA has a natural affinity for processing patterns organized within a pixel grid.

This project aims to create a classification tool, using the capabilities of CAs and EAs to help recognize cancer and the degree of its malignity on mammogram photos provided by the radiological cell from the University of York.
Acknowledgments

I would like to thank Dr. Michael A. Lones, for the inspiration, help and advice, my family for constant support and friends for helping me to see this project through to the end.
List of contents

Declaration of non-plagiarism ........................................................................................................... 3
Abstract ........................................................................................................................................ 4
Acknowledgements ....................................................................................................................... 5
List of contents ............................................................................................................................... 6
Introduction ................................................................................................................................... 7
Professional, ethical and legal issues .............................................................................................. 8
Aims, objectives and requirements ................................................................................................. 9
  Aim of the project ......................................................................................................................... 9
  Requirements of the system architecture .................................................................................... 9
  Functional and non-functional requirements ............................................................................. 9
Literature review ........................................................................................................................... 11
  Cellular Automata Overview ....................................................................................................... 11
  Mammogram Classification Overview .................................................................................... 13
  The Game of Life ....................................................................................................................... 14
  Different uses of Cellular Automata ......................................................................................... 17
  Genetic programming and evolutionary algorithms ................................................................. 21
Related work ................................................................................................................................ 23
Methodology .................................................................................................................................. 26
  Initial experiment idea ............................................................................................................... 26
  ECJ description .......................................................................................................................... 26
  Description of the breast cancer dataset .................................................................................. 30
Implementation ............................................................................................................................. 31
  Development process ................................................................................................................ 31
  Rule models ............................................................................................................................... 35
Results .......................................................................................................................................... 40
Evaluation .................................................................................................................................... 44
Final discussion ............................................................................................................................... 46
  Conclusion and reflection on the project requirements .............................................................. 46
Future work ................................................................................................................................... 47
Bibliography and references ......................................................................................................... 48
Appendices .................................................................................................................................... 50
Introduction

Cellular Automata are a discrete model consisting of a grid of cells which change their state depending on the state of their neighbours. They are a very flexible tool used in many disciplines in a variety of ways, but barely used for image classification. Next to no research has been conducted in this field in previous years, with only single articles being published. Even those articles have only touched on the basics of this idea.

One of these papers is “Learning Cellular automata rules for binary classification problem” written by Piwonska, Seredyński and Szaban in 2013 [5]. They used CA to classify whether a point on a surface lies over or under a plotted line of a function. Whilst the problem they have managed to solve was simplified and binary, instead of class-based, their method proved to be more accurate than the popular k-NN nearest neighbour. That suggests that if CA can be a viable classification tool in a simplified setting, it may a viable methodology to solve much more advanced problems.

The goal of this MSc project is to build on this idea and create a program that will use CA to classify images from a mammogram photo dataset according to the risk of cancer. To further enhance this idea, the rules that will describe CA behaviour will be created by applying genetic programming and evolutionary algorithms. This will let the program come up with its own solutions and hopefully create better rules than a human-devised heuristic would.

There are two main motivations behind this research. The first one is to prove that CA can be successfully used as a classifying tool. This is why the project is meant to be a proof of concept and there is no null hypothesis. After the software will be tested, results could be evaluated both as a classifying or a regression problem, meaning different metrics will be applied to evaluate the end results. Ideally, an accuracy of over 70% will be achieved while using k-fold cross validation and splitting the dataset into a training set and a test set. By comparison, research that compared different methods of breast cancer mass detection stated that best classifiers would reach sensitivity of 0.8 and area under the ROC curve over 0.75 [12]. While the article referred is from 2009, it still provides a valid reference point - it was provided as supporting material for the dataset used in the project.

The second idea that drives this project is how it would help people working in the biomedical field. Should it be successful, the software created will be a new way to help computer aided diagnostics to detect breast cancer. The dataset for this research was provided by the radiology cell from the University of York. If this research succeeds, it may help further cooperation between the universities and give new ideas how to improve the classification program.

Similarly to the motivation, the impact of this research could be divided into two main parts. If the program works as predicted and the concept that CA can be successfully used for classification is proved, then a brand-new tool could be created. This research could be continued to create a universal framework that thanks to its evolutionary nature could be able to classify any type of image data, not only mammogram photos. The impact on the medical field would probably be much less significant. Nevertheless, it could be a new way to help detect breast cancer, or even discover a risk of cancer with enough probability to suggest preventive measures.

This paper will consist of firstly, a discussion of the professional, ethical and legal issues concerning this project followed by description of the objectives, system architecture and requirements. Next part would be the literature review that includes explanations of the CA’s nature and main uses including ways to analyze mammograms using computer aided diagnostics and a description of the ideas behind genetic programming and evolutionary algorithms. Following chapter describes the project methodology, the ideas behind the experiment, third-party framework used for the program and the analyzed dataset. The implementation chapter is dedicated to the development process and predictive models used for the final experiments. Next chapter presents the results of the experiments and
analyzes them. Finally, a closing discussion of the project with reflection on the initial requirements and a bibliography finishes the paper.

Separate appendices include: risk assessment and sample result files from cross-validation. Sample parameter and output files, ROC curves for one of the models from the cross-validation, dataset, code for the problem files and the framework code (in JAR files) will be included separately due to their volume. The dataset will be uploaded separately.

**Professional, ethical and legal issues**

While the program that will be one of this project’s deliverables is not meant for wide-spread use, the software will be released under the MIT license. This license is a part of the open source initiative and, unless stated different in any future work, it will be used for the program in its current stage.

---

**Permission**

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

All the third-party libraries and packages will be credited. At the current time of the development of this project, the most important one is ECJ. It is a java based Evolutionary Computation and Genetic Programming System and will be used as a part of the program that evolves rules describing CA behaviour. The short synapse of ECJ license states [28]:

- You are free to redistribute this document.
- You may not modify, transform, translate, or build upon the document except for personal use.
- You must maintain the author’s attribution with the document at all times.
- You may not use the attribution to imply that the author endorses you or your document use.

The above licensing statements cover the professional and legal issues, but due to the nature of the dataset and the problem that the software is meant to help classify, ethical issues must be discussed. The information in the mammogram dataset provided by the University of York is fully anonymous, there is no way to identify the person that has the mammogram taken, based on the information in the dataset.

Apart from the privacy issues it is important to remember that the outcomes of this research may one day affect real people if the software developed for this project will be used as a part of medical research. This is addressed by objectively measuring the system’s performance and interpreting the evolved ruleset of CA. One of the tasks for this project is to enable the user to check all the rules that
have led to the output of the analysis. Thanks to this, the diagnosis of the software will not be a result of calculations done by a “black box”, but a clear and transparent process.

To sum up: the data is not identifiable with living people, it is not sensitive personal or confidential data, there are no human subjects involved and the author confirms that the project involves only standard IT equipment and does not expose anybody to more hazards than a conventional office environment.

Aims, objectives and requirements

Aim of the project

The main objective of the project is to create a program, or a set of programs that will use Cellular Automata (CA) with an evolved ruleset to classify images from a mammogram dataset. Classes will be divided according to the risk of cancer being present and if so, how malevolent or benign it might be. The results need to be accurate (over 70% accuracy achieved on the test folds) and replicable. The latter is especially important due to the genetic nature of the approach used and impact this research might have. The accuracy threshold is meant to be a way to show that the classifier output is not a result of a fluke or the imbalance of the dataset. By comparison, AUC of over 0.75 and sensitivity of over 0.8 are desirable metrics for a binary classifier [12]. Should CA classification become used in medical research it is important to make sure that the results are trustworthy, despite the non-deterministic nature of the computation.

Requirements of the system architecture

The intended system architecture is a pipeline consisting of several elements, each with a specified task:

- read information about the instance from the dataset and use it to create an initial setting for the CA. This can be achieved either by scaling the bitmap image of the mammogram segment and plotting a CA grid over it, or using details of pixel values of the image contained in a corresponding text file.
- evolve the CA rules using the evolutionary framework
- apply CA with evolved rules to the initial setting, based on the original dataset instance
- if the image acquired from using the CA is not a result on its own, translate the result into human-readable output/ classification statement.

The user will need to be able to analyze the process at each point; whether to check the initial setting of CA for each instance, to see the rules for CA updates and their evolution or how the CA modifies the mammogram image. The whole program is going to be written in JAVA programming language, due to the high availability of supporting libraries, especially the evolutionary framework ECJ.

Functional and non-functional requirements

To better understand how to describe the project, one has to understand their definitions of functional and non-functional requirements. For the purpose of this paper, functional requirements will be understood as specifying what the program should do. That means they describe the functions that the programs will have. Non-functional requirements will be interpreted as how the system performs while applying functions from the functional requirements, e.g. what limits its functionality or how fast the computations should run. Within the scope of this project most of the functional requirements overlap with the mandatory elements of the system architecture.
Functional requirements:
- Reading data from the dataset
- Creating a CA grid using this data
- Evolving CA rules
- Applying evolved rules to the CA grid created from the data
- Obtaining a result that is either a classification of regression statement
- Enabling the user to observe the process and check the CA rules
- Ensuring high accuracy of the program. While this notion may seem to be better described as non-functional, the success of this project depends solely on the results that CA based classifier will achieve. As mentioned earlier, anticipated accuracy (or similar metric) is meant to have a value of over 70%.

Non-functional requirements:
- Replicability of the results. CA is non-deterministic in its nature. That means that if a system based on it is meant to be used as a classification tool, results for an instance must be the same most of the time. Otherwise the program will not be viable for use, especially with medical data, where it can affect human lives
- Speed of the algorithm/computation. While that may not seem vital from the “proof of concept” point of view, it is important for any future work that the classification process computes with speed comparable to other methods used in the field
- Clarity of the output. One of the goals of this research is to create a tool that would be helpful to radiologists as a computer aided diagnosis method. To ensure this, the output of the program must be clear, e.g. an image modified by CA and a statement clearly classifying the cancer risk.
Literature Review

Cellular Automata Overview

Although the Cellular Automata were first used in 1950s by Ulam and Neumann [1], it wasn’t until John Conway’s “Game of Life” that has popularized the whole method [2]. It is a “zero player game” meaning that the evolution of patterns requires only initial input. Since then they have been used in many disciplines and fulfilling different roles. It has been proven to be Turing-complete [3]. Cook proved that the rule 110 elementary CA is universal, thus able to compute anything that can be expressed as an algorithm.

At its core, Cellular Automata consist of a grid of cells [4], laid out in any number of dimensions, usually two. Each of those cells can be assigned any of the finite number of states, e.g. “on”/“off” or “alive”/“dead”. After the initial setup of those states, in each time step the grid is updated. The updating happens according to a set of rules corresponding to the state of the current cell and it’s neighbourhood. The rules could be fixed, like in the “Game of Life”, or ever-changing, when Genetical Programming (GP) is used to achieve an optimal ruleset capable of solving a given problem. Piwonska [5] has managed to use it for binary classification and showed that evolved rules were achieving higher accuracy then some popular human devised heuristics.

The possibilities of custom-crafting the CAs to one’s needs are considerable. One could create a grid of a different shape. Morita and Imai [6] were using a hexagonal grid to understand better CAs with reversible rules, meaning that each pattern could be unequivocally traced to its initial state. In some situations, hexes prove to simulate an event better than a square grid, like in [6a] where a hexagonal grid is used to model and predict a forest fire spreading in homogenous and inhomogeneous environments.

The shape of the automaton is not the only part of a CA system that could be changed. Making the state of the cell continuous instead of discrete is another idea. This could be used for image segmentation [7] when, after achieving initial partition, each area would start evolving on its own. This proved especially useful when new difficulties emerged, due to edge detection on black and white or grayscale image.

Similarly, when trying to optimize real values, discrete states may not be enough. Afshar and Rohani succeeded in creating a hybrid CA system to optimise a sewer layout depending on changing pipe diameters [8]. They have devised two approaches, one using a discrete approach and one using continuous states. Both versions proved to converge on a solution of chosen benchmark problems much faster than traditional approach.

Yet another change of basic CA qualities is introducing asynchronous updates/ different timesteps to the computation. Hamid and Meybodi have used this idea to create a learning cellular network, which was later able to devise algorithms to control input traffic [9]. Cells that were being overloaded with input “calls” would change their update rate accordingly and their neighbouring cells would help to spread the workload after being notified of a timestep change. A different study [10] was investigating the change of robustness of the system in which CA were subject to an asynchronous update rates. The results have showed that each new update approach resulted in an appearance of a brand new behavior within the system, opening new computational possibilities for future studies.
The rules regarding CAs behavior do not need to be deterministic either. Fuks [11] presented a probabilistic CA to solve a density classification problem. He suggested that his results might be leading to a paradigm of how a local interaction between two or more systems is able to compute global properties.

Cellular Automata are a very flexible tool, able to implement biologically inspired solutions. They could be coded in a way that uniquely suits the needs of a problem and can present many intricate results, emerging from a relatively simple system or set of tools.
Mammogram Classification Overview

Breast cancer is one of the main reasons of death for women over 40 years old in Europe [12]. In the United States, on average, one in twelve women will develop it during her lifetime [12]. Luckily, mammogram screenings have been proven to be the most effective method of spotting the cancer in its early stages [13]. They have been proven to reduce the mortality rates up to 70% [14].

Sadly, the task of reading and interpreting the mammograms is quite hard and time consuming - a minute detail can decide whether a biopsy will be needed or not. The radiologists are under constant pressure and any false positives are a very costly mistake. One of the most common measures undertaken to improve the accuracy is having a second reading - another person examining the photo. That on its own has been proved to increase the cancer detection rate by roughly 10-15% [13]. The next logical step is to implement Computer Aided Diagnosis systems (CAD). Those are programs that are made specifically to help the radiologist interpret mammograms, usually by marking or encircling any anomalies suspected of causing cancer. There are many programs commercially available, all with their positive and negative sides. Experts agree though, that as those systems become more and more advanced they will become a suitable alternative for a second reading [14] and a great tool to verify if the first diagnosis may contain an error.

There are four main anomalies that one needs to look for when trying to determine first, the presence and second, the malignancy of cancer. They are usually separated into masses, calcifications, bilateral asymmetries and architectural distortions. Masses are the areas where the density of tissue changes for no apparent reason or it has an unnatural shape. Calcifications are small calcium deposits, usually grouped. They develop naturally as the woman ages and are often benign. They tend to be the easiest ones to identify as they show up as highly contrasting white spots on the photo. Bilateral asymmetry is defined as a presence of greater volume or density of breast tissue without a distinct mass, or more prominent ducts, in one breast as compared to the corresponding area in the other breast [14]. The architectural distortions are usually the hardest ones to detect - they are simply a distortion of a normal architecture of the tissue in a breast.

Most of the CAD systems work in a relatively similar way. They analyze the whole image pixel by pixel, looking for spots where the properties of a pixel change rapidly. That is why anomalies like masses or calcifications are the easiest ones to identify for a computer. When the program analyzes the mammogram, it recognizes them as areas with sudden change of contrast. That is why most of the researchers focus on finding an effective method to identify the other cancer markers. One of the problems with CAD is that while some of the methods rely on image enhancement, it has been proven that it often leads to misdiagnosis [13], the biggest problem being the change of contrast in the photo.

With all this in mind, everybody can agree that while a computer system cannot replace a skilled radiologist, its help in verifying an expert’s opinion proves invaluable. CAD has been proved as an effective tool in detecting cancer in its early and treatable stages.
The Game of Life

In 1970, British mathematician John Horton Conway created a CA known as the Game of Life, also known simply as Life [4]. It became famous instantly, praised for the complexity it offered and opening new ways to look at evolving patterns while remaining simple at its core. A cell can be either alive (on) or dead (off) and there are four rules that describe the behavior of the automata in each timestep [1]:

- Alive cell with fewer than two live neighbours dies due to underpopulation
- Alive cell with more than three live neighbours dies due to overpopulation
- Alive cell with exactly two or three live neighbours continues to live
- Dead cell with exactly three live neighbours becomes alive.

One of the most notable features of the game are the emerging patterns. The most popular of them are Still Lifes, Oscillators, Spaceships, Guns and Methuselahs.

Fig. 1 Beehive still life [http://conwaylife.com/w/images/3/3c/Beehive.png]

Still lifes are the simple static patterns that do not change unless their state is changed by the user or its environment.

Fig. 2 Pulsar oscillator [http://conwaylife.com/w/images/4/49/Pulsar.png]

Oscillators alternate between several configurations within a certain period.
Spaceships are patterns similar to the oscillators but they move through the grid, the most famous being the “glider”.

Guns are static patterns that create spaceships, called that because they tend to look like a gun shooting spaceships [4].

Lastly, methuselahs are patterns that start of simple and “live” for a very long time, before becoming static or dying. The best example of this type is known as the “acorn”. It starts off with 7 alive cells and over the period of 5206 timesteps grows to the size of 1057 alive cells, spawning 144 patterns [4b].

Behaviours like this cause great interest within the scientific community. Researchers from different fields find the game of life useful in their work. They find this useful because it demonstrates emergences of patterns and self-organizing. Philosopher Daniel Dennett used the game to illustrate the idea of consciousness and free will emerging from simple set of deterministic laws [4a]. Another famous use of the life CA is the Golly program. It is an open source software praised for the speed with which it can compute complicated patterns, the most notable being its version of the Universal Turing Machine [4]. The proof of game of life being Turing complete makes the whole system much more interesting from the theoretical point of view- after all it is able to compute everything that can be expressed as an algorithm. One can build logic gates using gliders, or create self-replicating patterns.

While this system may not be the most convenient way to compute, it cannot be denied that it made the CA much more accessible and famous. Its appearance created brand-new types of
programs, called “simulation-games”, which try to recreate life-like behaviours using simple rules and programs. This has opened up new possibilities in research and given an array of different people a whole new set of tools. Overall it is considered a scientific phenomenon.
Different uses of Cellular Automata

Scientists have found a multitude of uses for CA. In this section of the report I would like to describe some of the more interesting ideas. These range from ones that are implemented in programs that researchers and students all over the world use every day, to some that have managed to be the perfect solution for a specific problem.

One of the first popularized ways to use CAs was modeling. Probably the most famous example being the model of forest fires spreading. These simulations have become more intricate over the years. One of the original models was created in 1997 [15]. The scientists have defined their problem as predicting the front of the fire. The variables given were the rate of fire spreading, knowing where fire is at a certain moment and velocity and direction of wind. They investigated how the weather conditions and topography affect the flames and if homogenous or inhomogeneous outline of the forest is a major factor during the fire. The conclusion stated that the model was “in good agreement with the experience on fire spreading in real forests” [15]. This model has become the basis for algorithms simulating real forest fires and has been used and improved upon by researchers for the past 20 years.

One of the more popular ideas to improve it, would be to join the CA system with the geographic information system (GIS). That way the CA grid is spread over a factual map of the forest and incorporates topographical features. Such integration seemed to be a daunting task due to the rather static nature of GIS. A recent study [16] has tried to overcome these problems and create a tool with a friendly user interface and an easy way to incorporate changes in maps and conditions by the developers. This model was evaluated against historical data and the Prometheus system (which is “the national Canadian fire behaviour modelling tool based on elliptical wave propagation principle”) [16]. The new system has proved to be very accurate, especially in preventing overestimation of the fire spread area [16].

A completely different use of CA focuses on one-dimensional versions, called Elementary Cellular Automata. Stephen Wolfram started investigating these automata and would find many
uses for them. Later he would even implement some of these uses into his program Mathematica. His book “A New Kind of Science” [17] claims that simple programs may express complicated behaviours and in turn, some of those can generate patterns that could be observed in the natural world. One can use those simple programs to study these complex natural systems. A quite famous example would be the “Rule 30” CA, which has a space time diagram resembling a pattern on a sea snails shell (Fig. 7). It is used as a random number generator in Mathematica [17].

![Fig. 7 A comparison of a pattern on conus textile shell and rule 30 space-time diagram](http://artfail.com/automata) Photographer: Richard Ling

But what does “space-time diagram” mean in the context of CA? It is a way of showing how CA rules behave in a certain setting. It shows step by step what happens within the system. For the elementary CA, the top of the diagram is the initial setting. Each row below it is the next timestep. The elementary CA rules are based on neighbourhoods of three. The state of a cell in the next step depends on the state of itself and its nearest neighbours. That means that there are $2^3=8$ possible patterns that can be either 0 (dead/white), or 1 (alive/black). That leads us to $2^8=256$ possible rulesets [17]. Each of those rulesets had to be tested and analyzed in order to find any useful information or utility.

Another famous elementary CA rule is the “Rule 110”. It has been proven to be one of the simplest Turing complete systems known to man. There are studies [18] showing how one can use it for computation thanks to interaction between colliding gliders (patterns mentioned in the Game of Life chapter).

Another interesting use of CAs is image encryption. Security while communicating via the internet is a top priority. Often people would send images that would include sensible information. Image encryption could be used to “scramble” an image and make it look like static unless somebody would possess a key to decipher it (Fig. 8).
Whilst CA was already used for image security and encryption, a recent study [19] explored a new simple approach. It used elementary CA [17] which perceived the whole image as a matrix of integers. Each cell would be assigned a different value depending on its nearest neighbours. The method would only work for black and white or grayscale images and the possible values for each cell would span from 0 to 255. The key to encoding and later deciphering the image would be based on a pseudo-randomly generated attractor. An attractor is a set of values or a state to which the system tends to shift or evolve. Using it in this context would mean that the user would always know what the original value of a cell was after it was changed using a certain attractor. For better security, the image could be split into several areas, each of which could be encrypted using a different attractor [19].

Another recent study has created a cipher method by combining the Game of Life CA and Chaotic maps [20]. The latter is a concept that utilizes the mathematical aspect of the Chaos theory to create ciphers. Its complex nature is somewhat of a setback, but the researchers aim to solve it by linking it with CA. The algorithm first creates a chaotic map and uses CA to evolve it. Both the new map and image we want to encrypt are converted to matrices with values describing colour of a pixel. A separate algorithm uses those matrices to generate the encoded image. The researchers claim that although the computation speed is not compromised, their cipher is extremely secure and sensible with $10^{150}$ possible keys.

CA with its grid architecture has a natural affinity for processing patterns that are organized within a pixel grid. In 2014 a trio of scientists tried to gather information about the most popular uses of CA on images [21]. They have described a multitude of applications: image compression and resizing, edge detection, mapping, skeletonizing (a skeleton of a shape is a thin version of that shape which is equally distant to its original boundaries and often helps describing the topography of it), erosion (“eroding” the edges of a shape), dilatation (adding on another layers of pixels on top of it), forgery detection, image retrieval, noise reduction, segmentation, colour and shape matching, shading and even the recognition LEGO bricks [21].

Edge detection deserves a special mention. For a computer, an edge on an image is a variation of intensity of pixels at a specific position [21]. Edge detection is important for many tasks performed by computers, like pattern classification, object recognition or medical diagnosis [22]. CAs have recently gained popularity within this branch of science due to simplicity of the model.
Amrogowicz and Zhao [22] came up with a new model that uses outer totalistic rules to detect edges. Similar to the Game of Life, totalistic rules update the state of the cell based on the number of active/alive neighbours. Outer means that the state of the cell itself does not influence its own update (as opposed to inner totalistic) [22]. The study devised a set of the most useful update rules and evaluated them against a set of the most popular edge detection algorithms. The tests were conducted with both clear images and images affected by salt & pepper noise (pixels in the image may randomly become either black or white) or Gaussian noise (value of a pixel may be changed within the uniform distribution). The totalistic were able to find edges for even up to 5% noise when regular methods would fail. The outer totalistic rules have proven even more noise resistant.

CA have a considerable potential. There is next to no constraint on what one can model using them. Their architecture and rules can be adapted to any task. The examples described are either new and inventive ways to utilize CA or present classic problems that help to understand some of the basic concepts of CA.
Genetic programming and evolutionary algorithms

Computer science often takes inspiration from biology. The idea that the code may improve itself, just like an organism evolving, has led to genetic programming (GP) and evolutionary algorithms (EA). Both notions are intertwined. GP is a technique with which we evolve a population of computer programs [23]. We transform stochastically generation by generation, hopefully into better programs. The random nature of this process enables us to avoid the traps which deterministic methods might be caught by [23].

EAs are a set of metaheuristics that GP use to change its population [24]. That means that EA decides which elements of the population will evolve or stay in the population and which will be discarded. Quite often they will introduce a fitness function - a rule or equation that describes how well an element of the population is able to solve its purpose. A good example would be the Santa Fe Ant problem [27]. It describes a situation where an ant is placed on a grid with food scattered randomly. The program wants the ant to find all the food as quick as possible. In this example the fitness function could be the amount of food found. If all the food is found then it would be the amount of time spent looking for it.

The two people known for their work in this field are John Koza, with his book “Genetic Programming” [25], and Melanie Mitchell, with “An Introduction to Genetic Algorithms” [26]. Koza came up with one of the most successful and widely used forms of GP, known simply as trees.

Fig. 10 A tree representation of a mathematical equation [25]

Fig. 10 is an example of a simple tree program. It computes an equation \((A \times B) + C\). We can add more functions to the “realm” of our program. Those can be mathematical functions if our program needs to find a solution for an equation, or something more abstract, like turn left or right and check for food in case of the Santa Fe Ant.

Mitchell, in her book [26], described the principles of GP. These include the theoretical foundations behind it and how GP and EA can be used for problem solving and modelling as a scientific method. One of the chapters is dedicated specifically to evolving CAs. She conducted tests in which several sets of random CA update rules were treated as the GP population. Those rules would later be a subject of cross-over (two randomly chosen “parents” from the population have their properties mixed together to produce a “child”) and mutation (subject properties would be randomly changed, using a predefined distribution) [26]. This evolution would continue for a set number of “generations” (timesteps). To test this population of rules, a set of problems would be generated and presented as CA initial configuration. CAs would compute and find a solution for each set of rules that would then be examined with fitness functions corresponding to each problem [26]. The process would be repeated over and over until one set of rules would be able to converge on the solution, or until a number of generations has passed without satisfactory results.
There is one very important problem to be considered when using GP. How to present the problem and the code, so that we will be able to evolve it and receive results that are valuable. This is known as encoding or GP language representation - translating the code into “chromosomes” that we will be able to evolve and which later could be translated back into compilable code. One of the most popular encodings is the Tree-based GP introduced by Koza [25].

There are two characteristics we have to be aware of when trying to evolve our programs. The first is evolvability - the capacity of our program to become better due to evolution. This is why people are mostly evolving simple programs, like the ones meant to solve the Santa Fe Ant problem or optimize an equation, unlike programs like Matlab. The second characteristic is expressiveness - how easy it is to express different kinds of behaviours (if it is possible to do at all). Usually when choosing the encoding we have to decide on which trait we want to focus on in our program as they tend to be on the opposite sides of the same spectrum.

While writing the code for GP and EA is a viable option, there are many frameworks that are easy to implement, written specifically to help with evolving code. One of them is ECJ - a Java-based Evolutionary Computation Research System [28]. All of its files with full documentation are available online [28]. It is easy to use, has a host of features, is compatible with other GP oriented coding languages (e.g. Push) and, due to being written in Java, if its basic version is not sufficient a skilled programmer can write his own add-ons. Many common problems are already implemented in it and experiments can be run straight from the command line, without writing any code at all.

Genetic programming is a very powerful tool. It explores the idea that, instead of finding a solution, one can write a program that will repurpose itself to find the best answer, as long as we are able to express the task and know what kind of metrics to use to examine it. Finding a proper evolution method is also a daunting task. While cross-over and mutation are the most common approaches, the Essentials of Metaheuristics [24] describe 138 algorithms that can be used for this. Biologically inspired computation has lots of potential that just needs to be properly incorporated into research, to discover amazing solutions, not achievable with a human devised heuristic.
Related work

As mentioned in the abstract, classification is one of the problems that CA is hardly used for. Mathematical solutions are great for flat data, but researchers are still looking for methods viable for image classification. Luckily CA pixel grid affinity can prove extremely useful.

A study by Piwonska, Seredyński and Szaban [5] undertook the task of using CA for binary image classification. Their images consisted of a line being a plot of a function. The program would need to classify whether a point is under or over the line. The training set consisted of a set of points scattered across a 2D area. Each point would be assigned a class, “1” or “2”, accordingly over or under the plot line. The automata would be stretched over the area, first with 10x10 grid, later with 20x20.

Each automaton would be assigned one of three states, depending on the points of which classes were located within it. This process would follow four arbitral rules [5]:

- if there is no point in a cell (an empty cell), then a cell is in state 0 (a cell is marked in gray color),
- if there are points only from class 1 in a cell, then a cell is in state 1 (a cell is marked in white color),
- if there are points only from class 2 in a cell, then a cell is in state 2 (a cell is marked in black color),
- if there are points from both classes (class 1 and 2) in a cell, then a cell is in state 0 (a cell is marked in gray color).

![Fig. 11 a) instance of a problem mapped into CA b) Initial configuration of the CA derived from the mapping [5]](image)

Fig. 11 present the initial configuration of one of the test problems. The domain of the problem is a 2D area [0,1]x[0,1] populated with points, each with coordinates defining its placement in the area.

Once the initial configuration for the problem was obtained, GP was used to evolve rules for CA updates. Each rule would be encoded as a string consisting of “0”, “1” and “2”. The index in the string corresponds to the updated state of the cell depending on the current state of its neighbours. This is described by the authors in a following diagram (Fig. 12).
These “rule chromosomes” were evolved using one point cross-over and mutation with probability $p$. A random point on a length of a chromosome would be chosen. Values until that point would come from the first “parent” rule and after from the second “parent”. Afterwards each value in the chromosome would have a chance $p$ to mutate into a different value. Those rules would be tested and evolved for several generations which would allow the best set of rules to emerge. Those would later be used to evaluate the CA method against some of the popular classification methods.

The results of this study showed that the best rules proved to have higher accuracy than human designed methods (e.g. k-NN). The rules proved to be scalable, rules evolved on the 10x10 grid were working accurately on the 20x20 grid. Nevertheless, in the critical evaluation of those results one must remember that those methods were designed to solve a particular problem and they are not universal.

A more recent study concerns a very specific issue. Using conditionally matching rules (CMR) for evolving CAs [29]. The author defines CMR:

“A conditionally matching rule (CMR) represents a generalized rule of a transition function for determining a new cell state (in view of the table rules). Whilst, the basic transition rule specifies a new state for a specific combination of states in the cellular neighborhood, a single CMR may cover more than one combination. A CMR is composed of two parts: 1) a conditional part and 2) a new state. The number of items (size) of the conditional part corresponds to the number of cells in the cellular neighborhood” [29].

The condition would usually be a function evaluating the state of the cell’s neighborhood and returning a Boolean value (true/false) depending on if the condition was met or not. The rule would update the cell’s state depending on the set of those values.

The case studies included pattern replication for 2D CA and square calculation for 1D CA. The tests were meant to check whether the CMR would be able to solve multistate CA problems for which the conventional approaches have failed. The EA managed to provide each case study with a solution, what might be an argument suggesting CMR robustness. One of the more curious results was the fact that the algorithm was able to come up with several working methods for the pattern replication and achieve so called “diagonal replication”. This is a technique used to reduce the number of transitions/updates needed for the replication. This technique enabled the CMR devised ruleset to create 528 instances of a pattern in the same amount time (200 timesteps) that an algorithm known as the Byl’s loop created only 79 [29]. Achieving diagonal replication is a significant step in this discipline and allows researchers to solve more problems faster and more efficiently.
Attempts to create a multidimensional CA classifier has resulted in the creation of Cellular Automata-based Classifier (CAC) [30]. The research group responsible for it uses an elementary CA to choose a ruleset for CAC depending on a nature of a problem. This system is called Decision Support Elementary Cellular Automata (DS-ECA) [30]. The premise of the entire system revolves around the idea that a complicated problem and the dataset could be expressed as a set of matrices and the rules classifying new instances as vectors. EA is used to evolve the rules, while the CAC learns using the training set.

The evolved “rule-vectors” can later be applied to the training set (mapped onto a matrix). The result is a vector of values corresponding to the class numbers (index in the vector being the number of test instance). After running tests on both binary and multidimensional (up to 56 attributes), the algorithm proved to give results comparable to other classification algorithms. The biggest problem for CAC proved to be optimizing the time of learning while using the training set [30].

Using CA for classification is a relatively recent idea. Thanks to GP the algorithm can learn any vital information from the data provided. CA, thanks to its architecture, can find patterns emerging from this data. Researchers are starting to explore the possibilities these techniques offer and the sheer amount of available approaches is astounding. The advantage of CA is the fact that it is not being held back by human based ontologies and due to its evolving nature, it is able to find emerging patterns no matter what discipline it is applied to as long as it is translatable to CA grid.
Methodology

Initial experiment idea

The main idea for evaluation of the classifiers is to use cross-validation. It is a technique, which assumes that we can split the dataset into several, separate folds. The program would then take turns learning, or in this case evolving, the classifier on a training set consisting of some of the folds and examining it on test sets made out of remaining folds. The idea behind cross-validation is to try and achieve a non-biased result by repeatedly testing the classifier on different sets of folds.

In some cases, the experiment would be computationally expensive, especially considering that GP is perceived as a ‘slow’ method. Within the scope of this project, running exhaustive cross-validation for each model (number of update steps of CA, types and parameters of evolutionary pipeline, fitness functions to classify a processed image) would be computationally expensive. To solve this problem, sets of parameters for different models would be instead tested on a single training and test set. This would speed up the process and streamline the process of choosing which models should be evaluated by cross-validation and which discarded.

ECJ description

ECJ is the system at the core of this project’s simulations and experiments. Its creator, Sean Luke, describes it in his manual as: “general-purpose evolutionary computation framework which attempts to permit as many valid combinations as possible of individual representation and breeding method, fitness and selection procedure, evolutionary algorithm, and parallelism” [28]. To understand how ECJ is utilised in this project, it is vital to first understand the evolutionary pipeline of ECJ and its elements.

The top level class in ECJ is called ec.Evolve, most of the experiments in this project are run from the perspective of this class. Its only purpose is to create and set up an instance of ec.EvolutionState subclass. The “owner’s manual” [31] provides a diagram describing the dependencies between the main objects of the pipeline (Fig. 13).
The parameter database helps to set up the system and contains information from the parameter files, e.g., type of problem, function sets, size of the population and number of generations. Mersenne Twister is the (pseudo) random number generator used by ECJ. Output is the part of the system responsible for the logging functions and printing the experiment results in the console. Initializer creates the population, which in turn stores an array of subpopulations that contain individuals: in this project those are equivalent to an encoded ruleset, using either a chromosome of integers or a Koza tree. Breeder is responsible for breeding the population in between generations. Exchanger is responsible for moving individuals between subpopulations. In this project, each experiment has only one subpopulation, making the exchanger obsolete. The finisher is the part of the system responsible for “cleaning up” the system after the experiment. Statistics contain all the functions that enable additional logging during the experiment.

The most important part of ECJ from the perspective of this paper is the evaluator. This subsystem is responsible for evaluating the population. The Problem is a java class that must contain the evaluate() method. This method contains the experiment itself. In this project, it reads from the training sets, converts them to a CA grid and applies the update rules contained in the individual for a set number of timesteps. After processing the image, it is assigned a fitness value, that varies depending on which model is currently tested. This process is repeated for each individual for a number of generation specified in the parameter file.

As some individuals will prove to be better than others, the population will be bred in a way that hopefully will create individuals that have better fitness, eventually creating a classifier that will prove useful during evaluation on the test sets.
How the breeding process will proceed, how big the population will be and how many generations the experiment will run for - all that information will be contained in the parameter files. The manual describes them as: “the lifeblood of ECJ: practically everything in the system is defined by them. This makes ECJ highly flexible; but it also adds complexity to the system” [31]. To better understand what kind of information those files contain, it is useful to read through an example one. The following text is a copy of one of the files used for the experiment, with additional comments (marked by colour).

```plaintext
# Copyright 2006 by Sean Luke and George Mason University
# Licensed under the Academic Free License version 3.0
# See the file "LICENSE" for more information
part of the ECJ licence statement

parent.0 = ../../gp/koza/koza.params
marking the problems as a koza gp task

#pop.subpop.0.file=pop.in
pop.subpop.0.size = 20
generations = 20
information about the size of the population and number of generations that the population will evolve for
the commented out pop reference, points to a file that contains evolved classifiers, it is used for checking the performance of an individual on a test set
tests are run with a population of size 1 and for 1 generation only

gp.tc.0.init = ec.gp.koza.HalfBuilder
gp.tc.0.init.min-size = 4
gp.tc.0.init.max-size = 8
gp.tc.0.init.min-depth = 4
gp.tc.0.init.max-depth = 8
constraints for creating initial individuals in the population, different for the chromosome encoding

# We have one function set, of class GPFunctionSet
gp.fs.size = 1
gp.fs.0 = ec.gp.GPFunctionSet
# We'll call the function set "f0".
gp.fs.0.name = f0

# Whether to use elitism, and how many elite to use
# Set to 0 to turn elitism off
breed.elite.0 = 1
elitism means that some of the individuals will be able to pass into the next generation unaltered, ensuring that the quality of GP will not drop in between generations

# The size of the tournament used by tournament selection
#select.tournament.size = 3

gp.fs.0.size =9
gp.fs.0.func.0 = ec.app.tutorial4.neighbour1
gp.fs.0.func.0.nc = nc0
gp.fs.0.func.1 = ec.app.tutorial4.neighbour2
gp.fs.0.func.1.nc = nc0
gp.fs.0.func.2 = ec.app.tutorial4.neighbour3
gp.fs.0.func.2.nc = nc0
gp.fs.0.func.3 = ec.app.tutorial4.neighbour4
gp.fs.0.func.3.nc = nc0
gp.fs.0.func.4 = ec.app.tutorial4.customFuncSet.AddX
gp.fs.0.func.4.nc = nc1
```
gp.fs.0.func.5 = ec.app.tutorial4.customFuncSet.SubX
gp.fs.0.func.5.nc = nc1
gp.fs.0.func.6 = ec.app.tutorial4.customFuncSet.IfLess
gp.fs.0.func.6.nc = nc2
gp.fs.0.func.7 = ec.app.tutorial4.customFuncSet.IfMore
gp.fs.0.func.7.nc = nc2
gp.fs.0.func.8 = ec.app.tutorial4.customFuncSet.Medium2
gp.fs.0.func.8.nc = nc2
the function set currently being used
#gp.fs.0.func.9 = ec.app.tutorial4.customFuncSet.Medium3
#gp.fs.0.func.9.nc = nc3
#gp.fs.0.func.10 = ec.app.tutorial4.customFuncSet.Medium4
#gp.fs.0.func.10.nc = nc4
#gp.fs.0.func.11 = ec.app.tutorial4.neighbourSelf
#gp.fs.0.func.11.nc = nc0
commenting out functions currently not used
#gp.fs.0.func.10 = ec.app.tutorial4.Sin
#gp.fs.0.func.10.nc = nc1
#gp.fs.0.func.11 = ec.app.tutorial4.Tanh
#gp.fs.0.func.11.nc = nc1
#gp.fs.0.func.12 = ec.app.tutorial4.Tan
#gp.fs.0.func.12.nc = nc1
#gp.fs.0.func.13 = ec.app.tutorial4.Cos
#gp.fs.0.func.13.nc = nc1

# Add our statistics object
stat.num-children = 1
stat.child.0 = ec.app.tutorial4.crossValStatistics
stat.child.0.pop-file = pop.stat
stat.child.0.info-file = info.stat
declaring the statistics object and the name of the output files
originally used for obtaining individuals for revaluation, later discarded
#eval.problem = ec.app.tutorial4.regressionCAFold1
#eval.problem = ec.app.tutorial4.classCAFold1
eval.problem = ec.app.tutorial4.classCAROC
eval.problem.data = ec.app.tutorial4.DoubleData
declaring what kind of problem will be evaluated and what kind of data it will use
in this case DoubleData is simply a value of type double, formatted in a way that
makes it usable by the ECJ gp package
Description of the breast cancer dataset

The dataset was provided by Dr. Stephen Smith from the University of York and includes 135 images (sample one – Fig. 14) which correspond to 76 text files. Each of them is a piece of information about a small region of interest segmented by the radiologist from the original mammograms. Each instance usually consists of three files, although some text files have more than one corresponding image (to show off different markers). Their description as provided by Dr Michael Lones:

Three types of file per mammogram:

i) <origin><patient no.>-<breast>-<view>_<marker>.txt - summary of marked up regions

ii) <origin><patient no.>-<breast>-<view>_<marker>_output.txt - details and pixel values of marked up regions

iii) <origin><patient no.>-<breast>-<view>_<marker>_output.bmp - bitmap image of (ii)

Where:
<origin> is either Cam - from Cambridge, or Nor - Norwich
<patient no.> is a unique number for that origin
<breast> is either L or R for left or right
<view> is either CC (view from above ‘Cranial-Caudal’) or MLO (oblique or angled view called ‘mediolateral-oblique’)
<marker> the radiologist

Fig. 14 A sample image from the dataset (Cam020-R-MLO.bmp_FCanavan_MarkedRegion_1_Output)
Implementation

Development process

The goal of the project was to create a program capable of several actions:

- Reading date from the provided dataset
- Creating grids of cellular automata based on the obtained data
- Evolve a classifier
- Apply the evolved classifiers to test sets
- Present/Analyse results.

First, a decision on the programming language had to be made. The author has decided to write in Java, for two main reasons. First, the evolutionary framework ECJ is written in and designed to work with Java. Second, the author was not fully confident in his skills in this language, so it was a great opportunity to learn and expand his skill base.

The creation of the program was meant to follow the plan created in the research report. First a CA framework was meant to be created and then linked to the ECJ. After this step the experiments with evolving different classifiers could commence. Originally, each model was meant to be fully completed and tested before moving to the next one. The process proved to work better while the models were created and tested in parallel, often inspiring new improvements in the CA or experiment framework.

The CA framework was split into several packages, each serving a different purpose:

- cellUtilities – included classes that represent cells of the CA, complete grids of CA, file readers and file writers. The core of the CA framework.
- neighbourhoodStrategies – classes utilising the strategy pattern, used for retrieving neighbours of a CA cell in a different way. In the end, those proved to make next to no difference for the results of the experiments. Von Neumann neighbourhood was used in all the later tests, due to its simplicity and being computationally ‘cheap’ (only 4 neighbours).
- updateStrategies – classes utilising the strategy pattern, used for updating the CA grid. Those correspond to classifiers and rulesets utilising the chromosome encoding.
- evolutionaryUtilities – classes meant to be a part of the ECJ evolutionary pipeline. Mostly classes concerning calculating the fitness value of an individual/classifier. It includes fitness context and strategy classes and the basicProblem class which is a discarded idea, meant to be the basic template for the evolutionary simulation. This package was created after setting up ECJ in the Eclipse environment.
- tests – package including most of the tests for the CA framework. Those are all classes with runnable ‘main’ classes. Idea to use Junit testing instead was considered not to match the requirements of the tests- Using ‘main’ classes was much more straightforward and quicker.

During creation of the framework, the first encountered problem was deciding what kind of data to use. The dataset provided included a couple of files per each instance, the most important being a text file with full information: the radiologist classifying the image, title of the image, timestamp, classification, dimensions of the image and pixel values of the whole image. The second important file was the image itself, saved in a bmp format. Most instances had a corresponding third file- a short summary including information from the first file without the pixel values or dimension of the image.
The question was, whether to use only information from the text files, or to incorporate reading pixel information from the actual images. After running some tests (included in the test package), the methods using only the text file proved to have shorter runtime and be more intuitive to use. Some of the code that acquired pixel values from the images was inspired by examples found on the stackoverflow.com website [https://stackoverflow.com/questions/6524196/java-get-pixel-array-from-image], but those classes were discarded and not used in any of the experiments.

Using only the text files for acquiring pixel values had one minor drawback. The image created using the text file data was not identical to the original picture. It would be a couple of shades darker but retain its features (Fig. 15).

![Fig. 15 The original image provided in the dataset (left) and the one created using data from the corresponding text file (right)](image)

After consulting the problem with the supervisor, it was decided that the image being slightly darker is not a problem, especially since all images should be normalised before applying CA to them. Each image in the dataset should be normalised to the same range of pixel values. Due to the nature of the data (cut-outs of a photo selected by the radiologist instead of the full photo) and different origins of parts of the set, non-normalised data would likely return abstract and non-meaningful results. Different boundaries were tested out for this purpose. Finally, it was decided that the best option would be using the global minimum and maximum pixel values from the dataset, as they were assumed to transform the images the least, as opposed to e.g. normalising the picture to the full gray-scale range (Fig. 16).
After running some tests (overallMinMaxTest.java) it was established that the lowest brightness in the whole set is equal to 60, and the highest equal to 238. The overall scale available in the Java buffered image is 0 to 255 for each of the main colours (red, green, blue) and to achieve a grayscale image value for each colour must be the same. The following equation was used to normalise pixel brightness:

\[
\text{normalisedBrightness} = \left( \frac{(\text{currentBrightness} - \text{oldMinimum}) \times (\text{newMaximum} - \text{newMinimum})}{(\text{oldMaximum} - \text{oldMinimum})} \right) + \text{newMinimum}
\]

Where old* values are the minimum and maximum pixel values of the image being processed, and new* values are the ones that the image is being normalized to (60-238). In the program this is done by calling the normalizeMap() or customNormalizeMap() methods on the cellMap object.

Next step in the development was creating the ways to access neighbours of a CA cell. To achieve this, a strategy pattern would be used. That way, the code would be flexible enough to allow adding more strategies later in development. For each strategy ‘family’ there would be an interface (describing the signature of the main method of the strategy), context (a class that would include all the utility methods, the main method would be executed from its level) and strategies (classes including the code for different main methods, e.g. retrieving neighbours, they would be passed on to the context class as one of its instance variables). Each neighbourhood strategy would return an array of cells, index of which would determine the position of the neighbour (Fig. 17). The boundary condition for each neighbourhood was initially the same. Should one of the neighbours of the cell be ‘outside’ of the picture (cell being on edge) an array of nulls would be returned and the cell would not be affected by the CA.
Some of the early tests (testSetExperimentTest) have shown that the type of neighbourhood is of next to no importance to the models used in the experiments. It was decided that only the von Neumann neighbourhood would be used, since it was the smallest one and bigger amount of neighbours would likely cause bloat while evolving the classifiers (the individual becoming excessively complicated and big without an improvement in performance).

With the CA framework ready, the ECJ had to be set up. The documentation [28] suggests two ways to do this: integrate ECJ into the ready program or treat the program code as a library for ECJ to use. The latter option was chosen - the ECJ manual suggested it as easier and less time consuming option. Initially, the experiments were meant to be running on a Linux virtual machine, ECJ being written to work under Unix systems. Luckily, ECJ could be also run in the Eclipse environment if it is properly configured. To use it on windows the user has to:

- Create a new project using the ECJ folder as a root directory
- Import the file system using the ECJ\ec folder and its contents
- Compile/build the project

To run the experiment itself the user has to:

- Run configurations with ec.Evo as the main class
- Put any parameters that would by written after the run command in the unix terminal into the arguments box, e.g. “-file ec/app/tutorial1/tutorial1.params” if the user would like to run the first tutorial experiment provided by ECJ.

Before running the simulations, a way to evaluate the individuals had to be added- to enable the evolution pipeline to work correctly. Each individual simply has to be able to return a fitness value- in the case of this project, this would mean creating a function, that would return a fitness after being fed the grid of evolved CA. The supervisor suggested starting off with simple functions, e.g. average pixel value of an image or brightness of a pixel in the middle of an image. New ways to judge individuals were meant to be explored later in the project. At this point the evolutionaryUtilities package was created.

Following stage included writing the problem classes for the experiments, function sets and parameter files. Some of the files were created on the base of files included in the ECJ library. Tutorial1 was used.

Fig. 17 Three basic types of neighbourhood used in the framework. 1 is Von Neumann neighbourhood, 2 is Moore’s and 3 is the extended Von Neumann neighbourhood.
as a basis for the chromosome encoding model, which treated the problem as a regression task. Tutorial 4 was utilised for the Koza Tree GP model.

The next problem was finding a way to revaluate the evolved individuals. ECJ accounts for evolving an individual, what equals to teaching a classifier in this situation. For the purpose of the project, a way to reuse the learned classifier had to be found. The chromosome encoding was the easiest one to retrieve. The signature of the update method had to be changed to allow passing the evolved rules as an instance variable. While training, this variable would be overridden by the genome variables provided by ECJ. To test an individual, one would simply need to input the rules chromosome into the update method.

Reusing Tree individuals proved much more difficult. Because trees are stored in ECJ as a tree of nodes, equivalent to classes from the function sets, they can’t be simply used outside of the framework, due to restrictions of how methods in the function set have to be written. Most online resources suggested creating an ‘outside’ copy of the set and writing a parser that would be able to read the printout that ECJ provides as a stringified version of the individual. Luckily, there seemed to be a simple workaround. ECJ allows the user to run the experiments with a pre-set population. Using the in-built statistics package, the program is able to save the population of the experiment after the classifier has been taught. The file including the population could be later reused, after slight modifications, to start a new experiment. That means that to test an evolved classifier, one simply needs to start the experiment on a different set of data (test instead of training) with a population size equal to one that will run for one generation with the evolved individual as the input population. This solution while quite simple, means that the testing process cannot be automated- the user must prepare the population file in between training and testing. This is not a problem at this stage of the project, but for the purpose of any future work, the process would need to be automated.

The main method of analysing the classifiers is using them for cross-validation. It means that the dataset is split into separate folds, which are later combined into different training and testing sets. Its purpose is to assess the classifier in a hopefully non-biased way (cross-validation helps to limit the problems like overfitting- ‘bending’ or complicating the model to fit the training data better, causing the model to behave poorly on real data). The dataset was manually split into 8 folds in a pseudo-random manner. Said folds were later combined to create 6 pairs of training and test sets. Each of those pairs would be later used for running an experiment for a certain parameter file. Final result would later be the combination of the results for the pairs, e.g. average accuracy of the classifier from all 6 sets.

The supervisor has suggested simplifying the problem- instead of classifying the risk of cancer as a number from 2 to 5 (as the radiologists did in the dataset), the task should be restricted to a binary problem, the risk would be either high (true) or low (false). He also suggested creating the ROC curve based on the classifier output, and use the area under it as a metric for calculating the fitness value of an individual. The code for creating and plotting the curve was provided by the supervisor.

**Rule models**

The models of CA rules for this project were split into two main types: rules encoded as a chromosome comprising of integer values and rules built as a Koza GP tree. There was also a single rule ‘dummyStrategy’ used as a test method for the CA framework. The motivation for the methods used in the function sets and representing the rules came mostly from literature [5,29,30] and discussions with supervisor. The rules must capture enable CA to modify the values of pixels both mildly and in steep bursts and to make ‘decisions’ based on the values of the cell neighbours. The latter led to including the ‘condition’ aspect of the chromosome encoding and ‘decision’ function for the koza GP.
**dummyStrategy**

This update rule is non-evolvable and was used only as a test for the CA framework. It changes the state of the cell to an average of its neighbours states, effectively blurring the image (Fig. 18).

![Fig. 18 The original, normalised test image (left) and a version processed for 20 timesteps of CA with the dummyStrategy update ruleset (right)](image)

**Chromosome encoded rules**

A chromosome representing the rules was depicted as a set of integers. The index of each would represent which neighbour of the cell is affected by the rule with an exception of the last gene—its value would determine the threshold value, a constant used by the update rules. Apart from the threshold, each integer would correspond to a predefined rule. The evolutionary pipeline would later change those chromosomes either by mutating them, using crossover or both. Mutating means that each gene in a chromosome would have a set probability into mutating into a different one (25% in simpleThresholdStrategy). Crossover means that new individual would be created by getting two “parent” chromosomes, copying part of one into the new “child” and filling in the rest with the equivalent genes from the second “parent”. Only one set of rules was created and tested using the chromosome encoded rules.

**simpleThresholdStrategy**

This update strategy was the chromosome encoded ruleset that included 9 hardcoded rules. They were a combination of 3 methods and 3 conditions. For each neighbour, if the condition was met, the method would be called to return a value. Those values, should they be returned, would later be averaged and become the new state of the cell. This was meant to enable different behaviour of the CA depending if the pixel would be situated on an edge, or in the middle of an area with certain brightness.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) return value of the neighbour</td>
<td>(1) call the method if neighbours value is smaller than the original cell by at least the threshold</td>
</tr>
<tr>
<td>(2) return value of the neighbour plus the threshold</td>
<td>(2) call the method if neighbours value is bigger than the original cell by at least the threshold</td>
</tr>
<tr>
<td>(3) return value of the neighbour minus the threshold.</td>
<td></td>
</tr>
</tbody>
</table>
(3) call the method if neighbours value is within the threshold distance of the original cell.

Genes correspond to the following combinations:

<table>
<thead>
<tr>
<th>Gene</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Condition</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Fig. 19 The test image after being processed for 20 timesteps of CA with the rule “1 1 1 0 2” (left) and “8 3 1 6 13” (right)

This enables us to read the encoding. The meaning of the ruleset ”8 3 1 6 13” from the example above (Fig. 19) means:

- 8 - Return (value of the first neighbour – threshold) if neighbour value is within the threshold distance of the original cell value
- 3 - Return (value of the second neighbour + threshold) if neighbour value is lower then the value of the original cell by at least the threshold
- 1 - Return the value of the third neighbour if it is higher then the value of the original cell by at least the threshold
- 6 - Return (value of the fourth neighbour – threshold) if neighbour value it is lower then the value of the original cell by at least the threshold
- 13 - The threshold value equals 13
- The new state of the cell is the average of values returned by neighbours (not all neighbours need to contribute)

**Koza GP tree rules**

Tree rulesets were created by building trees out of classes belonging to ‘function sets’. Each of those classes would describe an available node in a tree, e.g. leaf nodes (no children) would be returning the pixel value of a certain neighbour of a cell, while averaging values of nodes would require a node with two or three children (depending on a method). The trees generated from those nodes would describe
a sequence of methods to be called on each cell in the grid. Two main function sets were created, with both having retrieving neighbours of a cell, or the cells own value, as their only available leaf nodes.

First one was based on the regression problem set provided by ECJ. Methods would be mathematical operations and the tree would resemble an equation with the neighbour brightness values as variables. Classes were used in different configurations, with the full set including 18 functions. This set has proved to have one slight disadvantage. The mathematical operations, e.g. retrieving the cosine function of a pixel value or raising it to the power of two, would modify the original pixel so heavily, that it would cross the boundaries of the possible brightness. Due to the boundary conditions such value would be overwritten by the minimal or maximal normalised brightness (60 or 238). Effectively, this function set while being capable of classifying instances, would return images consisting only of pixels with two possible values, with all the features of an original image lost (Fig. 20). Poor performance and the ‘black box’ nature of the results (with emphasis on abstract nature of the processed images) lead to discarding this function set and developing the second one.

The customFuncSet package includes functions describing GP nodes with methods inspired by the original chromosome encoding from the previous stage of development. After exploring how using the regression set changes the pixel values of an image dramatically, it was decided that a much subtler approach is needed. Functions belonging to this set would be only to add or subtract “1” from the value, average the value of two, three or four nodes and most importantly, decide which node, or branch, should be passed higher up to the tree and which one should be discarded. This was implemented by writing two simple functions: ifLess and ifMore. They would require two children nodes and respectively return the value of the one that would be lower or higher. The function set used for the later experiments discarded the methods averaging values of 3 and 4 neighbours- trees that included them proved to be prone to bloat.

The tree individuals evolved by ECJ would be represented as a stringified sequence of method calls (each function in the set has to have its own toString() method). The ECJ manual [31] states the default print-out in the output file is a LISP style equation, were command before a pair of parenthesis describes the operation on the elements within them. The example image above [Fig. cb] was created using the rule:

\[(\text{med2} \ (+1 \ (\text{if<} \ (\text{med2} \ (\text{if<} \ (\text{med2} \ n4 \ n2) \ (+1 \ (+1 \ (+1 \ n4)))) \ n3) \ (+1 \ (+1 \ (+1 \ n4)))) \ (+1 \ (+1 \ n4)))\]

Such an output may seem unintuitive at first sight. Luckily, ECJ provides a parser within its evolutionary pipeline that is able to read rules in this form and use them to initialize a new population. This would
become the main way to access the evolved classifiers for tests. Luckily, strictly for the output purposes, ECJ provides a way to show the tree individuals in a clearer way. One of the is printing out the individuals using dot language. This could be read using software such as Graphviz to create an image (Fig. 21).

Fig. 21 The graphic representation of one of the evolved rulesets created using Graphviz software
Results

In the final stage of the project, three models were created to be used in full cross-validation tests. One of them utilized a chromosome encoding, two other Koza GP trees. They would either interpret the task as a regression or classification problem. For the regression task, the fitness value of the processed image would be translated into a number between 1.5 and 5.5 and assigned the risk class based on the closest integer (2,3,4 or 5). The fitness of an individual would be measured by how close the translated output of the classifier would be to the integer describing the risk assigned by the radiologist. After obtaining initial results, this approach proved to produce weak classifiers. As suggested by the supervisor, the classification approach would be used. The fitness value of the image would be translated similarly as in the regression task, but instead would only classify the risk to be high or low.

Using such criteria has changed the original nature of the problem into binary classification. This simplification was meant to streamline the process of finding a correct model that would eventually be used for a non-binary problem. Firstly, the fitness of an individual would be based solely on the number of correct classifications (number of true positives and true negatives compared to the size of the test set). Metrics such as sensitivity and specificity would provide an extra set of information, but not be a part of the evolutionary process. Later, this approach would be changed, due to inability to judge which classifier was better, should they return the same number of correct predictions. The new way to interpret results would instead judge the individual rulesets by creating a ROC curve out of its classifications and measuring the are under the curve. The closer the size of the area would be to 1, the better the individual would be. For each model, the best individual of the evolution run would be later reevaluated on corresponding test set. While experiments for each model were repeated several times to assure that the results are reproducible, the results shown in the graphs and tables below come from simulations with the biggest number of generations and biggest populations.

Description of metrics

Correct Classifications (CC) and Correct Classifications Percentage (CC%) – number of correctly classified instances in a test set. CC% is the percentage derived from the number of Correct Classifications.

Accuracy – for binary classification accuracy is the rate of true positives and negatives to the number of instances in the test set

TP, TN – true positives and negatives, number of correctly classified instances in a binary problem.

FP, FN – false positives and negatives, number of incorrectly classified instances in a binary problem.

Sensitivity – rate of true positives to the sum of true positives and false negatives. It indicates how many of the relevant objects were retrieved. Binary problem only.

Specificity – rate of true negatives to the sum of true negatives and false positives. It indicates of the classifiers ability to detect the false, or in the case of this project low risk, class. Binary problem only.

AUC – ROC curve (Receiver Operating Characteristic) is used to describe the predictive capabilities of a binary classifier. One of the metrics it presents it the AUC – area under the curve. It is equal to the probability that the classifier will rank a random ‘true’ individual higher than a random ‘false’ one. In this case true and false are equivalent to respectively high and low cancer risk. The closer this metric to 1, the better the classifier is.

Mean squared error (MSE) – in the tables below the MSE between the classifier output and the factual cancer risk for a mammogram. Considering that the space of the problem ranges from 2 to 5, MSE of 1 would mean that on average the classifier is mistaken by a whole class. The lower this metric is, the better the classifier.
Standard deviation – a metric used to describe how dispersed are the values of instances in the set. In this case, it describes how consistent are the results of the classifier among different folds.

**basicFoldProblem1 problem and basicFoldProblem1 parameter files**

This problem uses a chromosome encoding to create a ruleset from a hardcoded set of rules (simpleThresholdUpdate), uses the mean average of the processed picture brightness as the image fitness. It is a regression classification task and the fitness of a ruleset is determined by the standard deviation of the classifier outputs computed for the training/test set. Cast version was created by casting the classifier output into an integer (using (int) method) to obtain the floor of the value. Non-cast version was obtained by assigning the risk of the closest integer, e.g. 3.4 would be classified as 3, 3.6 as 4, while both would be cast as 3. Later in the development a binary version was also tested.

<table>
<thead>
<tr>
<th>Fold</th>
<th>Cast CC</th>
<th>Cast CC%</th>
<th>Non-cast CC</th>
<th>Non-cast CC%</th>
<th>Binary CC</th>
<th>Accuracy</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12/29</td>
<td>41.38%</td>
<td>12/29</td>
<td>41.38%</td>
<td>19/29</td>
<td>65.51%</td>
<td>0.8814</td>
</tr>
<tr>
<td>2</td>
<td>9/28</td>
<td>32.14%</td>
<td>14/28</td>
<td>50%</td>
<td>20/28</td>
<td>71.43%</td>
<td>0.7389</td>
</tr>
<tr>
<td>3</td>
<td>11/29</td>
<td>37.93%</td>
<td>6/29</td>
<td>20.69%</td>
<td>17/29</td>
<td>58.62%</td>
<td>0.9432</td>
</tr>
<tr>
<td>4</td>
<td>10/28</td>
<td>35.71%</td>
<td>8/29</td>
<td>28.57%</td>
<td>14/28</td>
<td>50.00%</td>
<td>0.9197</td>
</tr>
<tr>
<td>5</td>
<td>12/29</td>
<td>41.38%</td>
<td>14/29</td>
<td>48.28%</td>
<td>18/29</td>
<td>62.19%</td>
<td>0.8179</td>
</tr>
<tr>
<td>6</td>
<td>11/29</td>
<td>37.93%</td>
<td>16/29</td>
<td>55.17%</td>
<td>21/29</td>
<td>72.43%</td>
<td>0.8061</td>
</tr>
<tr>
<td>Average</td>
<td>N/A</td>
<td>37.75%</td>
<td>N/A</td>
<td>40.68%</td>
<td>N/A</td>
<td>63.36%</td>
<td>0.8512</td>
</tr>
<tr>
<td>StDev.</td>
<td>N/A</td>
<td>0.035228</td>
<td>N/A</td>
<td>0.084183</td>
<td>N/A</td>
<td>0.084183</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**regressionCAFold1 problem and regressionCATestSet parameter files**

This problem uses a Koza GP tree to create a ruleset from functions reused from the ECJ regression package, uses the mean average of the processed picture brightness as the image fitness. The model was used to obtain results for both, regression and binary classification task. Similarly to the basicFoldProblem1 model, classification would be obtained in both cast and non-cast version. Fitness of a ruleset is determined by the number classification hits in relation to the set size after the classifier analyzes the training/test set.

<table>
<thead>
<tr>
<th>Fold</th>
<th>Cast CC</th>
<th>Cast CC%</th>
<th>Non-cast CC</th>
<th>Non-cast CC%</th>
<th>Binary CC</th>
<th>Accuracy</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6/28</td>
<td>20.69%</td>
<td>7/29</td>
<td>24.14%</td>
<td>15/29</td>
<td>51.72%</td>
<td>1.0201</td>
</tr>
<tr>
<td>2</td>
<td>0/28</td>
<td>0%</td>
<td>6/28</td>
<td>21.42%</td>
<td>13/28</td>
<td>46.42%</td>
<td>1.5714</td>
</tr>
<tr>
<td>3</td>
<td>0/29</td>
<td>0%</td>
<td>8/29</td>
<td>27.59%</td>
<td>15/29</td>
<td>51.72%</td>
<td>1.5172</td>
</tr>
<tr>
<td>4</td>
<td>7/28</td>
<td>25%</td>
<td>7/28</td>
<td>25%</td>
<td>14/28</td>
<td>50%</td>
<td>0.9916</td>
</tr>
<tr>
<td>5</td>
<td>0/29</td>
<td>0%</td>
<td>7/28</td>
<td>24.14%</td>
<td>18/29</td>
<td>62.07%</td>
<td>1.5862</td>
</tr>
<tr>
<td>6</td>
<td>9/29</td>
<td>31.03%</td>
<td>13/29</td>
<td>44.83%</td>
<td>18/29</td>
<td>62.07%</td>
<td>0.8450</td>
</tr>
<tr>
<td>Average</td>
<td>N/A</td>
<td>12.79%</td>
<td>N/A</td>
<td>27.85%</td>
<td>N/A</td>
<td>54%</td>
<td>1.2553</td>
</tr>
<tr>
<td>StDev.</td>
<td>N/A</td>
<td>0.143871</td>
<td>N/A</td>
<td>0.085485</td>
<td>N/A</td>
<td>0.065437</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**classCAROC problem and customSetCATest parameter files**

This problem uses a Koza GP tree to create a ruleset from functions written in the customFuncSet package, uses the mean average of the processed picture brightness as the image fitness. It is a binary classification task and the fitness of a ruleset is determined by the area under a ROC curve after the classifier analyzes the training set instead of its accuracy. It is important to note that AUC is not sensitive to class imbalance.
### Fold CC TP TN FP FN Accuracy Sensitivity Specificity AUC

<table>
<thead>
<tr>
<th>Fold</th>
<th>CC</th>
<th>TP</th>
<th>TN</th>
<th>FP</th>
<th>FN</th>
<th>Accuracy</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23/29</td>
<td>15</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>79.3%</td>
<td>1</td>
<td>0.57</td>
<td>0.90</td>
</tr>
<tr>
<td>2</td>
<td>19/28</td>
<td>15</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>67.9%</td>
<td>1</td>
<td>0.31</td>
<td>0.89</td>
</tr>
<tr>
<td>3</td>
<td>18/29</td>
<td>12</td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>62.1%</td>
<td>0.8</td>
<td>0.43</td>
<td>0.76</td>
</tr>
<tr>
<td>4</td>
<td>19/28</td>
<td>13</td>
<td>6</td>
<td>8</td>
<td>1</td>
<td>67.9%</td>
<td>0.93</td>
<td>0.43</td>
<td>0.66</td>
</tr>
<tr>
<td>5</td>
<td>22/29</td>
<td>16</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>75.9%</td>
<td>1</td>
<td>0.46</td>
<td>0.85</td>
</tr>
<tr>
<td>6</td>
<td>21/29</td>
<td>16</td>
<td>5</td>
<td>8</td>
<td>0</td>
<td>72.4%</td>
<td>1</td>
<td>0.38</td>
<td>0.74</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>72.88%</strong></td>
<td><strong>52.56%</strong></td>
<td><strong>20.32%</strong></td>
<td><strong>24.80%</strong></td>
<td><strong>2.32%</strong></td>
<td><strong>72.88%</strong></td>
<td><strong>0.955</strong></td>
<td><strong>0.43</strong></td>
<td><strong>0.80</strong></td>
</tr>
<tr>
<td><strong>StDev.</strong></td>
<td><strong>0.062169</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 21 Sample ROC curves of classifiers evolved in the cross-validation testing of classCAROC model**

The classCAROC being the model that had the best results, some additional tests were run to evaluate its performance better. Fig. 21 presents the ROC curves for the classifiers corresponding to the results.
in the table above. While rulesets evolved for some folds do not provide the same level of fitness (FOLD4), the model consistently has a curve above the diagonal line across the ROC space, meaning that even its weakest incarnation is better than a random choice. While this alone does not make the classifier “good”, its average AUC is 0.8, which according to the literature [12] is a satisfactory score.

Even with a working model, it is important to provide enough resources to the evolutionary process e.g. enough generations and big population. Fig. 22 presents a comparison of ROC curves for two rulesets evolved with different parameters and tested on the first fold. The “long” classifier was evolved with a population of 20 individuals over 20 generations, while the short with 10 over 10.

![ROC curves comparison](image)

*Fig. 22 Comparison of ROC curves for a ruleset evolved with different parameters, with (right) and without the confidence intervals (left)*

The dataset had a distinction between two types of markers being the possible cause of cancer, mass aberrations and calcifications. Showing what type of marker the incorrectly classified instances had could reveal some new dependencies within the model.

<table>
<thead>
<tr>
<th>Fold</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>Calcification</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>16</td>
</tr>
</tbody>
</table>

**Overall average accuracy of the classifiers**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>37.75%</td>
<td>40.68%</td>
<td>63.36%</td>
<td>12.79%</td>
<td>27.85%</td>
<td>54%</td>
<td>72.88%</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>0.035228</td>
<td>0.134263</td>
<td>0.084183</td>
<td>0.143871</td>
<td>0.085485</td>
<td>0.065437</td>
<td>0.062169</td>
</tr>
</tbody>
</table>
Evaluation

The results for the multi-classification problem are less than satisfactory, with the highest accuracy being 40.68% for the non-cast chromosome encoding. By comparison, with 4 different classes, a random choice would have a roughly 25% success rate. All those models tried to translate the average brightness of a picture to the risk class. This was done in a linear manner, meaning simply that darker images meant that the risk of cancer was lower, and brighter, that the risk was higher. This simple idea, corresponded to the appearance of markers like mass aberration on the original images. While assuming that such dependencies were linear proved to be an oversimplification, when the problem was changed to a binary one, the accuracy has increased, what eventually led to creation of the final model – classCAROC. It is worth mentioning that classifying an image not processed by CA at all would be accurate in 40-50% (binary) of instances depending on the fold.

The chromosome encoding while initially useful for testing purposes, was not fully investigated. Due to the simple nature of encoding, the genes/rulesets were mutated in a mostly random manner. Nevertheless, the rules provided allowed to make dark images dimmer, and bright images even brighter, enhancing any present markers. As Piwonska has shown [5] chromosome encoding for CA rules might provide spectacular results, but for a simplified version of a linearly separable problem – classifying images might need a more complicated approach.

The Koza tree GP with the regression function set has proved as effective as a random choice, with 27.85% accuracy with 4 classes to choose from and 54% for binary simplification. As mentioned earlier, assigning a class to a pre-processed image would provide similar results. Main reason for this might be poor accommodation of the regression function set to the realm of the problem. Those are mostly mathematical functions, which can add, multiply or return a value of a trigonometric function. They work if one is trying to describe an equation, but not when alternating a brightness of a picture. In java, to describe a colour, one needs a value between 0 and 255. Supplying a value outside those boundaries will result in an error while trying to save an image. With the boundary conditions of the CA framework, each extreme value would be reduced to a value acceptable by the image creator. Effectively, that means that each image will be reduced to noise consisting of only black and white pixels after only a couple of CA update steps, losing any resemblance to the original, and making the classification output random. Any further work on the regression model would require a better adjustment of the function set to the realm of the problem.

With such poor results, distinction between the cast and non-cast accuracies does not provide any interesting information. Both methods have accuracies below 50%. It is worth mentioning that the cast version favours the two middle classes (risk 3 and risk 4) while the other fits the linear separation of classes better.

The final model created classCAROC, was the first one to achieve the performance set as the initial goal of the project. Relatively high accuracy (up to 79%) shows that this model is capable of using CA to classify an image, having no other information than the picture itself and the risk that the radiologist has assigned to it (this might be one of the problems with the regression model- risk being a discrete and not continuous value in the dataset provided). This model has second best standard deviation out of all examined- meaning that the results achieved are consistent.

This model has one very important feature: high sensitivity. Sensitivity and specificity come usually at a tradeoff and one has to decide which is more important for a certain problem. With medical data, it is much better to raise a false alarm, than to omit a potentially life-threatening case. A popular argument mentions that to achieve a perfect sensitivity, it is enough to classify all instances as high risk, defying the purpose of the classifier in the first place. This model has a comparable number of true negatives and false positives, meaning that it might have problems with classifying instances at the border of high/low risk, but does not automatically assume that an instance belongs to a high-risk class.
As Fig. 22 shows, the classifier is able to improve quite quickly depending on the amount of resources provided for the evolutionary process. “Teaching” the short classifier took roughly 3 hours, while the long, which had twice as big population, twice as many generations and took 9 hours to compute. The length of the process might be a result of the hardware used to run the program, but it is important to acknowledge that the classifier was able to learn consistently. The replicability of results is further reinforced by the fact that in some cases, the evolutionary pipeline would come with different rulesets that would return exactly the same classifications for the whole fold.

The table showing what type of markers were present in incorrectly identified instances provides some interesting information. Twice as many errors were made while attempting to classify a mass aberration cancer than the one caused by calcification. With such a small dataset this information might not carry any significant information, but it confirms statements found in literature [12,13,14] – that calcifications in mammograms are much easier to spot both for human eyes and CAD programs. It may suggest that in any future work, classifiers should be able to distinguish patterns found in the image and not only evaluate the values of separate pixels.
Final discussion

Conclusion and reflection on the project requirements

This project was originally meant to be a proof of concept: can cellular automata be used as a classifier for images? Seeing that some similar research has already been done and proved that CA might be a valuable tool in such problems [5], this project aimed to tackle a more complicated problem. The task to create a non-binary classifier was not successful. The best result was achieved for a model with chromosome encoding: roughly 40% of instances classified correctly with 4 classes to choose. While the model and the classification criteria were streamlined to fit the scope and time frame of the project, it cannot be said that it fully explored the idea presented.

The analysis of the simplified problem provided much better results. The final model managed to achieve metrics comparable with results found in literature [12]. ClassCAROC reached 72.88% accuracy and sensitivity of 0.955 as a result of cross-validation. In authors opinion, this is a valid proof that CA can be a valid tool for image classification, especially if more of its capabilities will be explored and used.

In a project such as this one it is important to look back at the initial set of requirements, especially if it was meant to be a proof of concept. The pipeline created by linking the CA framework and ECJ fulfills most of the goals. It is capable of reading the data from the provided set and creates a CA grid based on this information. Several types of rulesets were created (one mutable, two evolvable in the final experiments) that could be utilized by the ECJ evolutionary pipeline. The results of classification are clear: the output is either a value in between 2 and 5 (either integer or a double) that corresponds to the malignancy of cancer, or a classification statement (risk of cancer high or low).

While the program enables the user to supervise the process, this aspect of the program was not fully developed. Each cell stores its state history, but nowhere in the project has it been used. The statistics classes of ECJ allow for detailed logging, but only some basic functions were utilized. At this stage, the project is still not user friendly, most of the changes in the experiments need to be made from the level of problem and parameter files. Creating visual output was only a part of tests and was not integrated into the final experiments for the sake of computation speed.

The amount of time that the algorithm would take to learn/evolve the classifier was meant to be another aspect to be taken into consideration. ECJ is a well optimized framework, despite genetic programming being perceived as an overall slow method. The CA framework itself and the way it was integrated into ECJ was not so well adjusted. The experiment would set up for each individual and torn down afterwards, repeatedly reading instances from the dataset and modifying them. ECJ provides ways to prevent such redundant coding, but due to the time constraints on the project and the author’s lack of experience using this framework, the experiment pipeline was left simpler and as a consequence- slower. Evolving the classCAROC model over 20 generations with a population of 20, with 13 steps of CA per experiment would take approximately 8.5 hours to complete. 10 generations and a population of 10 would compute between 2.5 and 3 hours. Such timespans would be unacceptable in a finished product, especially considering that such small populations and low number of generations would mark the results as highly unreliable. Luckily, the speed of the classifier during the testing was much higher. The algorithm would spend roughly a minute analyzing a test set of 28-29 instances.

The last important concerns were the replicability of the results and their accuracy. In a situation, where a program's output may influence human lives it is of utmost importance, that the classifier can be trusted and its results are not a fluke. Experiments were repeated many times with the same sets of parameters to establish that the results are consistent, especially important considering the small size of the dataset. While the evolved rulesets would not always be exactly the same, they would return
nearly the same output while examined on the test sets, despite the “black box” nature of the GP process.

The accuracy of the classifiers proved different for each model. Only the last model, classCAROC, had an average accuracy over 70%, thus fulfilling the requirements. Creating a predictive model that has reached the suggested threshold may suggest that a CA based image classifier might be a valid tool, once researched and developed further.

Should the author have a chance to start this project again, some design choices would have been changed. Initial focus would be put on creating a working model for binary classification and expanding it to multiple classes. More emphasis would be put on modular integration of the CA and ECJ framework. Running the experiments would be scripted and probably more than one computer would be used for this purpose. Results would be documented more carefully, along with the parameter files from corresponding tests- at one point results of cross-validation from the regressionCAFold1 were lost due to being overwritten by mistake.

**Future Work**

The idea of using Cellular Automata to create a classifier is worth further investigation. A simple model that used an average brightness of a processed picture proved to have accuracy of 72% and sensitivity of 0.955. Those are results for a binary problem, despite the original idea of creating a tool for non-binary classification. Any future work should focus on the following ideas:

- Use a different and bigger dataset
- Create new models, e.g. one that would utilize a multi-class ROC curves or a different type of GP (cartesian instead of a Koza tree) or ones exploring different aspects of CA (non-probabilistic, asynchonotic, different cell-shapes)
- Devote more resources to the evolutionary process, allowing for tests with bigger populations and more generations
- New ways to evaluate individual and more fitness functions than ones derived from the processed image average brightness
- Adjust function sets of existing models
- Script the cross-validation process to speed up the computation, currently experiment for each fold has to be started manually
- Optimize the CA framework
- Explore the idea of utilizing ECJ multithreading facilities
- Change the code within the experiment pipeline to be more modular.
Bibliography and references


Appendices
Appendix 1 – Risk assessment ........................................................................................................................................... 50
Appendix 2 – Cross-validation result files .......................................................................................................................... 51

Appendix 1 – Risk assessment form

MACS Risk Assessment Form (Project)

Student: Filip Wieslaw Bartoszewski

Project Title: Using cellular automata for non-binary classification

Supervisor: Dr. Michael A. Lones

<table>
<thead>
<tr>
<th>Risk</th>
<th>Present (give details) (tick if present)</th>
<th>Control Measures and/or Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Office environment- includes purely software projects</td>
<td>X – the creation of the classification framework and all the experiments are purely a software project</td>
<td>Nothing</td>
</tr>
<tr>
<td>Unusual peripherals e.g. Robot, VR helmet, haptic device, etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unusual Output e.g. Laser, loud noises, flashing lights etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other risks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix 2 – Sample results files (created by copying the ECJ console output)

**basicFoldProblem1**

FOLD1

Cam006-L.CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -2.1741573033707864 Non-cast: 2.601123595505618 Correct?: false Newcast: 3 Correct?: false

Cam018-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -2.056179775280899 Non-cast: 2.758426966292135 Correct?: false Newcast: 3 Correct?: false

Cam020-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.8258426966292136 Non-cast: 2.601123595505618 Correct?: true Newcast: 3 Correct?: false

Cam035-L.CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -0.606741573033708 Non-cast: 4.691011235955056 Correct?: true Newcast: 5 Correct?: true

Cam035-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: 0.3258426966292136 Non-cast: 4.601123595505618 Correct?: true Newcast: 5 Correct?: false

Cam042-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.5505617977528088 Non-cast: 3.567415730337079 Correct?: true Newcast: 4 Correct?: false

Cam045-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.80988764044944 Non-cast: 2.5786516853932584 Correct?: true Newcast: 3 Correct?: false

Cam052-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.5393258426966292 Non-cast: 2.2191011235955056 Correct?: true Newcast: 2 Correct?: true

Cam053-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -2.2078651685393256 Non-cast: 2.556179775280899 Correct?: false Newcast: 3 Correct?: true

Cam072-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 4 devFitness: 2.106741573033708 Non-cast: 4.308988764044944 Correct?: false Newcast: 4 Correct?: false

Cam079-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 1.1573033707865168 Non-cast: 4.376404494382022 Correct?: false Newcast: 4 Correct?: false

Cam079-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -0.5898876404494384 Non-cast: 4.713483146067416 Correct?: true Newcast: 5 Correct?: true

Cam079-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 5 devFitness: 0.9157303370786511 Non-cast: 5.387640449438202 Correct?: false Newcast: 5 Correct?: false

Cam073-L-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.601123595505618 Non-cast: 3.3651685393258424 Correct?: false Newcast: 3 Correct?: false

Cam073-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.196629213483146 Non-cast: 3.095505617977528 Correct?: true Newcast: 3 Correct?: true

Cam074-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 2 devFitness: -1.904494382022472 Non-cast: 2.960674157303371 Correct?: false Newcast: 3 Correct?: false

Cam074-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 2 devFitness: -0.022471910112359605 Non-cast: 2.803370786516854 Correct?: false Newcast: 3 Correct?: true

Cam074-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 5 devFitness: 1.6797752808988768 Non-cast: 5.073033707865168 Correct?: false Newcast: 5 Correct?: false

Cam075-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 5 devFitness: 0.6797752808988768 Non-cast: 5.073033707865168 Correct?: false Newcast: 5 Correct?: false

Nor341553-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.3314606741573032 Non-cast: 3.2752808988764044 Correct?: true Newcast: 3 Correct?: true
FOLD2

Cam012-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.4325842696629216 Non-cast: 3.410112359550561 Correct?: true Newcast: 3 Correct?: true

Cam012-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.2640449438202248 Non-cast: 3.1853932584269664 Correct?: true Newcast: 3 Correct?: true

Cam013-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -1.904494382022472 Non-cast: 2.960674157303371 Correct?: false Newcast: 3 Correct?: true

Cam013-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -0.7696629213483144 Non-cast: 3.140449438202247 Correct?: false Newcast: 3 Correct?: false

Cam016-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: -0.8202247191011236 Non-cast: 3.073033707865169 Correct?: false Newcast: 3 Correct?: false

Cam016-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: -1.2303370786516856 Non-cast: 3.8595505617977537 Correct?: false Newcast: 4 Correct?: false

Cam020-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.7415730337078652 Non-cast: 2.48876404494382 Correct?: true Newcast: 2 Correct?: true

Cam020-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: -1.803370786516854 Non-cast: 3.0955056179775758 Correct?: false Newcast: 3 Correct?: false

Cam036-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.2471910112359552 Non-cast: 3.837078651685393 Correct?: false Newcast: 4 Correct?: true

Cam045-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.6573033707865168 Non-cast: 2.3764044943820224 Correct?: true Newcast: 2 Correct?: true

Cam052-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.7191011235955056 Non-cast: 3.2078651685393256 Correct?: false Newcast: 3 Correct?: false
Cam053-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.6516853932584272 Non-cast: 3.297752808988764 Correct?: false Newcast: 3 Correct?: false

Cam725-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -0.9269662921348321 Non-cast: 4.26404493820225 Correct?: false Newcast: 4 Correct?: false

Cam727-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -1.0786516853932584 Non-cast: 4.061797752808989 Correct?: false Newcast: 4 Correct?: true

Cam731-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 3.18539325842664 Non-cast: 3.747191011235955 Correct?: false Newcast: 3 Correct?: true

Cam738-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.264044938202248 Non-cast: 3.297752808988764 Correct?: false Newcast: 3 Correct?: false

Cam739-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 3.1179775280898876 Non-cast: 3.8146067415730336 Correct?: false Newcast: 3 Correct?: true

Cam746-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 2.691011235955056 Correct?: true Newcast: 3 Correct?: false

Cam747-L-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 2.533707865168539 Correct?: true Newcast: 3 Correct?: true

Cast accuracy: 9/28 0.32142857142857145

Non-Cast accuracy: 14/28 0.5

Binary accuracy: 20/28 0.7142857142857143

Standard deviation: 0.7389646869983949
Cam035-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: 0.25842696629213435 Non-cast: 4.511235955056179 Correct?: true Newcast: 5 Correct?: false

Cam035-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 1.0056179775280896 Non-cast: 4.174157303370786 Correct?: false Newcast: 4 Correct?: false

Cam042-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.7359550561797752 Non-cast: 3.8146067415730336 Correct?: true Newcast: 4 Correct?: false

Cam042-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.786516853932584 Non-cast: 3.8820224719101124 Correct?: true Newcast: 4 Correct?: true

Cam042-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: -0.09550561797752799 Non-cast: 4.039325842696629 Correct?: true Newcast: 4 Correct?: true

Cam045-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.786516853932584 Non-cast: 3.342696629213483 Correct?: true Newcast: 4 Correct?: true

Cam045-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: 1.5 Non-cast: 1.601123595505618 Correct?: false Newcast: 4 Correct?: false

Cam045-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.3483146067415728 Non-cast: 3.29752808988764 Correct?: true Newcast: 3 Correct?: false

Cam045-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: 1.0056179775280896 Non-cast: 4.174157303370786 Correct?: false Newcast: 4 Correct?: false

Cam045-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 1.2247191011235952 Non-cast: 4.466292134831461 Correct?: true Newcast: 4 Correct?: true

Cam047-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.8786404494382024 Non-cast: 4.01685393258427 Correct?: true Newcast: 4 Correct?: true

Cam047-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.8786404494382024 Non-cast: 4.01685393258427 Correct?: true Newcast: 4 Correct?: true

Cam047-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: -0.11235955056179758 Non-cast: 4.01685393258427 Correct?: true Newcast: 4 Correct?: true


Cam047-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 1.0056179775280896 Non-cast: 4.174157303370786 Correct?: true Newcast: 4 Correct?: false

Cam072-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 4 devFitness: 1.9887640449438202 Non-cast: 4.151685393258427 Correct?: false Newcast: 4 Correct?: false

Cam072-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: -0.9101123595505616 Non-cast: 4.286516853932584 Correct?: false Newcast: 4 Correct?: false

Cam072-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 5 devFitness: -1.6179775280898876 Non-cast: 3.342696629213483 Correct?: false Newcast: 3 Correct?: false

Cam072-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 4 devFitness: -1.6179775280898876 Non-cast: 3.342696629213483 Correct?: false Newcast: 3 Correct?: false

Cam076-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 1.3483146067415728 Non-cast: 3.29752808988764 Correct?: true Newcast: 3 Correct?: true

Cam076-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: -1.7359550561797752 Non-cast: 3.1853932584269664 Correct?: false Newcast: 3 Correct?: false

Cam079-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: 0.1741573033707864 Non-cast: 4.398876404494382 Correct?: true Newcast: 4 Correct?: true

Cam079-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: 0.1741573033707864 Non-cast: 4.398876404494382 Correct?: true Newcast: 4 Correct?: true


Cam079-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: 0.1741573033707864 Non-cast: 4.398876404494382 Correct?: true Newcast: 4 Correct?: true

Cam079-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: 0.1741573033707864 Non-cast: 4.398876404494382 Correct?: true Newcast: 4 Correct?: true
Nor500128-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: 0.3764044943820224 Non-cast: 4.668539325842697 Correct?: true Newcast: 5 Correct?: false

Nor500129-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.4157303370786516 Non-cast: 3.3876404494382024 Correct?: false Newcast: 3 Correct?: false

Nor780016-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.303370786516854 Non-cast: 1.904494382022472 Correct?: false Newcast: 2 Correct?: true

Cast accuracy: 11/29 0.3793103448275862
Non-Cast accuracy: 6/29 0.20689655172413793
Binary accuracy: 17/29 0.5862068965517241
Standard deviation: 0.9432390546299885

FOLD4

Cam006-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.7359550561797752 Non-cast: 3.1853932584269664 Correct?: false Newcast: 3 Correct?: false

Cam013-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.6853932584269664 Non-cast: 3.252808988764045 Correct?: false Newcast: 3 Correct?: false

Cam018-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.7528089887640448 Non-cast: 3.162921348314607 Correct?: false Newcast: 3 Correct?: false

Cam020-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.6348314606741576 Non-cast: 3.1179775280898876 Correct?: false Newcast: 3 Correct?: false

Cam020-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 1.0786516853932584 Non-cast: 2.938202247191011 Correct?: true Newcast: 3 Correct?: false

Cam036-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: -0.04494382022471921 Non-cast: 4.106741573033708 Correct?: true Newcast: 4 Correct?: true

Cam042-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.584269662921348 Correct?: true Newcast: 3 Correct?: false

Cam042-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.196629213483146 Non-cast: 3.904494382022472 Correct?: false Newcast: 4 Correct?: true

Cam042-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.5 Non-cast: 3.5 Correct?: false Newcast: 4 Correct?: true

Cam045-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.6179775280898876 Non-cast: 3.342696629213483 Correct?: false Newcast: 3 Correct?: false

Cam052-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.8539325842696632 Non-cast: 3.0280898876404496 Correct?: false Newcast: 3 Correct?: false

Cam072-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -0.7921348314606735 Non-cast: 4.443820224719101 Correct?: false Newcast: 4 Correct?: false

Cam079-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.6348314606741576 Non-cast: 3.679775280898764 Correct?: true Newcast: 4 Correct?: false
Cam731-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 1.1404494382022472 Non-cast: 4.353932584269662 Correct?: false Newcast: 4 Correct?: false

Cam739-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.4325842696629216 Non-cast: 3.4101123595505616 Correct?: true Newcast: 3 Correct?: true

Cam746-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 1.044938202247192 Non-cast: 2.89325842696292 Correct?: true Newcast: 2 Correct?: false

Cam747-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: 0.8876404494382024 Non-cast: 4.01685393258427 Correct?: false Newcast: 4 Correct?: false

Cam747-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.4325842696629216 Non-cast: 3.4101123595505616 Correct?: false Newcast: 3 Correct?: false

Cam750-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: 0.12359550561797761 Non-cast: 4.331460674157303 Correct?: false Newcast: 4 Correct?: true

Nor341553-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.702247191011236 Non-cast: 3.769662921348315 Correct?: true Newcast: 4 Correct?: true

Nor500103-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 1.0786516853932584 Non-cast: 2.938202247191011 Correct?: true Newcast: 3 Correct?: false

Cast accuracy: 10/28 0.35714285714285715
Non-Cast accuracy: 8/28 0.2857142857142857
Binary accuracy: 14/28 0.5
Standard deviation: 0.9197431781701446

FOLDS

Cam012-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.4662921348314608 Non-cast: 3.455056179775281 Correct?: true Newcast: 3 Correct?: true

Cam016-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.6685393258426968 Non-cast: 3.2752808988764044 Correct?: false Newcast: 3 Correct?: false

Cam018-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.7191011235955056 Non-cast: 3.2078651685393256 Correct?: false Newcast: 3 Correct?: false

Cam042-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.5168539325842696 Non-cast: 3.5224719101123596 Correct?: true Newcast: 4 Correct?: false

Cam045-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.8370786516853932 Non-cast: 3.050561797752809 Correct?: false Newcast: 3 Correct?: false

Cam052-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.8426966292134832 Non-cast: 2.6235955056179776 Correct?: true Newcast: 3 Correct?: false
Cast accuracy: 12/29 0.41379310344827586
Non-Cast accuracy: 14/29 0.4827586206896552
Binary accuracy: 18/29 0.6206896551724138
Standard deviation: 0.8179000387446725

FOLD6
Cam006-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -1.9550561797752808 Non-cast: 2.893258426966292 Correct?: false Newcast: 3 Correct?: false
Cam012-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.24719101123595522 Non-cast: 3.162921348314607 Correct?: true Newcast: 3 Correct?: true
Cam016-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.3146067415730336 Non-cast: 3.7471910112359555 Correct?: false Newcast: 4 Correct?: false
Cam020-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 1.01123595505618 Non-cast: 2.848314607415728 Correct?: true Newcast: 3 Correct?: false
Cam020-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.8370786516853932 Non-cast: 3.050561797752809 Correct?: false Newcast: 3 Correct?: false
Cam042-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.4662921348314608 Non-cast: 3.455056179775281 Correct?: true Newcast: 3 Correct?: true
Cam042-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.702247191011236 Non-cast: 3.230337086516856 Correct?: false Newcast: 3 Correct?: false
Cam045-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -1.9382022471910112 Non-cast: 2.9157303370786516 Correct?: false Newcast: 3 Correct?: false
Cam045-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.6404494382022472 Non-cast: 2.353932584269663 Correct?: true Newcast: 3 Correct?: true
Cam053-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -2.039325842696629 Non-cast: 2.780898864044944 Correct?: false Newcast: 3 Correct?: false
Cam725-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -0.9438202247191008 Non-cast: 4.241573033707866 Correct?: false Newcast: 4 Correct?: false
Cam725-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.297752808988764 Non-cast: 3.769662921348315 Correct?: false Newcast: 4 Correct?: false
Cam727-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.4494382022471912 Non-cast: 3.567415730337079 Correct?: false Newcast: 4 Correct?: false
Cam729-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.398876404494382 Non-cast: 3.363168539258424 Correct?: true Newcast: 3 Correct?: true
Cam738-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.21348314606741603 Non-cast: 3.1179775280898876 Correct?: true Newcast: 3 Correct?: true
Cam739-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.21348314606741603 Non-cast: 3.1179775280898876 Correct?: true Newcast: 3 Correct?: true
Cam739-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.365168539258424 Non-cast: 3.3202247191011236 Correct?: false Newcast: 3 Correct?: false
Cam740-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -2.106741573033708 Non-cast: 2.69101235955056 Correct?: false Newcast: 3 Correct?: false
Cam750-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.1797752808988764 Non-cast: 3.9269662921348316 Correct?: false Newcast: 4 Correct?: true
Cam755-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.5 Non-cast: 3.5 Correct?: false Newcast: 4 Correct?: true

Cam757-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: 0.039325842696628754 Non-cast: 4.219101123595506 Correct?: true Newcast: 4 Correct?: true

Nor190001-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.2808988764044944 Non-cast: 3.792134831460674 Correct?: false Newcast: 4 Correct?: true

Nor341553-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.5505617977528088 Non-cast: 3.567415730337079 Correct?: true Newcast: 4 Correct?: false

Nor500039-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.4382022471910112 Non-cast: 2.0842696629213484 Correct?: true Newcast: 2 Correct?: true

Nor500103-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.1853932584269664 Non-cast: 1.7471910112359555 Correct?: false Newcast: 2 Correct?: true

Nor500103-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.3483146067415728 Non-cast: 3.702247191011236 Correct?: false Newcast: 4 Correct?: true

Nor780007-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.5561797752808988 Non-cast: 2.241573033707865 Correct?: true Newcast: 2 Correct?: true

Nor780016-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.8707865168539328 Non-cast: 3.0056179775280896 Correct?: false Newcast: 3 Correct?: false

Nor780026-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.2865168539325844 Non-cast: 1.8820224719101124 Correct?: false Newcast: 2 Correct?: true

Cast accuracy: 11/29 0.3793103448275862
Non-Cast accuracy: 16/29 0.5517241379310345
Binary accuracy: 21/29 0.7241379310344828
Standard deviation: 0.8060829135993801
regressionCAFold1

FOLD1

Cam006-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.5 Non-cast: 3.5 Correct?: false Newcast: 4 Correct?: false

Cam018-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: -1.5 Non-cast: 3.5 Correct?: false Newcast: 3 Correct?: false

Cam020-L-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.4831460674157304 Non-cast: 3.4775280898876404 Correct?: false Newcast: 3 Correct?: false

Cam035-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.5 Non-cast: 3.5 Correct?: false Newcast: 4 Correct?: false

Cam035-L-ML0.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.5168539325842696 Non-cast: 3.4775280898876404 Correct?: false Newcast: 3 Correct?: false

Cam035-R-ML0.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.5168539325842696 Non-cast: 3.5224719101123596 Correct?: true Newcast: 4 Correct?: false

Cam042-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.5 Non-cast: 3.5 Correct?: false Newcast: 4 Correct?: true
Nor500128-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.5 Non-cast: 3.5 Correct?: false New-cast: 4 Correct?: true
Nor500129-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.4831460674157304 Non-cast: 3.4775280898876404 Correct?: false New-cast: 3 Correct?: false
Nor500129-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: -0.5168539325842696 Non-cast: 3.4775280898876404 Correct?: false New-cast: 3 Correct?: false
Nor500176-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.5674157303370784 Non-cast: 3.5898876404494384 Correct?: false New-cast: 4 Correct?: false
Nor500176-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.5168539325842696 Non-cast: 3.4775280898876404 Correct?: false New-cast: 3 Correct?: false
Nor780016-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.5168539325842696 Non-cast: 3.4775280898876404 Correct?: false New-cast: 3 Correct?: false
Nor780026-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.4831460674157304 Non-cast: 3.4775280898876404 Correct?: false New-cast: 3 Correct?: false

Cast accuracy: 6/29 0.20689655172413793
Non-Cast accuracy: 7/29 0.2413793103448276
Standard deviation: 1.0201472297559084
Binary classification hits: 15 0.5172413793103449

FOLD2
Cam012-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam012-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam013-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam013-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam016-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam016-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam020-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: true
Cam020-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam036-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam045-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: true
Cam052-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam053-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Cam725-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam727-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam731-L-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam731-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam738-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam739-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam746-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Cam747-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam747-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Cam750-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Nor190001-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Nor341553-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Nor500103-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Nor500145-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Nor780007-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Nor780007-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cast accuracy: 0/28 0.0
Non-Cast accuracy: 6/28 0.21428571428571427
Standard deviation: 1.5714285714285714
Binary classification hits: 13 0.4642857142857143

FOLD3
Cam018-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam035-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam035-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam035-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam042-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam042-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam042-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam045-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam045-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam052-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam725-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam727-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam727-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true

Cam739-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true

Cam740-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam746-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam747-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam755-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam757-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Cam757-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Nor500039-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false

Nor500039-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true

Nor500103-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true

Nor500103-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true

Nor500128-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Nor500129-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: true
Nor500176-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: false
Nor780016-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false New-cast: 2 Correct?: true

Cast accuracy: 0/29 0.0
Non-Cast accuracy: 8/29 0.27586206896551724
Standard deviation: 1.5172413793103448
Binary classification hits: 15 0.5172413793103449

FOLD4
Cam006-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -1.1123595505617976 Non-cast: 4.01685393258427 Correct?: false New-cast: 4 Correct?: false
Cam013-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: -0.9550561797752799 Non-cast: 4.039325842696629 Correct?: true New-cast: 4 Correct?: true
Cam018-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -1.0786516853932584 Non-cast: 4.061797752808989 Correct?: false New-cast: 4 Correct?: false
Cam036-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.196629213483146 Non-cast: 3.904494382022472 Correct?: false New-cast: 4 Correct?: true
Cam042-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 0.9719101123595504 Non-cast: 4.129213483146067 Correct?: false New-cast: 4 Correct?: false
Cam042-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: -0.1123955056179758 Non-cast: 4.01685393258427 Correct?: true New-cast: 4 Correct?: true
Cam042-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: -0.0786516853932584 Non-cast: 4.061797752808989 Correct?: true New-cast: 4 Correct?: true
Cam045-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -1.0786516853932584 Non-cast: 4.061797752808989 Correct?: false New-cast: 4 Correct?: true
Cam052-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -1.0617977528089888 Non-cast: 4.084269662921349 Correct?: false New-cast: 4 Correct?: false
Cam053-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 4 devFitness: -1.0786516853932584 Non-cast: 4.061797752808989 Correct?: false New-cast: 4 Correct?: false
Cam727-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.196629213483146 Non-cast: 3.904494382022472 Correct?: false New-cast: 4 Correct?: false
Cam729-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 0.9550561797752808 Non-cast: 4.106741573033708 Correct?: false New-cast: 4 Correct?: false
Cam731-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.7528089887640448 Non-cast: 3.83708651685393 Correct?: true New-cast: 4 Correct?: false
Cam739-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 0.9382022471910112 Non-cast: 4.084269662921349 Correct?: false Newcast: 4 Correct?: false
Cam746-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 4 devFitness: 1.9213483146067416 Non-cast: 4.061797752808989 Correct?: 4 Correct?: false
Cam747-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.7359550561797752 Non-cast: 3.8146067415730336 Correct?: true Newcast: 4 Correct?: false
Cam747-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 4 devFitness: -0.21348314606741603 Non-cast: 3.8820224719101124 Correct?: true Newcast: 4 Correct?: false
Cam750-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.2640449438202248 Non-cast: 3.8146067415730336 Correct?: false Newcast: 4 Correct?: true
Cam757-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.6516853932584272 Non-cast: 3.702247191011236 Correct?: true Newcast: 4 Correct?: false
Nor341553-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.786516853932584 Non-cast: 3.8820224719101124 Correct?: false Newcast: 4 Correct?: false
Nor500103-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.8370786516853932 Non-cast: 3.949438202247191 Correct?: false Newcast: 4 Correct?: false
Nor500145-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.2640449438202248 Non-cast: 3.8146067415730336 Correct?: true Newcast: 4 Correct?: true
Nor780016-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.7191011235955056 Non-cast: 3.792134831460674 Correct?: false Newcast: 4 Correct?: false
Nor780016-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.2640449438202248 Non-cast: 3.8146067415730336 Correct?: true Newcast: 4 Correct?: true
Nor780026-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 4 devFitness: 1.904494382022472 Non-cast: 4.039325842696629 Correct?: false Newcast: 4 Correct?: false

Cast accuracy: 7/28 0.25
Non-Cast accuracy: 7/28 0.25
Standard deviation: 0.9915730337078646
Binary classification hits: 14 0.5

FOLDS
Cam012-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam016-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam018-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam042-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam045-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam052-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Cam052-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam053-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam725-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam727-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam729-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam729-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam731-L-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam738-L-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam739-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam739-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Cam740-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam740-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam740-R-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam747-L-CC.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam750-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 1 devFitness: -3.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam755-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cam757-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 1 devFitness: -2.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Nor341553-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Nor500039-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Nor500103-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Nor500129-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Nor500176-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: true
Nor780007-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 1 devFitness: -1.0 Non-cast: 1.5 Correct?: false Newcast: 2 Correct?: false
Cast accuracy: 0/29 0.0
Non-Cast accuracy: 7/29 0.2413793103448276
Standard deviation: 1.5862068965517242
Binary classification hits: 18  0.6206896551724138

FOLD6
Cam006-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.7696629213483144 Non-cast: 3.140449438202247 Correct?: false Newcast: 3 Correct?: false
Cam012-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 0 devFitness: 0.4494382022471912 Non-cast: 3.4325842696629216 Correct?: true Newcast: 3 Correct?: true
Cam016-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.146067415730368 Non-cast: 3.9719101123595504 Correct?: true Newcast: 4 Correct?: true
Cam020-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.1797752808988764 Non-cast: 3.073033707865169 Correct?: false Newcast: 3 Correct?: false
Cam020-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -1.146067415730368 Non-cast: 3.9719101123595504 Correct?: true Newcast: 4 Correct?: true
Cam045-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: 1.0449438202247192 Non-cast: 2.893258426966292 Correct?: true Newcast: 3 Correct?: false
Cam053-R-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -2.00561797752809 Non-cast: 2.828516853932584 Correct?: false Newcast: 3 Correct?: false
Cam725-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.9101123595505616 Non-cast: 4.286516853932584 Correct?: true Newcast: 4 Correct?: false
Cam725-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.2640449438202248 Non-cast: 3.8146067415730336 Correct?: true Newcast: 4 Correct?: true
Cam727-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 3 devFitness: -1.4494382022471912 Non-cast: 3.5674157303370797 Correct?: false Newcast: 4 Correct?: false
Cam729-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.415730337086516 Non-cast: 3.3876404494382024 Correct?: true Newcast: 3 Correct?: true
Cam729-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 4 devFitness: 0.8876404494382024 Non-cast: 4.016853932584274 Correct?: false Newcast: 4 Correct?: false
Cam739-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.297752808988764 Non-cast: 3.2303370786516856 Correct?: true Newcast: 3 Correct?: true
Cam739-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 3 devFitness: 1.6348314606741576 Non-cast: 3.6797752808988764 Correct?: false Newcast: 4 Correct?: false
Cam740-L-CC.bmp_FCanavan_Output.txt True risk: 5 Prediction: 2 devFitness: -2.1235955056179776 Non-cast: 2.668539325842697 Correct?: false Newcast: 3 Correct?: false
Cam750-R-CC.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.1629213483146068 Non-cast: 3.949438202247191 Correct?: false Newcast: 4 Correct?: true
Cam755-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.3314606741573032 Non-cast: 3.7247191011235956 Correct?: false Newcast: 4 Correct?: true
Cam757-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 4 devFitness: 0.07303370786516794 Non-cast: 4.26404943820225 Correct?: true Newcast: 4 Correct?: true

Nor190001-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.264049438202248 Non-cast: 3.8146067415730336 Correct?: false Newcast: 4 Correct?: true

Nor341553-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Prediction: 3 devFitness: 0.5674157303370784 Non-cast: 3.5898876404494384 Correct?: true Newcast: 4 Correct?: false

Nor500039-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.5730337078651684 Non-cast: 2.264044943820225 Correct?: true Newcast: 2 Correct?: true

Nor500103-R-CC.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.2528089887640448 Non-cast: 1.8370786516853932 Correct?: true Newcast: 2 Correct?: true

Nor500103-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.23033707865168562 Non-cast: 3.859550561797753 Correct?: false Newcast: 4 Correct?: true

Nor780007-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 2 devFitness: 0.691011235955056 Non-cast: 2.4213483146067416 Correct?: true Newcast: 2 Correct?: true

Nor780016-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Prediction: 3 devFitness: -0.702247191011236 Non-cast: 3.2303370786516856 Correct?: false Newcast: 3 Correct?: false

Nor780026-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Prediction: 1 devFitness: 0.2696629213483144 Non-cast: 1.8595505617977528 Correct?: true Newcast: 2 Correct?: true

Cast accuracy: 9/29 0.3103448275862069
Non-Cast accuracy: 13/29 0.4482758620689655
Standard deviation: 0.845021309569934
Binary classification hits: 18 0.6206896551724138

classCAROC

FOLD1


Cam018-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.398876404494382 Difference: -0.8258426966292136 Verdict: TP Marker: Mass

Cam020-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.7247191011235956 Difference: 1.6685393258426968 Verdict: FP Marker: MicroCalc

Cam035-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.915730337078651 Difference: -0.438202471910112 Verdict: TP Marker: Mass

Cam035-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.870786516853933 Difference: 0.5280898876404496 Verdict: TP Marker: Mass


Cam042-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.758426966292134 Difference: 0.44382022471910076 Verdict: TP Marker: Mass


Cam052-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.455056179775281 Difference: 1.4662921348314608 Verdict: TN Marker: Mass

Cam053-R-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.1966292134831455 Difference: -0.97752808988764 Verdict: TP Marker: Mass


Cam729-R-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.5898876404494384 Difference: -1.4325842696629216 Verdict: TP Marker: MicroCalc

Cam729-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.8033707865168545 Difference: 0.477528089887639595 Verdict: TP Marker: Mass

Cam738-L-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: false Prediction: 4 Non-cast: 4.331460674157303 Difference: 0.12359550561797761 Verdict: TP Marker: Mass

Cam739-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.4213483146067416 Difference: -0.308988764044944 Verdict: TN Marker: MicroCalc

Cam740-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.376404494382022 Difference: -0.8426966292134832 Verdict: TP Marker: Mass

Cam740-R-CC.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 4 Non-cast: 4.151685393258427 Difference: 0.98876404494382 Verdict: FP Marker: Mass

Cam746-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.039325842696629 Difference: 0.404493820224727 Verdict: TN Marker: MicroCalc

Cam750-R-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.915730337078651 Difference: 0.5617977528089888 Verdict: TP Marker: Mass

Nor341553-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 4 Non-cast: 4.578651685393258 Difference: 1.3089887640449444 Verdict: FP Marker: Mass

Nor500039-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.646067415730338 Difference: 0.3595505617977528 Verdict: TP Marker: Mass

Nor500038-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 1.6573033707865168 Difference: 0.1179775280898876 Verdict: TN Marker: MicroCalc

Nor780026-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 1 Non-cast: 1.6573033707865168 Difference: 0.1179775280898876 Verdict: TN Marker: MicroCalc

True positives: 15
True negatives: 8
False positives: 6
False negatives: 0
Mass errors: 5
Calcification errors: 1
Precision: 0.7142857143
Recall/Sensitivity: 1.0
Specificity: 0.5714285714
Area under curve: 0.9023809524
Optimal threshold: 3.9044943820
Tree size: 16

FOLD2
Cam012-R.CC.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 4 Non-cast: 4.2415730337 Difference: 0.5617977528 Verdict: FP Marker: Mass
Cam012-R.MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.7022471910 Difference: 0.6516853935 Verdict: FP Marker: Mass
Cam020-R.MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.9943820224 Difference: -1.1292134831 Verdict: TP Marker: MicroCalc
Cam036-R.CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.5561797528 Difference: -0.2921348314 Verdict: TP Marker: Mass
Cam052-R.MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.3998764044 Difference: -0.8258426962 Verdict: TP Marker: Mass
Cam725-R.CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.1179775280 Difference: -0.2865168539 Verdict: TP Marker: Mass
Cam727-L.MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.2303370786 Difference: -0.2022471910 Verdict: TP Marker: MicroCalc
Cam731-L.CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.9831460674 Difference: 0.6123595506 Verdict: TP Marker: Mass
Cam731-L.MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 5 Non-cast: 5.1629213483 Difference: 1.7471910112 Verdict: FP Marker: Mass
Cam739-L-CC.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 4 Non-cast: 4.01685393258427 Difference: 0.8876404494382024 Verdict: FP Marker: MicroCalc

Cam746-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.544943820224719 Difference: 1.5337078651685392 Verdict: FP Marker: Mass

Cam747-L-CC.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.8258426966292136 Difference: -0.00561797728090012 Verdict: TN Marker: Mass


Cam750-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.870786516853933 Difference: -0.4719101123595504 Verdict: TP Marker: MicroCalc

Nor190001-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.511235955056179 Difference: 0.25842696629213435 Verdict: TP Marker: Mass

Nor341553-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 1 Non-cast: 1.8595505617977528 Difference: 0.2696629213483144 Verdict: TN Marker: MicroCalc

Nor500103-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.646067415730338 Difference: 0.3595505617977528 Verdict: TP Marker: Mass

Nor500145-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.758426966292134 Difference: 0.44382022471910076 Verdict: TP Marker: Mass

Nor780007-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.792134831460674 Difference: 1.7191011235955056 Verdict: FP Marker: MicroCalc

Nor780007-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.90449382022472 Difference: 0.803370786516854 Verdict: FP Marker: Mass

19/28
True positives: 15
True negatives: 4
False positives: 9
False negatives: 0
Mass errors: 6
Calcification errors: 3

Precision: 0.625
Recall/Sensitivity: 1.0
Specificity: 0.3076923076923077
Area under curve: 0.8871794871794871
Optimal threshold: 4.117977528089888
Tree size: 33

FOLD3
Cam018-R-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.3876404494382024 Difference: -1.5842696629213484 Verdict: FN Marker: Mass
Filip Wieslaw Bartoszewski

Research Report

Cam035-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.2078651685393265 Difference: -0.2191011235955056 Verdict: TP Marker: Mass

Cam035-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.8033707865168545 Difference: 0.4775280898876395 Verdict: TP Marker: Mass


Cam042-L-CC.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.8146067415730336 Difference: 0.7359550561797752 Verdict: FP Marker: Mass

Cam042-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 4 Non-cast: 4.556179775280899 Difference: 1.2921348314606735 Verdict: FP Marker: Mass

Cam042-R-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.960674157303371 Difference: 0.595505617977528 Verdict: TP Marker: Mass


Cam045-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.904494382022472 Difference: -1.196629213483146 Verdict: TP Marker: MicroCalc


Cam052-R-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.949438202247191 Difference: -1.1629213483146068 Verdict: TP Marker: Mass

Cam725-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.443820224719101 Difference: 0.20786516853932646 Verdict: TP Marker: Mass

Cam727-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.376404494382022 Difference: -0.842696292134832 Verdict: TP Marker: Mass


Cam739-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.567415730337079 Difference: 1.5505617977528088 Verdict: FN Marker: MicroCalc

Cam740-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.1853932584269664 Difference: -1.7359550561797752 Verdict: FN Marker: Mass

Cam746-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.544943820224719 Difference: 0.5337078651685392 Verdict: FP Marker: Mass


Cam755-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.174157303370786 Difference: 0.005617977528089568 Verdict: TP Marker: Mass


Cam757-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.6011235955056168 Difference: 0.3258426966292136 Verdict: TP Marker: MicroCalc

Nor500039-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.511235955056179 Difference: 0.25842696629213435 Verdict: TP Marker: Mass

Nor500039-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.331460674157303 Difference: 0.6235955056179776 Verdict: TN Marker: MicroCalc

Nor500103-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.668539325842697 Difference: 0.8764044943820224 Verdict: TN Marker: MicroCalc
Nor500103-R-CC.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 1 Non-cast: 1.6573033707865168 Difference: 0.1179775280898876 Verdict: TN Marker: MicroCalc

Nor500128-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.98314606741573 Difference: 0.6123595505617976 Verdict: TP Marker: MicroCalc

Nor500129-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.7808988640044944 Difference: 0.9606741573033708 Verdict: TN Marker: MicroCalc

Nor500176-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.713483146067416 Difference: -0.58988644494384 Verdict: TP Marker: Mass

Nor780016-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 1 Non-cast: 1.702247191011236 Difference: 0.1516853925842678 Verdict: TN Marker: MicroCalc

18/29
True positives: 12
True negatives: 6
False positives: 8
False negatives: 3
Mass errors: 8
Calcification errors: 3
Precision: 0.6
Recall/Sensitivity: 0.8
Specificity: 0.42857142857142855
Area under curve: 0.7571428571428571
Optimal treshold: 4.174157303370786
Tree size: 26

FOLD4

Cam013-R-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.668539325842697 Difference: -0.6235955056179776 Verdict: TP Marker: Mass

Cam013-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.89325842696292 Difference: 0.5449438202247192 Verdict: TP Marker: Mass

Cam018-R-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.6797752808988764 Difference: -1.365168539258424 Verdict: TP Marker: Mass


Cam020-R-CC.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.904494382022472 Difference: 1.803370786516854 Verdict: FP Marker: MicroCalc

Cam036-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.421348314606742 Difference: 0.19101123595505598 Verdict: TP Marker: Mass
Cam042-L-CC.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.837085361685393 Difference: 0.752808987640448 Verdict: FP Marker: Mass

Cam042-R-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.00561797752809 Differe: 0.62913431460672 Verdict: TP Marker: Mass

Cam042-R-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 4.938202247191011 Difference: 0.5786516853932584 Verdict: TP Marker: Mass


Cam052-R-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.01685393258427 Difference: -0.6123595505617976 Verdict: TP Marker: Mass

Cam053-R-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.443820224719101 Difference: -0.792134314606735 Verdict: TP Marker: Mass

Cam072-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: false Prediction: 2 Non-cast: 2.39886404449382 Difference: -0.3258426966292136 Verdict: TN Marker: MicroCalc

Cam074-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.4775280898876404 Difference: 1.4831460674157304 Verdict: TN Marker: Mass

Cam074-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 4 Non-cast: 5.320224719011123 Difference: -0.8651685393258433 Verdict: FN Marker: Mass

Cam074-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 4 Non-cast: 5.073033707865168 Difference: 0.6797752809888777 Verdict: TP Marker: MicroCalc

Cam075-R-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 1 Non-cast: 1.7247191011235956 Difference: 0.1685393258426968 Verdict: TN Marker: MicroCalc


Nor500103-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.803370786516854 Difference: 0.9775280898876404 Verdict: TN Marker: MicroCalc

Nor500145-L-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 4.758426966292134 Difference: 0.4438202247191008 Verdict: TP Marker: Mass

Nor780016-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 1 Non-cast: 1.7247191011235956 Difference: 0.16853932584269682 Verdict: TN Marker: MicroCalc

Nor780026-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 1 Non-cast: 1.747191011235955 Difference: 0.1853932584269664 Verdict: TN Marker: MicroCalc

True positives: 13
True negatives: 6
False positives: 8  
False negatives: 1  
Mass errors: 5  
Calcification errors: 4  
Precision: 0.6190476190476191  
Recall/Sensitivity: 0.9285714285714286  
Specificity: 0.42857142857142855  
Area under curve: 0.6556122448979592  
Optimal threshold: 3.758426966292135  
Tree size: 15

FOLD5
Cam016-R-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.308988764044944 Difference: 0.10674157303370801 Verdict: TP Marker: Mass
Cam018-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.91573037078651 Difference: -0.4382022471910112 Verdict: TP Marker: Mass
Cam052-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: false Prediction: 3 Non-cast: 3.9269662921348316 Difference: 1.8202247191011236 Verdict: FP Marker: Mass
Cam052-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.780898876404494 Difference: -0.5393258426966288 Verdict: TP Marker: Mass
Cam053-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.758426966292134 Difference: 0.44382022471910076 Verdict: TP Marker: Mass
Cam075-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.117977528089888 Difference: 0.713483146067416 Verdict: TP Marker: Mass
Cam077-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.095505617977528 Difference: -0.3037078651685445 Verdict: TP Marker: Mass
Cam079-R-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.174157303370786 Difference: -0.9943820224719104 Verdict: TP Marker: MicroCalc
Cam079-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.410112359550562 Difference: 0.9325842696629216 Verdict: TP Marker: Mass
Cam071-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.455056179775281 Difference: 0.9662921348314608 Verdict: TP Marker: Mass
Cam078-L-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.780898876404494 Difference: 0.46067415730337125 Verdict: TP Marker: Mass

Cam740-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.792134831460674 Difference: -1.280898764044944 Verdict: TP Marker: Mass

Cam740-L-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.073033707865168 Difference: -0.32022471910112316 Verdict: TP Marker: Mass


Cam747-L-CC.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.1853932584269664 Difference: 0.260449438202248 Verdict: TN Marker: Mass

Cam750-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.41011235955056 Difference: -0.06741573033707837 Verdict: TP Marker: Mass

Cam755-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.6911235955056 Difference: 0.393258426966292 Verdict: TP Marker: Mass

Cam757-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.365168539325843 Difference: -0.8988764044943824 Verdict: TP Marker: MicroCalc

Nor341553-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.0842696629213484 Difference: 0.4382022471910112 Verdict: TN Marker: MicroCalc

Nor500039-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.6911235955056 Difference: 0.893258426966292 Verdict: TN Marker: MicroCalc

Nor500103-R-CC.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.2191011235955056 Difference: 0.5393258426966292 Verdict: TN Marker: MicroCalc

Nor500129-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.6460674157303368 Difference: 0.8595505617977528 Verdict: TN Marker: MicroCalc

Nor500176-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.6235955056179776 Difference: 0.8426966292134832 Verdict: TN Marker: MicroCalc

Nor780007-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 4 Non-cast: 4.466292134831461 Difference: 1.2247191011235952 Verdict: FP Marker: Mass

True positives: 16
True negatives: 6
False positives: 7
False negatives: 0
Mass errors: 5
Calcification errors: 2
Precision: 0.6956521739130435
Recall/Sensitivity: 1.0
Specificity: 0.46153846153846156
Area under curve: 0.846153846153846
Optimal threshold: 4.466292134831461
Tree size: 39
Cam006-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.005617977528099 Difference: -0.3707865168539328 Verdict: TP Marker: Mass

Cam012-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 4 Non-cast: 4.398876404494382 Difference: 1.1741573033707864 Verdict: FP Marker: Mass

Cam016-R-MLO.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.185393258426966 Difference: -0.2359550561797752 Verdict: TP Marker: Mass


Cam042-R-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 4 Non-cast: 4.93820224791011 Difference: 1.5786516853932584 Verdict: FP Marker: Mass

Cam053-R-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.556179775280889 Difference: -0.7078651685393265 Verdict: TP Marker: Mass

Cam725-R-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.387640449438202 Difference: -0.08426966292134885 Verdict: TP Marker: Mass

Cam725-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.050561797752809 Difference: 0.6629213483146064 Verdict: TP Marker: Mass

Cam727-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.848314606471573 Difference: -0.48876404494382086 Verdict: TP Marker: Mass


Cam738-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.073033707865168 Difference: 1.6797752808988768 Verdict: FP Marker: Mass

Cam739-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.140449438202247 Difference: 0.23033707865168562 Verdict: TN Marker: MicroCalc

Cam794/R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.99438202247191 Difference: 1.8707865168539328 Verdict: FP Marker: MicroCalc

Cam745-L-CC.bmp_FCanavan_Output.txt True risk: 5 Correct high/low prediction?: true Prediction: 3 Non-cast: 3.6123595505617976 Difference: -1.4157303370786516 Verdict: TP Marker: Mass

Cam850-R-CC.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.230337078651685 Difference: 0.797752808988764 Verdict: TP Marker: Mass

Cam855-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.623595505617978 Difference: 0.3426966292134832 Verdict: TP Marker: Mass

Cam737-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.2078651685393265 Difference: 0.780988764044944 Verdict: TP Marker: MicroCalc

Nor19001-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.848314606471573 Difference: 0.5112359550561791 Verdict: TP Marker: Mass
Nor341553-L-MLO.bmp_FCanavan_Output.txt True risk: 3 Correct high/low prediction?: false Prediction: 5 Non-cast: 5.275280898876405 Difference: 1.8314606741573032 Verdict: FP Marker: Mass

Nor500039-R-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.668539325842697 Difference: 0.8764044943820224 Verdict: TN Marker: MicroCalc

Nor500103-R-CC.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.1516853932584272 Difference: 0.48876404494382 Verdict: TN Marker: MicroCalc

Nor500103-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 5 Non-cast: 5.117977528089888 Difference: 0.713483146067416 Verdict: TP Marker: Mass


Nor780016-R-MLO.bmp_FCanavan_Output.txt True risk: 4 Correct high/low prediction?: true Prediction: 4 Non-cast: 4.421348314606742 Difference: 0.19101123595505598 Verdict: TP Marker: MicroCalc

Nor780026-L-MLO.bmp_FCanavan_Output.txt True risk: 2 Correct high/low prediction?: true Prediction: 2 Non-cast: 2.1516853932584272 Difference: 0.48876404494382 Verdict: TN Marker: MicroCalc

True positives: 16
True negatives: 5
False positives: 8
False negatives: 0
Mass errors: 5
Calcification errors: 3
Precision: 0.666666666666666
Recall/Sensitivity: 1.0
Specificity: 0.38461538461538464
Area under curve: 0.7355769230769231
Optimal threshold: 4.174157303370786
Tree size: 22