Analysis of the Wisconsin Breast Cancer Dataset and Deep Learning for Breast Cancer Detection

Aishwarya Kamble

Aishwaryakamble217@gmail.com

ABSTRACT

Deep Learning techniques are compared in this report. These methods are used to create two classifiers that must discriminate benign from malignant breast lumps. To create the classifier, the WBCD (Wisconsin Breast Cancer Diagnosis) dataset is employed. Early detection and treatment can allow patients to have proper treatment and consequently reduce rate of morbidity of breast cancer.

In this report, we present the most recent breast cancer detection and classification models that are deep learning based models by analyzing them in the form of comparative study.

The best accuracy in this report was achieved by the sequential algorithm I.e. 96% of accuracy.

Throughout this paper, the expression “False-Negative” is used to name the instances that were classified as Benign but in reality are malignant, and “False-Positive” is for the instances misclassified as Malignant.

OBJECTIVE

This study aims to establish a deep learning model to classify patients in categories Benign and Malignant respectively.

INTRODUCTION

Cancer arises when the abnormal body’s cells start to separate and come in contact with normal cells and make them malignant. Breast cancer is most frequently occurring and harmful disease in the world. Breast cancer considered either invasive or non-invasive. Invasive is cancerous, malignant and spreads in other organs. Non-invasive is pre-cancerous, remains in its original organ. It eventually develops into invasive breast cancer. The portion of body that contains the breast cancer is glands and milk ducts that carry the milk. Breast cancer spread to other organs frequently and make them malignant. It also spreads through the bloodstream to other organ.

Building a classifier using deep learning can be a challenging task if the dataset used is not on its best format or if it is not being correctly interpreted. Therefore, a considerable portion of this work will be spent preparing and comprehending the dataset to avoid problems such as overfitting.

DATASET

The Dataset used for the analysis is downloaded from UCI machine learning repository maintained by the University of California, Irvine. The dataset contains 569 samples of malignant and benign tumor cells.

The first two columns in the dataset store the unique ID numbers of the samples and the corresponding diagnosis (M=malignant, B=benign), respectively.

The columns 3-32 contain 30 real-value features that have been computed from digitized images of the cell nuclei, which can be used to build a model to predict whether a tumor is benign or malignant.

1= Malignant (Cancerous) - Present (M)

0= Benign (Not Cancerous) -Absent (B)

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

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- Perimeter
- Area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- Symmetry
- fractal dimension ("coastline approximation" - 1)

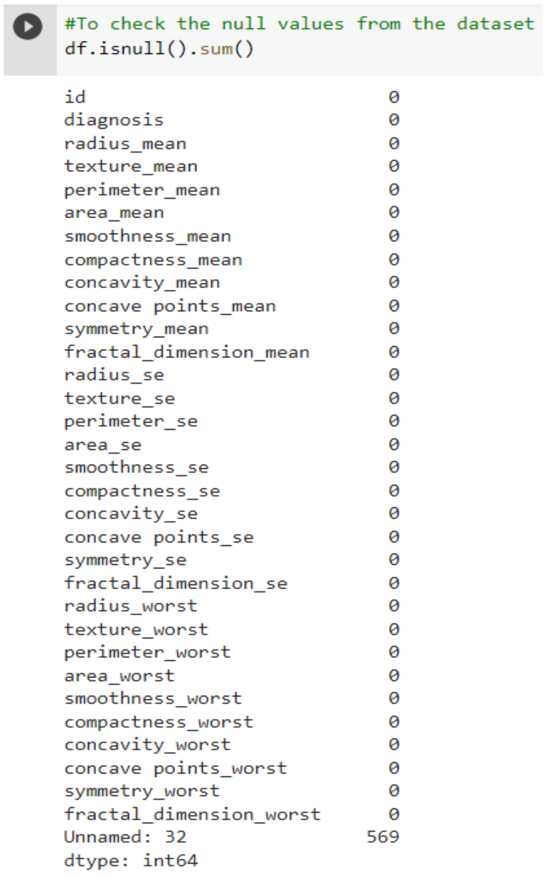
The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

Methodology Used

The process of solving this case study was modeled in steps to obtain the desired output.

Organizing input data:

The dataset which we have consists 33 columns, 569 entries I.e. rows. Firstly, we will check whether the information have null values or not and we found out that there are no null values in our dataset.


```
#To check the null values from the dataset
df.isnull().sum()
```

```
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave points_mean         0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
fractal_dimension_worst     0
Unnamed: 32               569
dtype: int64
```

Data Pre-Processing and EDA:

Data preprocessing in refers to the technique of

preparing (cleaning and organizing) the raw data to make it suitable for

building and training models.

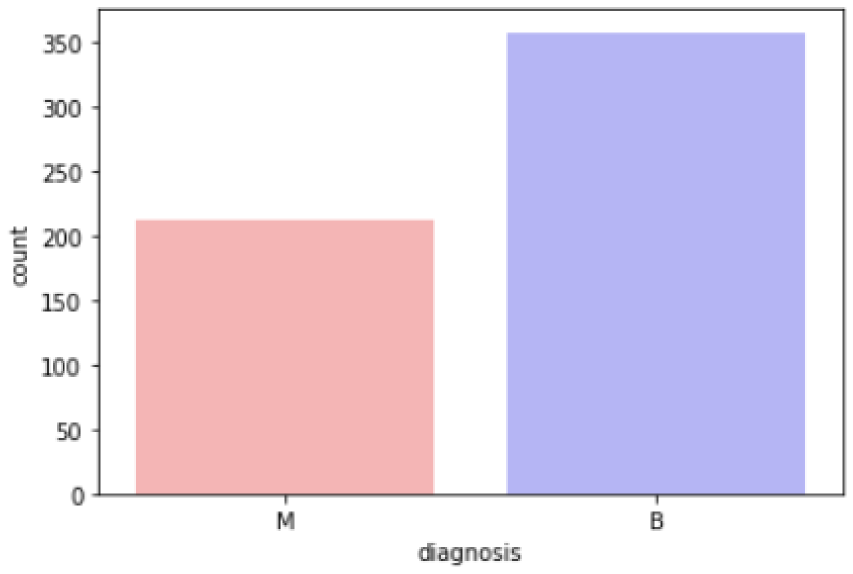
Data Cleaning :

- From the dataset we need to drop "id" column because I find the relationship between id and the cancer cell detection are least compatible with each other.

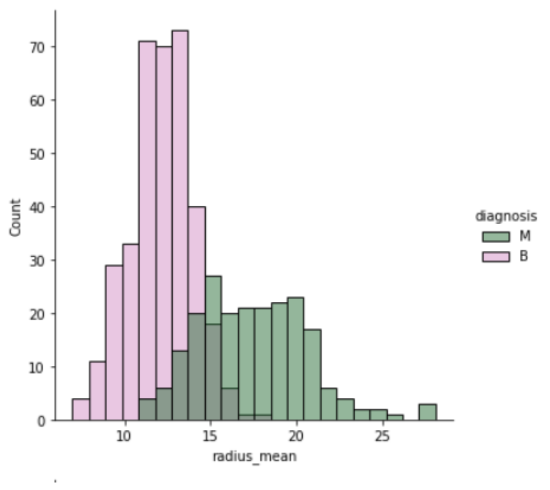
- We need to drop “Unnamed :32” column because it contains maximum null values.

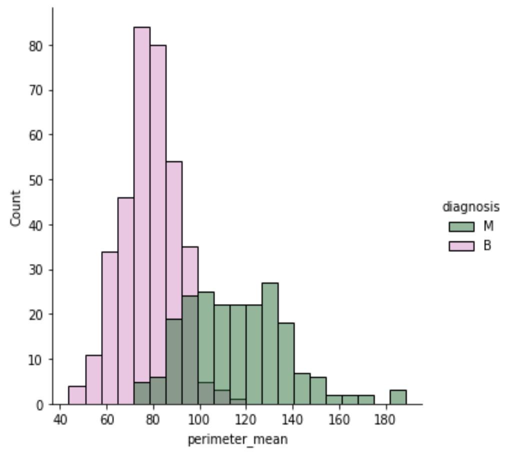
Visual representation of the target column, Show the counts of observations in each categorical bin using bars just to check whether the data is imbalanced or not and I got to know that data is balanced.

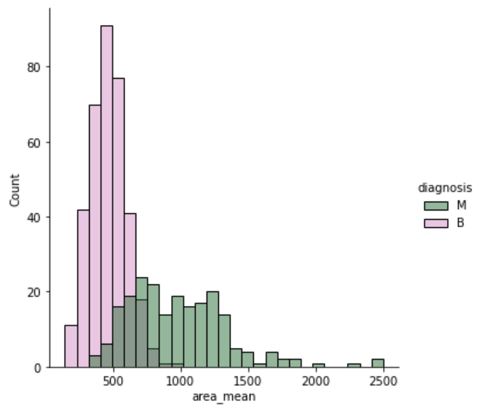
```
Number of Benign:  357
Number of Malignant :  212
```

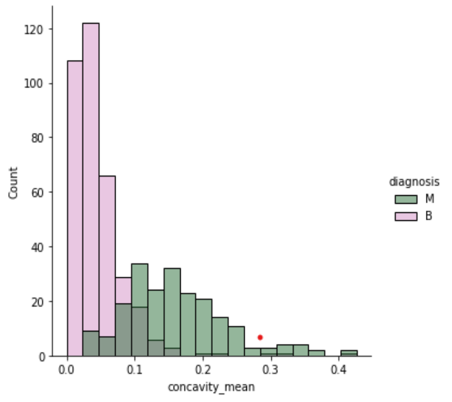


I have used distplot which represents the overall distribution of continuous data variables and from the representation I got to know that lower the values of radius mean, perimeter mean, area mean lower the chances of tumor and higher the values of concavity mean, perimeter_se , area mean higher the chances of tumor.









I have plotted heatmap which contains values representing various shades of the same colour for each value to be plotted. Usually, the darker shades of the chart represent higher values than the lighter shade for a quite different value a completely distinct colour can also be used.

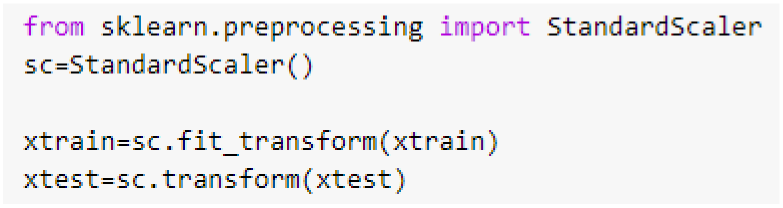
<u>**Data Sampling:**</u>

First, we need to divide the dataset target and feature(x, y) as main dataset being a one-to-one mapping between input variables to the target output classification or value.

In the data-sampling step, a subset of data is extracted from the input dataset. This is performed for sampling and for dividing the data into the two classes of training and testing data. I have kept test size is 0.30 I.e. I am sending 30% of the data for testing and remaining 70% data for training.

Data Scaling:

When we create the artificial neural network then we must scale the data into smaller numbers because the deep learning algorithms multiplies the weights and input data of the nodes and takes lots of time, so for reducing that time we must scale the data. For scaling i have used standard scaler, we scale the training and testing dataset. The values of different features have different ranges and majority of those values are outside the range of [0, 1]. However, the new deep learning model requires the input values in the range of [0, 1] for better performance. To this end, the values of the dataset are scaled into the range of [0, 1] as follows:

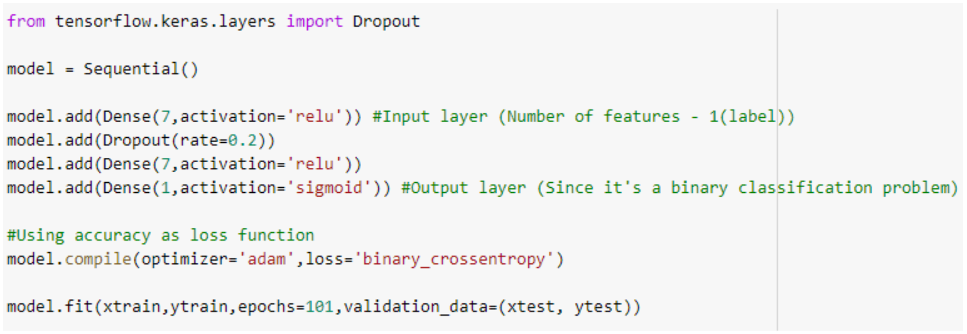


```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()

xtrain=sc.fit_transform(xtrain)
xtest=sc.transform(xtest)
```

Implementing the Deep Learning Model:

The popular open source Keras deep learning library is utilized for the implementation of the deep learning model.



```
from tensorflow.keras.layers import Dropout

model = Sequential()

model.add(Dense(7,activation='relu')) #Input layer (Number of features - 1(label))
model.add(Dropout(rate=0.2))
model.add(Dense(7,activation='relu'))
model.add(Dense(1,activation='sigmoid')) #Output layer (Since it's a binary classification problem)

#Using accuracy as loss function
model.compile(optimizer='adam',loss='binary_crossentropy')

model.fit(xtrain,ytrain,epochs=101,validation_data=(xtest, ytest))
```

This dataset contains 569 samples. In this case, using the one hidden layers of deep learning network is appropriate to get the best performance.

Activation Function:

Activation function defines the output of input or set of inputs or in other terms defines node of the output of node that is given in inputs.

They decide to activate or deactivate neurons to get the desired output. It also performs a nonlinear transformation on the input to get better results on a complex neural network.

In this dataset I have used 2 activation functions:

ReLU: Rectified linear unit or ReLU is most widely used activation function right now which ranges from 0 to infinity, all the negative values are converted into zero, and this conversion rate is so fast that neither it can map nor fit into data properly which creates a problem, but where there is a problem there is a solution.

We should not use ReLu in the output layer because it does not allow negative output values.

Sigmoid: It is commonly used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.

DROPOUTS:

Dropout means that those neurons, which are selected at random, are ignored by the units (i.e., neurons). By 'not knowing' this means that during a certain forward or backward pass these units are not considered.

The challenge for software-based neural networks is they must find ways to reduce the noise of billions of neuron nodes communicating with each other, so the networks' processing capabilities are not overrun. To do this, a network eliminates all communications that are transmitted by its neuron nodes not related to the problem or training that it is working on. The term for this neuron node elimination is dropout.

Here I have kept dropout rate is 0.2 I.e., 20% of the neurons will get dropped.

Optimizers:

An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improve the accuracy.

In this dataset I have used adam optimizer because adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

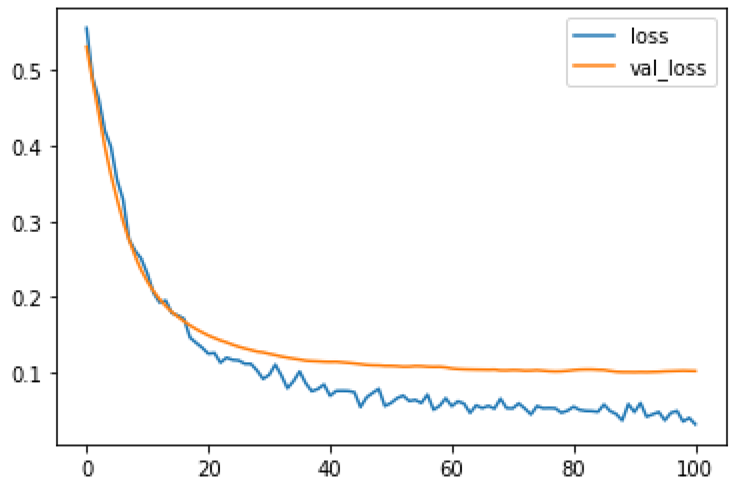
Epochs:

An epoch means training the neural network with all the training data for one cycle. In an epoch, we use all of the data exactly once. A forward pass and a backward pass together are counted as one pass.

Here I have passed 101 Epochs.

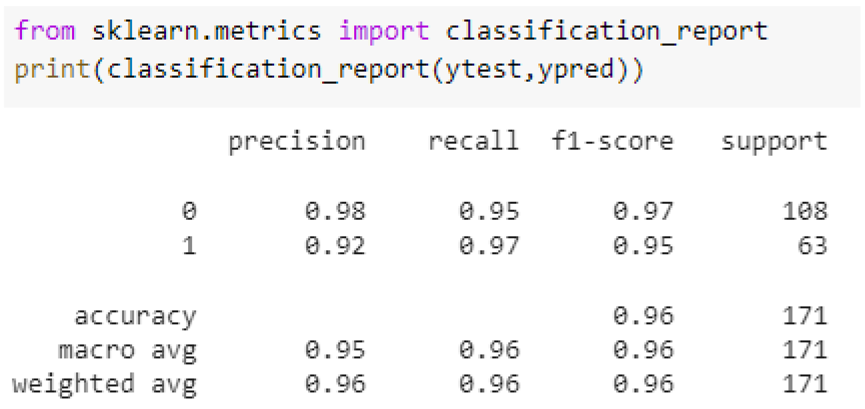
Training and Validation loss visual representation:

The purpose of this is to diagnose the model's performance and identify which aspects need tuning.



Classification Report

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below



```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.97       108
           1       0.92      0.97      0.95        63

    accuracy                           0.96       171
   macro avg       0.95      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```

Precision – What percent of your predictions were correct?

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives.

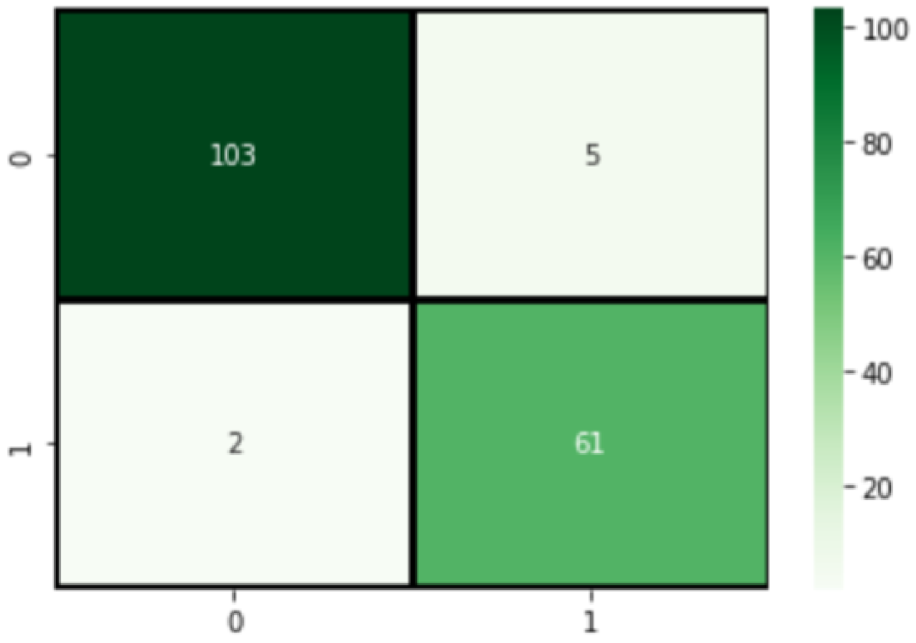
Recall – What percent of the positive cases did you catch?

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

F1 score – What percent of positive predictions were correct?

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

Confusion Matrix



TN / True Negative: when a case was negative and predicted negative(103)

TP / True Positive: when a case was positive and predicted positive(61)

FN / False Negative: when a case was positive but predicted negative(2)

FP / False Positive: when a case was negative but predicted positive(5)

Conclusion:

Breast cancer is considered to be one of the significant causes of death in women. Early detection of breast cancer plays an essential role to save women's life. Breast cancer detection can be done with the help of deep learning algorithms. In this report, we focus on how to deal with the FN and FP i.e our model must have low value for FN because they actually have cancer but model detected them as negative in order to do that, we have decreased the threshold to 20% in result the recall gets higher and the value of FN will get lower to enhance the classification accuracy of detecting breast cancer.