11. RANDOM FORESTS

The random forest is an ensemble model made up of many decision trees that uses two key concepts that give it the name random:

1. random sampling of training data points when building trees
2. random subsets of features considered when splitting nodes

Despite its simplicity, it is one of the most powerful ML algorithms available today.

The random forest is used for both regression and classification and it is provided via `sklearn` `RandomForestRegressor()` and `RandomForestClassifier()` classes.

The random forest utilizes techniques called **B**OOTSTRAPPING and **AGG**REGAT**ING**, commonly known as **BAGGING**.

- **Random sampling of training data instances with replacement (BOOTSTRAP)**

  When training, each tree in the random forest learns from a random sample of the data instances. Data instances are drawn with replacement, known as bootstrapping[1], which means that some data instances will be used multiple times in a single tree. The idea is that by training each tree on different data instances, although each tree might have high variance with respect to a particular set of training data, overall, the entire forest will have lower variance but not at the cost of increasing the bias.[2]

- **Random subsets of features for splitting nodes**

  The other main concept is that only a random subset of all features is considered for splitting each node in each decision tree. This helps reduce the correlation between the trees. For example, if there is only one strong predictor, most of the trees in the collection will use that predictor in the top split and they will be very similar to each other.

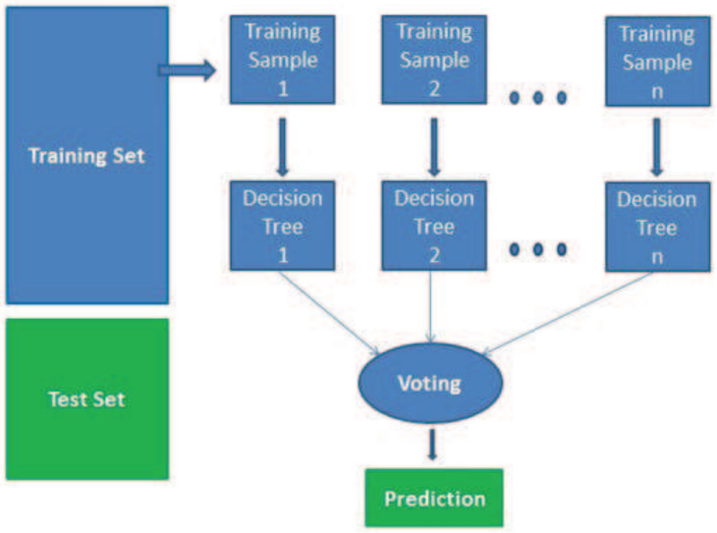
- **Random forest predictions (AGGREGATING)**

  The randomness also implies that each time the algorithm is run on the same data, it will produce a slightly different model. An individual decision tree has high variance, but when we combine many of them together the resultant variance will be low as each decision tree gets perfectly trained on a particular sample data and hence the output doesn't depend on one decision tree but on multiple decision trees.

  A prediction on a regression problem is the average of the predictions across the trees in the ensemble. A prediction on a classification problem is the majority vote for the class label across the trees in the ensemble.

[1] Drawing data instances without replacement is called *pasting*.

[2] When random subsets of the training dataset are drawn as random subsets of the features, the method is known as *random subspaces*. When base estimators are built on subsets of both data instances and features, then the method is called *random patches*.



https://www.datacamp.com/tutorial/random-forests-classifier-python

Random forest hyper-parameters in `sklearn`

- Number of samples (data instances) to be used in individual decision trees

  Each decision tree is fit on a bootstrap sample drawn from the training dataset with replacement. This can be turned off by setting `bootstrap = False` and in that case the whole training dataset will be used to train each decision tree which is not recommended.

  When `bootstrap = True`, the `max_samples` argument can be set between 0 and 1 to control the percentage of the size of the training dataset to make the bootstrap sample used to train each decision tree. For example, by setting `max_samples = 0.5`, each decision tree will be fit on a bootstrap sample containing 50% training data instances. A smaller value of `max_samples` will make trees more different, and a larger value will make the trees more similar. The default setting is `max_samples = None` which will make the bootstrap sample size the same as the size of the training dataset.

- Number of features used in each split[3]

  The number of features used in each split is set via `max_features` argument. In regression, it is recommended to be set to the one-third of the total number of features. For classification problems, usually it is set to be the square root of the total number of features. For example, if we have 20 features, this would mean about 6-7 features (20/3) for regression problems, and about 4 features ($\sqrt{20}$) for classification problems.

- Number of trees

  Typically, the number of trees is increased until the model performance stabilizes and no further improvement is seen. Intuition might suggest that more trees will lead to over-fitting, although this is not the case given the stochastic nature of the learning algorithm. The number of trees can be set via `n_estimators` and defaults to 100.

[3]To make base trees even more random, we can use random thresholds for each feature instead of searching for the best possible thresholds. A forest of such extremely random trees is called an *extremely randomized trees (extra-trees).*

- Depth of the decision trees

  Deeper trees are often more over-fit to the training data, but are also less correlated, which in turn may improve the performance of the ensemble. Depths from 1 to 10 seem to be effective.

Out-of-bag (OOB) error

The OOB error is a straightforward way to estimate the test error of a bagged model using bootstrap without the need to perform cross-validation.

On average, each bagged tree makes use of around 63% of the training set observations. The remaining 37% of the observations are not used to fit a given bagged tree and are called "out-of-bag observations".

Remark: Assume that there are n data observations in the training dataset and that the size of the bootstrap subset is also n. The probability of not picking a data observation in a random draw is

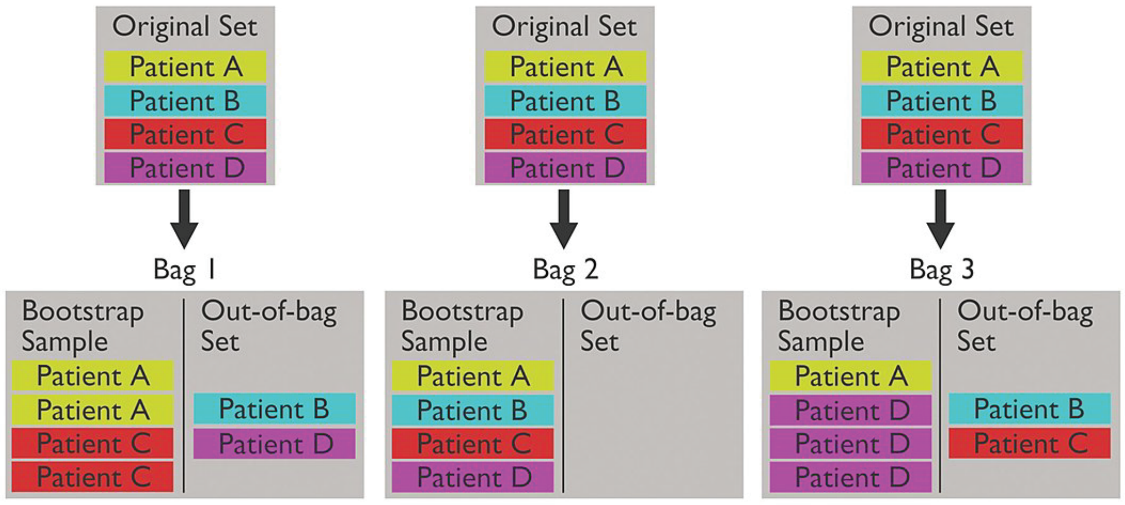
$$\frac{n-1}{n} = 1 - \frac{1}{n}$$

Using that sampling is with replacement, the probability of not picking this data observation in n draws is

$$\left(1 - \frac{1}{n}\right)^n$$

which has a limit

$$\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = e^{-1} \approx 0.368 \approx 37\%$$



https://en.wikipedia.org/wiki/Out-of-bag_error

We predict the response of the ith data observation using each tree for which that observation was OOB. Note that when using cross-validation technique, each fold has been seen by the model, while OOB score prevents leakage.

It can be shown that with the number of trees in the forest being sufficiently large, the OOB error is virtually equivalent to the leave-one-out cross-validation error.

Advantages of Random Forests:

- Can be used for both classification and regression.
- A highly accurate and robust model.
- Does not suffer from over-fitting because it takes into account predictions from all trees in the forest.
- Gives the relative feature importance, which helps in selecting the most contributing features for the model.

Disadvantages of Random Forests:

- May be slow in generating predictions since all the trees in the forest have to make a prediction for the given input.
- Difficult to interpret compared to a single decision tree where we can easily make a decision by following the path in the tree. Bagging improves prediction accuracy at the expense of interpretability.

Python code: Lecture_11_RF.ipynb

References and Reading Material:

[1] *Hands-On Machine Learning with Scikit Learn, TensorFlow and Keras*, Geron et al., pp 189-199

[2] *An Introduction to Statistical Learning*, James at al., Section 8.2.2