Exploring Different Machine Learning Models

# Data Preparation

At first, we defined a function to prepare the dataframe. For all the models, we require the independent variable/s (X) and the dependent variable (y) in a specific format.

As of now, the prepare\_dataframe() function will prompt the user for the name of a CSV file in a specific datasets folder, but in the future this can be adapted to handle all directories and input data types.

The variables are extracted from the CSV and the user is prompted to select the dependent variable to predict (y), and the rest are grouped into an independent X.



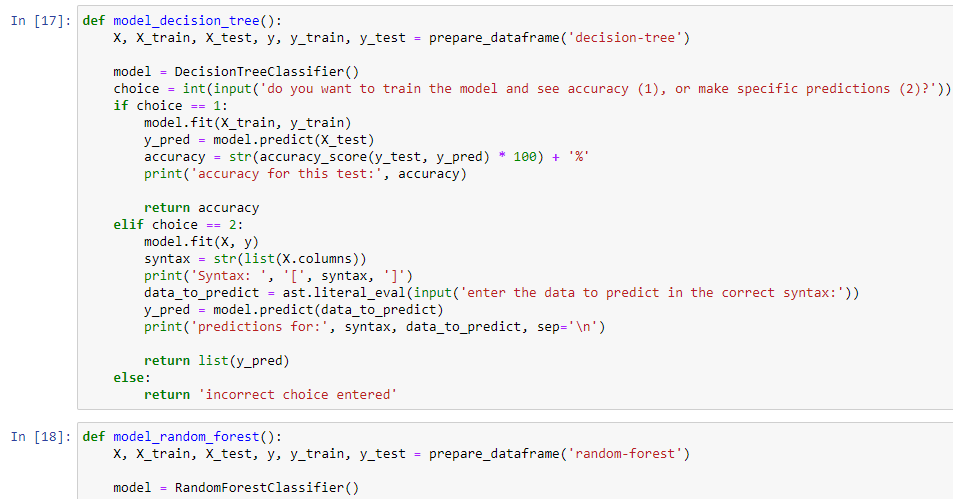
# Preventing Overfitting

In order to prevent overfitting and to test the accuracy of the model, the independent variables dataframe (X) and the dependent variable dataframe (y) are split into training and testing sets (X\_train, X\_test, y\_train, y\_test).

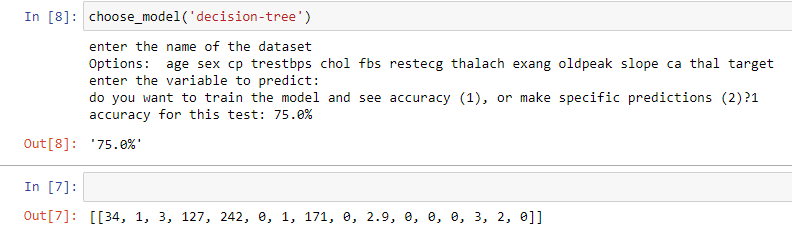
Currently, the percentage of the dataframe used for testing is hard-coded to 17% (approximately ⅙), but this can easily be changed programmatically in the future if needed.

# Training the Classifiers

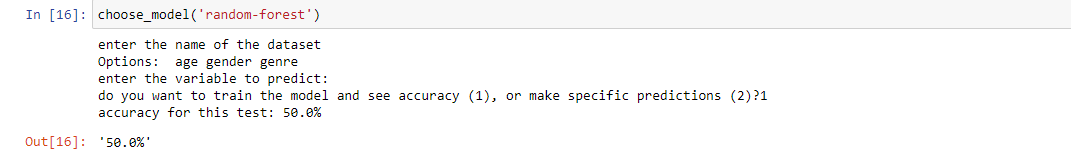
Currently, the two classifiers that are being tested are decision tree and random forest classifiers. The models are fitted with the X\_train and y\_train dataframes, and are tested with the X\_test and y\_test dataframes.

At the moment, we can train the model and get back the accuracy, or make specific predictions.

Generally, decision tree classifiers have an approximately 77.87% accuracy, which aligns to our testing of the decision tree model.

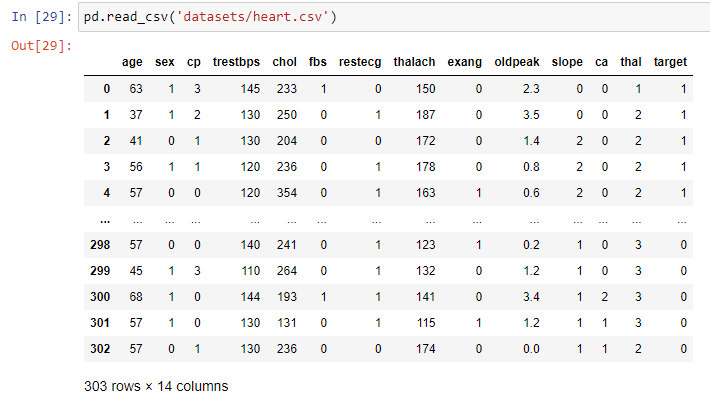


The random forest model fared slightly worse, but this is partly due to the small size of the dataset. For multivariable analysis, random forest classification is superior and more accurate.



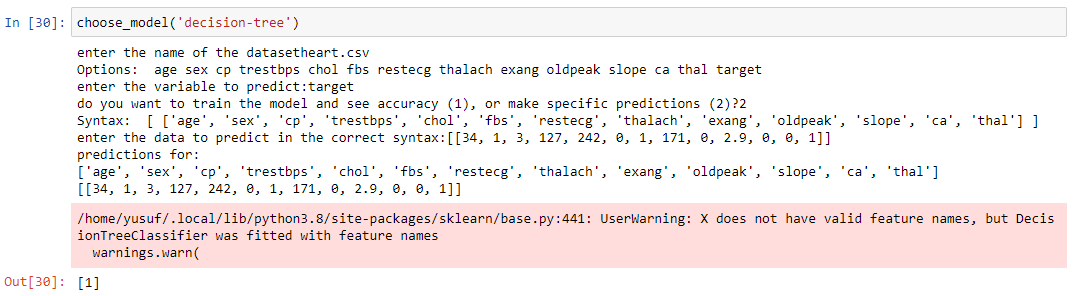
# Making Predictions with the Models

For this example dataset ‘heart.csv’, there are 13 variables that the model intakes to predict whether the patient will have heart disease (target = 1) or no heart disease (target = 0).



Testing this entry: [[34, 1, 3, 127, 242, 0, 1, 171, 0, 2.9, 0, 0, 1]] (34 year old male, chest pain 3, resting blood pressure 127, serum cholesterol 242mg/dl, fasting blood sugar < 120mg/dl, resting electrocardiographic result 1, maximum heart rate 171bpm, no exercise-induced angina, oldpeak 2.9, slope 0, 0 major fluoroscopic vessels coloured, thal 1). Putting this into the model predicts that this particular patient will have heart disease (Output shows 1, which is affirmative).

We are hoping to collate a lot of models together soon.



# Comparison of Ensemble Classifiers

Performed on heart.csv with a train to test data ratio of 4:1. Each time the code was run, a different group of train and test data was selected, with the same split ratio.

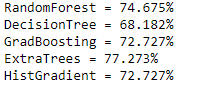
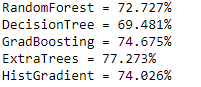
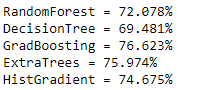
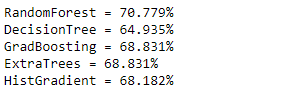
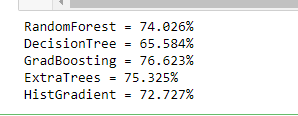
The data from heart.csv has 303 entries with 12 input variables and a binary classification label.

|  | 1st run | 2nd run | 3rd run | 4th run | 5th run | Mean aft.5 |
| --- | --- | --- | --- | --- | --- | --- |
| Random Forest | 74.026% | 70.779% | 72.078% | 72.727% | 74.675% | 72.857% |
| Decision Tree | 65.584% | 64.935% | 69.481% | 69.481% | 68.182% | 67.533% |
| Gradient Boosting | 76.623% | 68.831% | 76.623% | 74.675% | 72.727% | 73.896% |
| Extra Trees | 75.325% | 68.831% | 75.974% | 77.273% | 77.273% | 74.935% |
| Histogram Gradient Boosting | 72.727% | 68.182% | 74.675% | 74.026% | 72.727% | 72.467% |

From these 5 runs ExtraTreesClassifier performed the best on average, however on run 2 the RandomForestClassifier outperformed ExtraTrees which had the same accuracy as the GradientBoostingClassifier. RandomForestClassifier also had the best consistency over the 5 runs, with a range of 3.896%; whereas ExtraTreesClassfier had the highest range at 8.442%.

To get a more appropriate performance indication of these classifier algorithms, more accuracy scores need to be noted and compared at different train to test ratios. Also, these algorithms should be tested on larger data sets as some algorithms may have a better performance on larger data sets.

Results displayed on Jupyter Notebook



# Investigation of Neural Network Algorithms

Neural networks have the ability to learn from events and make decisions through commenting on similar events. They have the highest degree of accuracy, given that enough data is provided. That being said, neural networks are not always the most viable or efficient algorithm to use in certain scenarios.

To create an algorithm we would attach weights and biases on our input from sensors or databases, which would affect the outcomes of the results; these weights and biases would be used in our hidden layers and each hidden layer would have its own unique weight and bias. This would occur until each process goes through all the hidden layers and at the output layer we would receive the results, which would help us create a prediction from the given data.

There are disadvantages to this method, one being that training an algorithm to the point in which it’s reliable enough to get information from, would take a long time and a lot of processing power. This is still a topic we are exploring and more information on how it would be used in our project is being investigated.