The data samples had 568 instances of data and 32 features

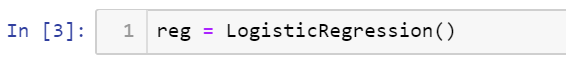


This part of code is used to important libraries that will be used later in the code such as pandas which is the library or the tool to control data frames and data sets and perform operations on them, and secondly the model it self from the “sklearn” package and the model I thought would be efficient after trying some other models was the logistic regression model.

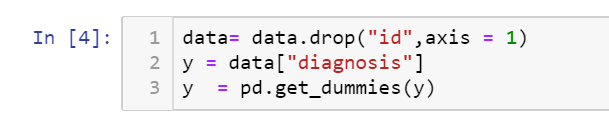


This block is simply importing the file after being processed and given names for each column and the second line is for dropping unnecessary column filled with empty data after the preprocessing operation

This preprocessing operation was done using an external program for data mining and will be illustrated later



This block creating an object for the model so I can modify it later



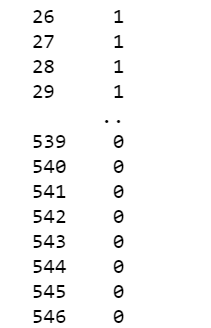
After exploring the data I found that id number is not necessary so I removed it

y will be our target variable and it contains “M” for malignant or “B” for benign

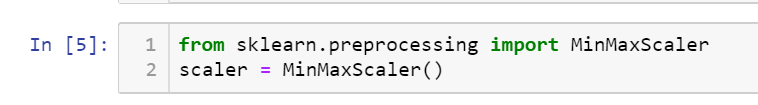
and as there is only two categories I thought it would suit it better not to use OneHotEncoder library to give each a number representing it

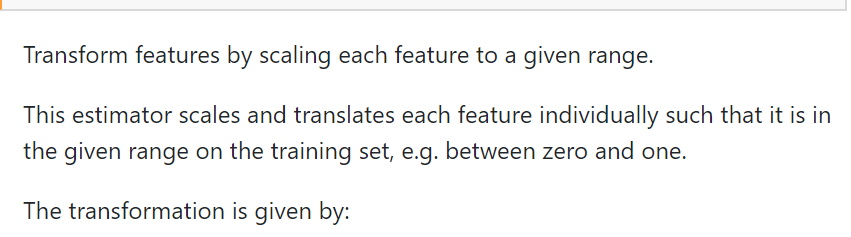
so I just used dummies which creates two column one of them with one “1” if the diagnosis is malignant and the other with “1” if the diagnosis is benign

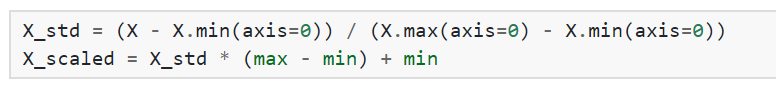
I thought it would suit it better to keep the first column for logical reasons

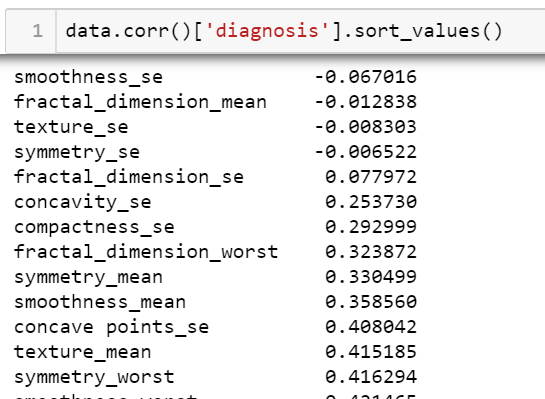


So after that it would be 1 for malignant and 0 for benign





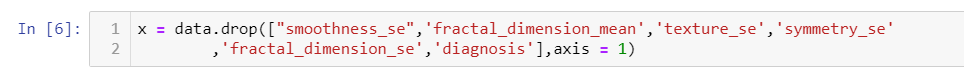




After seeing the co-relations between the diagnosis and the features I thought I would remove the least features with a co-relation factor under 0.33

There is know scientific reason behind it

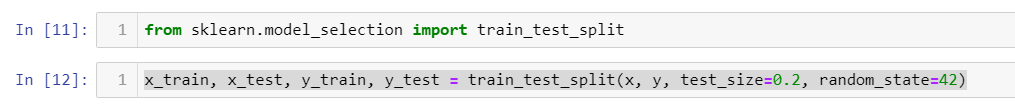
It was a try and error and it was better to be reduced





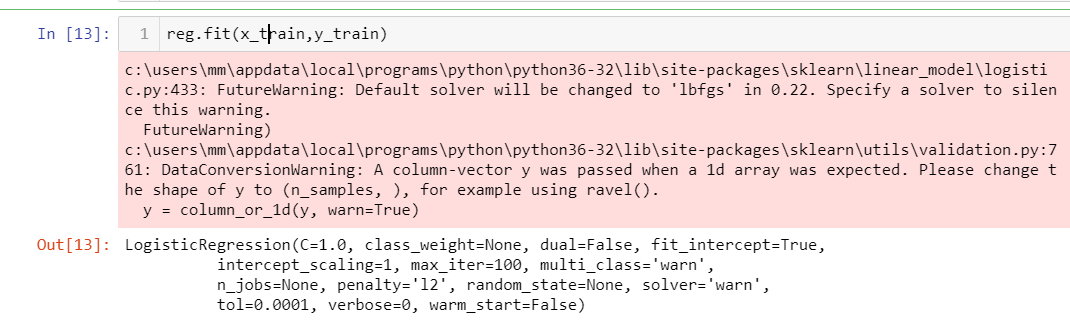


(dropping the benign truth column)



This block of code is used to split the instances between training and testing to test the accuracy of the model

It is a convention to use 20% but I tried it on 20% and 30% as a test set to “try and error” and I specified a random state so I can test it over and over



Fitting the model caused a warning but it was actually talking about the next version so there was no problem using my current version

The model’s parameters were as follows

C=1.0,class\_weight=None, dual=False,fit\_intercept=True,

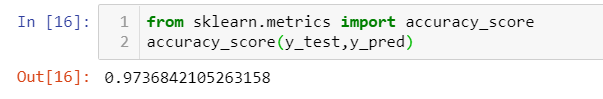
intercept\_scaling=1, max\_iter=100, multi\_class='warn',

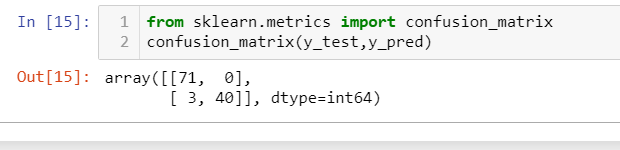
n\_jobs=None, penalty='l2', random\_state=None, solver='warn', tol=0.0001, verbose=0, warm\_start=False



The line of code is to save the predicted by model values

For 20% test set:

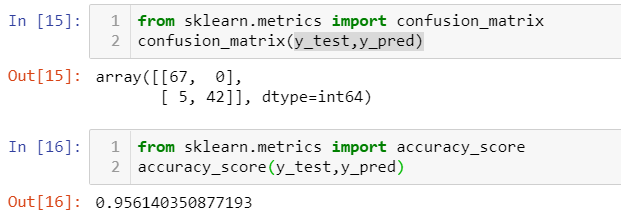




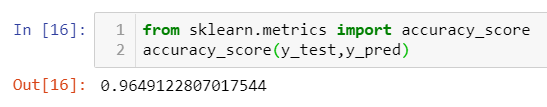
The model turned out pretty well

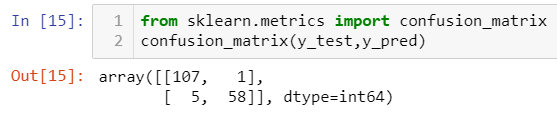
It came with 97% right predictions and a good confusion matrix

20% on another random seed:



And for 30% test set:





The accruarcy was roughly 96%

After wards a model based on the whole dataset should be built

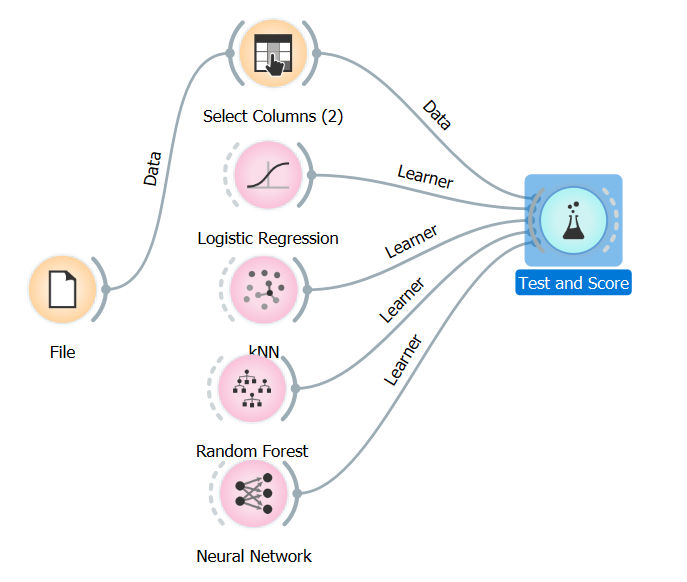
The program I used to prepare the data and then tried a couple of other models is called “Orange data mining”

As me being unexperienced in machine learning engineering I used that tool to help me

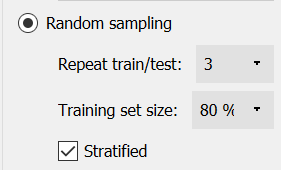
And it was very simple to use, so I used it also to build my model

I plugged the data to application and tried a couple of different classification models to do a “try and error” operation on the set

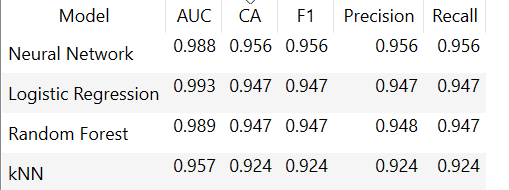
I tried logistic regression, random forest ,k-nearest neighbors and even simple neural networks



I used random sampling with also 20% as test value



The results were very good all above 90% accuracy(counting by CA(classification accuracy))



And on the top neural network was the best model