



Intelligent Systems

# Breast Cancer Detection

Dr. Abdel-Rahman Hedar

Team members:  
Abdelrahman Maher Makhoulf  
Abdullah Ahmed Abdelbaset  
Demean Rafiq  
Yasmeen Matter

## Table of Contents

### Table of Contents

<b>I. Introduction.....</b>	<b>2</b>
<b>1.1 Problem.....</b>	<b>2</b>
<b>1.2 Purpose.....</b>	<b>2</b>
<b>1.3 Solution.....</b>	<b>2</b>
<b>II. Methodology .....</b>	<b>4</b>
<b>2.1 Use case diagram.....</b>	<b>4</b>
<b>2.2 System scenario .....</b>	<b>5</b>
<b>III. Experimental Simulation.....</b>	<b>5</b>
<b>3.1 Importing libraries.....</b>	<b>5</b>
<b>3.2 Prepare the data.....</b>	<b>6</b>
<b>3.3 Building ANN model.....</b>	<b>8</b>
<b>3.4 Testing.....</b>	<b>10</b>
<b>4. Results and Discussion.....</b>	<b>10</b>
<b>5.Conclusions.....</b>	<b>10</b>
<b>6.References.....</b>	<b>11</b>
<b>7.Project Source Codes.....</b>	<b>12</b>

# 1.Introduction

Breast cancer is the most common cancer amongst women in the world. It accounts for 25% of all cancer cases and affected over 2.1 million people in 2015 alone. It starts when cells in the breast begin to grow out of control. These cells usually form tumors that can be seen via X-ray or felt as lumps in the breast area.

## 1.1 Project problem

Breast cancer is a complex disease. There are several subtypes of breast cancer and may options for treatment. While many people may have similar diagnoses or are prescribed similar treatments, no two people's experiences are exactly the same.

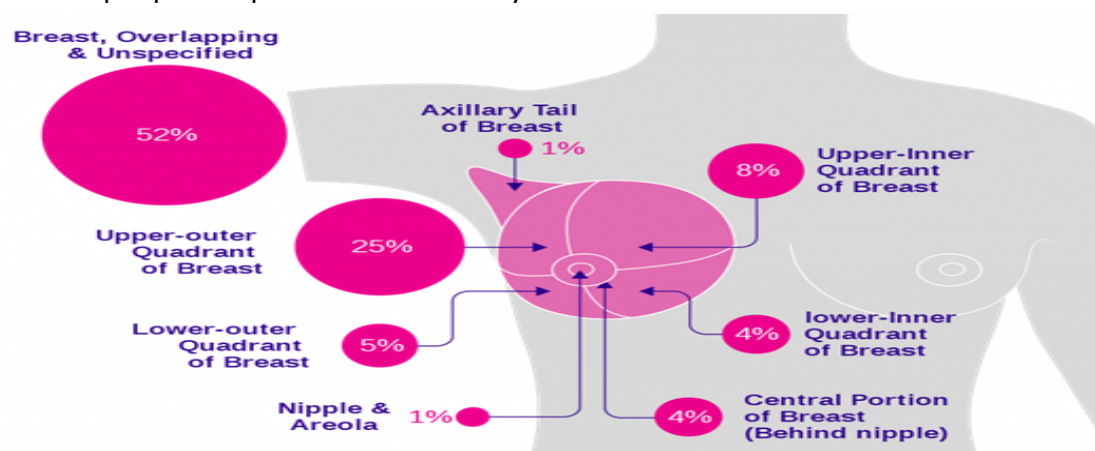


Figure 1: The position of Breast Cancer and its affect

## 1.2 Purpose

In this project we try to help doctors in the process of diagnosing the breast cancer and save the doctors' time and facilitate the process of diagnosis and decision-making as quickly as possible, and because of the large amount of information that must be in the doctors' mind, it is difficult for them to make the right decision quickly. By building an intelligent model, it will be easy to know the tumor is malignant and benign!

## 1.3 Solution

Computational methods and techniques such as machine learning are often used in medical applications to overcome some of these challenges and to provide dependable information for diagnoses. Machine learning is the generic name for computer algorithms that model a problem according to the data of that problem.

Machine learning can improve the estimation of results with classification techniques that are based on the organized data in predetermined categories. Machine-learning classification algorithms are focused on the most economical use of the maximum possible amount of data to achieve the highest success and performance. The architecture of machine-learning methods and the statistical

distributions of the observed data typically lead to the training of classification models realized by minimizing a loss of function. The classification algorithm is subsequently trained on the dataset to find the connection between input variables and the output. The training is considered complete when the algorithm reaches an adequate level of accuracy, and the algorithm is next applied to a new dataset.

Deep Learning models can be used for a variety of complex tasks one of these models is Artificial Neural Networks (ANN) for Regression and classification, it is an information processing paradigm that is inspired by the way the biological nervous system such as brain process information. It is composed of large number of highly interconnected processing elements(neurons) working in unison to solve a specific problem.

Artificial neuron models are at their core simplified models based on biological neurons. This allows them to capture the essence of how a biological neuron function. We usually refer to these artificial neurons as “perceptron”. So now let’s look at what a perceptron looks like.

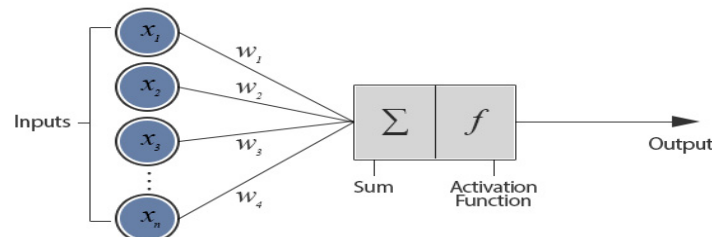


Figure 2: Architecture of a simple perceptron.

As shown in the diagram above a typical perceptron will have many inputs and these inputs are all individually weighted. The perceptron weights can either amplify or de-amplify the original input signal. For example, if the input is 1 and the input’s weight is 0.2 the input will be decreased to 0.2. These weighted signals are then added together and passed into the activation function. The activation function is used to convert the input into a more useful output. There are many different types of activation function but one of the simplest would be step function. A step function will typically output a 1 if the input is higher than a certain threshold, otherwise its output will be 0.

## 2. Methodology

The methodology of the system involves the medical data of the tumor, nearly 30 properties depending on our data, and then introducing these features to intelligent learning model, then the smart learning model determine whether the tumor is malignant and benign.

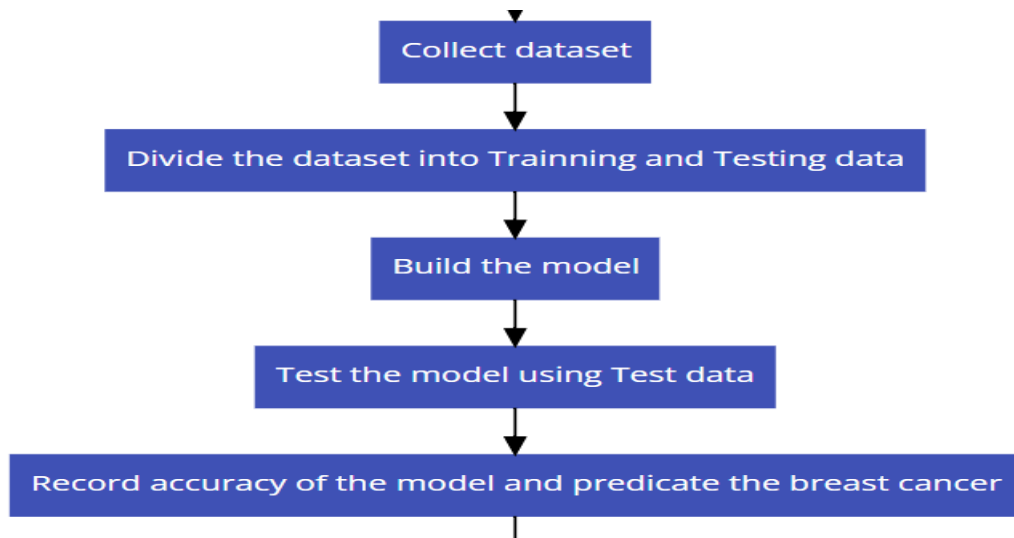


Figure 3: Process Flowchart

First, get our dataset that contains a lot of cases, then, divide the dataset into Training and Testing data, then, use model from “tensorflow” library that built using Gradient Descent Algorithm and Back Propagation Algorithm. The next step is testing our model using test data. After all these steps we get accuracy of the model and predicate the breast cancer.

### 2.1 Use case diagram:

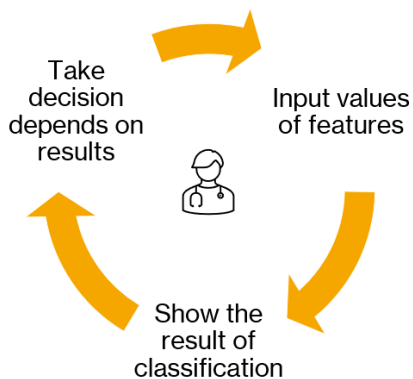


Figure 4: Use case diagram

## 2.3 System scenario

At first doctor must enter the medical features data of patient then press enter to show results. Then the system will take these features then enter this features value into intelligent model to classify the tumor as malignant and benign.

## 3. Experimental Simulation

We used Python programming language because it has ‘TensorFlow’ library, TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks, TensorFlow was developed by the Google Brain team for internal Google use in research and production. We used the IDLE (short for Integrated Development and Learning Environment), it is an integrated development environment for python written in python, It is packaged as an optional part of the Python packaging.

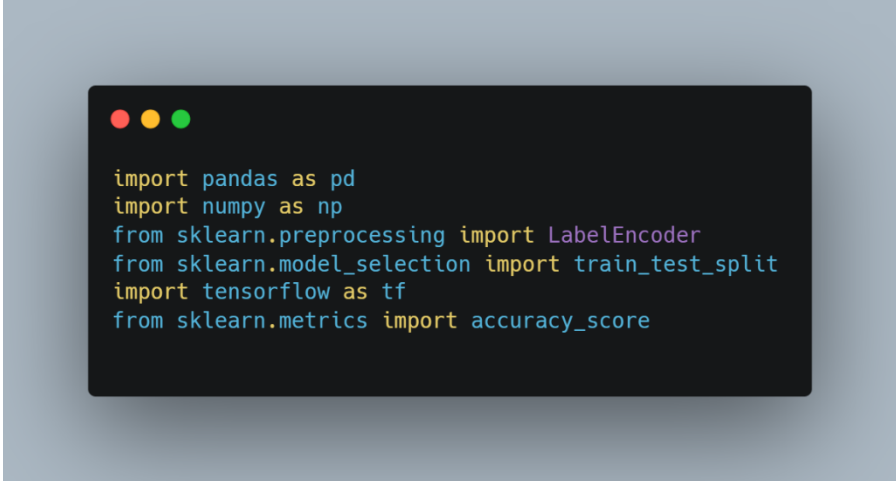
**Note:** We use this site [‘Carbon | Create and share beautiful images of your source code’](#) to make our code in beautiful images to make it easy to read. We use [‘https:// ideone.com’](https://ideone.com) to store our source code and get a link for it and [“code2flow - online interactive code to flowchart converter”](#) To create our flowcharts.

So, in case of solving this problem we will divide it into main 3 steps:

1. Import libraries.
2. Prepare data.
3. Building the ANN model.

Here we will explain each part of these processes:

### 3.1 Importing libraries:

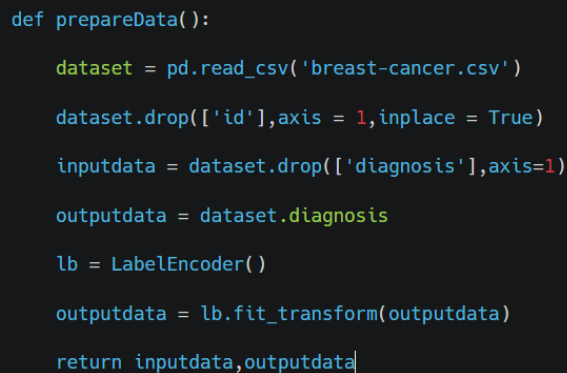


```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import tensorflow as tf
from sklearn.metrics import accuracy_score
```

Figure 5: Import libraries section

We import 'pandas' library to open the dataset that we will use to train and test the model, 'numpy' to prepare data into array that will be pass to the training function and test, 'tensorflow' to create our ANN model with a number of neurons we determined, 'LabelEncoder' that convert values from 'M' for malignant to '1' and 'B' for benign to '0', and 'accuracy\_score' from 'sklearn.metrics' to compare the results of test data and its correct result to calculate the accuracy of the model.

### 3.2 Prepare the data



```
def prepareData():
    dataset = pd.read_csv('breast-cancer.csv')
    dataset.drop(['id'],axis = 1,inplace = True)
    inputdata = dataset.drop(['diagnosis'],axis=1)
    outputdata = dataset.diagnosis
    lb = LabelEncoder()
    outputdata = lb.fit_transform(outputdata)
    return inputdata,outputdata
```

Figure 6: Prepare data function

This a function that considers the first step of our program, our data set contains samples that have been grouped into M (Breast cancer positive) and B (Breast cancer negative).

The file contains 'Id', 'Diagnosis' columns, and 30 features:

1. Radius mean
2. Texture mean
3. Perimeter mean
4. Area mean
5. Smoothness mean
6. Compactness mean
7. Concavity mean
8. Concave points mean
9. Symmetry mean
10. Fractal dimension mean
11. Radius se
12. Texture se
13. Perimeter se
14. Area se
15. Smoothness se
16. Compactness se
17. Concavity se
18. Concave points se
19. Symmetry se
20. Fractal dimension se
21. Radius worst
22. Texture worst
23. Perimeter worst
24. Area worst
25. Smoothness worst
26. Compactness worst
27. Concavity worst
28. Concave points worst
29. Symmetry worst
30. Fractal dimension worst

Our data set has 63% 'B' and 37% 'M':

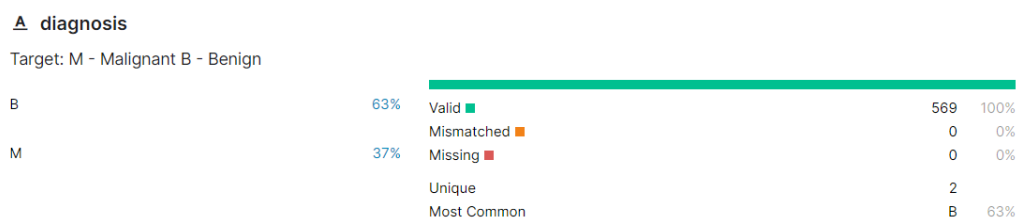


Figure 7: Diagnosis M and B percentage for our data set



We need to get the data to divide it into training and testing set.

```
dataset = pd.read_csv('breast-cancer.csv')
```

Figure 8: Read the data set from csv file

```
dataset.drop(['id'],axis = 1,inplace = True)  
inputdata = dataset.drop(['diagnosis'],axis=1)  
outputdata = dataset.diagnosis
```

Figure 9: Drop id column from data set file and drop diagnosis from data set to get input data

Then we use LabelEncoder to make the output data fit with 1s and 0s:

```
lb = LabelEncoder()  
outputdata = lb.fit_transform(outputdata)
```

Figure 10: labelEncoder transform

### 3.3 Building ANN model

This function creates an ANN model with 9 neurons as input layer, 9 neurons as hidden layer, and 1 neuron for output layer. After creating model train it on our dataset that we pass it as a parameter.

```
def buildTheModel(trainingData,trainingDtaLabels):

    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(9,activation = 'relu',input_dim = 30),
        tf.keras.layers.Dense(9,activation = 'relu'),
        tf.keras.layers.Dense(1,activation = 'sigmoid')])

    model.compile(optimizer = 'adam',loss='binary_crossentropy',metrics=['accuracy'])

    model.fit(trainingData,trainingDataLabels,batch_size = 100,epochs = 100)

    return model
```

Figure 11: Build model and training function

We build a function to test the accuracy of our model that compare the results of our model and the real results and returns the accuracy of the model using accuracy\_score function (figure 12).

```
def accuracy(testData,testDataLabels):

    predictedLabels = model.predict(testData)

    predictedLabels = (predictedLabels>0.5)

    accuracy = accuracy_score(testDataLabels,predictedLabels)

    return accuracy
```

Figure 12: Accuracy function

Because of our program is console so after call prepare function and model build, we build a cool function to communicate with user and make him understood the program faster. The application works infinite, take input and check if it has any exceptions then output “invalid input” and continue. If the input is correct then pass it to function model.predict (input) then \*100 to get the value of percentage of the disease with this person .

```

def runTheApplication():
    flag = True
    again = ""
    start = "Enter the data of the person to predict separated by \",\" : \n"
    while(flag):
        try:
            evaluationdata = list(map(float, input(again + start).strip().split(',')))
            evaluationdata = np.array(evaluationdata).reshape(1,30)
            predictedLabel = model.predict(evaluationdata)
            print('\nPercentage of being Malignant is : ',predictedLabel[0]*100,'% \n')
            predictedLabel = (predictedLabel>0.5)
            if predictedLabel == 1:
                print('The diagnosis is \"M\" for Malignant \n')
            else:
                print('The diagnosis is \"B\" for Benign \n')
        except:
            print('Invalid input \n')
            again = "again, "

```

Figure 13: Run application function

### 3.4 test cases

For testing our application we implemented the “RunTheApplication” method shown in figure 13- which also used form the user- that takes its input as a 30 parameters separated by commas – the data about the patient to predict her tumor – then we preprocessed these data to match out model’s input, then it shows the result as percentage and a classification label as shown in figure 14.

## 4. Results and Discussion

The result of our program consists of two lines, the percentage of the tumor to be malignant, and the result – malignant or benign- according to this percentage.

As an analysis of the output and results we used a known-labels data for testing to have the chance of comparing our model’s results with the real results, and the accuracy of the model is about 95%, for example, the input we used in figure 14 is a 30 attribute row that represents the needed data

for one person's tumor to be predicted by our model, so the model predicted the percentage of being malignant "which is 2.19%" and then classifies the tumor to one of the two classes "M for malignant, B for benign", in our case it is B, which is obviously observed from the percentage.

This result matches the row label in the dataset of the test data.

Figure 14: testing output

```
again, Enter the data of the person to predict separated by "," :
0
Invalid input

again, Enter the data of the person to predict separated by "," :
9.504,12.44,60.34,273.9,0.1024,0.06492,0.02956,0.02076,0.1815,0.06905,0.2773,0.9768,1.909,15.7,0.009606,0.01432,0.01985,0.01421,0.02027,0.002968,10.23,15.66,65.13,314.9,0.1324,0.1148,0.08867,0.06227,0.245,0.07773

1/1 [ ] ~ ETA: 0s
1/1 [ ] ~ 0s 78ms/step

Percentage of being Malignant is : [2.1909206] %

The diagnosis is "B" for Benign

again, Enter the data of the person to predict separated by "," :
|
```

## 5. Conclusions

We are planning to integrate a smarter UI that reads the data from an image using an ANN, we also need more data for training our model to be more accurate and more general by applying a votes across the world we collect the data. We should learn more about ANN to increase our model's accuracy may be by applying a GA for finding the weights values that scores minimum error between the output and the model's output.

We tried to cover all the main points that describe our problem (Breast Cancer Detection), methodology of our algorithm to predict the tumor is malignant and benign, describe the programming languages and environments used in our project, describe our source code, test, and discusses the output of program.

In conclusion we would like to say that there are a lot of things that we have not identified which ones we will use in the project as how to build a neural network from scratch, we didn't create a graphical user interface because it is out of our purpose now. In the future we will create a GUI for our project to increase facilitation using our project results.

.....

## 6.References

Intelligent Systems A Modern Approach Book, Crina Grosan, Ajith Abraham, chapter12: Artificial Neural Networks.

- [Challenges to the early diagnosis and treatment of breast cancer in developing countries - PMC \(nih.gov\)](#)
- [Better breast cancer detection | IEEE Journals & Magazine | IEEE Xplore](#)
- [Automating Breast Cancer Detection with Deep Learning | by Sheng Weng | Insight \(insightdatascience.com\)](#)
- [Frontiers | Association of Breast Cancer Screening Behaviors With Stage at Breast Cancer Diagnosis and Potential for Additive Multi-Cancer Detection via Liquid Biopsy Screening: A Claims-Based Study | Oncology \(frontiersin.org\)](#)
- [Automating Breast Cancer Detection with Deep Learning | by Sheng Weng | Insight \(insightdatascience.com\)](#)
- [Breast Cancer Explained - Dr Susan Love Foundation for Breast Cancer Research \(drsusanloveresearch.org\)](#)
- Implementation help resources:
  - Data set: [Breast Cancer Dataset | Kaggle](#)
  - [Implementing Artificial Neural Network training process in Python - GeeksforGeeks](#)
  - [How To Create a Neural Network In Python – With And Without Keras - ActiveState](#)
  - [Transfer Learning | Pretrained Models in Deep Learning \(analyticsvidhya.com\)](#)
  - [Training, saving and loading Artificial Neural Networks in Keras \(opengenius.org\)](#)
  - [scikit learn - How to save and load my neural network model after training along with weights in python? - Stack Overflow](#)
  - [Introduction to Artificial Neural Networks - KDnuggets](#)
  - [Introduction to Artificial Neural Networks - Part 1 \(theprojectspot.com\)](#)

## 7.Project Source Codes.

[CnVT4A - Online Python Interpreter & Debugging Tool - Ideone.com](#)

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import tensorflow as tf
from sklearn.metrics import accuracy_score

# Getting the dataset and applying a preprocessing to match our purpose
def prepareData():

    dataset = pd.read_csv('breast-cancer.csv')

    dataset.drop(['id'],axis = 1,inplace = True)

    inputdata = dataset.drop(['diagnosis'],axis=1)

    outputdata = dataset.diagnosis

    lb = LabelEncoder()

    outputdata = lb.fit_transform(outputdata)

    return inputdata,outputdata

# Building the ANN model using the dataset
def buildTheModel(trainingData,trainingDtaLabels):

    model = tf.keras.models.Sequential([
        tf.keras.layers.Dense(9,activation = 'relu',input_dim = 30),
        tf.keras.layers.Dense(9,activation = 'relu'),
        tf.keras.layers.Dense(1,activation = 'sigmoid')])

    model.compile(optimizer = 'adam',loss='binary_crossentropy',metrics=['accuracy'])

    model.fit(trainingData,trainingDataLabels,batch_size = 100,epochs = 100)

    return model

# Calculating the accuracy of our model
def accuracy(testData,testDataLabels):

    predatedLabels = model.predict(testData)

    predatedLabels = (predatedLabels>0.5)

    accuracy = accuracy_score(testDataLabels,predatedLabels)

    return accuracy

# Using the neural network after completing its training and testing
```

Figure 15: source code part 1

```
def runTheApplication():  
    flag = True  
    again = ""  
    while(flag):  
        try:  
            evaluationdata = list(map(float, input(again + "Enter the data of the person to  
predict separated by \",\" : \n").strip().split(',')))  
            evaluationdata = np.array(evaluationdata).reshape(1,30)  
            predectedLabel = model.predict(evaluationdata)  
            print('\nPercentage of beinng Malignant is : ',predectedLabel[0]*100,'% \n')  
            predectedLabel = (predectedLabel>0.5)  
            if predectedLabel == 1:  
                print('The diagnosis is \"M\" for Malignant \n')  
            else:  
                print('The diagnosis is \"B\" for Benign \n')  
        except:  
            print('Invalid input \n')  
            again = "again, "  
  
# Split the data labels  
inputData,outputData = prepareData()  
  
# Divide the data into training and testing data  
trainingData,testData,trainingDataLabels,testDataLabels =  
train_test_split(inputData,outputData,test_size = 0.2,random_state = 40)  
  
# Building the ANN  
model = buildTheModel(trainingData,trainingDataLabels)  
  
# Calculating the accuracy over the test data  
accuracy = accuracy(testData,testDataLabels)  
print('The accuracy is : ',accuracy,'\n')  
runTheApplication()
```

Figure 16: source code part 2