**Decision tree(DTLearner)**

* A decision tree is a schematic, tree-shaped diagram used to determine a course of action or show a statistical probability. Each branch of the decision tree represents a possible decision, occurrence or reaction.
* The tree is structured to show how and why one choice may lead to the next, with the use of the branches indicating each option is mutually exclusive.
* A decision tree is a graphical depiction of a decision and every potential outcome of making that decision. It can range from something simple to a complex undertaking.
* Decision trees give people an effective and easy way to understand the potential options of a decision and its range of possible outcomes. This helps people identify every potential option and weigh each course of action against the risks and rewards each option can yield.

**Details regarding code**

Create object of DTLearner and call addLearner with parameters:(‘input\_train\_features’,’ train\_labels’)

CLASS: DTLearner:

arguments:

1. leaf size: leaf size is used to define minium number of items must be in leaf node
2. verbose:verbose is used to show model info.
3. tree: used to pass exising trained tree model

* **FUNCTION: author**

-->returns *Georgia Tech username*

* **FUNCTION : addEvidence**

Input:

1.dataX: A numpy ndarray of X values of data to add

2.dataY: A numpy 1D array of Y training values

Output: An updated tree matrix for DTLearner

Functionality**:**

It first checks whether tree is currently None or not , if so then simply assign new\_tree to it otherwise it append to existing tree

If verbose equals True then it calls get\_learner\_info to print model informations

It calls private function ‘\_\_build\_tree’ to build actual tree

* **FUNCTION: \_\_build\_tree**

Functionality:

It select best feature to split on based on maximum pearson correlation with labelsY and split from median of the best feature.

* **FUNCTION: \_\_tree\_search**

A private function to be used with query. It recursively searches

the decision tree matrix and returns a predicted value for point

Parameters:

point: A numpy 1D array of test query

row: The row of the decision tree matrix to search

Returns :

pred: The predicted value

Functionality:

First of all it checks If splitting value of feature -1 if so then we have reached a leaf so return it.

If point value is less than splitting value then go to left child else go to right child by calling \_\_tree\_search recursively.

**Random Tree(RTLearner)**

It is same as decision tree except one difference, it select feature randomly all other functionalities are same as DTLearner.

**Bootstrap Aggregating(Bagging)**

Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and [regression](https://en.wikipedia.org/wiki/Regression_analysis). It also reduces variance and helps to avoid [overfitting](https://en.wikipedia.org/wiki/Overfitting).

Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

**BagLearner:**

* class BagLearner:
* Parameters:

-learner: A LinRegLearner, DTLearner, or RTLearner.

-bags: The number of learners to be trained using Bootstrap Aggregation.

-boost: If true, boosting will be implemented.

-verbose: If True, information about the learner will be printed out

-kwargs: Keyword arguments to be passed on to the learner's constructor.

-Returns: An instance of Bag Learner.

Functionality:

It basically creates bags(different training data to train model) and randomly picks data from original training set to overcome problem of overfitting and train learner (name provided in LearnerList)