## Breast Cancer Detection with Logistic Regression

```
#importing libraries
import numpy as np
import sklearn.datasets


#getting the dataset
breast_cancer = sklearn.datasets.load_breast_cancer()
#https://www.kaggle.com/ucim/breast-cancer-wisconsin-data
print(breast_cancer)
```

```
     {'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
             1.189e-01],
            [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
             8.902e-02],
            [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
             8.758e-02],
            ...,
            [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
             7.820e-02],
            [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
             1.240e-01],
            [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
             7.039e-02]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
            0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
            1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
            1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
            1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
            0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
            1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
            0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
            1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
            1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
            0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
            0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
            1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
            1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
            1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
            1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
            1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
            1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]), 'target_names': array(
            'mean smoothness', 'mean compactness', 'mean concavity',
            'mean concave points', 'mean symmetry', 'mean fractal dimension',
            'radius error', 'texture error', 'perimeter error', 'area error',
            'smoothness error', 'compactness error', 'concavity error',
```

```
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'), 'filename': '/usr/lo
```

```
X = breast_cancer.data
Y = breast_cancer.target
```

```
print(X)
print(Y)
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

```
print(X.shape, Y.shape)
```

```
(569, 30) (569,)
```

## Import data to the Pandas Data Frame

```
import pandas as pd
```

```
data = pd.DataFrame(breast_cancer.data, columns = breast_cancer.feature_names)
```

```
data['class'] =  breast_cancer.target
```

```
data.head()
```

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | symm |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 | 0. |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 | 0. |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 | 0. |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 | 0. |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.1980 | 0.10430 | 0. |

```
data.describe()
```

| | concave points error | symmetry error | fractal dimension error | worst radius | worst texture | worst perimeter | worst area | wo smoothn |
|---|---|---|---|---|---|---|---|---|
| 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000 |
| 0.011796 | 0.020542 | 0.003795 | 16.269190 | 25.677223 | 107.261213 | 880.583128 | 0.13 |
| 0.006170 | 0.008266 | 0.002646 | 4.833242 | 6.146258 | 33.602542 | 569.356993 | 0.02 |
| 0.000000 | 0.007882 | 0.000895 | 7.930000 | 12.020000 | 50.410000 | 185.200000 | 0.07 |
| 0.007638 | 0.015160 | 0.002248 | 13.010000 | 21.080000 | 84.110000 | 515.300000 | 0.11 |
| 0.010930 | 0.018730 | 0.003187 | 14.970000 | 25.410000 | 97.660000 | 686.500000 | 0.13 |
| 0.014710 | 0.023480 | 0.004558 | 18.790000 | 29.720000 | 125.400000 | 1084.000000 | 0.14 |
| 0.052790 | 0.078950 | 0.029840 | 36.040000 | 49.540000 | 251.200000 | 4254.000000 | 0.22 |

```
print(data['class'].value_counts())
```

```
1    357
0    212
Name: class, dtype: int64
```

```
print(breast_cancer.target_names)
```

```
['malignant' 'benign']
```

```
data.groupby('class').mean()
#0 = malignant
#1 = benign
```

| class | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity |
|---|---|---|---|---|---|---|---|
| 0 | 17.462830 | 21.604906 | 115.365377 | 978.376415 | 0.102898 | 0.145188 | 0.160775 |
| 1 | 12.146524 | 17.914762 | 78.075406 | 462.790196 | 0.092478 | 0.080085 | 0.046058 |

## Train and Test Data Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y)
```

```
print(Y.shape, Y_train.shape, Y_test.shape)
```

```
(569,) (426,) (143,)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1)
#test_size --> to specify the percentage of test data needed
```

```
print(Y.shape, Y_train.shape, Y_test.shape)
```

```
(569,) (512,) (57,)
```

```
print(Y.mean(), Y_train.mean(), Y_test.mean())
```

```
0.6274165202108963 0.630859375 0.5964912280701754
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1, stratify=Y)
#stratify --> for correct distribution of data as of the original data
```

```
print(Y.mean(), Y_train.mean(), Y_test.mean())
```

```
0.6274165202108963 0.626953125 0.631578947368421
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1, stratify=Y, random_s
#random_state --> specific split of data, each value of random_state splits the data differen
```

```
print(X_train.mean(), X_test.mean(), X.mean())
```

```
61.31637960106119 67.04963097269005 61.890712339519624
```

```
print(X_train)
```

```
[[1.490e+01 2.253e+01 1.021e+02 ... 2.475e-01 2.866e-01 1.155e-01]
 [1.205e+01 1.463e+01 7.804e+01 ... 6.548e-02 2.747e-01 8.301e-02]
 [1.311e+01 1.556e+01 8.721e+01 ... 1.986e-01 3.147e-01 1.405e-01]
 ...
 [1.258e+01 1.840e+01 7.983e+01 ... 8.772e-03 2.505e-01 6.431e-02]
 [1.349e+01 2.230e+01 8.691e+01 ... 1.282e-01 2.871e-01 6.917e-02]
 [1.919e+01 1.594e+01 1.263e+02 ... 1.777e-01 2.443e-01 6.251e-02]]
```

## Logistic Regression

```
#import logistic regression from sklearn
from sklearn.linear_model import LogisticRegression
```

```
classifier = LogisticRegression()   #loading the logistic regression model to the variable "c
```

```
#training the model on training data
classifier.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Convergenc
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

## Evaluation of the Model

```
#import accuracy_score
from sklearn.metrics import accuracy_score
```

```
prediction_on_training_data = classifier.predict(X_train)
accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)
```

```
print('Accuracy on training data :', accuracy_on_training_data)
```

```
Accuracy on training data : 0.951171875
```

```
#prediction on test_data
prediction_on_test_data = classifier.predict(X_test)
accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)


print('Accuracy on test data :', accuracy_on_test_data)
```

```
Accuracy on test data : 0.9298245614035088
```

## Detecting whether the Patient has Breast Cancer in Benign or Malignant Stage

```
input_data = (17.99,10.38,122.8,1001,0.1184,0.2776,0.3001,0.1471,0.2419,0.07871,1.095,0.9053,
#change the input_data to numpy_array to make prediction
input_data_as_numpy_array = np.array(input_data)
print(input_data)
```

```
(17.99, 10.38, 122.8, 1001, 0.1184, 0.2776, 0.3001, 0.1471, 0.2419, 0.07871, 1.095, 0.90
```

```
#reshape the array we are predicting the output for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)


#prediction
prediction = classifier.predict(input_data_reshaped)
print(prediction)   #returns a list with elements 0 [if Malignant] and 1 [if Benign]
```

```
[0]
```

```
if (prediction[0]==0):
    print("The Breast Cancer is at Malignant Stage.")
else:
    print("The Breast Cancer is at Benign Stage.")
```

```
The Breast Cancer is at Malignant Stage.
```

✓ 0s completed at 1:11 PM