

# *Application of Neural Network in Diagnosis of Breast Cancer*

Subject module project in Computer Science

4th semester project

## **Group members:**

Asiya Mohamad Yusuf Muse - amym@ruc.dk - 73143,  
Amirreza Khoshbakht - amirreza@ruc.dk -76743,  
Mehmet Daskaya - daskaya@ruc.dk - 76742,  
Ömer Burak Dogan - obd@ruc.dk -76741,  
Sara Angelucci - angelucci@ruc.dk - 73156

**Supervisor:** Torben Braüner - torben@ruc.dk

May 23, 2023



# 1 Abstract

The study investigates the use of neural networks in breast cancer diagnosis. The model is used to analyze the dataset, aiming for an accurate classification of breast cancer cases as benign or malignant, based on features given in the dataset. Therefore, a brief introduction to Artificial neural networks, their structure and function are given, the choice of model and the programming language of choice. Afterwards, the planning process is presented, including the considerations taken regarding the program and its design. Methods of the approach used are set forth, such as the data structure, algorithm, and the utilization of a graphical user interface. The technical parts of the project are also displayed, such as procedure and prototype, along with a description of the program and an explanation of the different parts of the code. The results of the testing of the model are shown. According to the results observed, the model achieved high accuracy in predicting the target in both the training set and test set

# Contents

<b>1</b>	<b>Abstract</b>	<b>2</b>
<b>2</b>	<b>Keywords</b>	<b>5</b>
<b>3</b>	<b>Introduction</b>	<b>5</b>
<b>4</b>	<b>Problem formulation</b>	<b>5</b>
<b>5</b>	<b>Research Question</b>	<b>6</b>
<b>6</b>	<b>Background</b>	<b>6</b>
6.1	Description of the dataset . . . . .	6
6.2	Artificial intelligence . . . . .	10
6.3	Machine learning . . . . .	10
6.4	Artificial Neural Network . . . . .	10
6.5	Programming Languages . . . . .	12
<b>7</b>	<b>Structure and Function</b>	<b>12</b>
7.1	Feed-forward . . . . .	12
7.2	Backpropagation . . . . .	13
7.3	Activation Function . . . . .	14
7.4	Overfitting and underfitting . . . . .	14
7.4.1	Overfitting . . . . .	15
7.4.2	Underfitting . . . . .	16
<b>8</b>	<b>Program and process considerations</b>	<b>16</b>
<b>9</b>	<b>Design consideration</b>	<b>17</b>
<b>10</b>	<b>Methodology</b>	<b>18</b>
10.1	The program language of choice . . . . .	18
10.2	Data structure . . . . .	19
10.3	Algorithm . . . . .	20
10.4	Scalability . . . . .	25
<b>11</b>	<b>Procedure</b>	<b>27</b>
11.1	Data Processing . . . . .	27
11.2	Model architecture . . . . .	28
11.3	Training . . . . .	29

11.4 Evaluation . . . . .	29
<b>12 Prototype</b>	<b>29</b>
<b>13 Test of prototype</b>	<b>30</b>
<b>14 Testing the Learning Rate</b>	<b>34</b>
<b>15 Program description</b>	<b>35</b>
15.1 Network . . . . .	35
15.1.1 Neural Network Constructor . . . . .	36
15.1.2 feedForward() . . . . .	36
15.1.3 backpropagation() . . . . .	36
15.2 Helper Classes . . . . .	38
15.2.1 FileReaderHelper . . . . .	38
15.2.2 DataAndTrainingHelper . . . . .	39
15.2.3 Matrix . . . . .	40
15.3 GraphicalUserInterface . . . . .	40
<b>16 Calculations</b>	<b>40</b>
16.1 Output calculation . . . . .	41
16.2 Error calculation . . . . .	41
16.3 Updating weights between hidden-output layer . . . . .	41
16.4 Updating weights between input-hidden layer . . . . .	42
<b>17 Test of the program</b>	<b>42</b>
17.1 Prediction Testing . . . . .	42
17.2 Training Testing . . . . .	44
<b>18 User guide</b>	<b>46</b>
<b>19 Ethical Dilemma and Risks</b>	<b>51</b>
<b>20 Conclusion</b>	<b>52</b>
<b>21 Bibliography</b>	<b>54</b>

## 2 Keywords

Neural network, data, output, input, learning, layer, training, model, breast cancer

## 3 Introduction

Breast cancer is the most frequent type of cancer worldwide. and the importance of early detection for effective treatment and increased survival rates has long been recognized [1]. Machine learning techniques, particularly artificial neural networks, have shown promising results in medical diagnosis in recent years [2] . The current study investigates the application of neural networks in breast cancer detection by training them on a dataset of breast cancer patients to properly categorize tumors as benign or malignant based on several characteristics such as size, form, and texture. The project's purpose is to get a deeper understanding of neural networks' potential for enhancing the accuracy and efficiency of breast cancer diagnosis

## 4 Problem formulation

The problem of determining breast cancer can be formulated as a binary classification task using neural networks. The goal is to create a model that accurately classifies whether a given input, typically a set of features derived from diagnostic images or patient medical records indicate the presence (malignant) or absence (benign) of breast cancer.

We begin with a dataset of instances, each representing a particular case with a set of input features (e.g., characteristics of cell nuclei from a digitized image of a fine needle aspirate of a breast mass) and a target output (i.e., malignant or benign). The dataset is divided into a training set, which the neural network uses to learn, and a testing set, which is used to evaluate the network's performance [3].

The learning process involves presenting the neural network with the input features and adjusting the network's weights and biases based on the discrepancy between its predictions and the actual output. This adjustment process is iterated multiple times, to minimize the error made by the network. Once the network is trained, it can be used to classify new instances. Given a set of input features for a particular case, the network will provide a prediction of whether the case is malignant or benign. The performance of the network can be evaluated using several metrics, such as accuracy, precision, recall,

and F1 score [4].

The main challenge in this problem is to select and design the most suitable neural network architecture and training procedure to ensure high accuracy and generalization ability, while avoiding overfitting the training data. Additionally, interpretability of the model is a crucial factor, as the decisions made by the model have significant implications for patient treatment and care [5].

## 5 Research Question

What is the accuracy and reliability of using neural networks in breast cancer diagnosis, and how does it compare to traditional methods?

## 6 Background

*This section aims to provide the needed knowledge and information to fully understand the project.*

### 6.1 Description of the dataset

The dataset selected for the experiment is from the machine learning repository website. Which is a publicly available dataset that contains information about breast cancer tumor characteristics. The dataset contains 699 instances, with each representing measurements of breast cancer tumors. These measurements are collected from digitalized images of a breast mass. Each instance is labeled as either malignant or benign based on the analysis of the biopsy of the sample. The output is either 4 for malignant or 2 for benign. Moreover, the dataset consists of 10 attributes and ID numbers [6].

The features are:

- Sample code number: ID number
- Clump thickness: 1-10
- Uniformity of cell shape: 1-10

- Uniformity of cell size: 1-10
- Marginal adhesion: 1-10
- Single Epithelial cell size: 1-10
- Bare nuclei: 1-10
- Bland chromatin: 1-10
- Normal nucleoli: 1-10
- Mitosis: 1-10
- Class: 2 for benign, 4 for malignant

## **Visualization of the dataset**

The report aims to classify the dataset features (inputs) based on their output and by analyzing the output, the bellow graph shows the distribution of the output. To conduct such a graph dataset has to be put into a data frame that consists of 11 columns and 699 rows. Furthermore, a heatmap was performed to visualize the correlation and impact of each feature on the output. From the heatmap it could be concluded that the Nuclei, Cshape, and Csize are the most correlated features with the output, hence are more likely to determine the final output (Figures 1, 2, 3, 4).

```
]: data
```

```
]:
```

	id	Clump	Csize	Cshape	Adhesion	SECSize	Nuclei	Chromatin	Nucleoli	Mitosis	Class
0	1000025.0	5.0	1.0	1.0	1.0	2.0	1.0	3.0	1.0	1.0	2.0
1	1002945.0	5.0	4.0	4.0	5.0	7.0	10.0	3.0	2.0	1.0	2.0
2	1015425.0	3.0	1.0	1.0	1.0	2.0	2.0	3.0	1.0	1.0	2.0
3	1016277.0	6.0	8.0	8.0	1.0	3.0	4.0	3.0	7.0	1.0	2.0
4	1017023.0	4.0	1.0	1.0	3.0	2.0	1.0	3.0	1.0	1.0	2.0
...	...	...	...	...	...	...	...	...	...	...	...
694	776715.0	3.0	1.0	1.0	1.0	3.0	2.0	1.0	1.0	1.0	2.0
695	841769.0	2.0	1.0	1.0	1.0	2.0	1.0	1.0	1.0	1.0	2.0
696	888820.0	5.0	10.0	10.0	3.0	7.0	3.0	8.0	10.0	2.0	4.0
697	897471.0	4.0	8.0	6.0	4.0	3.0	4.0	10.0	6.0	1.0	4.0
698	897471.0	4.0	8.0	8.0	5.0	4.0	5.0	10.0	4.0	1.0	4.0

699 rows × 11 columns

Figure 1: The figure, obtained from our own project, shows the data set which is distributed in columns and rows.

```
In [21]: # Show the statistics of the dataset (mean, standard deviation, minimum, maximum, quartiles)
print("Statistics of the dataset:")
data.describe()
```

Statistics of the dataset:

```
Out[21]:
```

	id	Clump	Csize	Cshape	Adhesion	SECSize	Nuclei	Chromatin	Nucleoli	Mitosis	Class
count	6.990000e+02	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000
mean	1.071704e+06	4.417740	3.134478	3.207439	2.806867	3.216023	3.544656	3.437768	2.866953	1.589413	0.344778
std	6.170957e+05	2.815741	3.051459	2.971913	2.855379	2.214300	3.601852	2.438364	3.053634	1.715078	0.475636
min	6.163400e+04	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	8.706885e+05	2.000000	1.000000	1.000000	1.000000	2.000000	1.000000	2.000000	1.000000	1.000000	0.000000
50%	1.171710e+06	4.000000	1.000000	1.000000	1.000000	2.000000	1.000000	3.000000	1.000000	1.000000	0.000000
75%	1.238298e+06	6.000000	5.000000	5.000000	4.000000	4.000000	5.000000	5.000000	4.000000	1.000000	1.000000
max	1.345435e+07	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	1.000000

Figure 2: The figure, obtained from our own project, shows the overall description of the data set. The mean, minimum, maximum values as well as the standard deviation and count of elements in each column.



Distribution of Cancer Diagnosis

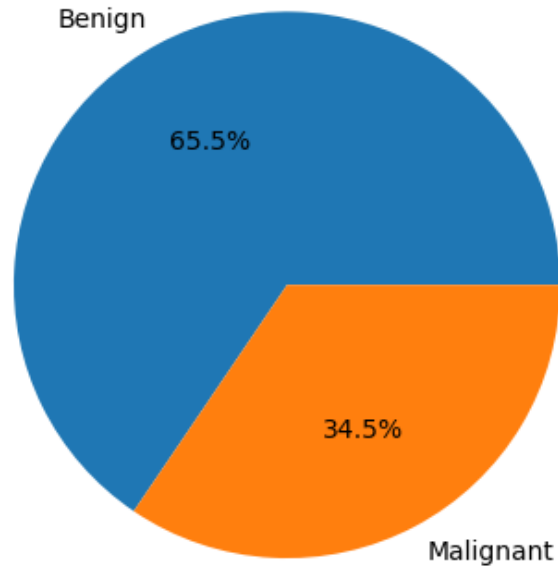


Figure 3: The pie chart, obtained from our own project, shows the distribution of the target.

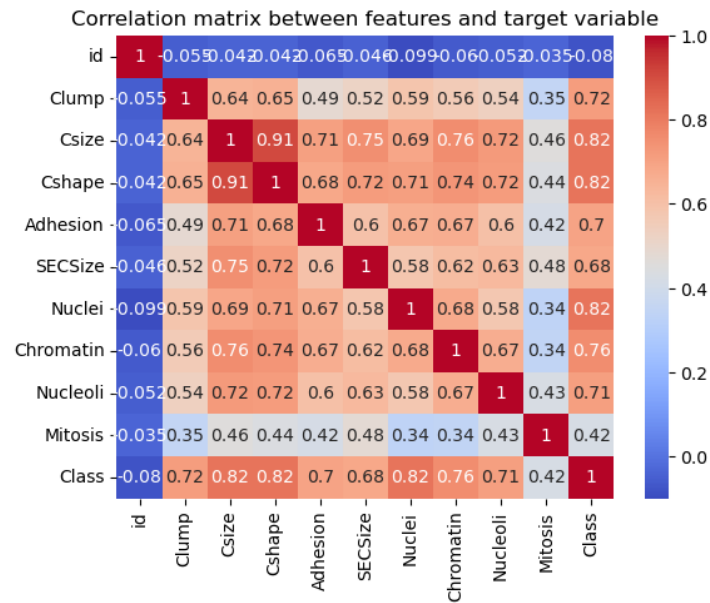


Figure 4: The heatmap, obtained from our own project, shows the correlation between the features and the target.

## 6.2 Artificial intelligence

At its simplest form, artificial intelligence is a field, which combines computer science and robust datasets, to enable problem-solving. It also encompasses sub-fields of machine learning and deep learning, which are frequently mentioned in conjunction with artificial intelligence. These disciplines are comprised of AI algorithms which seek to create expert systems which make predictions or classifications based on input data [7].

## 6.3 Machine learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy [8].

## 6.4 Artificial Neural Network

Artificial Neural Networks (ANNs), also referred as neural networks, are modern systems and computational approaches for machine learning, knowledge demonstration, and eventually the application of learned information to maximize the output responses of complex systems. ANN are used in many industries due to their capacity to learn from large datasets and provide precise predictions [9].

AAN is a data processing model that is based on the physiology of biological nervous system specifically the brain [9]. A neuron is the basic unit by which the brain functions [10]. The neuron consists of cell body with nucleus, an axon and synapses which are formed of dendrites (Figure 5) [11]. Electrical signals are transport from one end of a neuron to other. The signal is transmitted from dendrites through the axon body to the synapses which are connected to other neurons [10]. An artificial neuron resembles a biological neuron in that it has a neuron body and connections to other neurons [11]. They take in some data process it and give in an output [11].

A Neural network is composed of multiple neurons which are connected by links. Each link has a numeric weight associated to it. Weights indicate the impact of input on the calculation of the network output [11]. Each neurons input is calculated through multiplying the input with its corresponding weight and adding a bias. Afterwards output of the same neuron is calculated by adding them in activation function. The output of the neuron is a new input for the neighboring neuron (Figure 6) [10].

Also, ANN are used in natural language processing (NLP) to recognize linguistic patterns and perform operations such as sentiment analysis, language translation, and speech synthesis. Virtual assistants, chatbots, and voice recognition systems all benefit

from NLP. They also help medical professionals accurately diagnose patients, analyze large volumes of medical data, and determine which drugs will work best for a given patient based on their genetic profile.

There are several types of neural networks, the model chosen for this project is a multi-layer perceptron which is a feed forward neural network that consists of three types of layers. The first type is an input layer, the second is a hidden layer and the third output layer. The input layer is responsible for receiving input signals which are processed in the hidden layer. Moreover, the prediction occurs at the output layer. The smallest MLP consists of only one hidden layer and an input and output layer. Multiple hidden layers can be placed between the input and output layers. Data flows from the input layer to the output layer. The neurons are trained using a learning algorithm called back-propagation [12].

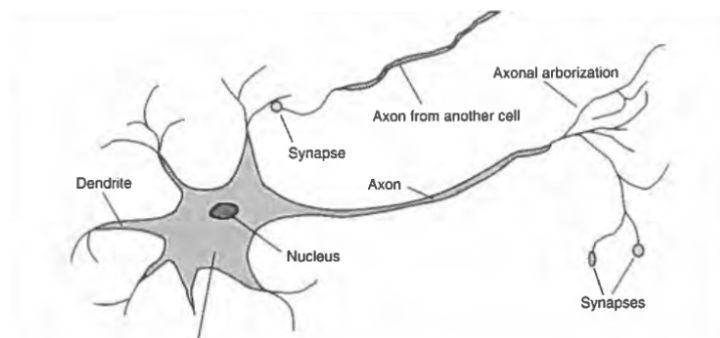


Figure 5: The figure shows the structure of a neuron, with all the components labelled [13].

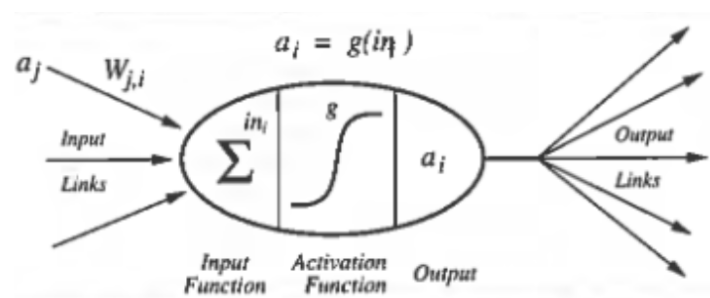


Figure 6: The figure represents artificial neuron 4 [13].

## 6.5 Programming Languages

A programming language is a collection of instructions, directives, and other syntax used to write software. Code-writing languages used by programmers are referred to as "high-level languages." This code may be converted into a "low-level language," which is understood by the computer hardware immediately after compilation. [14].

High-level languages are designed to be straightforward to read and comprehend. As a result, programmers are able to create source code naturally utilizing logical words and symbols. [14].

High-level languages include C++, Java, Perl, and PHP as examples. Since the source code needs to be first compiled in order to execute, "compiled languages" like C++ and Java are so named. Languages like Perl and PHP are referred to as "interpreted languages" because the source code may be executed through an interpreter without being compiled. In general, compiled programming languages are used to construct software programs, whereas interpreted programming languages are used to execute scripts, such as those that produce content for dynamic websites. [14].

## 7 Structure and Function

*This section allows the reader to understand the structure and function of the subject under investigation, making it easier to interpret results and draw conclusions.*

### 7.1 Feed-forward

Feedforward is a fundamental concept in neural networks that involves the transmission of information from input nodes to output nodes via intermediate layers without the use of a feedback loop [15]. In other words, information flows only one way, from the input layer to the output layer, via a series of hidden layers containing weights and biases that transform the input data into a useful output. Forward propagation is the process of passing information through the layers, and it is the foundation of many popular neural network architectures such as multi-layer perceptron (MLPs) and convolutional neural networks (CNNs) [16]. Feedforward neural networks have demonstrated great success in a variety of tasks such as image classification, natural language processing, and speech recognition, to name a few [17, 18].

Feedforward neural networks offer several benefits for this project. As the project involves predicting the likelihood of an event based on a set of input variables, a feedforward

network is well-suited for this task due to its ability to identify complex relationships between the inputs and outputs. By processing the input variables through multiple layers of neurons, a feedforward network can capture non-linear patterns and dependencies, which may not be possible with simpler models. Additionally, feedforward networks can be trained using a variety of techniques, such as backpropagation, which allows the network to learn from the data and improve its predictions over time. This flexibility and adaptability make feedforward networks an attractive option for this project. Finally, feedforward networks are widely used and well-understood, which means there are many resources available to help with the design, train, and evaluation of the network. Overall, the benefits of feedforward neural networks make them a compelling choice for this project.

## 7.2 Backpropagation

Backpropagation is a commonly used technique in machine learning for training artificial neural networks (ANNs) on a given dataset [19]. In the context of breast cancer tissue data, the goal of backpropagation is to help ANNs to accurately classify cells as either cancerous or non-cancerous based on different features of the tissue.

The backpropagation algorithm works by first randomly assigning weights to the connections between the input and output layers of the neural network. The network then goes through a forward pass, where input data is processed through the neuron layers until the output layer provides a predicted classification. This classification is then compared to the actual output value, and the error between them is calculated [19].

In order to reduce this error and improve the accuracy of the network, the backpropagation algorithm adjusts the weights of the connections in the network by "backpropagating" the error through each neuron layer in reverse order. This involves calculating the gradient of the error with respect to each weight in the network, and adjusting each weight in proportion to its contribution to the overall error. This process is repeated for multiple epochs until the error rate has reached an acceptable level [19].

In the case of breast cancer tissue data, the backpropagation algorithm may be used to identify which features of the cells are most indicative of cancerous growth. By adjusting the weights of the network in response to the error rate, the network may be able to learn which characteristics of the tissue are most strongly associated with cancer and use this knowledge to classify new samples with greater accuracy.

### 7.3 Activation Function

In early neural network architectures, the sigmoid function was widely used as an activation function, and it is still used in some contexts today [15, 20]. This smooth, S-shaped function maps any input to a value between 0 and 1, making it especially useful for binary classification tasks where the neural network output must be interpreted as a probability of belonging to a specific class. However, there are some limitations to the sigmoid function, such as the vanishing gradient problem, which can make training deep neural networks more difficult [15]. Despite these limitations, the sigmoid function is still a popular activation function in many situations. The sigmoid function is given by this formula:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

where  $x$  is the input to the function.

### 7.4 Overfitting and underfitting

The main problem in machine learning is that we must perform well on fresh, previously unseen inputs and those on which we trained our model. Generalization refers to the ability to perform well on previously unseen inputs [15].

In machine learning, the goal is to not only minimize the training error but also to have a low generalization error or test error, which is the expected value of the error on new inputs drawn from the distribution of inputs encountered in practice. This distinguishes machine learning from optimization problems where only the training error needs to be minimized [15].

Underfitting and overfitting are two main issues that can arise when training machine learning models. Underfitting occurs when a model is overly simple and fails to capture the data's complexity. As a result, the model is unable to learn the underlying patterns and relationships in the data and performs poorly on both training and testing data. To clarify, the model does not fit the training data well and does not make accurate assumptions to new data.

Overfitting, on the contrary, occurs when a model is overly complicated and closely fits the training data. This can result in a model that is extremely accurate on training data but performs poorly on fresh, unseen data. When a model learns to recognize noise in the training data rather than the genuine underlying patterns, this is referred to as overfitting.

In both situations, the objective is to get the best generalization performance by achieving the ideal balance between model complexity and the amount of accessible data. A well-fitted model should be able to perform effectively on new, previously unknown data (Figure 7) [15].

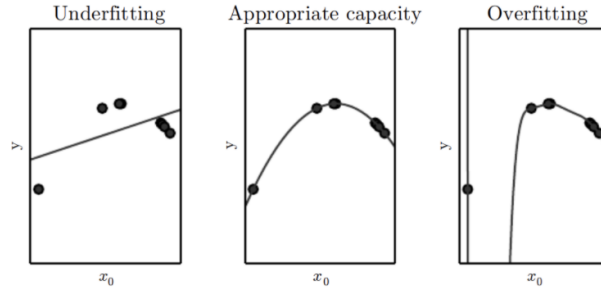


Figure 7: Underfitting, optimum capacity and overfitting [15].

#### 7.4.1 Overfitting

One of the main issues in training neural networks is overfitting, where the network learns to solve the training problem well but fails to generalize to new inputs. Overfitting is observed when the test error first reaches a minimum and then starts to increase again. Overfitting occurs when the model fits random errors or noise instead of the underlying relationship. Various studies have highlighted this issue and provided insights into its nature. Figure 1 shows a graphical representation of the phenomenon [21].

When a model underfits, it has a high bias and low variance, while overfitting is characterized by low bias and high variance. Often, even a small increase in bias can lead to a significant decrease in variance.

If the learning algorithm is overly focused on the training data and is not able to generalize well to new data, it is possible can try the following strategies to address the issue of overfitting:

- Use fewer features: Reducing the number of features used in the model can lower the variance and prevent overfitting [22].
- Collect more data: Gathering additional training data can help to reduce overfitting by providing more examples for the model to learn from [22].

- Early stopping: Stopping the training of the model early, before it starts to overfit, can help to prevent overfitting [21].

#### 7.4.2 Underfitting

Underfitting is the opposite of overfitting. It occurs when the model is not able to capture the variability of the data. For example, if a linear classifier is trained on a data set that is a parabola, the resultant classifier will have no predictive power or ability to properly map the training data. This is due to using a model that is too simple to describe the data [21].

If the algorithm used for learning is not fitting the data well and has high bias, there are certain recommendations that can be followed to improve the situation and avoid underfitting:

- Use more features: To improve the model's ability to make predictions, we should consider adding more features to it [22].
- Try a more complicated model: Consider using a more intricate model to improve bias but be careful not to make it too complex as it could lead to overfitting [22].

## 8 Program and process considerations

To reach the desired model structure several discussions were made. A variety of neural network models were studied. The ideal model is a simple neural network that can be built and perform accurate predictions in a limited time which is also beginner friendly.

Several considerations were taken place. The type of neural network, the programming language, algorithms, and design of the neural network and the time required to accomplish such a task. First, the programming language was decided based on familiarity and effectiveness of the language to carry out the task. The language chosen for programming is Java which was coded in an IDE called IntelliJ. The idea for the project was to establish a neural network from scratch. Therefore, a simple model was to be created and then scaled after grasping concrete knowledge behind the fundamentals. As a starter, the model was built as a perceptron the 1st neural network model which is a single-layer neural network. Afterward, the program was scaled to include other layers with all input.

The Neural network used is MLP which is a feed-forward neural network that utilizes backpropagation as its algorithm. also after further investigation, multiple sources have



been associating the effectiveness of the model with the usage of backpropagation.

Lastly, the design of the neural network was based on research that the significance of multiple hidden layers is quite doubtful if inputs are large, and the same result can be obtained via multiple interrelations between the inputs and one hidden layer. The training process was trial and error where the weights and biases kept getting updated as well as the learning rate to obtain the desired result. The overfitting problem was discussed to find the best method to overcome it. The validation of the model was done by inputting unseen data into the model and evaluating the results.

## 9 Design consideration

1. **Architecture of neural networks** An important factor in this research is the architecture of the neural network design. A feedforward neural network is ideally suited for this task since the objective is to reliably identify breast cancers as benign or malignant based on tumor features. In order to analyze the wide range of tumor properties, feedforward networks must be able to capture complicated interactions and nonlinear patterns. There will be an input layer, many hidden layers, and an output layer in the neural network's unique design. Through testing and optimization, the number of neurons in each layer and the selection of activation functions will be established. Striking a balance between a network that is sufficiently intricate to capture the nuances of the data and one that prevents overfitting is crucial.
2. **Prepare a Dataset** Before training the neural network, the dataset from the UCI Machine Learning Repository must be carefully prepared. We'll take actions like cleaning up the data, dealing with missing values, and normalizing or standardizing the characteristics. To guarantee a fair assessment of the trained models, the dataset will also be divided into training, validation, and testing sets. The proper stratification will be used to keep both benign and malignant tumors represented in each dataset subgroup.
3. **Instruction and Assessment** Backpropagation and gradient descent methods will be used to adjust the network's weights and biases throughout the training phase. Accuracy, precision, recall, and F1-score are some suitable assessment measures that will be used to assess the network's performance. To evaluate the trained models' capacity for generalization, cross-validation techniques may be used. Regularization methods like dropout or L1/L2 regularization may be used to avoid overfitting. To improve the network's architecture, learning rate, regularization strength, and other important parameters, hyperparameter tweaking will be used.

To avoid overfitting and increase training effectiveness, early quitting may also be used.

4. A graphical user interface (GUI) will be created to improve comprehension of the breast cancer detection procedure. Users may enter tumor characteristics and view the forecast provided by the trained neural network using the GUI. The user-friendly design of this interactive tool will make it easier to understand and convey the model's predictions. The GUI will help medical practitioners understand the aspects affecting the diagnosis and develop confidence in the model's predictions.
5. Comparison with Traditional Methods and Validation The efficiency of the neural network methodology will be evaluated by contrasting the trained models' performance with that of more conventional techniques like mammography and biopsy. In order to establish the superiority or complementarity of the neural network models, their accuracy and dependability will be measured against these standards. To reach relevant findings, statistical analysis and the proper significance testing techniques will be used.

In conclusion, significant thought should be given to the neural network design, dataset preparation, training, assessment, GUI display, and ethical issues. These factors will all affect the project's success and legitimacy. This research seeks to enhance the accuracy and reliability of breast cancer diagnosis, which will eventually result in more efficient treatment and better patient outcomes.

## 10 Methodology

*This section aims to describe the approach and process taken in conducting the research or experiment. It provides a detailed explanation of the methods used.*

### 10.1 The program language of choice

Java is a well-liked and adaptable programming language that is suitable for creating software for a variety of businesses, including the healthcare and medical fields. Here are some advantages and disadvantages of using it, based on the perspective of the group:

- Pros:
  - Platform-independent: Java is a flexible language for creating applications that may be used on a variety of devices, including desktop computers, mobile phones, and tablets. Java programs can operate on numerous platforms and operating systems, this makes it accessible for medical purposes such as the diagnosis of cancer [23].

- Robust and secure: Java contains built-in error-handling and security safeguards, making it a dependable language for creating applications that need high security requirements, such those used in the healthcare industry [23].
  - Large developer community: The Java programming language has a sizable and vibrant developer community that produces and upholds a variety of libraries and frameworks that can be utilized to more quickly and effectively construct medical applications [23].
- Cons:
    - Java programs are memory-hungry, which makes them unsuitable for tasks requiring a lot of resources, such as those found in some medical facilities.
    - Slow start-up times: Java programs frequently take a long time to start up, which can be problematic for programs that must operate swiftly and effectively, particularly in time-sensitive medical situations [23].
    - Steep learning curve: Java is a verbose, sophisticated language that takes a lot of technical expertise to utilize successfully, making it difficult for developers who are just starting out in the field [23].

## 10.2 Data structure

The dataset used in this project comes in a text file. To use the dataset, it must be interpreted, separated, and stored. A data structure is a way to store and organize data in a computer's memory.

In this project, several types of non-primitive data structures are used. Two lists which are an array list and a linked list. Additionally, a two-dimensional array is used as well. An array is a group of identical data components kept in contiguous memory regions. A linked list is made up of nodes that are organized linearly. Each node in a single linked list keeps a reference to an object that is an element of the sequence as well as a reference to the node after it. The two important nodes in a linked list are the head and the tail. Since the addition of a node or subtraction of elements happens in the head's node. The tail node shows where the linked list ends. The linked list instance should keep a reference to the head. An array list keeps track of the elements in an array, the difference between an array and an array list is the size. Array list size can be changed meanwhile arrays have a fixed size. Array store different types of primitive data structures and objects meanwhile array list stores only objects. These structures play a significant role in the storage of the data and facilitate the operations needed to form the neural network model. Operations such as adding, removing, accessing, searching, and updating the elements are facilitated. To do such operations typically,

elements of the linked list, array list, or array are put into a loop that iterates over all the elements to either find the position to add the element or erase it. It also could replace an existing value with a new one and update the array list, linked list, or array. Moreover, this kind of data structure allows full access to all elements of the array by using their indices. Last, but not least, arithmetic operations can be applied to them [24].

### 10.3 Algorithm

As stated before the model is a multi-layer perception neural network. The model consists of an input layer that takes 10 inputs which are the features of the dataset and a hidden layer with 6 hidden units (neurons) moreover an output layer. The algorithm used in this project is backpropagation. The typical MLP consists of two steps a forward pass step and a backward pass step.

#### Forward Pass

In the forward pass part of the neural network, the information follows from the input to the output. Each neuron has an input and an output the output of a layer is the input of another layer until it reaches the output. In Figure 8, a simple example of the MLP model is illustrated. The model takes up Inputs  $I_1$  and  $I_2$  and multiplies them with their corresponding weights then add a bias to them.

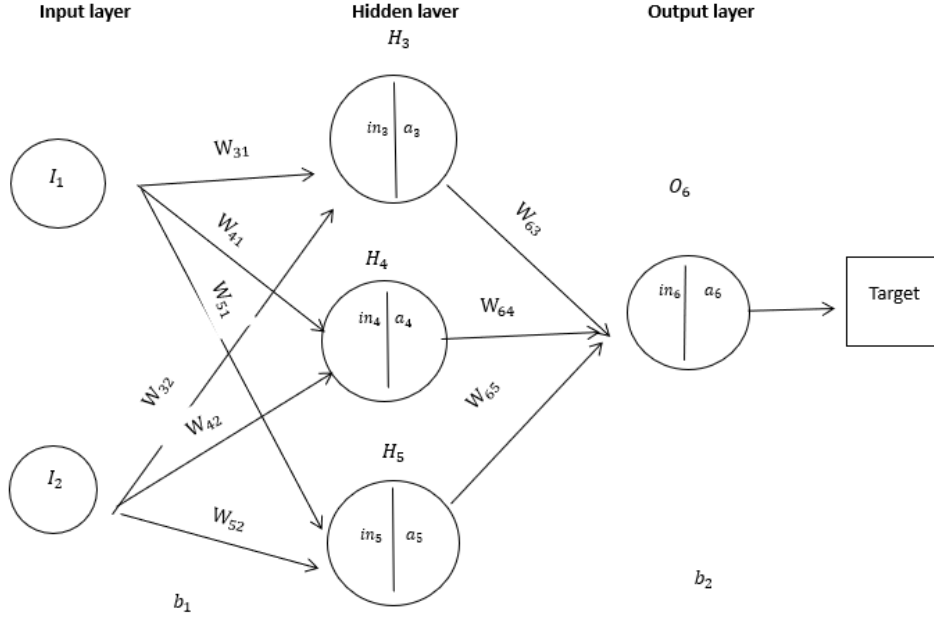


Figure 8: Neural network with 2 inputs one hidden layer and one output.

Before calculations are presented it is worthy to announce notations. In the examples  $I$  is the input,  $a$  is the output of a neuron,  $W$  is the weight,  $b$  is the bias and  $in$  is the input for a neuron. The model calculations are divided into 3 layers input layer, hidden layer, output layer.

### Input layer

In the input layer the inputs are multiplied with the weights and a bias is added to them.

### Hidden layer

To calculate the input for the hidden layer  $H_3$   $I_1$  is multiplied by  $W_{31}$  and  $I_2$  is multiplied by  $W_{32}$ . To make the neural network more complicated a bias is added to each unit in the hidden layer and the output layer. Making the input for  $H_3$  be  $in_3 = I_1 * W_{31} + I_2 * W_{32} + b_1$ . These parameters are passed in the sigmoid function of  $H_3$ :

$$\text{function } \sigma(in_3) = 1 - \frac{1}{(1 + e^{-in_3})} \quad (2)$$

The output of the hidden layer is conducted by passing the parameters to the sigmoid function. Making the output of the  $H_3$  unit be:

$$a_3 = \sigma(in_3) = 1 - \frac{1}{1 + e^{-in_3}} \quad (3)$$

The same is done for the other hidden layers, as well as for example the input of the  $H_4$ , being:

$$in_4 = I_1 * W_{41} + I_2 * W_{42} + b_1 \quad (4)$$

And the output is:

$$a_4 = \sigma(in_4) = 1 - \frac{1}{1 + e^{-in_4}} \quad (5)$$

### Output layer

The output neuron takes the output of the hidden layer. The input of the output layer is calculated by taking the output of the hidden layer and multiply it with a new weight then add a bias to it. The output of  $H_3$ , is multiplied by a new weight  $W_{63}$  and as well as the output of  $H_4$ ;  $W_{64}$ ; and  $H_5$  with  $W_{65}$ .

$$in_6 = H_3 * W_{63} + H_4 * W_{64} + H_5 * W_{65} + b_2 \quad (6)$$

The final output which is the predicted output by the model is calculated throw passing the input  $in_6$  to the sigmoid function [11, 13].

$$a_6 = \sigma(in_6) = 1 - \frac{1}{1 + e^{-in_6}} \quad (7)$$

### Backward Pass

To estimate the accuracy of the model an error function is conducted which subtracts the target output from the predicted output. Normally there is a difference between the predicted output and the target. In order to minimize the error backpropagation is utilized. In this method the contribution of each weight to the error is calculated to then update the weight. In this method the flow of the network starts from the output to the input taking the opposite direction of the network.

$$E_{rr} = T - a_6 \quad (8)$$

## Updating Weights

To change the weight  $W_{64}$ , the following calculations are done.  $\alpha$  is the learning rate. The partial derivative of the error in respect to the weight is taken. The learning rate is a random number.

$$W_{64} \leftarrow W_{64} + \alpha \frac{\partial E_{rr}}{\partial W_{64}} \quad (9)$$

$$\frac{\partial E_{rr}}{\partial W_{64}} = \frac{\partial E_{rr}}{\partial a_6} * \frac{\partial a_6}{\partial in_6} * \frac{\partial in_6}{\partial W_{64}} \quad (10)$$

$$\frac{\partial E_{rr}}{\partial a_6} = \frac{\partial (T - a_6)}{\partial a_6} = T - 1 \quad (11)$$

$$\frac{\partial a_6}{\partial in_6} = \sigma'(in_6) \quad (12)$$

$$\frac{\partial in_6}{\partial W_{64}} = a_4 \quad (13)$$

## Derivative of sigmoid function

The derivative of sigmoid function is as follows:

$$\sigma(in_6) = 1 - \frac{1}{1 + e^{-in_6}} \quad (14)$$

$$\sigma'(in_6) = 1 - \frac{e^{-in_6}}{(1 + e^{-in_6})^2} \quad (15)$$

$$\sigma'(in_6) = \sigma(in_6)[1 - \sigma(in_6)] \quad (16)$$

$$\sigma(in_6) = a_6 \quad (17)$$

$$\sigma'(in_6) = a_6(1 - a_6) \quad (18)$$

$$\frac{\partial E_{rr}}{\partial W_{64}} = (T - 1) * (a_6) * (1 - a_6) * a_4 \quad (19)$$

The first half of the equation is denoted by Delta.

$$\delta_6 = (T - 1) * (a_6) * (1 - a_6) \quad (20)$$

$$\frac{\partial E_{rr}}{\partial W_{64}} = \delta * (a_4) \quad (21)$$

## Updating biases

Biases are updated through the same mechanism by finding the slope where bias is when the error is the minimum. Error is differentiated to the respect of bias [11, 13].

$$\frac{\partial E_{rr}}{\partial b_2} = \frac{\partial E_{rr}}{\partial a_6} * \frac{\partial a_6}{\partial in_6} * \frac{\partial in_6}{\partial b_2} \quad (22)$$

$$\frac{\partial E_{rr}}{\partial b_1} = \frac{\partial E_{rr}}{\partial a_6} * \frac{\partial a_6}{\partial in_5} * \frac{\partial in_5}{\partial b_2} \quad (23)$$



To avoid long calculations a general equation which is derived from the formal calculations that is applicable to all weights and biases is used.

$$\frac{\partial E_{rr}}{\partial W_{jk}} = \delta_j^{(l)} * a_k^{(l-1)} \quad (24)$$

$$\frac{\partial E_{rr}}{\partial b_j^{(l)}} = \delta_j^{(l)} \quad (25)$$

## 10.4 Scalability

The scalability of the project, aimed at detecting malignant cells based on scans, can be assessed considering the information below.

### Project Scope

The necessity for scalability in managing a potentially high number of medical data is implied by the project's goal of detecting cancerous cells from scans. To increase the precision and dependability of the established models, it could be required to add more scans and grow the dataset as the project goes on.

### Team Composition

The team is made up of five people who are studying computer science at different levels, two of which with backgrounds in molecular biology. This broad range of abilities permits a project-wide interdisciplinary approach. The team's capacity to draw on its collective experience, adjust to new difficulties, cope with ever-more complicated datasets, or use cutting-edge technologies is what defines its scalability.

### Project Timeline

Given that the project's completion date is scheduled for the end of May, scalability issues need to be taken into account to make sure that the project can be finished on time. The team should make appropriate plans for their duties and deadlines, including room for flexibility in case the project's scope is increased or new difficulties arise.

### Resources

Evaluate the project's necessary resources' availability and scalability. This involves having access to computer systems and software tools for data preparation, training, and evaluating models, as well as medical scans. As the project develops, think about whether the resources can handle any future increases in the amount of data or processing demands.

### **Data Size and Complexity**

Because the project entails scanning for cancerous cells, scaling requires taking into account the possibility of an increase in the size and complexity of the dataset. Make sure the algorithms and approaches you choose can handle increasing data quantities without sacrificing performance and accuracy.

### **Computational Requirements**

Analyze the project's computing needs, including the processing power, memory, and storage space required for developing and assessing the models. Analyze the computing infrastructure of the team's scalability to support anticipated growth in data amount and model complexity.

### **Collaboration and Communication**

Options include those presented through university meetings, social media applications, and regular interactions with the supervisor. Make that the chosen communication channels can expand to meet the needs of the project, such as accommodating prospective new team members or a growing workload.

### **Evaluation Metrics**

Select the best evaluation metrics to judge the project's scalability. This may take into account things like the capacity to handle bigger datasets, more demanding processing needs, and the ability to process data in the allotted amount of time.

In conclusion, the project's scalability must take into account the capacity to manage an expanding dataset, adjust to changing computational needs, and scale communication and collaboration techniques. By taking care of these issues, the team can make sure that the project stays on schedule and makes the most use of the resources available while successfully advancing toward its objective of identifying cancerous cells based on scans.

## 11 Procedure

This section presents a throw explanation of the procedure. The procedure consists of 3 steps which are Data processing, model architecture, training and evaluation.

### 11.1 Data Processing

To make use of the data it had be interpreted using a class called `BufferedReader` which is a class that makes reading text from a character input stream easier by buffering the characters. It has methods such as `readLine` that reads data line by line. The dataset is split into 2 sections, one is for the training dataset and the other section is left for validation. Afterwards inconsistency in dataset is investigated and 16 missing values are handled (Figure 9). Python language was used for handling missing values, description, and visualization of the distribution of the dataset. Python is known for its built-in libraries which facilitate data visualization. Missing values are handled via a series of codes that looks for the location of the missing values and replace them. Usually, missing values are replaced by the mean values of the column that they are missing from. All 16 missing values are located in a single column named `Nuclei` and therefore replaced by 1. The target values which are either 2 or 4 are changed into binary 0 and 1. 4 is changed to 1 and 2 is changed to 0 (Figure 10).

```

# Count the number of missing values in each column
missing_values = (data == '?').sum()

# Print the number of missing values in each column
print("Number of missing values in each column:")
print(missing_values)

Number of missing values in each column:
id          0
Clump       0
Csize       0
Cshape      0
Adhesion    0
SECSIZE     0
Nuclei     16
Chromatin   0
Nucleoli    0
Mitosis     0
Class       0
dtype: int64

# Replace missing values (represented as '?') with numpy's NaN
data = data.replace('?', np.nan)

# Impute missing values with the mean value of the corresponding feature
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
data = pd.DataFrame(imputer.fit_transform(data))

# Set column names again because of column names are removed after imputation
data.columns = columns

```

Figure 9: The figure, obtained from our own project, shows how the missing values were handled.

	id	Clump	Csize	Cshape	Adhesion	SECSIZE	Nuclei	Chromatin	Nucleoli	Mitosis	Class
0	1000025.0	5.0	1.0	1.0	1.0	2.0	1.0	3.0	1.0	1.0	0.0
1	1002945.0	5.0	4.0	4.0	5.0	7.0	10.0	3.0	2.0	1.0	0.0
2	1015425.0	3.0	1.0	1.0	1.0	2.0	2.0	3.0	1.0	1.0	0.0
3	1016277.0	6.0	8.0	8.0	1.0	3.0	4.0	3.0	7.0	1.0	0.0
4	1017023.0	4.0	1.0	1.0	3.0	2.0	1.0	3.0	1.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
694	776715.0	3.0	1.0	1.0	1.0	3.0	2.0	1.0	1.0	1.0	0.0
695	841769.0	2.0	1.0	1.0	1.0	2.0	1.0	1.0	1.0	1.0	0.0
696	888820.0	5.0	10.0	10.0	3.0	7.0	3.0	8.0	10.0	2.0	1.0
697	897471.0	4.0	8.0	6.0	4.0	3.0	4.0	10.0	6.0	1.0	1.0
698	897471.0	4.0	8.0	8.0	5.0	4.0	5.0	10.0	4.0	1.0	1.0

Figure 10: The figure, obtained from our own project, indicates the dataset after handling missing values and changing the target values to 0 and 1.

## 11.2 Model architecture

An MLP neural network was chosen for this project. All the features of the dataset are taken as input. Each input is connected to a hidden layer via weight. However, input( $X_1$ - $X_n$ ) are connected to several neurons of the hidden layer. The hidden layer

is connected to the output. To facilitate these connections several matrices are created which are linked lists.

The matrix stores the training data where the elements are going to be extracted from. The second matrix stores the weights from the inputs to the hidden layer. The third matrix stores weights from the hidden layer to the output. The fourth matrix stores bias for the hidden layer and the fifth the bias for the output. The sixth and seventh for the input and the output. This arrangement simplifies the multiplication of the weights with the  $x_n$  input and the outputs of the hidden layer. In the hidden layer the result of the multiplication of the input by weights and addition of the bias is put into sigmoid function. The output of the sigmoid function is then multiplied by the third matrix and a bias is added to it. Finally, the output is stored in a list.

### 11.3 Training

The model is trained by 75% of the dataset. Training of the model consists of forward flow of the data from the input to the output and comparing the output with the actual output (Target). Since the weights and biases are randomized, there is a difference between the actual output and the predicted output. To solve this problem backpropagation is used. By constantly tracing back from the out to the different weights and updating them until the desired output is reached.

### 11.4 Evaluation

To validate and evaluate the model unseen data has been fed to the model which is a 25% of the total dataset .The accuracy of the model is obtained by calculating the difference between the actual value and the target.

## 12 Prototype

In this section, we will introduce the prototype of our program and provide a comprehensive overview of the tests conducted to evaluate its functionality before proceeding with further program development. The program code will not be extensively discussed here but will be presented in detail in the subsequent section dedicated to the "Program Description". Initially, the development of our neural network began with the creation of a rudimentary model that enabled us to assess the program's functionality.

The prototype is consisting of two classes of Network and Main. In the network class we have a functioning neural network that can take the number of inputs, outputs, and the number of the hidden neurons. The `train()` function in the Network class is responsible

for calculating the hidden neurons' value, predict the output and calculate the error and make changes to the biases and weights accordingly. After training the network at the Main class, the `predict()` function can be used to test the network and predict the result.

## 13 Test of prototype

The neural network was tested using the XOR problem, a classical test case for evaluating the performance of a neural network. The OR problem was also given to the neural network, however because of the simplicity of that problem, we decided to omit that from the report and try the XOR problem. Our test methodology involved training the network using various combinations of input and output values, and then comparing the predicted outputs to the actual values (Figure 11). We evaluated the accuracy of the network by measuring its mean squared error, which provides a quantitative measure of how closely the predicted values match the actual values [25].

```

The weights between inputs and hidden layer before the training:
Weight between Input i1 and p1 : 0.9264983194831131
Weight between Input i1 and p2 : 0.3942307684025166
Weight between Input i1 and p3 : -0.24419983330156625
Weight between Input i2 and p1 : -0.6053450552124937
Weight between Input i2 and p2 : -0.7875764787910031
Weight between Input i2 and p3 : 0.09121746844755174

The weights between hidden layer and output before the training:
Weight between Hidden p1 and o1: 0.8401988105797005
Weight between Hidden p2 and o1: -0.865582318825941
Weight between Hidden p3 and o1: 0.3693049209931776

The weights between inputs and hidden layer after the training:
Weight between Input i1 and p1 : 7.665677642104512
Weight between Input i1 and p2 : -10.125766966371494
Weight between Input i1 and p3 : -7.326307225480817
Weight between Input i2 and p1 : -7.35419422230273
Weight between Input i2 and p2 : -10.066325606122795
Weight between Input i2 and p3 : 7.550948063869745

The weights between hidden layer and output after the training:
Weight between Hidden p1 and o1: 6.534314336776651
Weight between Hidden p2 and o1: -3.086401794809779
Weight between Hidden p3 and o1: 6.536484636284789

```

*Figure 11: The figure, obtained from our own project, shows the changes made between the input, output, and hidden layer weights after the training.*

In this report, we provide an overview of the architecture and algorithms of the prototype neural network, describe our test methodology, and present the results of the tests. Our findings indicate that the prototype neural network is capable of accurately predicting the output of the XOR problem, with a mean squared error of less than 0.1. We also discuss the limitations of the prototype network and future directions for its development, including the use of more complex architectures and algorithms for solving more advanced problems.

The network is consisting of two input neurons, three hidden neurons, and one output neuron (Figure 12). We keep track of the weights between inputs and hidden layer, and the hidden layer and output to see if the network is working correctly. The start weights are generated with a random double between -1 and 1. Finally, we predict the result for the XOR with two inputs to see whether the results are correct or not. The reason for

choosing three hidden neurons is that compared to two hidden neurons, three neurons increased the accuracy of the predictions by 4-6%.

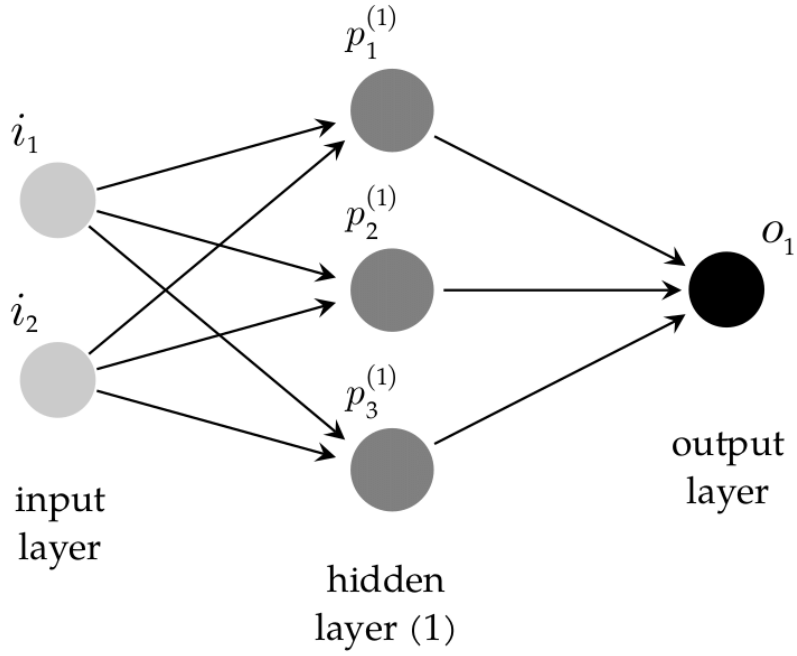


Figure 12: The figure, obtained from our own project, shows the prototype network representation.

In figures 13 and 14 there can be seen changes made to the weights between input, hidden layer, and output during the training process, the training was done 130,000 times to get to the optimal accuracy. After the process of training, inputs were given to the network to predict the outcome.



Truth Table		
B	A	Q
0	0	0
0	1	1
1	0	1
1	1	0

Figure 13: The figure, obtained from our own project, shows the XOR truth table.

```

0 XOR 0 = 0.06489482528362879
0 XOR 1 = 0.9717740326581343
1 XOR 0 = 0.8927508339138596
1 XOR 1 = 0.07144561312597444

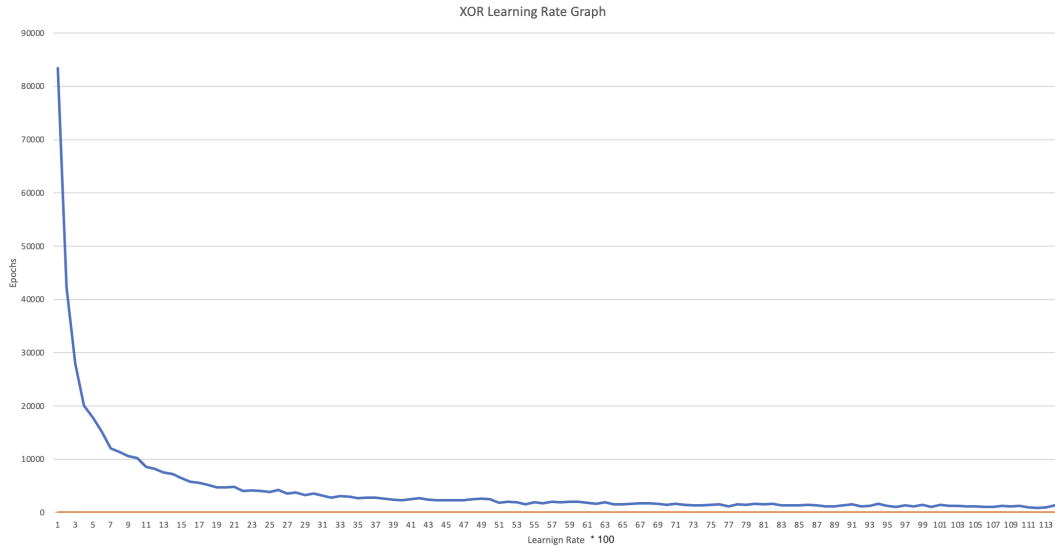
```

Figure 14: The figure, obtained from our own project, shows the XOR predicted output by the network.

## 14 Testing the Learning Rate

In order to see the connection between the learning rate and the number of epochs, a method was written to train the network for a certain number of times with small increases on epochs with the learning rate ranging between 0.01 and 1.14. The result then was written to a CSV file and a graph was generated to show the epochs and learning rate connection. This test was done for the XOR neural network. The test starts with learning rate of 0.01, and the epochs will increase until the result of XOR inputs have the error less than 0.1; then the learning rate is increased by 0.01 and the test is continued. It is clear to see that as the learning rate increases, the number of epochs decreases. This makes good sense, as the weights are thus adjusted by a larger factor, the greater the learning rate. Therefore, backpropagation does not have to be carried out as many times, and thus the number of epochs decreases.

However, with the XOR neural network, backpropagation could not be completed at a learning rate higher than 0.14 where the network started to predict results with errors greater than 40%. This could possibly be due to the learning rate being too large and the minimum values for the error not being met. Finding an optimal learning rate is a crucial hyperparameter tuning step in neural network training, as setting it too low may result in slow convergence, while setting it too high may lead to unstable or unsuccessful training. In figure 15 the X-Axis values are shown as (learning rate \* 100) for the sake of graph readability.



*Figure 15: Displaying the relationship between the learning rate and the number of epochs during the backpropagation process of our prototype of XOR prediction network (figure obtained from our prototype).*

## 15 Program description

The program code is consisting of three packages: GUI, NeuralNetwork, and Helper-Classes. In the next section, NeuralNetwork class is discussed in detail. A UML diagram of the program is shown in the figure below.

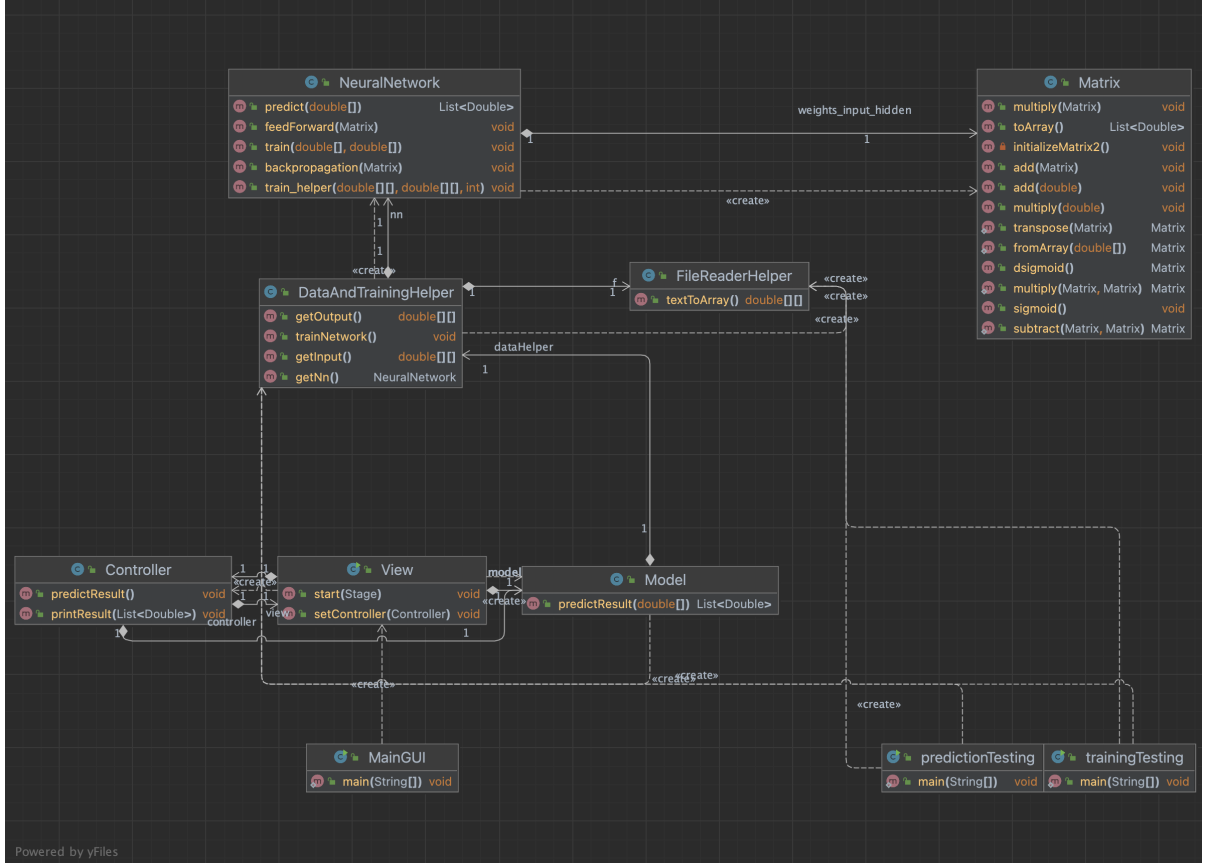


Figure 16: UML diagram of the project.

### 15.1 Network

This class is responsible for initiating the neural network with the desired number of neurons and training the network with the help of `feedforward()` and `backpropagation()` methods. The constructor and the methods of this class are explained in this section. Main data structure that is used in this class is the **Matrix** class which is written to help the network training. Finally, the learning rate in this class is assigned a specific value that has been empirically determined through rigorous testing and experimentation.

### 15.1.1 Neural Network Constructor

The `NeuralNetwork` class takes the number of inputs, number of outputs, and the number of hidden neurons and creates a three-layered network, one input, one output, and one hidden layer. The network is fully connected, meaning all the input, output, and hidden neurons are connected and have corresponding weights and biases between them. All the neurons' values, weights, and biases are stored in a separate matrix with the help of the `Matrix` class. During the creation of the weights and biases matrices, a random value between 0 and 0.1 is generated for all of them. This way, the initial values for the weights in the two weight layers are set with random values (Figure 17).

```
public NeuralNetwork(int i, int h, int o) {  
    //Create the weight matrices  
    weights_input_hidden = new Matrix(h, i);  
    weights_hidden_output = new Matrix(o, h);  
  
    //Create the bias matrices  
    bias_hidden = new Matrix(h, cols: 1);  
    bias_output = new Matrix(o, cols: 1);  
}
```

Figure 17: Figure obtained from our `NeuralNetwork` Constructor.

### 15.1.2 feedForward()

`FeedForward` method takes the input matrix and calculates a predicted output. This is done by multiplying the weights between the input layer and the hidden, adding the biases to the value, and running the value through the sigmoid function. The same procedure is done between the hidden layer and the output layer to generate the predicted output (Figure 17).

### 15.1.3 backpropagation()

`Backpropagation` method is responsible for training the network and updating the weights and biases in the network. As previously mentioned, the method is based on the pseudocode of Russel and Norvig [13], which can be seen in figure 18. Here we will review our `backpropagation()` method in relation to this pseudocode.

```

public void feedForward(Matrix input) {
    //CALCULATE THE HIDDEN NEURONS
    hidden = Matrix.multiply(weights_input_hidden, input);
    hidden.add(bias_hidden);
    hidden.sigmoid();

    //CALCULATE THE PREDICTED OUTPUT
    output = Matrix.multiply(weights_hidden_output, hidden);
    output.add(bias_output);
    output.sigmoid();
}

```

*Figure 18: Figure obtained from our Feedforward Method.*

Following this code, the backpropagation method calculates the error, gradient, and the delta; and adds the delta to the weights, and the gradient to the biases. This procedure is done twice, first time for the weights between output and hidden layer, and second time for the weights between hidden layer and inputs (Figure 20).

```

function BACK-PROP-UPDATE(network, examples,  $\alpha$ ) returns a network with modified weights
  inputs: network, a multilayer network
           examples, a set of input/output pairs
            $\alpha$ , the learning rate

  repeat
    for each e in examples do
      /* Compute the output for this example */
      O  $\leftarrow$  RUN-NETWORK(network, Ie)
      /* Compute the error and  $\Delta$  for units in the output layer */
      Erre  $\leftarrow$  Te - O
      /* Update the weights leading to the output layer */
       $W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \text{Err}_i^e \times g'(in_i)$ 
      for each subsequent layer in network do
        /* Compute the error at each node */
         $\Delta_j \leftarrow g'(in_j) \sum_i W_{j,i} \Delta_i$ 
        /* Update the weights leading into the layer */
         $W_{k,j} \leftarrow W_{k,j} + \alpha \times I_k \times \Delta_j$ 
      end
    end
  until network has converged
  return network

```

Figure 19: Pseudocode for backpropagation [13].

## 15.2 Helper Classes

Three helper classes are used to utilize the dataset and the network: FileReaderHelper, DataAndTrainingHelper, and Matrix class. These classes are explained in detail in this section.

### 15.2.1 FileReaderHelper

This class is responsible for getting the dataset from the text file and creating a 2D array of organized attributes. A bufferreader is used to read the dataset file and write the values into the array. There are 6 unknown attribute values in the whole dataset, marked with a ? mark in the dataset file, which was changed to the value 1 (the reasoning is explained in section 11.1: Data Processing).

```

public void backpropagation(Matrix target) {
    //CALCULATE THE ERROR AND GRADIENT
    error = Matrix.subtract(target, output);
    gradient = output.dsigmoid();
    gradient.multiply(error);
    gradient.multiply(learning_rate);

    //CALCULATE THE DELTA
    Matrix hidden_transpose = Matrix.transpose(hidden);
    Matrix who_delta = Matrix.multiply(gradient, hidden_transpose);

    //ADD THE GRADIENT AND DELTA TO WEIGHT AND BIAS OF HIDDEN/OUTPUT AND OUTPUT
    weights_hidden_output.add(who_delta);
    bias_output.add(gradient);

    //CALCULATE GRADIENT AND DELTA FOR INPUT/HIDDEN LAYER
    Matrix who_T = Matrix.transpose(weights_hidden_output);
    Matrix hidden_errors = Matrix.multiply(who_T, error);

    Matrix h_gradient = hidden.dsigmoid();
    h_gradient.multiply(hidden_errors);
    h_gradient.multiply(learning_rate);

    Matrix i_T = Matrix.transpose(input);
    Matrix wih_delta = Matrix.multiply(h_gradient, i_T);

    weights_input_hidden.add(wih_delta);
    bias_hidden.add(h_gradient);
}

```

Figure 20: Figure obtained from our Backpropagation Method.

### 15.2.2 DataAndTrainingHelper

In this class, with the help of FileReaderHelper, the 2D array of the dataset is sent to the Network class using the trainNetwork() method and the trained neural network is returned for further usage in the other classes.

In this class, the number of dataset values that are used to train the network is set to

525 which is 75% of all the values in the dataset. Moreover, the number of hidden layer's neurons and epochs are hardcoded based on trial and error; with hidden neurons being 6 and epochs being 350000 to ensure the maximum accuracy in prediction.

The rationale behind hardcoding the number of epochs lies in the presence of specific values within the dataset that are deemed untrainable (these values are seen in the confusion matrix in section 17.2). Consequently, the objective of achieving an error threshold for the entirety of the data became unattainable despite attempts to increase the number of epochs. Hence it was decided on a specific number to be hardcoded for epochs inside the code.

### 15.2.3 Matrix

Matrix class is created to convert the 2D array of the dataset into a matrix and methods such as add, multiply, transpose, subtract, and sigmoid function for the neural network. The matrix is implemented using a linked list data structure.

## 15.3 GraphicalUserInterface

The "View" class is critical to the implementation of the breast cancer prediction project's graphical user interface (GUI). It extends JavaFX's "Application" class, which serves as the JavaFX application's entry point. The "View" class initializes the "Model" and "Controller" objects within its constructor, laying the groundwork for the Model-View-Controller (MVC) architecture. This design pattern encourages concern separation and improves code maintainability. The "View" class configures the UI components in the "start" function by employing layout containers such as "GridPane" and "VBox." The numerous combo boxes, labels, and the output text field are constructed and configured, allowing users with an easy-to-use interface for entering data and evaluating prediction results. When the "predictButton" is clicked, an event handler is attached to it, which triggers the prediction process and interacts with the "Controller" object. The class specifies the style and appearance of the GUI with meticulous attention to detail, creating a visually appealing and user-friendly experience. Overall, the "View" class is an important part of the breast cancer prediction project since it allows for effective communication between the user and the underlying prediction model.

## 16 Calculations

In our project, we employed neural networks for breast cancer diagnosis. The calculations involved in this process are critical for the understanding and the effective operation of the system. This section provides a detailed explanation of these calculations, which are divided into four subsections: Output Calculation, Error Calculation, Updating weights



between hidden-output layer, and Updating weights between input-hidden layer.

## 16.1 Output calculation

Output calculation in a neural network refers to the generation of predictions or classifications. In our project, we focused on classifying cells as benign or malignant. Understanding output calculation requires an understanding of two steps: linear combination and activation function.

Initially, a linear combination of weights and input data is calculated. This calculation involves multiplying each input data by its corresponding weight and adding up the products. This process is a core element of most machine learning algorithms and forms the basis for how a model interprets input data [15].

After the linear combination, the resulting value is passed through an activation function. This function, which can be sigmoid, has the purpose of introducing non-linearity to the output of a neuron. This is necessary because most real-world data is non-linear, and we want neurons to learn from such data [17]. The activation function converts the linear combination to the final output value of the neuron.

## 16.2 Error calculation

Error calculation is an important metric in a neural network. It is a numerical indicator of network's success that represents the gap between the network's estimates and actual values. We used the Mean Squared Error (MSE) function for our project, which is commonly used in regression [20].

The MSE is calculated by taking the difference between each predicted and actual value, squaring this difference, and then averaging the squared differences over all instances. The squaring component of the MSE ensures larger errors are penalized more, which encourages the model to learn more precise predictions.

## 16.3 Updating weights between hidden-output layer

Adjusting the weights between the hidden and output layers is an important step in training our neural network. The error calculated from the output of the network using the back propagation algorithm is used to update these weights and it is aimed to reduce future errors [26].

The update is governed by the error gradient, which is a direction in the weight space that shows where the error increases. We adjust the weights to minimize the error by moving in the opposite direction.

The size of each step in the weight space is determined by the learning rate. This very important parameter ensures that the algorithm does not exceed the optimum point or move too slowly towards it [27]. Thus, the accuracy of our model increases in successive iterations.

## 16.4 Updating weights between input-hidden layer

In a neural network, the weights between the input and the hidden layers affect how the network interprets the input. Adjusting these weights is very important in learning key features for accurate predictions.

First, we calculate the error terms for the hidden layer, which are derived from the error terms that scale with the link weights of the output layer. This step effectively propagates the error back from the output to the hidden layer.

Then we adjust the weights between the input and hidden layers using these error terms. Each weight is updated according to a formula that includes the learning rate, latent error term, and input value. This process ensures that updates are proportionate and error-minimizing.

The weight update process is iterated over multiple periods, gradually improving the weights to optimize network performance [26].

## 17 Test of the program

A detailed testing strategy was used to assure the accuracy and dependability of this project. The testing procedure was divided into two parts: prediction testing and training testing.

### 17.1 Prediction Testing

A collection of input data representing breast cancer features was fed into the neural network using the "predict" method for prediction testing. To evaluate the network's

performance, the anticipated output was compared to the expected output. To evaluate the neural network's capacity to generalize and produce correct predictions, multiple test cases containing a varied range of input data were created. These test cases were created with the remaining 25% of the data remaining in the dataset. The code for this test is available in the "TestingTheNetwork" package in the project.

Upon conducting the aforementioned tests, it was observed that the neural network exhibited remarkable performance in predicting the correct outcomes for all test entries. Notably, predictions with accuracies below 70% were limited to a range of one to five instances out of a total of 174 values provided as input to the network. This outcome signifies the network's exceptional predictive capabilities, as it consistently achieved high accuracy in predicting the outcomes of the breast cancer attributes with a minimal margin of error (Figures 21 and 22).

```
90.16677626417426% accuracy on data number 692 That this tumor is Malignant!
96.89054906702724% accuracy on data number 693 That this tumor is Benign!
96.45351861121895% accuracy on data number 694 That this tumor is Benign!
96.45116376090294% accuracy on data number 695 That this tumor is Benign!
97.08388902535802% accuracy on data number 696 That this tumor is Benign!
90.1809954388504% accuracy on data number 697 That this tumor is Malignant!
89.08079947291596% accuracy on data number 698 That this tumor is Malignant!
89.10428167231285% accuracy on data number 699 That this tumor is Malignant!
Data 554 has less than 70% accuracy!
Data 556 has less than 70% accuracy!
Data 622 has less than 70% accuracy!
Data 658 has less than 70% accuracy!
Number of data with less than 70% accuracy: 4
Actual Benign: 136, Actual Malignant: 38
False Benign: 0, False Malignant: 2
```

*Figure 21: Figure obtained from our Prediction Testing.*

		<b>Actual Outcome</b>	
<b>Predicted Outcome</b>		<b>Benign</b>	<b>Malignant</b>
	<b>Benign</b>	136	0
	<b>Malignant</b>	2	36

*Figure 22: Confusion matrix for prediction testing.*

## 17.2 Training Testing

During training, the network's ability to learn and adjust its weights and biases was evaluated. To simulate the training data, synthetic input-output pairs were created with the 75% of the data. The network was trained with the "train" method, which adjusted the parameters of the network based on the calculated error and gradient. To ensure convergence and optimization, the training process was repeated for multiple epochs. The network's accuracy and convergence were assessed by comparing the predicted output to the ground truth values. Notably, predictions with accuracies below 70% were limited to a range of 19 to 28 instances out of a total of 525 values provided as input to the network (Figures 23 and 24).

```

Data 223 has less than 70% accuracy!
Data 233 has less than 70% accuracy!
Data 253 has less than 70% accuracy!
Data 260 has less than 70% accuracy!
Data 297 has less than 70% accuracy!
Data 299 has less than 70% accuracy!
Data 316 has less than 70% accuracy!
Data 320 has less than 70% accuracy!
Data 353 has less than 70% accuracy!
Data 357 has less than 70% accuracy!
Data 416 has less than 70% accuracy!
Data 435 has less than 70% accuracy!
Data 490 has less than 70% accuracy!
Data 495 has less than 70% accuracy!
Number of data with less than 70% accuracy: 20
Actual Benign: 322, Actual Malignant: 203
False Benign: 1, False Malignant: 16

```

*Figure 23: Figure obtained from our Training Testing.*

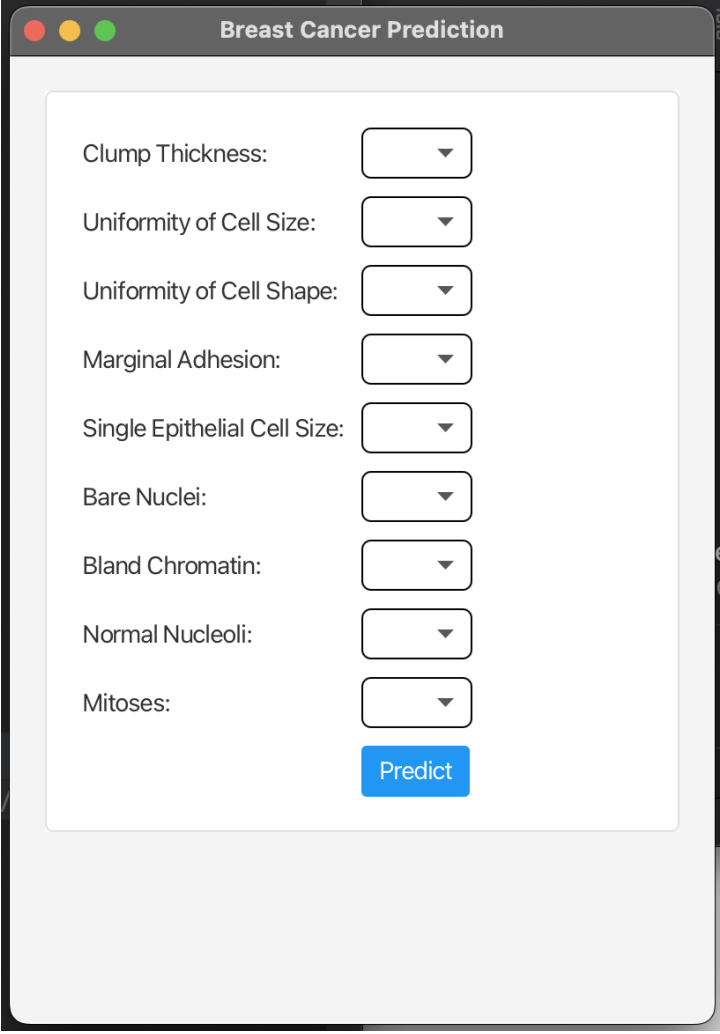
Predicted Outcome	Actual Outcome	
	Benign	Malignant
Benign	321	1
Malignant	16	187

*Figure 24: Confusion matrix for training testing.*

## 18 User guide

In this section, we will provide a brief guide to using our program through the GUI. User is able to choose the 9 features of the tumor and use the predict button to see the result of the prediction.

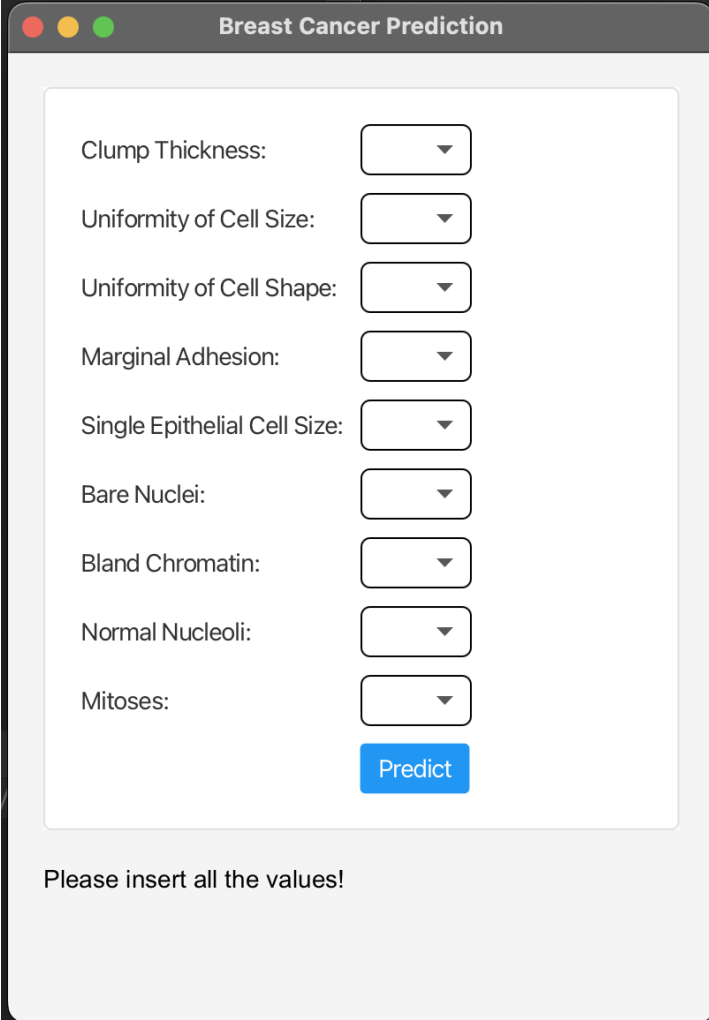
1. In the start of the application, the interface in figure 25 will be shown.



The image shows a graphical user interface (GUI) for a Breast Cancer Prediction application. The window has a title bar with the text "Breast Cancer Prediction" and three colored window control buttons (red, yellow, green) on the left. The main content area is a light gray rectangle containing a white rounded rectangle. Inside this white area, there are nine labels for features, each followed by a dropdown menu with a downward arrow. The features are: Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, and Mitoses. Below these dropdowns is a blue button with the text "Predict" in white.

*Figure 25: Figure obtained from our Initial Run.*

2. In the event that the user does not provide any input values within the designated boxes, an error prompt will be displayed, notifying the user of the omission (Figure 26).



The image shows a macOS-style window titled "Breast Cancer Prediction". Inside the window, there is a list of nine input fields, each with a label and a dropdown menu. The labels are: "Clump Thickness:", "Uniformity of Cell Size:", "Uniformity of Cell Shape:", "Marginal Adhesion:", "Single Epithelial Cell Size:", "Bare Nuclei:", "Bland Chromatin:", "Normal Nucleoli:", and "Mitoses:". Below these fields is a blue button labeled "Predict". At the bottom of the window, there is a text prompt: "Please insert all the values!".

Feature	Input Type
Clump Thickness:	Dropdown
Uniformity of Cell Size:	Dropdown
Uniformity of Cell Shape:	Dropdown
Marginal Adhesion:	Dropdown
Single Epithelial Cell Size:	Dropdown
Bare Nuclei:	Dropdown
Bland Chromatin:	Dropdown
Normal Nucleoli:	Dropdown
Mitoses:	Dropdown

Predict

Please insert all the values!

*Figure 26: Figure obtained from our Error Prompt.*

3. In cases where the tumor is determined to be either benign or malignant, the output prompt will present the accuracy percentage of the prediction along with the corresponding prediction outcome (Figure 27 and 28).

The image shows a software window titled "Breast Cancer Prediction". Inside the window, there is a list of nine features, each with a corresponding dropdown menu. The features and their selected values are: Clump Thickness (4), Uniformity of Cell Size (8), Uniformity of Cell Shape (8), Marginal Adhesion (5), Single Epithelial Cell Size (4), Bare Nuclei (5), Bland Chromatin (10), Normal Nucleoli (4), and Mitoses (1). Below these fields is a blue button labeled "Predict". At the bottom of the window, the prediction result is displayed: "Tumor is Malignant with 89.89% accuracy".

Feature	Value
Clump Thickness:	4
Uniformity of Cell Size:	8
Uniformity of Cell Shape:	8
Marginal Adhesion:	5
Single Epithelial Cell Size:	4
Bare Nuclei:	5
Bland Chromatin:	10
Normal Nucleoli:	4
Mitoses:	1

Predict

Tumor is Malignant with 89.89% accuracy

*Figure 27: Figure obtained from our Malignant Tumor Outcome.*



The image shows a software window titled "Breast Cancer Prediction". Inside the window, there is a list of nine features, each with a corresponding dropdown menu. All dropdown menus are currently set to the value "1". The features are: Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, and Mitoses. Below these input fields is a blue button labeled "Predict". At the bottom of the window, a text box displays the prediction result: "Tumor is Benign with 97.48% accuracy".

Feature	Value
Clump Thickness:	1
Uniformity of Cell Size:	1
Uniformity of Cell Shape:	1
Marginal Adhesion:	1
Single Epithelial Cell Size:	1
Bare Nuclei:	1
Bland Chromatin:	1
Normal Nucleoli:	1
Mitoses:	1

Predict

Tumor is Benign with 97.48% accuracy

Figure 28: Figure obtained from our Benign Tumor Outcome.

4. If the neural network fails to provide a definitive prediction for a given case, a warning prompt will be displayed to the user. This prompt will include the network's predicted outcome, acknowledging its inconclusive nature while still presenting the available information (Figure 29).

Breast Cancer Prediction

Clump Thickness: 1

Uniformity of Cell Size: 10

Uniformity of Cell Shape: 1

Marginal Adhesion: 1

Single Epithelial Cell Size: 6

Bare Nuclei: 2

Bland Chromatin: 2

Normal Nucleoli: 1

Mitoses: 2

Predict

The network cannot predict the result of this tumor.  
The predicted output is: 0.46

Figure 29: Inconclusive Outcome

## 19 Ethical Dilemma and Risks

Breast cancer prediction is an important area of medical research, and the use of neural networks has shown promising results in improving diagnostic accuracy. To train a breast cancer prediction neural network, various tumor features such as mitosis, cell shape, size, and so on are used. Based on these input features, the neural network analyzes and classifies breast cancer cases using advanced machine learning techniques. Healthcare professionals may be able to improve early detection and provide timely interventions for patients at risk of developing breast cancer by leveraging the power of neural networks. The use of a breast cancer prediction neural network has the potential to significantly improve patient outcomes and reduce the burden of this devastating disease.

However, the use of neural networks in healthcare raises ethical concerns as well as inherent risks that must be carefully considered. One major concern is the possibility of algorithmic bias, in which the neural network's predictions differ based on different factors. If not addressed properly, these biases have the potential to exacerbate existing healthcare disparities and lead to unequal treatment or access to care. Furthermore, the use of neural networks raises concerns about data privacy and security. Patient data, which is critical for network training, must be handled with extreme caution to ensure confidentiality and prevent unauthorized access or misuse.

Another ethical consideration is the predictability and transparency of the neural network's predictions. While neural networks can achieve remarkable accuracy, they frequently operate as black boxes, making understanding the underlying decision-making process difficult. Because healthcare providers may struggle to explain and justify the network's predictions to patients or regulatory bodies, this lack of interpretability raises concerns about accountability. Furthermore, there is a risk of overreliance on the neural network, which could lead to a disregard for clinical expertise or the human aspect of patient care.

In conclusion, while the development of this project holds great promise for improving diagnosis and patient outcomes, it is critical to recognize and address the ethical issues it raises. Protecting against biases, ensuring data privacy, promoting transparency, and maintaining a balanced integration of clinical expertise are all important factors in harnessing the potential of neural networks while minimizing associated risks in breast cancer prediction and other healthcare applications.

## 20 Conclusion

The group managed to program an artificial neural network with an input layer, a hidden layer and an output layer. This network utilizes the forward propagation technique to transmit inputs through the layers and employs backpropagation algorithms for network training. The development of these forward propagation and backpropagation algorithms drew inspiration from the foundational work of Russel and Norvig [13], which served as a fundamental reference throughout the duration of the project. Moreover, the programming approach undertaken in this study was characterized by an iterative process. Initially, a basic prototype was programmed, and subsequently, the program was refined and expanded upon to enhance its functionality and effectiveness.

One of the complexities encountered during our design process pertained to effectively managing data storage and structuring within the network to facilitate optimal accessibility in the `feedForward()` and `backpropagation()` methods. In order to facilitate efficient and convenient access to the network's essential data, a `Matrix` class was developed as a resolution to this problem. This class serves as a container, holding all the required information such as the values and weights of the inputs, outputs, hidden neurons.

The study has applied MLP neural network model on a breast cancer dataset to assess its accuracy in correctly classifying breast cancer cases. To achieve this goal the error was minimized using a gradient descent algorithm. The model has high prediction accuracy for both training set and testing set. It can be seen from the analysis of the obtained results that the accuracy of the model for the test data set is higher producing only 2 false malignant predictions and 4 predictions with accuracy less than 70%. Meanwhile predictions of the training set had 16 false malignant, 1 false benign and 20 predictions with accuracy lower than 70%. One contributing factor to the inaccurate predictions can be attributed to the identification of specific values within the dataset as untrainable. These errors serve as a catalyst for engaging in a discussion regarding the ethical quandaries and potential hazards associated with the utilization of this project for diagnosing breast cancer in individuals.

In conclusion, the study not only evaluated the accuracy of the MLP neural network model in classifying breast cancer cases but also demonstrated technical proficiency in developing and refining the underlying algorithms and data structures. While the model performed well on both the training and testing sets, the presence of inaccuracies, particularly false malignant predictions and lower accuracy scores, calls for further investigation. Identifying specific values in the dataset as untrainable was discovered to be a factor in these inaccuracies. As a result, the study sparks a critical debate about the ethical implications and potential risks of using this project to diagnose breast cancer

in individuals. We can strive for responsible and ethical implementation of predictive technologies in healthcare by addressing these challenges and engaging in comprehensive dialogue.

## 21 Bibliography

- [1] *World Health Organization - Breast cancer*. Retrived May, 18, 2023. URL: <https://www.who.int/news-room/fact-sheets/detail/breast-cancer>.
- [2] Andre Esteva et al. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *Nature* 542.7639 (Feb. 2017), pp. 115–118. ISSN: 1476-4687. DOI: 10.1038/nature21056. URL: <https://doi.org/10.1038/nature21056>.
- [3] William H. Wolberg, W.Nick Street, and O.L. Mangasarian. “Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates”. In: *Cancer Letters* 77 (2-3 Mar. 1994), pp. 163–171. ISSN: 03043835. DOI: 10.1016/0304-3835(94)90099-X.
- [4] Marina Sokolova and Guy Lapalme. “A systematic analysis of performance measures for classification tasks”. In: *Information Processing Management* 45 (4 July 2009), pp. 427–437. ISSN: 03064573. DOI: 10.1016/j.ipm.2009.03.002.
- [5] Travers Ching et al. “Opportunities and obstacles for deep learning in biology and medicine”. In: *Journal of The Royal Society Interface* 15 (141 Apr. 2018), p. 20170387. ISSN: 1742-5689. DOI: 10.1098/rsif.2017.0387.
- [6] *UCI, Machine Learning Repository: Breast Cancer Wisconsin (original) data set*. Retrived May, 18, 2023. URL: <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+280original29>.
- [7] *What is Artificial Intelligence (AI)?* Retrived March, 7, 2023. URL: <https://www.ibm.com/topics/artificial-intelligence>.
- [8] *What is machine learning?* Retrived March, 7, 2023. URL: <https://www.ibm.com/topics/machine-learning>.
- [9] *Artificial Neural Network Systems*. Retrived May, 18, 2023. URL: [https://www.researchgate.net/publication/350486076\\_Artificial\\_Neural\\_Network\\_Systems](https://www.researchgate.net/publication/350486076_Artificial_Neural_Network_Systems).
- [10] Su-Hyun Han et al. “Artificial Neural Network: Understanding the Basic Concepts without Mathematics”. In: *Dementia and Neurocognitive Disorders* 17 (3 2018), p. 83. ISSN: 1738-1495. DOI: 10.12779/DND.2018.17.3.83. URL: [/pmc/articles/PMC6428006/](https://pmc/articles/PMC6428006/) <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6428006/?report=abstract> <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6428006/>.
- [11] Igor Livshin. *Artificial Neural Networks with Java*. Apress, 2019, pp. 1–3. DOI: 10.1007/978-1-4842-4421-0.
- [12] S. Abirami and P. Chitra. “Energy-efficient edge based real-time healthcare support system”. In: *Advances in Computers* 117 (1 Jan. 2020), pp. 339–368. ISSN: 0065-2458. DOI: 10.1016/BS.ADCOM.2019.09.007.

- [13] Stuart J. Russell ; Peter Norvig. *Artificial intelligence : a modern approach* /. Prentice Hall, 1995, pp. 106–110. ISBN: 0131038052.
- [14] *Programming Language - Techterms (September 23, 2011)*. Retrived March, 14, 2023. URL: [https://techterms.com/definition/programming\\\_language](https://techterms.com/definition/programming\_language).
- [15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [16] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.
- [17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521 (7553 2015), pp. 436–444.
- [18] Lei Zhang, Shuai Wang, and Bing Liu. “Deep learning for sentiment analysis: A survey”. In: *WIREs Data Mining and Knowledge Discovery* 8.4 (2018), e1253. DOI: <https://doi.org/10.1002/widm.1253>. eprint: <https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1253>. URL: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1253>.
- [19] 2020. URL: <https://deepai.org/machine-learning-glossary-and-terms/backpropagation>.
- [20] Michael A. Nielsen. *Neural Networks and Deep Learning*. 2015. URL: <http://neuralnetworksanddeeplearning.com>.
- [21] Haider Khalaf Jabbar and Rafiqul Zaman Khan. “Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study)”. In: Research Publishing Services, 2014, pp. 163–172. ISBN: 978-981-09-5247-1. DOI: 10.3850/978-981-09-5247-1\_017.
- [22] Sinan Ozdemir. *Principles of Data Science*. Packt, 2016, pp. 298–301.
- [23] 2023. URL: <https://wesrom.com/insights/engineering-insights/is-java-still-relevant-in-2023-pros-and-cons-of-using-java/>.
- [24] Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser. *Data Structures and algorithms in Java, 6th edition*. 2014. URL: <https://www.wiley.com/en-us/Data+Structures+and+Algorithms+in+Java+%2C+6th+Edition-p-9781118771334?fbclid=IwAR0nl75l36afhsgx4uwfStxWZcfgJv1VSJ91W2u8AaKea1EFDD2bMNkK2pk>.
- [25] Christopher M. Bishop. *Pattern recognition and machine learning*. URL: <https://link.springer.com/book/9780387310732>.
- [26] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (6088 Oct. 1986), pp. 533–536. ISSN: 0028-0836. DOI: 10.1038/323533a0.

- [27] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: (Sept. 2016).