



# Breast Cancer Prediction

Module Code: CMP7161  
Module Coordinator: Yevgeniya Kovalcuk  
Student Name: Muhammad Waleed  
Student Id: 18157492  
Submitted on: Date: 12th May, 2020

## Table of Content

<b>1. Abstract</b>	<b>Page 3</b>
<b>2. Aim</b>	<b>Page 3</b>
<b>3. Introduction</b>	<b>Page 3</b>
<b>3.1. Prevention</b>	
<b>4. Background</b>	<b>Page 4</b>
<b>5. About Dataset</b>	<b>Page 4</b>
<b>6. Problem Statement</b>	<b>Page 6</b>
<b>7. Summary of Approach</b>	<b>Page 6-15</b>
<b>7.1. Getting and Pre-processing the Dataset</b>	
<b>7.2. Splitting the Dataset</b>	
<b>7.3. Creating and Training Models</b>	
<b>7.4. Testing Models</b>	
<b>8. Models</b>	<b>Page 15-17</b>
<b>8.1. Logistic Regression</b>	
<b>8.2. Decision Tree Classifier</b>	
<b>8.3. Random Forest Classifier</b>	
<b>9. Models Comparison</b>	<b>Page 17</b>
<b>10. Conclusion</b>	<b>Page 18</b>
<b>11. Reference</b>	<b>Page 18</b>

## 1. Abstract:

Breast cancer is a common cancer that is threatening the lives of all the women in the whole world. We can save the lives of women by knowing about breast cancer in its early stage. The early diagnosis and prognosis of a cancer type have become a necessity in cancer research, as it can facilitate the patients. In this report we present a review of how Machine Learning (ML) approaches employed in the modeling of breast cancer detection. This report helps us in accurately detecting breast cancer in patients at an early stage. The predictive models discussed here are based on various supervised ML techniques as well as on different input features and data samples.

## 2. Aim:

Our aim is to develop machine learning algorithms to detect breast cancer in patients at an early stage.

## 3. Introduction:

Cancer is the leading cause of death among females worldwide. It caused more deaths than other diseases, such as tuberculosis or malaria. The World Health Organization for Cancer Research (WHO) agencies (i.e., the International Agency for Cancer Research (IARC) and the American Cancer Society) reported that there were 17.1 million new cancer cases worldwide in 2020. The WHO predicts that cancer incidence will increase to 27.5 million by 2040, with 16.3 million deaths due to cancer.

Breast cancer is one of the most dangerous and common reproductive cancers that affect mostly women worldwide. The early diagnosis of BC can improve the prognosis and chance of survival significantly, as it can promote timely clinical treatment to patients. Many techniques have been developed for early detection and treatment of breast cancer and to reduce the number of deaths. The correct diagnosis of BC and the classification of patients into malignant or benign groups is the subject of much research. Because of its unique advantages in critical features detection from complex BC datasets, machine learning (ML) is widely recognized as the methodology of choice in BC pattern classification and forecast modeling. Nowadays, the demand for machine learning is growing.

### 3.1. Prevention:

One must lower the risk of developing breast cancer as there is no proven way to completely prevent this cancer but if it is early diagnosed then it can be cured. Our Machine Learning model helps in detecting breast cancer at its early stage and this will help you in developing measures for the prevention of breast cancer.

- One way is the removal of the breast for those women who have BRCA1 and BRCA2 genetic mutations. It reduces the risk of breast cancer.
- Taking drugs for prevention of breast cancer by the recommendation of their medical specialists. **Tamoxifen** is a type of drug called a selective estrogen receptor modulator. This type of drug is often used for patients who have breast cancer. Tamoxifen blocks the effects of estrogen on tumor growth. It can also lower the risk of breast cancer.
- **Raloxifene** is also called selective estrogen receptor modulator and can lower the risk of breast cancer.
- **Involving in some physical activity** lowers the risk of breast cancer.

#### 4. Background:

Machine learning (ML) has become an important part of medical imaging research. ML methods evolve from manual seeded input to automated initialization. Progress in the field of ML has led to more intelligent and self-sufficient computer-aided diagnosis (CAD) systems, as the learning potential of ML methods is steadily improving. More automated methods are emerging with deep feature learning and representation. Recent advances of ML with deep and comprehensive representation approach, commonly called deep learning (DL) approaches, have had a very significant impact on improving the diagnostic capabilities of CAD systems.

Breast cancer was not this much common among the women like it is nowadays. Women were unaware of this cancer. People are making huge efforts to increase awareness regarding Breast cancer but few women in some areas of the country have adequate knowledge of the risk factors and preventive measures or screening techniques for early detection of BC. Breast cancer is the top cancer in women worldwide and is raising particularly in developing countries where the majority of breast cancer cases are diagnosed in late stages. Almost 70% of women with breast cancer are aged over 50 years, and only 5% are younger than 40 years old. Approximately **700,000 cases** are reported annually worldwide, of which 57% of these cases present in developing countries. According to WHO estimates it represents 10% of all cancers diagnosed worldwide and constituted **22%** of all new cancers in 2000 in women making it the most common cancer in females. Pakistan alone has the highest rate of Breast Cancer than any other Asian country as approximately 90000 new cases are diagnosed every year out of which 40000 dies. According to research conducted approximately 1 out of every 9 women are likely to suffer from this disease at any point in their lives and about 77% of invasive breast cancer occurred in women above 50 years, but if diagnosed early the survival rates approach 90%. Many of the patients do not even know about this cancer. This is all done by awareness. Awareness against this deadly cancer can save women lives to a great extent. By using a machine-learning algorithm we can detect BC in early stages and can save lives.

#### 5. About Dataset:

The Dataset used in this project is taken from Kaggle. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe the characteristics of the cell nuclei present in the image. In the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

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.

All feature values are recoded with four significant digits.

Class distribution: 357 benign, 212 malignant

Attribute Information:

1. ID number
2. Diagnosis (M = malignant, B = benign)
3. 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 ( $\text{perimeter}^2 / \text{area} - 1.0$ )
  - concavity (severity of concave portions of the contour)
  - concave points (number of concave portions of the contour)
  - symmetry
  - fractal dimension ("coastline approximation" - 1).

**Id:** It contains id number of patients.

**Diagnosis:** The diagnosis of breast tissues (M = malignant, B = benign)

**Radius\_mean:** Mean of distances from the center to points on the perimeter.

**texture\_mean:** standard deviation of gray-scale values

**perimeter\_mean:** mean size of the core tumor

**area\_mean:** Mean of smoothness (local variation in radius lengths)

**smoothness\_mean:** mean of local variation in radius lengths

**compactness\_mean :** mean of  $\text{perimeter}^2 / \text{area} - 1.0$

**concavity\_mean:** mean of the severity of concave portions of the contour

**concave points\_mean:** mean for the number of concave portions of the contour

**symmetry\_mean:** Mean of similar points.

**fractal\_dimension\_mean:** mean for "coastline approximation" - 1

**radius\_se:** standard error for the mean of distances from the center to points on the perimeter

**texture\_se:** standard error for standard deviation of gray-scale values

**perimeter\_se:** standard error of the mean size of the core tumor.

**area\_se:** Standard error for area.

**smoothness\_se:** standard error for local variation in radius lengths

**compactness\_se:** standard error for  $\text{perimeter}^2 / \text{area} - 1.0$

**concavity\_se:** standard error for the severity of concave portions of the contour.

**Concave\_points\_se:** standard error for the number of concave portions of the contour.

## 6. Problem statement:

Our main concern is to detect Breast cancer in most women at an early stage. From the past few years Machine learning methods have become a powerful source for medical researchers to detect deadly diseases and cancer. These methods can detect patterns and relationships between complex datasets, and they can potentially predict future outcomes for cancer types. Predicting if the cancer diagnosis is benign or malignant based on several observations. We will develop machine learning algorithms to detect breast cancer and other cancer cells at its early stage in patients.

## 7. Summary of Approaches:

The summary gives a brief description of the implementation of both approaches and modeling with the comparison. It is divided into three steps:

- Getting and preprocessing the dataset
- Splitting the data
- Creating and training the models
- Testing the models

### 7.1. Getting and preprocessing the data set

In this step we get the data and preprocess it to further use it for training and testing purposes.

#### Import the libraries:

Followings are some libraries that are required for this project:

- Pandas
- Matplotlib
- Seaborn
- sklearn

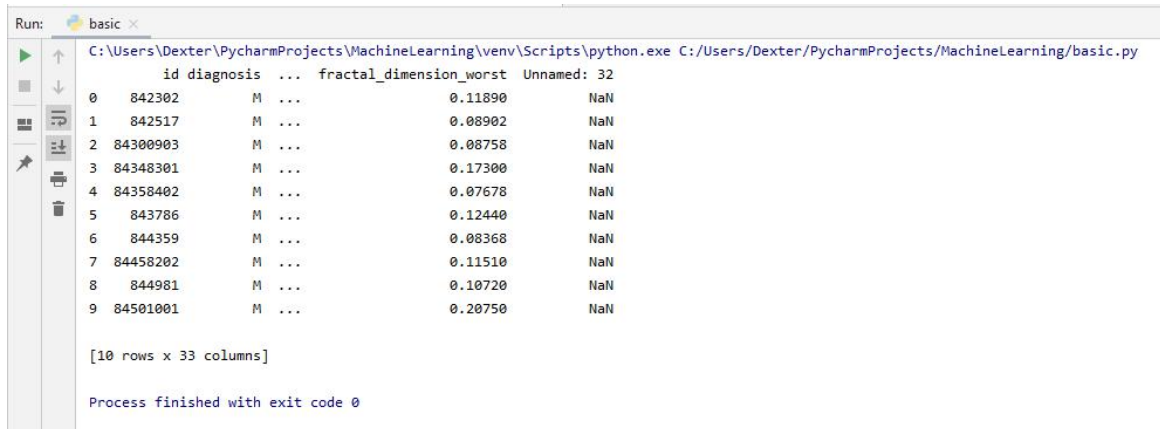
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 from sklearn.preprocessing import LabelEncoder
5 from sklearn.model_selection import train_test_split
6 from sklearn.preprocessing import StandardScaler
7 from sklearn.linear_model import LogisticRegression
8 from sklearn.tree import DecisionTreeClassifier
9 from sklearn.ensemble import RandomForestClassifier
10 from sklearn.metrics import confusion_matrix
```

## Load the Dataset:

Now load the data from CSV file using pandas.

```
7      #Load the data
8      df = pd.read_csv('data.csv')
9
10     #check the first 10 lines of the imported data
11     print(df.head(10))
```

## [output]:



```
Run: basic x
C:\Users\Dexter\PycharmProjects\MachineLearning\venv\Scripts\python.exe C:/Users/Dexter/PycharmProjects/MachineLearning/basic.py
id diagnosis ... fractal_dimension_worst Unnamed: 32
0 842302 M ... 0.11890 NaN
1 842517 M ... 0.08902 NaN
2 84300903 M ... 0.08758 NaN
3 84348301 M ... 0.17300 NaN
4 84358402 M ... 0.07678 NaN
5 843786 M ... 0.12440 NaN
6 844359 M ... 0.08368 NaN
7 84458202 M ... 0.11510 NaN
8 844981 M ... 0.10720 NaN
9 84501001 M ... 0.20750 NaN

[10 rows x 33 columns]
Process finished with exit code 0
```

## Data cleansing and Data pre-processing:

After loading the dataset, data should be analyzed for better predictions. Pre-processing on data might improve the model accuracy and be more reliable. It does not always have to improve our results but when we are conscious of the features and use a proper input, we might reach some outcomes easier. It gives information about data and helps to find out that our data is clean or not, is there any missing values or it is balanced or not, etc.

## Data type:

Followings are the data types of each column

```

id                int64
diagnosis         object
radius_mean       float64
texture_mean      float64
perimeter_mean    float64
area_mean         float64
smoothness_mean   float64
compactness_mean  float64
concavity_mean    float64
concave points_mean float64
symmetry_mean     float64
fractal_dimension_mean float64
radius_se         float64
texture_se        float64
perimeter_se      float64
area_se           float64
smoothness_se     float64
compactness_se    float64
concavity_se      float64
concave points_se float64
symmetry_se       float64
fractal_dimension_se float64
radius_worst      float64
texture_worst     float64
perimeter_worst   float64
area_worst        float64
smoothness_worst  float64
compactness_worst float64
concavity_worst   float64
concave points_worst float64
symmetry_worst    float64
fractal_dimension_worst float64

```

## Shape and Missing Values Removal:

The following picture shows the shapes of data set that has an Unnamed column that has to be removed

The screenshot shows a Jupyter Notebook interface. The top bar indicates the file path: `C:\Users\Dexter\PycharmProjects\MachineLearning\venv\Scripts\python.exe C:/Users/Dexter/PycharmProjects/MachineLearning/basic.py`. The left sidebar shows a file explorer with a folder icon and a file icon. The main area displays a list of variables and their shapes. The variables are: `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), and `dtype: int64` (569, 32).

## Cleaning the data:

Deleting uncleaness and empty values from the dataset to make it clean and more accurate.



```

16  #Count the empty (NaN, NAN, na) values in each column
17  print(df.isna().sum())
18  #Drop the column with all missing values (na, NAN, NaN)
19  #NOTE: This drops the column Unnamed
20  df = df.dropna(axis=1)
21
22  #Get the new count of the number of rows and cols
23  df.shape

```

### Get the new count:

Checking the count of rows and columns after cleaning

```
(569, 32)
```

### Visualize the count:

Count between malignant and benign

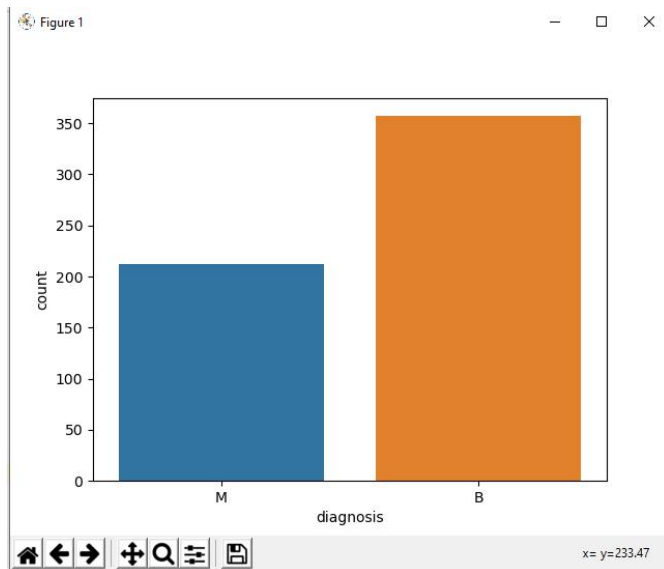
```

25  #Get a count of the number of 'M' & 'B' cells
26  count_M_and_B = df['diagnosis'].value_counts()
27
28  #Visualize this count
29  sns.countplot(df['diagnosis'], label="Count")
30  plt.show()

```

### [output]:

Visual output can be seen in the following histogram



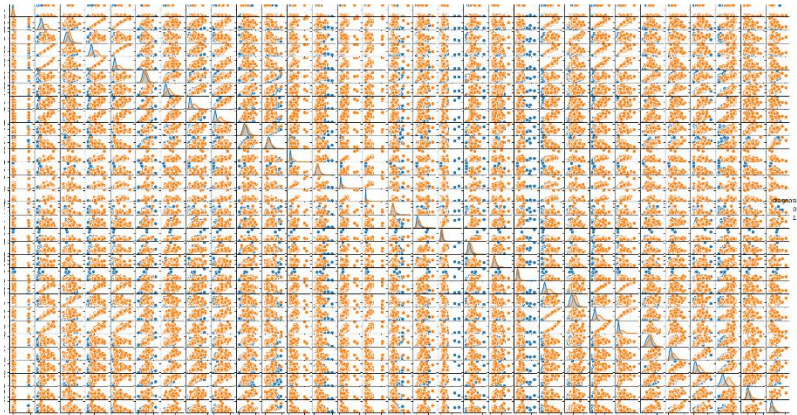
### Encode the categorical data:

Encoding the categorical data so that model will use this to get accurately trained

```
33     #Encoding categorical data values (  
34  
35     labelencoder_Y = LabelEncoder()  
36     df.iloc[:,1]= labelencoder_Y.fit_transform(df.iloc[:,1].values)  
37     print(labelencoder_Y.fit_transform(df.iloc[:,1].values))  
38     sns.pairplot(df, hue="diagnosis")  
39  
40     df.corr()  
41     plt.show()
```

### [output]:

Following Scatter plots can be seen in the following picture



Category are converted into bits that is malignant is converted to 1 and benign to 0

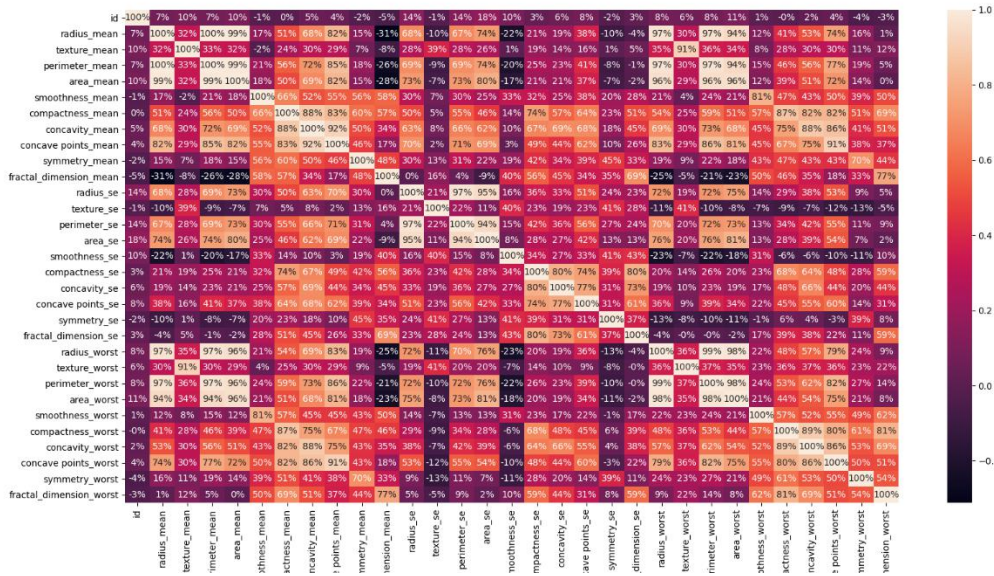
[illegible]

### Visualize the correlation:

The following code will generate a correlation heatmap between different columns.

```
38 #visualize the correlation
39 plt.figure(figsize=(20,20))
40 sns.heatmap(df.corr(), annot=True, fmt='.0%')
41 plt.show()
```

[output] :



## 7.2. Splitting the data:

In this step we can split the data that will be used for training the Machine learning models.

### Split the data:

Splitting the dataset into two parts for testing and training purposes, 75 percent will be used to train the model and on the rest of 25 percent will check the accuracy of how the model has performed. We will compare the test data with the outputs from our model to check the accuracy.

```
43 X = df.iloc[:, 2:31].values
44 Y = df.iloc[:, 1].values
45
46
47 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
48
```

### Features scaling:

Scale the data to bring all features to the same level of magnitude, which means the feature / independent data will be within a specific range for example 0–100 or 0–1.

```
50 #Feature Scaling
51
52 sc = StandardScaler()
53 X_train = sc.fit_transform(X_train)
54 X_test = sc.transform(X_test)
55
```

### 7.3. Creating and training the models:

In this step we will create the models and train them according to the dataset splitted in **Step 2**.

#### Creating the models:

Create a function to hold many different models (e.g. Logistic Regression, Decision Tree Classifier, Random Forest Classifier) to make the classification. These are the models that will detect if a patient has cancer or not. Within this function, I will also print the accuracy of each model on the training data

```
62 def models(X_train, Y_train):
63     # Using Logistic Regression
64
65     logistics = LogisticRegression(random_state=0)
66     logistics.fit(X_train, Y_train)
67
68     # Using DecisionTreeClassifier
69
70     Dtree = DecisionTreeClassifier(criterion='entropy', random_state=0)
71     Dtree.fit(X_train, Y_train)
72
73     # Using RandomForestClassifier
74
75     RFC = RandomForestClassifier(n_estimators=45, criterion='entropy', random_state=0)
76     RFC.fit(X_train, Y_train)
77
78     # print model accuracy on the training data.
79     print('[0]Logistic Regression Training Accuracy:', logistics.score(X_train, Y_train))
80     print('[1]Decision Tree Classifier Training Accuracy:', Dtree.score(X_train, Y_train))
81     print('[2]Random Forest Classifier Training Accuracy:', RFC.score(X_train, Y_train))
82
83     return logistics, Dtree, RFC
```

#### Running the models:

Create the model that contains all of the models and look at the accuracy score on the training data for each model to classify if a patient has cancer or not.

```
85     #running the model with the pre processed data
86     model = models(X_train,Y_train)
87
```

### 7.4. Testing the models:

In this step we will test the models by checking its accuracy and creating a confusion matrix so that we know about how much trust we can put in this model.

#### Confusion matrix and accuracy:



The confusion matrix tells us how many patients each model misdiagnosed (number of patients with cancer that were misdiagnosed as not having cancer also known as a false negative, and the number of patients who did not have cancer that was misdiagnosed with having cancer also known as false positive) and the number of correct diagnoses, the true positives, and true negatives.

False Positive (FP) = A test result that incorrectly indicates that a particular condition or attribute is present.

True Positive (TP) = Sensitivity (also called the true positive rate, or probability of detection in some fields) measures the proportion of actual positives that are correctly identified as such.

True Negative (TN) = Specificity (also called the true negative rate) measures the proportion of actual negatives that are correctly identified as such.

False Negative (FN) = A test result that indicates that a condition does not hold, while it does. For example, a test result that indicates a person does not have cancer when the person does have it

```
88     # Testing Accuracy with Confusion Matirx
89     for i in range(len(model)):
90         Cmartix = confusion_matrix(Y_test, model[i].predict(X_test))
91
92         TN = Cmartix[0][0]
93         TP = Cmartix[1][1]
94         FN = Cmartix[1][0]
95         FP = Cmartix[0][1]
96
97
98         print('Model[{}] Testing Accuracy = "{}!"'.format(i, (TP + TN) / (TP + TN + FN + FP)))
99         print(Cmartix)
100        print("=====")
```

### Output of the program:

The following output shows the confusion matrix and the accuracy of the models on the test data.

```
Run: basic x
C:\Users\Dexter\PycharmProjects\MachineLearning\venv\Scripts\python.exe C:/Users/Dexter/PycharmProjects/MachineLearning/basic.py
[0]Logistic Regression Training Accuracy: 0.9906103286384976
[1]Decision Tree Classifier Training Accuracy: 1.0
[2]Random Forest Classifier Training Accuracy: 1.0
Model[0] Testing Accuracy = "0.9440559440559441!"
[[86  4]
 [ 4 49]]
=====
Model[1] Testing Accuracy = "0.951048951048951!"
[[84  6]
 [ 1 52]]
=====
Model[2] Testing Accuracy = "0.972027972027972!"
[[87  3]
 [ 1 52]]
=====
Process finished with exit code 0
```

## 8. Models:

Following are the models used:

### 8.1. Logistics Regression:

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

Followings are the types of logistic regression:

- Binary (e.g. Tumor Malignant or Benign)
- Multi-linear functions fail Class (e.g. Cats, dogs or Sheep's)

We will use Binary Logistics Regression to perform analysis.

We used sklearn's Python Library to implement this Model and get the predictions.

```
63         # Using Logistic Regression
64
65         logistics = LogisticRegression(random_state=0)
66         logistics.fit(X_train, Y_train)
67
```

[output]:

```
[0]Logistic Regression Training Accuracy: 0.9906103286384976

Model[0] Testing Accuracy = "0.9440559440559441!"
[[86  4]
 [ 4 49]]
```

Its Training Accuracy is 99% where Testing accuracy is 94%

## 8.2. Decision Tree Classifier:

Decision Tree Analysis is a general, predictive modeling tool that has applications spanning several different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

We used sklearn's Python Library to implement this Model and get the predictions.

```
68 # Using DecisionTreeClassifier
69
70 Dtree = DecisionTreeClassifier(criterion='entropy', random_state=0)
71 Dtree.fit(X_train, Y_train)
```

[output]:

```
[1]Decision Tree Classifier Training Accuracy: 1.0
```

```
Model[1] Testing Accuracy = "0.951048951048951!"
```

```
[[84 6]
 [ 1 52]]
```

Its Training Accuracy is 100% where Testing accuracy is 95%

## 8.3. Random Forest Classifier:

Random forest is a supervised learning algorithm that is used for both classifications as well as regression. However, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees mean more robust forests. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution using voting. It is an ensemble method that is better than a single decision tree because it reduces the over-fitting by averaging the result.

How it works:

- Pick at random K data points from the training set.
- Build the decision tree associated with those K data points.
- Choose the number n tree of trees you want to build and repeat steps 1 & 2.



- For a new data point, make each one of your n tree trees predict the value of Y for the data point, and assign the new data point the average across all of the predicted Y values.

We used sklearn's Python Library to implement this Model and get the predictions.

```

73     # Using RandomForestClassifier
74
75     RFC = RandomForestClassifier(n_estimators=45, criterion='entropy', random_state=0)
76     RFC.fit(X_train, Y_train)

```

**[output]:**

```
[2]Random Forest Classifier Training Accuracy: 1.0
```

```
Model[2] Testing Accuracy = "0.972027972027972!"
```

```
[[87  3]
 [ 1 52]]
```

Its Training Accuracy is 100% where Testing accuracy is 97%

## 9. Model Comparison:

Followings points shows the comparison between models:

- Logistic Regression with Testing accuracy 94% with 4 False Positive (FP)and 4 False Negative (FN)
- Decision Tree Classifier with Testing accuracy 95% with 6 False Positive (FP)and 1 False Negative (FN)
- Random Forest Classifier with Testing accuracy 97% with 3 False Positive (FP)and 1 False Negative (FN)

From the above accuracy and metrics, the best performing model in the test data is the Random Forest Classifier, which has an accuracy score of 97%. So, I will prefer that model to detect cancer cells in patients. Categorize test data and show both the random forest classification model assessment and the actual values of the patient showing cancer.

## **10. Conclusion:**

On the bases of the above representations and modeling approaches, it is clear that the random forest classifier gives the most accurate output. By using this, one can detect breast cancer in its early stage to save the lives of women.

I notice the model, misdiagnosed a few patients as having cancer when they didn't and it misdiagnosed patients that did have cancer as not having cancer. Although this model is good, when dealing with the lives of others I want this model to be better and get its accuracy as close to 100% as possible or at least as good as if not better than doctors

## **11. Reference:**

Followings are the reference which helped us in this project

- [1] Dataset available at <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data>
- [2] Documentation for library used: [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html)
- [3] Support: <http://libanswers.bcu.ac.uk/>