RANDOM FORESTS

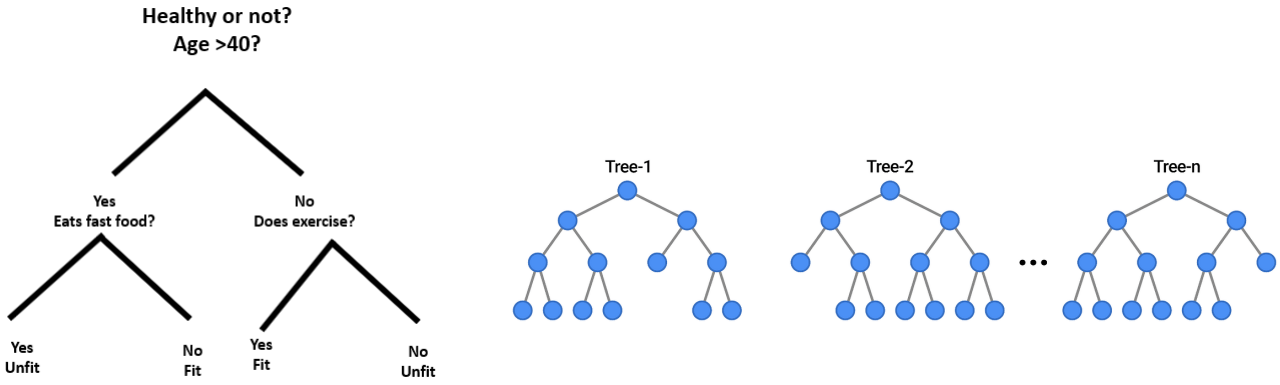
Authors: G. H. Zerwes, *Universidade Federal do Espirito Santo, Vitória, Brazil*

Research Supervisor: L. Homri, *Arts et Métiers ParisTech, Metz, France*

1 INTRODUCTION

The Random Forests algorithm is a powerful ensemble technique that can be used both for prediction and classification problems. It consists of training a group of decision trees, which by themselves tend to over-fit the data, but as a group can be used to make accurate predictions.

The algorithm works by creating subsets of the training group X, with a random group of features. Each of these subsets is then used to train a decision tree. A decision tree consists of a series of 'questions' asked about the data that will divide them according to the answer to the question.



(a) Decision tree

(b) Random Forest

For example, for classifying if a person is healthy or not, a possible question to be asked is 'Does the person exercise?". The samples that answer 'yes' to the question go to a 'leaf node', whereas the remaining samples go to another leaf. The idea is that after a series of questions asked, all the samples in a leaf will have the same class, and that will be the predicted value for the sample. The image 1a shows an example of a decision tree.

The Random Forest algorithm creates then, a group of decision trees, each trained with different features and data, so that they go through different decision processes. After that, the output class will be the most voted class by the trees. Figure 1b shows a diagram of the Random Forest algorithm.

2 Pseudo-code

To use this algorithm on your data, the pseudo-code below shows a procedure that should be followed.

```
#Step 1: Gather the data. The X data is the independent variable and should be already
    pre-treated. The Y array is the vector containing the labels for each class.

X = [x_value_1, x_value_2, ...]
Y = [y_value_1, y_value_2, ...]

#Step 2: Fit the model.
model = RandomForest.fit(X,Y)

#Step 3: Make new predictions.
y_prediction = model.predict(X)
```

3 Evaluating the model and other recommendations

After having fitted the model with data, it's recommended to evaluate the performance with some metrics, to see if the model was correctly trained, or not.

A popular metric for evaluating the model performance is the accuracy. This metric measures how many of the samples were correctly labeled, over the total of samples. Usually, the higher the accuracy, the better the performance, however, too high of an accuracy can also be a sign of an over-fitted model.

Another metric that is widely used is the confusion matrix. This gives an insight into the classes that are being confused by others. In other words, it shows the amount of false negatives and positives in a table format. Figure 2 has an example of a confusion matrix, in which the correctly classified samples are shown in green, and the incorrect ones, in red.

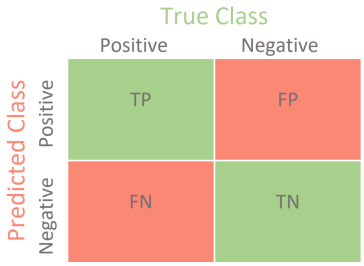


Figure 2: Example of Confusion Matrix

At last, another useful insight that the Random Forest can give is the feature importance plot. This graph, shown as an example in figure 3, shows which of the features have the most weight in making the prediction, compared to the others.

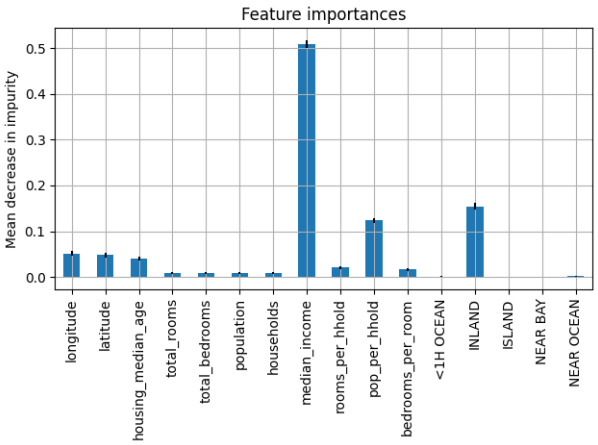


Figure 3: Example of Feature Importance plot

References

Bishop, C. M. (2006). *Pattern Recognition And Machine Learning*. Number 758. Springer.