ession-a-machine-learning-approach

July 18, 2024

1 Breast Cancer Prediction Using Logistic Regression: A Machine Learning Approach –Vignesh Prabhu

Breast cancer is one of the most common cancers affecting women worldwide. Early detection and accurate prediction are crucial for effective treatment and better patient outcomes. This project utilizes logistic regression, a powerful machine learning technique, to predict the likelihood of breast cancer. By analyzing relevant medical data, our model aims to assist healthcare professionals in making informed decisions, ultimately contributing to improved patient care and survival rates

Importing the Dependencies

[1]:
```
import numpy as np
import pandas as pd
import sklearn.datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection & Processing

[2]:
```
# loading the data from sklearn
cancer_dataset = sklearn.datasets.load_breast_cancer()
```

[ ]:
```
print(cancer_dataset)
```

[4]:
```
# loading the data to a data frame
data_frame = pd.DataFrame(cancer_dataset.data, columns = cancer_dataset.
 ↪feature_names)
```

[5]:
```
# print the first 5 rows of the dataframe
data_frame.head()
```

[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

```
   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  …  worst radius  worst texture  worst perimeter  \
0                 0.07871  …         25.38          17.33           184.60
1                 0.05667  …         24.99          23.41           158.80
2                 0.05999  …         23.57          25.53           152.50
3                 0.09744  …         14.91          26.50            98.87
4                 0.05883  …         22.54          16.67           152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

[6]:
```
# adding the 'target' column to the data frame
data_frame['label'] = cancer_dataset.target
```

[7]:
```
# print last 5 rows of the dataframe
data_frame.tail()
```

[7]:
```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
564        21.56         22.39          142.00     1479.0          0.11100
565        20.13         28.25          131.20     1261.0          0.09780
566        16.60         28.08          108.30      858.1          0.08455
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263

     mean compactness  mean concavity  mean concave points  mean symmetry  \
564           0.11590         0.24390              0.13890         0.1726
565           0.10340         0.14400              0.09791         0.1752
```

```
566          0.10230         0.09251              0.05302         0.1590
567          0.27700         0.35140              0.15200         0.2397
568          0.04362         0.00000              0.00000         0.1587

     mean fractal dimension  …  worst texture  worst perimeter  worst area  \
564                 0.05623  …          26.40           166.10      2027.0
565                 0.05533  …          38.25           155.00      1731.0
566                 0.05648  …          34.12           126.70      1124.0
567                 0.07016  …          39.42           184.60      1821.0
568                 0.05884  …          30.37            59.16       268.6

     worst smoothness  worst compactness  worst concavity  \
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000

     worst concave points  worst symmetry  worst fractal dimension  label
564                0.2216          0.2060                  0.07115      0
565                0.1628          0.2572                  0.06637      0
566                0.1418          0.2218                  0.07820      0
567                0.2650          0.4087                  0.12400      0
568                0.0000          0.2871                  0.07039      1

[5 rows x 31 columns]
```

[8]:
```
# number of rows and columns in the dataset
data_frame.shape
```

[8]:
```
(569, 31)
```

[ ]:
```
# getting some information about the data
data_frame.info()
```

[10]:
```
# checking for missing values
data_frame.isnull().sum()
```

[10]:
```
mean radius               0
mean texture              0
mean perimeter            0
mean area                 0
mean smoothness           0
mean compactness          0
mean concavity            0
mean concave points       0
mean symmetry             0
```

```
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
label                      0
dtype: int64
```

[ ]:
```
# statistical measures about the data
data_frame.describe()
```

[12]:
```
# checking the distribution of Target Varibale
data_frame['label'].value_counts()
```

[12]:
```
label
1    357
0    212
Name: count, dtype: int64
```

[13]:
```
data_frame.groupby('label').mean()
```

[13]:
```
       mean radius  mean texture  mean perimeter   mean area  mean smoothness  \
label
0        17.462830     21.604906      115.365377  978.376415         0.102898
1        12.146524     17.914762       78.075406  462.790196         0.092478

       mean compactness  mean concavity  mean concave points  mean symmetry  \
label
0              0.145188        0.160775             0.087990       0.192909
1              0.080085        0.046058             0.025717       0.174186
```

```
       mean fractal dimension  …  worst radius  worst texture  \
label                          …
0                    0.062680  …     21.134811      29.318208
1                    0.062867  …     13.379801      23.515070

       worst perimeter   worst area  worst smoothness  worst compactness  \
label
0           141.370330  1422.286321          0.144845           0.374824
1            87.005938   558.899440          0.124959           0.182673

       worst concavity  worst concave points  worst symmetry  \
label
0             0.450606              0.182237        0.323468
1             0.166238              0.074444        0.270246

       worst fractal dimension
label
0                     0.091530
1                     0.079442

[2 rows x 30 columns]
```

Separating the features and target

[14]:
```
X = data_frame.drop(columns='label', axis=1)
Y = data_frame['label']
```

[ ]:
```
print(X)
```

[ ]:
```
print(Y)
```

Splitting the data into training data & Testing data

[17]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
↪random_state=2)
```

[18]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(569, 30) (455, 30) (114, 30)
```

Model Training

[19]:
```
model =LogisticRegression(solver='lbfgs', max_iter=3000)
```

[26]:
```
# training the Logistic Regression model using Training data

model.fit(X_train, Y_train)
```

[26]:
```
LogisticRegression(max_iter=3000)
```

Model Evaluation

Accuracy Score

[27]:
```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

[22]:
```
print('Accuracy on training data = ', training_data_accuracy)
```

```
Accuracy on training data =  0.9692307692307692
```

[28]:
```
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

[24]:
```
print('Accuracy on test data = ', test_data_accuracy)
```

```
Accuracy on test data =  0.9298245614035088
```

Building a Predictive System

[25]:
```
input_data = (13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885,0.
↪05766,0.2699,0.7886,2.058,23.56,0.008462,0.0146,0.02387,0.01315,0.0198,0.
↪0023,15.11,19.26,99.7,711.2,0.144,0.1773,0.239,0.1288,0.2977,0.07259)

# 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 datapoint
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The Breast cancer is Malignant')

else:
  print('The Breast Cancer is Benign')
```

```
[1]
The Breast Cancer is Benign
```

This project demonstrates that logistic regression can effectively predict breast cancer, aiding early detection and improving patient outcomes. The model's accuracy and interpretability make it a valuable tool for healthcare professionals, highlighting the potential of machine learning in advancing medical diagnosis.