

# Deep Learning Techniques for Breast Cancer Risk Prediction using IBM Cloud

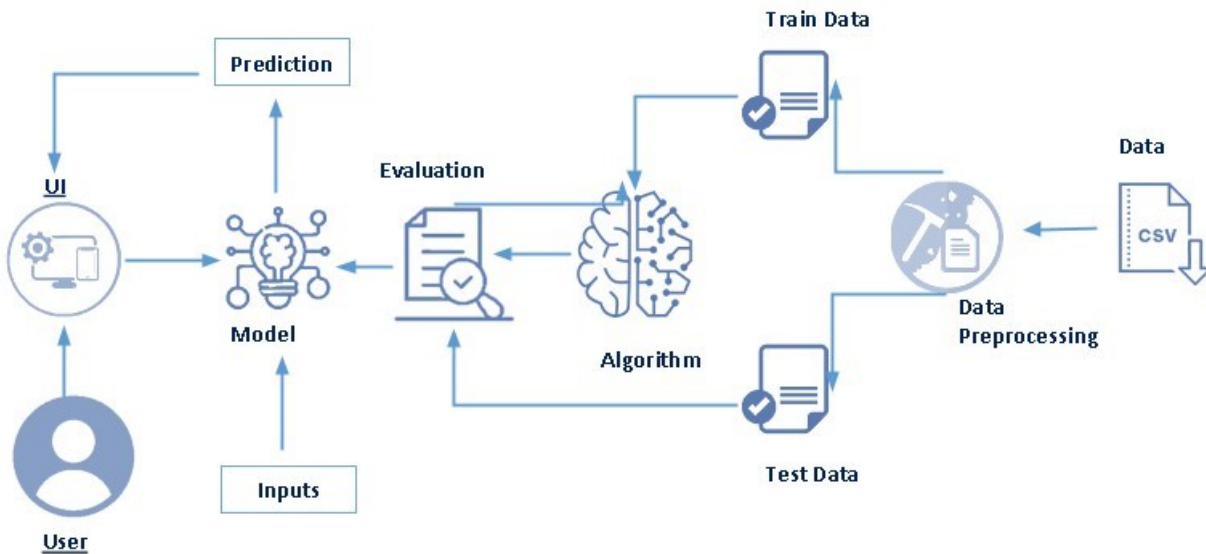
## Project Description:

Breast cancer is one of the main causes of cancer death worldwide. Computer-aided diagnosis systems showed the potential for improving diagnostic accuracy. But early detection and prevention can significantly reduce the chances of death. It is important to detect breast cancer as early as possible.

The goal is to classify images into two classifications of malignant and benign. As early diagnostics significantly increases the chances of correct treatment and survival. In this application, we are helping the doctors and patients to classify the Type of Tumour for the specific image given with the help of Neural Networks.

We will be using deep learning algorithm CNN, NumPy, TensorFlow, Keras, OpenCV and some other deep learning techniques. We will do Flask Integration and IBM Watson Deployment also.

## Technical Architecture:



## Pre-requisites:

To complete this project , you must required following software's, concepts and packages

### ► Anaconda navigator, PyCharm and Google Collab:

- \* Refer the link below to download Anaconda navigator and PyCharm
- \* Link: <https://www.anaconda.com/products/distribution>
- \* Link: <https://youtu.be/5mDYijMfSzs>
- \* Link: <https://colab.research.google.com/>

### ► Python Packages:

- \* Open anaconda prompt as a administrator
- \* Type "pip install numpy" and click enter.
- \* Type "pip install pandas" and click enter.
- \* Type "pip install matplotlib" and click enter.
- \* Type "pip install scikit-learn" and click enter.
- \* Type "pip install tensorflow=1.14.0" and click enter.
- \* Type "pip install keras=2.4.4" and click enter.
- \* Type "pip install opencv-python" and click enter.
- \* Type "pip install Flask" and click enter.
- \* Link: [https://youtu.be/akj3\\_wTploU](https://youtu.be/akj3_wTploU)

## Prior Knowledge:

You must have prior knowledge of the following topics to complete this project.

- **ML Concepts:**

- ♦ Supervised Learning:

- <https://www.javatpoint.com/supervised-machine-learning>

- [https://youtu.be/kE5QZ8G\\_78c](https://youtu.be/kE5QZ8G_78c)

- ♦ Unsupervised Learning:

- <https://www.javatpoint.com/unsupervised-machine-learning>

- [https://www.youtube.com/watch?v=kE5QZ8G\\_78c](https://www.youtube.com/watch?v=kE5QZ8G_78c)

- ♦ Regression, Classification and Clustering:

- [https://www.youtube.com/watch?v=6za9\\_mh3uTE](https://www.youtube.com/watch?v=6za9_mh3uTE)

- <https://www.geeksforgeeks.org/ml-classification-vs-clustering/>

- ♦ Artificial Neural Networks:

- <https://www.youtube.com/watch?v=DKSZHN7jftI>

- <https://www.javatpoint.com/artificial-neural-network>

- ♦ Convolution Neural Networks (CNN) :

- [https://www.youtube.com/watch?v=umGJ30-15\\_A](https://www.youtube.com/watch?v=umGJ30-15_A)

- <https://www.geeksforgeeks.org/introduction-convolution-neural-network>

- **Flask Basics:**

- [https://www.youtube.com/watch?v=lj4I\\_CvBnt0](https://www.youtube.com/watch?v=lj4I_CvBnt0)

## Project Objectives:

By the end of this project you will know:

- Preprocess the images.
- Applying the CNN algorithm on the dataset
- How deep neural networks are predicting the cancer is benign or malignant
- How to find the accuracy of the model.
- Able to build web applications using Flask framework.

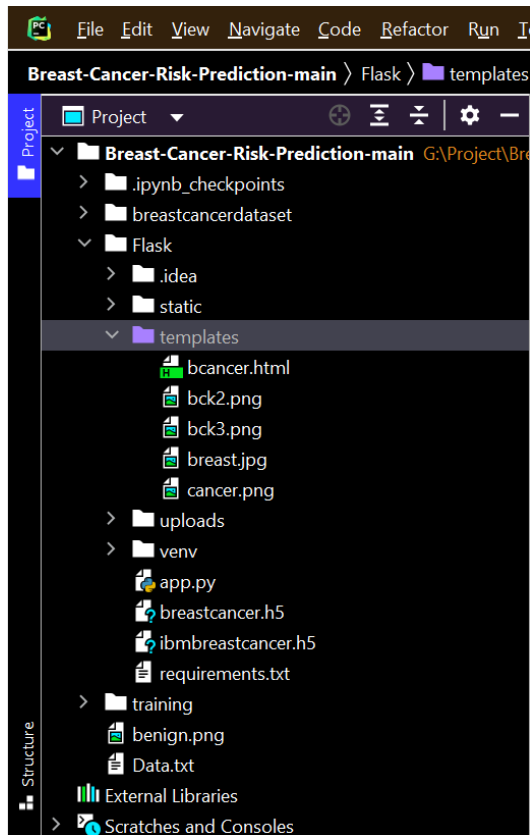
## Project Flow:

- Download the dataset.
- Classify the dataset into train and test sets.
- Add the neural network layers.
- Load the trained images and fit the model.

- Test the model
- Save the model and it's dependencies.
- Build a Web application using flask that integrates the model built.

## Project Structure:

Create the Project folder which contains files as shown below



- We are building flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- breastcancer.h5 is our saved model. Further we will use this model for flask integration.
- ibmbreastcancer.h5 model file used in the IBM Watson
- Training folder contains model training files and breastcancerdataset contains the training and testing datasets.

## Milestone 1: Data Collection

Create Train and Test folders with each folder having subfolders with images of breast cancer and who don't. You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc. The folder contains the datasets which can be used for training.

### Activity 1: Download the dataset

There are many popular open sources for collecting the data. Like kaggle.com, UCI repository e.t.c.

In this project we have used the dataset from kaggle.com. Please refer the link given below to download the dataset.

Link: <https://www.kaggle.com/datasets/paultimothymooney/breast-histopathology-images>

## Milestone 2: Image Preprocessing

Image Pre-processing includes the following main tasks

- Import ImageDataGenerator Library
- Configure ImageDataGenerator Class
- Applying ImageDataGenerator functionality to the training set and test set.

**Note:** The ImageDataGenerator accepts the original data, randomly transforms it , and return only the new transformed data.

### Activity 1: Import ImageDataGenerator Library and Configure it

ImageDataGenerator class is used to load the images with different modifications like considering the zoomed image, flipping the image and rescaling to a range of 0 and 1.

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

### Activity 2: Apply ImageDataGenerator functionality to training and testing dataset

Specify the path of both folders in the flow\_from\_directory method.

```
x_train = train_datagen.flow_from_directory(r"/content/drive/MyDrive/SmartBidge/Breast-Cancer-Risk-Prediction-main/breastcancerdataset/train", target_size=(180, 180))
x_test = test_datagen.flow_from_directory(r"/content/drive/MyDrive/SmartBidge/Breast-Cancer-Risk-Prediction-main/breastcancerdataset/test", target_size=(180, 180))

Found 103 images belonging to 2 classes.
Found 22 images belonging to 2 classes.
```

## Milestone 3: Model Building

The neural network model is to be built by adding different network layers like convolution, pooling, flattening, dropout and neural layers.

In this milestone, we start building our model by:

1. Initializing the mode
2. Adding Convolution layers
3. Adding Pooling layers
4. Flatten layer
5. Full connection layers which include hidden layers

At last, we compile the model with layers we added to complete the neural network structure.

### Activity 1: Import The Required Model Building Libraries

Import the libraries that are required to initialize the neural network layer, create and add different layers to the neural network layer model.

```
[ ] from keras.preprocessing.image import ImageDataGenerator
    from keras.models import Sequential
    from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
    from keras.layers import MaxPooling2D
```

### Activity 2: Initialize The Model

Initialize the neural network layer by creating a reference/object to the sequential class.

```
[ ] model=Sequential()
```

### Activity 3: Add The Convolution layer

The first layer of the neural network model, the convolution layer will be added. To create a convolution layer, Convolution 2D class is used. It takes a number of feature detectors, feature detector size, expected input shape of the image, activation function as arguments. This layer applies feature detectors on the input image and returns a feature map (features from the image).

#### Activation Function:

These are the functions that helps us to decide if we need to activate the node or not. These functions introduce non-linearity in the networks.

```
[ ] model.add(Conv2D(64,(3, 3),activation='relu', input_shape=(75, 75, 3)))
```

### Activity 4: Add The Pooling Layer

#### Max Pooling

selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

After the convolution layer, a pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. The efficient size of the pooling matrix is (2,2). It returns the pooled feature maps.

**Note:** Any number of convolution layers, pooling and dropout layers can be added.

```
[ ] model.add(MaxPooling2D(pool_size = (2,2)))
```

### Activity 5: Add The Flatten Layer

The flatten layer is used to convert n-dimensional arrays to 1-dimensional arrays. This 1D array will be given as input ANN layers.

```
[ ] model.add(Flatten())
```

### Activity 6: Adding The Dense Layers

Three dense layers are added which usually takes the number of units/neurons. Specifying the activation function, kind of weight initialization is optional.

```
[ ] model.add(Dense(units= 40 ,kernel_initializer='random_uniform',activation = 'relu'))  
  
[ ] model.add(Dense(units= 1,activation = 'softmax',kernel_initializer= 'uniform'))
```

**Note:** Any number of convolution, pooling and dense layers can be added according to the data.

### Activity 7: Compile The Model

After adding all the required layers, the model is to be compiled. For this step, loss function, optimizer and metrics for evaluation can be passed as arguments.



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

+ Code + Text

## Activity 8: Fit And Save The Model

Fit the neural network model with the train and test set , the number of epochs, and validation steps.

### Accuracy, Loss:

Loss value implies how poorly or well a model behaves after each iteration of optimization. An accuracy metric is used to measure the algorithm's performance in an interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage.

The weights are to be saved for future use. The weights are saved in as .h5 file using save()

```
[ ] model.save('breastcancer.h5')
```

## Milestone 4: Test The Model

The model is to be tested with different images to know if it is predicting correctly.

### Activity 1: Import The Packages And Load The Saved Model

Import the packages that are used to load the model and get the predictions.

```
[ ] from keras.models import load_model
    from keras.preprocessing import image
    import numpy as np
    from tensorflow.keras.models import load_model
```

```
▶ model = load_model("breastcancer.h5")
```

## Activity 2: Load The Test Image, Pre-Process It And Predict

Pre-processing the image includes converting the image to array and resizing according to the model. Give the pre-processed image to the model to know to which class your model belongs to.

```
[ ] img = image.load_img('/content/drive/MyDrive/SmartBidge/Breast-Cancer-Risk-Prediction-main/benign.png',target_size = (64,64))

[ ] x = image.img_to_array(img)
    x = np.expand_dims(x,axis = 0)

▶ pred = np.argmax(model.predict(x), axis=-1)
    #model.predict_classes(x)

[ ] pred
```

## Milestone 5: Application Building

After the model is built, we will be integrating it into a web application so that normal users can also use it. The users need to give the X-ray images to know the predictions.

### Activity 1: Build A Flask Application

**Step 1: Load the required packages.**

```

2
3     from __future__ import division, print_function
4     # coding=utf-8
5     import sys
6     import os
7     import glob
8     import numpy as np
9     from tensorflow.keras.preprocessing import image
10
11
12     from keras.applications.imagenet_utils import preprocess_input, decode_predictions
13
14     from keras.models import load_model
15     from keras import backend
16     from tensorflow.keras import backend
17     #from tensorflow.keras.models import Sequential, load_model
18
19     import tensorflow as tf

```

## Step 2: Initialize graph, load the model, initialize the flask app and load the model

Graph elements are required to work with tensorflow. So, graph elements are created explicitly. Instance of Flask is created and the model is loaded using load\_model from keras.

```

import tensorflow as tf

# global graph
graph=tf.compat.v1.get_default_graph()

from skimage.transform import resize

# Flask utils
from keras.saving.save import load_model
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer

# Define a flask app
app = Flask(__name__)

# Load your trained model
model = load_model("ibmbreastcancer.h5")
print('Model Loaded. Check http://127.0.0.1:5000/')

```

## Step 3: Configure the home page

```
model = load_model("ibmbreastcancer.h5")
print('Model Loaded. Check http://127.0.0.1:5000/')
```

```
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('bcancer.html')
```

#### Step 4: Pre-process the frame and run

Pre-process the captured frame and give it to the model for prediction. Based on the prediction the output text is generated and sent to the HTML to display.

```
48
49 @app.route('/predict', methods=['GET', 'POST'])
50 def upload():
51     if request.method == 'POST':
52         # Get the file from post request
53         f = request.files['image']
54
55         # Save the file to ./uploads
56         basepath = os.path.dirname(__file__)
57         file_path = os.path.join(
58             basepath, 'uploads', secure_filename(f.filename))
59         f.save(file_path)
60         img = image.load_img(file_path, target_size=(64, 64))
61
62         x = image.img_to_array(img)
63         x = np.expand_dims(x, axis=0)
64         # with graph.as_default():
65         preds = model.predict(x)
66         if preds[0][0]==0:
67             text = "The tumor is benign.. Need not worry!"
68         else:
69             text = "It is a malignant tumor... Please Consult Doctor"
70         text = text
71
72         # ImageNet Decode
73
74         return text
75
```

## Step 5: Run the flask application

Run the flask application using run method. By default, the flask runs on 5000 port. If the port is to be changed, an argument can be passed and the port can be modified.

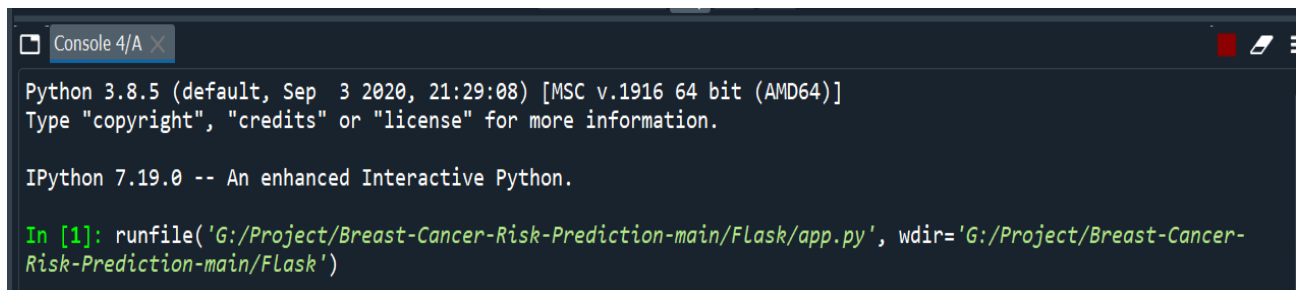
```
75
76 if __name__ == '__main__':
77     app.run(debug=False, threaded = False)
78
```

## Activity 2: Build The HTML Page And Execute

Build an HTML page to take a microscopic image as an input and display the output that is passed from the flask app. You can use project files for building a web page.

### Step 1: Run the application

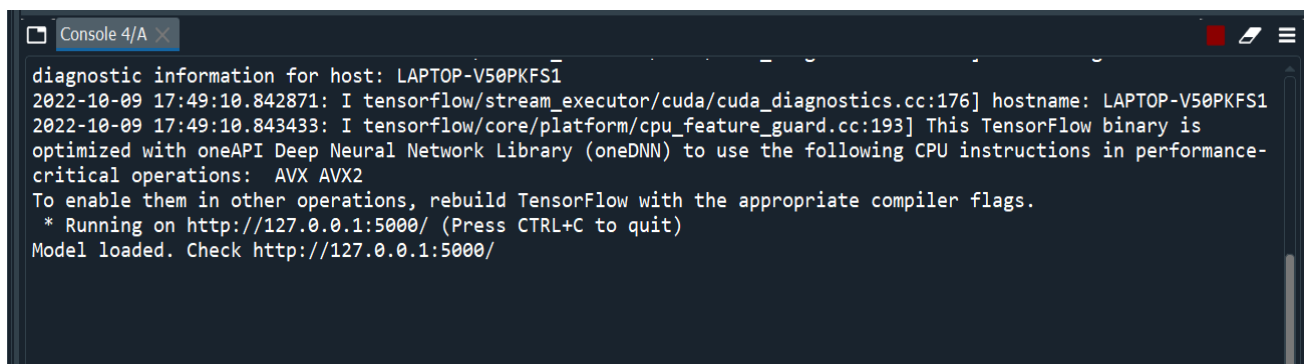
In anaconda prompt, navigate to the folder in which the flask app is present. When the python file is executed the localhost is activated on 5000 port and can be accessed through it



```
Console 4/A X
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.

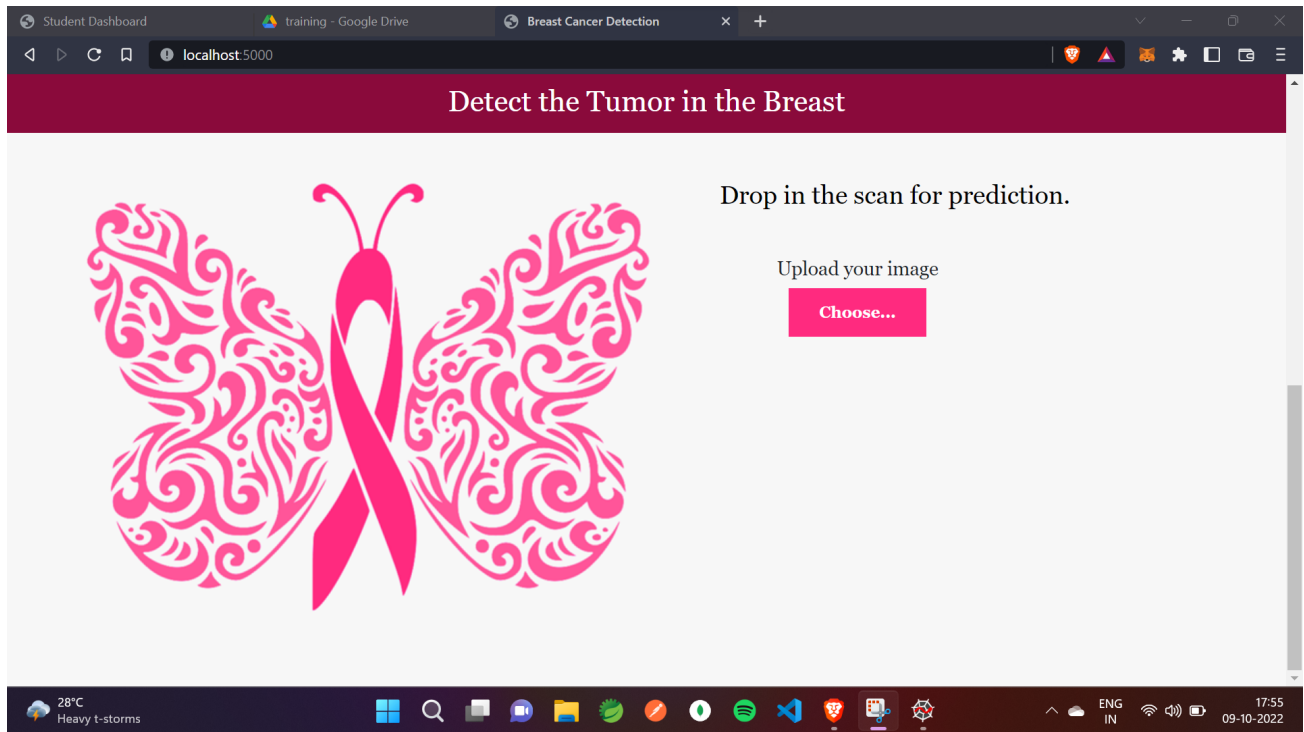
In [1]: runfile('G:/Project/Breast-Cancer-Risk-Prediction-main/Flask/app.py', wdir='G:/Project/Breast-Cancer-Risk-Prediction-main/Flask')
```



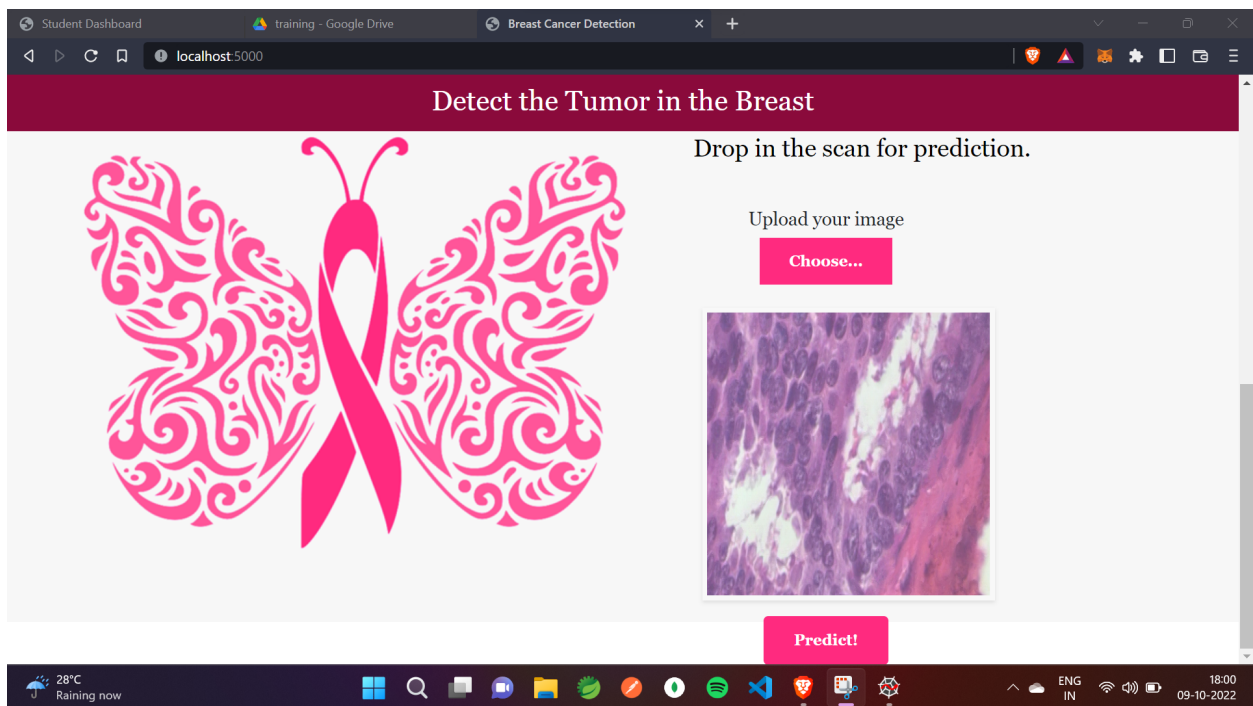
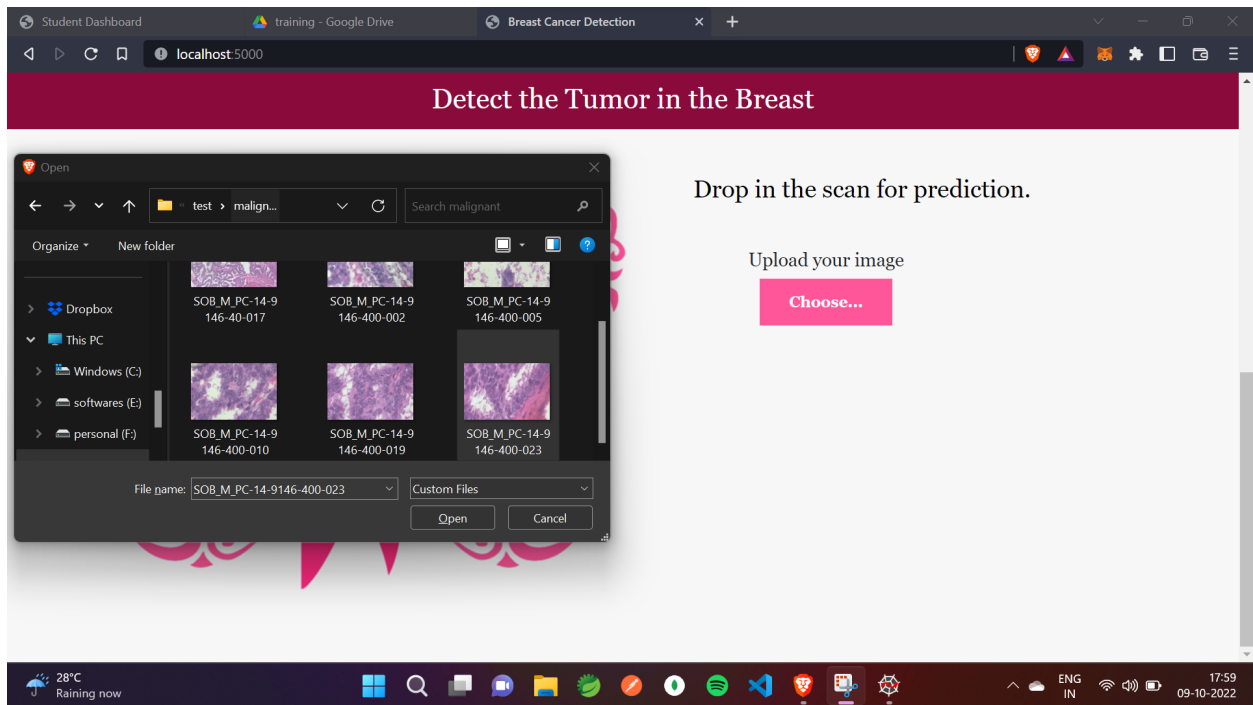
```
Console 4/A X
diagnostic information for host: LAPTOP-V50PKFS1
2022-10-09 17:49:10.842871: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: LAPTOP-V50PKFS1
2022-10-09 17:49:10.843433: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is
optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-
critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Model loaded. Check http://127.0.0.1:5000/
```

## Step 2: Open the browser and navigate to localhost:5000 to check your application

The home page looks like this. When you click on the button "Drop the scan " you will be able to predict the section.



In this section, you can browse and choose the image you want to predict and then click on the predict to get the predictions.




Student Dashboard

training - Google Drive

Breast Cancer Detection

localhost:5000

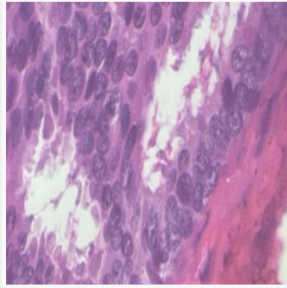
Detect the Tumor in the Breast



Drop in the scan for prediction.

Upload your image

Choose...



Prediction: It is a malignant tumor... Please Consult Doctor

28°C

Raining now

Windows

Search

Task View

Microsoft Edge

File Explorer

Google Chrome

Spotify

Visual Studio Code

Discord

Telegram

WhatsApp

Signal

Zoom

Skype

Slack

Telegram

WhatsApp

Signal

Zoom

Skype

Slack

ENG

IN

18:01

09-10-2022