BREAST CANCER CLASSIFICATION USING DEEP LEARNING

A MINI PROJECT REPORT

Submitted by

NIDIN	**1905034**
SANJAY PRATAP T K	**1905046**
YOKESH R S	**1905062**
ABUTHAGEER	**2005201**

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

COIMBATORE INSTITUTE OF TECHNOLOGY

(Government Aided Autonomous Institution Affiliated to Anna University)

COIMBATORE-641 014

ANNA UNIVERSITY - CHENNAI 600 025

JUNE 2022

COIMBATORE INSTITUTE OF TECHNOLOGY
(Government aided Autonomous Institution Affiliated to Anna University)
COIMBATORE – 641014

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this mini project **"Breast Cancer Classification using Deep Learning"** is the bonafide work of **NIDIN S (1905034), SANJAY PRATAP T K (1905046), YOKESH R S (1905062), and ABUTHAGEER S (2005201)** under my supervision during the academic year 2021-2022.

Dr.G.Kousalya, M.E.,Ph.D	**Dr.S.P.Abirami, Ph.D**
HEAD OF THE DEPARTMENT,	**ASSISTANT PROFESSOR,**
Department of CSE	Department of CSE,
Coimbatore Institute of Technology,	Coimbatore Institute of
Technology,Coimbatore – 641 014.	Coimbatore – 641 014.

Submitted for **19CS65 Mini Project** held on

____

Internal Examiner **External Examiner**

Place:

Date:

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We express our sincere thanks to our Secretary **Dr.R.Prabhakar** and our Principal **Dr. Rajeswari A** for providing us a great opportunity to carry out our work. The following words are a small part to express our gratitude to them. This work is the outcome of their inspiration and product of plethora of their knowledge and rich experience.

We record the deep sense of gratefulness to **Dr.G.Kousalya**, Head of the Department of Computer Science and Engineering, for his encouragement during this tenure.

We equally tender our sincere thankfulness to our project guide **Dr.S.P.Abirami**, Assistant Professor, Department of Computer Science and Engineering, for her invaluable suggestions and guidance during this course.

During the entire period of study, the entire staff members of the Department of Computer Science and Engineering have offered ungrudging help. It is also a great pleasure to acknowledge the unfailing help we have received from our friends.

It is matter of great pleasure to thank our parents and family members for their constant support and co-operation in the pursuit of this endeavor.

ABSTRACT

Globally, breast cancer is the most common cancer among women, and the most likely cause of female cancer deaths as per statistical analysis given by WHO. High-income countries (HICs) have made the most progress in improving breast cancer outcomes. Between 1990 and 2014, breast cancer death rates dropped by 34% in the US attributable to the combination of improved earlier detection and effective adjuvant therapies. By contrast, breast cancer is an increasingly urgent problem in low- and middle-income countries (LMICs), where historically low incidence rates have been rising by up to 5% per year. In view of earlier detection of breast cancer, the research aims in developing a classification model using CNN. The model primarily uses Wisconsin Dataset for training the model and classifies the cancer tissues. The accuracy of the model is evaluated and found to be better compared to other existing models. The system could be further improved in integrating the image inputs to the optimized features.

CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION:

Breast Cancer is the most affected disease between women all over the world. All most 25% of all cancers with an estimated 1.67 million new cancer cases diagnosed in 2012 and its incidence is increasing day by days. It is also said that Breast cancer is the second leading cause of death for women worldwide.

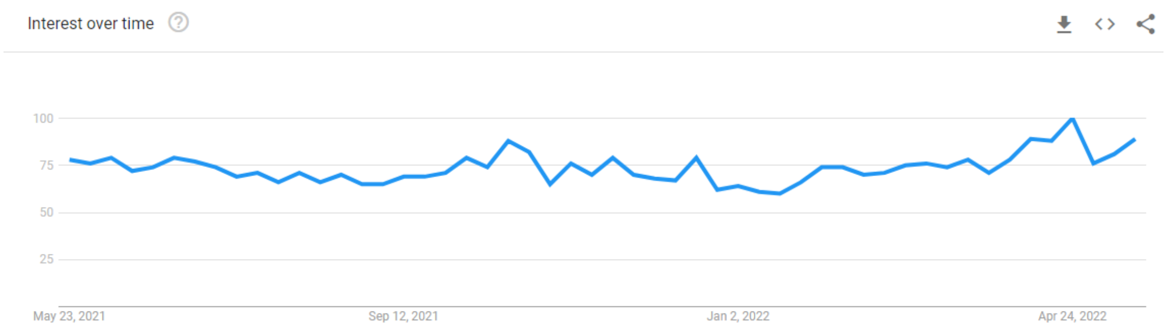
Early detection of cancer can reduce the risk of deaths for cancer patients**.** To increase the survival rate the early diagnosis of breast cancer and a trustworthy detection model is required.

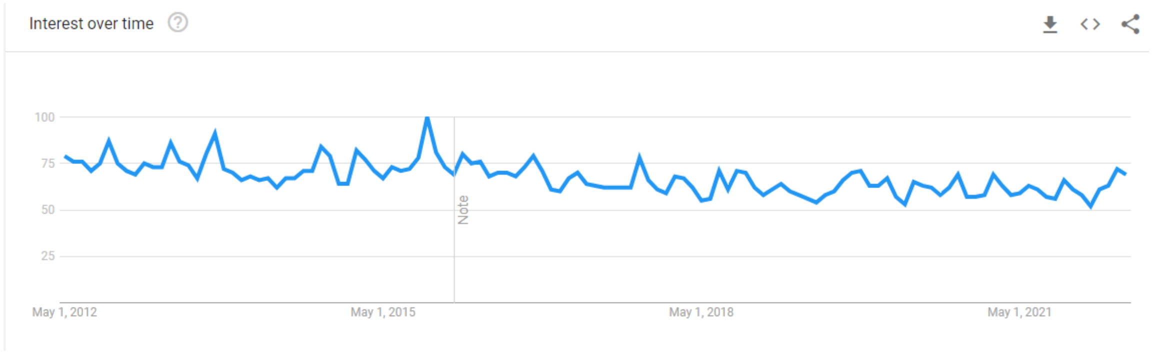
The goal of the research is to identify and classify Malignant and Benign patients by proposing a simple and efficient model for early detection of breast cancer with minimal error percentage.

This model uses an Artificial neural network and produces Accuracy of 98% and f1 score as 98% - Benign and 97% - Malignant .

1.2. Google trend analysis:

The diagram shown below displays the google trend analysis about the domain of this project.





.

CHAPTER 2

LITERATURE SURVEY

2.1 Breast Cancer Detection using K-nearest Neighbor Machine Learning Algorithm (2016)

Authors: Moh'd Rasoul Al-hadidi,Abdulsalam Alarabeyyat,Mohannad Alhanahnah.

Description

In this paper, we proposed a new method to detect the breast cancer with high accuracy.This method consists of two main parts, in the first part the image processing techniques are used to prepare the mammography images for feature and pattern extraction process. The extracted features are utilized as an input for a two types of supervised learning models, which are Back Propagation Neural Network (BPNN) model and the Logistic Regression (LR) model .The LR model resulted in Accuracy of 45% with 750 features and BPNN with 93.7 % which is comparatively extremely good but with only 240 features.

Advantages

1. They have used image processing to obtain the features for training the prediction model and used have BPNN and Logistic regression.

Disadvantages

1 .The accuracy rate of the model greatly affects with the increase in the features and the maximum accuracy they could bring out was 93.7% only.

2.2 Efficient Approaches for Accuracy Improvement of Breast Cancer Classification Using Wisconsin Dataset (2017)

Authors: Shajib Ghosh ,Jubaer Hossain, Dr.Shaikh Anowarul Fattah , Dr. Celia Shahnaz, Asir Intisar Khan

Description:

This paper deals with different statistical and deep learning analysis of Wisconsin Breast Cancer Database for improving the accuracy in detection and classification of breast cancer based on different attributes. Applying Naïve Bayes, SVM, Logistic Regression, KNN, Random Forest, MLP and CNN classifiers, higher accuracy is obtained which is up to 98% to 99%.

Advantages:

1. The proposed goes through a lot of classification models and compares the accuracy for each.

Disadvantages

1. Harder to optimize the network as convergence time significantly increases.

2.3 Breast Cancer Diagnosis Using deep learning Algorithm (2018)

Authors: Naresh Khuriwal , Dr. Nidhi Mishra

Description:

In this paper they've proposed the deep learning method convolutional neural network that mostly used for classification of images dataset. It is basically divided into three parts first collection of dataset and applied pre-processing algorithm for scaled and filter data then split dataset for training and testing purpose and generate some graph for visualization data. They've concluded deep learning technology is a good way for diagnosis breast cancer with Wisconsin Breast Dataset. This database provides 569 rows and 30 features in the dataset. In this paper they've **just used 11 features** for diagnosis .After the implementation they've achieved 98.67% accuracy.

Advantages:

1. Uses Convolutional Neural network faster and better Accuracy

Disadvantages:

1. Uses only 11 features of given 30 features.

2.4 Breast Cancer Detection Based on Deep Learning Technique (2019)

Author: Nur Syahmi Ismail and Cheab Sovuthy.

Description:

In this paper, they’ve used deep learning technique using VGG16 and ResNet50 network to classify between normal tumour and abnormal tumour using IRMA dataset and have implemented for normal and abnormal breast cancer detection. The classification methods were evaluated using three performance evaluations which are precision, recall, and accuracy rate. The best result of classification accuracy was VGG16 with 94%. Compared to ResNet50 with 91.7% in term of accuracy.

Disadvantage:

1. Usage of already available Model
2. Poor accuracy.

2.5 Breast Cancer Prediction Using Machine Learning (2020)

Author: Ramik Rawal.

Description:

In this paper they have used four algorithms SVM, Logistic Regression, Random Forest, and KNN to predict the outcome the breast cancer. It broadly speaks about **three** domains. First domain is prediction of cancer before diagnosis, second domain is prediction of diagnosis and treatment and third domain focuses on outcome during treatment. The accuracy obtained by SVM (97.13%) is better than the accuracy obtained Naïve Bayes and k-NN that have an accuracy that varies between 95.12 % and 95.28 %. It was also clear with SVM having highest value of correctly classified instances and the lower value of incorrectly classified instances than the other classifiers.

Advantage:

1. Better preprocessing Technique
2. Average accuracy of different models was high.

Disadvantage:

1. Efficiency of benign class and malignant class got the highest for two different algorithms such as SVM for benign class and K-NN for malignant class

CHAPTER 3
METHODOLOGY

3.1 INTRODUCTION

This chapter explains the entire methodology that has been used in the system in detail. The methods used are Artificial Neural Network and Convolutional Neural Networks. It also covers the hardware and software specifications requirements that have been used to work on this system.

3.2 METHODS USED

3.2.1 Convolutional Neural Network:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning system that can take an input image, assign relevance (learnable weights and biases) to various aspects/objects in the image, and distinguish between them. When compared to other classification methods, the amount of pre-processing required by a ConvNet is significantly less.

The architecture of a ConvNet is inspired by the organisation of the Visual Cortex and is akin to the connectivity pattern of Neurons in the Human Brain. Individual neurons can only respond to stimuli in a small area of the visual field called the Receptive Field. A number of similar fields can be stacked on top of each other to encompass the full visual field.In convolutional neural networks, the major building elements are convolutional layers.

A convolution is the outcome of applying a filter to an input and activating it. The location and strength of a detected feature in an input, like as an image, are shown by a map of activations termed a feature map, which is created by repeatedly applying the same filter to an input.
The capacity of convolutional neural networks to learn a large number of filters in parallel particular to a training dataset under the restrictions of a certain predictive modelling

problem, such as image classification, is its unique feature. As a result, extremely specialised traits appear on input photographs that can be identified everywhere.

3.2.2 Artificial Neural Network:

Biological neural networks establish the structure of the human brain, and the phrase "Artificial Neural Network" is taken from them. Artificial neural networks, like the human brain, have neurons that are coupled to one another in various layers of the networks. Nodes are the name for these neurons. An artificial neural network is made up of three or more interconnected layers. Input neurons make up the first layer. These neurons send input to deeper layers, which then deliver the final output data to the final output layer. All of the inner layers are concealed and are made up of units that use a series of transformations to modify the information received from layer to layer. Each layer serves as both an input and an output layer, allowing the ANN to comprehend more complicated things. The neural layer is the collective name for these inner layers. The neural layer's units attempt to learn about the data collected by weighing it according to the ANN's internal framework. These principles enable units to provide a changed result, which is subsequently sent to the following layer as an output.

Backpropagation, a mechanism by which the ANN can alter its output results by taking errors into account, is used in another set of learning rules. Each weight is adjusted in accordance to how much they contributed to the inaccuracy. As a result, the error is utilised to reweight the ANN's unit connections to account for the discrepancy between the desired and actual outcomes. Over time, the ANN will "learn" how to reduce the likelihood of errors and undesirable outcomes.

3.3 STEPS OF IMPLEMENTATION:

1. Dividing the dataset:

The dataset that is used in the ANN model is WISCONSIN dataset and the histopathological images of malignant, benign and normal classes are given as input to the EfficientNet B7 model for classification.
The Wisconsin dataset is first divided into 70 % for training the ANN model and the remaining 30% for testing or validating the model. The WISCONSIN dataset consists of 33 features and 569 rows of data.

2. Preprocessing the dataset:

The two unwanted features of the WISCONSIN dataset namely the ‘id’,’Unnamed’ columns are removed from the dataset. The diagnosis class feature is encoded with 0 for Benign and 1 for Malignant. MinMaxScaler() is imported from sklearn library and the data is scaled in the particular common range of values.

And for the images the mask images removed from the dataset ,which are unnecessary and the total image size reduces to 780.Whenever the image is passed into the trained model we divide the pixel of each image by 224x224 to reduce the image size , indeed reducing the image size.

3. Build and train the model:

The CNN network consists a set of customized layers such as conv2D, max-pooling layer, activation layer, dropout layer, dense layer. The augmented images were fed to the network which in turn returns the feature learnt from the image to the output layer which detects the diseases present in the particular leaf.

4. Output:

The ANN which is trained using the WISCONSIN dataset gives the accuracy and the predicts for any input value of the same features that is used to train the model. And the EfficientNet B7 model which is trained with the image dataset gives the prediction either of three classes Benign or Malignant or Normal for any histopathological image given as input.

3.4 SYSTEM SPECIFICATIONS:

Hardware specifications:

- Operating System: Windows 10
    - Processor : Ryzen 3/ Intel i5
- Hard disk : 500 GB
- RAM : 8GB (minimum)

Software specifications:

- IDE : Google Colab, Jupyter Notebook
- Coding Language : Python

3.5 SUMMARY:

All the methods used are explained in detail in this chapter. The tools used in the current implementation are also given above. It also provides the steps of implementations with a brief description of the process under each step.

CHAPTER 4
SYSTEM ANALYSIS

4.1 INTRODUCTION

This chapter explains the existing system and the proposed system in detail.

4.2 EXISTING SYSTEM

The existing system for Breast cancer Classification include just only the predictor using the simple learning algorithms and have ended up with good accuracy, on the other hand with few input as training images and with is insufficient to really prove a given data benign or malignant and less accuracy.

4.3 PROPOSED SYSTEM

Our model uses EfficientNet b7 as a base model for training and testing the data, which are most advanced model developed with CNN and really a good model for classification using image datasets. The model uses 780 image datasets including benign, malignant and normal for training the model gives best result for further classification on input of any histopathological image.

4.4 SUMMARY:

This chapter gives an overview of the existing system and the proposed system. The proposed system uses powerful Convolutional Neural Networks of EfficientNet B7 to accurately predict for the given image sample as benign or malignant or normal sample.

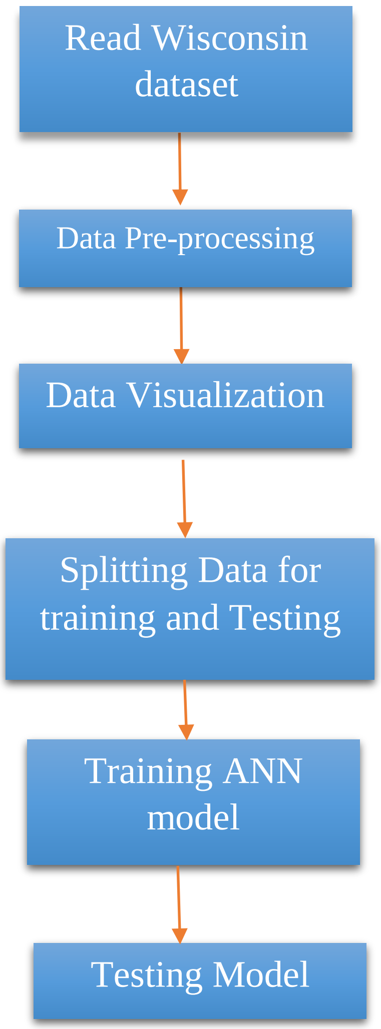
CHAPTER 5
SYSTEM DESIGN

5.1. INTRODUCTION:

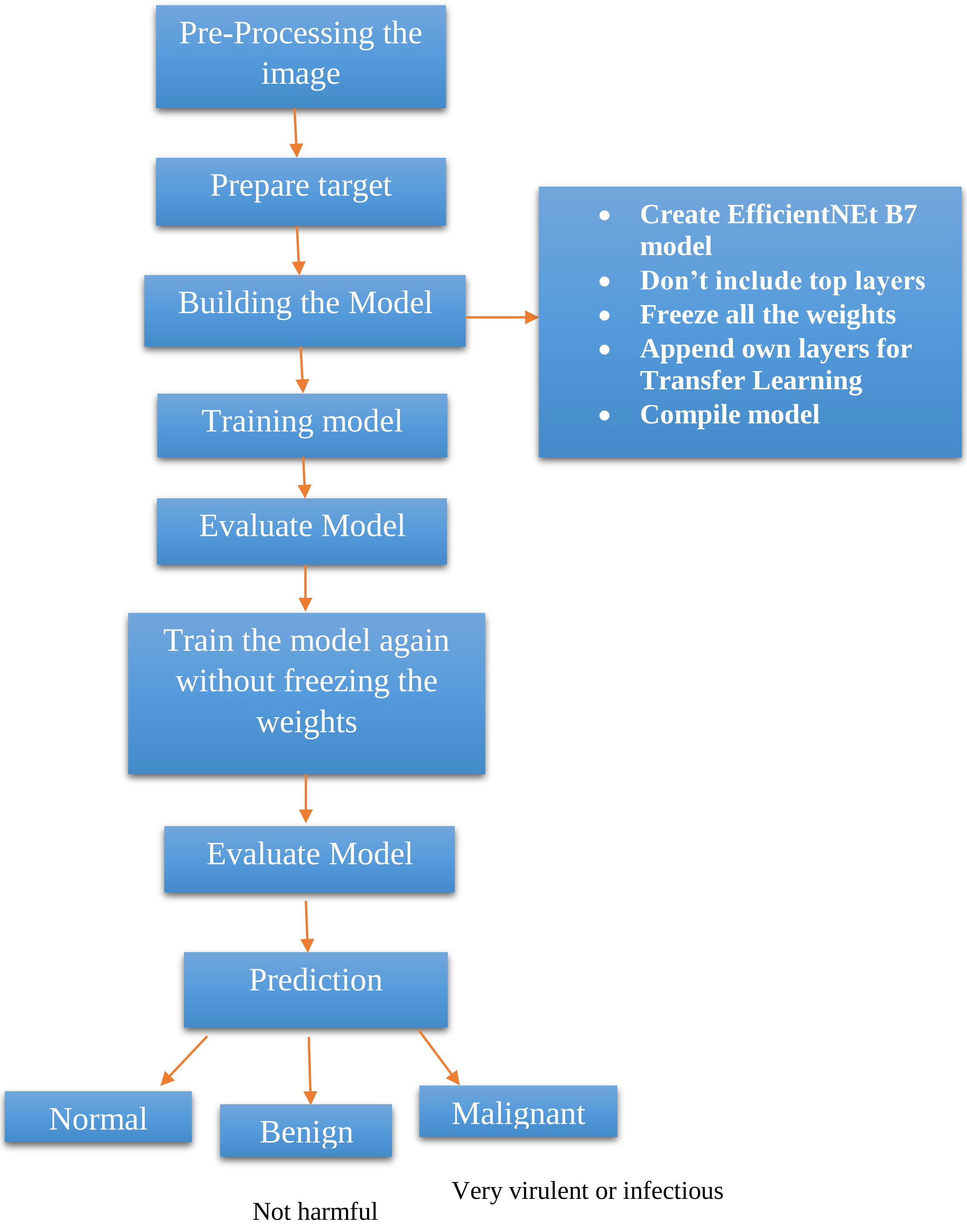
This chapter discusses the flow of this system.

5.2. MODEL AND SYSTEM ARCHITECTURE:

i) ANN Model :



ii) EfficientNet B7:



5.3. SUMMARY:

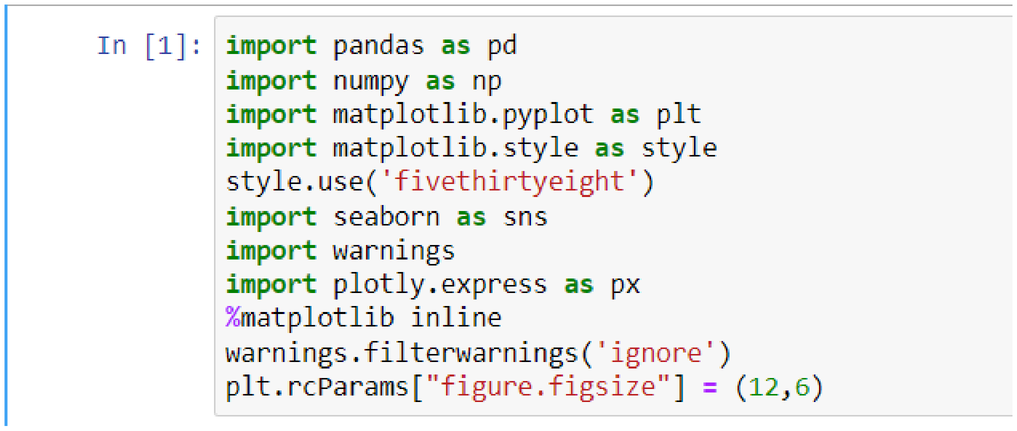
This chapter provides the overall system design and model architecture which facilitates better understanding.

CHAPTER 6

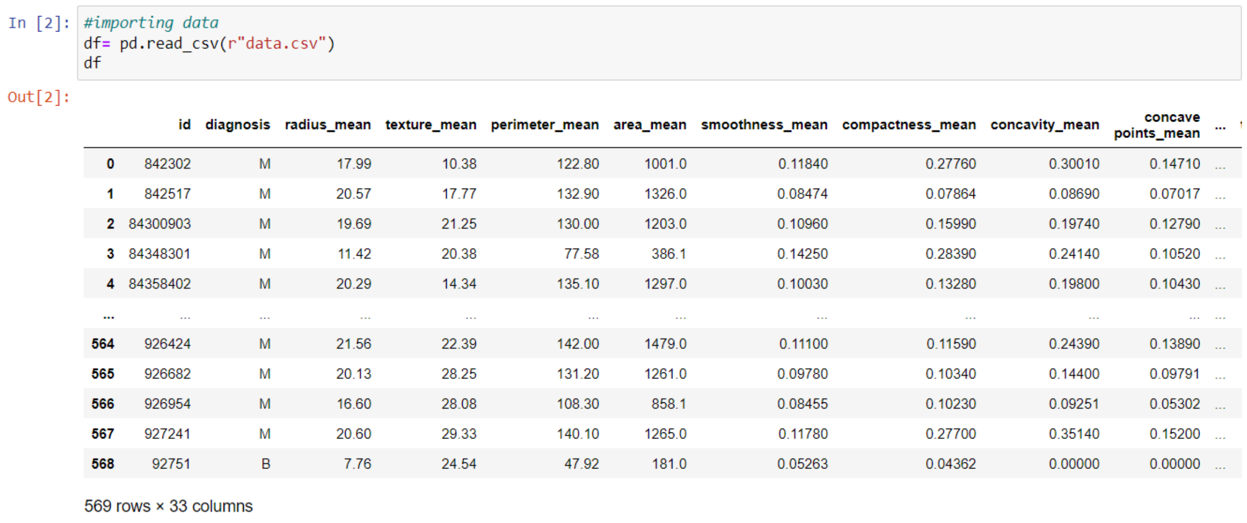
IMPLEMENTATION & RESULT

6.1) ANN MODEL:

Fig.6.1.1 Importing libraries and dataset:



```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib.style as style
        style.use('fivethirtyeight')
        import seaborn as sns
        import warnings
        import plotly.express as px
        %matplotlib inline
        warnings.filterwarnings('ignore')
        plt.rcParams["figure.figsize"] = (12,6)
```



```
In [2]: #importing data
        df= pd.read_csv(r"data.csv")
        df
Out[2]:
```

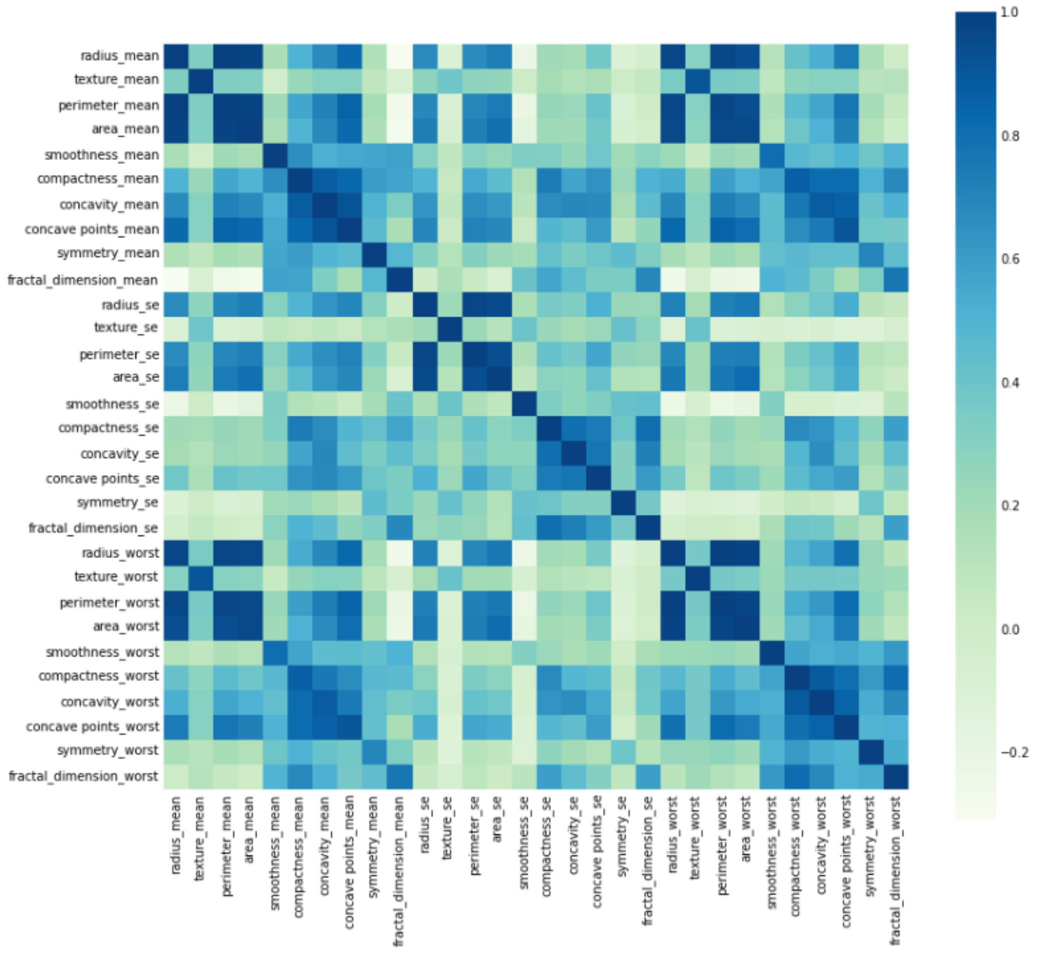
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...
...	...	...	...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...

569 rows × 33 columns

Fig.6.1.2 Data Pre-processing and Data Visualization:

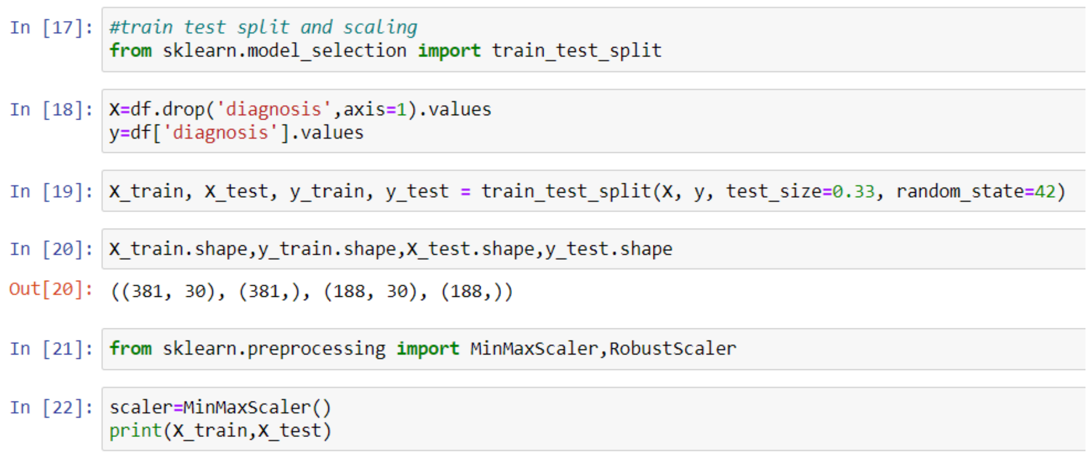
```
In [3]: #dropping useless data
        df.drop(['id','Unnamed: 32'],inplace=True,axis=1)
```

```
In [9]: #correlations
        plt.figure(figsize=(12,12))
        sns.heatmap(df.drop('diagnosis',axis=1).corr(),cmap='GnBu',square=True);
```



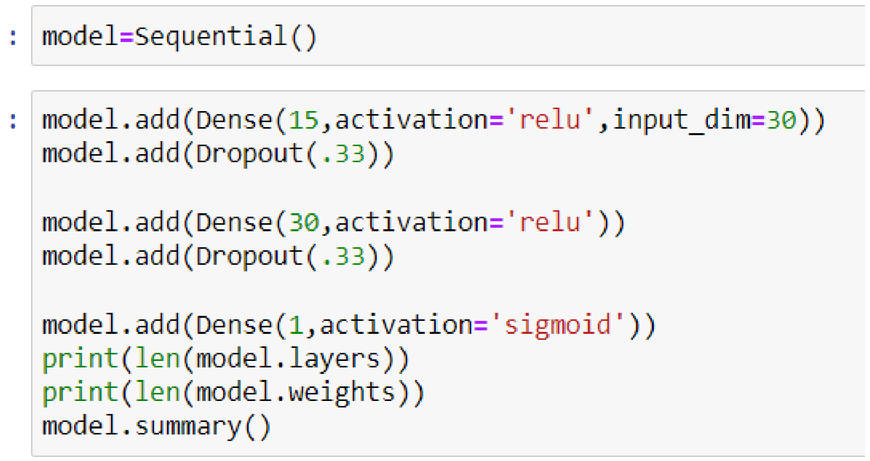
```
In [10]: #converting Malignant and Benign to 1-0
         df['diagnosis']=df['diagnosis'].map({'B':0,'M':1})
```

Fig.6.1.3 Splitting dataset for train and test:

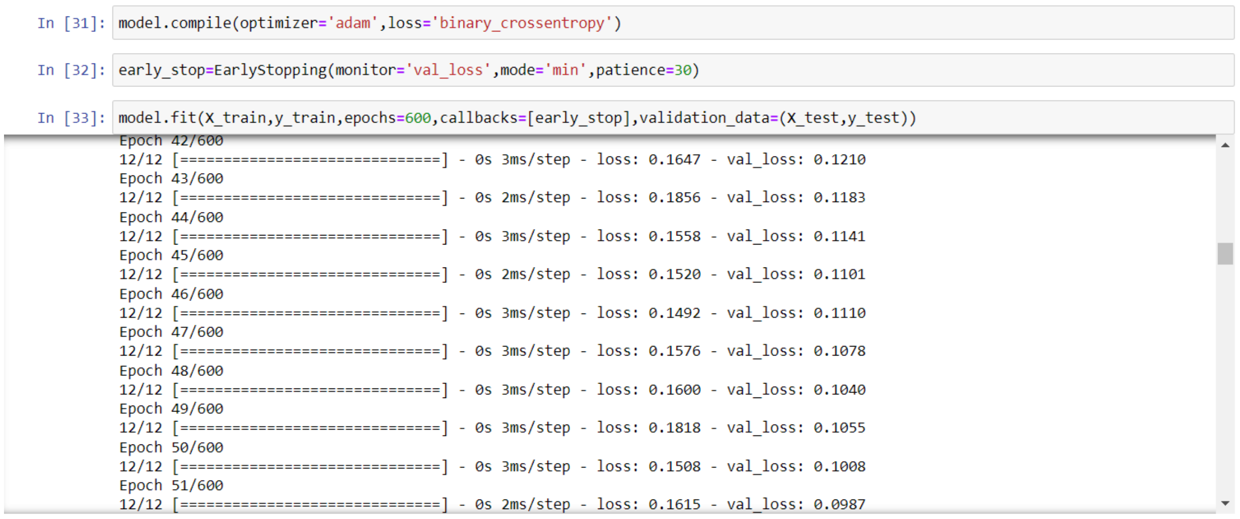


```
In [17]: #train test split and scaling
         from sklearn.model_selection import train_test_split

In [18]: X=df.drop('diagnosis',axis=1).values
         y=df['diagnosis'].values

In [19]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

In [20]: X_train.shape,y_train.shape,X_test.shape,y_test.shape

Out[20]: ((381, 30), (381,), (188, 30), (188,))

In [21]: from sklearn.preprocessing import MinMaxScaler,RobustScaler

In [22]: scaler=MinMaxScaler()
         print(X_train,X_test)
```

Fig.6.1.4 Train the model:

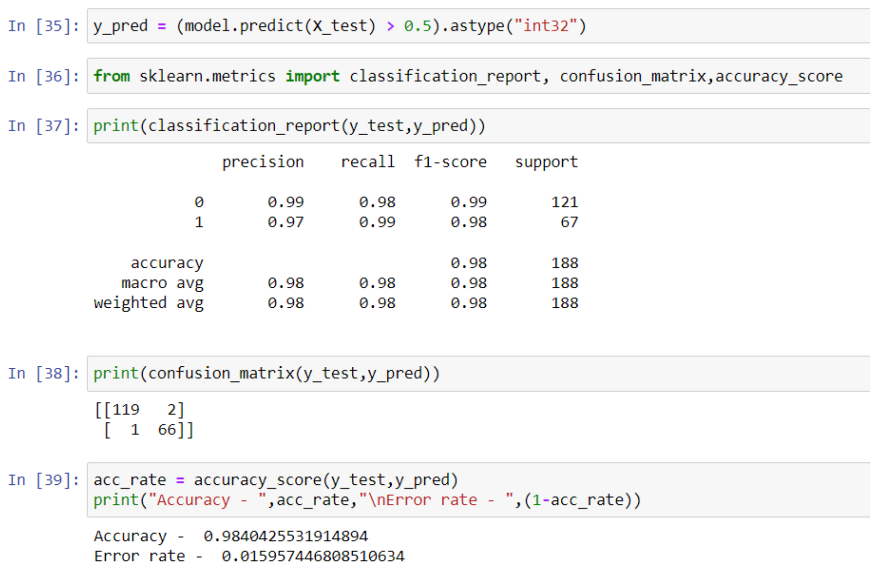


```
: model=Sequential()

: model.add(Dense(15,activation='relu',input_dim=30))
  model.add(Dropout(.33))

  model.add(Dense(30,activation='relu'))
  model.add(Dropout(.33))

  model.add(Dense(1,activation='sigmoid'))
  print(len(model.layers))
  print(len(model.weights))
  model.summary()
```



```
In [31]: model.compile(optimizer='adam',loss='binary_crossentropy')

In [32]: early_stop=EarlyStopping(monitor='val_loss',mode='min',patience=30)

In [33]: model.fit(X_train,y_train,epochs=600,callbacks=[early_stop],validation_data=(X_test,y_test))
         Epoch 42/600
         12/12 [==============================] - 0s 3ms/step - loss: 0.1647 - val_loss: 0.1210
         Epoch 43/600
         12/12 [==============================] - 0s 2ms/step - loss: 0.1856 - val_loss: 0.1183
         Epoch 44/600
         12/12 [==============================] - 0s 3ms/step - loss: 0.1558 - val_loss: 0.1141
         Epoch 45/600
         12/12 [==============================] - 0s 2ms/step - loss: 0.1520 - val_loss: 0.1101
         Epoch 46/600
         12/12 [==============================] - 0s 3ms/step - loss: 0.1492 - val_loss: 0.1110
         Epoch 47/600
         12/12 [==============================] - 0s 3ms/step - loss: 0.1576 - val_loss: 0.1078
         Epoch 48/600
         12/12 [==============================] - 0s 3ms/step - loss: 0.1600 - val_loss: 0.1040
         Epoch 49/600
         12/12 [==============================] - 0s 3ms/step - loss: 0.1818 - val_loss: 0.1055
         Epoch 50/600
         12/12 [==============================] - 0s 3ms/step - loss: 0.1508 - val_loss: 0.1008
         Epoch 51/600
         12/12 [==============================] - 0s 2ms/step - loss: 0.1615 - val_loss: 0.0987
```

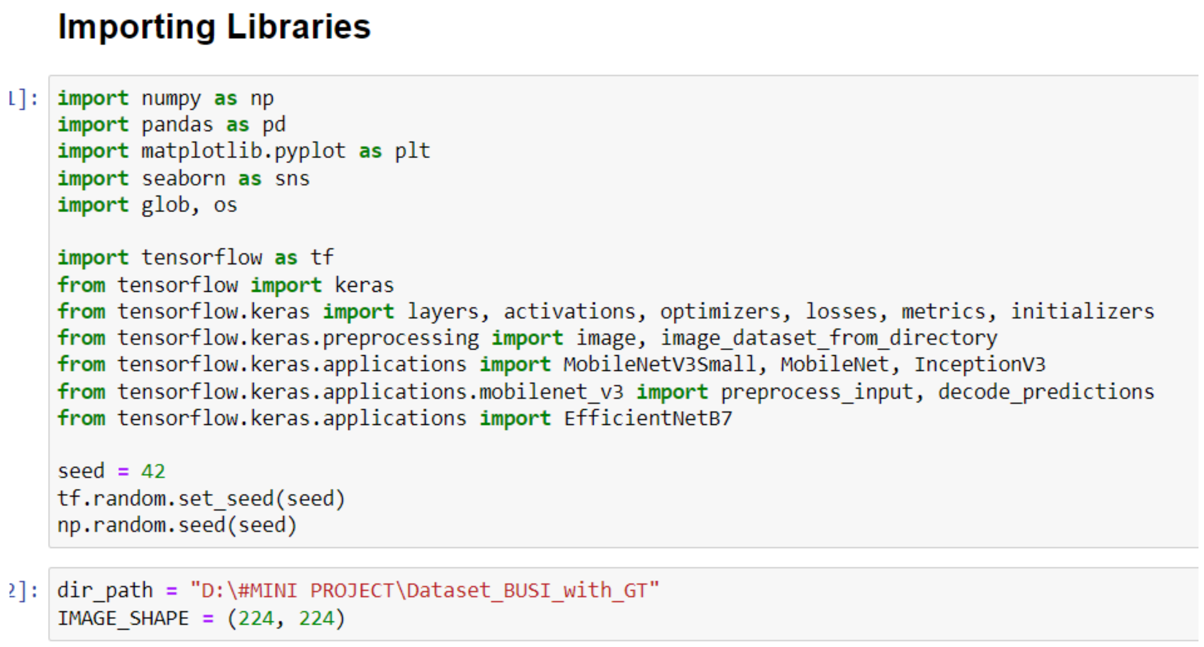
Fig.6.1.5 Testing the Model and producing the confusion matrix:



```
In [35]: y_pred = (model.predict(X_test) > 0.5).astype("int32")

In [36]: from sklearn.metrics import classification_report, confusion_matrix,accuracy_score

In [37]: print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       0.99      0.98      0.99       121
           1       0.97      0.99      0.98        67

    accuracy                           0.98       188
   macro avg       0.98      0.98      0.98       188
weighted avg       0.98      0.98      0.98       188


In [38]: print(confusion_matrix(y_test,y_pred))

[[119   2]
 [  1  66]]

In [39]: acc_rate = accuracy_score(y_test,y_pred)
         print("Accuracy - ",acc_rate,"\nError rate - ",(1-acc_rate))

Accuracy -  0.9840425531914894
Error rate -  0.015957446808510634
```

6.2)EfficientNet B7 model :

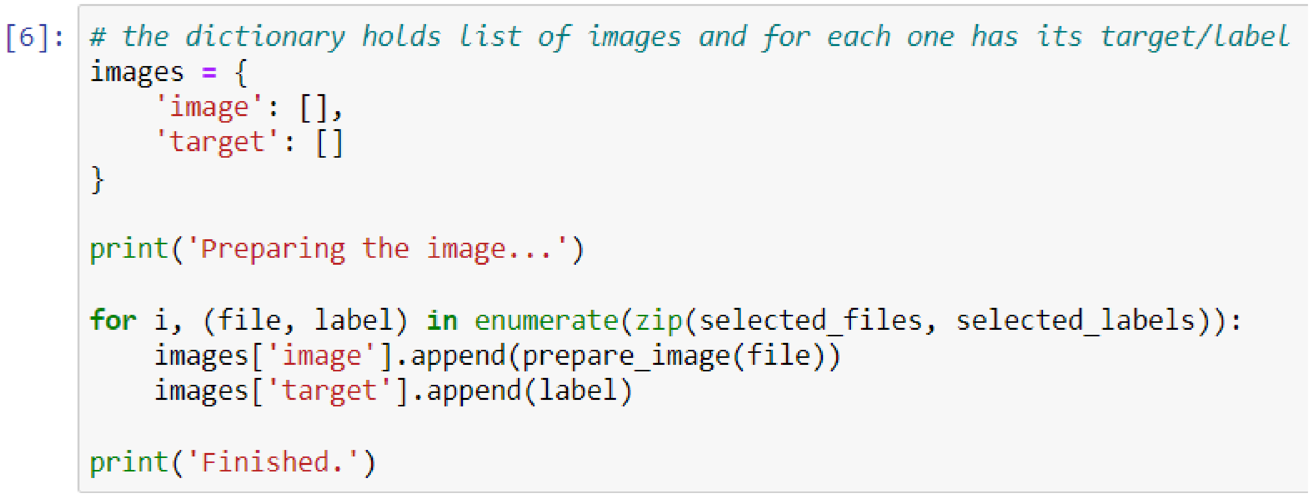
Fig.6.2.1 Importing libraries and dataset:



Importing Libraries

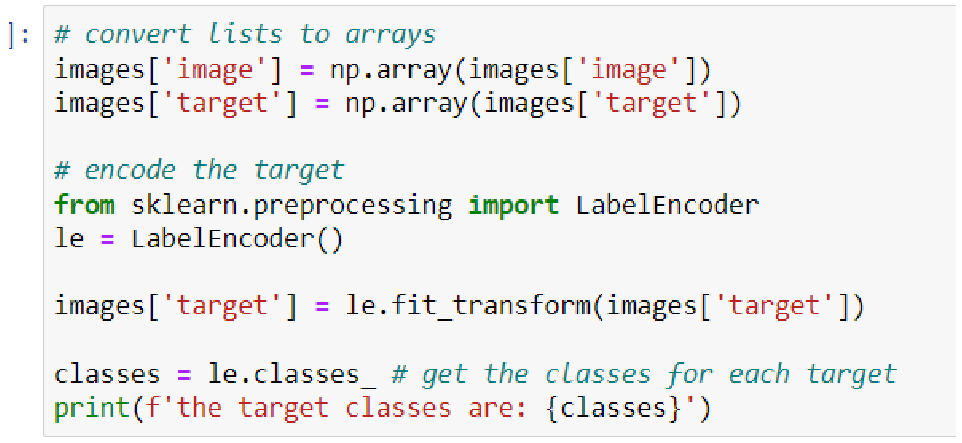
```
1]: import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    import glob, os

    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import layers, activations, optimizers, losses, metrics, initializers
    from tensorflow.keras.preprocessing import image, image_dataset_from_directory
    from tensorflow.keras.applications import MobileNetV3Small, MobileNet, InceptionV3
    from tensorflow.keras.applications.mobilenet_v3 import preprocess_input, decode_predictions
    from tensorflow.keras.applications import EfficientNetB7

    seed = 42
    tf.random.set_seed(seed)
    np.random.seed(seed)

2]: dir_path = "D:\#MINI PROJECT\Dataset_BUSI_with_GT"
    IMAGE_SHAPE = (224, 224)
```

Fig.6.2.2 Image Pre-processing:



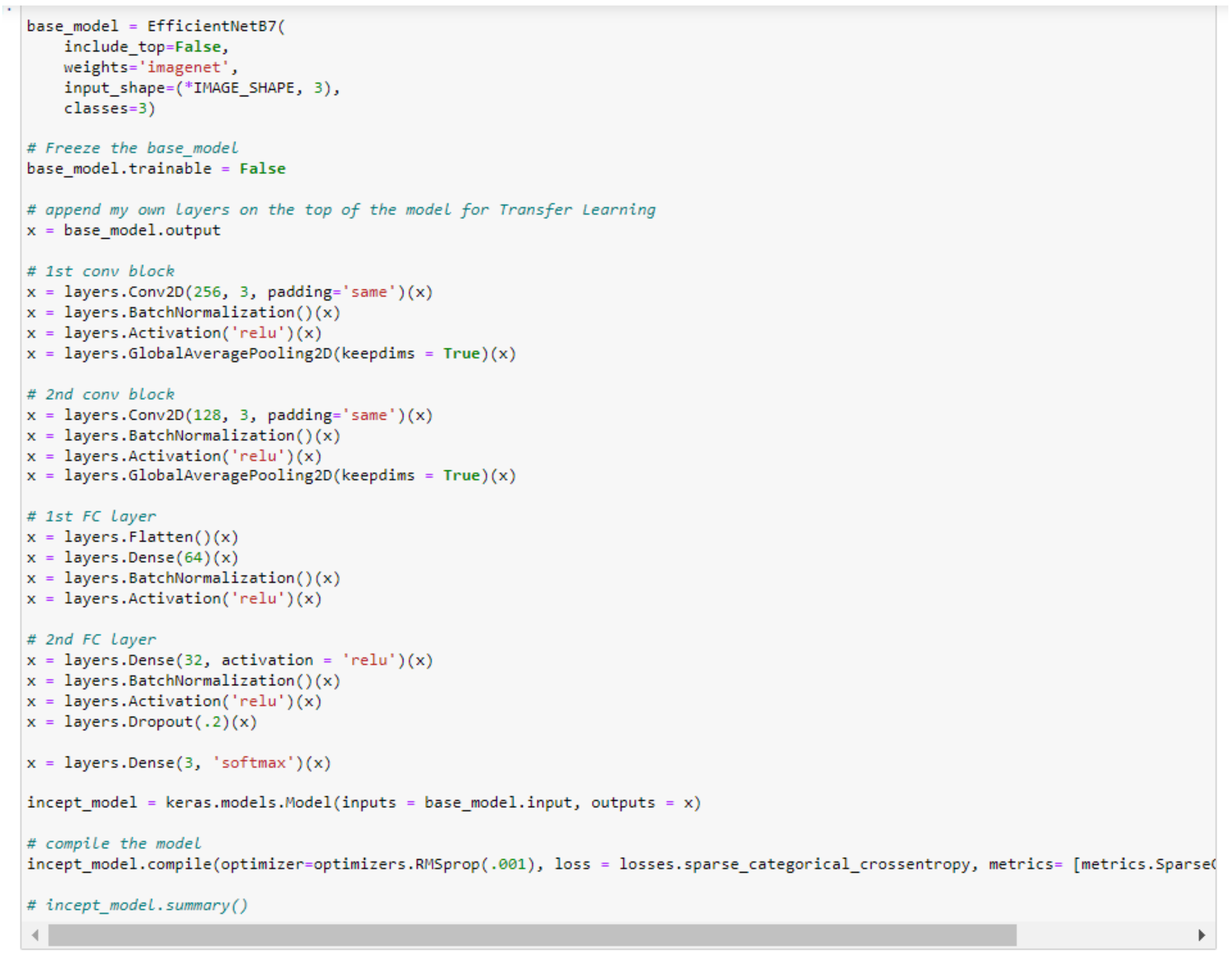
```
[6]: # the dictionary holds list of images and for each one has its target/label
     images = {
         'image': [],
         'target': []
     }

     print('Preparing the image...')

     for i, (file, label) in enumerate(zip(selected_files, selected_labels)):
         images['image'].append(prepare_image(file))
         images['target'].append(label)

     print('Finished.')
```

Fig.6.2.3 Identifying images and encoding class names:



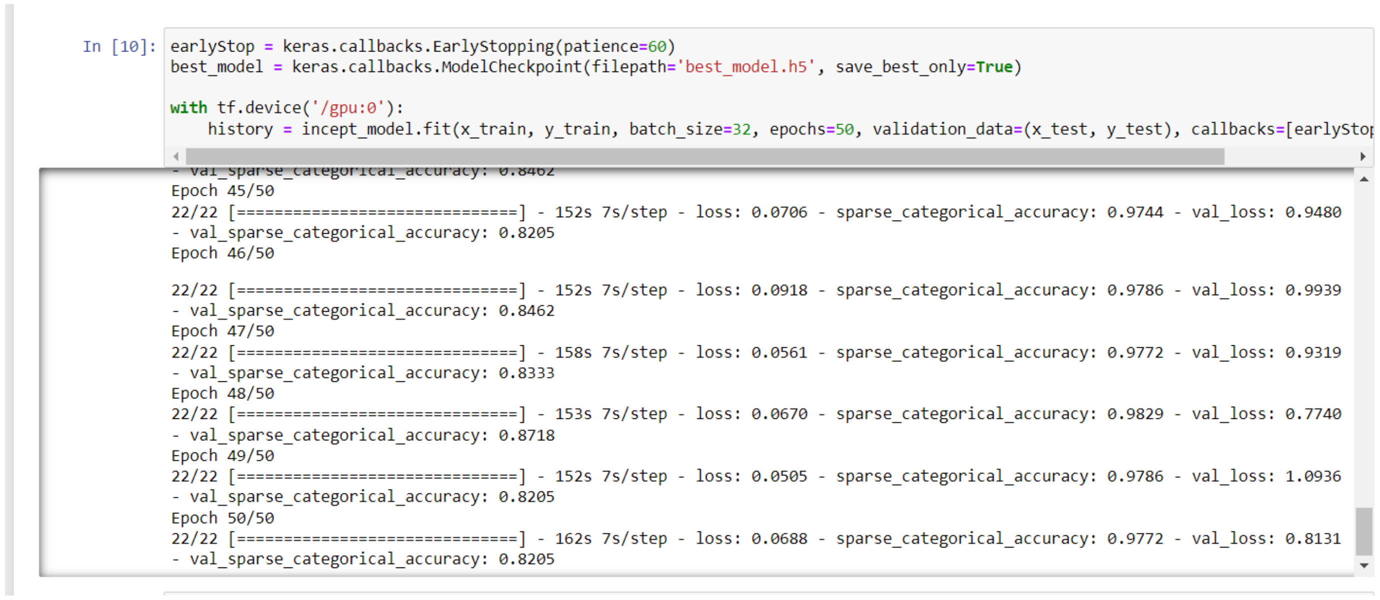
```
]: # convert lists to arrays
   images['image'] = np.array(images['image'])
   images['target'] = np.array(images['target'])

   # encode the target
   from sklearn.preprocessing import LabelEncoder
   le = LabelEncoder()

   images['target'] = le.fit_transform(images['target'])

   classes = le.classes_ # get the classes for each target
   print(f'the target classes are: {classes}')
```

Fig.6.2.3 Build the model :



```
base_model = EfficientNetB7(
    include_top=False,
    weights='imagenet',
    input_shape=(*IMAGE_SHAPE, 3),
    classes=3)

# Freeze the base_model
base_model.trainable = False

# append my own layers on the top of the model for Transfer Learning
x = base_model.output

# 1st conv block
x = layers.Conv2D(256, 3, padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.Activation('relu')(x)
x = layers.GlobalAveragePooling2D(keepdims = True)(x)

# 2nd conv block
x = layers.Conv2D(128, 3, padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.Activation('relu')(x)
x = layers.GlobalAveragePooling2D(keepdims = True)(x)

# 1st FC layer
x = layers.Flatten()(x)
x = layers.Dense(64)(x)
x = layers.BatchNormalization()(x)
x = layers.Activation('relu')(x)

# 2nd FC layer
x = layers.Dense(32, activation = 'relu')(x)
x = layers.BatchNormalization()(x)
x = layers.Activation('relu')(x)
x = layers.Dropout(.2)(x)

x = layers.Dense(3, 'softmax')(x)

incept_model = keras.models.Model(inputs = base_model.input, outputs = x)

# compile the model
incept_model.compile(optimizer=optimizers.RMSprop(.001), loss = losses.sparse_categorical_crossentropy, metrics= [metrics.Sparse(

# incept_model.summary()
```

Fig.6.2.4 Train the model :



```
In [10]: earlyStop = keras.callbacks.EarlyStopping(patience=60)
         best_model = keras.callbacks.ModelCheckpoint(filepath='best_model.h5', save_best_only=True)

         with tf.device('/gpu:0'):
             history = incept_model.fit(x_train, y_train, batch_size=32, epochs=50, validation_data=(x_test, y_test), callbacks=[earlyStop
```

```
- val_sparse_categorical_accuracy: 0.8462
Epoch 45/50
22/22 [==============================] - 152s 7s/step - loss: 0.0706 - sparse_categorical_accuracy: 0.9744 - val_loss: 0.9480
- val_sparse_categorical_accuracy: 0.8205
Epoch 46/50

22/22 [==============================] - 152s 7s/step - loss: 0.0918 - sparse_categorical_accuracy: 0.9786 - val_loss: 0.9939
- val_sparse_categorical_accuracy: 0.8462
Epoch 47/50
22/22 [==============================] - 158s 7s/step - loss: 0.0561 - sparse_categorical_accuracy: 0.9772 - val_loss: 0.9319
- val_sparse_categorical_accuracy: 0.8333
Epoch 48/50
22/22 [==============================] - 153s 7s/step - loss: 0.0670 - sparse_categorical_accuracy: 0.9829 - val_loss: 0.7740
- val_sparse_categorical_accuracy: 0.8718
Epoch 49/50
22/22 [==============================] - 152s 7s/step - loss: 0.0505 - sparse_categorical_accuracy: 0.9786 - val_loss: 1.0936
- val_sparse_categorical_accuracy: 0.8205
Epoch 50/50
22/22 [==============================] - 162s 7s/step - loss: 0.0688 - sparse_categorical_accuracy: 0.9772 - val_loss: 0.8131
- val_sparse_categorical_accuracy: 0.8205
```

Fig.6.2.4 plot for Loss, value loss, accuracy and value accuracy.

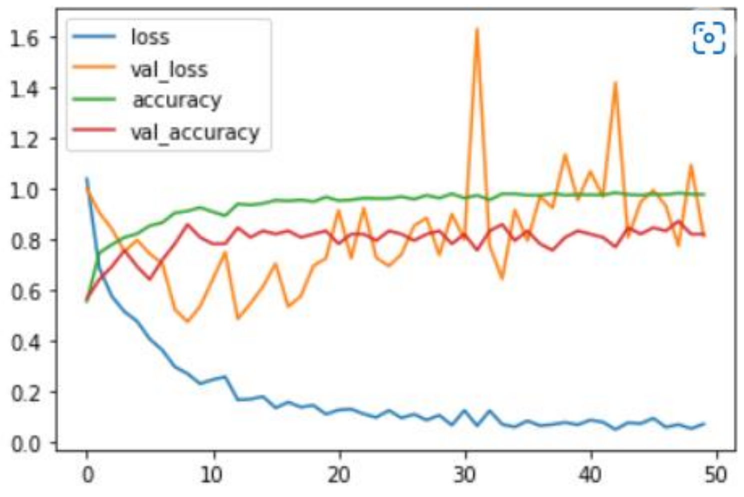
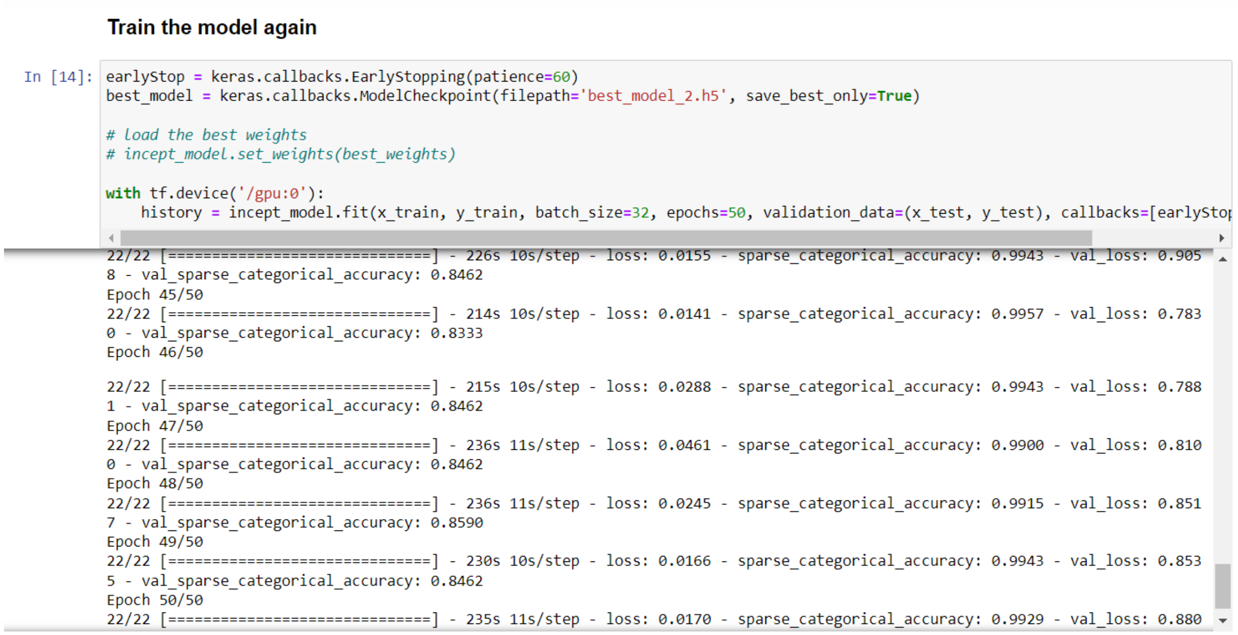


Fig.6.2.5 Training the Model again (Transfer Learning).

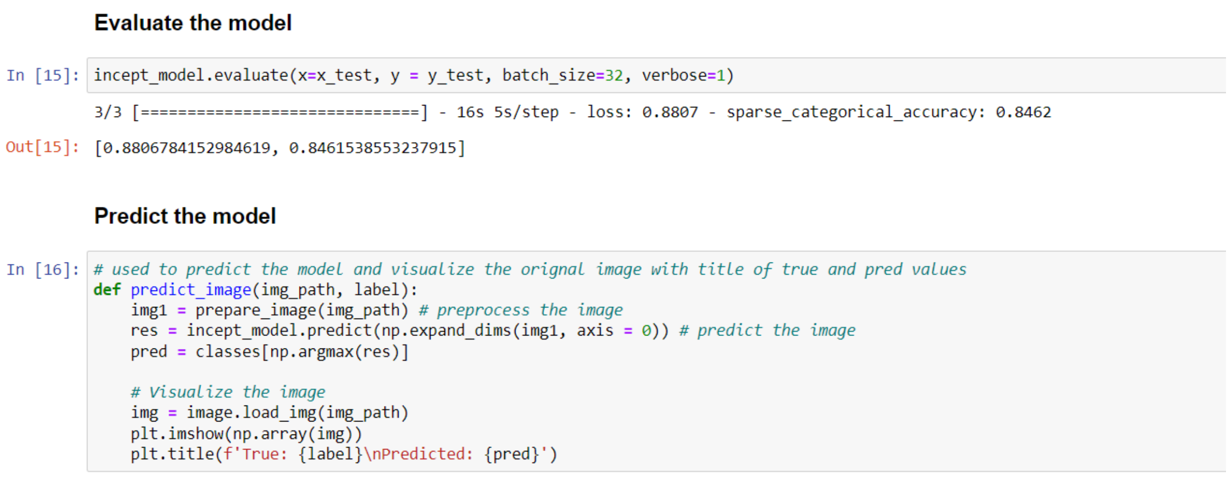


Train the model again

```
In [14]: earlyStop = keras.callbacks.EarlyStopping(patience=60)
         best_model = keras.callbacks.ModelCheckpoint(filepath='best_model_2.h5', save_best_only=True)

         # load the best weights
         # incept_model.set_weights(best_weights)

         with tf.device('/gpu:0'):
             history = incept_model.fit(x_train, y_train, batch_size=32, epochs=50, validation_data=(x_test, y_test), callbacks=[earlyStop
```

```
22/22 [==============================] - 226s 10s/step - loss: 0.0155 - sparse_categorical_accuracy: 0.9943 - val_loss: 0.905
8 - val_sparse_categorical_accuracy: 0.8462
Epoch 45/50
22/22 [==============================] - 214s 10s/step - loss: 0.0141 - sparse_categorical_accuracy: 0.9957 - val_loss: 0.783
0 - val_sparse_categorical_accuracy: 0.8333
Epoch 46/50

22/22 [==============================] - 215s 10s/step - loss: 0.0288 - sparse_categorical_accuracy: 0.9943 - val_loss: 0.788
1 - val_sparse_categorical_accuracy: 0.8462
Epoch 47/50
22/22 [==============================] - 236s 11s/step - loss: 0.0461 - sparse_categorical_accuracy: 0.9900 - val_loss: 0.810
0 - val_sparse_categorical_accuracy: 0.8462
Epoch 48/50
22/22 [==============================] - 236s 11s/step - loss: 0.0245 - sparse_categorical_accuracy: 0.9915 - val_loss: 0.851
7 - val_sparse_categorical_accuracy: 0.8590
Epoch 49/50
22/22 [==============================] - 230s 10s/step - loss: 0.0166 - sparse_categorical_accuracy: 0.9943 - val_loss: 0.853
5 - val_sparse_categorical_accuracy: 0.8462
Epoch 50/50
22/22 [==============================] - 235s 11s/step - loss: 0.0170 - sparse_categorical_accuracy: 0.9929 - val_loss: 0.880
```

Fig.6.2.6 Evaluate and predict the model.



Evaluate the model

```
In [15]: incept_model.evaluate(x=x_test, y = y_test, batch_size=32, verbose=1)

         3/3 [==============================] - 16s 5s/step - loss: 0.8807 - sparse_categorical_accuracy: 0.8462

Out[15]: [0.8806784152984619, 0.8461538553237915]
```

Predict the model

```
In [16]: # used to predict the model and visualize the orignal image with title of true and pred values
         def predict_image(img_path, label):
             img1 = prepare_image(img_path) # preprocess the image
             res = incept_model.predict(np.expand_dims(img1, axis = 0)) # predict the image
             pred = classes[np.argmax(res)]

             # Visualize the image
             img = image.load_img(img_path)
             plt.imshow(np.array(img))
             plt.title(f'True: {label}\nPredicted: {pred}')
```

Predicted samples.

```
In [17]: predict_image(dir_path + '/benign/benign (10).png', 'benign')
```

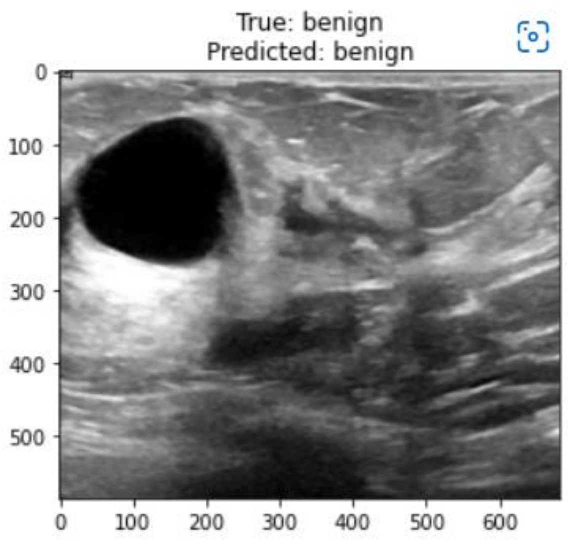


Fig.6.2.7

```
In [19]: predict_image(dir_path + '/malignant/malignant (10).png', 'malignant')
```

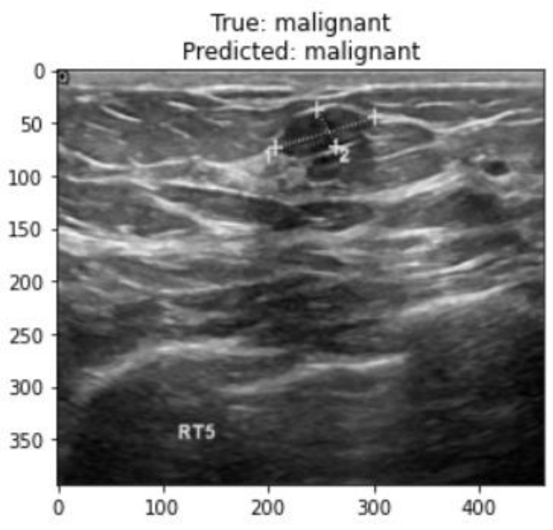


Fig.6.2.8

```
In [20]: predict_image(dir_path + '/normal/normal (10).png', 'normal')
```

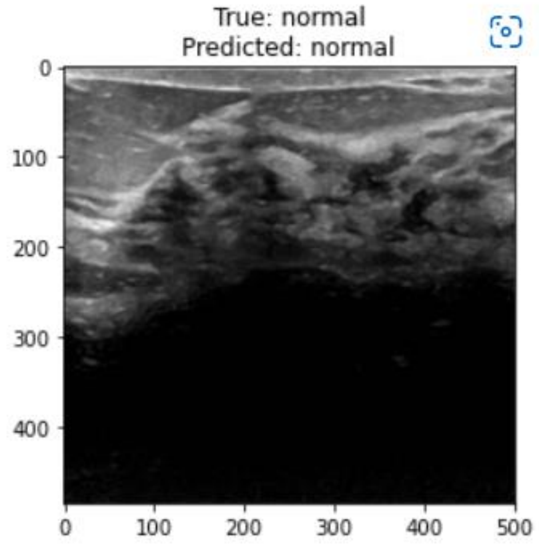


Fig.6.2.9

CHAPTER 7
CONCLUSION

7.1 INTRODUCTION:

This section gives the conclusion of the system, its applications in the real world.

7.2 CONCLUSION:

The project uses a deep learning technique convolutional neural system in that, a latest model of EfficientNet B7 which is trained of using the image datasets and another model of Artificial neural Network that is trained using the Wisconsin dataset. Both the models can be used for classification of breast cancer. The given datasets is properly pre-processed and used for training and testing the model.

7.3 APPLICATION:

The trained model can be used in the hospitals for diagnosing the patients whether they are affected with breast cancer or not. If affected what is the stage of the tumour either benign or Malignant, which will be helpful for the early diagnosis and could save lives of the people.

CHAPTER 8
REFERENCES

[1] Moh'd Rasoul Al-hadidi, Abdulsalam Alarabeyyat, Mohannad Alhanahnah Breast Cancer Detection using K-nearest Neighbor Machine Learning Algorithm 2016 9th International Conference on Developments in eSystems Engineering.

[2] Shajib Ghosh, Jubaer Hossain , Dr. Shaikh Anowarul Fattah, Dr. Celia Shahnaz , Asir Intisar Khan Efficient Approaches for Accuracy Improvement of Breast Cancer Classification Using Wisconsin Database 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC) 21 - 23 Dec 2017, Dhaka, Bangladesh.

[3] Naresh Khuriwal , Dr Nidhi Mishra **Breast Cancer Diagnosis Using Deep Learning Algorithm** International Conference on Advances in Computing, Communication Control and Networking (ICACCCN2018).

[4] Nur Syahmi Ismail , Cheab Sovuthy Breast Cancer Detection Based on Deep Learning Technique *Universiti Teknologi PETRONAS* Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia.

[5] Hari Krishna Timmana, Rajabhushanam C Breast Malignant Detection using Deep Learning Model Proceedings of the International Conference on Smart Electronics and Communication (ICOSEC 2020) IEEE Xplore Part Number: CFP20V90-ART; ISBN: 978-1-7281-5461-9.

[6] Ramik Rawal , BREAST CANCER PREDICTION USING MACHINE LEARNING 2020 JETIR May 2020, Volume 7, Issue 5.