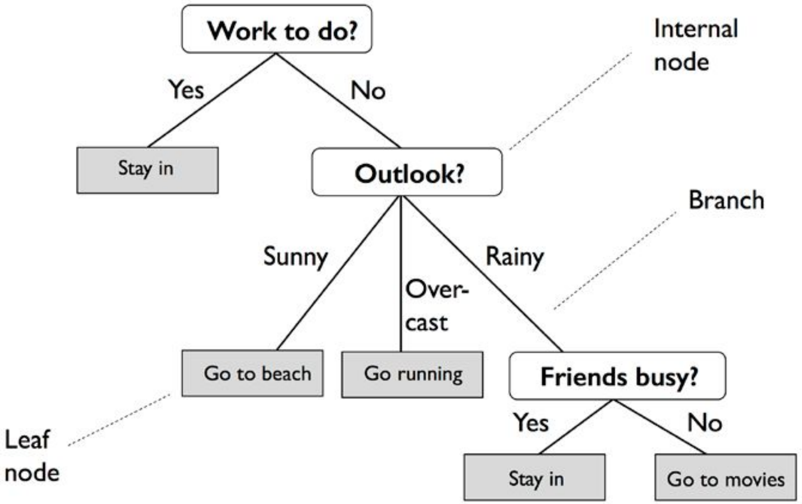
Decision Trees For Classification and Regression

Jose Luis Paniagua Jaramillo
jlpaniagua@uao.edu.co

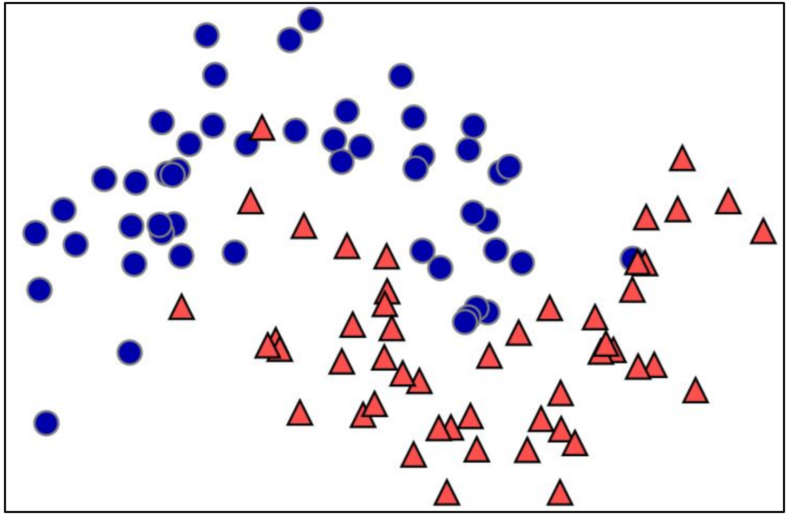
Decision Trees
For Classification

Decision Tree



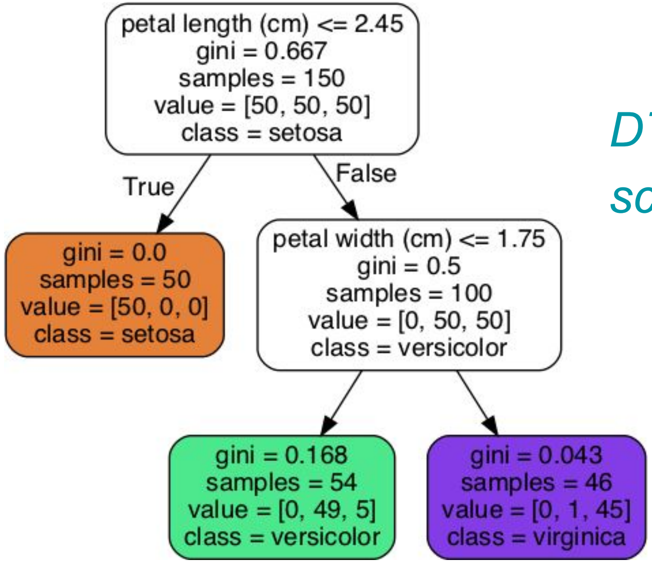
We can think of this model as breaking down our data by making a decision based on asking a series of questions.

Decision Tree



Tests on continuous data are of the form “ Is feature i larger than value a?”

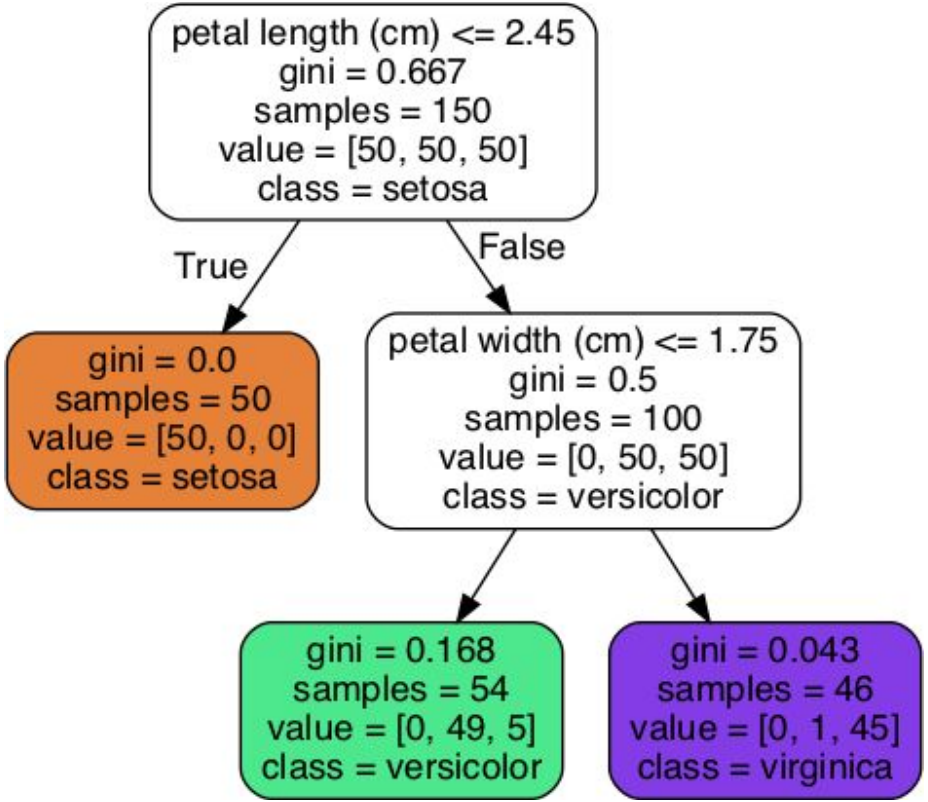
Building DTs



DTs don't require feature scaling or centering at all.

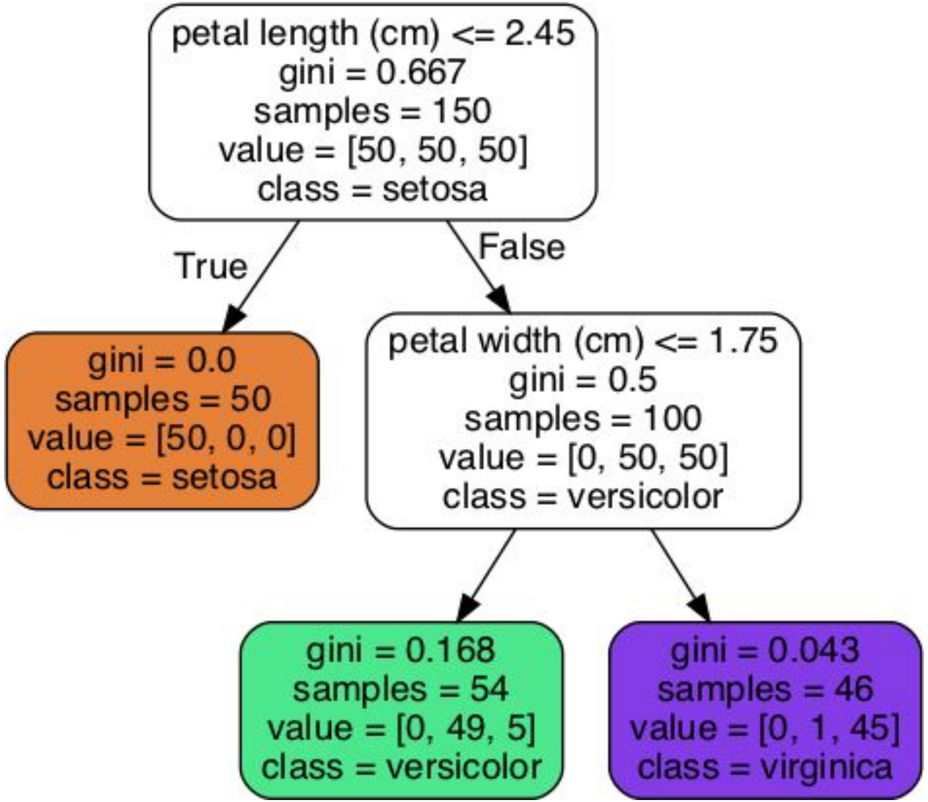
To build a tree: The algorithm searches over all possible tests and finds the one that is most informative about the target variable

Building DTs



- node’s samples attribute counts how many training instances it applies to.
- node’s value attribute tells you how many training instances of each class this node applies to.
- node’s gini attribute measures its impurity: a node is “**pure**” (gini=0) if all training instances it applies to belong to the same class.

Building DTs



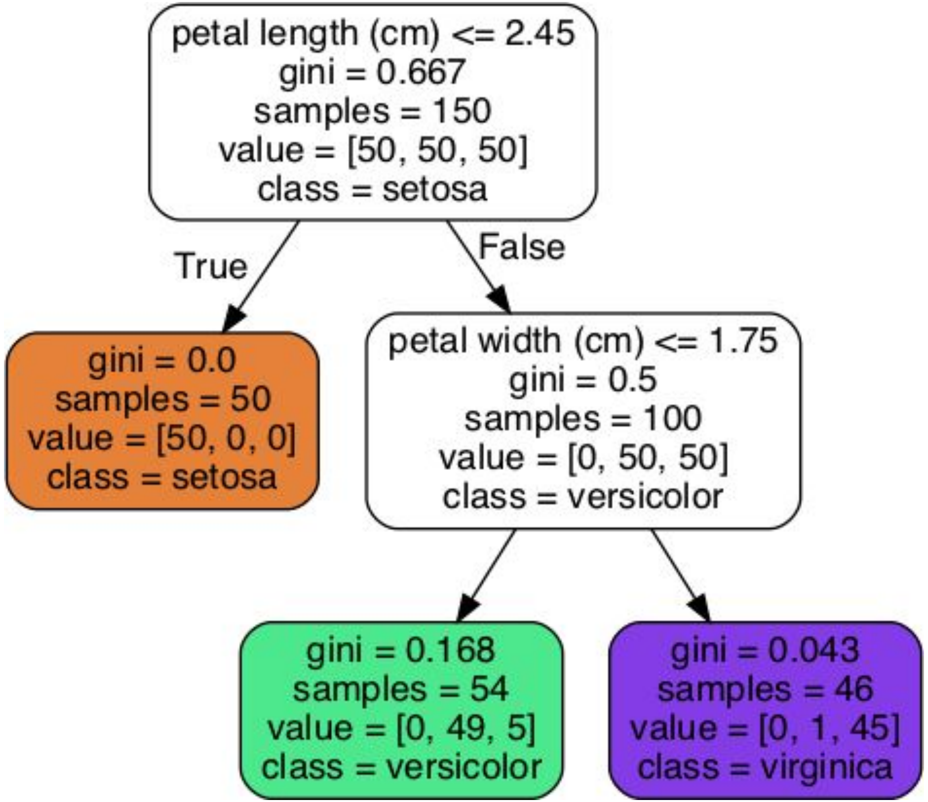
Gini impurity

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

pi,k is the ratio of class k instances among the training instances in the i-th node.

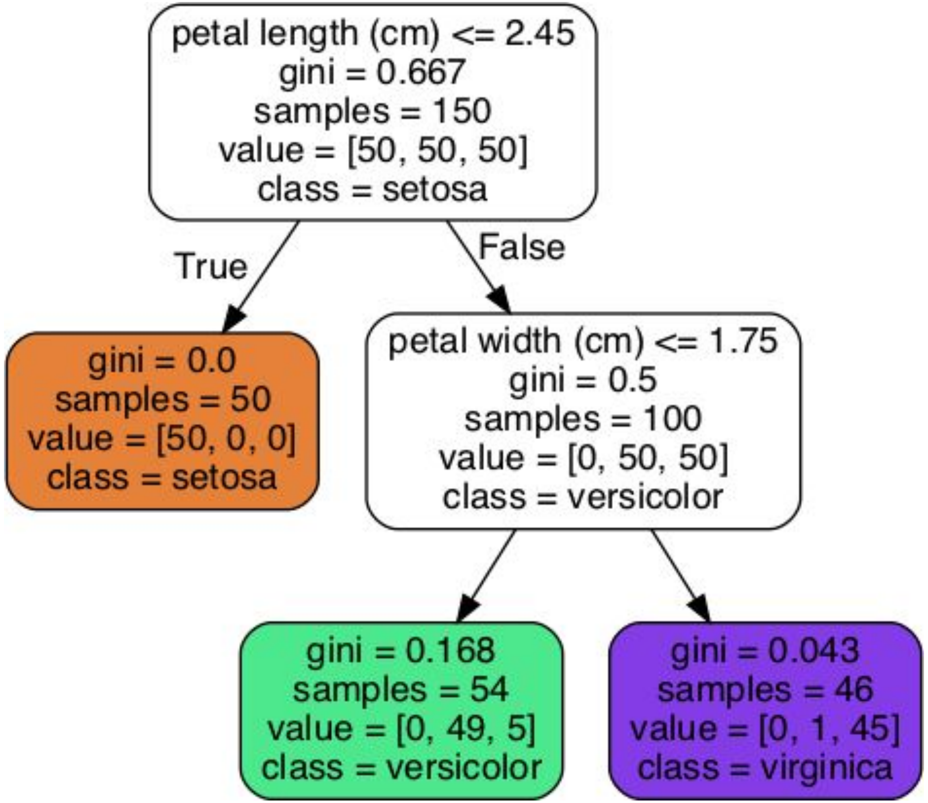
what is the gini score in the depth-2 left node?

DTs: Model Interpretation



Decision tree classifiers are attractive models if we care about interpretability

DTs: Training Algorithm



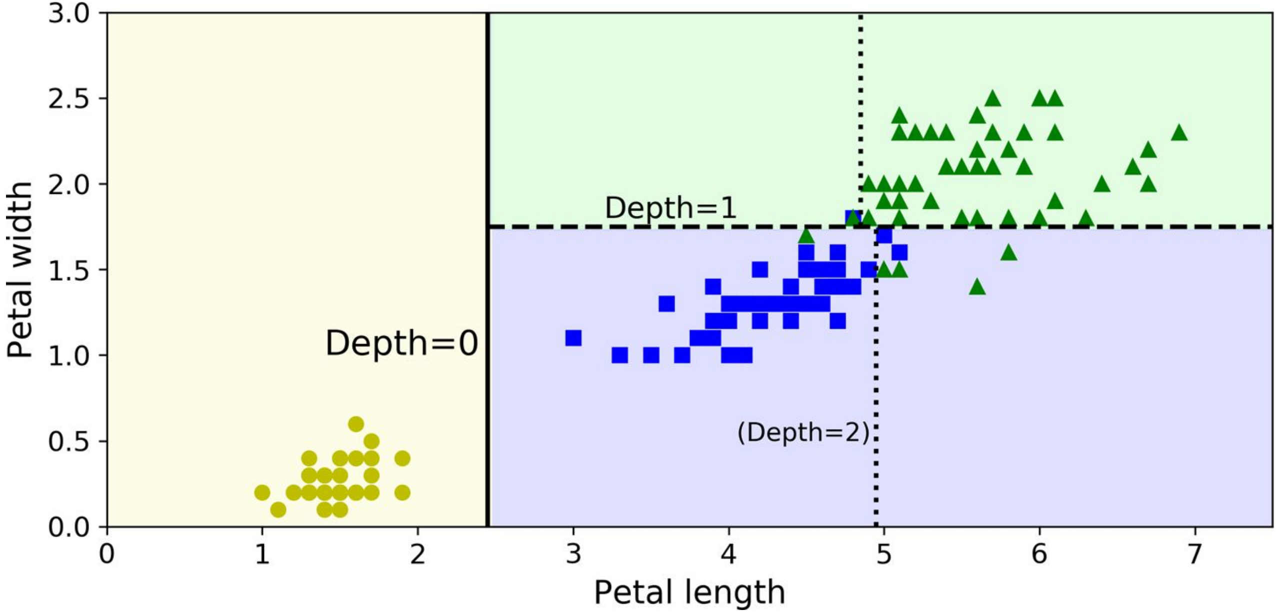
The algorithm works by first splitting the training set into two subsets using a single feature k and a threshold tk. How does it choose k and tk?

DTs: Training Algorithm

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

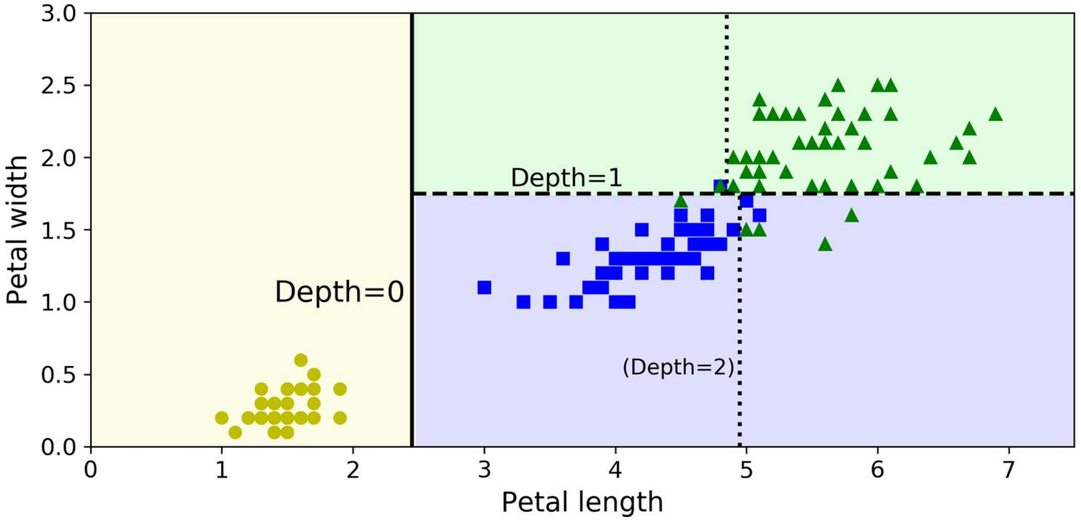
$$\text{where } \begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$$

Predicting with DTs



A prediction on a new data point is made by checking which region of the partition of the feature space the point lies in , and then predicting the majority target or the single target in the case of pure leaves) in that region

DTs: Complexity



Building a tree as described before, continuing until all leaves are pure leads to models that are very complex (overfitting)

Comparing all features on all samples at each node results in a training complexity of

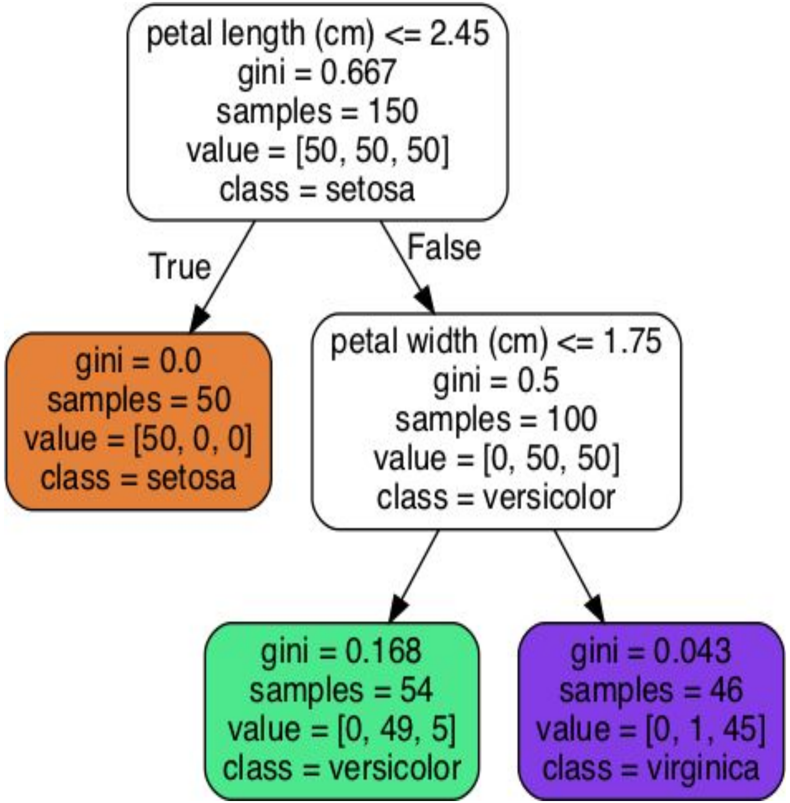
$$O(n \times m \log_2(m))$$

DTs: Controlling Complexity of DTs

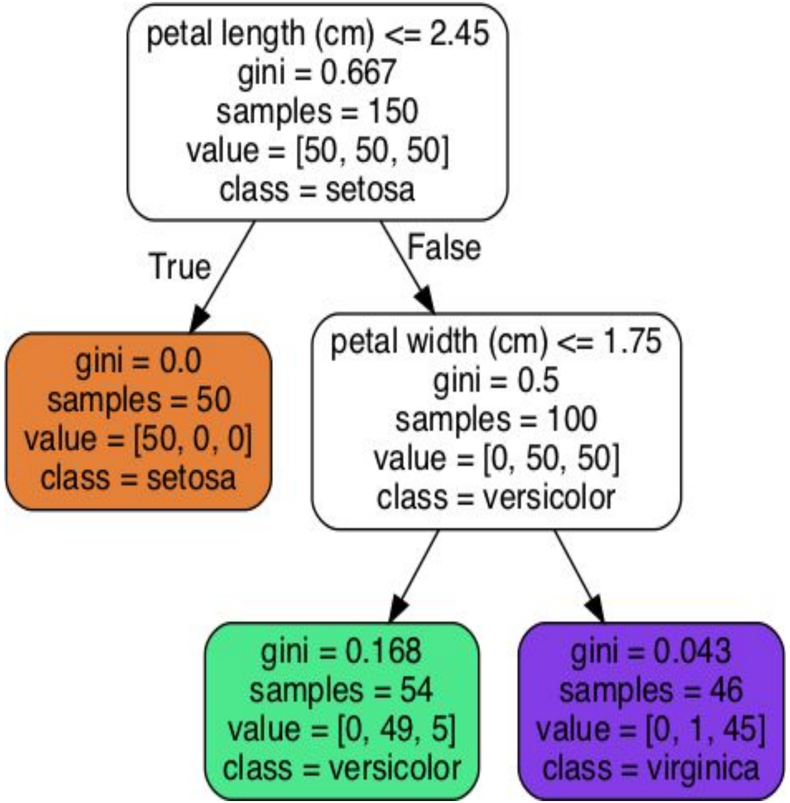
1. Stopping the creation of the tree early pre pruning

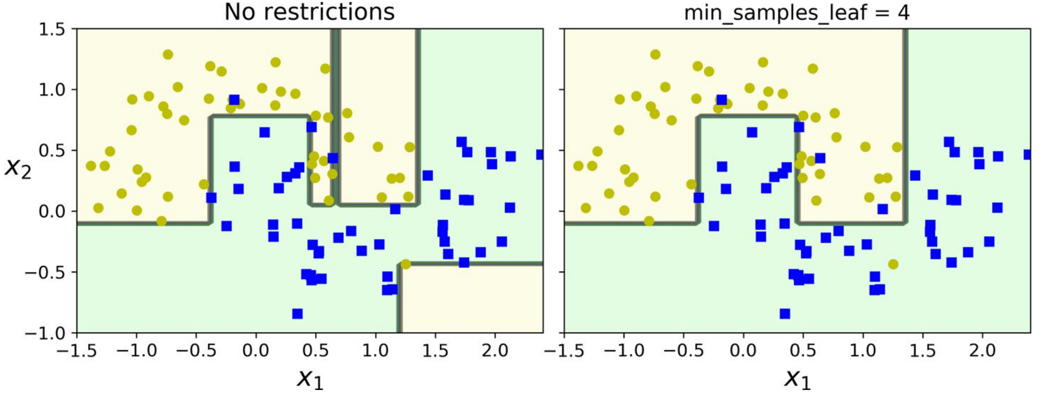
   a. Limiting the maximum depth of the tree (*max_depth*)
   b. Limiting the maximum number of leaves (*max_leaf_nodes*)
   c. Requiring a minimum number of points in a node to keep splitting it (*min_samples_leaf*)

2. Building the tree but then removing or collapsing nodes that contain little information post pruning or pruning



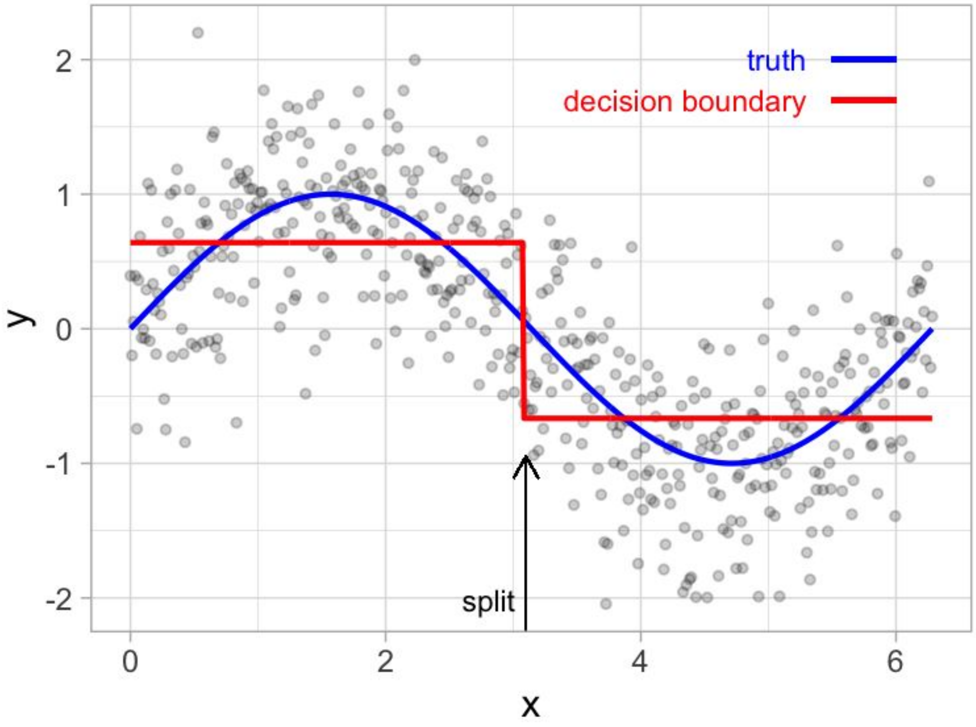
DTs: Controlling Complexity of DTs





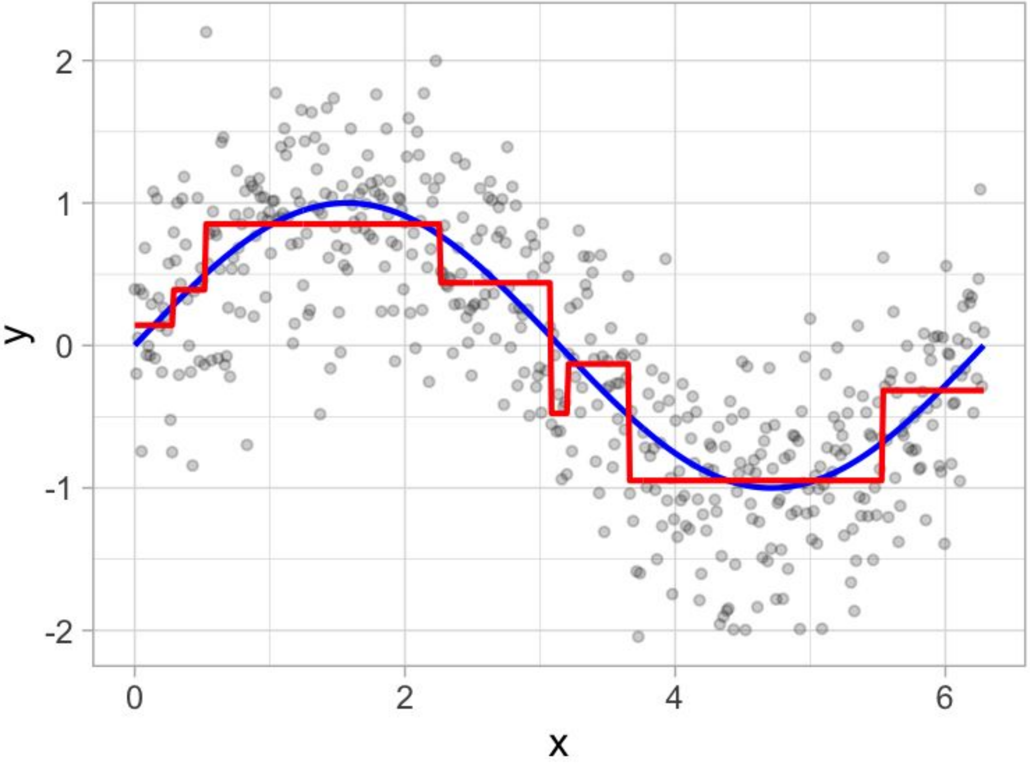
Decision Trees
For Regression

DTs for regression



- Usage and analysis similar to classification DTs.
- DTs splits the data into groups and predicts a fixed value for each of the groups.

DTs for regression



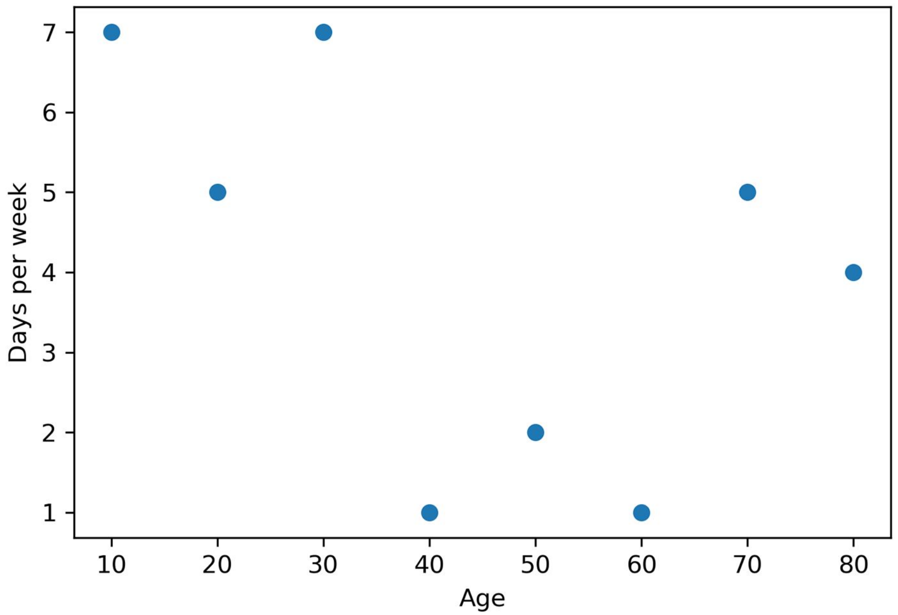
- The way to split the data is by using the features, exactly like we did for classification problems.
-

DTs for regression

- Consider the following problem: we have an app, and we want to predict the level of engagement of the users in terms of how many days per week they used it.

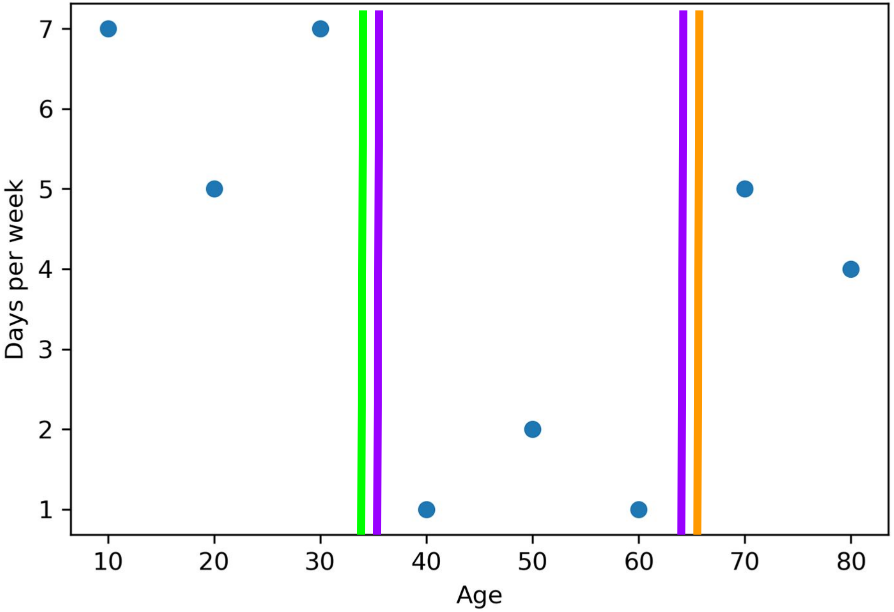
Age	Engagement
10	7
20	5
30	7
40	1
50	2
60	1
70	5
80	4

DTs for regression



How to split the data?

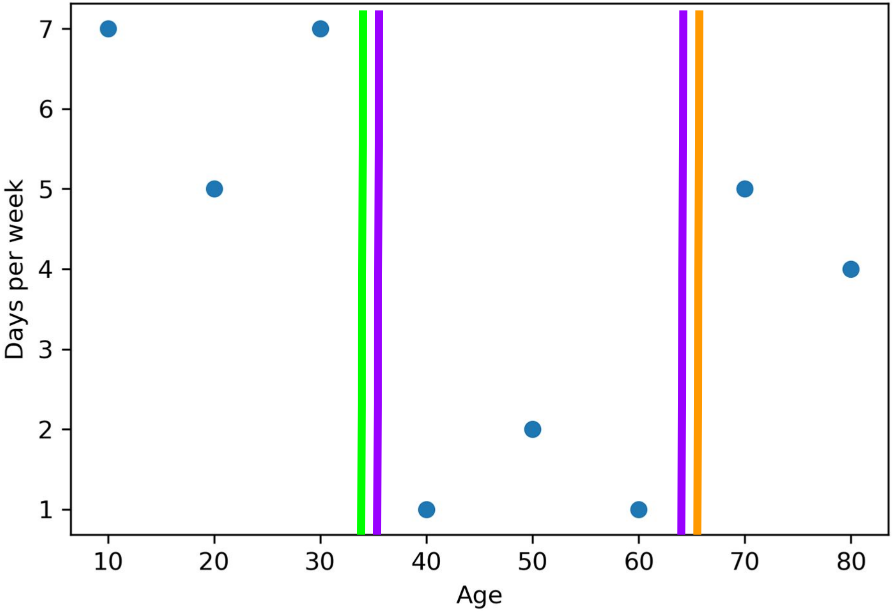
DTs for regression



How to split the data?

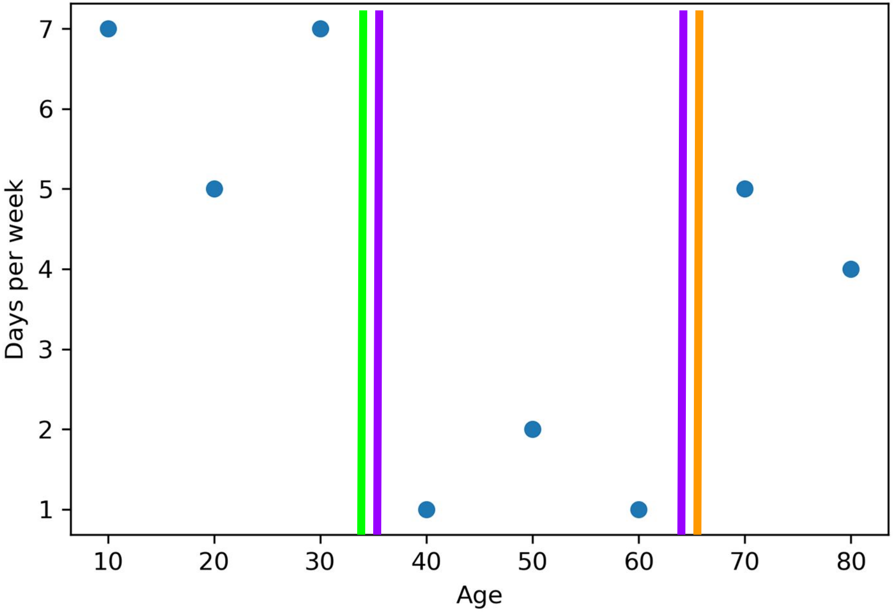
- If the user is 34 years old or younger, the engagement is 6 days per week.
- If the user is between 35 and 64, the engagement is 1 day per week.
- If the user is 65 or older, the engagement is 3.5 days per week.

DTs for regression: training



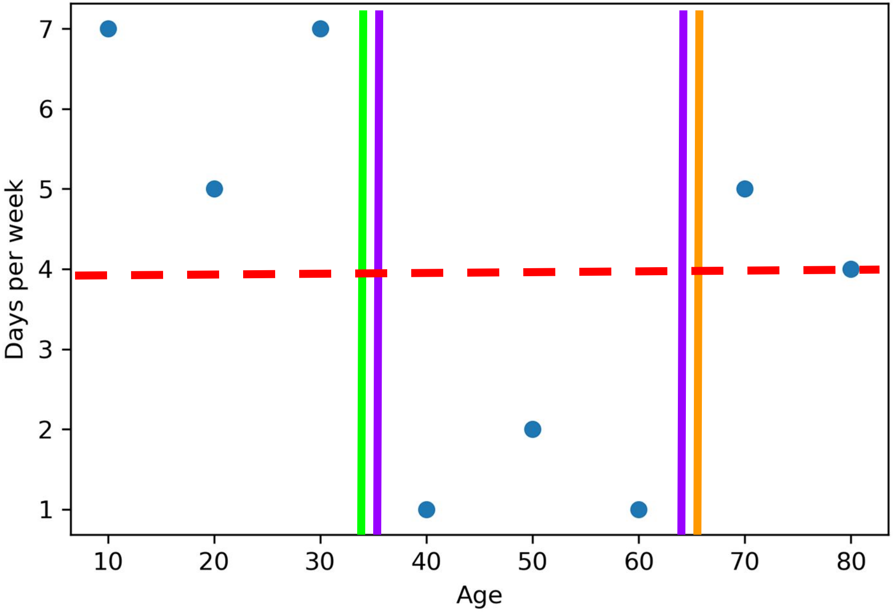
- Now, we have to fit a horizontal line as close as possible to the dataset. **Where should we fit this horizontal line?**

DTs for regression: training



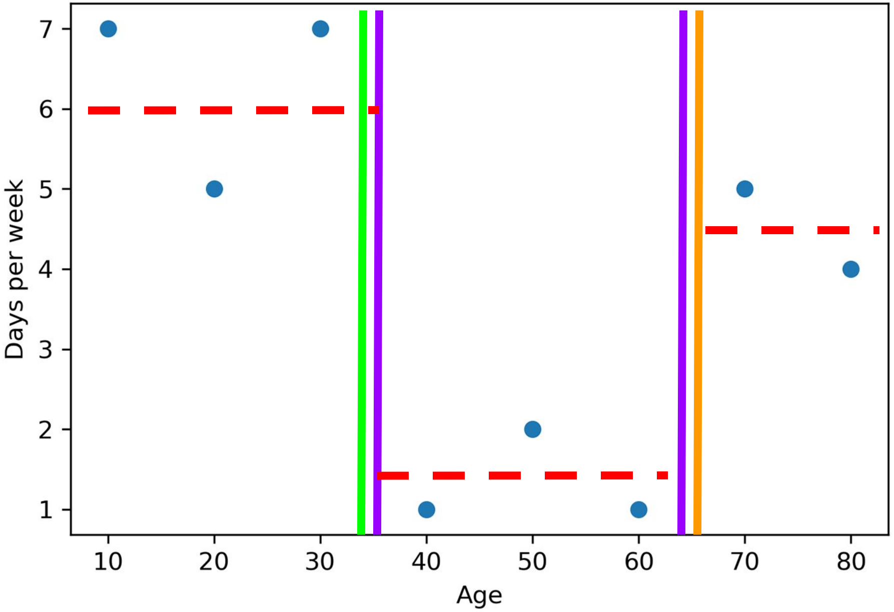
- Now, we have to fit a horizontal line as close as possible to the dataset. **Where should we fit this horizontal line?**

DTs for regression: training



- A reasonable answer is: at a height equal to the average of the target.

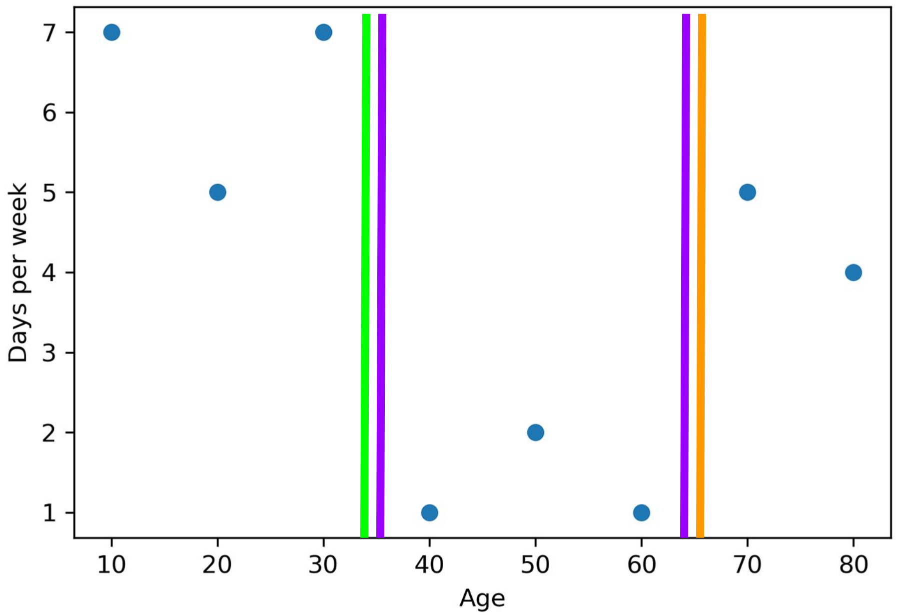
DTs for regression: training



- What If we had to use two horizontal segments, how should we fit them as close as possible to the data?

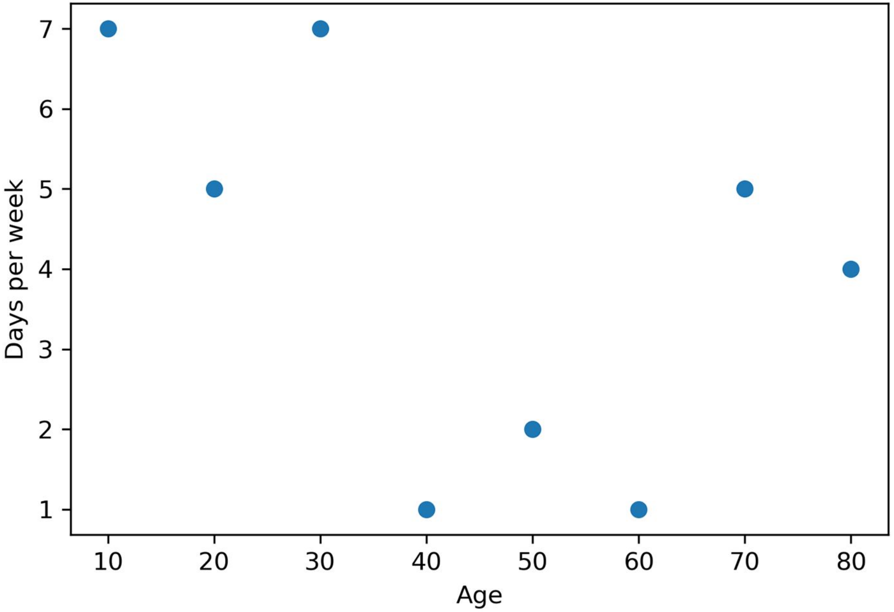
- We can continue following this process until we have broken down the data into several groups in which their targets are very similar.

DTs for regression: training



- The algorithm used for training a regression decision tree is very similar to the one we used for training a classification decision tree.
- The only difference is that for classification trees, Gini index, and for regression trees, we use the mean square error (MSE).

DTs for regression: training



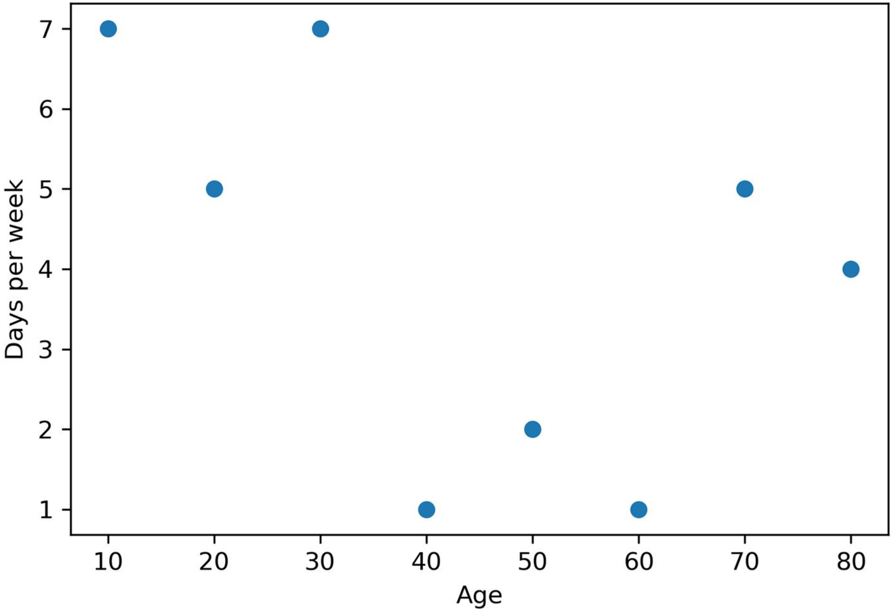
- For training, we carry out the following steps:

1. For each of the smaller datasets, we predict the average value of the labels.

2. We calculate the mean square error of the prediction.

3. We select the cutoff that gives us the smallest square error.

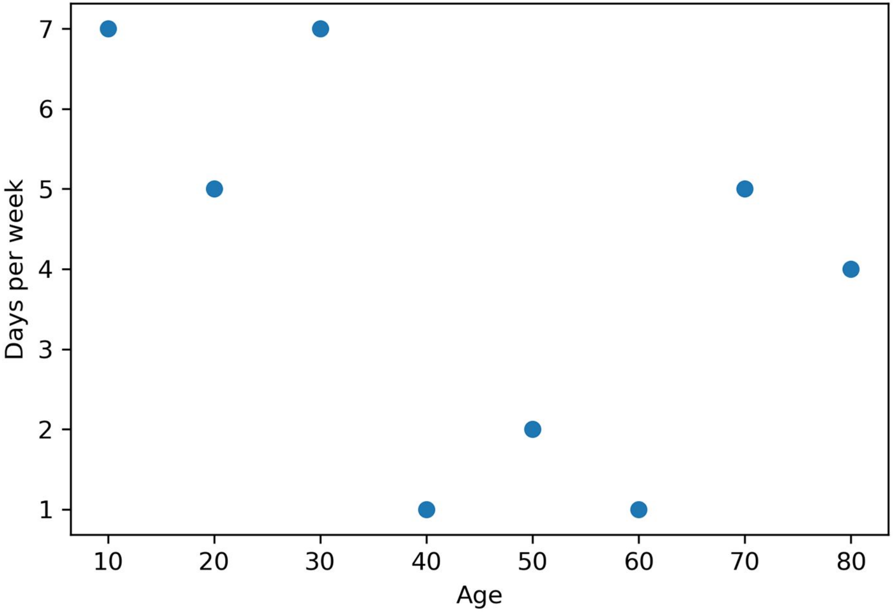
DTs for regression: training



- Recall that when a feature is numerical, we consider all the possible ways to split it.

- The split gives us two smaller datasets, which we call the **left dataset** and the **right dataset.**

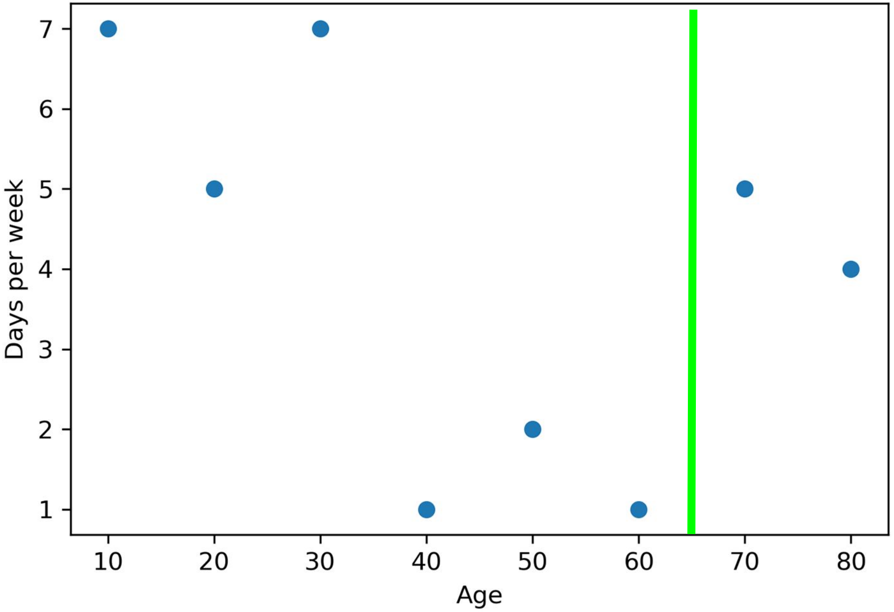
DTs for regression: training



Example:

- Consider a cutoff equal to 65.
- Calculate the two smaller datasets.
- Predict the average of the targets.
- Calculate the MSE.

DTs for regression: training



Example:

- 65 is the best cutoff?

DTs for regression: training

Cutoff	Labels left set	Labels right set	Prediction left set	Prediction right set	MSE
0	{}	{7,5,7,1,2,1,5,4}	None	4.0	5.25
15	{7}	{5,7,1,2,1,5,4}	7.0	3.571	3.964
25	{7,5}	{7,1,2,1,5,4}	6.0	3.333	3.917
35	**{7,5,7}**	**{1,2,1,5,4}**	**6.333**	**2.6**	**1.983**
45	{7,5,7,1}	{2,1,5,4}	5.0	3.0	4.25
55	{7,5,7,1,2}	{1,5,4}	4.4	3.333	4.983
65	{7,5,7,1,2,1}	{5,4}	3.833	4.5	5.167
75	{7,5,7,1,2,1,5}	{4}	4.0	4.0	5.25
100	{7,5,7,1,2,1,5,4}	{}	4.0	none	5.25

The next steps are to continue splitting the left and right datasets recursively in the same fashion.

Parameters and strengths

- Setting either *max_depth* , *max_leaf_nodes* , or *min_samples_leaf* is sufficient to prevent overfitting
- The resulting model can easily be visualised and understood by non experts
- Algorithms are completely invariant to scaling of the data
- Features on different scales, or a mix of binary and continuous features

Weaknesses

- Even with the use of of pre pruning, DTs tend to overfit and provide poor generalisation performance.

- In most applications ensemble methods are usually used in place of a single DT.