CITS5508 Machine Learning

Débora Corrêa (Unit Coordinator and Lecturer)

2023

Quiz

______ margin classification is a SVM model where the objective is to find a good balance between keeping the street as large as possible and limit the margin violations. For SVM classification, margin violations are instances falling ______ the margin. For SVM regression, margin violations are instances falling ______ the margin.

The words that correctly fill the gaps in the above sentence are:

(a) Hard, outside, within.
(b) Hard, within, outside.
(c) Soft, outside, within.
(d) Soft, within, outside.

Quiz

Consider the training objective for a SVM classification model:

$$\underset{\mathbf{w},b,\boldsymbol{\zeta}}{\text{minimize}} \ \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{m}\zeta^{(i)}$$

$$\text{subject to: } t^{(i)}\left(\mathbf{w}^\top\mathbf{x}^{(i)} + b\right) \geq 1 - \zeta^{(i)} \text{ and } \zeta^{(i)} \geq 0, \ \text{for } i = 1, 2, \cdots, m$$

where $\mathbf{w}$ is the feature weights vector, $\mathbf{x}^{(i)}$ is the feature vector of instance i, $\zeta^{(i)} \geq 0$ measures how much the i^{th} instance is allowed to violate the margin, $t^{(i)} = 1$ if training data i is a positive instance, and $t^{(i)} = -1$ if training data i is a negative instance.

I. Changes in the hyperparameter C controls margin violations.
II. For SVM classification models, reducing C makes the street larger, increasing margin violations. If your model is overfitting, you can regularize it by reducing C.
III. For SVM classification models, reducing C makes the street smaller, reducing margin violations. If your model is overfitting, you can regularize it by increasing C.

Which of the alternatives is correct?

(a) Two sentences are wrong.

(b) Two sentences are correct.

(c) All sentences are wrong.

(d) All sentences are correct.

Quiz

Briefly explain the role of the margin when training a SVM classification model. After training, what is the used as the decision boundary for prediction?

Quiz

Briefly explain the role of the margin when training a SVM classification model. After training, what is the used as the decision boundary for prediction?

Example answer: Training a SVM means finding the maximum margin hyperplane that maximizes the distance between the classes. During training, the margins are defined by the hyperplane $\mathbf{w}^T\mathbf{x}^{(i)} + b = 1$ *for the positive class, and the hyperplane* $\mathbf{w}^T\mathbf{x}^{(i)} + b = -1$ *for the negative class. Instances falling inside the margin are considered margin violations. The objective of the SVM classification training function is to maximize the margin while limiting the number of margin violations. After training, the decision boundary is given by the line in the middle of the margins, defined by* $\mathbf{w}^T\mathbf{x}^{(i)} + b = 0$*. That is, if the result of the weighted sum of the feature vectors is positive, then the predicted class is the positive class; otherwise it is the negative class.*

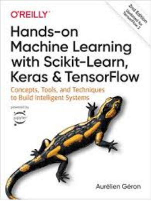
Today

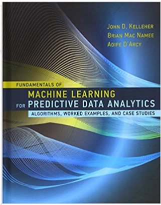
Decision Trees.

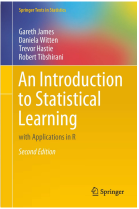
Hands-on Machine Learning with Scikit-Learn & TensorFlow (chapter 6)

Fundamentals of Machine Learning for Predictive Data Analytics (chapter 4)

An Introduction to Statistical Learning (chapter 8)


O'REILLY
2nd Edition
Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
Aurélien Géron


John D. Kelleher
Brian Mac Namee
Aoife D'Arcy
Fundamentals of Machine Learning for Predictive Data Analytics
Algorithms, Worked Examples, and Case Studies



Decision Trees (DTs)

Like SVMs, DTs are versatile ML algorithms that can perform both classification and regression tasks. They are very powerful algorithms, capable of fitting complex datasets.

Classification and Regression Trees (CARTs) splits data using predictor variables where end nodes have the prediction for the target value.

DTs are also the fundamental components of Random Forests, which are popular ML algorithms available today.

Topics

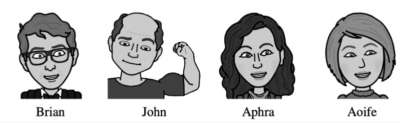
Here are the main topics we will cover:

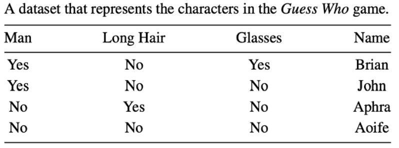
- Training and Visualising a Decision Tree
- Making Predictions
- Estimating Class Probabilities
- The CART Training Algorithm
- Computational Complexity
- Gini Impurity or Entropy
- Regularization Hyperparameters
- Pruning
- Regression
- Limitations

General Idea

Guess Who, a two-player game: one player chooses one card from the deck containing a picture of a character and the other player tries to guess which character is on the card by making a series of questions for which the answers can only be “yes” or “no”.

To win the game, you need to guess the character with a small number of questions.



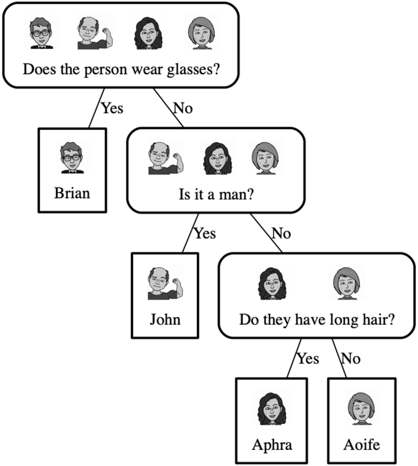

A dataset that represents the characters in the *Guess Who* game.

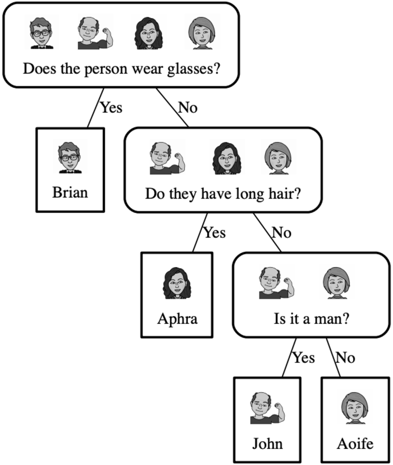
Man	Long Hair	Glasses	Name
Yes	No	Yes	Brian
Yes	No	No	John
No	Yes	No	Aphra
No	No	No	Aoife

Someone picked `Brian`. What you should ask first?

1. Is it a man?
2. Does the person wear glasses?

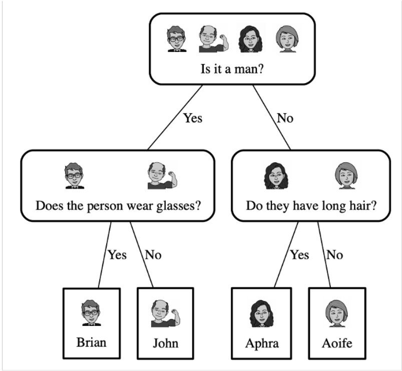
General Idea (Cont.)





The average number of questions one needs to ask per game is $\frac{1+2+3+3}{4} = 2.25$.

General Idea (Cont.)



The average number of questions one needs to ask per game is $\frac{2+2+2+2}{4} = 2$.

On average, an answer to Q1 is more informative than an answer to Q2.

Regression Decision Tree: Example

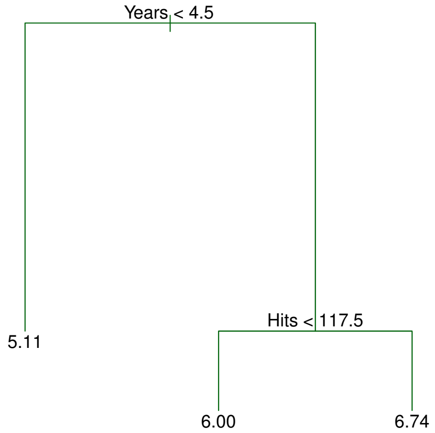
Hitters: Major League Baseball Data from the 1986 and 1987 seasons[1].

Task: predict the `Salary` (1987 annual salary on opening day in thousands of dollars) based on `Years` (number of years in the major leagues) and `Hits` (number of hits in 1986).

Fit a regression tree on log(Salary).

[1] Available at https://rdrr.io/cran/ISLR/man/Hitters.html

Regression Decision Tree: Example



Terminal (leaf) nodes: the mean of the response variable for the instances that fall on that node (5.11, 6.00, 6.74).

Regression Decision Tree: Example interpretation

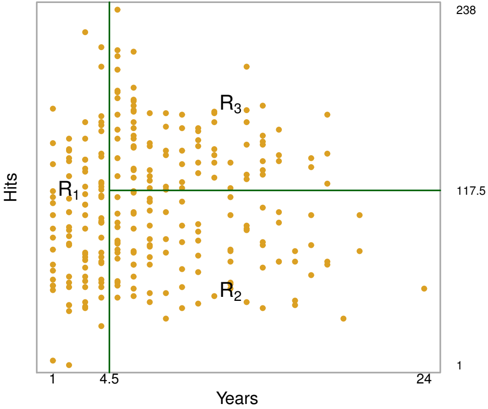
The first split is on `Years`.

- Inexperienced: players having four or fewer years experience in major leagues.

The second split is on `Hits`.

- Experienced, but not so good hitters: players with four or more years experience that made 117 or fewer hits in 1986.
- Experienced, good hitters: players with four or more years experience that made 118 or more hits in 1986.

Regression Decision Tree: Example Alternative Visualisation



Terminal (leaf) nodes presented as three regions.

Regression Decision Tree: Example Prediction

We can use the predicted salary for each region:

- Inexperienced: $\exp^{5.107} = \$165,174$
- Experienced, not good hitters: $\exp^{5.999} = \$402,834$
- Experienced, good hitters: $\exp^{6.740} = \$845,346$

The decision tree was fitted using $log(Salary)$, therefore we need to transform back for the predictions to be on the original scale.

Training and Visualising a Decision Tree

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
iris = load_iris(as_frame=True)
X_iris = iris.data[["petal length (cm)", "petal width (cm)"]
y = iris.target
tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X_iris, y_iris)
```

Then can use `export_graphviz()` method to output a graph definition file and display using graphviz package.

Decision Tree Visualized

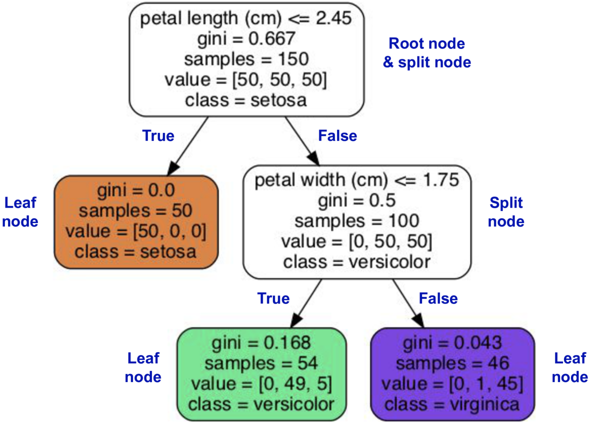


Figure 6-1. Iris Decision Tree

(**Note:** To run this chapter's Python code of the author, you must install an additional package using the command: `conda install python-graphviz`.)

Terminology

Split (or internal) node: A node within the tree. At a given internal node:

- The split label (of the form $X_j <= t_j$) indicates the left-hand branch resulting from that split
- Conversely, the right-hand branch corresponds to $X_j > t_j$.

Branch: Links or edges connecting nodes.

Leaf (or terminal) node: A node at the end of a branch, where prediction is done.

The three is the resulting structure including all these components.

The overall idea of a decision tree model is to find out which feature is more informative to ask questions about, and consider the effects of different answers to these questions.

Making Predictions

- Use the tree to make a decision for a single instance by following the sequence of questions and answers down from the node, left or right depending on the answer.
- Class of leaf node is the output class.
- Tree nodes also tell us number of training samples allocated to each node, and broken down by class.
- *Gini impurity* (low if most of the training instances on that node are from just one class):

  $$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

  where $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

A “White-Box” Classifier: (i.e. human understandable)

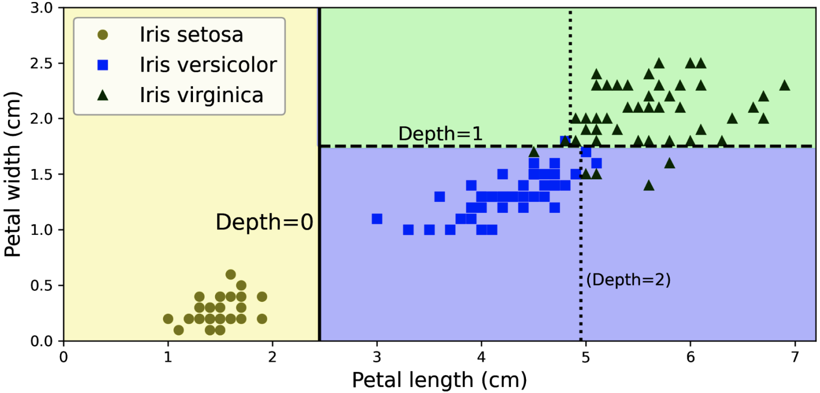
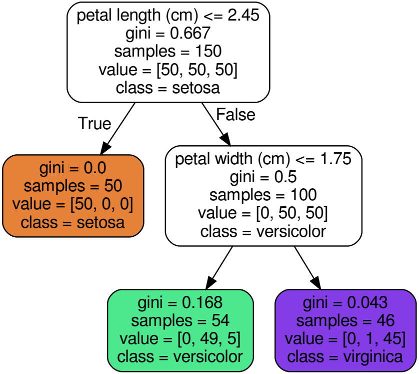


Figure 6-2. Decision Tree decision boundaries

- *White box models* – e.g., decision trees
- *Black box models* – e.g., random forests, neural networks

Estimating Class Probabilities

You can use the class breakdowns in each leaf node to get estimates of the probabilities of class membership for each instance that ends up at a certain leaf.



```
>>> tree_clf.predict_proba([[5, 1.5]])
array([[ 0. , 0.90740741, 0.09259259]])
>>> tree_clf.predict([[5, 1.5]])
array([1])
```

The CART Training Algorithm

Classification And Regression Tree (CART) is the algorithm used to train Decision Trees.

- Divide the feature space - the set of possible values for $X_1, X_2, ..., X_n$ - into J distinct and non-overlapping regions, $R_1, R_2, ..., R_J$
- *Regression*: Predict every instance that falls into the region R_j as the mean of the responsible values for the training instances in R_j
- *Classification*: Predict every instance that falls into the region R_j as the class with the highest probability given the training instances in R_j

The CART Training Algorithm

The considered region shapes are boxes.

The boxes are determined by a recursive binary splitting

- Top down: it starts at the top of the tree (the root node) and goes down.
- *Greedy algorithm*: it searches for an optimal split at the node level, then repeats the process recursively at subsequent levels. But overall solution is not guaranteed to be optimal.

Thus, the algorithm:

- Finds the best predictor X_j and cutpoint t_j that minimises the cost function ($R_1(j, t_j) = \{X|X_j <= t_j\}$ and $R_2(j, t_j) = \{X|X_j > t_j\}$).
- Repeat the process for the new subspaces.
- Stop when it cannot find a split that will reduce impurity or when a stopping condition is satisfied (e.g. max_depth hyperparameter).

The CART Training Algorithm - Classification

The algorithm first splits the training set in two subsets using a single feature X_j and a threshold t_j (e.g., "petal length" ≤ 2.45 cm?). How does it choose X_j and t_j? It searches for the pair (X_j, t_j) that produces the purest subsets (weighted by their size). The cost function that the algorithm tries to minimize is given by

$$J(X_j, t_j) = \frac{m_{left}}{m} G_{\text{left}} + \frac{m_{right}}{m} G_{\text{right}}$$

where m_{left} and m_{right} are the number of instances in the left and right subsets, respectively.

Note: we use a *Greedy Algorithm* to get a reasonable solution, not necessarily optimal (which would take too long to find).

Computational Complexity

Making predictions requires traversing the Decision Tree from the root to a leaf.

Traversing the Decision Tree requires going through roughly $O(\log_2(m))$ nodes. Since each node only requires checking the value of one feature, the overall prediction complexity is just $O(\log_2(m))$, independent of the number of features. So predictions are very fast, even when dealing with large training sets.

However, the training algorithm compares all features (or less if `max_features` is set) on all samples at each node. This results in a training complexity of $O(nm\log_2(m))$.

Gini Impurity or Entropy

- By default, the Gini impurity measure is used
- *Entropy* is an alternative (by setting the `criterion` hyperparameter to "`entropy`")
- From thermodynamics: *entropy* is a measure of molecular disorder
- Later more widespread use, e.g., in *Shannon's information theory* it measures the average information content of a message
- Frequently used as an impurity measure in ML: a set's entropy is zero when it contains instances of only one class.

Gini Impurity or Entropy

- Definition of Entropy H_i of node i:

$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log_2(p_{i,k})$$

- Most of the time it does not make a big difference which one you use: they lead to similar trees.
- Gini impurity is slightly faster to compute, so good default.
- But when they differ, entropy tends to produce slightly more balanced trees.
- In the Iris dataset example: the depth-2 left node has an entropy equal to $-(49/54)\log_2(49/54) - (5/54)\log_2(5/54) \approx 0.445$.

Regularization Hyperparameters

DTs make few assumptions about the training data: they are thus referred to as *nonparametric models*, very (maybe too) adaptive.

As a result, the tree structure is likely to over fit the data.

A smaller tree with fewer splits (regions) might lead to lower variance and better interpretation at the cost of bias.

Therefore, we need to restrict the maximum depth or prune the decision three.

Regularization Hyperparameters

To avoid overfitting, we need to restrict DTs' freedom during training, e.g., setting the max_depth hyperparameter smaller.

DecisionTreeClassifier has other similar parameters to restrict the shape of the decision tree: min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, max_features.

Increasing min_⋆ hyperparameters or reducing max_⋆ hyperparameters regularizes the model.

Pruning is another regularization strategy that we will discuss in the following.

Example Regularization of a Decision Tree

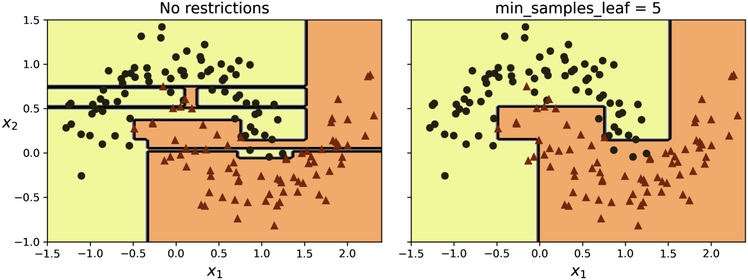


Figure 6-3. Regularization using min_samples_leaf

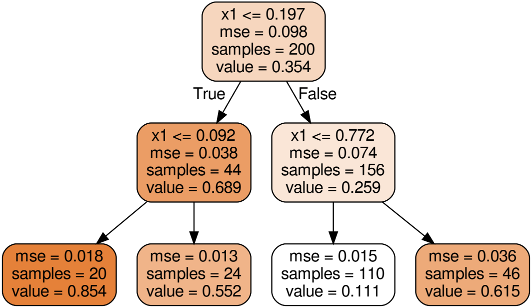
Left: Decision tree trained with the default hyperparameters (i.e., no restrictions)

Right: Decision tree trained with `min_samples_leaf=5`.

The model on the left is overfitting, and the model on the right will probably generalize better.

Regression

Example tree trained on a noisy quadratic (`max_depth=2`).



x1 <= 0.197
mse = 0.098
samples = 200
value = 0.354

True

False

x1 <= 0.092
mse = 0.038
samples = 44
value = 0.689

x1 <= 0.772
mse = 0.074
samples = 156
value = 0.259

mse = 0.018
samples = 20
value = 0.854

mse = 0.013
samples = 24
value = 0.552

mse = 0.015
samples = 110
value = 0.111

mse = 0.036
samples = 46
value = 0.615

Figure 6-4. A Decision Tree for regression

```
import numpy as np
from sklearn.tree import DecisionTreeRegressor

X_quad = np.random.rand(200, 1) - 0.5 # a single random input feature
y_quad = X_quad ** 2 + 0.025 * np.random.randn(200, 1)

tree_reg = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg.fit(X_quad, y_quad)
```

... Or with depth 3 ...

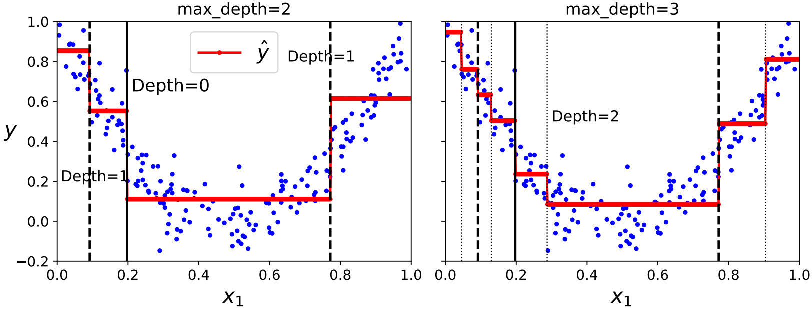


Figure 6-5. Predictions of two Decision Tree regression models

Training consists of trying to minimize the MSE.

CART cost function for regression

$$J(X_j, t_j) = \frac{m_{left}}{m} MSE_{\text{left}} + \frac{m_{right}}{m} MSE_{\text{right}}$$

where

$$MSE_{node} = \frac{\sum_{i} \in \ node \left(\hat{y}_{node} - y^{(i)}\right)^2}{m_{node}}$$

and

$$\hat{y}_{node} = \frac{\sum_{i} \in \ node \ y^{(i)}}{m_{node}}$$

Prone to Overfitting

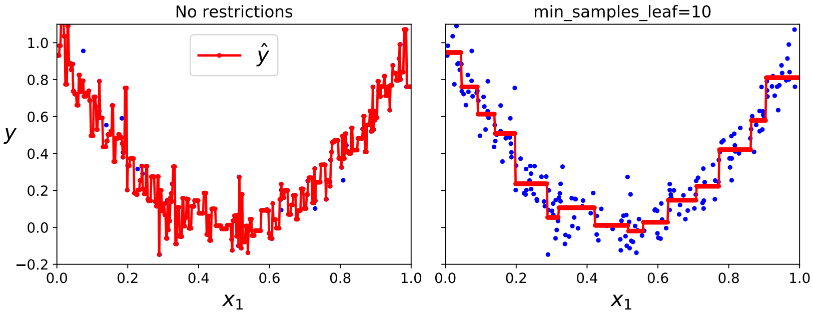


Figure 6-6. Regularizing a Decision Tree regressor

- Left: without any regularization (i.e., using the default hyperparameters). Overfitting the training data is obvious.
- Right: minimum number of samples per leaf node is set to 10, resulting in a much more reasonable model.

Sensitivity to rotation

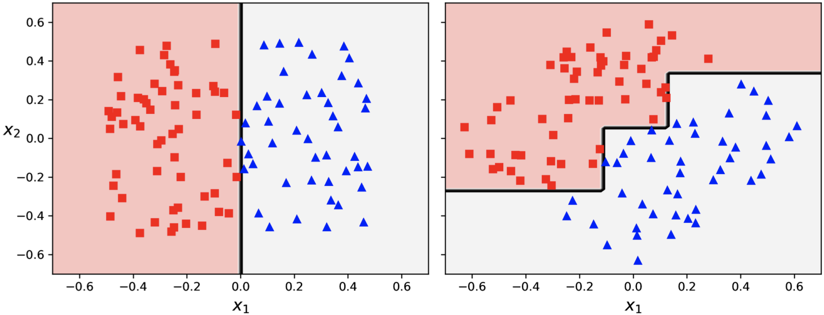


Figure 6-7. Sensitivity to training set rotation

(later in the unit we see PCA which can help)

DTs are High Variance Models

Small changes to the hyperparameters or to the data may produce very different models.

Even retraining the decision tree on the exact same data may produce a very different model.

Random forests (RF) may overcome this.

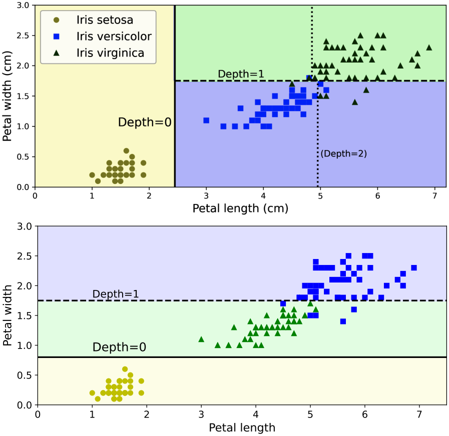


Figure 6-9. Sensitivity to training set details

Pruning a Regression Decision Tree

A smaller tree with fewer splits (that is, fewer regions $R_1, ..., R_J$) might lead to lower variance and better interpretation at the cost of a little bias.

We want to select the tree that gives the lowest error on the test set. One alternative is to estimate the cross-validation error for every possible subtree, but that can be impractical.

Another alternative it to grow a large tree T_0, and then `prune` it back to obtain a `subtree`.

Pruning a Regression Decision Tree

Consider a sequence of trees according to a tuning parameter α. For each value of α there corresponds a subtree $T \subset T_0$ that minimizes

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

where $|T|$ is the number of terminal nodes of the subtree T, R_m is the rectangle corresponding to the mth terminal node and $\hat{y}_{R_m}$ is the predicted response associate with R_m (the mean of the training observations in R_m).

The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.

- $\alpha = 0$: T will equal T_0 (just measures the training error)
- Increasing α: price to pay for having a tree with many terminal nodes (therefore favouring a smaller subtree).

Pruning Algorithm - Key Steps

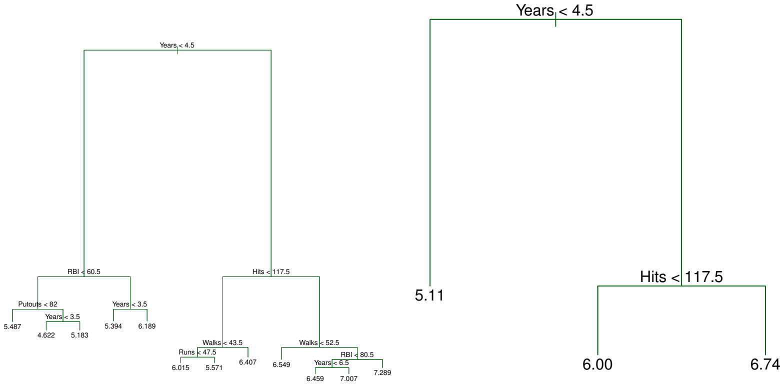
Build a regression tree on the training data.

Obtain a sequence of subtrees with different number of terminal nodes by varying α.

Perform cross-validation to estimate the average validation error for each α or subtree.

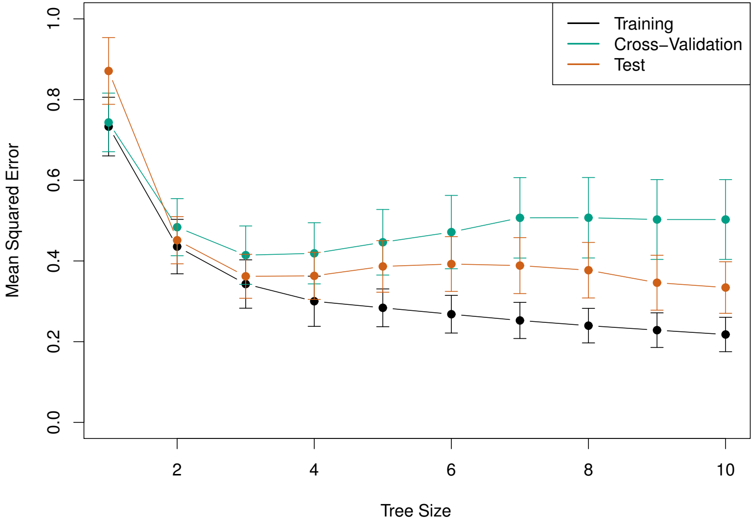
Choose α that minimizes the average error.

Baseball example



Regression tree analysis for the Hitters data. The unpruned tree (left) and pruned tree (right)

Baseball example



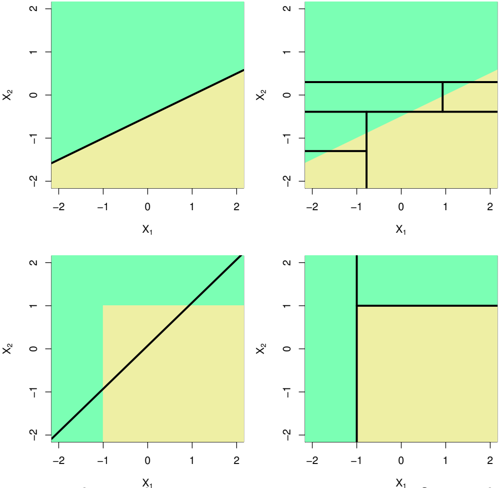
The training, cross-validation, and test MSE as a function of the number of leaf nodes, with standard error bands

Decision Trees versus Linear Models

Decision Trees (DTs) is usually more suitable for non-linear and complex relationships between the features and the response variable. In those cases, DTs may outperform classical approaches.

But, if the relationship between the features and the response variable is well approximated by a linear model, then a linear regression may outperform a DT.

Decision Trees versus Linear Models



Top: Linear decision boundary, linear regression performs better. Bottom: Non-linear decision boundary, decision tree performs better.

Summary for Chapter 6

- Training and Visualising a Decision Tree
- Making Predictions
- Estimating Class Probabilities
- The CART Training Algorithm
- Computational Complexity
- Gini Impurity or Entropy
- Regularization Hyperparameters
- Regression
- Instability

For next lecture

Assignment 2 will be realised this week. Read the instructions on LMS.

Work through Assignment 2 and attend the supervised lab. The Unit Coordinator or a casual Teaching Assistant will be there to help.

Read the instructions for the mid-semester test on LMS. The test day is Tuesday, April 18th, 2023 and will cover all material discussed until and including week 6.

Read Chapter 7 on Ensemble Learning and Random Forests.

And that’s all for the sixth lecture.

Have a good week.