EAST WEST UNIVERSITY

Project Report

Breast Cancer Classification with Neural Network

Course Code: CSE366

Course Title: Artificial Intelligence

Section No: 01

Semester: Summer'22

Submitted To

Jesan Ahammed Ovi

Senior Lecturer

Department of CSE, East West University

Submitted By

Name: Shafia Hasnin

ID: 2020-1-60-209

Name: Zarin Tasnim Nuzhat

ID: 2020-1-60-211

Date of Submission: 13-09-2022

Problem statement:

In this project we are going to train and build a predictive system to classify breast cancer. We are going to use a dataset of breast cancer as training dataset and the system will predict whether the cancer is Malignant or Benign based on the given information.

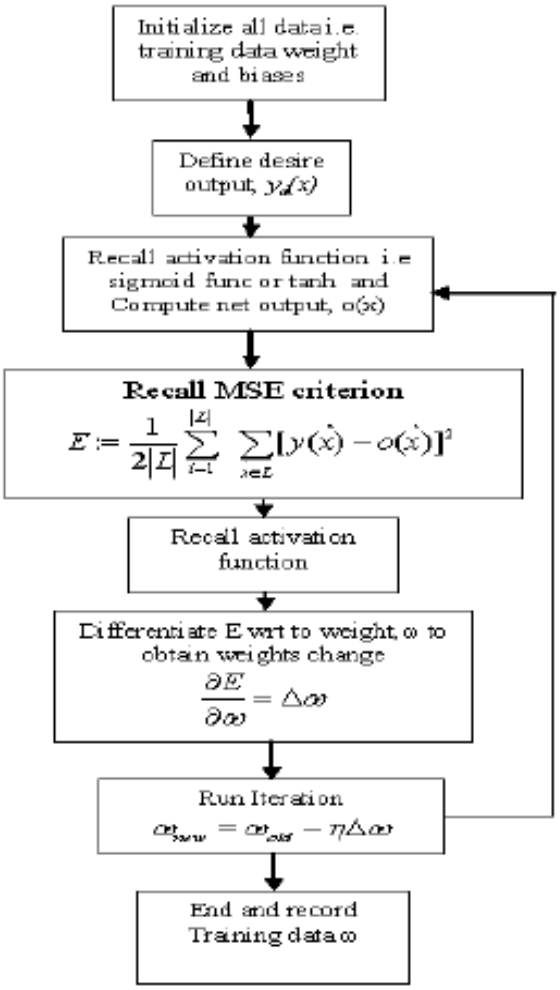
System Requirements:

1. Processor: Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.20 GHz
2. RAM: 4GB
3. Operating System: Windows
4. IDE: Google Colaboratory

System Design:

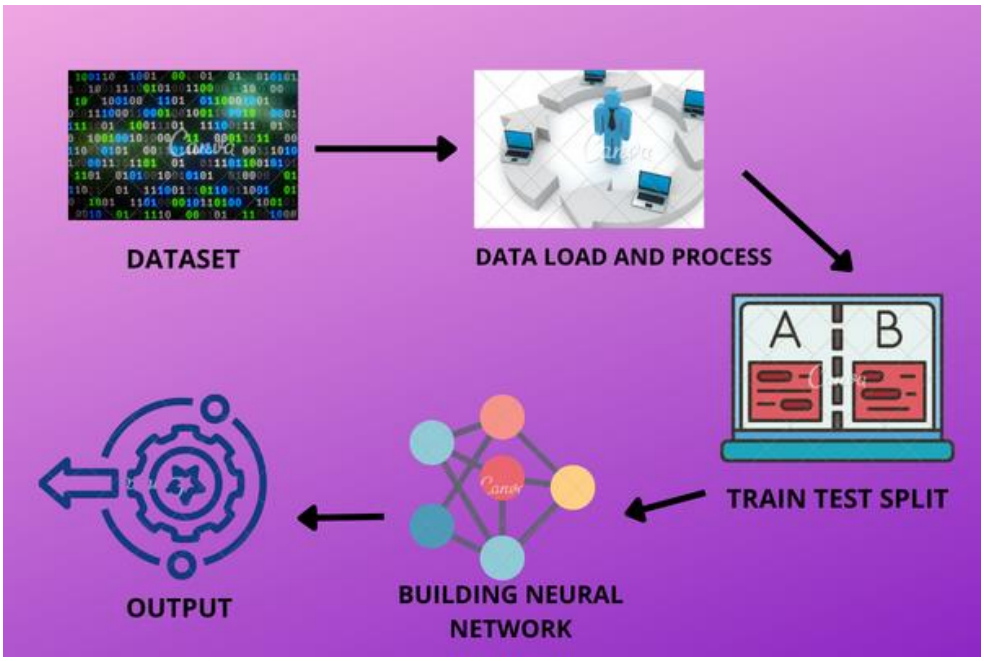
Algorithm:

We have used **Adam** (adaptive moment estimation*)* optimization algorithm. The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.



Architecture of the project:

First we import and load dataset from sklearn dataset to train our system. Then after training and testing we get our predicted output.



Implementation:

Importing the necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
tf.random.set_seed(3)
from tensorflow import keras
```

Loading the data from sklearn datasets

```
breast_cancer_dataset = sklearn.datasets.load_breast_cancer()
```

Building the Neural Network

```
#setting up the layers of the Neural Network

model = keras.Sequential([
                          keras.layers.Flatten(input_shape=(30,)),
                          keras.layers.Dense(20, activation='relu'),
                          keras.layers.Dense(2, activation='sigmoid')
])
                        ])
```

Here we have used ReLU (Rectified Linear Unit) activation function for our inner/hidden layers and Sigmoid activation function for output layer.

The main advantage of using the ReLU function over other activation functions is that the neurons will only be deactivated if the output of the linear transformation is less than 0.

Sigmoid activation function transforms the values between the range 0 and 1 as we have reshaped the trained and tested data between 0 and 1.

Compiling the Neural Network

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

Separating the features and target

```
X = data_frame.drop(columns='label', axis=1)
Y = data_frame['label']
```

Splitting the data into training data and testing data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, r
andom_state=2)
```

Standardizing the data

```
scaler = StandardScaler()

X_train_std = scaler.fit_transform(X_train)
```

```
X_test_std = scaler.transform(X_test)
```

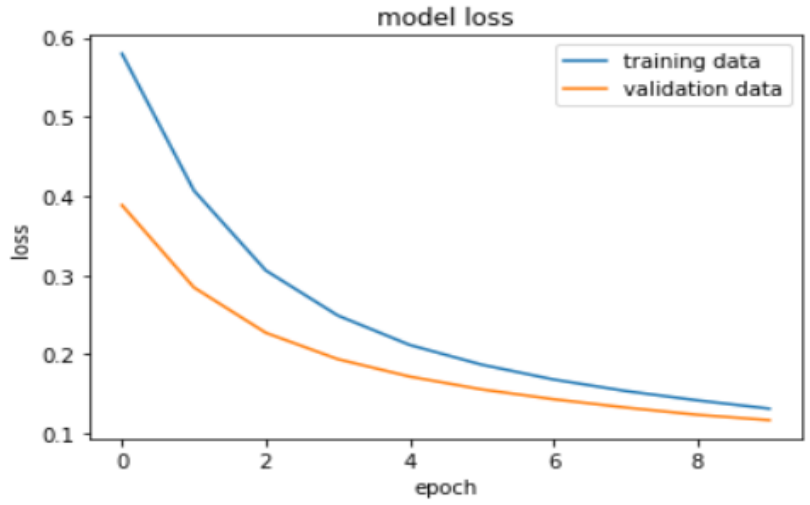
Training the Neural Network

```
history = model.fit(X_train_std, Y_train, validation_split=0.1, epochs=10)
```

Visualizing Accuracy and Loss

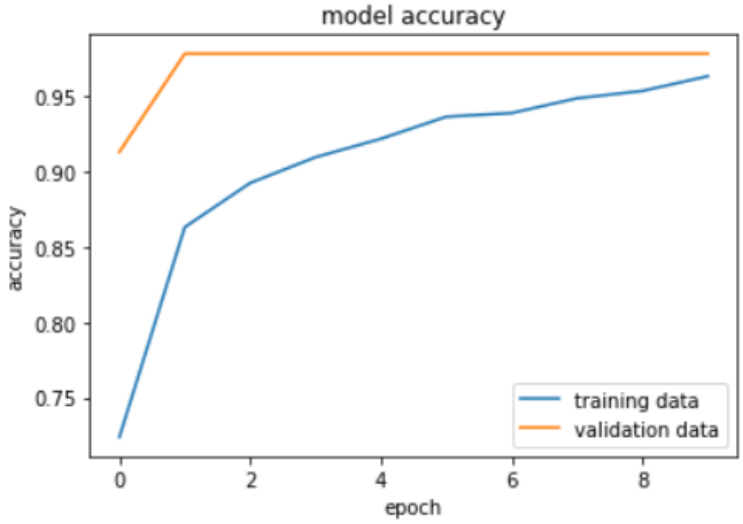
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')

plt.legend(['training data', 'validation data'], loc = 'upper right')
```

```
<matplotlib.legend.Legend at 0x7f7c4eefff90>
```



```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')

plt.legend(['training data', 'validation data'], loc = 'lower right')
```

```
<matplotlib.legend.Legend at 0x7f7c4a735350>
```



Converting the prediction probability to class labels

```
Y_pred_labels = [np.argmax(i) for i in Y_pred]
print(Y_pred_labels)
```

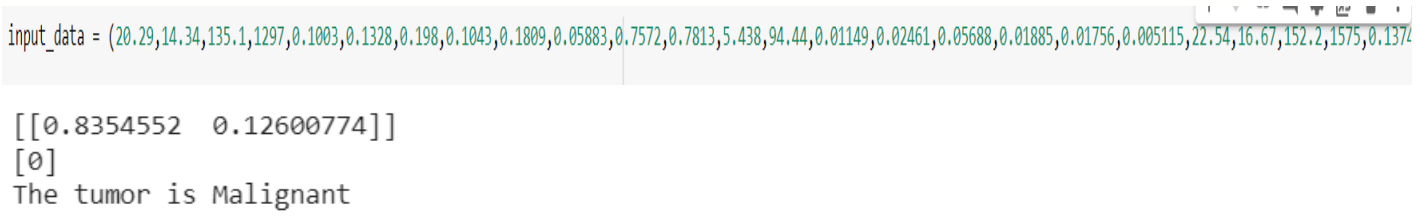
Predictive System

```
input_data = ()

# change the input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array as we are predicting for one data point
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardizing the input data
input_data_std = scaler.transform(input_data_reshaped)

prediction = model.predict(input_data_std)
```

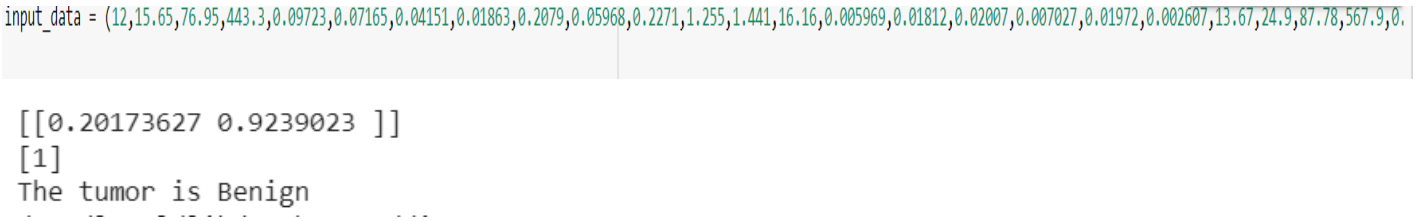
```
print(prediction)

prediction_label = [np.argmax(prediction)]
print(prediction_label)

if(prediction_label[0] == 0):
  print('The tumor is Malignant')

else:
  print('The tumor is Benign')
```

Here after processing and standardizing the train and test data we send it to our predictive system and the system gives the prediction output.

Testing Results:



Here, we sent the column values of a Malignant tumor and the system predicted it correctly.



Here, we sent the column values of a Benign tumor and the system predicted it correctly.

Future Scope:

Limitations:

- We used a limited set of data as training dataset.

Future Scope:

- This predictive system can be trained with a large set of dataset for better performance.