Machine Learning

Decision trees: classification

Sebastian Engelke

Master in Business Analytics

UNIVERSITÉ
DE GENÈVE

Classification trees

- Classification trees are very similar to regression trees, except that they are used to predict a qualitative response $Y \in \mathcal{G} = \{1, \ldots, q\}$ instead of a quantitative one.
- The prediction in a region R_j is the constant $\widehat{y}_{R_j}$ that equals the most commonly occurring class of the training observations in R_j, i.e., the $g = 1, \ldots, q$ that maximizes

$$\widehat{p}_{jg} = \frac{1}{n_j} \sum_{i: x_i \in R_j} \mathbf{1}\{y_i = g\},$$

so that the classifier is

$$\widehat{G}(x) = \sum_{j=1}^{J} \widehat{y}_{R_j} \mathbf{1}\{x \in R_j\}.$$

- The principles of pruning and subtrees are the same as for regression trees.
- For classification, the RSS cannot be used as loss function and instead on uses measures of node impurity for the terminal node R_j:

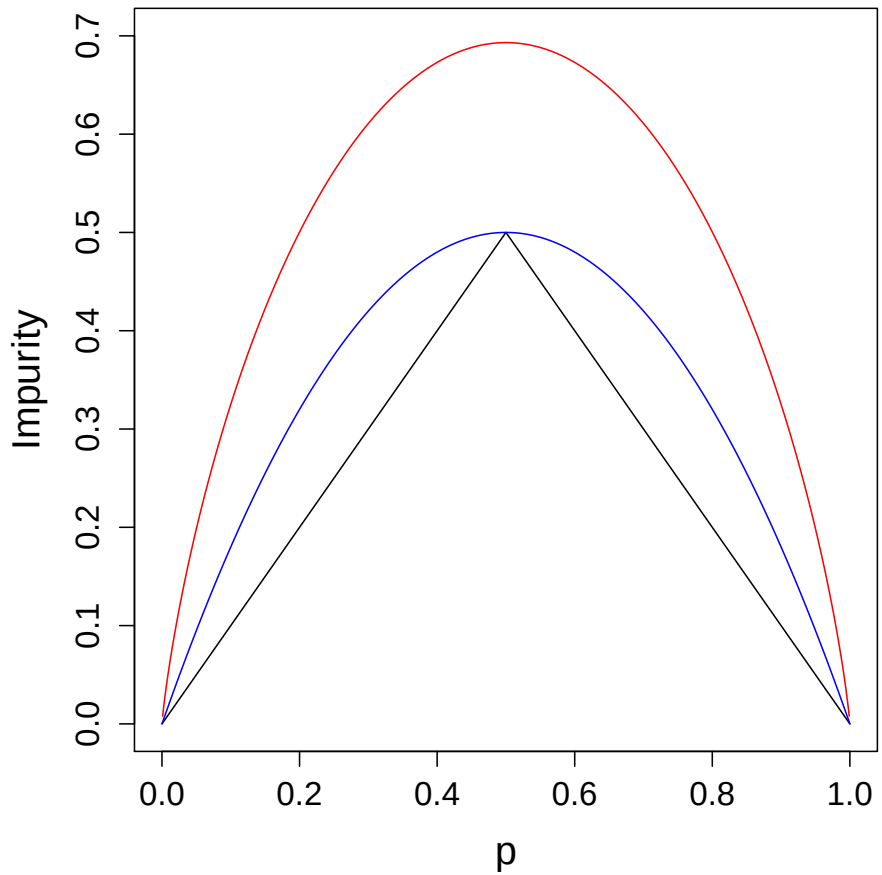
(1) misclassification error: $M = \frac{1}{n_j} \sum_{i: x_i \in R_j} \mathbf{1}\{y_i \neq \widehat{y}_{R_j}\}$;

(2) Gini index: $G = \sum_{g=1}^{q} \widehat{p}_{jg}(1 - \widehat{p}_{jg})$;

(3) Cross-entropy/deviance: $D = -\sum_{g=1}^{q} \widehat{p}_{jg} \log \widehat{p}_{jg}$.

Node impurity measures

- In the case of two classes, the misclassification error is $1 - \max(p, 1-p)$ (black), the Gini index is $2p(1-p)$ (blue), and the cross-entropy is $-p\log p - (1-p)\log(1-p)$ (red).
- The misclassification impurity is non-differentiable and thus bad for optimization.
- The Gini index and the cross-entropy will `favor pure nodes` with $p_{jg} \approx 0$ or $p_{jg} \approx 1$; they are very `similar numerically`.

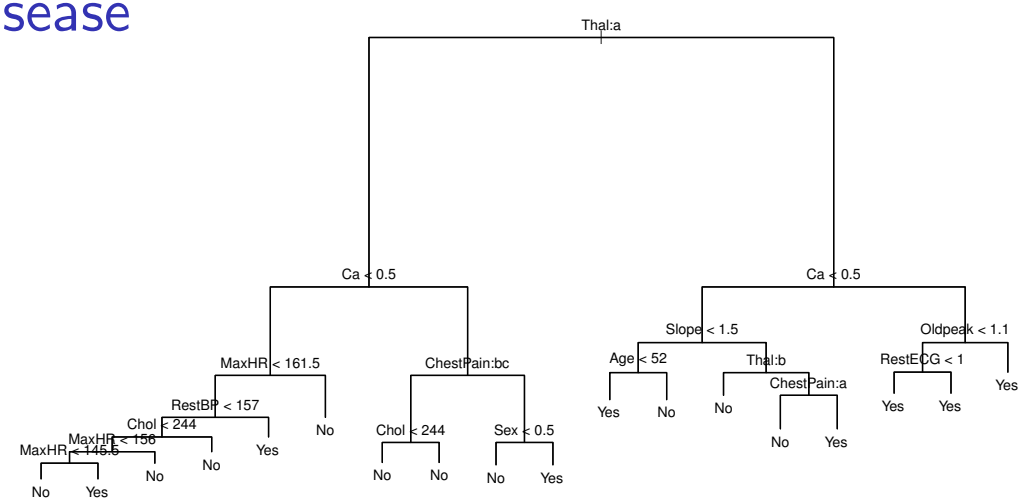


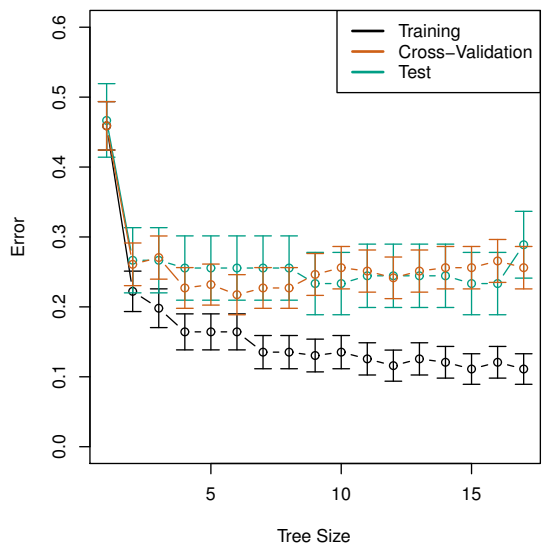
Example: heart disease

The `heart` data set in the `ISLR` package contains whether or not the patient has a heart disease (AHD), i.e. $G \in \{\texttt{Yes, No}\}$, for 303 patients who presented with chest pain, with 13 predictors (`sex, age, chol,...`). A subset of the data:

	AHD	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR
1	No	63	1	typical	145	233	1	2	150
2	Yes	67	1	asymptomatic	160	286	0	2	108
3	Yes	67	1	asymptomatic	120	229	0	2	129
4	No	37	1	nonanginal	130	250	0	0	187
5	No	41	0	nontypical	130	204	0	2	172
6	No	56	1	nontypical	120	236	0	0	178
7	Yes	62	0	asymptomatic	140	268	0	2	160
8	No	57	0	asymptomatic	120	354	0	0	163
9	Yes	63	1	asymptomatic	130	254	0	2	147
10	Yes	53	1	asymptomatic	140	203	1	2	155
11	No	57	1	asymptomatic	140	192	0	0	148
12	No	56	0	nontypical	140	294	0	2	153
13	Yes	56	1	nonanginal	130	256	1	2	142
14	No	44	1	nontypical	120	263	0	0	173
15	No	52	1	nonanginal	172	199	1	0	162
16	No	57	1	nonanginal	150	168	0	0	174
17	Yes	48	1	nontypical	110	229	0	0	168
18	No	54	1	asymptomatic	140	239	0	0	160
19	No	48	0	nonanginal	130	275	0	0	139
20	No	49	1	nontypical	130	266	0	0	171

Example: heart disease





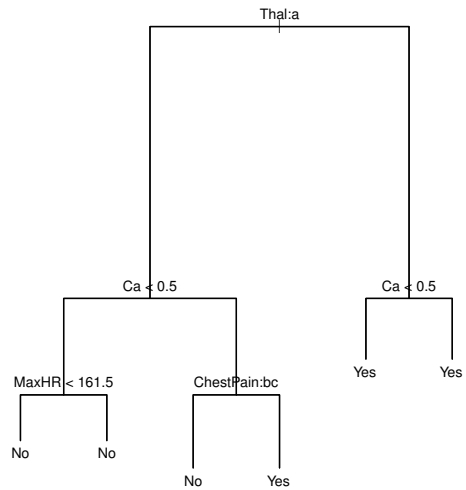


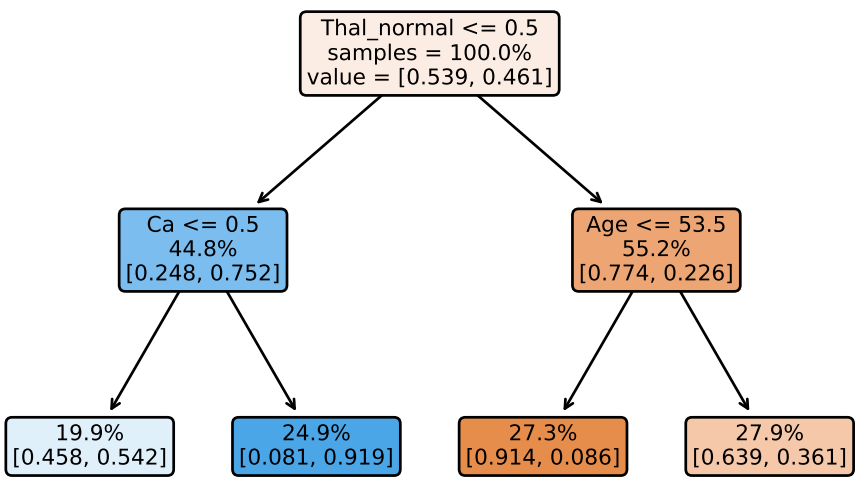
Figure: `Heart` data. Top: The unpruned tree. Bottom left: CV error, training, and test error, for different sizes of the pruned tree. Bottom right: The pruned tree corresponding to the minimal CV error.

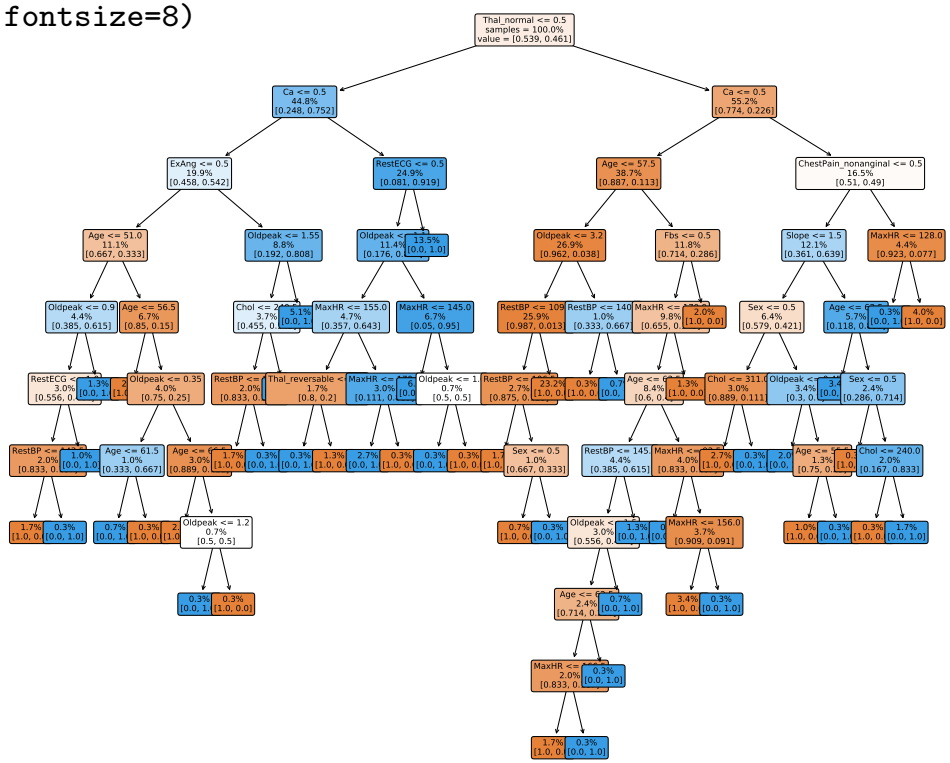
Comments

- In the above tree there are some qualitative predictors, e.g., Thal and ChestPain.
- For a predictor with r possible unordered values, the tree considers the best split into one of the $2^{r-1} - 1$ possible partitions of the r values into two groups.
- For instance, the split ChestPain:bc splits the tree such that the left-hand branch coming out of that node consists of observations with the second and third values of the ChestPain variable.
- Some of the splits yield two terminal nodes that have the same predicted value. The split is performed because it leads to increased node purity, that is, in one of the terminal nodes we are more certain about the prediction.
- Trees can handle missing predictors values; for details see Section 9.2.4 in [ESL].

Tree growing in python

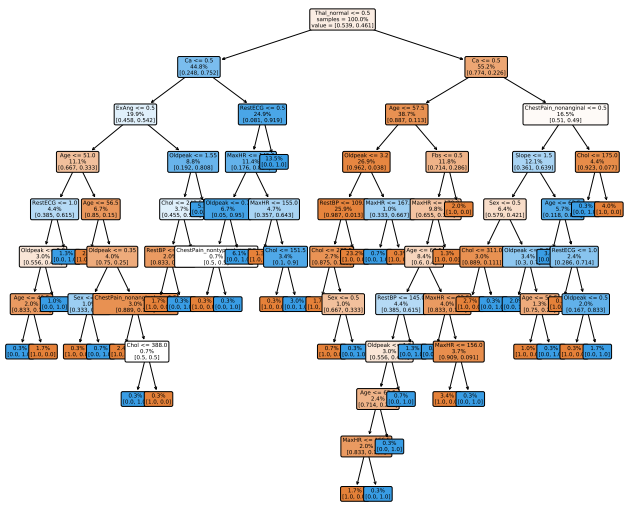
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
heart = pd.read_csv("Heart.csv")
X, y = heart.drop("AHD", axis=1), heart["AHD"]
tree1 = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=50)
tree1.fit(X, y)
plot_tree(tree1, feature_names=X.columns.tolist(), impurity=False, label="root",
          filled=True, proportion=True, rounded=True, fontsize=8)
tree2 = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=1)
tree2.fit(X, y)
plot_tree(tree2, feature_names=X.columns.tolist(), impurity=False, label="root",
          filled=True, proportion=True, rounded=True, fontsize=8)
```

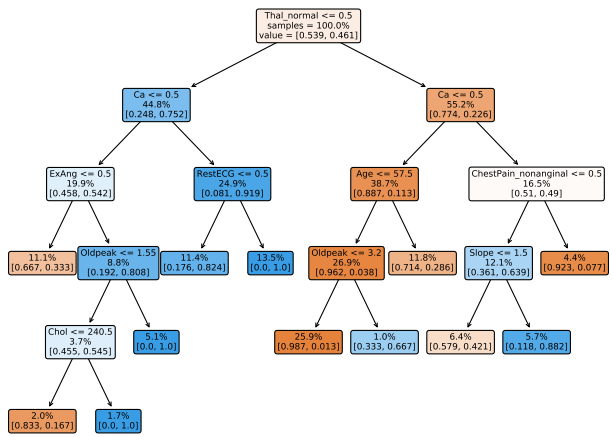


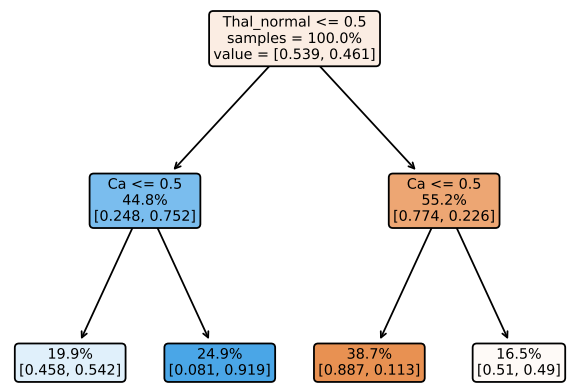


Tree pruning in python

```
tree_full = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=1, ccp_alpha=0)
tree_full.fit(X, y)
plot_tree(tree_full, feature_names=X.columns.tolist(), impurity=False, label="root",
          filled=True, proportion=True, rounded=True, fontsize=8)
tree_pruned1 = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=1, ccp_alpha=0.02)
tree_pruned1.fit(X, y)
plot_tree(tree_pruned1, feature_names=X.columns.tolist(), impurity=False, label="root",
          filled=True, proportion=True, rounded=True, fontsize=8)
tree_pruned2 = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=1, ccp_alpha=0.05)
tree_pruned2.fit(X, y)
plot_tree(tree_pruned2, feature_names=X.columns.tolist(), impurity=False, label="root",
          filled=True, proportion=True, rounded=True, fontsize=8)
```

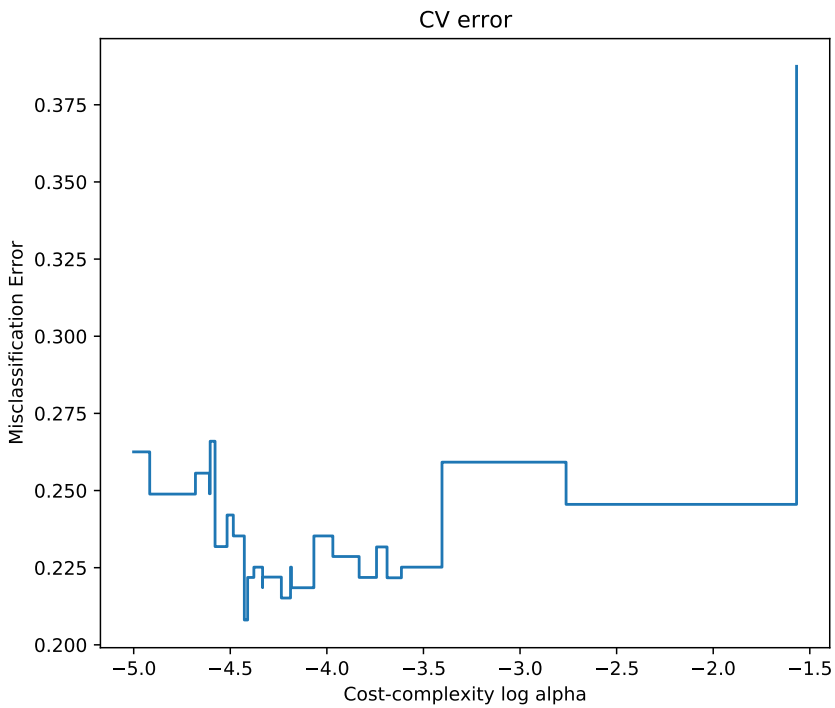


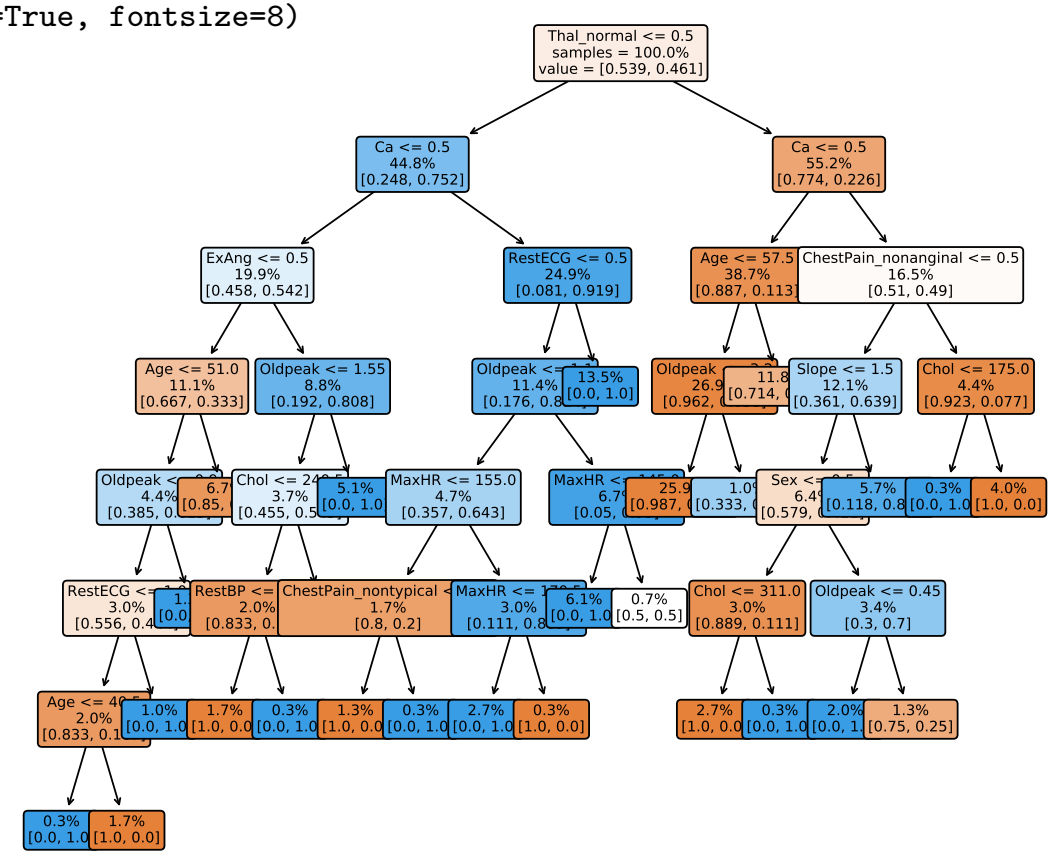




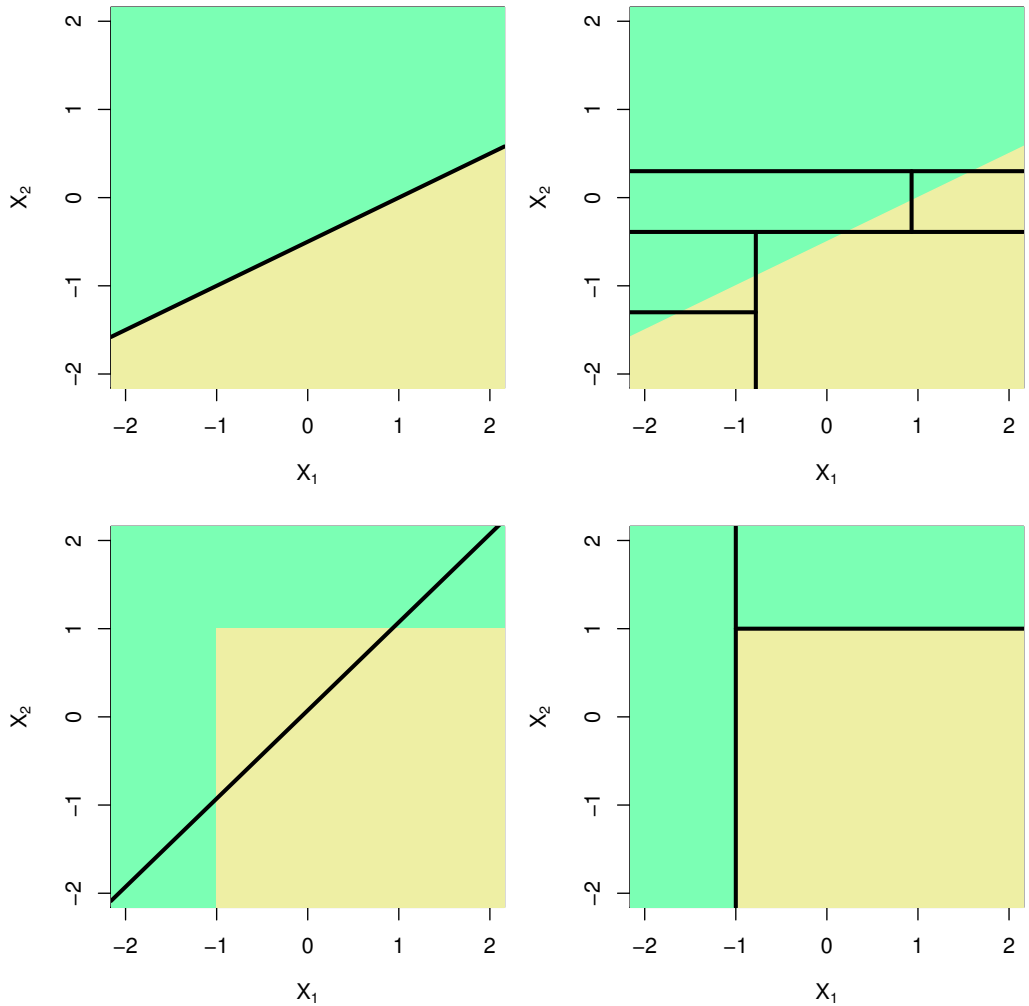
Cost-complexity pruning in python

```
tree_full = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=1, ccp_alpha=0)
path = tree_full.cost_complexity_pruning_path(X, y)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
hyper_parameters = {"ccp_alpha" : grid_alphas}
treeCV = GridSearchCV(estimator=tree_full, param_grid=hyper_parameters, cv=folds)
treeCV.fit(X, y)
best_alpha = alpha_grid[np.argmin(1-treeCV.cv_results_["mean_test_score"])]
best_tree = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=1, ccp_alpha=best_alpha)
best_tree.fit(X,y)
plot_tree(best_tree, feature_names=X.columns.tolist(), impurity=False, label="root",
          filled=True, proportion=True, rounded=True, fontsize=8)
```





Trees versus linear classifiers



Pros and cons of trees

Pros:

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- They mirror human decision-making.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert.
- Can handle qualitative predictors without using dummy variables.

Cons:

- Trees generally have poor predictive performance.
- A small change in the data can completely change the tree.

Although a single tree usually gives poor predictions, very accurate models can be obtained by combining many trees (bagging, random forests, boosting).