Project 1 Report

Miguel Muniz

Decision Tree Node

- def init(self, feature=None, threshold=None, left=None, right=None, *, value=None):

Contains the index of features and the values of thresholds to split on. Also contains values for the left and right nodes as well as the value for leaf nodes.

- is_leaf_node(self):

Detect whether the node is a leaf using the value from the init method.

Decision Tree

- init

sets up the method and has a variable

- def _entropy(self, y):

creates the entropy(randomness) value by taking the labels and processing them according to the formula $H = \sum_{c \in C} -p(c) \log_2(p(c))$

- def _information_gain(self, X_column, y, threshold):

find's the Parent entropy of a the y(labels) and takes the H value to be used in the calculation of the information gain. Before it calculates it splits the features based on the threshold and calculates the information gain for each of the split. The formula use is:
$IG = H - \sum_{t \in T} p(t) H(t)$

- def _best_split(self, X, y, features):

splits the tree based on the best feature and threshold. It does this by calling the information gain function

- def _build_tree(self, X, y, depth):

Build's the tree and handles conditions for max depth and the labels which it then call

- def _most_common_label(self, y):

Returns the bincount of the labels and returns the label with the highest value.

- def fit(self, X, y):

Takes the build tree function and call with the parameters containing features, labels and depth.

- def _traverse_tree(self, x, node):

Traverses the tree and returns the predictions for the given x value.

- def predict(self, X):

returns an numpy array that calls the traverse tree functions with the features.

- def accuracy(self, X, y):

Gathers the accuracy from taking the predictions from the features and then seeing if their sum by the labels are correct.