Classifying Tumors with Logistic Regression

# The Problem

Given a dataset, create a machine learning model that is able to predict the likelihood that a new sample is either a malignant or benign tumor.

# The Purpose

The purpose of this paper is to take a systematic walkthrough of both the decision-making process and execution of the machine learning model in order that the reader may understand exactly how the classification prediction was created.

# The Data

From the UCI Machine Learning Repository:

Each record represents follow-up data for one breast cancer case. These are consecutive patients seen by Dr. Wolberg since 1984, and include only those cases exhibiting invasive breast cancer and no evidence of distant metastases at the time of diagnosis.   
  
The first 30 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.

Attribute Information: (class attribute has been moved to last column)

# Attribute Domain

-- -----------------------------------------

1. Sample code number id number

2. Clump Thickness 1 - 10

3. Uniformity of Cell Size 1 - 10

4. Uniformity of Cell Shape 1 - 10

5. Marginal Adhesion 1 - 10

6. Single Epithelial Cell Size 1 - 10

7. Bare Nuclei 1 - 10

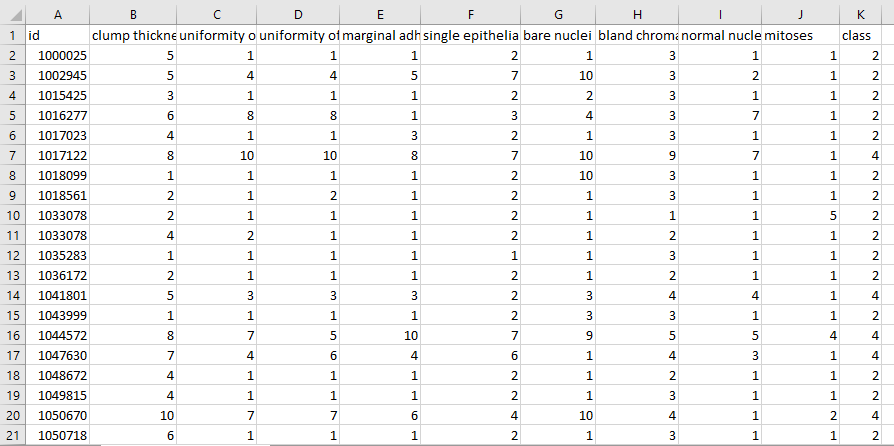
8. Bland Chromatin 1 - 10

9. Normal Nucleoli 1 - 10

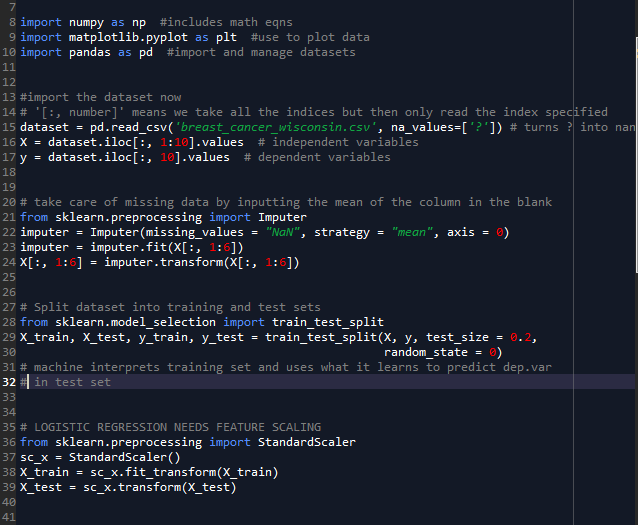
10. Mitoses 1 - 10

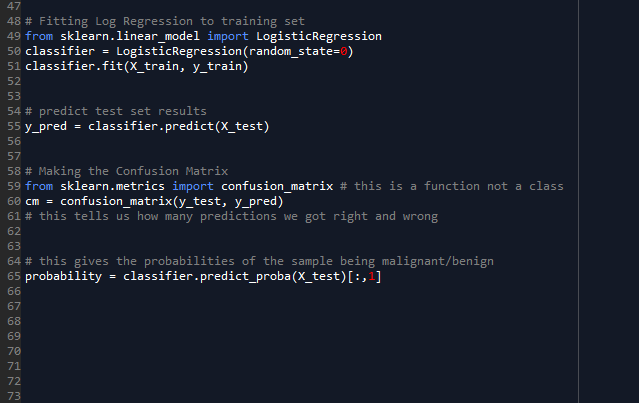
11. Class: (2 for benign, 4 for malignant)

Here is a sample of what the data look like:



# The Code





# Walking Through the Code

Lines 9-10:

These lines import libraries that are commonly used in machine learning. In this case we will not actually use the matplotlib library as that is used for plotting data. Since we have ten inputs we have ten dimensions which cannot be visualized without dimension reduction. While using dimension-reducing techniques like Principal Compent Analysis is possible, it would not be worth potentially increasing variance in order to visualize the data. Instead, we can just look at the numerical accuracies of the model.

Lines 13-17:

This is where the data from the .csv file is imported using the pandas library. The first ten columns are made to be the independent variables while the last column is the dependent variable. Adding “na\_values = [‘?’]” is an attribute that is an initial step to take care of missing data. Often times datasets come with missing data, and they can be filled with a number of different placeholders. In this case, the file uses ‘?’. However, with this code they are changed into “NaN” (not a number), to be formatted to be used with the Imputer class later on.

Lines 20-24:

These lines actually replace the missing data. The imported Imputer class finds missing data and can replace them with a number of substitutes, two of which being the mean or median of that particular column or row. The strategy attribute is where you decide what method you want to use to replace the data. Since the data all come in ranges between one and ten, it made more sense to use the mean since there was no possibility of outliers skewing the result. The axis attribute simply refers to if it is finding the mean of columns or rows. “axis=0” uses columns, “axis=1” uses rows. The imputer object was created using the Imputer class and was then fit to the independent variables. There is actually only one column with missing data in the dataset but the fit and transform methods in lines 23-24 require matrices and not vectors, so that is why there is a range of indices. The fit method fits the imputer object to our independent variables (in this case, columns one through five, but it can be any range) and the transform method actually edits our X variable to reflect the changes that have been made to the missing values.

Lines 27-32:

These lines are relatively straightforward. There needs to be a training set for our machine learning model to learn from, then a test set to test our model on new data to see how it performs. Four variables are defined at once using this method. “test\_size=0.2” means we use 80% of the data for the training set and 20% of the data for the test set. The random\_state attribute is unnecessary but helps with bug-checking replicated code by taking the same random instances from the dataset.

Lines 35-39:

This is the last data preprocessing step in order to prepare the data to be fit to our classification model. Feature Scaling is used to ensure that Euclidian distances between instances are not skewed as to make one variable dominant in how much it affects the instance. A common example is if one were to plot Age vs Salary. Without feature scaling, salaries would have a disproportionate level of influence on regression or classification models because it they are so much larger than age. Most machine learning models come with feature scaling, but logistic regression, the model used here, does not include it automatically and must have it added.

Lines 48-51:

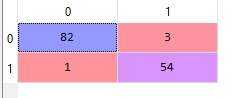
The logistic regression model is finally made using the training set data. The LogisticRegression class is imported from the Sci-kit learn library, then an object of the class is created. Lastly, the object is fit to the training set data.

Lines 54-55:

After the logistic regression model is made, we can use it to predict the outcomes of the test set. We create our model based on the training set, then test to see how good the model is on the test set. The model takes new input from X\_test and applies it to the model, giving output into the variable y\_pred. The predicted results can be compared to the actual results of y\_test to see how the model performed.

Lines 58-61:

These lines create a confusion matrix. A confusion matrix creates a visual to easily see how well the model did. The top left corner is True Negative, top right is False Positive (Type I error), bottom left is False Negative (Type II error), and the bottom right is True Positive.



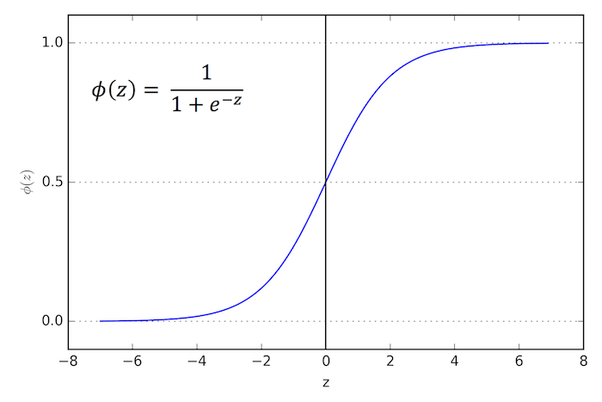
Here, it can easily be seen that the model did well: 136 correct predictions, 4 incorrect predictions. In addition, there was only one False Negative, which is especially important in the medical field.

Lines 64-65:

This adds another aspect for further analysis; it provides the model’s calculated likelihood of the instance being either malignant or benign. If the model predicts over a 50% likelihood, it will classify the instance as malignant.

# Analysis and Reasoning

There is not a large amount to analyze about the results besides a couple of quick calculations. Clearly, the model performed very well with an accuracy of 97.14%. In addition, its False negative rate is 1.85%. There are many more ways to analyze the model performance but those are the two that I think are the most important here. In the medical field, overall accuracy is important as is avoiding false negatives. False negatives have the potential to prevent people from receiving treatment when there likelihood of recovery is dramatically improved the earlier they start. This concept is also why a logistic regression model was chosen. Logistic regression is a method of classification that is also probabilistic and takes into account the stochasticity of data. A logistic regression model looks like this:



The y-axis represents probabilities. The logistic regression model will take all the inputs (in our case, ten) and give an output between 0 and 1. However, our dependent variable is binary and categorical (yes or no, or 1 or 0) so how does the model predict a 1 or 0 if its output is neither? The output is a probability of a category, so if the probability of the category being 1 is more than 50%, it will choose 1, and vice versa. This is especially important for this particular data set because a doctor may decide a patient should start treatment if there is a 20%, 30%, or 40% likelihood of a tumor being malignant. Using this model and including the last piece of code allows a doctor to look at the likelihood of a tumor being malignant and make an informed decision from there.

# Acknowledgments

I got the data from the UCI Machine Learning Repository. Here is their pseudo APA citation they ask us to include:

Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.