Breast Cancer Classification

StartOnAI Machine Learning Tutorial 2

This is a tutorial which navigates the process of building a simple Breast Cancer Classifier

What is a Breast Cancer Classifier and how do we approach it?

A breast cancer classifier is a program which will predict malignancy in breast. The classifier can only have two outputs, benign or malignant, so we must utilize an algorithm which produces a binary output. In this case we will be using a logistic regression, an algorithm which produces an output between 0 and 1, unlike a linear regression which can produce values that exceed 0 and 1. For a better visualization see Figure 1.

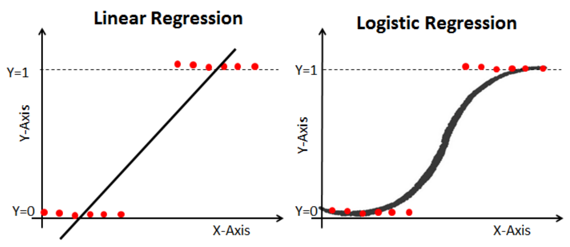


Figure 1: Linear Regression vs. Logistic Regression

What is a Logistic Regression and how does it work?

Logistic regression is a technique which predicts a binary output rather than a continuous one like a linear regression does. A logistic regression fits a *sigmoid* function or an “S”” shaped function to the data as seen in Figure 1. You might be wondering how does the algorithm predict a binary output if the function has many intermediate values between 0 and 1. To understand this we must consider the graph for a logistic regression.

$$f(x) = \frac{1}{1+e^{-x}}$$

Figure 2: Sigmoid Function

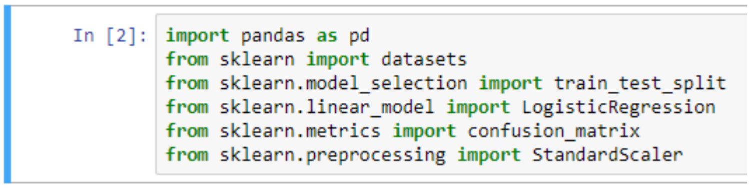
We know Y axis is the dependent variable and the X axis as the independent variable. Based on the independent variable we try to predict what the dependent variable might be. When using logistic regression, we consider the values on the Y axis as the probability of an event to occur. So to produce a binary result and avoid intermediate values we create a threshold, for example, 0.5. If the a data point is located at 0.5 or above we classify it as a 1 and if it is below 0.5 we classify it as a 0.

The use of a sigmoid function makes sense because it contains asymptotes at $y = 0$ and $y = 1$ meaning we can't ever be 100% sure that an event is going to occur or not occur. To really drive the concept home we can use the example of a student taking a pass or fail course. Here we consider hours studied against whether the student will pass or fail. Lets assume after 10 hours of studying the chance he will pass is 50% so a Y value of 0.5. If he studies more than 10 hours the chance he will pass increase and increases until a point where the cost of studying isn't worth the very slight increase in the chance he passes. At some point the dependent variable is not as likely to change the output and that is another reason why we use a logistic regression because it is able to accurately consider diminishing returns.

Please take a look at *Introduction to Logistic Regression* in *More Resources* for an in depth look into the mathematical concepts behind logistic regression.

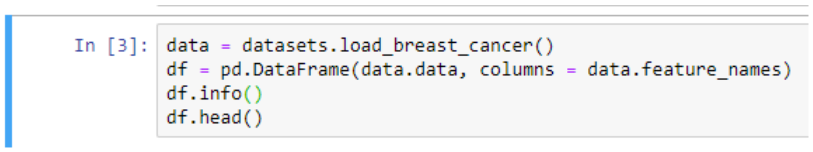
How do we build a Breast Cancer Classifier?

1. We begin by importing the necessary libraries as shown below. The data set we will be using is by the University of Wisconsin hospital and is available in **sklearn**.



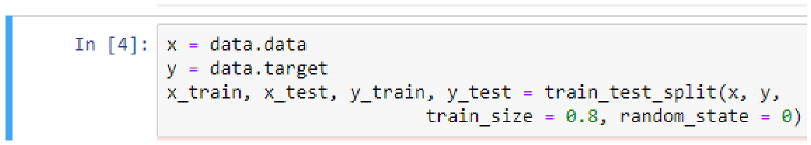
```
In [2]: import pandas as pd
        from sklearn import datasets
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import confusion_matrix
        from sklearn.preprocessing import StandardScaler
```

2. Next we import our data from **sklearn** and load it into a pandas DataFrame so we can easily work with the data. Then we use **.info()** and **.head()** to familiarize ourselves with the data we are working with. After running these function we can see the features we will have to consider and the size of the data set. See More Resources for the output.



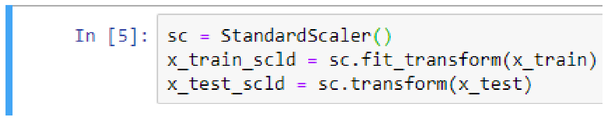
```
In [3]: data = datasets.load_breast_cancer()
        df = pd.DataFrame(data.data, columns = data.feature_names)
        df.info()
        df.head()
```

3. We can also initialize our independent and dependent variables and then utilize the **train_test_split** function [1] to split our data into testing and training data.



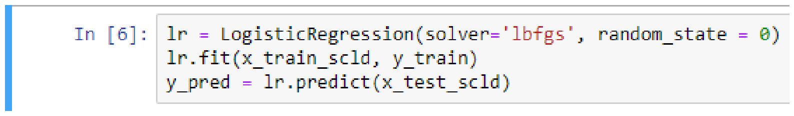
```
In [4]: x = data.data
        y = data.target
        x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                train_size = 0.8, random_state = 0)
```

4. Our next step is feature scaling or the scaling the various features in a data set. Feature scaling is when there is a data set with varying magnitudes and instead of letting a single feature dominate our prediction we scale that feature's magnitude down so no feature is too dominant over the others.



```
In [5]: sc = StandardScaler()
        x_train_scld = sc.fit_transform(x_train)
        x_test_scld = sc.transform(x_test)
```

5. Now we build our logistic regression model and pass in our training data. For more information on the solver parameter please see this answer by stack overflow user Yahya.



```
In [6]: lr = LogisticRegression(solver='lbfgs', random_state = 0)
        lr.fit(x_train_scld, y_train)
        y_pred = lr.predict(x_test_scld)
```

6. Our final step is testing the accuracy our regression, for this we will use a confusion matrix. A confusion matrix consists of 2 by 2 matrix with the first column represents our predicted negative results, the second column representing our predicted positive results, the first row representing the actual negative results, and the second row representing the actual positive results. A confusion matrix can help calculate recall, precision, and accuracy, all very important metrics for measuring how well an algorithm performs.

[1]Running this may give a FutureWarning, it is warning us that we haven't explicitly declared the test size but sklearn automatically will handle it.

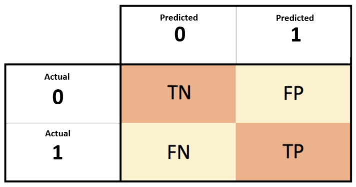
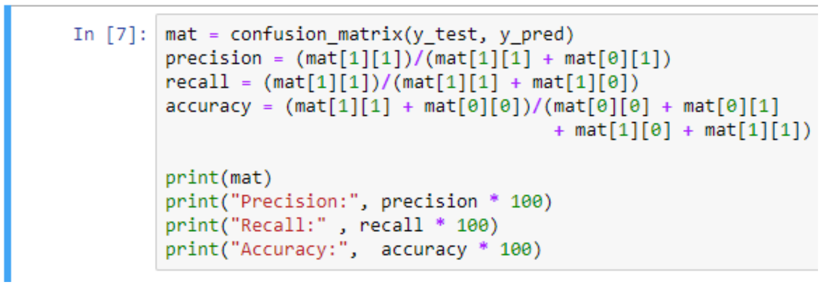


Figure 3: Confusion Matrix

A confusion matrix gives us 4 values: true negative, false positive, false negative, and true positive. True negative means what you predicted is positive and it is true, in our case an we predict that an individual has a malignant tumor and actually has one. False positive means we predicted that the cancer was benign but was actually malignant, false negative means that we predicted that the cancer was malignant but was actually benign, and true positive means we predicted it was benign and it was actually benign. With this information we can calculate the recall, precision, and accuracy. See More Resources for the output.



```
In [7]: mat = confusion_matrix(y_test, y_pred)
        precision = (mat[1][1])/(mat[1][1] + mat[0][1])
        recall = (mat[1][1])/(mat[1][1] + mat[1][0])
        accuracy = (mat[1][1] + mat[0][0])/(mat[0][0] + mat[0][1]
                                             + mat[1][0] + mat[1][1])


        print(mat)
        print("Precision:", precision * 100)
        print("Recall:" , recall * 100)
        print("Accuracy:",  accuracy * 100)
```

Precision, Recall, and Accuracy

$$\frac{TP}{TP+FP}$$

Precision Formula

$$\frac{TP}{TP+FN}$$

Recall Formula

$$\frac{TP+TN}{TP+TN+FP+FN}$$

Accuracy Formula

Precision is the total number of correct positive classifications from the cases which were predicted to be positive. For precision we use the true positive as the numerator and both true positive and false positive as the denominator because these were the cases that were predicted to be positive. Recall is the total number of correct positive classifications from the cases which are actually positive. For recall we use true positive as the numerator but this time we use true positive and false negative as the denominator because we want to account for positive cases which the algorithm missed. Accuracy is the proportion of correct classification from all the cases, therefore we use all the cases which our algorithm considered positive over all the cases in total.

More Resources

1. More information on logistic regression
   (a) Introduction to Logistic Regression by Ayush Pant
   (b) Understanding Logistic Regression in Python by Avinash Navlani
   (c) Logistic Regression Explained by Jaime Zornoza
2. More information on confusion matrices
   (a) Understanding Confusion Matrix by Sarang Narkhede
   (b) Decoding the Confusion Matrix by Prateek Sharma
3. Code Walkthrough Outputs

References

1. Goel, Vishabh. "Building a Simple Machine Learning Model on Breast Cancer Data." Medium, Towards Data Science, 12 Oct. 2018, towardsdatascience.com/building-a-simple-machine-learning-model-on-breast-cancer-data-eca4b3b99fa3.
2. Allen, Michael. "66. Machine Learning. Your First Ml Model! Using Logistic Regression to Diagnose Breast Cancer." Python for Healthcare Modelling and Data Science, 15 June 2018, pythonhealthcare.org/2018/04/15/66-machine-learning-your-first-ml-model-using-logistic-regression-to-diagnose-breast-cancer/.