CS 559: Decision Tree

Lecture 8

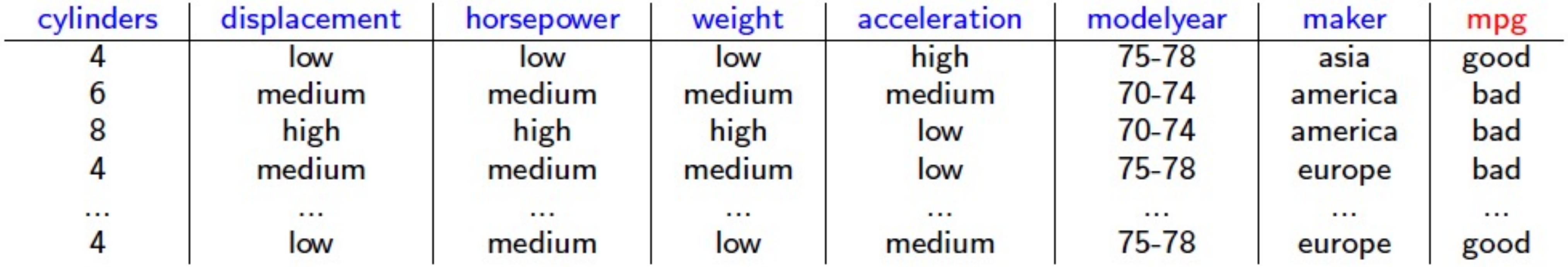
Outline

Decision Tree

- General Idea of Decision Tree
- Classification
- Regression

Decision Tree - Example

Table: From the UCI repository



cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	mpg
4	low	low	low	high	75-78	asia	good
6	medium	medium	medium	medium	70-74	america	bad
8	high	high	high	low	70-74	america	bad
4	medium	medium	medium	low	75-78	europe	bad
...	...	...	...	...	...	...	...
4	low	medium	low	medium	75-78	europe	good

- 40 observations
- Goal: predict MPG (good or bad)
- Need to find: $f: X \rightarrow Y$
- Discrete features/attributes

Decision Tree

- It is also commonly in data mining.
- One of the supervised learning techniques by decision makings.
- It concludes about the target value (leaves) from the observations about the item (branches).
- Used for both classification and regression.

Decision Tree

- Classification And Regression Tree (CART)
  - First introduced by Breiman in 1984
  - *Ensemble methods* – construct more than one decision tree.
  - *Boosted trees* – Build an ensemble by training each new instance to emphasize the training instances previously mid-modeled, e.g., **AdaBoost**.
  - *Bootstrap aggregated (or bagged)* – an early ensemble method that builds multiple decision trees by repeatedly resampling training data with replacement and voting the trees for a consensus prediction, e.g., **random forest**.
  - Other decision tree learning algorithms.
    - ID3, C4.5, Chi-square automatic interaction detection, MARS

Decision Tree - Example

cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	mpg
4	low	low	low	high	75-78	asia	good
6	medium	medium	medium	medium	70-74	america	bad
8	high	high	high	low	70-74	america	bad
4	medium	medium	medium	low	75-78	europe	bad
...	...	...	...	...	...	...	...
4	low	medium	low	medium	75-78	europe	good

$$f(x_i) \rightarrow y$$

- Each internal node tests an attribute x_i
- Each branch assigns an attribute value $x_i = v$
- Each leaf assigns a class
- To classify input x: traverse the tree from root to leaf, output the labeled y

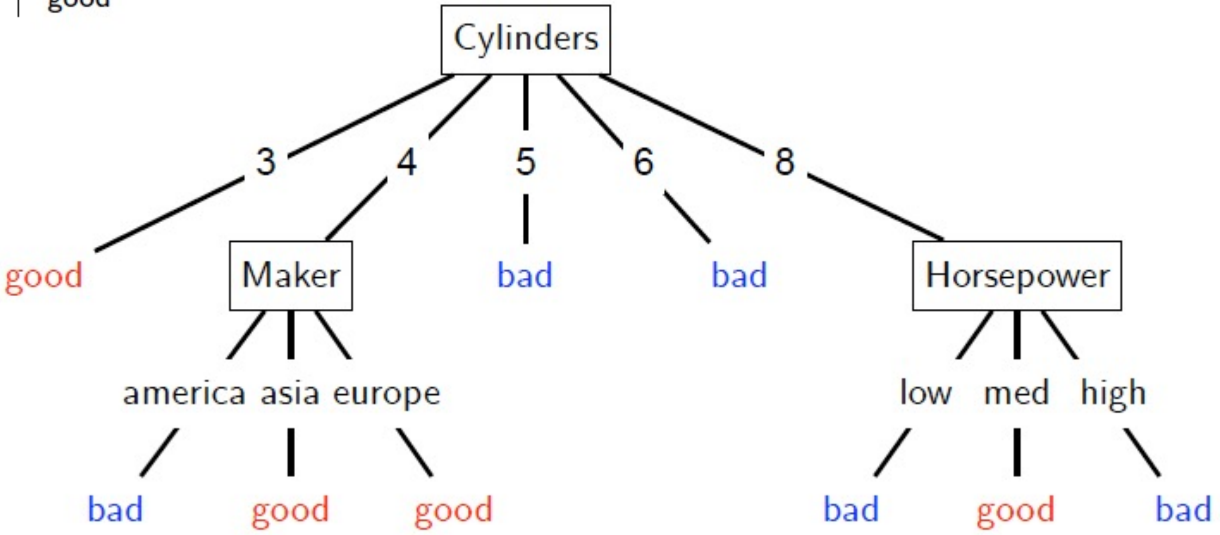


Figure: Human interpretable!

- Decision trees can represent any function of the input attributes
- For Boolean functions, one path to leaf gives a truth table row
- But, it could require exponentially many nodes...

Decision Tree - Example

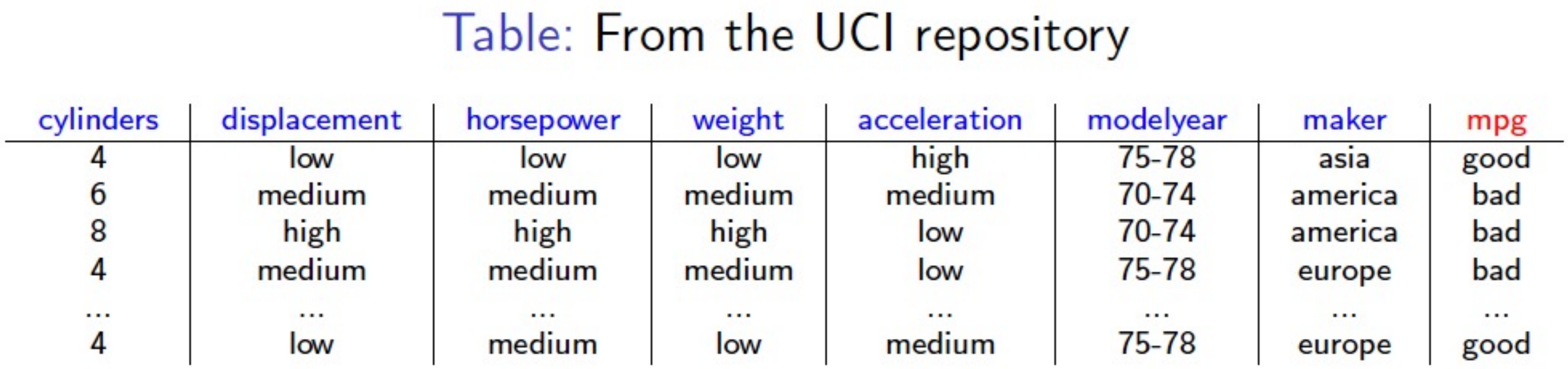
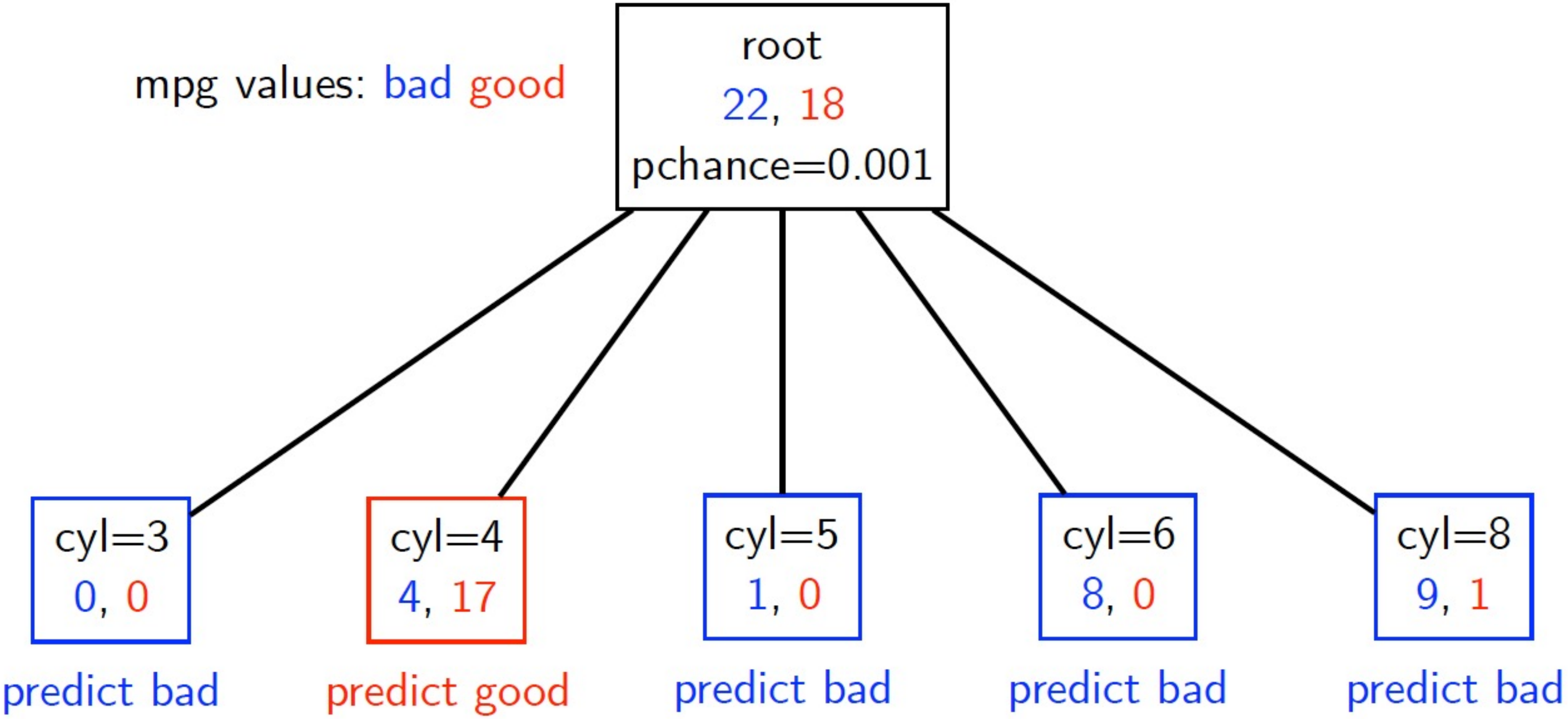


Table: From the UCI repository

cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	mpg
4	low	low	low	high	75-78	asia	good
6	medium	medium	medium	medium	70-74	america	bad
8	high	high	high	low	70-74	america	bad
4	medium	medium	medium	low	75-78	europe	bad
...	...	...	...	...	...	...	...
4	low	medium	low	medium	75-78	europe	good

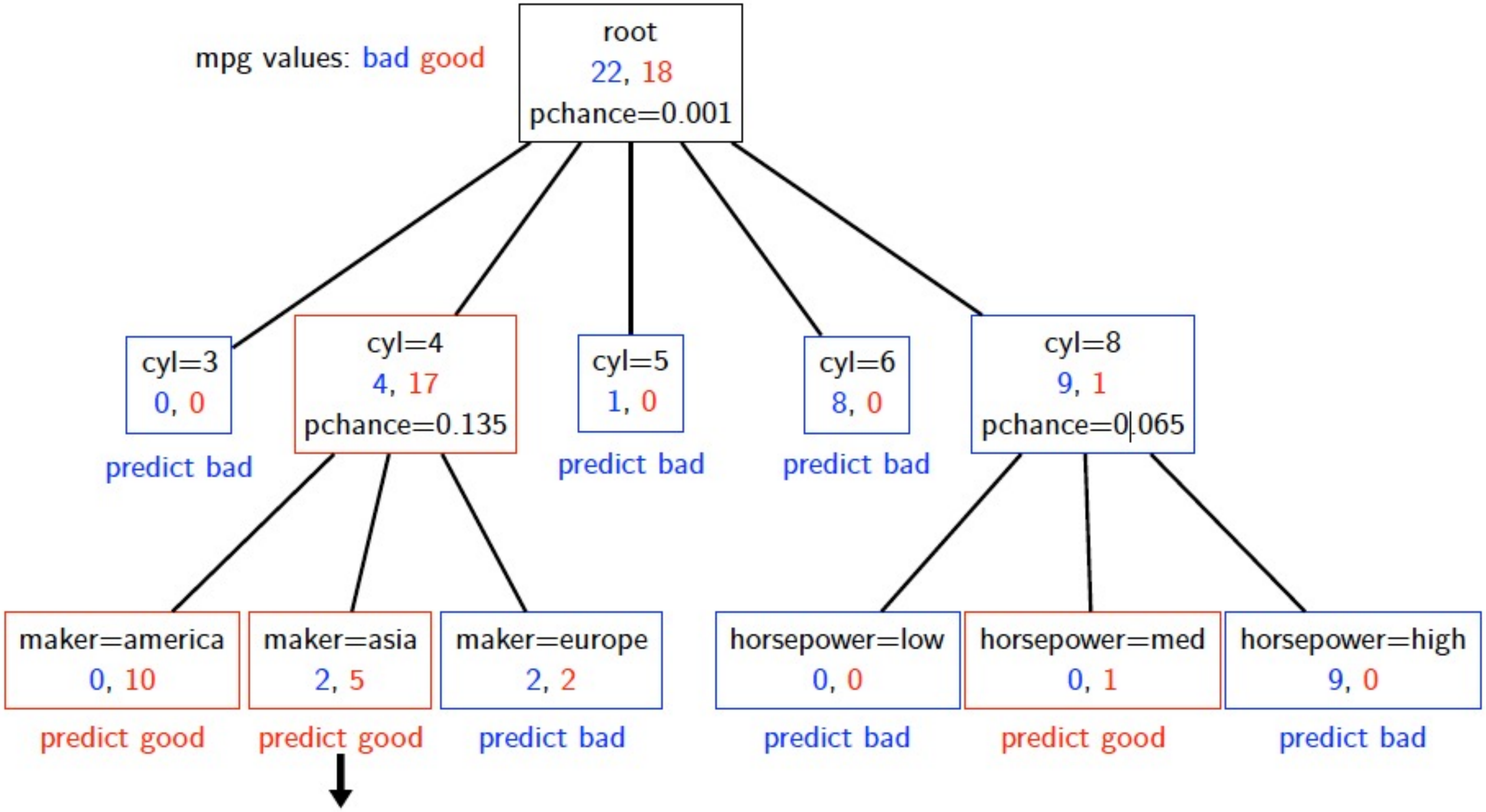
- Predict mpg=bad
- Is this a good tree? Total we get $(22+, 18\,-)$, which means we are correct on 22 bad and wrong on 18 good.

Decision Tree - Example



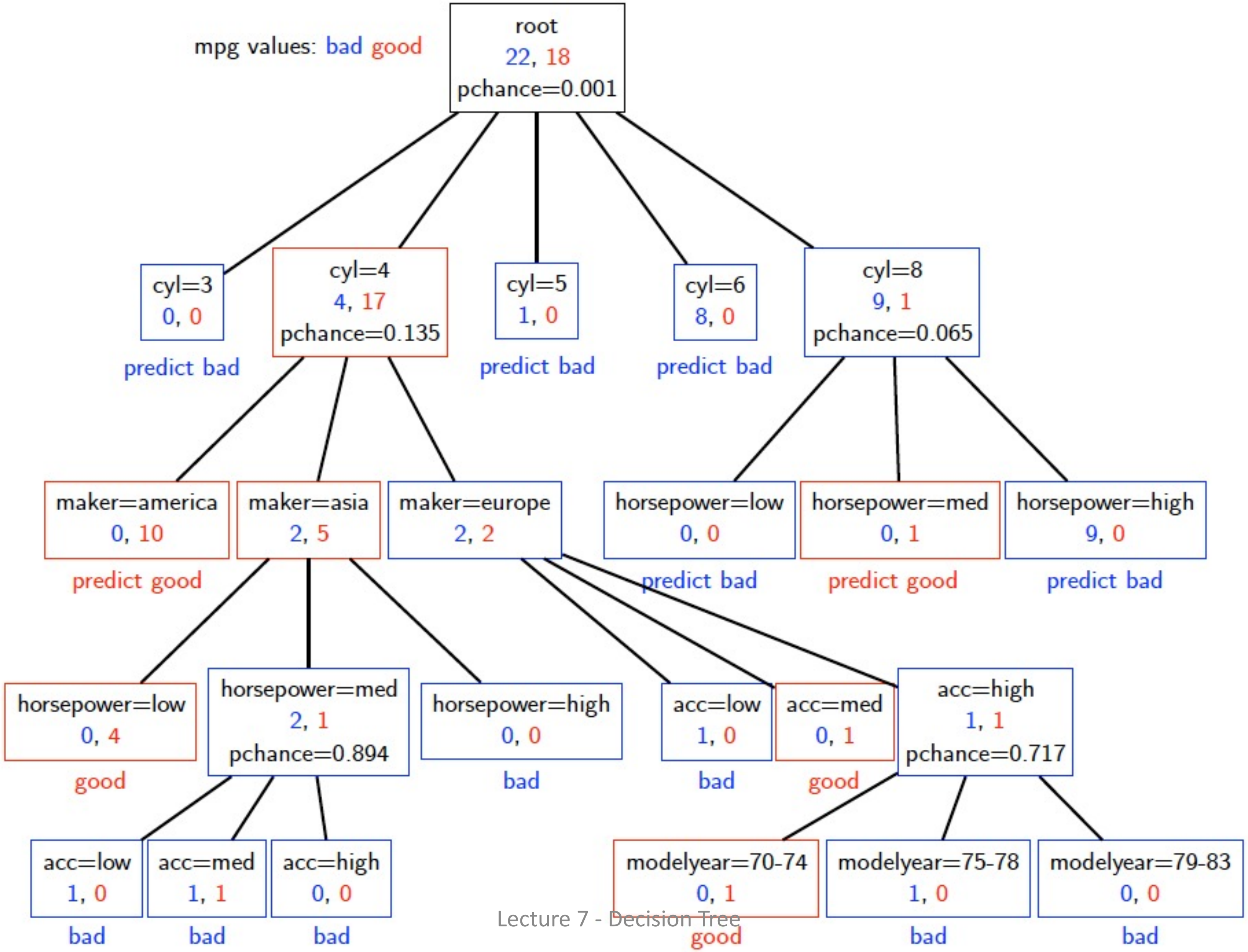
- Take the original dataset
- Partition it according to the values of the attribute we split on
- Build tree from these records (cyl=4, cyl=5, cyl=6, cyl=8)

Decision Tree - Example

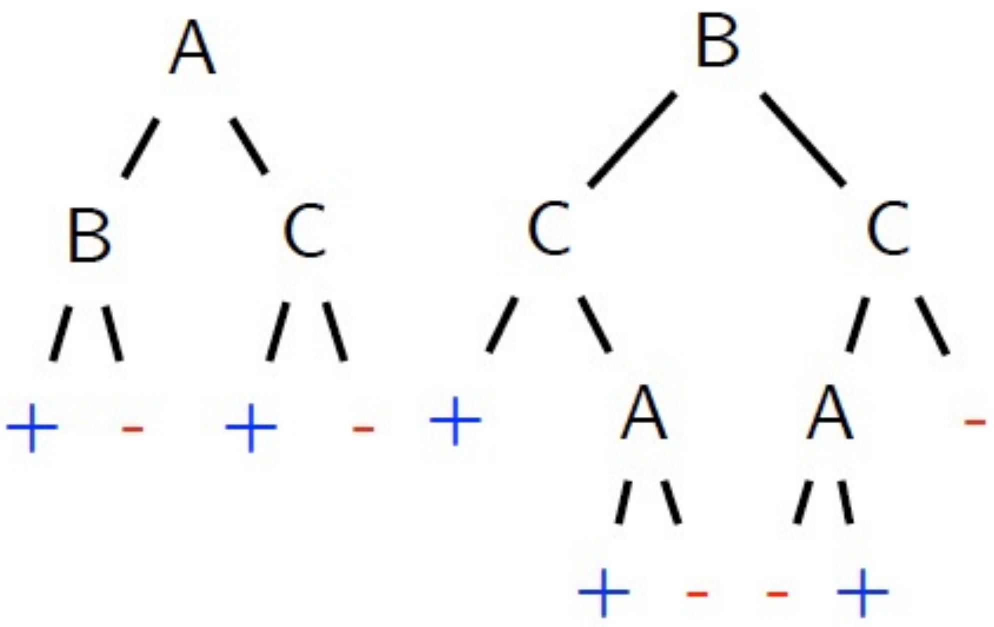


recursively build a tree from these records
in which cyl=4 and maker=Aisa

Decision Tree - Example



Decision Tree - Example



- Many trees can represent the same concept
- But, not all trees will have the same size!
- Which tree do we prefer?

- Learning the simplest (smallest) decision tree is an NP (nondeterministic polynomial time)-complete problem
- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on next best attribute (feature)
  - Recurse

➢ Idea: use counts as leaves to define **probability distribution**, so we can measure uncertainty.

Decision Tree - Entropy

- Entropy $H(Y)$ of a random variable Y:

$$H(Y) = -\sum_{i=1}^{K} P(Y = y_i) \log P(Y = y_i)$$

- More uncertainty, more entropy!
- Information theory interpretation: H(Y) is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)

High Entropy

- Y is from a uniform like distribution
- Flat histogram
- Values sampled from it are less predictable

Low Entropy

- Y is from a varied distribution(peaks and valleys)
- Histogram has many lows and highs
- Values sampled from it are more predictable

Decision Tree - Entropy

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$$H(Y) = -\sum_{i=1}^{K} P(Y = y_i) \log P(Y = y_i)$$

$$P(Y = T) = 5/6$$
$$P(Y = F) = 1/6$$
$$H(Y) = -\frac{5}{6}\log\left(\frac{5}{6}\right) - \frac{1}{6}\log\left(\frac{1}{6}\right) = 0.65$$

Decision Tree - Entropy

$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{K} P(Y = y_i \mid X = x_j) \log P(Y = y_i \mid X = x_j)$$

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$$P(X_1 = T) = 4/6$$
$$P(X_1 = F) = 2/6$$
$$H(Y \mid X_1) = -\frac{4}{6}(1 \log 1 + 0 \log 0)$$
$$-\frac{2}{6}\left[\frac{1}{2}\log\left(\frac{1}{2}\right) + \frac{1}{2}\log\left(\frac{1}{2}\right)\right] = \frac{2}{6}$$

Decision Tree – Information Gain

Decrease in entropy (uncertainty) after splitting:

$$IG(X) = H(Y) - H(Y \mid X)$$

$$IG = H(Y) - H(Y|X_1) = 0.65 - 0.33 = 0.32$$

We prefer the split $IG(X_1) > 0$.

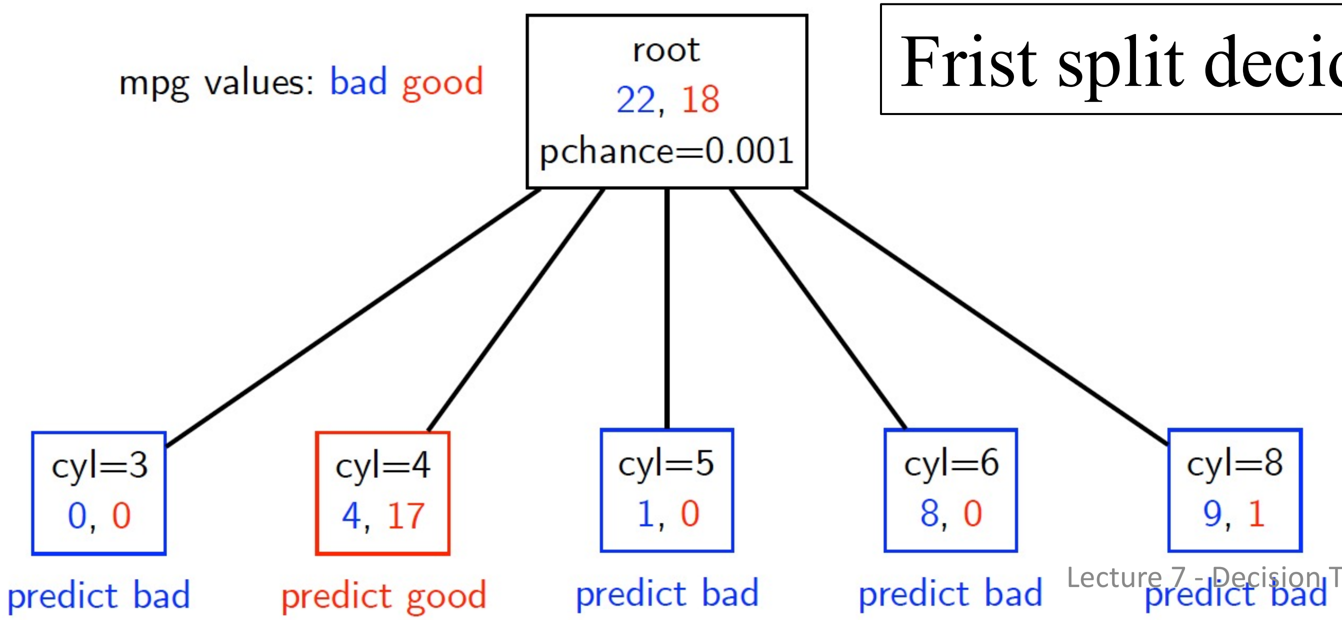
X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on next best attribute (feature) – **Use IG to select attribute**
  - Recurse

Decision Tree - Example

Table: Information gains using the training data set (40 records)

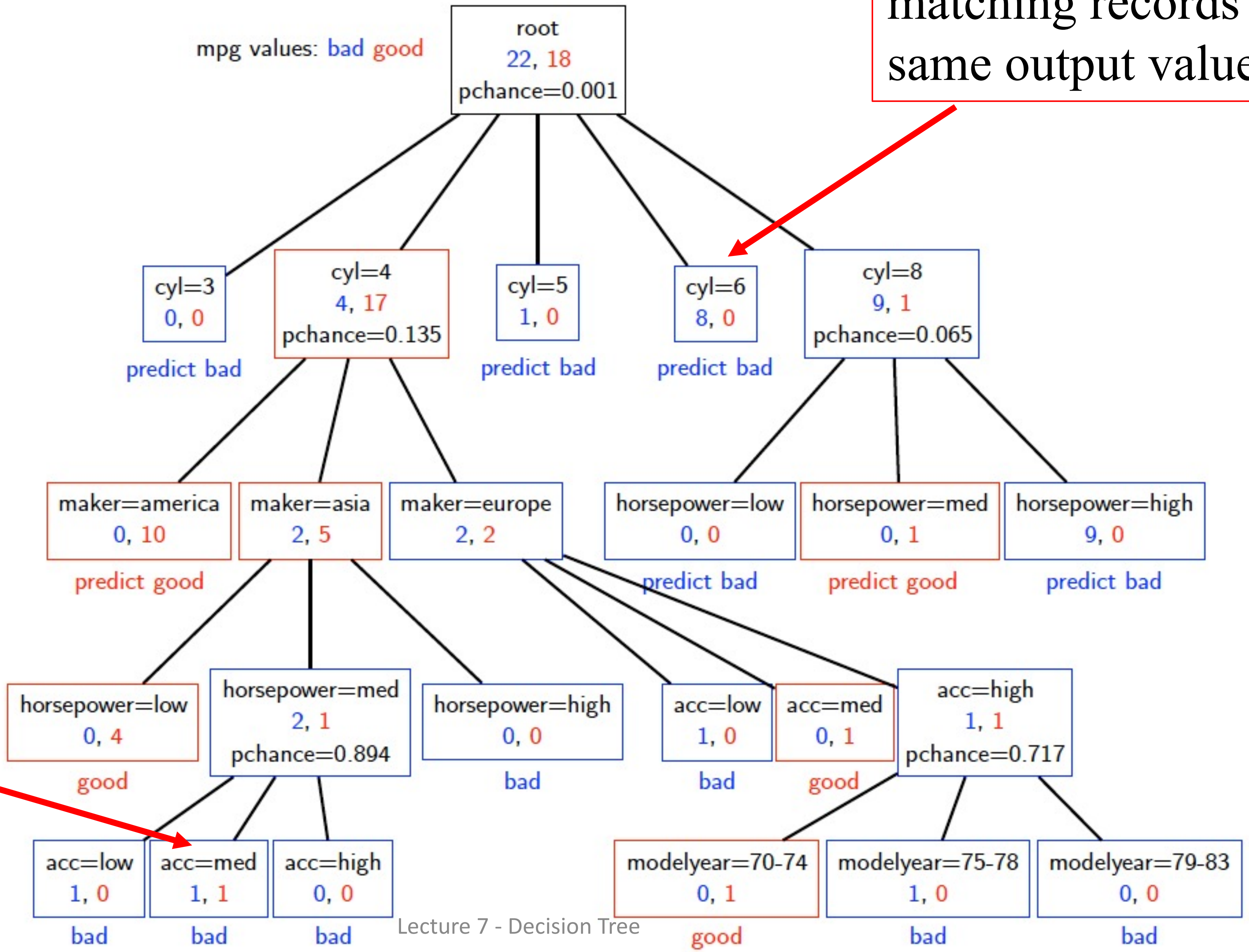
Input	value	Info Gain
cylinders	(3,4,5,6,8)	0.507
displacement	(low, medium, high)	0.223
horsepower	(low, medium, high)	0.388
weight	(low, medium, high)	0.304
acceleration	(low, medium, high)	0.064
modelyear	(70-74, 75-78, 79-83)	0.268



Frist split decided (cylinder), but when do we stop?

Decision Tree - Example

Don’t split a node if all matching records have the same output value.

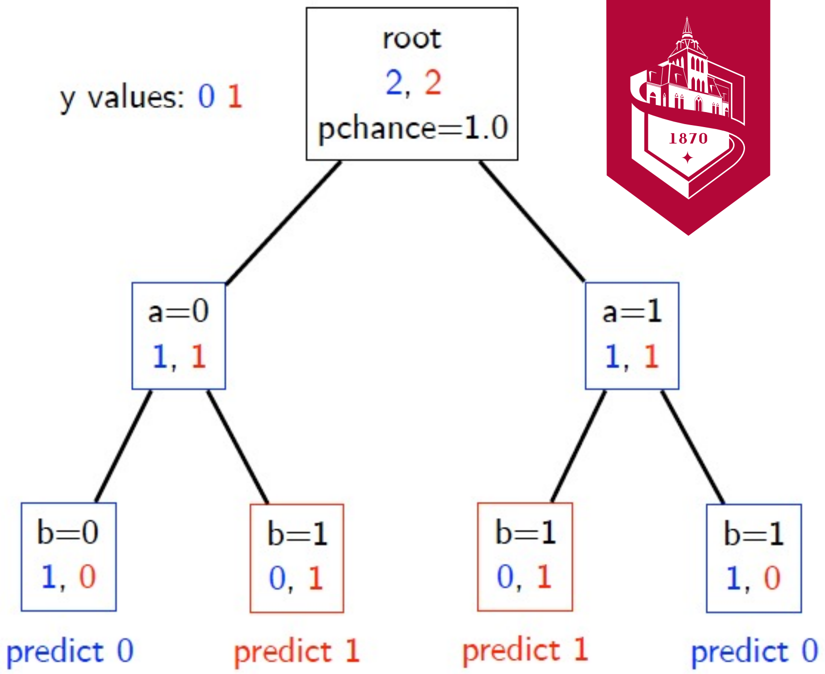


Don’t split a node if all data points are identical on remaining attributes

Decision Tree – Gini Index

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

$$y = a \text{ or } b$$
$$IG(a) = 0$$
$$IG(b) = 0$$



Gini Index used by the CART algorithm, the perfect score is 0:

$$IG(p) = 1 - \sum_{i=1}^{K} p_j^2$$

- Let p_j be the fraction of items labeled with class j in the set.
- A perfect separation results in a Gini score of 0.

Decision Tree – Overfit

Standard decision trees have no learning bias.

- Training set error is almost zero
- Lots of variance
- Must introduce some bias towards simpler trees

Many strategies for picking simpler trees

- Fixed depth
- Fixed number of leaves
- Or something smarter

Decision Tree – How to build small trees

Two reasonable approaches

- Optimize on the held-out (validation) set
  - If growing the tree larger hurts performance, then stop growing
  - Requires a large amount of data
- Use statistical significance testing
  - Test if the improvement for any split is likely due to noise. If so, don't do the split
  - Prune the tree bottom-up. A chi-squared test can tell us how likely it is that deviations from a perfect split are due to chance. Delete splits in which $p_{chance} >$ $\mathrm{max}_{\mathrm{chance}}$

Decision Tree – Real-valued inputs

What should we do if some of the inputs are real-valued?

- Infinite number of possible split values
- Finite dataset, only finite number of relevant splits.

Proposed solution:

- One branch for each numeric value?
- Hopeless: hypothesis with such a high branching factor will shatter any dataset and overt.

Decision Tree – Threshold Splits

Binary tree: split on attribute X at value t:

- One branch: $X < t$
- Other branch: $X \geq t$

Requires small change:

- Allow repeated splits on the same variable
- How does this compare to “branch on each value” approach?

Search through possible value of t (seems hard!!!)

But only a finite number of t’s are important:

- Sort data according to X into $\{x_1, \ldots, x_m\}$
- Consider split points of the form $x_i + \frac{x_{i+1} - x_i}{2}$
- Moreover, only splits between example of different classes matter.

Decision Tree – Picking the best threshold

Suppose X is real valued with threshold *t:*

Want $IG(Y \mid X{:}\,t)$, the information gain for Y when testing if X is greater than or less than *t*

Define

- $H(Y|X{:}\,t) = P(X < t)H(Y|X < t) + P(X \geq t)H(Y \mid X \geq t)$
- $IG(Y|X{:}\,t) = H(Y) - H(Y \mid X{:}\,t)$
- $IG^*(Y|X) = \max_t IG(Y \mid X{:}\,t)$

Use $IG^*(Y \mid X)$ for continuous variables.

Decision Tree vs. Linear Regression

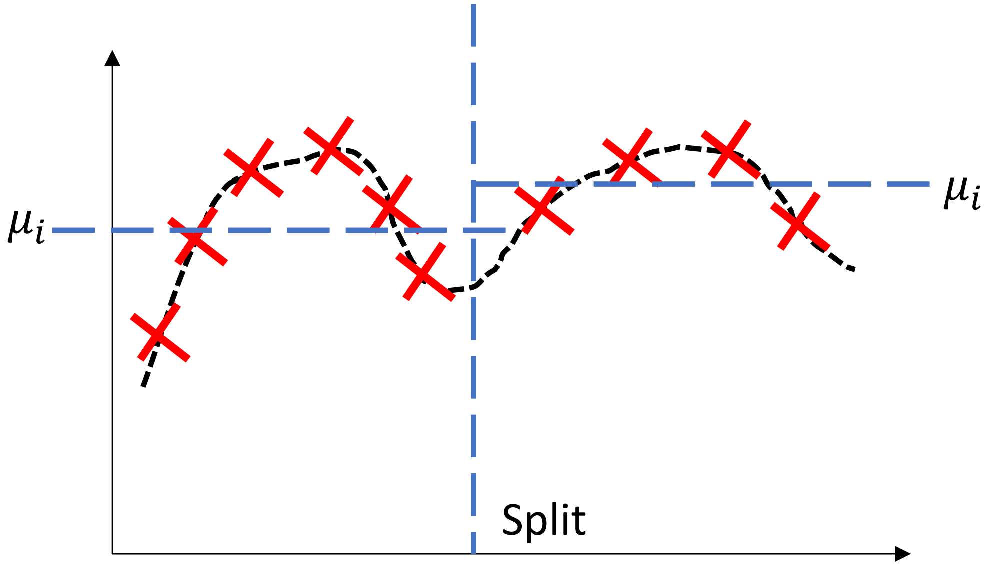
Decision trees

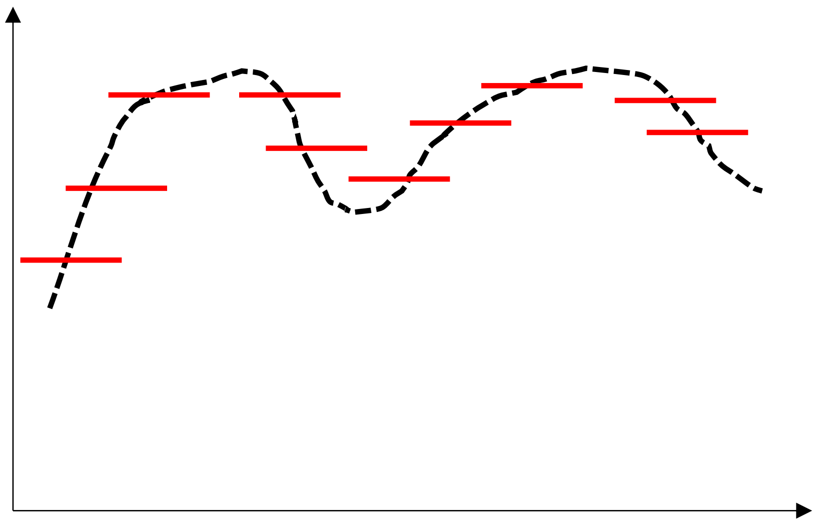
- Can solve both classification and regression problem
- If the relationship is non-linear it will outperform Linear regression
- Can build models that are easy to explain

Linear regression

- In a linear relationship, linear regression will likely outperform Decision trees.
- Cannot easily handle categorial variables.

Decision Trees - Regressions





Loss Function: $l(S) = \frac{1}{|S|}\sum_{(x,y)\in S}(y-\mu)^2$ where $\mu = \frac{1}{|S|}\sum y$

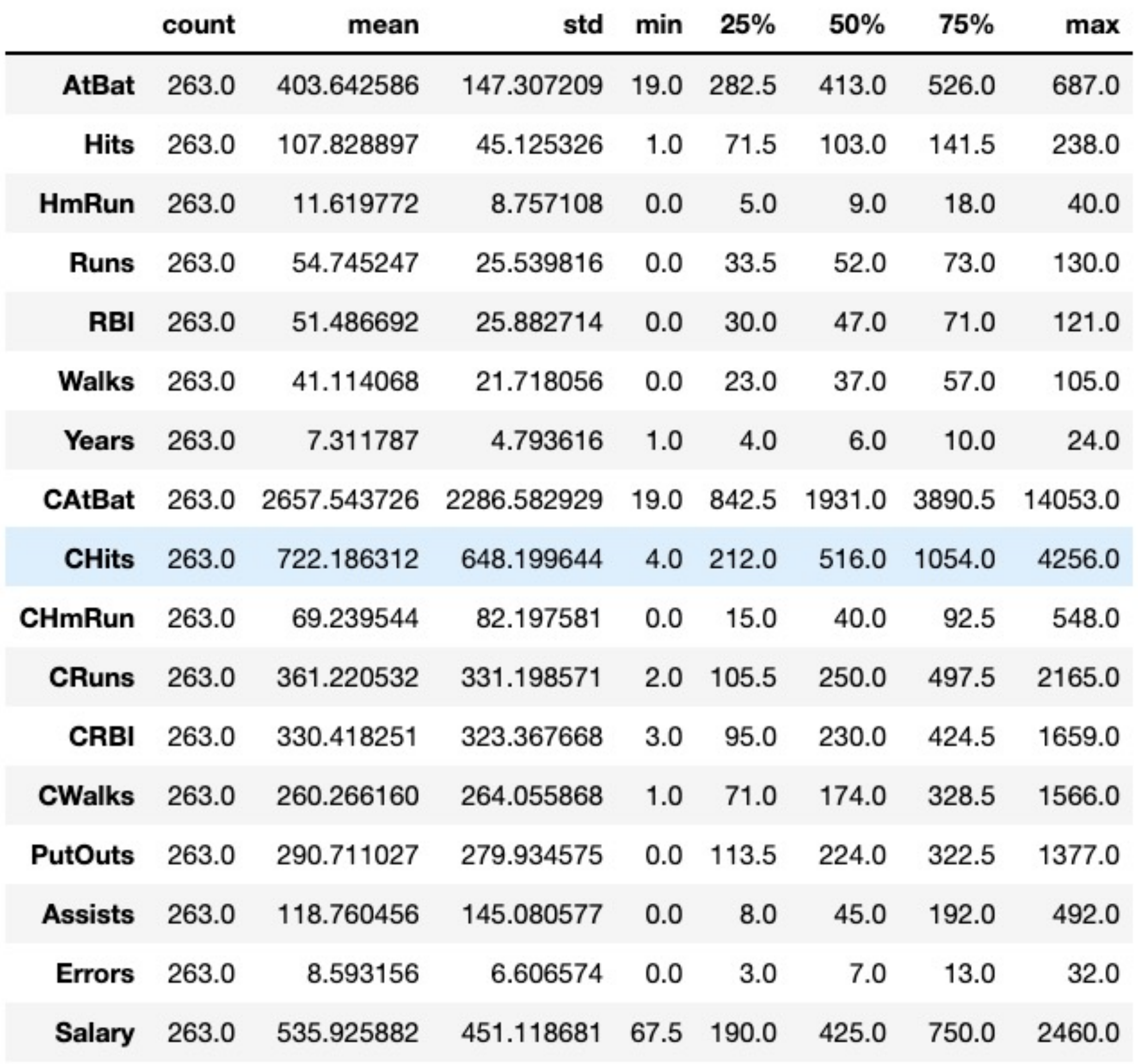
Goal: how to get to close to μ

But: becomes variance $(y-\mu)^2$ problem. Balancing the bias-variance tradeoff is the key!

- Limiting the depth of tree: Bias vs. Variance
- # of leaves: each leaf predicts one value - smoothness

We can use any loss functions

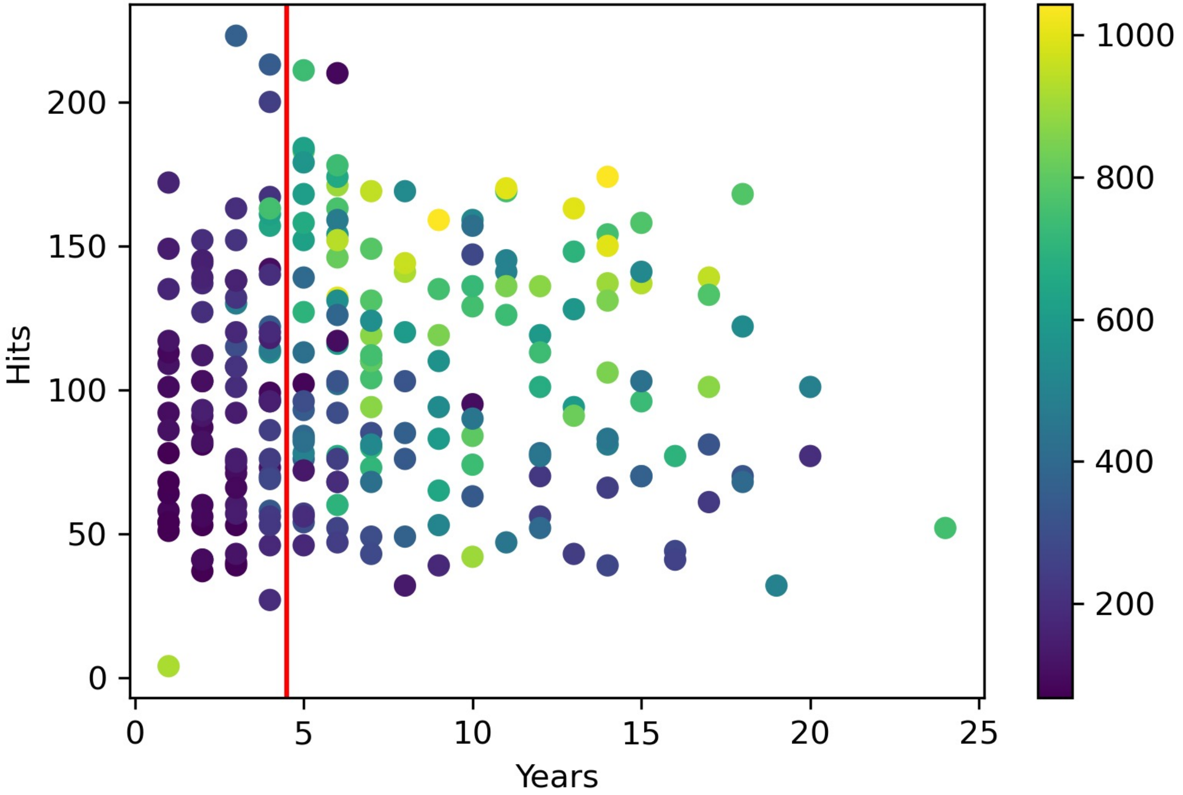
Decision Trees - Regressions

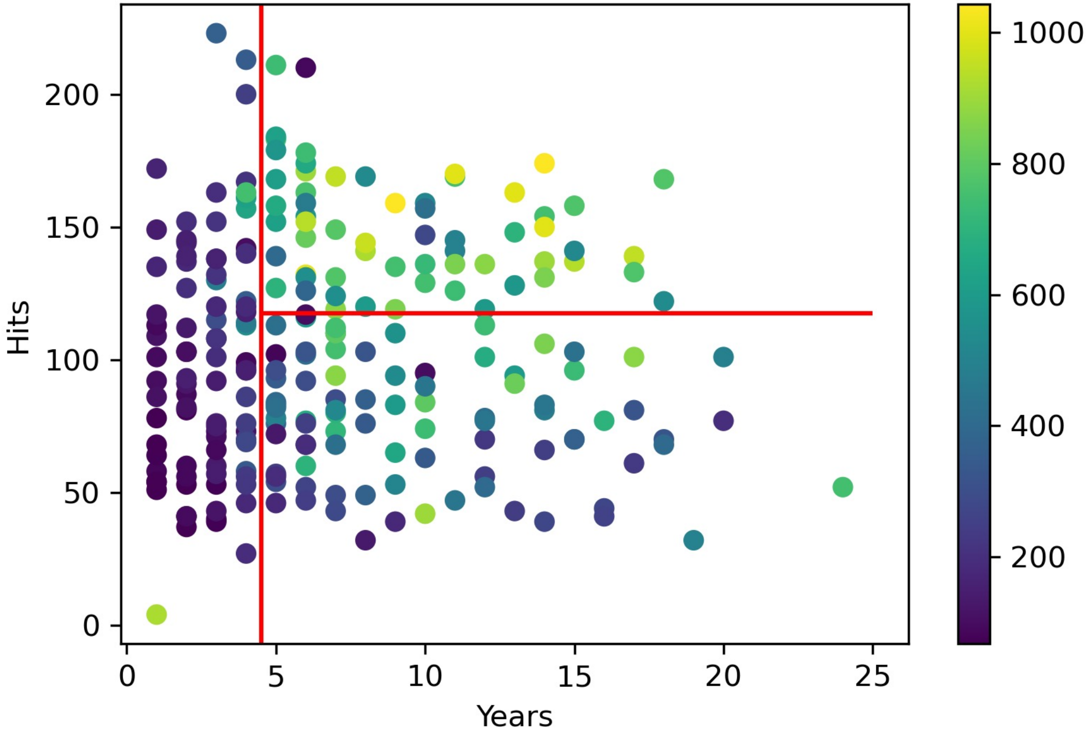


	count	mean	std	min	25%	50%	75%	max
AtBat	263.0	403.642586	147.307209	19.0	282.5	413.0	526.0	687.0
Hits	263.0	107.828897	45.125326	1.0	71.5	103.0	141.5	238.0
HmRun	263.0	11.619772	8.757108	0.0	5.0	9.0	18.0	40.0
Runs	263.0	54.745247	25.539816	0.0	33.5	52.0	73.0	130.0
RBI	263.0	51.486692	25.882714	0.0	30.0	47.0	71.0	121.0
Walks	263.0	41.114068	21.718056	0.0	23.0	37.0	57.0	105.0
Years	263.0	7.311787	4.793616	1.0	4.0	6.0	10.0	24.0
CAtBat	263.0	2657.543726	2286.582929	19.0	842.5	1931.0	3890.5	14053.0
CHits	263.0	722.186312	648.199644	4.0	212.0	516.0	1054.0	4256.0
CHmRun	263.0	69.239544	82.197581	0.0	15.0	40.0	92.5	548.0
CRuns	263.0	361.220532	331.198571	2.0	105.5	250.0	497.5	2165.0
CRBI	263.0	330.418251	323.367668	3.0	95.0	230.0	424.5	1659.0
CWalks	263.0	260.266160	264.055868	1.0	71.0	174.0	328.5	1566.0
PutOuts	263.0	290.711027	279.934575	0.0	113.5	224.0	322.5	1377.0
Assists	263.0	118.760456	145.080577	0.0	8.0	45.0	192.0	492.0
Errors	263.0	8.593156	6.606574	0.0	3.0	7.0	13.0	32.0
Salary	263.0	535.925882	451.118681	67.5	190.0	425.0	750.0	2460.0

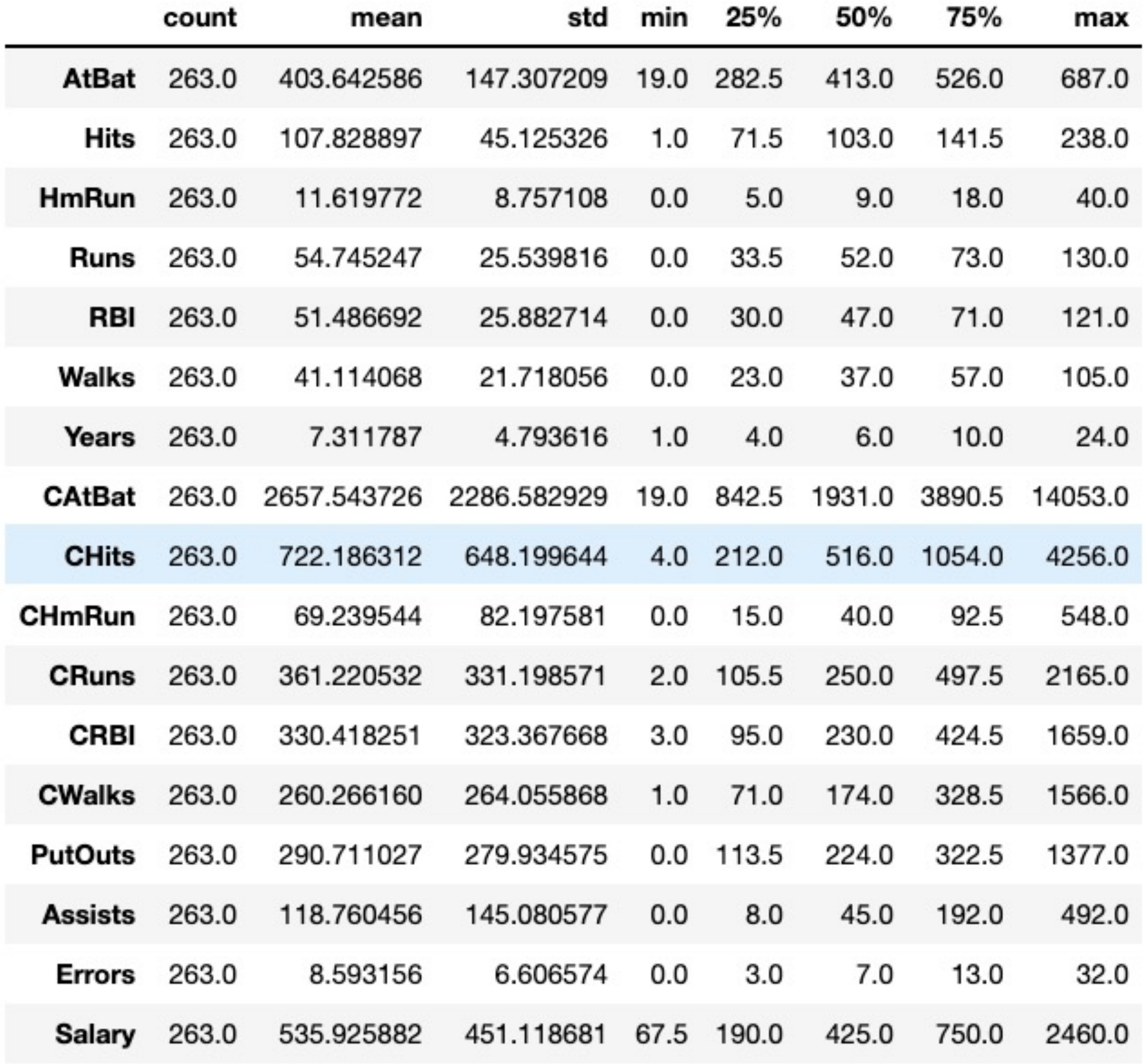
We are going to predict baseball players’ salaries

Decision Trees - Regressions

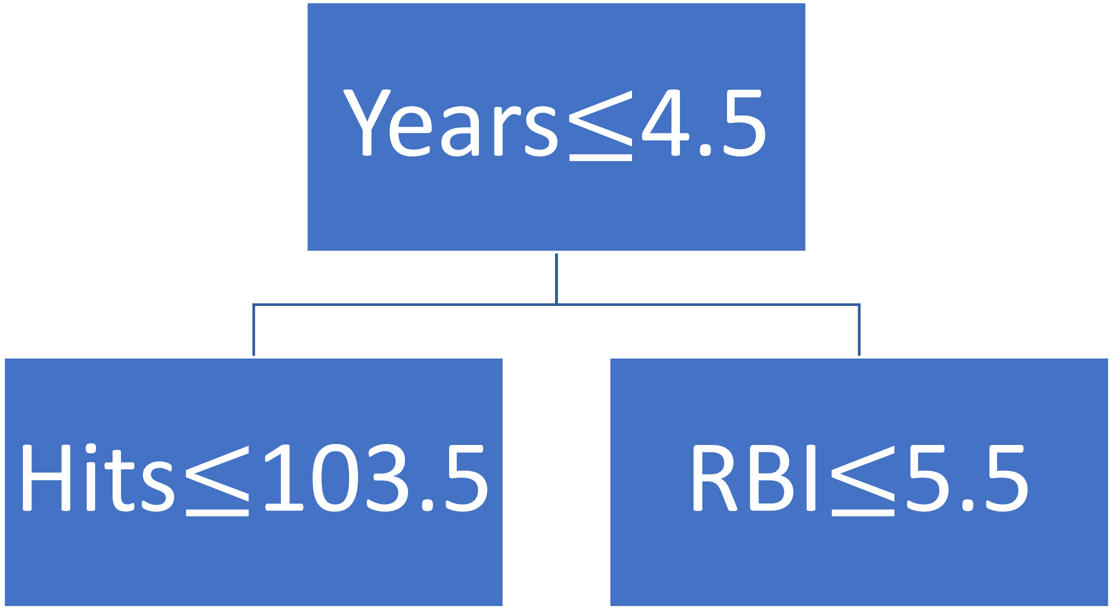




Decision Trees - Regressions



	count	mean	std	min	25%	50%	75%	max
AtBat	263.0	403.642586	147.307209	19.0	282.5	413.0	526.0	687.0
Hits	263.0	107.828897	45.125326	1.0	71.5	103.0	141.5	238.0
HmRun	263.0	11.619772	8.757108	0.0	5.0	9.0	18.0	40.0
Runs	263.0	54.745247	25.539816	0.0	33.5	52.0	73.0	130.0
RBI	263.0	51.486692	25.882714	0.0	30.0	47.0	71.0	121.0
Walks	263.0	41.114068	21.718056	0.0	23.0	37.0	57.0	105.0
Years	263.0	7.311787	4.793616	1.0	4.0	6.0	10.0	24.0
CAtBat	263.0	2657.543726	2286.582929	19.0	842.5	1931.0	3890.5	14053.0
CHits	263.0	722.186312	648.199644	4.0	212.0	516.0	1054.0	4256.0
CHmRun	263.0	69.239544	82.197581	0.0	15.0	40.0	92.5	548.0
CRuns	263.0	361.220532	331.198571	2.0	105.5	250.0	497.5	2165.0
CRBI	263.0	330.418251	323.367668	3.0	95.0	230.0	424.5	1659.0
CWalks	263.0	260.266160	264.055868	1.0	71.0	174.0	328.5	1566.0
PutOuts	263.0	290.711027	279.934575	0.0	113.5	224.0	322.5	1377.0
Assists	263.0	118.760456	145.080577	0.0	8.0	45.0	192.0	492.0
Errors	263.0	8.593156	6.606574	0.0	3.0	7.0	13.0	32.0
Salary	263.0	535.925882	451.118681	67.5	190.0	425.0	750.0	2460.0

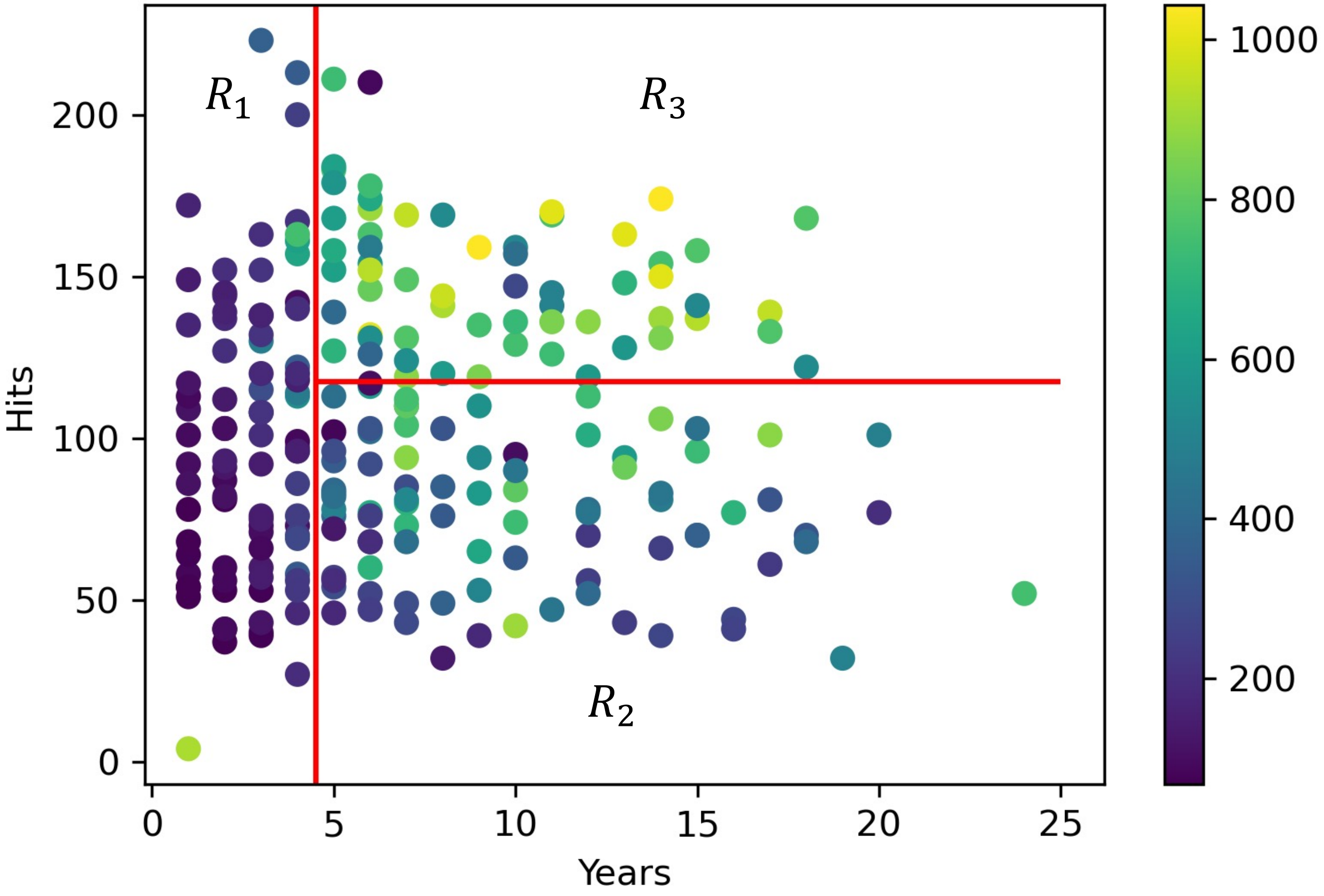


Years≤4.5

Hits≤103.5

RBI≤5.5

Decision Trees - Regressions



The decision tree ends up partitioning the feature space into three regions:
R_1: players have less than 4.5 years of experience.
R_2: players have greater than 4.5 years of experience and had fewer than 103.5 hits.
R_3: players have greater than 4.5 years of experience and had greater than 103.5 hits.

Decision Trees - Regressions

The process of building a regression tree can be simplified into the following two steps:

1. Segment the predictor space (all possible values of $x_1, x_2, x_3, \ldots, x_p$) into J distinct and non-overlapping regions $(R_1, R_2, \ldots, R_J)$.
2. For every observation that falls into a specific region, R_j, predict the mean of the responsive values for the training observations that fell within R_j.

Questions:

- How do we decide where exactly to segment the predictor space?
- How do we come upon the regions $R_1, R_2, \ldots, R_J$?

Decision Trees - Regressions

Theoretically, it is possible that regions in the feature space could have any shape; however, if the region splits were to not follow some specific pattern, it would be difficult to represent the resulting model by a decision tree.

- For ease of interpretation, rectangular/box-like segments will allow us to construct a decision tree for our predictive model.

The goal now becomes much simpler – find rectangular boxes $R_1, R_2, \ldots, R_J$ such that the residual sum-square, RSS, is minimized, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2$$

- We aim to minimize the squared differences of the response as compared to the mean response for the training observations within the j^{th} region.

Decision Trees - Regressions

While the goal itself seems straightforward, it is computationally infeasible to consider every possible segmentation of the feature space into R_j regions.

- The minimization problem isn't as easily solvable as we have seen with other machine learning methods, especially as the number of regions increases.

Tree-based methods provide an approximation by implementing both a top-down and a greedy approach called recursive binary splitting:

- The method is top-down because the feature space is split into binary components in a successive fashion, creating new branches of the tree to potentially be split themselves.
- The method is greedy because splits are made at each step of the process based on the best result possible at that given step.
    - The splits are not made based on what might eventually lead to a better segmentation in future steps.

Decision Trees - Regressions

The recursive binary splitting process depends on the greatest reduction in RSS based on the predictor x_j and the cutting-point s that end up partitioning the space into the regions:

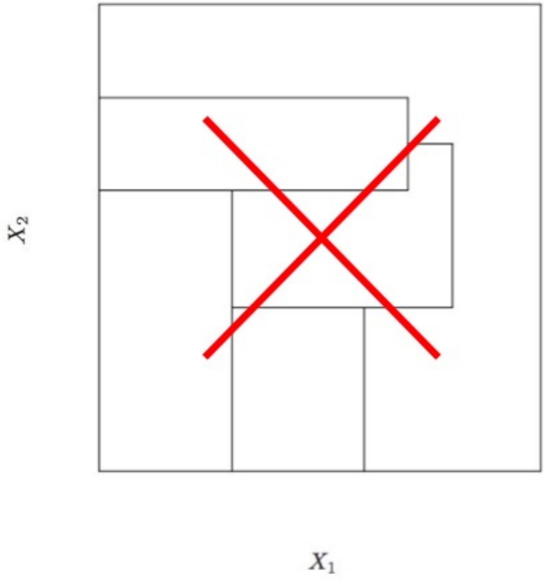
- $R_1(j,s) = \{x | x_j < s\}$
- $R_2(j,s) = \{x | x_j \geq s\}$

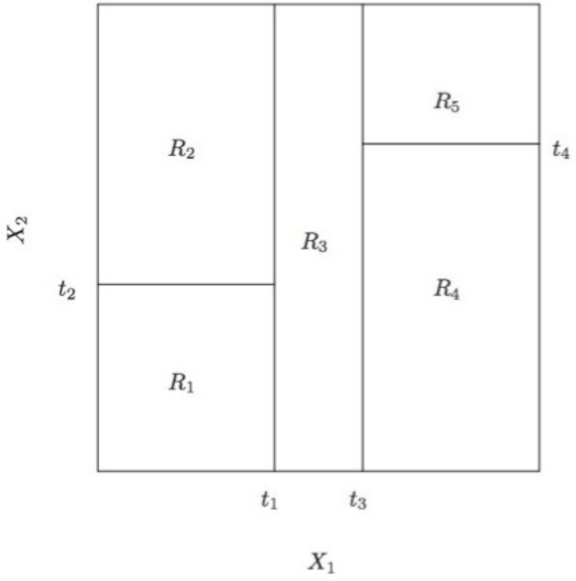
In other words, the splitting process seeks the values of j & s that will end up minimizing the following:

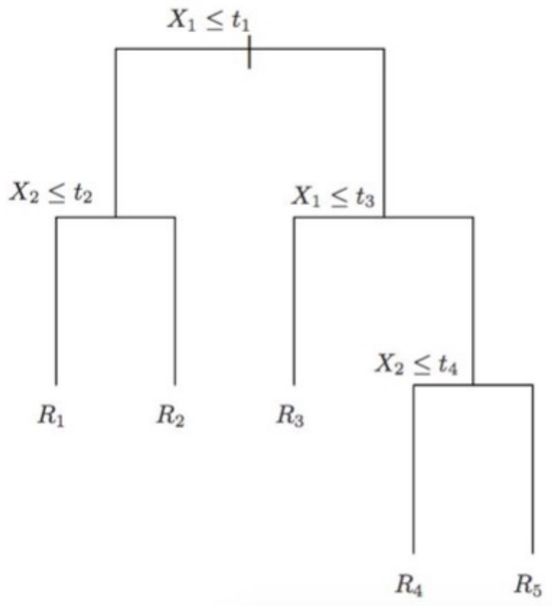
$$\sum_{i:x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{i:x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2$$

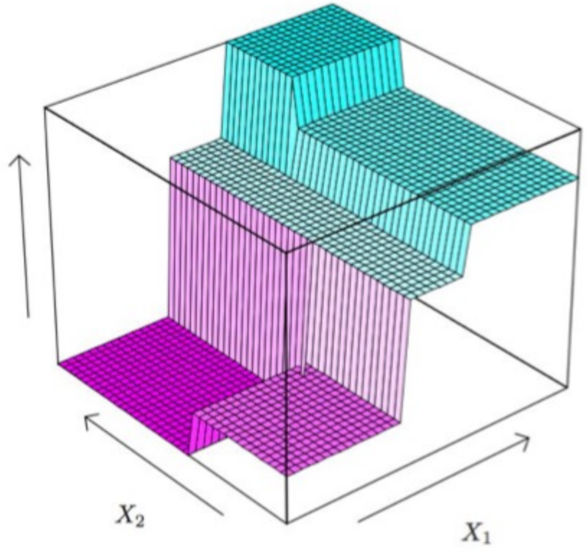
This process is repeated by considering each of the newly created regions as the new overall feature spaces to segment.

Decision Trees - Regressions









Le

Decision Trees - Regressions

The recursive binary splitting process as described above is likely to induce overfitting, thus leading to poor predictive performance on new observations.

- Consider the extreme case in which each observation has its own terminal node (this model will have high variance). The RSS will be exactly 0 on the training set.

What if we try fitting a tree with fewer regions? Theoretically, this should lead to lower variance with a cost of some bias, but ultimately lead to better prediction.

- Grow the tree to a certain extent until the reduction in the RSS at a split doesn’t surpass a certain threshold.
- Problem: Although a split might not be incredibly valuable in reducing the RSS early on in a tree, it might lead to a future split that does reduce the RSS to a large extent. What is one possible solution?

Decision Trees - Regressions

We've seen that deciding on the number of splits prior to building a tree isn't the best strategy.

- What if we first built a tree that is very large and then pruned it back in order to obtain a suitable subtree?

The best subtree will be the one that yields the lowest test error rate.

- Given a subtree, we can estimate the test error by implementing the cross-validation process, but this is too cumbersome because the large number of possible subtrees. We still need a better process!

In practice, rather than checking every single possible subtree, the process of cost complexity pruning (i.e., weakest link pruning) allows us to select a smaller set of subtrees for consideration.

Decision Trees - Regressions

Consider a sequence of trees indexed by a non-negative tuning parameter α. For each value of α there corresponds a subtree T such that the following is criterion minimized:

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

where

- $|T|$ indicates the total number of terminal nodes of subtree T.
- R_m is the subset region of the feature space corresponding to the m^{th} terminal node.

The tuning parameter α helps balance the tradeoff between the overall complexity of the tree and its fit to the training data:

- Small values of α yield trees that are quite extensive (have many terminal nodes).
- Large values of α yield trees that are quite limited (have few terminal nodes).

This process is very similar to the shrinkage/regularization method utilized in ridge and lasso regression!

Decision Trees - Regressions

It can be shown that as the value of the tuning parameter α increases, branches of the overall tree are pruned in a nested manner.

- Thus, it is possible to obtain a sequence of subtrees as a function of α.

As with any other tuning parameter, in order to select the optimal value of α we implement cross-validation.

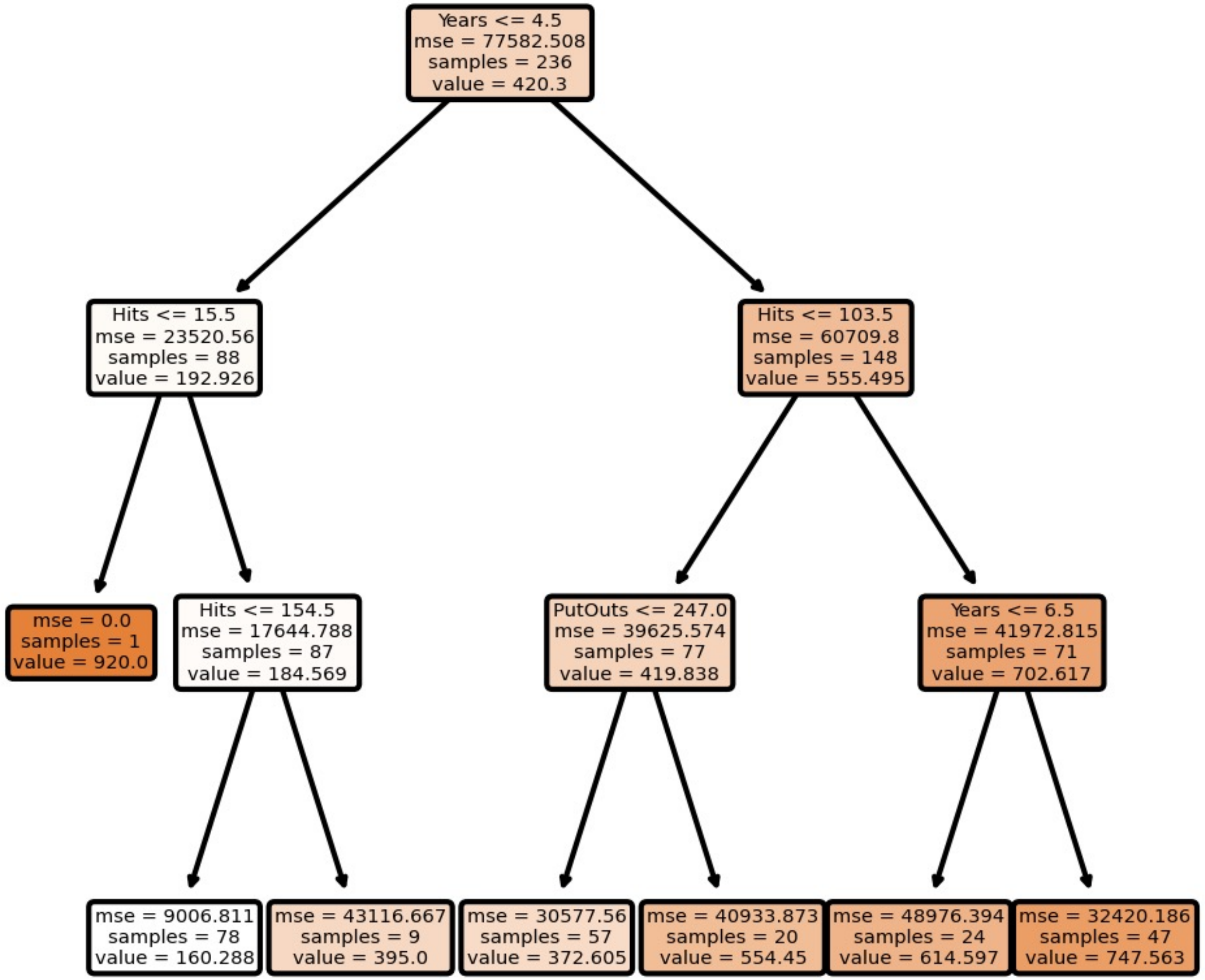
Ultimately, the subtree that is used for prediction is built using all of the available data with the determined optimal value of α.

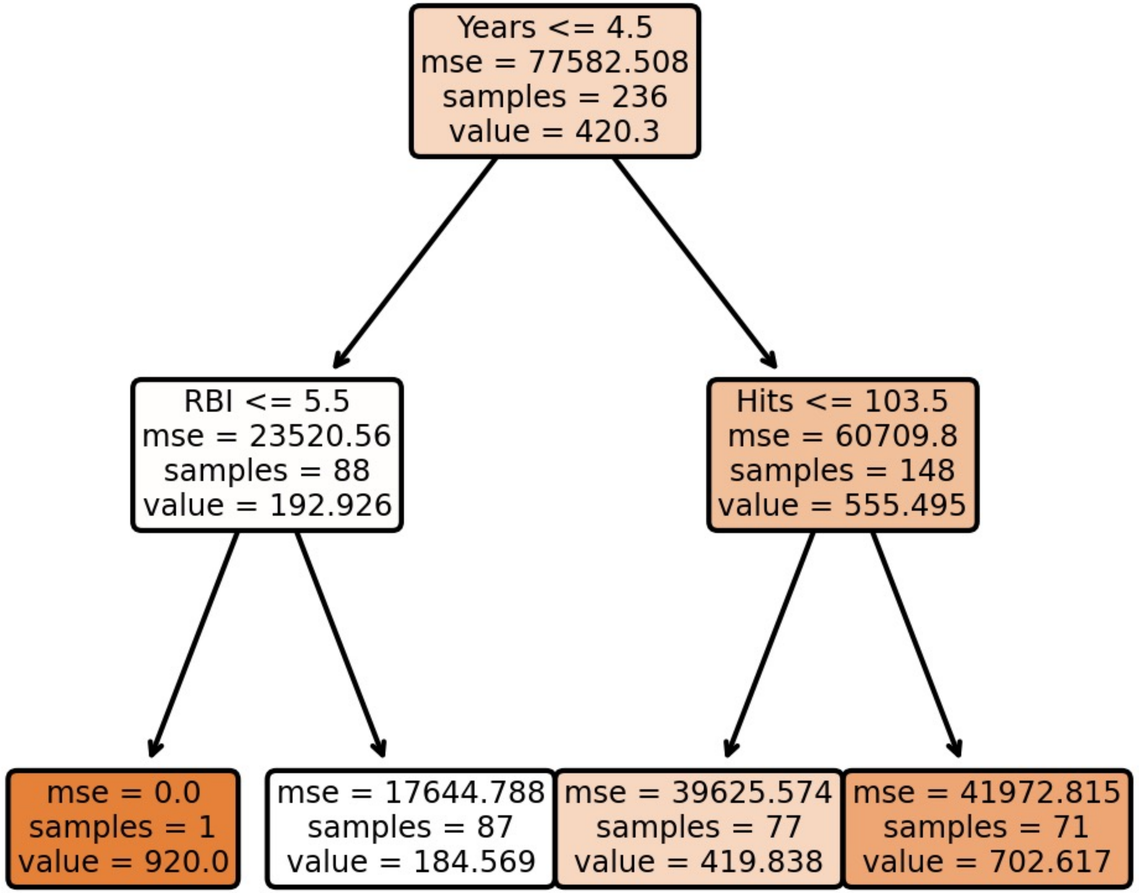
Decision Trees - Regressions

Incorporating these ideas, the algorithm for building a regression tree is as follows:

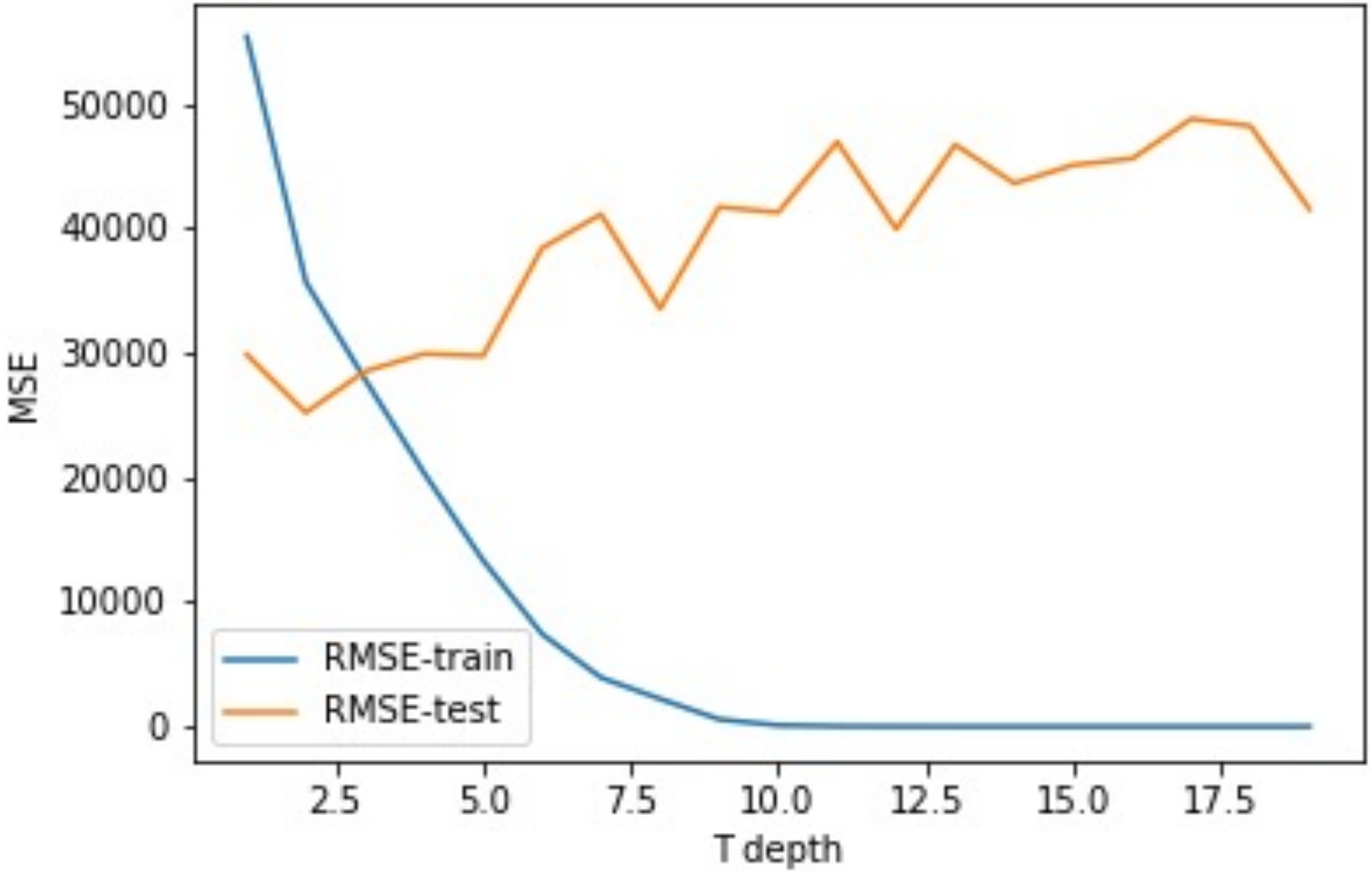
1. Use recursive binary splitting to build a large tree on the training data; stop before each observation falls into its own leaf (e.g., when each terminal node has fewer than 5 observations, etc.).
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees as a function of α.
3. Use K-fold cross-validation to choose the best α.
    1. For each of the K folds:
        1. Repeat steps 1 and 2 on all but the K^{th} fold of the training data.
        2. Evaluate the mean squared prediction error on the data in the
        3. left-out K^{th} fold as a function of α .
    2. Average the errors for each α; select the α that minimizes this criterion.
4. Return the subtree of the overall tree from step 2 that corresponds to the best α.

Decision Trees - Regressions





Decision Trees - Regressions



Decision Tree – Summary

- Without a heavy mathematical background, it is easy to interpret a decision tree (especially if it is small); linear regression requires the understanding of an equation.
- Decision trees can graphically depict a higher dimensionality easier than linear regression and still be interpreted by a novice.
- The process can easily adapt to qualitative predictors without the need to create and interpret dummy variables.
- Decision trees are often believed to reflect a more “human” decision-making process as compared to other machine learning methods.
- While relatively non-complex among other supervised learning procedures, as a trade-off their predictive accuracy tends to be lower and thus not as competitive.
- What if we could combine the benefits of multiple trees in order to yield an overall prediction? Taking the penalty of decreased interpretative value, could this potentially increase our predictive accuracy?
    - ➢ Bagging
    - ➢ Random Forests
    - ➢ Boosting