Decision Trees

Applied ML in Engineering - Exercise 07

TU Berlin, Summer Term 2023

Prof. Dr.-Ing. Merten Stender – merten.stender@tu-berlin.de

Students are asked to implement a basic decision tree model from scratch using object-oriented programming in Python. The exemplary data set, also used in the lecture slides, is provided in the file `decision_tree_dataset.txt` where the first two columns provide the coordinates and the last column provides the targets (0 and 1 for the binary classification problem). Remember that the left child of a split will always carry all data points that are $\leq$ the splitting threshold, and the right child will contain all data that are $>$ the splitting condition.

A backbone of the source code is provided via ISIS to illustrate the structure of the class. A second class `Node` is provided. A node carries the split condition (`split_dim` for the index of the feature space along which to split; `split_val` for the actual value at which to split; $x \leq x_{\text{split}}$ links to the **left** child node), both of its child nodes, and - if it is a leaf node - the prediction value obtained from a majority vote during training.

Problem 0

Familiarize yourself with the code provided. Spot the functions that you were implementing in exercise 06. Investigate the `main` function and the evaluations done there. Feel free to add additional comments for understanding the code better. Note that this skeleton of code will not run as it is. You will first have to fill in the gaps.

Problem 1

Implement various helper functions that will be required for the generation of a decision tree. Fill the voids indicated by `???` in the provided python implementation:

- `def _majority_vote()` : return index of the majority class label
- `def _grow_tree()` : complete base condition statement
- `def _grow_tree()` : line 178. add the best split returning `split_dim` and `split_val` .
- `def _grow_tree()` : lines 187 ff. implement the recursive growth of the tree, i.e. the generation of `child_left` and `child_right` .
- `_traverse_tree()` : line 216. Complete the base condition
- `_traverse_tree()` : lines 220, 222. Complete the tree traversal mechanism, i.e. the data flow based on the split condition.
- `fit()` : attach the grown tree to the root node.
- `predict()` : return the decision tree predictions as a `np.ndarray` .

Problem 2

Validate your implementation using the data set provided (or the minimal 1-dimensional data set provided in the main, note that the subsequent plots will not work for that data set) to check your code. Use `print()` statements to track down issues and bugs. After successful implementation of the decision tree: play around with the hyperparameters and their effect on the prediction quality.