Decision Tree

Wavelix

1 Requirement

1. Read and understand a decision tree and its visualization code.
2. Use a portion of the data as the test dataset, and verify the classification performance of the decision tree use the test dataset. Evaluate using accuracy as the metric.
3. install graphviz and configure the correct path.

2 Code Understanding

The python script `decision_tree.py` defines a decision tree model, including methods for buiding, training and making prediction.

The class `Node` represents the basic components of the decision tree. Each node can be either a leaf node or an internal node. A leaf node holds a `value`, which is the predicted label for that node. An internal node use `feature index` to store the feature being split at tha node. A dictionary `children` maps feature valuer to subsequent nodes.

The constructor accepts a `gain threshold` parameter. which sets the minimum required information gain for a feature to be considered as a split point. Lower thresholds allow deeper trees, while higher thresholds result in simper tree.

The method `_entropy` calculates the entropy of a target variable Y, using the formula:

$$H(Y) = -\sum p(Y = y)\log_2(p(Y = y))$$

The method `_conditional_entropy` calculates the conditional entropy of Y given a specific feature, which measures the average entropy across splits on that feature. `_information_gain` calculates the information gain by the formula:

$$IG = H(Y) - H(Y|\text{feature})$$

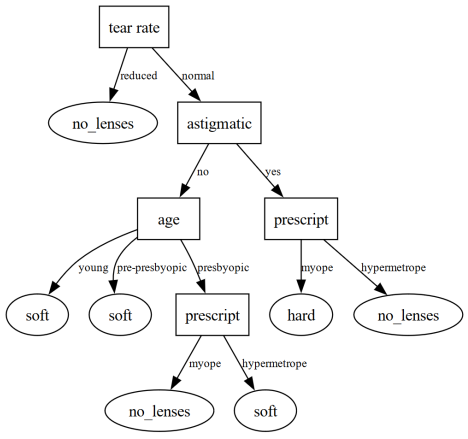
The method `_select_feature` identifies the feature with the highest information gain from the available features. If the best gain exceeds the `gain_threshold`, the feature index is returned; otherwise, it returns `None`, indicating no further splits are necessary.

`_build_tree` is the core method that builds the decision tree. Each node starts with a default predicted label, which is the most frequent class in the given subset of y. If all labels in the subset are the same, the node becomes a leaf. Otherwise, it selects the best feature for splitting using `_select_feature`. For the chosen feature, the data is partitioned into subsets based on feature values. Subtrees are recursively built for each subset. The process stops when no features can be selected for further splits, resulting in a leaf node.

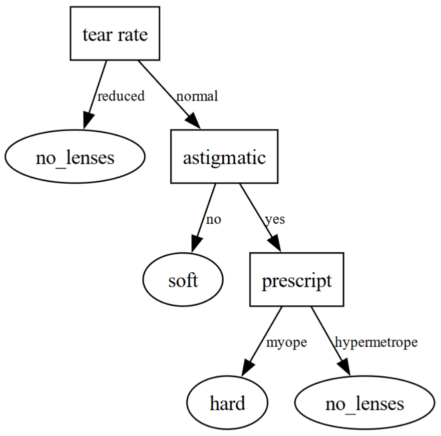
3 Code Modification and Analysis

In `lenes_dt.ipynb`, before building the decision tree, I split the dataset into a training set and a testing set, with a ratio of 7:3. Then, the training set is used to build the decition tree, and we use the tree to make prediction on testing tree. Accuracy is used to evaluate the results.

Apply `np.random.seed(1)`, and the accuracy is 0.8889. Apply `np.random.seed(2)`, and the accuracy is 0.7778.



(a) np.random.seed(1)



(b) np.random.seed(2)

Different random seeds can lead to different results. Some possible reasons are:

1. The dataset is too small.
2. Randomly spliting can result in certain feature values being overrepresented or underrepresented in the training set.
3. When feature values have close information gain scores, small changes in the training data due to different splits can lead to entirely different features being selected at certain nodes.

Larger datasets tend to reduce the impact of random sampling noise on tree construction. After that, instead of a single train-test split, perform multiple splits and average the performance metrics across folds. This reduces the influence of any one random split.