Decision Trees

Information Gain at decision trees

When making decision trees, two different methods are used to find the best feature to split a dataset on: Gini impurity and Information Gain. An intuitive interpretation of Information Gain is that it is a measure of how much *information* the individual features provide us about the different classes.

Gini impurity

When making decision trees, calculating the Gini impurity of a set of data helps determine which feature best splits the data. If a set of data has all of the same labels, the Gini impurity of that set is 0. The set is considered pure. Gini impurity is a statistical measure - the idea behind its definition is to calculate how accurate it would be to assign labels at random, considering the distribution of actual labels in that subset.

Decision trees leaf creation

When making a decision tree, a leaf node is created when no features result in any information gain. Scikit-Learn implementation of decision trees allows us to modify the minimum information gain required to split a node. If this threshold is not reached, the node becomes a leaf.

Optimal decision trees

Creating an optimal decision tree is a difficult task. For example, the greedy approach of splitting a tree based on the feature that results in the best current information gain doesn't guarantee an optimal tree. There are numerous heuristics to create optimal decision trees, and each of these methods proposes a unique way to build the tree.

Decision Tree Representation

In a decision tree, leaves represent class labels, internal nodes represent a single feature, and the edges of the tree represent possible values of those features.
Unlike other classifiers, this visual structure gives us great insight about the algorithm performance.

Decision trees pruning

Decision trees can be overly complex which can result in overfitting. A technique called pruning can be used to decrease the size of the tree to generalize it to increase accuracy on a test set. Pruning is not an exact method, as it is not clear which should be the ideal size of the tree. This technique can be made bottom-up (starting at the leaves) or up-bottom (starting at the root).

Decision Trees Construction

Decision Trees are usually constructed from top to bottom. At each level of the tree, the feature that best splits the training set labels is selected as the “question” of that level. Two different criteria are available to split a node, Gini Index and Information Gain. The convenience of one or the other depends on the problem.

Random Forest definition

A Random Forest Classifier is an ensemble machine learning model that uses multiple unique decision trees to classify unlabeled data. If compared to an individual decision tree, Random Forest is a more robust classifier but its interpretability is reduced.

Random Forest overfitting

Random Forests are used to avoid overfitting. By aggregating the classification of multiple trees, having overfitted trees in the random forest is less impactful. Reduced overfitting translates to greater generalization capacity, which increases classification accuracy on new unseen data.

Random Forest feature consideration

When creating a decision tree in a random forest, a random subset of features are considered as the best feature to split the data on. By splitting the data in a random subset of features, all estimators are trained considering different aspects of the data, which reduces the probability of overfitting.

Random Forest aggregative performance

A random forest classifier makes its classification by taking an aggregate of the classifications from all the trees in the random forest. For classification, this aggregate is a majority vote. For regression, this could be the average of the trees in the random forest. This aggregation allows the classifier to capture complex non-linear relations from the data. The model performance is far superior than a linear model.

Bagging at Random Forest

Trees in a random forest classifier are created by using a random subset of the original dataset with replacement. This process is known as bagging. Bagging prevents overfitting, given that each individual tree is trained on a subset of original data.