Exploring the Heart Disease dataset

# Introduction

Have we all experienced a creative block when we get our hands on a new dataset? We did haven’t we! There is only one possible solution to this, jump into the deep-end of the metaphorical pool of data analysis and get our hands dirty. Sometimes the best way to get out of a writing block is to just start writing, anything and everything that comes to mind. Later we end up realizing that what we wrote in fact makes sense. In the same way, to stop a ‘coding’ block from festering any further, there is no better way other than to read in the dataset, and just start putting in some code. Rest assured, we will get our notebook (jupyter notebook, I hope) into shape in no time.

In this post, the goal is to walkthrough the process of implementing a simple supervised classification algorithm, that classifies between 2 classes. What this post is not is a data munging or wrangling tutorial. I will deal with this concept at a later stage in a different blog post. Will link it below as soon as it is done, don’t worry. If you are a beginner and want to just do something, that works, and looks like data analysis, then read on.

Looking around in the Kaggle datasets repository, this UCI dataset containing presence and absence of heart disease caught my eye. The dataset is clean, as in all variables are recoded ( are in numeric format) and there is a target variable to predict. Convenient!

# Background of the dataset

This dataset was prepared and subset(ed) from an original 76 attribute strong database. Lots of other variables were dropped keeping only those that seem significant or relevant to model the target. Of all the published studies, only these 14 variables are referred to out of the 76. Looks like the work that needs to be done is cut out for us.

The variables present are (in order of appearance):

1. Age
2. Sex
3. Chest pain type (4 values) ‘CP’
4. Resting blood pressure ‘trbps’
5. Serum cholesterol in mg/dl ‘chol’
6. Fasting blood sugar >120 mg/dl ‘fbs’
7. Resting electrocardiographic results ‘restecg’
8. Maximum heart rate achieved ‘thalach’
9. Exercise induced Angina ‘exang’
10. ST depression induced by exercise relative to rest ‘old peak’
11. The slope of the peak exercise ST segment ‘slope’
12. Number of major vessels (0-3) coloured by fluoroscopy ‘ca’
13. 3=normal, 6=fixed defect, 7=reversible defect ‘thal’
14. ‘Target’ 1=heart disease, 0=no heart disease

# Analysis

I decided to use logistic regression out of the box. To check how well the model works in predicting the target, without any transformations to the independent variables. Using statsmodels package a Logit model is prepared easily enough. Just remember to add an intercept term to the data frame. This is necessary for the statsmodels package, as it is not done automatically. Puzzling, I know. This is one of the boiler plate things that needs to be checked. The stats models package was created by statisticians for statisticians. It is not meant for production level code, unlike sklearn. But we are getting ahead of ourselves, let’s hold off this discussion for later.

## Performance

Before going ahead and interpreting the logistic model output, let us see how well our model performs. How capable is our model in correctly predicting which cases are indeed heart disease cases and which aren’t? Both are quite important in their own right.

The output (output from the statsmodels) of the logistic regression does not contain R2. This is a convenient indicator to judge a model upon in linear regression and multiple linear regression. However, we do not have the same convenience in logistic regression. Why there is no R2 for logistic regression deals with the cost function and how the logistic regression is designed in general. We would be better served by looking through the logistic regression chapter from any good statistics book. The most convenient way that I prefer is looking at a combination of 3 scores – Accuracy, Precision and Recall. Through experience, looking at these 3 scores we get a general idea at how well the classification model performs.

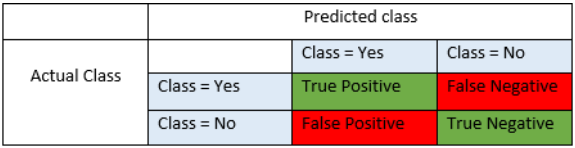


Figure https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

To understand the three measures we need to understand the confusion matrix above. Just have a look at the above diagram. It is quite self explanatory.

We get an intuitive understanding to the performance measures by just looking at the formulae for each.

(Note: True positive (TP), True negative (TN), False positive (FP), and False negative (FN) )

Accuracy = (TP+TN) / (TP+FP+FN+TN)

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

We will go into greater detail of why and how of these model evaluation metrics in a future blog post. For now let us finish our analysis

In this analysis, we use the built-in methods of sklearn’s model selection library. It contains accuracy, precision and recall along with a confusion matrix calculator as functions that work out of the box. Without getting into fine details, we can directly feed in predicted and actual values of the target variable in the test set, to judge the model performance.

Here we obtain:

Accuracy = 82.4 %

Precision = 77.2 %

Recall = 93.6 %

We can see from the above scores that our model performs quite well overall. It is very good at catching almost all true heart disease cases. It however achieves this by throwing more false positive cases. As in, it trades off precision score in favour of a higher Recall score.

# Conclusions and inferences

This analysis was to test out how easy it is to get a logistic regression model up and running on a clean dataset. The dataset provided to us here is pre cleaned, wrangled and researched such that it includes only the most important variables to predict heart disease. It is a well-constructed dataset for complete beginners, as it does not require any more transformations or cleaning. A normal out of the box logistic regression model performs quite well. Going further, more accurate models can be tested out using the SKlearn’s comprehensive machine learning model library. Although beginners would be better served by picking up a slightly more messy dataset to put their skills to the test.