GSJ: Volume 10, Issue 6, June 2022, Online: ISSN 2320-9186
www.globalscientificjournal.com

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING & TECHNOLOGY

MINI PROJECT REPORT

SUBJECT TITLE: Minor Project I

SUBJECT CODE: 15CS375L

AICDM

"An Intelligent Cancer severity detection model."

By

Paras Jain-RA1711003010088

Under the supervision of

Ms.Aswathy K Cherian (102217)

Assistant Professor (O.G)

Department of Computer Science and Engineering, SRM Institute of Science and Technology, Kattankulathur Campus.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

BONAFIDE CERTIFCATE

Certified that this project report on "**Cancer severity detection**" is the bonafide work of " **Paras Jain (RA1711003010088)**" who carried out the project work as part of their course 15CS375L - MINOR PROJECT I.

SIGNATURE **SIGNATURE**

Course Instructor

Ms.Aswathy K Cherian

(102217)

Head Of Department

Computer Science and Engineering

INTERNAL EXAMINER

Acknowledgement

In performing our minor project, we had to take the help and guidelines of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much immense pleasure and satisfaction.

We would like to show our gratitude towards Mrs D .VIJI MA'AM, Mentor, SRM Institute of Science and Technology for giving us a good guideline for the minor project throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment.

Many people, especially our classmates and other team members, have made valuable comment suggestions which gave us an inspiration to improve our project. We thank all the people for their help directly and indirectly to complete our assignment.

Table of Contents

Abstract

Judgement is a core skill in the medical industry that relies upon different levels of data combined together. Cancer offers a uniquely complex context, thus increasing the need for high skill decision making involving the condition of patient, their ability to take the treatment, evolution of the disease and finally the response to the treatment. Radiographic assessment of the disease mostly relies upon visual tests, interpretation of which can be handled very efficiently using Artificial Intelligence. The interpretation of the cancer imaging can be revolutionized by the help of Machine Learning(Artificial Intelligence). This minor project aims to implement Artificial intelligence on cancer, specifically breast tumours through medical imaging and data analysis using the core principles of Machine Learning. The desire to improve the efficiency of medical science continues to drive innovations into practice, including Artificial Intelligence. With the ever increasing demand for health care and the huge amount of data that it produces on a daily basis, the optimization of clinical work has become increasingly critical. Artificial intelligence excels at recognizing complex patterns in images and data, thus offers the opportunity to transform image interpretation from a purely qualitative and subjective task to one that is quantifiable. In addition, Artificial Intelligence may quantify information from images that is not detectable by humans or the level of data analysis that is not possible by a human and thereby greatly enhances clinical decision judgement.

We have made use of PYCHARM, a machine learning software library and worked with a very efficient compiler, which is, PYCHARM to create a simple model to the get the rank of GATE aspirants using the parameters as the input.

Introduction

Breast cancer is one of the most common cancers among women worldwide, representing the majority of new cancer cases and cancer-related deaths according to global statistics, making it a significant public health problem in today’s society.

The early diagnosis of BC can improve the prognosis and chance of survival significantly, as it can promote timely clinical treatment to patients. Further accurate classification of benign tumours can prevent patients undergoing unnecessary treatments. Thus, the correct diagnosis of BC and classification of patients into malignant or benign groups is the subject of much research. Because of its unique advantages in critical features detection from complex BC datasets, machine learning (ML) is widely recognized as the methodology of choice in BC pattern classification and forecast modelling.

Classification and data mining methods are an effective way to classify data. Especially in medical field, where those methods are widely used in diagnosis and analysis to make decisions.

Screening guidelines recommended:

Mammography . The most important screening test for breast cancer is the mammogram. A mammogram is an X-ray of the breast. It can detect breast cancer up to two years before the tumour can be felt by you or your doctor.

Women age 40–45 or older who are at average risk of breast cancer should have a mammogram once a year. The chance of getting breast cancer increases as women age. Nearly 80 percent of breast cancers are found in women over the age of 50.
A woman has a higher risk of breast cancer if her mother, sister or daughter had breast cancer, especially at a young age (before 40). Having other relatives with breast cancer may also raise the risk.

Genetic factors. Women with certain genetic mutations, including changes to the BRCA1 and BRCA2 genes, are at higher risk of developing breast cancer during their lifetime. Other gene changes may raise breast cancer risk as well.

Women at high risk should have yearly mammograms along with an MRI starting at age 30.

In this model, we have used attributes of the breast as input to detect the level of severeness of breast cancer so that proper further steps can be taken for any further course of action.

Data Preparation

Attribute Information:

Ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness ($perimeter^2$ / area — 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension (“coastline approximation” — 1)

The mean, standard error and “worst” or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

This analysis aims to observe which features are most helpful in predicting malignant (Severe) or benign (Not so harmful) cancer and to see general trends that may aid us in model selection and hyper parameter selection. The goal is to classify whether the breast cancer is benign or malignant. To achieve this we have used machine learning classification methods to fit a function that can predict the discrete class of new input.

For this, we have to make use of deep neural networking. A deep neural network is simply an artificial neural network with more layers. The theory behind why this should be more effective is that by adding more layers to the network, it’s able to learn more complicated patterns in the data and perform better analyses.

Input data

Any neural network requires input data, this is what’s used to perform predictions, train the network, and test its accuracy. For the input of this network, we decided to use the Wisconsin Diagnostic Breast Cancer dataset. This is a collection of data collected from biopsies and their given diagnoses determined from that data. In total, the data set contains 569 examples, 357 benign and 212 malignant. In addition, each example contains 30 measurements calculated from each biopsy representing 10 different features of a cell nucleus.

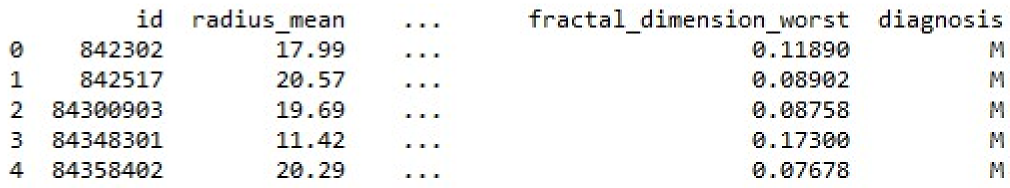
Data Sets

The dataset has already been divided into their respective training sets and test data sets, so, we had to upload them.We divided the data in test and training set to prevent any form of overfitting which is harmful for the model.

(code for uploading datasets)

```
import pandas as pd
file = files.upload()
X_train = pd.read_csv("xtrain.csv", header=None)
Y_train = pd.read_csv("ytrain.csv", header=None)
X_test = pd.read_csv("xtest.csv", header=None)
Y_test = pd.read_csv("ytest.csv", header=None)
```

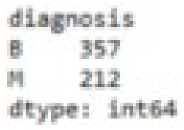
We can examine the data set using the pandas' **head()** method.
dataset.head()



```
         id  radius_mean   ...     fractal_dimension_worst  diagnosis
0    842302        17.99   ...                     0.11890          M
1    842517        20.57   ...                     0.08902          M
2  84300903        19.69   ...                     0.08758          M
3  84348301        11.42   ...                     0.17300          M
4  84358402        20.29   ...                     0.07678          M
```

print("Cancer data set dimensions : {}".format(dataset.shape))Cancer data set dimensions : (569, 32)

We can observe that the data set contain 569 rows and 32 columns. ‘*Diagnosis*’ is the column which we are going to predict , which says if the cancer is M = malignant or B = benign. 1 means the cancer is malignant and 0 means benign. We can identify that out of the 569 persons, 357 are labeled as B (benign) and 212 as M (malignant).



```
diagnosis
B    357
M    212
dtype: int64
```

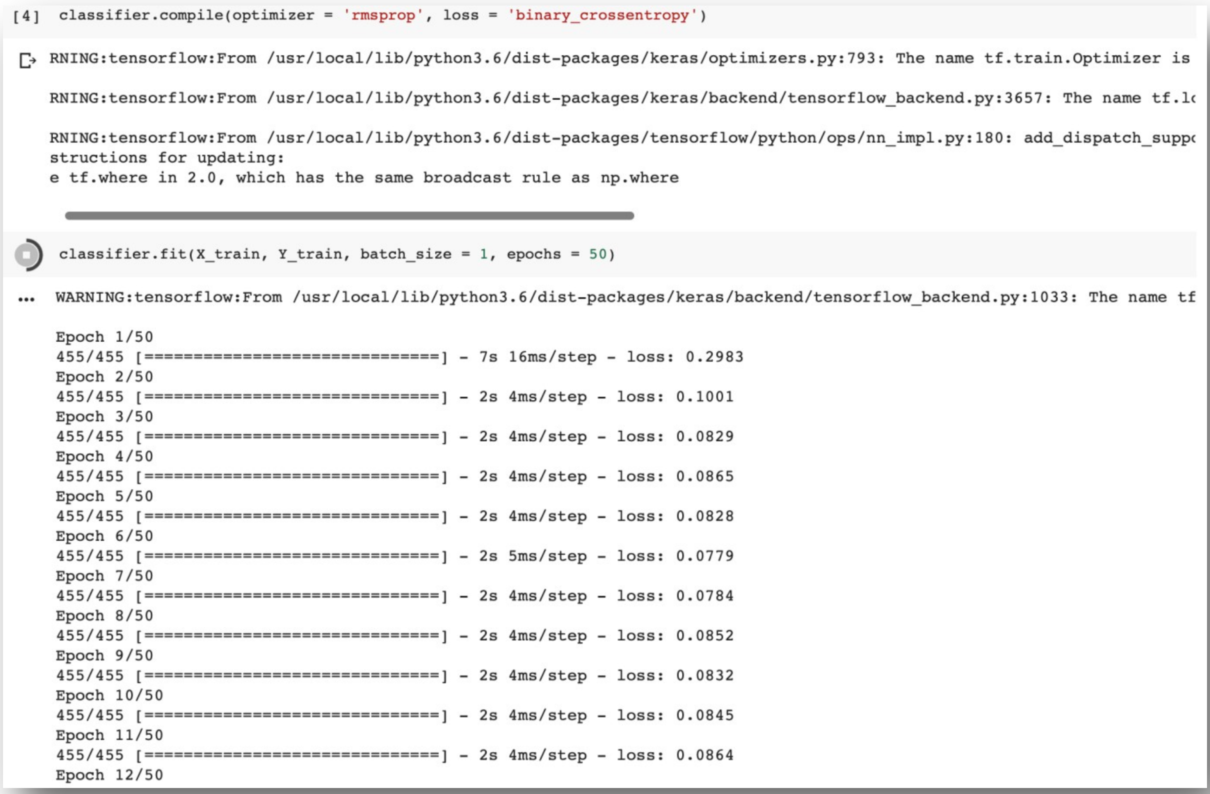
DNN

```
from keras.models import Sequential
from keras.layers import Dense
classifier = Sequential()
classifier.add(Dense(units = 16, activation = 'relu',
input_dim = 30))
classifier.add(Dense(units = 8, activation = 'relu'))
classifier.add(Dense(units = 8, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

We have used Relu and sigmoid actication functions in the model.

Activation *functions* are really important for a Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable.***Their main purpose is to convert a input signal of a node in a A-NN to an output signal.***

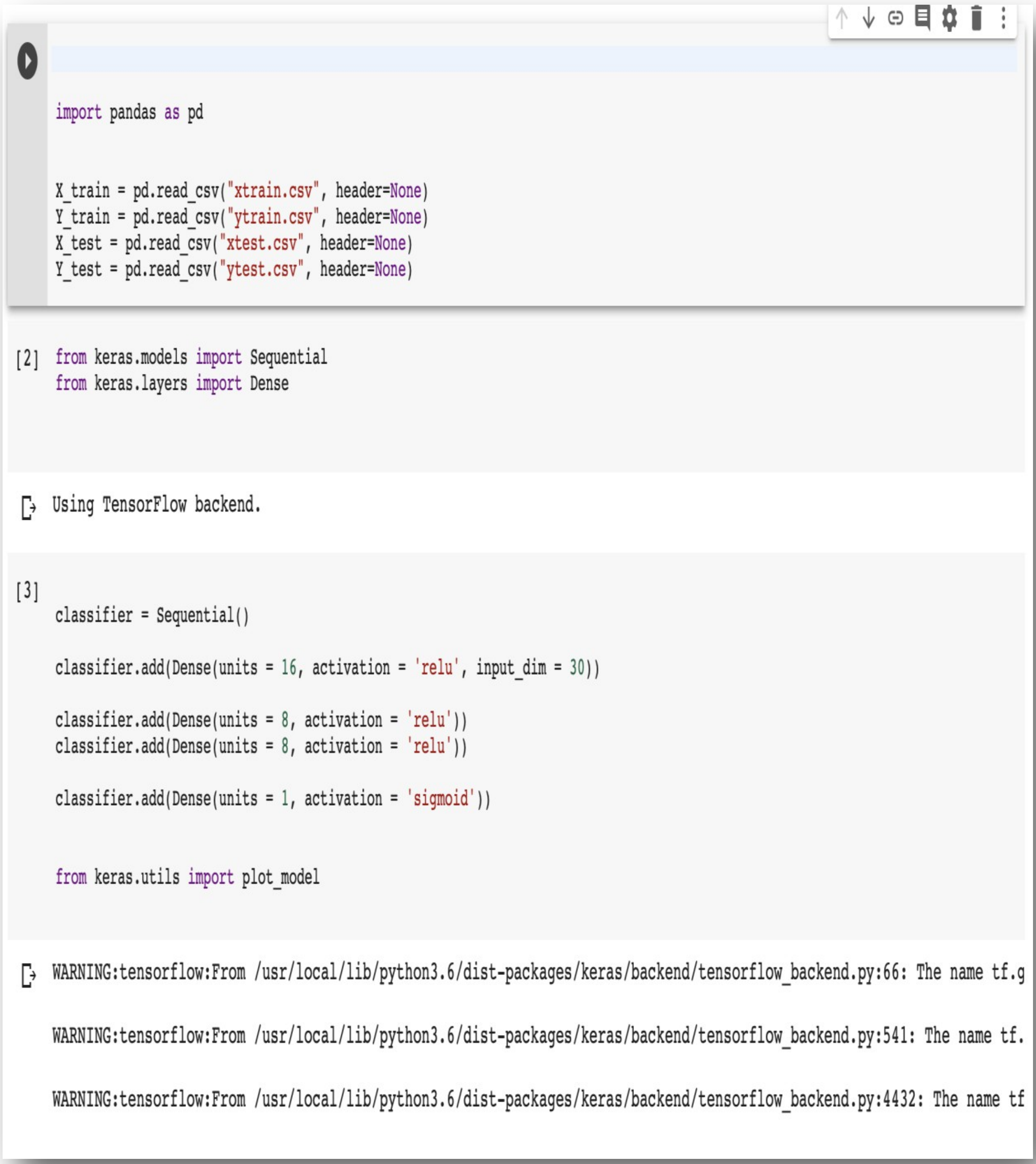
ReLU stands for **rectified linear unit**, and is a type of activation function. Mathematically, it is defined as y = max(0, x).



```
[4] classifier.compile(optimizer = 'rmsprop', loss = 'binary_crossentropy')

RNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is

RNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3657: The name tf.lo

RNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/nn_impl.py:180: add_dispatch_suppo
structions for updating:
e tf.where in 2.0, which has the same broadcast rule as np.where

classifier.fit(X_train, Y_train, batch_size = 1, epochs = 50)

... WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf

Epoch 1/50
455/455 [==============================] - 7s 16ms/step - loss: 0.2983
Epoch 2/50
455/455 [==============================] - 2s 4ms/step - loss: 0.1001
Epoch 3/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0829
Epoch 4/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0865
Epoch 5/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0828
Epoch 6/50
455/455 [==============================] - 2s 5ms/step - loss: 0.0779
Epoch 7/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0784
Epoch 8/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0852
Epoch 9/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0832
Epoch 10/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0845
Epoch 11/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0864
Epoch 12/50
```

Layers we used:

This network has 5 layers. The first with 30 input nodes, the second with 16 nodes, the third and forth with 8 nodes, and the final output layer with just one node for the output value.



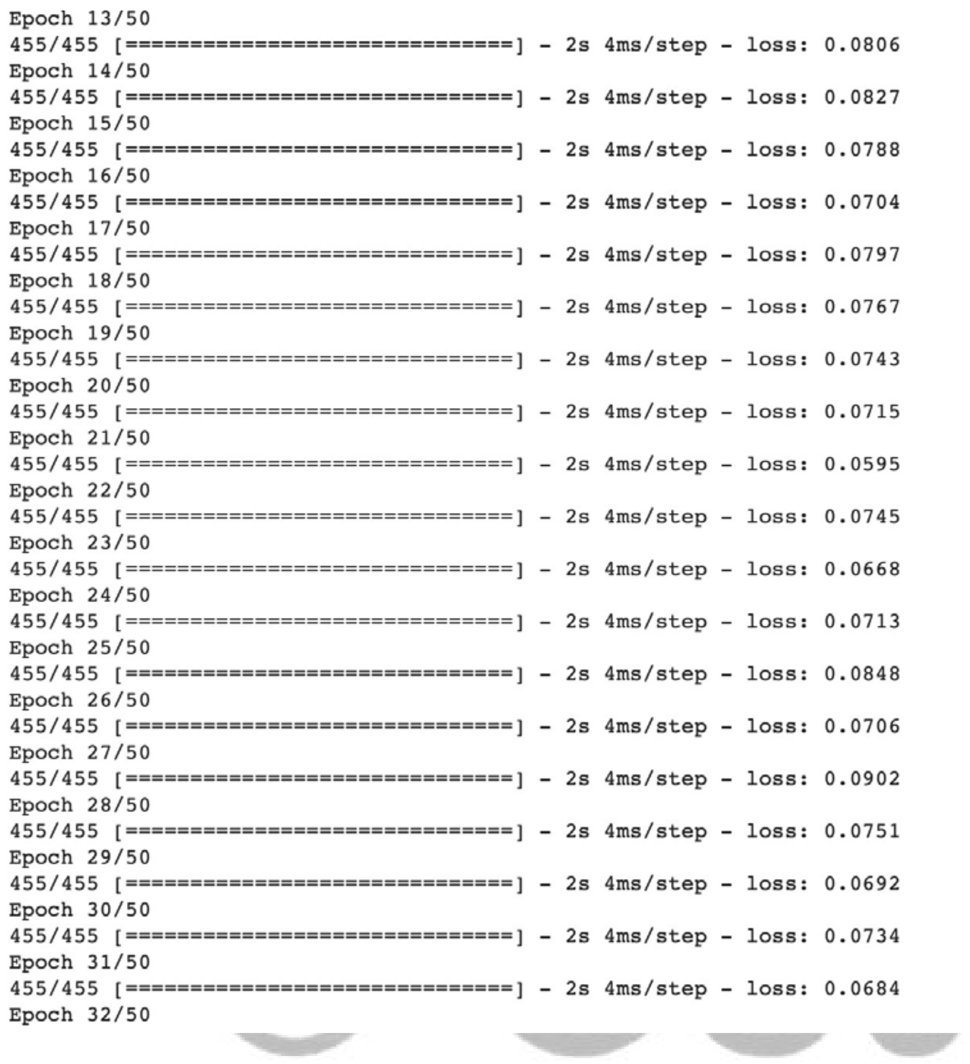
```
import pandas as pd


X_train = pd.read_csv("xtrain.csv", header=None)
Y_train = pd.read_csv("ytrain.csv", header=None)
X_test = pd.read_csv("xtest.csv", header=None)
Y_test = pd.read_csv("ytest.csv", header=None)
```

```
[2] from keras.models import Sequential
    from keras.layers import Dense
```

```
Using TensorFlow backend.
```

```
[3]
    classifier = Sequential()

    classifier.add(Dense(units = 16, activation = 'relu', input_dim = 30))

    classifier.add(Dense(units = 8, activation = 'relu'))
    classifier.add(Dense(units = 8, activation = 'relu'))

    classifier.add(Dense(units = 1, activation = 'sigmoid'))


    from keras.utils import plot_model
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66: The name tf.g

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541: The name tf.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4432: The name tf
```



```
Epoch 13/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0806
Epoch 14/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0827
Epoch 15/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0788
Epoch 16/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0704
Epoch 17/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0797
Epoch 18/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0767
Epoch 19/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0743
Epoch 20/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0715
Epoch 21/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0595
Epoch 22/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0745
Epoch 23/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0668
Epoch 24/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0713
Epoch 25/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0848
Epoch 26/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0706
Epoch 27/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0902
Epoch 28/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0751
Epoch 29/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0692
Epoch 30/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0734
Epoch 31/50
455/455 [==============================] - 2s 4ms/step - loss: 0.0684
Epoch 32/50
```

Training the model

Training a neural network means teaching it to recognise patterns in a dataset and output the correct predictions.We do this by adjusting the weights and biases through gradient descent and backpropagation.The process of calculating how inaccurate the prediction was is known as backpropagation and the process of adjusting the connections to reduce the inaccuracy is known as gradient descent.

classifier.fit(X_train, Y_train, batch_size = 1, epochs = 50)

X_train is the input training data and **Y_train** is the desired corresponding predictions for that data.

Batch size refers to how many examples we want to use for each round of gradient descent (each round of adjusting the connections).

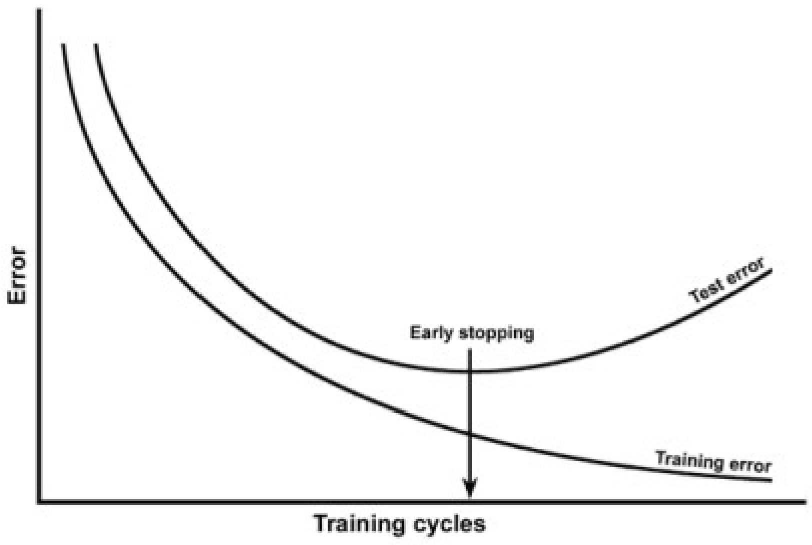
Epochs is the final parameter given to the function. It refers to how many times we want the training algorithm to go through the entire set of examples. Each time the algorithm performs backpropagation and gradient descent on all the training examples then it is counted as one epoch.

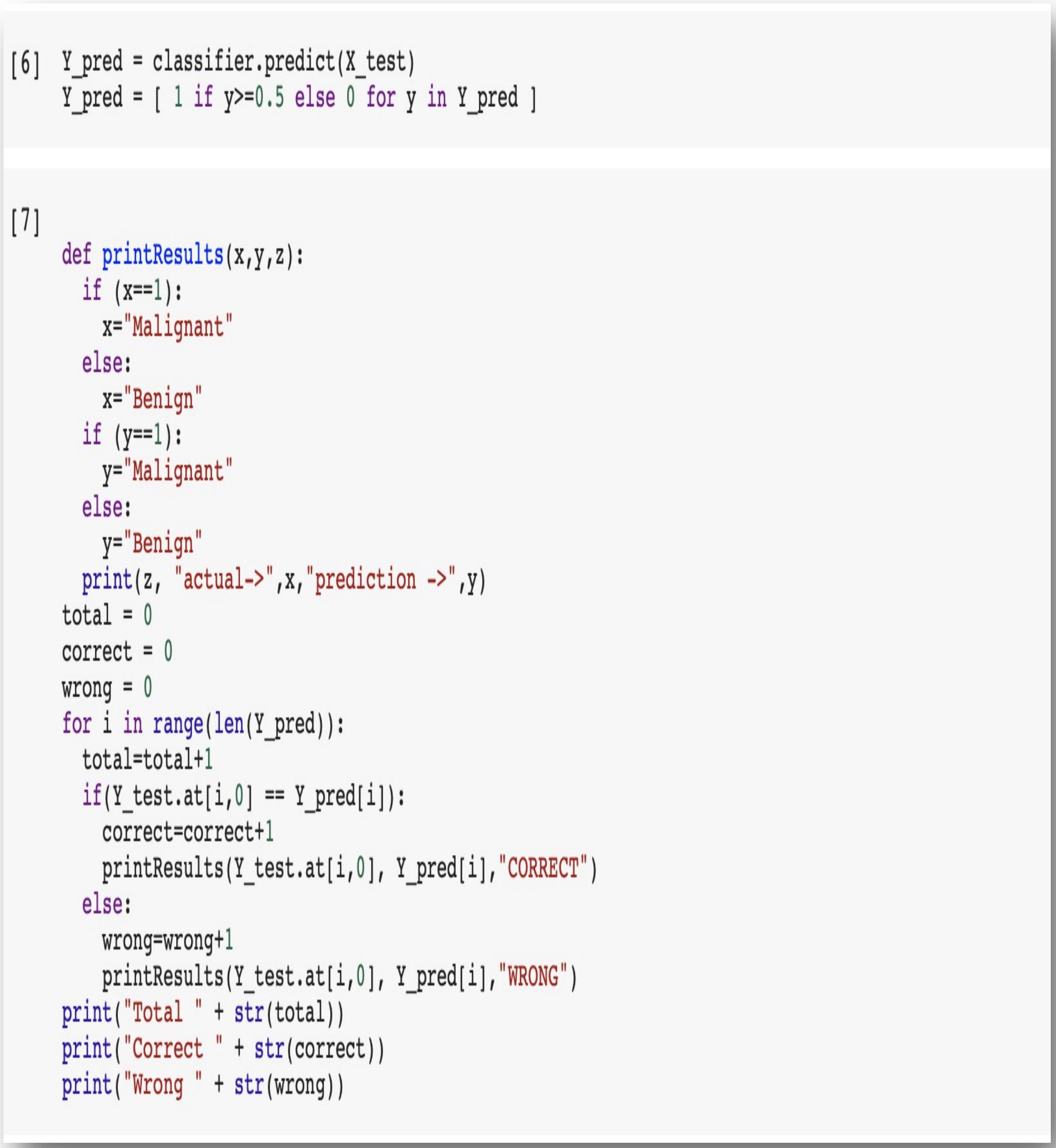
Testing the Model

In a neural network, we have a set of inputs, which results in an output. To test our neural network we want to perform predictions on data it hasn't seen before. This is when we use the data we separated out earlier. All we had to do was feed this data as input to the network and get back the predictions which it predicts. What happens is when we feed through training data, the system compares the output it gets with the expected output. It then applies feedback which adjusts the weighted trigger levels of some of the neurons. Then it tries again with another piece of data, and makes an adjustment. This is done with the following code.

Y_pred = classifier.predict(X_test)

Y_pred = [1 if y>=0.5 else 0 for y in Y_pred]





```
[6]  Y_pred = classifier.predict(X_test)
     Y_pred = [ 1 if y>=0.5 else 0 for y in Y_pred ]
```

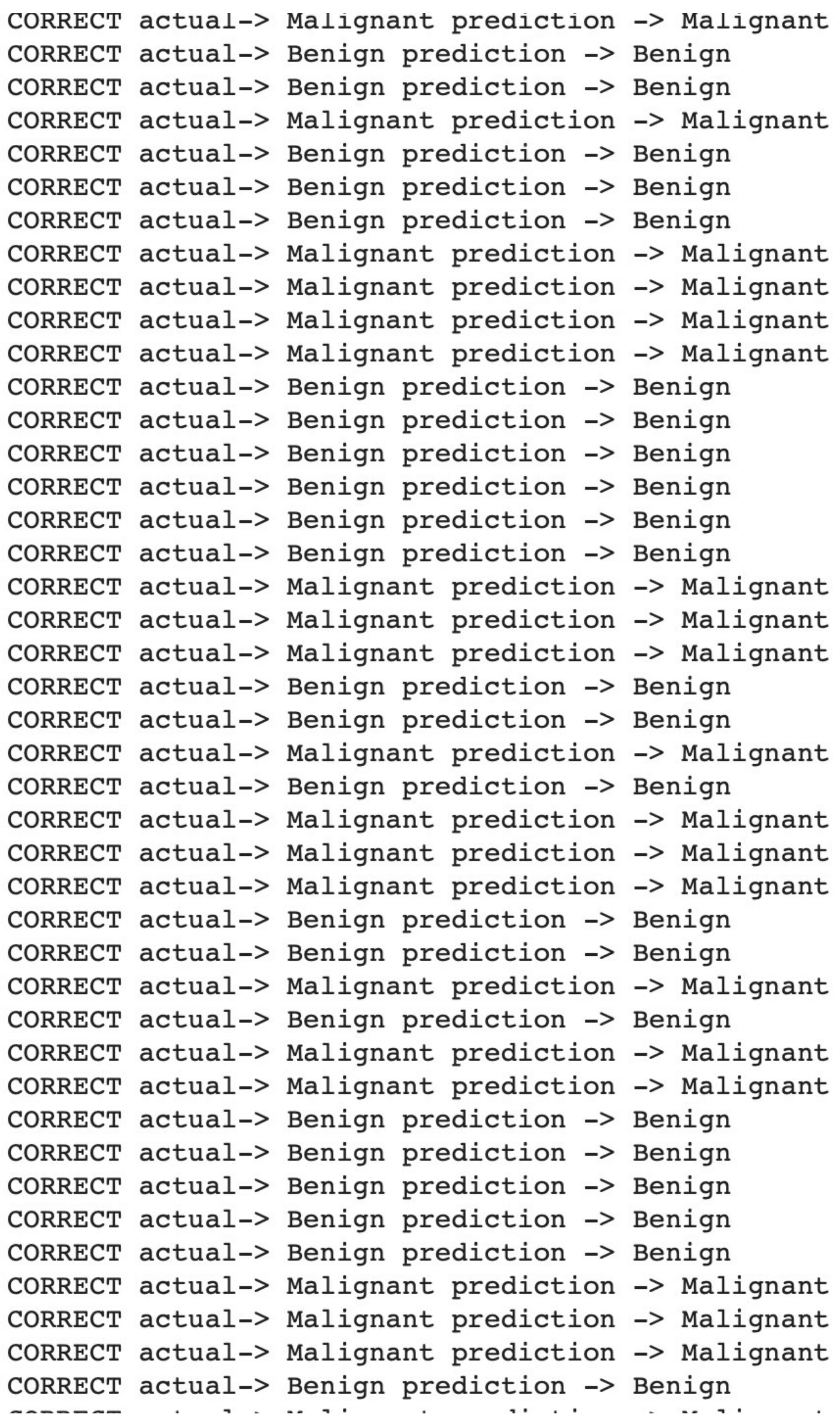
```
[7]
     def printResults(x,y,z):
       if (x==1):
         x="Malignant"
       else:
         x="Benign"
       if (y==1):
         y="Malignant"
       else:
         y="Benign"
       print(z, "actual->",x,"prediction ->",y)
     total = 0
     correct = 0
     wrong = 0
     for i in range(len(Y_pred)):
       total=total+1
       if(Y_test.at[i,0] == Y_pred[i]):
         correct=correct+1
         printResults(Y_test.at[i,0], Y_pred[i],"CORRECT")
       else:
         wrong=wrong+1
         printResults(Y_test.at[i,0], Y_pred[i],"WRONG")
     print("Total " + str(total))
     print("Correct " + str(correct))
     print("Wrong " + str(wrong))
```

Code:

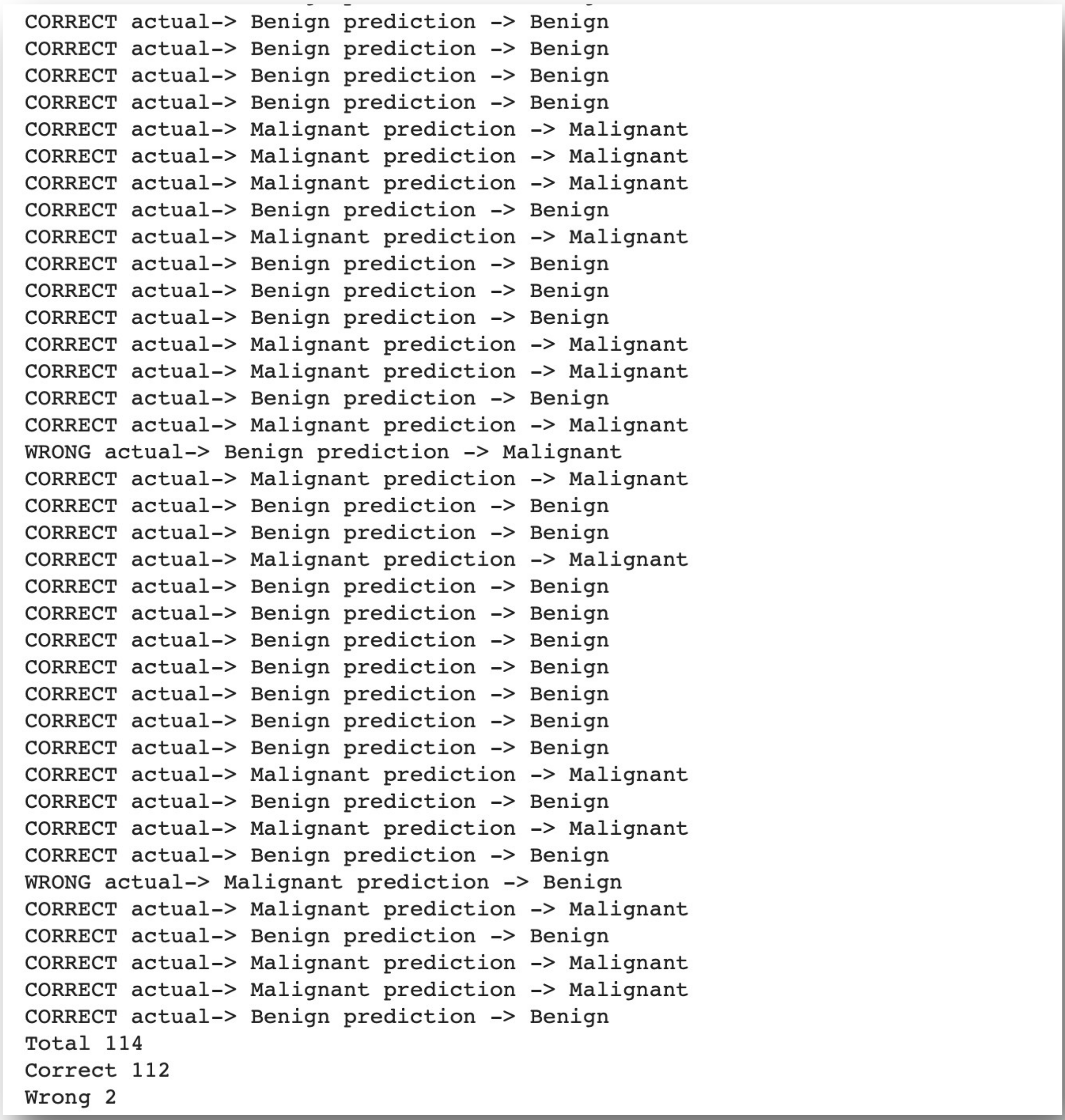
```
def printResults(x,y,z):
 if (x==1):
 x="Malignant"
 else:
   x="Benign" if
 (y==1):
 y="Malignant"
 else:
   y="Benign"
 print(z, "actual->",x,"prediction ->",y)
total = 0
correct = 0
wrong = 0
for i in range(len(Y_pred)):
 total=total+1
 if(Y_test.at[i,0] == Y_pred[i]): #if the prediction is correct
   correct=correct+1
   printResults(Y_test.at[i,0], Y_pred[i],"CORRECT")
 else: #if the prediction is wrong
```

```
    wrong=wrong+1

    printResults(Y_test.at[i,0], Y_pred[i],"WRONG")
print("Total " + str(total))
print("Correct " + str(correct))
print("Wrong " + str(wrong))
```

Now that we have the predictions to better understand them we can print them out with the following code. This code also counts the number of correct and incorrect predictions.



```
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
```



```
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
WRONG actual-> Benign prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
WRONG actual-> Malignant prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Malignant prediction -> Malignant
CORRECT actual-> Benign prediction -> Benign
Total 114
Correct 112
Wrong 2
```

Accuracy

Our model managed to classify 112 out of 114 biopsies correctly**.**

That's a 98.2% accuracy.

Conclusion and Future Work

The diagnosis and treatment of cancer is a key component of any overall cancer control plan. Human body is a complex structure. Finding the intensity of cancer using visual test a can be pretty tough for even a experienced doctor and even a slight delay or deviation from the affected area can be life threatening for the patient. We intended to remove this simple human error in which we succeeded.

Our accuracy was better than what eye can visualise and our project on finding the audacity of cancer was successful and is fully working for industrial use.

The project can be further be used to detect cancer along with the area it is affecting and the size of tumour can be detected also. The model can further be enhanced to tell the approach use for the treatment of the cancer like chemotherapy, radiation therapy etc.

References

- https://classroom.udacity.com/courses/ud187
- https://github.com/SBU-BMI/quip_cancer_segmentation
- https://github.com/MohamedAliHabib/Brain-Tumor-Detection/blob/master/.ipynb_checkpoints/Brain%20Tumor%20Detection-checkpoint.ipynb
- https://www.tensorflow.org/tutorials
- https://codelabs.developers.google.com/