Predicting Breast Cancer using Logistic Regression

Meghana Vasudeva
Person#: 50290586
University at Buffalo,
The State University of New York Buffalo,
New York 14260
mvasudev@buffalo.edu

Abstract

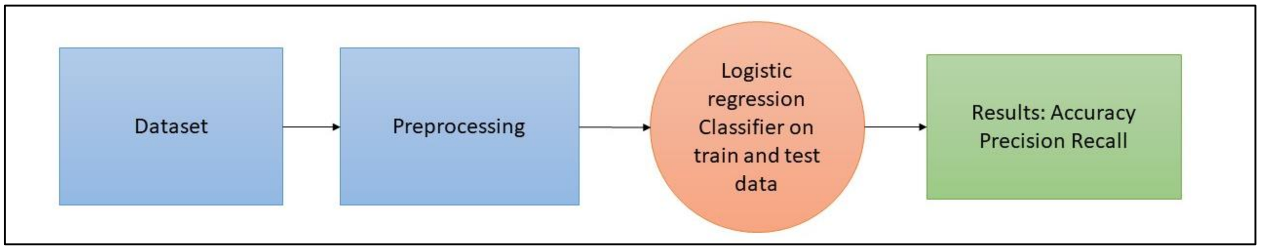
In this project I have performed classification on a binary classification problem using Logistic regression. We know that the breast cancer is one of the most common cancers among women across the globe leading to many deaths in a year and it is a serious health issue in the recent times. Detecting and diagnosing breast cancer at a very early stage will improve the chance of survival by a huge extent. The primary identification and prediction of type of the cancer would help out in order to assist and supervise the patients. Classification of tumor with respect to benign and malignant helps patients undergo treatments that are not necessary. Hence, this subject is being researched as the correct diagnosis of breast cancer and classification of patients into malignant or benign groups is will help thousands of people. Breast cancer Wisconsin dataset is collected from the UCI machine learning repository has 569 instances with 31 attributes. This dataset is then preprocessed and fed into the logistic regression classifier for further analysis of data. The logistic regression model that I have proposed can effectively discriminate between benign and malignant breast disease and identify the most important features associated with breast cancer.

1. Introduction

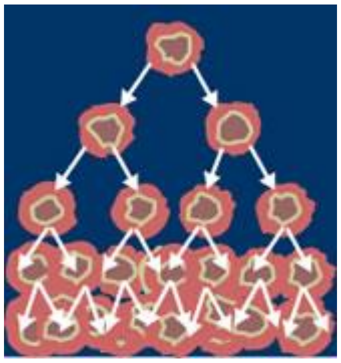
Breast cancer is a cancer that are developed in the cells of the breasts. It is one of the most common cancers after skin cancer which have been diagnosed in women. The breast cancer is categorized into benign cancer, In situ cancer and Invasive cancer. Benign cancers will not spread to other organs and can be detected by mammography. The other two types are concentrated in the ducts. Early detection of cancer into cancerous(malignant) and non-cancerous(benign) group is a must. According to a recent study breast cancer, survival rates have increased and the death count due to cancer has drastically reduced. This is because of the factors such as analysis and detection of cancer cells at a very early stage which will render help in early diagnosis of the affected patients.

Machine learning is a data analysis method where the analysis is automated and the machine learns from the data which identifies patters and makes decisions based on the previous data it has learned about. Machine learning allows the classifiers to learn from previous instances to analyze huge and complex datasets. Medical industries are using machine learning as a primary tool to detect and classify benign and malignant tumors. This detection cannot be performed by basic statistical analysis as it might not be so accurate. Machine learning algorithms have largely helped in detection of cancer cells into cancerous and non-cancerous groups.

In the process of analysis, we first have a dataset which is preprocessed and fed into a classifier. The preprocessed data is then divided into parts so that the machine learns from the part of data and then tests and predicts correctly on the rest of the data. Here we used the logistic regression to predict if the cancer cells benign or malignant. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.


Dataset
Preprocessing
Logistic regression Classifier on train and test data
Results: Accuracy Precision Recall

2. Dataset



Here in this project the dataset in use is the Wisconsin Diagnostic Breast Cancer dataset where the features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The dataset contains 569 instances with 32 attributes.

The following give the attribute information of the dataset. The dataset looks at the predictor classes M: malignant or B: benign breast mass. There are two main classifications of tumors. One is known as benign and the other as malignant. A benign tumor is a tumor that does not invade its surrounding tissue or spread around the body. A malignant tumor is a tumor that may invade its surrounding tissue or spread around the body.

The first two columns give the details of

I. Sample ID
II. Diagnosis which is Malignant / Benign

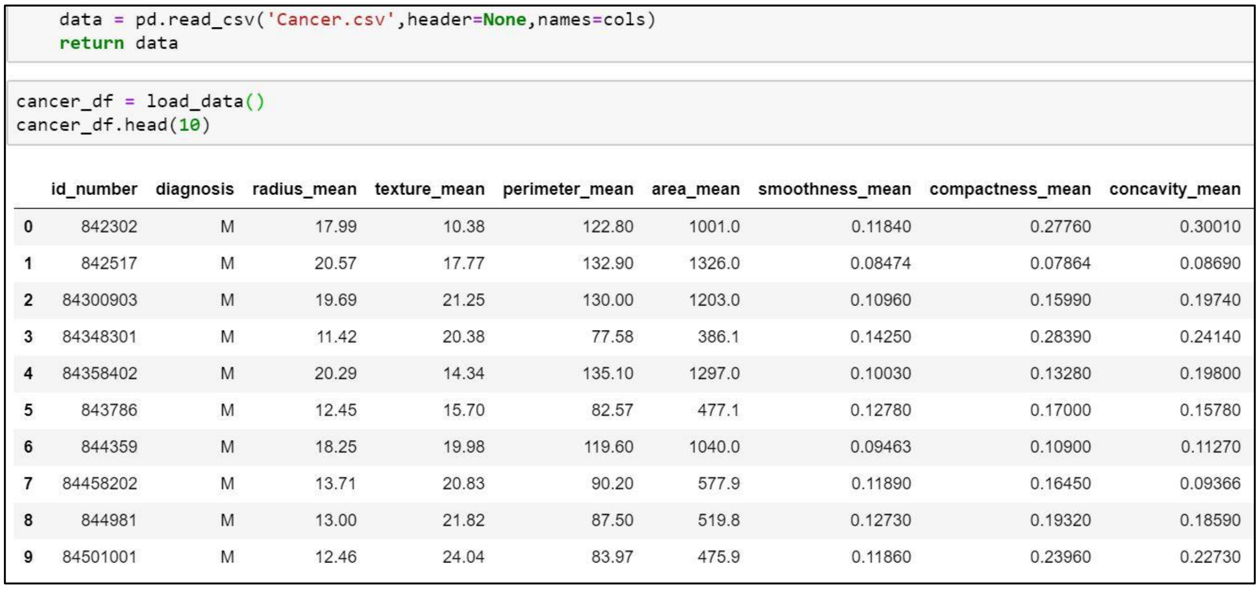
For each cell nucleus, the following ten features were measured:

I. Radius (mean of all distances from the center to points on the perimeter)
II. Texture (standard deviation of gray-scale values)
III. Perimeter
IV. Area
V. Smoothness (local variation in radius lengths)
VI. Compactness (perimeter^2 / area - 1.0)
VII. Concavity (severity of concave portions of the contour)
VIII. Concave points (number of concave portions of the contour)
IX. Symmetry
X. Fractal dimension (“coastline approximation” - 1)

For each feature three additional measures are given:

I. Mean
II. Standard error
III. Largest/ “worst”

This dataset is divided into 3 parts Training, Validation and Testing in the ratio 80% 10% and 10% respectively. On these partitioned data we will run the logistic regression model to find our results. Here is the representation of a few columns and values from the dataset.



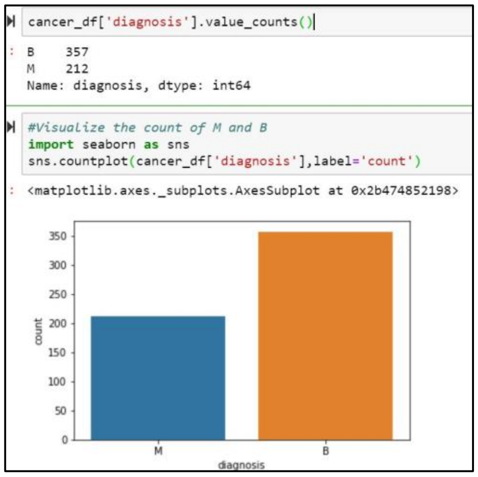
```
    data = pd.read_csv('Cancer.csv',header=None,names=cols)
    return data
```

```
cancer_df = load_data()
cancer_df.head(10)
```

	id_number	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270
7	84458202	M	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366
8	844981	M	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590
9	84501001	M	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730

Wisconsin Diagnostic Breast Cancer dataset

Visualizing the diagnosis column, we can see that the column has data regarding cells being malignant or benign. Analyzing this column, I found out that there are 357 benign cancer tumors which is approximately 62% and 212 malignant which is 37%. We can use pair plot to find the M and B cells relationship between the features. Here the blue is malignant points and orange are benign points.

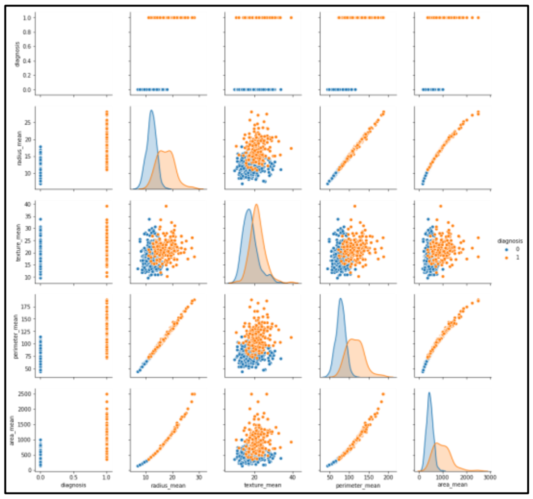


```
cancer_df['diagnosis'].value_counts()
```

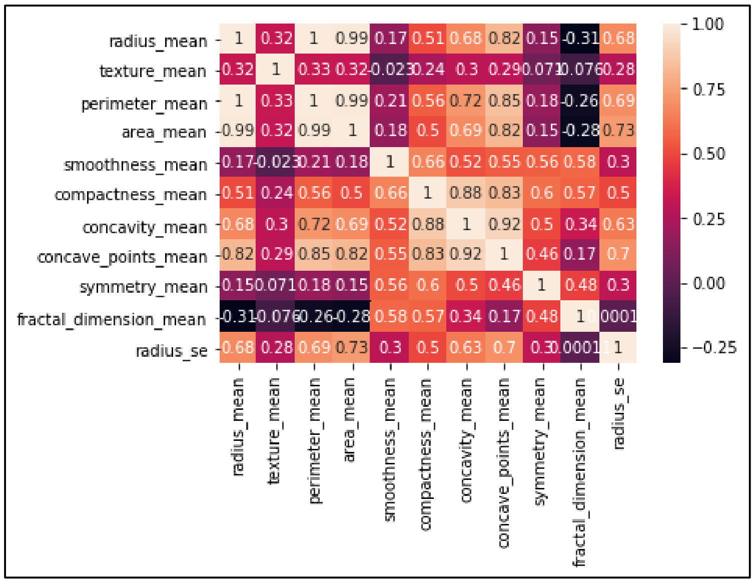
```
B    357
M    212
Name: diagnosis, dtype: int64
```

```
#Visualize the count of M and B
import seaborn as sns
sns.countplot(cancer_df['diagnosis'],label='count')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2b474852198>
```

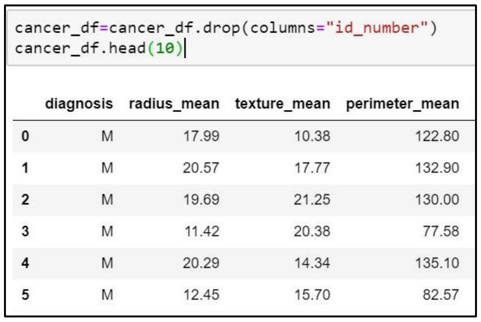


Further exploring the dataset, we can find the correlation between the features that is all the 10 features. We can see how one column can influence the other column. It can be a positive or a negative correlation or influence. If there is a 0 value, then there is no relation between the columns.



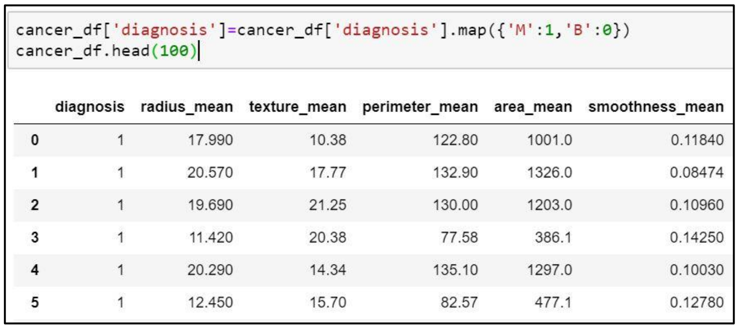
3. Preprocessing

Processing of data is one of the most essential steps in classification problems. The data needs to be cleaned and processed by removing the incorrect, redundant and noisy unwanted data cells before we start working on it. Data preprocessing can be achieved by various subsequent means such as reduction of data, data cleaning, data integration, data transformation. So we have categorical data in case of the diagnosis column that we saw in the above case. Categorical data are the data that contain label values instead of numeric values. In our case we will need to convert the M's and B's in the diagnosis column into 1's and 0's respectively and remove the unwanted column which is the sample_id.



```
cancer_df=cancer_df.drop(columns="id_number")
cancer_df.head(10)
```

	diagnosis	radius_mean	texture_mean	perimeter_mean
0	M	17.99	10.38	122.80
1	M	20.57	17.77	132.90
2	M	19.69	21.25	130.00
3	M	11.42	20.38	77.58
4	M	20.29	14.34	135.10
5	M	12.45	15.70	82.57

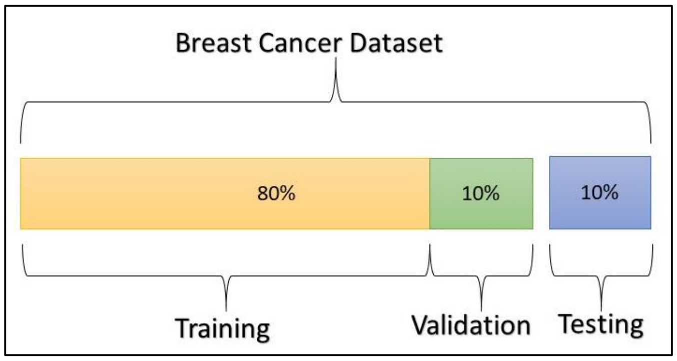


```
cancer_df['diagnosis']=cancer_df['diagnosis'].map({'M':1,'B':0})
cancer_df.head(100)
```

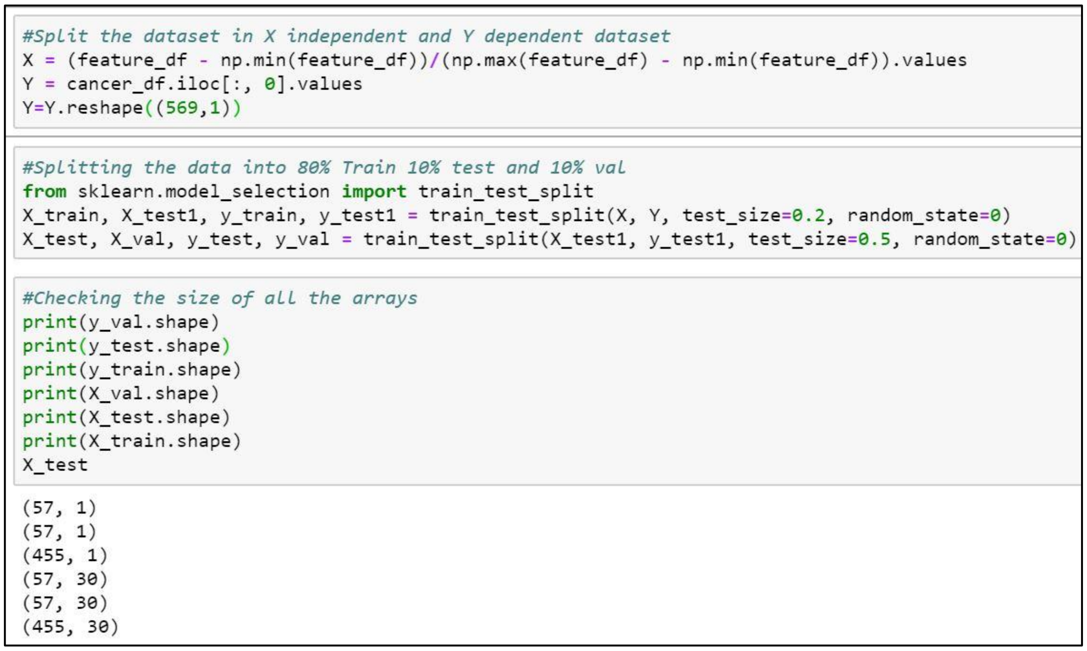
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	1	17.990	10.38	122.80	1001.0	0.11840
1	1	20.570	17.77	132.90	1326.0	0.08474
2	1	19.690	21.25	130.00	1203.0	0.10960
3	1	11.420	20.38	77.58	386.1	0.14250
4	1	20.290	14.34	135.10	1297.0	0.10030
5	1	12.450	15.70	82.57	477.1	0.12780

Dropped the sample_id column and converted diagnosis to numerical values

Next data partitioning is into train test and validation data. So this partitioning is done so that the machine will learn for the partitioned data so the training sample will be used to fit the data whereas the validation sample will provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. Testing sample of data is used to provide an unbiased evaluation of a final model fit on the training dataset.



The total data is now divided into 80% training data 10% testing data and 10% validation data. Here I have used sklearn library to preprocess and split the data. As the one split can be done to give two parts I performed split twice get divide that data into 3 sets.



```
#Split the dataset in X independent and Y dependent dataset
X = (feature_df - np.min(feature_df))/(np.max(feature_df) - np.min(feature_df)).values
Y = cancer_df.iloc[:, 0].values
Y=Y.reshape((569,1))
```

```
#Splitting the data into 80% Train 10% test and 10% val
from sklearn.model_selection import train_test_split
X_train, X_test1, y_train, y_test1 = train_test_split(X, Y, test_size=0.2, random_state=0)
X_test, X_val, y_test, y_val = train_test_split(X_test1, y_test1, test_size=0.5, random_state=0)
```

```
#Checking the size of all the arrays
print(y_val.shape)
print(y_test.shape)
print(y_train.shape)
print(X_val.shape)
print(X_test.shape)
print(X_train.shape)
X_test
```

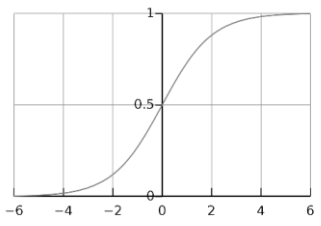
```
(57, 1)
(57, 1)
(455, 1)
(57, 30)
(57, 30)
(455, 30)
```

4. Architecture

Logistic regression is a classification algorithm which is used to assign observations to different and discrete set of classes. In linear regression we see that the output is linear whereas in logistic regression transforms its output using the logistic sigmoid function which will return a probability value which can then be mapped to two or more discrete classes. Our case of breast cancer analysis is a binary logistic regression where we have 2 values to classify the cancer i.e. M and B.

To map any predicted values to probabilities we use the sigmoid function. This function will map any real value into another value between 0 and 1. Here if we define the threshold classifier output

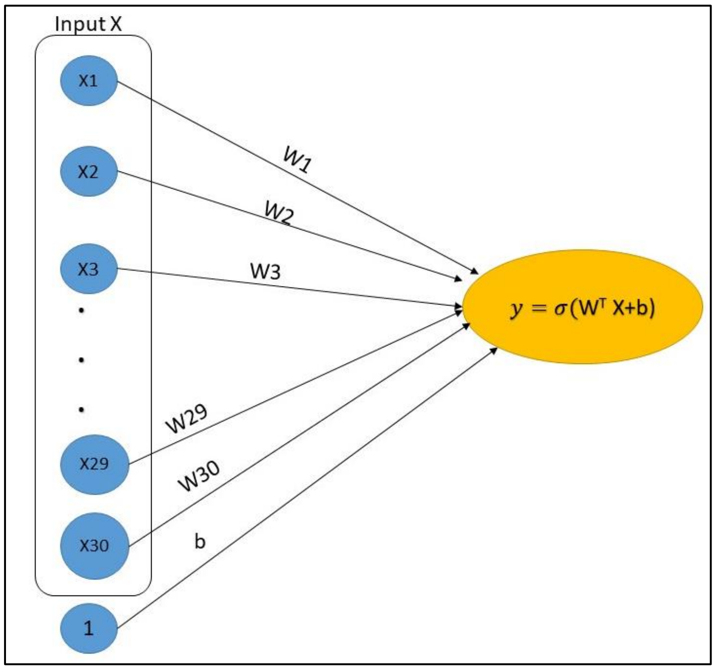
$h_\Theta(x)$ at 0.5
if $h_\Theta(x) >= 0.5$, predict "y =1"
if $h_\Theta(x) < 0.5$, predict "y =0"
$h_\Theta(x) = g(\Theta^T x)$
$g(z) = 1/1+e^{-z}$ This is the sigmoid function.



Sigmoid function

A logistic regression works on the principle of the equation **$y=W^TX+b$** where the X represents the input array W is the array of weights and B is the bias. The function Y is then sent to a sigmoid function which gives the probability depending on value of Y.

Hence the final function looks like **$y=\sigma(W^TX+b)$**. In case of our cancer data we have 30 input values for the 30 features that we have which we will specify weights and bias and feed it into a sigmoid function to compute the predictions.



Computational graph for logistic regression

Here we will use the following cost function which is also called as cross-entropy. This can be divided into two cost functions one for y=1 and one for y=0.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\mathrm{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if y = 1}$$

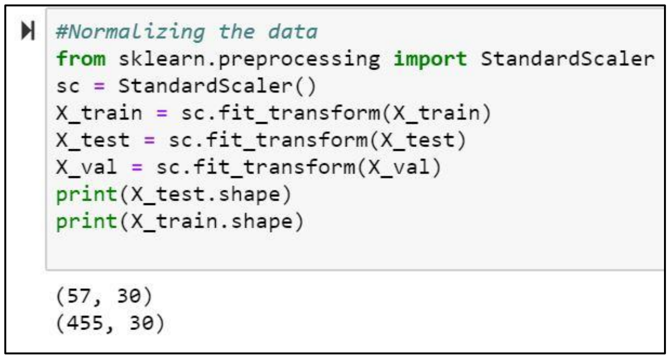
$$\mathrm{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if y = 0}$$

When we join these two functions and compressed into one we get the cost function which is used in this prediction.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

5. Analysis and Results

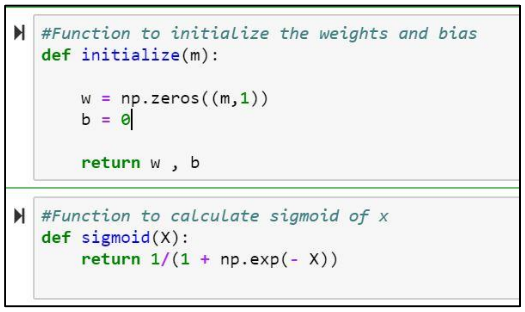
The data which was partitioned into train, test and validation are now normalized to get an optimal result. The main goal to normalize the entire data is to change all the numeric values in the dataset to a same scale keeping in mind the different ranges. Here I have used sklearn preprocessing library's Standard scalar function to normalize the dataset.



```
#Normalizing the data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
X_val = sc.fit_transform(X_val)
print(X_test.shape)
print(X_train.shape)
```

```
(57, 30)
(455, 30)
```

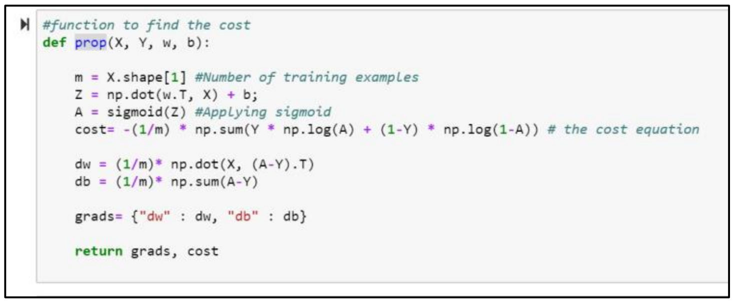
The next step is to initialize the weights and bias and the learning rate. These are the 3 hyperparameters that we will initialize and vary to get the correct accuracy. The python code snippets for the following process is as below. Here I have stored the cost at every 10 iterations and calculated accuracy of the train test and validation sets



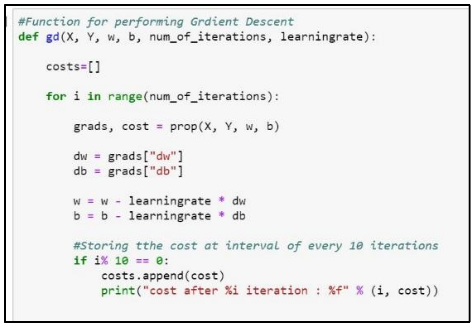
```
#Function to initialize the weights and bias
def initialize(m):

    w = np.zeros((m,1))
    b = 0

    return w , b
```

```
#Function to calculate sigmoid of x
def sigmoid(X):
    return 1/(1 + np.exp(- X))
```

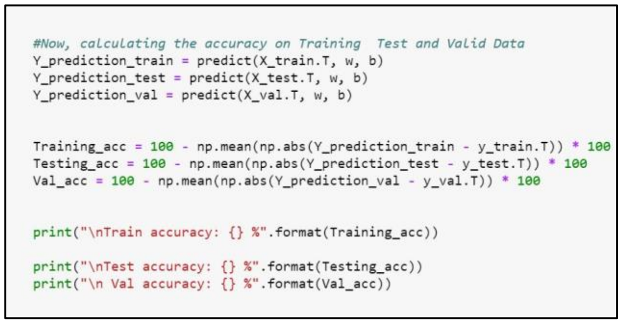
.



```
#function to find the cost
def prop(X, Y, w, b):

    m = X.shape[1] #Number of training examples
    Z = np.dot(w.T, X) + b;
    A = sigmoid(Z) #Applying sigmoid
    cost= -(1/m) * np.sum(Y * np.log(A) + (1-Y) * np.log(1-A)) # the cost equation

    dw = (1/m)* np.dot(X, (A-Y).T)
    db = (1/m)* np.sum(A-Y)

    grads= {"dw" : dw, "db" : db}

    return grads, cost
```

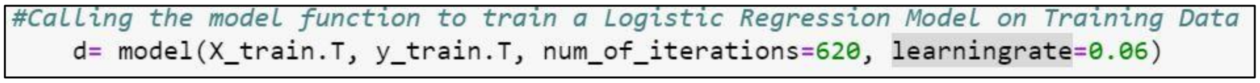


```
#Function for performing Grdient Descent
def gd(X, Y, w, b, num_of_iterations, learningrate):

    costs=[]

    for i in range(num_of_iterations):

        grads, cost = prop(X, Y, w, b)

        dw = grads["dw"]
        db = grads["db"]

        w = w - learningrate * dw
        b = b - learningrate * db

        #Storing tthe cost at interval of every 10 iterations
        if i% 10 == 0:
            costs.append(cost)
            print("cost after %i iteration : %f" % (i, cost))
```



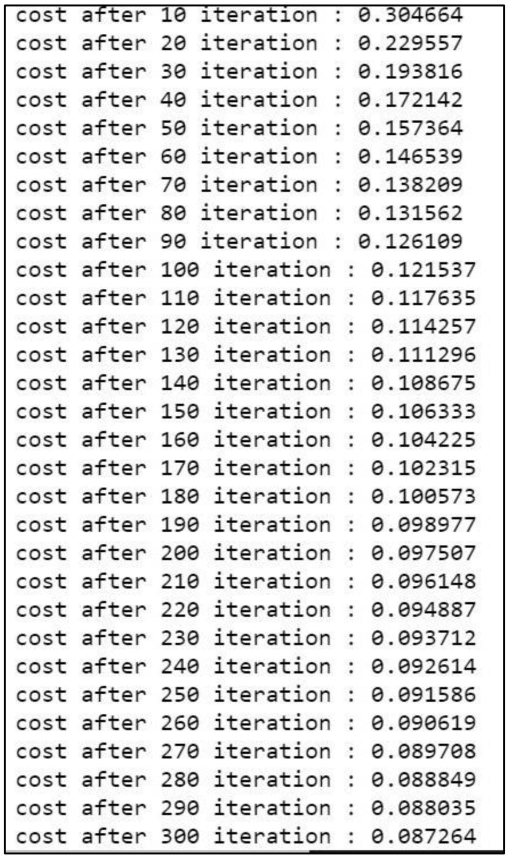
```
#Now, calculating the accuracy on Training  Test and Valid Data
Y_prediction_train = predict(X_train.T, w, b)
Y_prediction_test = predict(X_test.T, w, b)
Y_prediction_val = predict(X_val.T, w, b)


Training_acc = 100 - np.mean(np.abs(Y_prediction_train - y_train.T)) * 100
Testing_acc = 100 - np.mean(np.abs(Y_prediction_test - y_test.T)) * 100
Val_acc = 100 - np.mean(np.abs(Y_prediction_val - y_val.T)) * 100


print("\nTrain accuracy: {} %".format(Training_acc))

print("\nTest accuracy: {} %".format(Testing_acc))
print("\n Val accuracy: {} %".format(Val_acc))
```

Here I have changed the learningrate from the range of 0.1 to 0.001 to vary my accuracy. The next parameter I have varied was the number of iterations from 100 to 2000. For an optimal result I have chosen the value of learning rate at 0.06 and the number of iterations to be 620. Looking at the accuracy score on the training data for each model to classify if a patient has cancer or not.

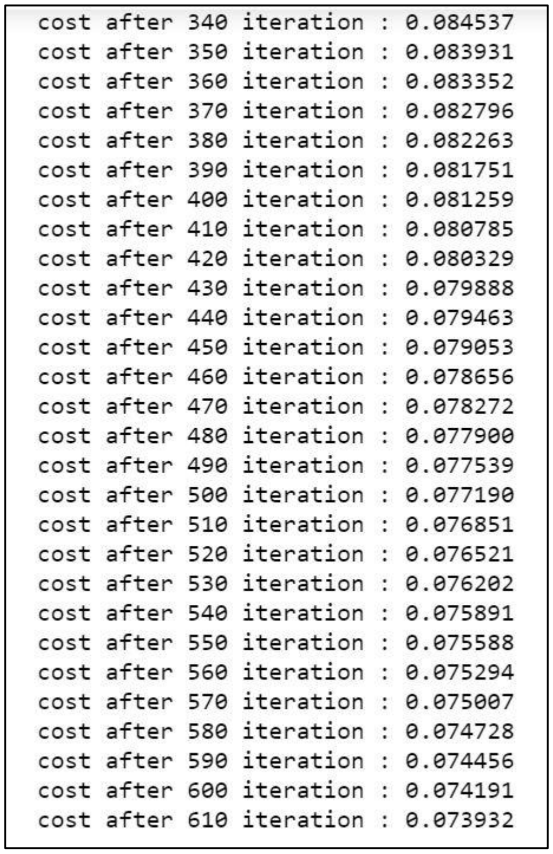


```
#Calling the model function to train a Logistic Regression Model on Training Data
    d= model(X_train.T, y_train.T, num_of_iterations=620, learningrate=0.06)
```

Outputs:
In the model we see that the cost will decrease in every iteration.

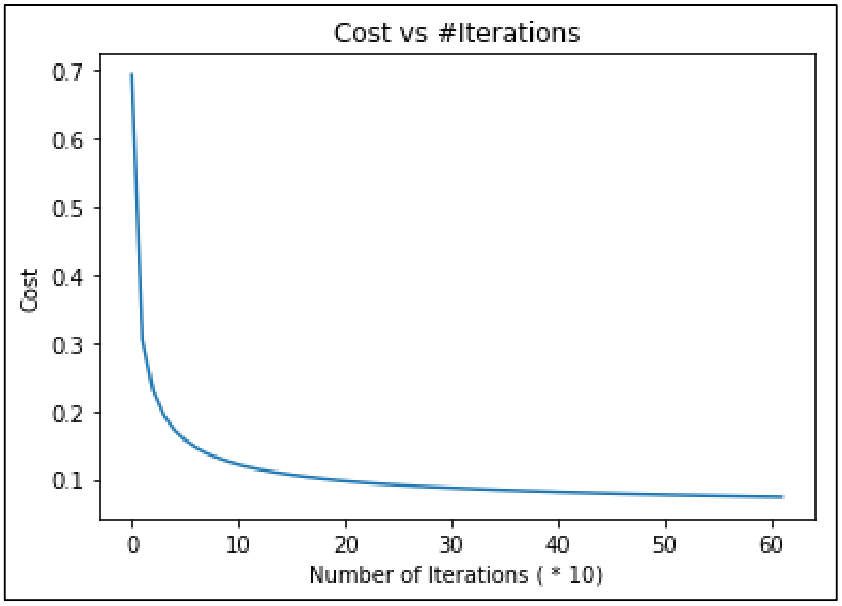


```
cost after 10 iteration : 0.304664
cost after 20 iteration : 0.229557
cost after 30 iteration : 0.193816
cost after 40 iteration : 0.172142
cost after 50 iteration : 0.157364
cost after 60 iteration : 0.146539
cost after 70 iteration : 0.138209
cost after 80 iteration : 0.131562
cost after 90 iteration : 0.126109
cost after 100 iteration : 0.121537
cost after 110 iteration : 0.117635
cost after 120 iteration : 0.114257
cost after 130 iteration : 0.111296
cost after 140 iteration : 0.108675
cost after 150 iteration : 0.106333
cost after 160 iteration : 0.104225
cost after 170 iteration : 0.102315
cost after 180 iteration : 0.100573
cost after 190 iteration : 0.098977
cost after 200 iteration : 0.097507
cost after 210 iteration : 0.096148
cost after 220 iteration : 0.094887
cost after 230 iteration : 0.093712
cost after 240 iteration : 0.092614
cost after 250 iteration : 0.091586
cost after 260 iteration : 0.090619
cost after 270 iteration : 0.089708
cost after 280 iteration : 0.088849
cost after 290 iteration : 0.088035
cost after 300 iteration : 0.087264
```

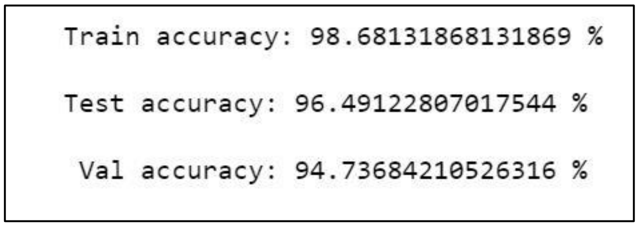


```
cost after 340 iteration : 0.084537
cost after 350 iteration : 0.083931
cost after 360 iteration : 0.083352
cost after 370 iteration : 0.082796
cost after 380 iteration : 0.082263
cost after 390 iteration : 0.081751
cost after 400 iteration : 0.081259
cost after 410 iteration : 0.080785
cost after 420 iteration : 0.080329
cost after 430 iteration : 0.079888
cost after 440 iteration : 0.079463
cost after 450 iteration : 0.079053
cost after 460 iteration : 0.078656
cost after 470 iteration : 0.078272
cost after 480 iteration : 0.077900
cost after 490 iteration : 0.077539
cost after 500 iteration : 0.077190
cost after 510 iteration : 0.076851
cost after 520 iteration : 0.076521
cost after 530 iteration : 0.076202
cost after 540 iteration : 0.075891
cost after 550 iteration : 0.075588
cost after 560 iteration : 0.075294
cost after 570 iteration : 0.075007
cost after 580 iteration : 0.074728
cost after 590 iteration : 0.074456
cost after 600 iteration : 0.074191
cost after 610 iteration : 0.073932
```

This is the graph for the cost vs the number of iterations:

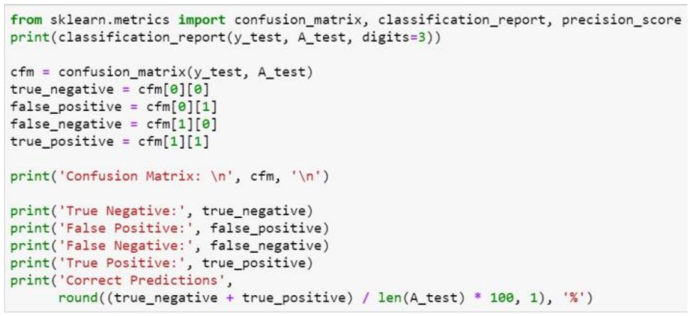


The accuracy for train dataset was approximately 98% test was at 96% and the one on validation dataset was 94%

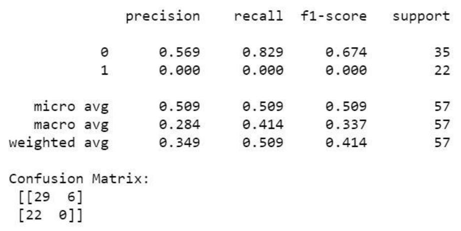


```
Train accuracy: 98.68131868131869 %

Test accuracy: 96.49122807017544 %

 Val accuracy: 94.73684210526316 %
```

The confusion matrix tells us how many patients each model misdiagnosed (number of patients with cancer that were misdiagnosed as not having cancer a.k.a false negative, and the number of patients who did not have cancer that were misdiagnosed with having cancer a.k.a false positive) and the number of correct diagnosis, the true positives and true negatives.
I ran the confusion matrix on my test dataset but the results in the confusion matrix were not accurate even after tuning the hyperparameters to a great extent.



```python
from sklearn.metrics import confusion_matrix, classification_report, precision_score
print(classification_report(y_test, A_test, digits=3))

cfm = confusion_matrix(y_test, A_test)
true_negative = cfm[0][0]
false_positive = cfm[0][1]
false_negative = cfm[1][0]
true_positive = cfm[1][1]

print('Confusion Matrix: \n', cfm, '\n')

print('True Negative:', true_negative)
print('False Positive:', false_positive)
print('False Negative:', false_negative)
print('True Positive:', true_positive)
print('Correct Predictions',
      round((true_negative + true_positive) / len(A_test) * 100, 1), '%')
```



```
             precision    recall  f1-score   support

          0      0.569     0.829     0.674        35
          1      0.000     0.000     0.000        22

  micro avg      0.509     0.509     0.509        57
  macro avg      0.284     0.414     0.337        57
weighted avg      0.349     0.509     0.414        57

Confusion Matrix: 
 [[29  6]
 [22  0]]
```

6. Conclusion

As there is increase in women with breast cancer Machine learning and its algorithms will play a very important role in predicting cancer in a them and helping them get diagnosed at a very early stage eliminating the risks. With technology coming into modern world and with the abundance of resources we need to extract these resources and analyze them to reach the desired result. In the above model I got an accuracy of 98% which says that logistic regression is one of the best methods to predict breast cancer.

References:

UCI Machine Learning Repository:
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
Logistic Regression |Classification by Andrew NG :https://www.youtube.com/watch?v=-la3q9d7AKQ
UB notes on Logistic Regression:https://ublearns.buffalo.edu/bbcswebdav/pid-5108334-dt-content-rid-25438520_1/courses/2199_23170_COMB/4.3.2-LogisticReg.pdf
Breast Cancer Detection using Machine Learning by Randerson:https://medium.com/@randerson112358/breast-cancer-detection-using-machine-learning-38820fe98982