1 Overview

In this exercise sheet, you will implement the Decision Tree machine learning model to solve classification problems. Edit the Jupyter notebook `Decision_Tree.ipynb`. A simple class hierarchy for a tree is already provided. Your task is to implement the construction of the tree step by step.

Ultimately, we want to implement a function `build_tree` that accepts a data set and returns a complete tree. Our data set generally has the form of a list of data points. Each data point consists of a tuple (x, y), where x is a list of the individual features and y is the class (coded as an integer or string).

1.1 Auxiliary Functions (10 points)

We need two auxiliary functions: `count_occurance_of_class` and `get_unique_classes`.

- `count_occurance_of_class` accepts a data set in the above format and a class, and counts the number of occurrences of this class in the data set. The return value is the count.
- `get_unique_classes` also accepts a data set in the above format and returns a list containing all occurring classes. Each class should only be included once, even if it occurs several times in the data set.

a) Argue whether unit tests are required for this function. Which cases should be tested?
Not really, they are simple.

b) Add the required unit tests.

c) Implement the functions.

I needed the following time to complete the task: 5min

1.2 Gini Index (10 points)

Next, we need a function that calculates the Gini index for a data set in the above format.

a) Argue whether unit tests are needed for this function. Which cases should be tested?
No, it's just implementing a simple formula.

b) Add the required unit tests.

c) Implement the functions.

I needed the following time to complete the task: 5min

1.3 Data Set Split (5 points)

Next, we need a function that performs a split on a data set. The function `split_data` accepts a data set in the above-mentioned format, the index of a feature f at which is to be split and a split value. It returns a tuple $(left, right)$. Here, $left$ and $right$ are each subsets of the data set. For all data points in $left$, feature f is less than or equal to the limit value; for all data points in $right$, feature f is greater than the limit value.

a) Argue whether unit tests are needed for this function. Which cases should be tested?
At best, if the index of the feature to split on is valid, can also be done by assertion

b) Add the required unit tests.

c) Implement the functions.

I needed the following time to complete the task: 5min

1.4 Select Split (10 points)

Now let's look at selecting good splits. Implement the function `select_best_split`. This first generates a list of possible splits for a data set in the above format. with possible splits. This contains the split at the median for at least every feature in the data set. A split is defined as a tuple $(feature, value)$, where feature is the index of the features of the best split, and value is the threshold value from which the split is to be made. The function executes the splits on a trial basis and evaluates the result: useless splits (a subset is empty) are sorted out, the remaining splits are evaluated using the Gini index. The function returns a split.

a) Argue whether unit tests are needed for this function. Which cases should be tested?

b) Add the required unit tests.

c) Implement the functions.

d) Optional: Implement the optimization that a split is evaluated based on a sample of the data set. Determine experimentally from what size of data set this is faster.

e) Optional: Argue in which cases it could be useful to split at points other than the median of the values. Implement an additional split strategy.
Using the median we assume that we can split the data in the middle of the values. But the class split doesn't have to be there obviously. Or we have 3 classes and using the median we would split one class in half which also doesn't help.
It might make sense to find a value which perfectly splits one class from the rest so that split for class 0 is $\max(datapoint_{class=0}) + \epsilon$

I needed the following time to complete the task: 20min

1.5 Build Tree (20 points)

With the help of the auxiliary functions already implemented, we can now build a Decision Tree.

a) Argue whether unit tests are required for this function. Which cases should be tested? What role can visualizations play?
It could make sense to test by using a simple 2 classes dataset to see if the algorithm is terminating (recursion problems).

b) Add the required unit tests.

c) Implement the functions.

d) Test your tree using the synthetic data set and the iris data set.

Recompile the latex to display the results.

I needed the following time to complete the task: 10min