Decision Tree

***Structure**: Takes the shape of a binary tree. Raw data is denoted as the 'root node', which is recursively splitted until boudary conditions are met, producing 'leaf nodes'. In the process of descending down this contructed tree , a classification task is completed. If desired, a regression task can be done if the average of the leaf nodes is calculated.*

***Splitting Criteria**: The overall criteria of splitting in Decision Tree is to reach a certain degree of 'purity' at the leaf nodes, so the goal of every split on the way is to increase the purity of the daughter nodes we get. Two indexes are widely used in Decision Tree to evluate the split: information gain and Gini coefficient.*

***Information Gain**: The concept of information gain is based on the concept of information entropy. Information gain is defined by the following equation:*

$$\mathrm{E}(\mathrm{X}) = -\sum_{k=1}^{\mathrm{n}} \rho_k \log_2 \rho_k$$

In which, X is a set of samples and ρ_k is the ratio of samples of the k-class to the whole set. When E(X) is small, the 'purity' of the set of samples is high, with the minimum of E(X) being , meaning all the samples of X are of the same class. To evaluate the result of a split, we can define information gain based on information entropy using the following equation:

$$\mathrm{G}(\mathrm{X}, \mathrm{f}) = \mathrm{E}(\mathrm{X}) - \sum_{v=1}^{V} \frac{|\mathrm{X}^v|}{|\mathrm{X}|} E(\mathrm{X}^v)$$

In which, f is a discrete feature with V different values and X^v is the samples with the v-th value. Thus, we can calculate the information gain if we use feature 'f' as a splitting criteria. We would like to choose the feature that has the maximum information gain at a specific node as the splitting criteria at that node.

***Gini coefficient**: Gini coefficient reflects the probability of picking two samples from a certain set X that belongs to different classes. It is defined by the following equation:*

$$\mathrm{G}(\mathrm{X}) = 1 - \sum_{k=1}^{n} {p_k}^2$$

And for a certain feature 'f', the Gini coefficient is defined by the following equation:

$$\mathrm{G}(\mathrm{X}, \mathrm{f}) = \sum_{v=1}^{V} \frac{|\mathrm{X}^v|}{|\mathrm{X}|} \mathrm{G}(\mathrm{X}^v)$$

We would like to choose the feature 'f' that minimizes the G(X, f) to split at a certain node.

***Boundary Conditions**: Typically, the bounary conditions is reached on the following conditions: (1) the pre-set tree depth is reached; (2) the leaf nodes are pure enough.*

***Sidenotes**: The splitting criteria of normal decision trees decides that only one feature can be used in one certain split, so the classification boudaries we get in the feature space is always aligned to the axels, being one of the main holdbacks of Decision Tree.*

Random Forest

***Outline**: Random Forest is an 'ensemble' classification method that is based on Decision Tree. The idea is that instead of using a single classifier to do the work, we use multiple trees to do their own classification, and then choose the major result as the final classification result.*

***Structure**: Suppose there is a set of N samples, we perform bootstrap sampling on the set for a total of T times and get T training sets; suppose there are M features, bootstrap sampling is also applied and we randomly choose m(m<<M) features and get T subsets of features. Then we construct a separate decision tree using every training set and subset of features. After constructing T trees, a random forest is constructed. When a single sample is classified, it descends down every decision tree and get a result from each one of them, and the majority of the result is adopted as the final result.*

***Randomness**: Randomness is an important factor affecting the performance of an rf-classifier. There are two sources of randomness in the construction of a random forest: the random selected training samples and random selected features. The bootstrap sampling method guarantee that every decision tree constructed is unique yet unbiased, and the random selected features reduces the correlation of the decision trees, thus improves the performance. However, a too small m value may sacrifice the classification power of individual trees, and hold back the performance as well. So for an rf-classifier, choosing an appropriate m value is the key to get good classification results.*

***Out-of-bag Error**: Out-of-bag error is the method used to evaluate different m values in an rf-classifier. Since every training set is constructed using a bootstrap sampling method, for every sub-tree, approximately 1/3 of all the samples won't be used in the construction, and these samples are called 'out-of-bag' samples for the sub-tree. The OOB error of an rf-classifier is calculated using the following method. For every sample, we descend it down the trees that views the sample as OOB sample, and get the classification result of the sample. Then the overall result is the plurality vote of the subset of trees. For all the samples, the classification error ratio is the OOB error of the rf-classifier, and of course we want it to be minimized.*

***Sidenotes**: Since random forest is based on decision trees, it suffers from the same weakness as traditional decision trees, which is stated above. And to settle this, SPORF is introduced.*

SPORF(Sparse Projection Oblique Randomer Forest)

***Outline**: The idea of SPORF is to introduce oblique classification boundaries into RF method, thus introduces yet another source of randomness, and improves the overall performance of the RF-classifier. It settles the problem of traditional Decision Trees and RFs and have better results.*

***Algorithm**: The trees in a SPORF uses Gini impurity as the splitting criteria, and there is usually 500 trees in a SPORF, which is determined by experience. For a dataset with n samples and p features, a traditional RF randomly select m features to form a subset and search within it the best feature to do the split. For example, with a dataset of 5 features, if we set the m at 3, we may produce a matrix that takes the form of the following one:*

$$\mathrm{F} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We can see that each column has only one non-zero number because of the axel-aligned characteristic of normal decision trees. However, in a SPORF, this feature selection matrix may have $\lceil \lambda pm \rceil$ non-zeros numbers of 1 or -1, λ being the sparse factor, allowing the oblique factor. Thanks to this random projection method, the RF is 'randomer' and perform better.

***Parameters**: The parameters that needs tuning in a SPORF is m and λ, in which m is the same case as in a normal RF. Also, these are usually hyperparameters, pre-determined by experience. In a SPORF, m is set to be $p^{1/4}$, $p^{1/2}$, $p^{3/4}$ and p^2; λ is set to be 1/p, 2/p, 3/p, 4/p and 5/p.*