S&DS 355 / 555

Introductory Machine Learning

(Decision) Trees

Tuesday, September 24th

Prof. Elisa Celis

Yale

Updates and Announcements

Updates and Announcements

Hello!

- My office hours for this week's material will be on Monday, Sept 30 from 1:30-3:30pm.

Updates and Announcements

Hello!

- My office hours for this week's material will be on Monday, Sept 30 from 1:30-3:30pm.
- Assignment 2 is due on Thursday, Sept 26.

Updates and Announcements

Hello!

- My office hours for this week's material will be on Monday, Sept 30 from 1:30-3:30pm.
- Assignment 2 is due on Thursday, Sept 26.
- Quiz 1 will be given during the first 10 minutes of class on Oct 1 (includes this week's material!).

Classification and Regression Trees (CART)

Trees provide alternative ways of modeling nonlinear relationships, and give a **nonparametric** approach that does not require any assumptions about the underlying data.

Classification and Regression Trees (CART)

Trees provide alternative ways of modeling nonlinear relationships, and give a **nonparametric** approach that does not require any assumptions about the underlying data.

- Can be used for either classification or regression.

Classification and Regression Trees (CART)

Trees provide alternative ways of modeling nonlinear relationships, and give a **nonparametric** approach that does not require any assumptions about the underlying data.

- Can be used for either classification or regression.
- Feature variables can be categorical or quantitative.

Classification and Regression Trees (CART)

Trees provide alternative ways of modeling nonlinear relationships, and give a **nonparametric** approach that does not require any assumptions about the underlying data.

- Can be used for either classification or regression.
- Feature variables can be categorical or quantitative.
- Yields a set of interpretable decision rules (popular in medicine).

Classification and Regression Trees (CART)

Trees provide alternative ways of modeling nonlinear relationships, and give a **nonparametric** approach that does not require any assumptions about the underlying data.

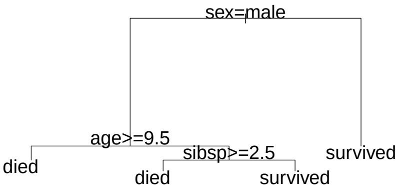
- Can be used for either classification or regression.
- Feature variables can be categorical or quantitative.
- Yields a set of interpretable decision rules (popular in medicine).
- Predictive ability is often mediocre, *but* can be improved with ideas of resampling (will be covered on Thursday).

Trees

Trees

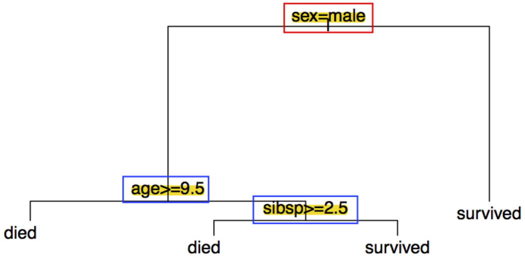
Trees

Modeling Titanic survival (classification tree):



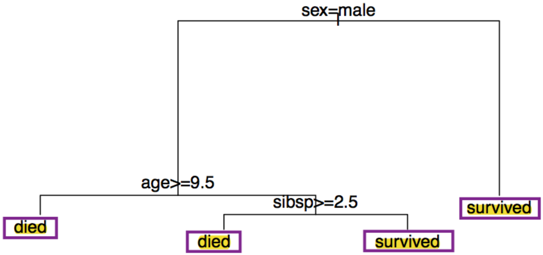
Tree terminology

Internal nodes are points where the predictor space is split.



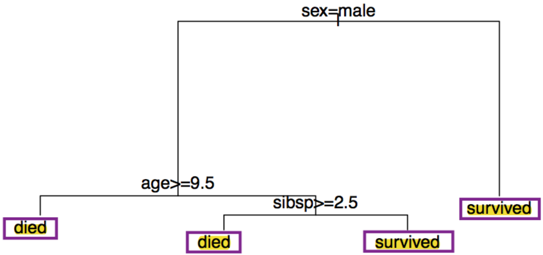
Tree terminology

Terminal nodes (or **leaves**) are the ends of the tree where no further splitting occurs.



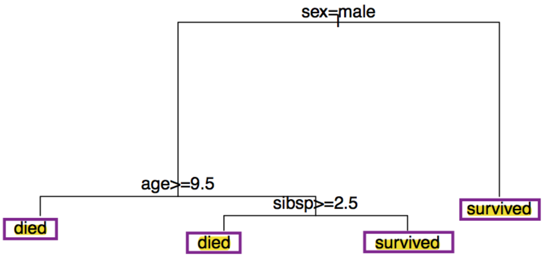
Tree terminology

Terminal nodes (or **leaves**) are the ends of the tree where no further splitting occurs.



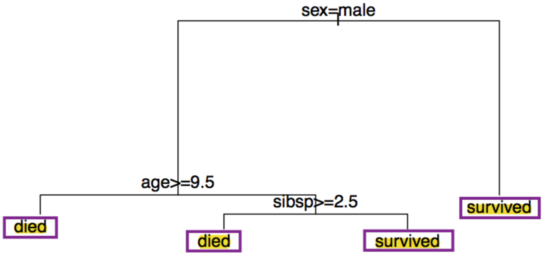
Denote these J regions as $R_1, \ldots, R_J$.

Tree terminology



- $R_1 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i \geq 9.5\}$
- $R_2 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i \geq 2.5\}$
- $R_3 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i < 2.5\}$
- $R_4 = \{i : \text{sex}_i \neq \text{male}\}$

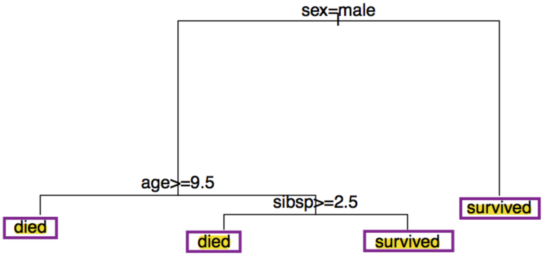
Tree terminology



- $R_1 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i \geq 9.5\}$
- $R_2 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i \geq 2.5\}$
- $R_3 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i < 2.5\}$
- $R_4 = \{i : \text{sex}_i \neq \text{male}\}$

Regions are disjoint and cover the whole space.

Tree terminology

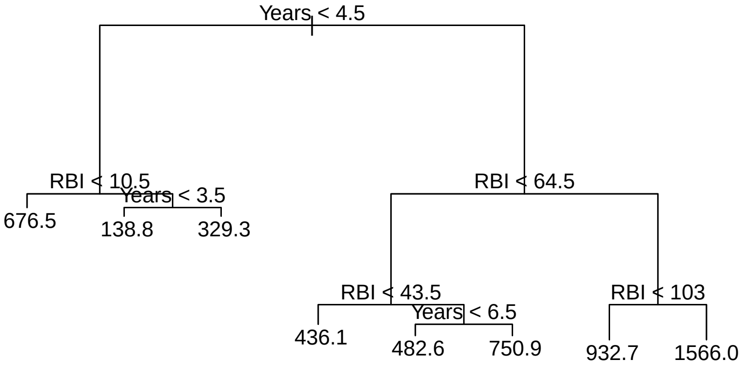


- $R_1 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i \geq 9.5\}$
- $R_2 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i \geq 2.5\}$
- $R_3 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i < 2.5\}$
- $R_4 = \{i : \text{sex}_i \neq \text{male}\}$

This is a classification tree – each region corresponds to a decision.

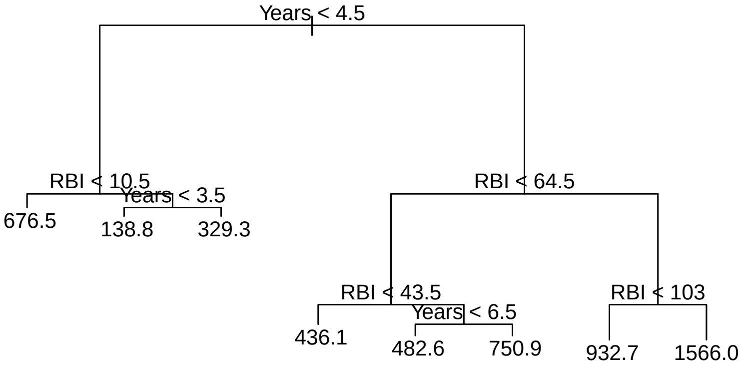
Regression tree example

Baseball hitter salaries (1987) (in $1,000s):



Regression tree example

Baseball hitter salaries (1987) (in $1,000s):



This is a regression tree – each region corresponds to an output value.

Prediction using trees

The prediction $\widehat{y}_{R_j}$ is a function of all training observations i in R_j.

Prediction using trees

The prediction $\widehat{y}_{R_j}$ is a function of all training observations i in R_j.

- Regression: $\widehat{y}_{R_j} = \bar{y}_{i:i \in R_j}$

Prediction using trees

The prediction $\widehat{y}_{R_j}$ is a function of all training observations i in R_j.

- Regression: $\widehat{y}_{R_j} = \bar{y}_{i:i \in R_j}$
- Classification: $\widehat{y}_{R_j} =$ most frequently occurring class y_i for $i \in R_j$

Prediction using trees

The prediction $\widehat{y}_{R_j}$ is a function of all training observations i in R_j.

- Regression: $\widehat{y}_{R_j} = \bar{y}_{i:i \in R_j}$
- Classification: $\widehat{y}_{R_j} =$ most frequently occurring class y_i for $i \in R_j$

To predict, trace a test observation to a leaf R_j based on the sequence of conditions. Predict $\widehat{y}_{R_j}$.

Prediction using trees

The prediction $\widehat{y}_{R_j}$ is a function of all training observations i in R_j.

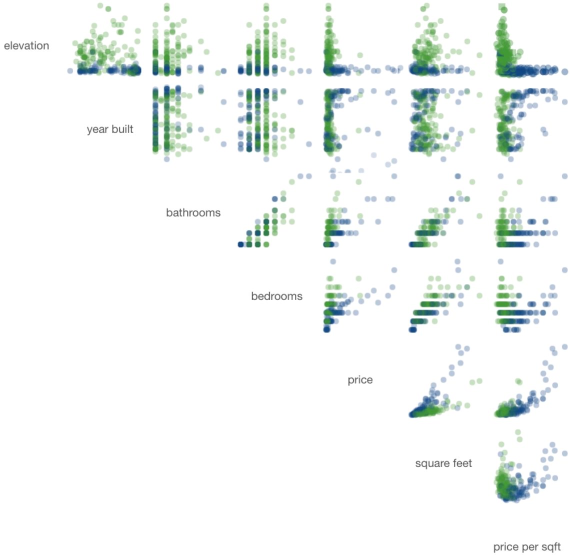
- Regression: $\widehat{y}_{R_j} = \bar{y}_{i:i \in R_j}$
- Classification: $\widehat{y}_{R_j} =$ most frequently occurring class y_i for $i \in R_j$

To predict, trace a test observation to a leaf R_j based on the sequence of conditions. Predict $\widehat{y}_{R_j}$.

Fitting a tree boils down to identifying the appropriate set of regions $R_1, \ldots, R_J$ that "best" describes the relationship between X and y.

How to build a (classification) tree?

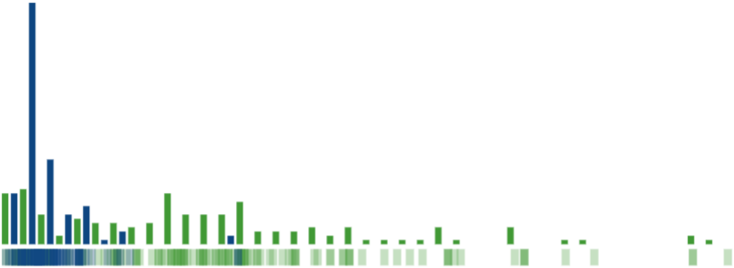
Example: New York vs San Francisco housing data.



How to build a (classification) tree?

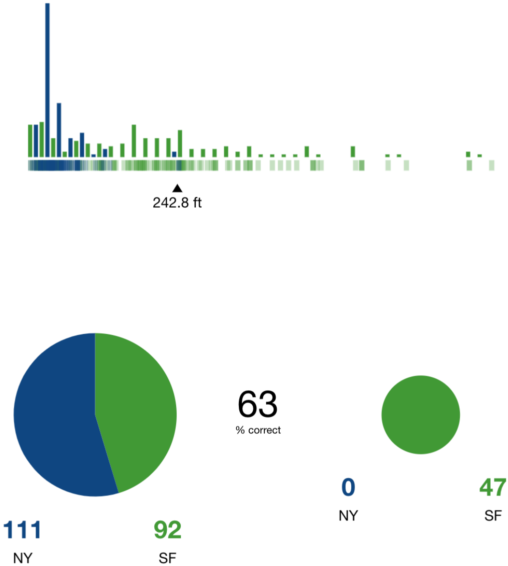
Let's start with one feature: elevation.
Where should we create a split?

Rule of thumb: try to split the data into two homogeneous parts.



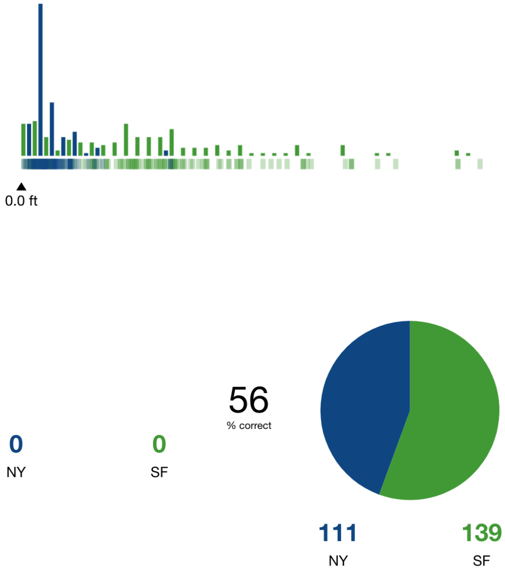
How to build a (classification) tree?

Rule of thumb: try to split the data into two homogeneous parts.



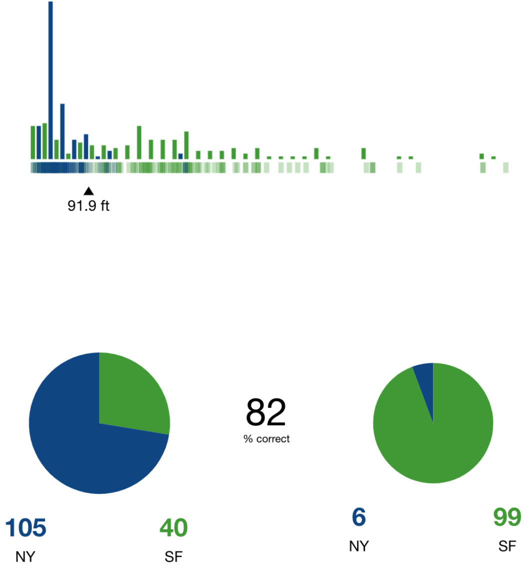
How to build a (classification) tree?

Rule of thumb: try to split the data into two homogeneous parts.



How to build a (classification) tree?

Rule of thumb: try to split the data into two homogeneous parts.



How to build a (classification) tree?

More formally...

Let $\widehat{p}_{jk}$ be the proportion of training observations of class k in R_j.

Recall: $\widehat{y}_j$ is *most commonly occurring class* of training observations in R_j: i.e., $\widehat{y}_j = \arg\max_k \widehat{p}_{jk}$

How to build a (classification) tree?

Let $\widehat{p}_{jk}$ be the proportion of training observations of class k in R_j.

Want to encourage higher **node purity.** Do this by minimizing:

How to build a (classification) tree?

Let $\widehat{p}_{jk}$ be the proportion of training observations of class k in R_j.

Want to encourage higher **node purity.** Do this by minimizing:

- Classification error rate (as in the example above)

$$E = \sum_{j=1}^{J} |R_j|(1 - \max_k(\widehat{p}_{jk}))$$

How to build a (classification) tree?

Let $\widehat{p}_{jk}$ be the proportion of training observations of class k in R_j.

Want to encourage higher **node purity.** Do this by minimizing:

- Classification error rate (as in the example above)

$$E = \sum_{j=1}^{J} |R_j|(1 - \max_k(\widehat{p}_{jk}))$$

- Gini index

$$G = \sum_{j=1}^{J} |R_j| \sum_{k=1}^{K} \widehat{p}_{jk}(1 - \widehat{p}_{jk})$$

How to build a (classification) tree?

Let $\widehat{p}_{jk}$ be the proportion of training observations of class k in R_j.

Want to encourage higher **node purity.** Do this by minimizing:

- Classification error rate (as in the example above)

$$E = \sum_{j=1}^{J} |R_j|(1 - \max_k(\widehat{p}_{jk}))$$

- Gini index

$$G = \sum_{j=1}^{J} |R_j| \sum_{k=1}^{K} \widehat{p}_{jk}(1 - \widehat{p}_{jk})$$

- Other potential metrics...

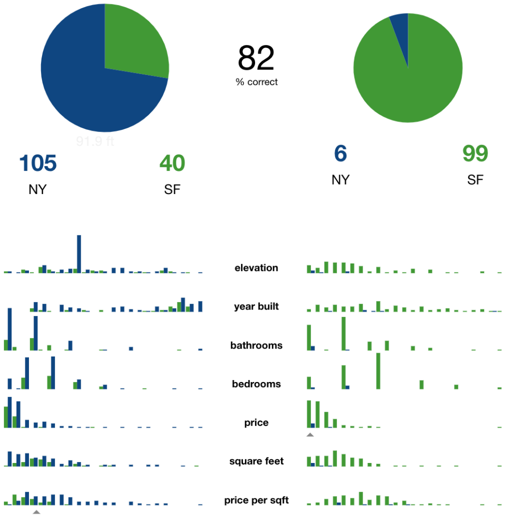
How to build a (classification) tree?

So far we have only made the split at the **root** of the tree.

For each branch, we can select another feature to split on.

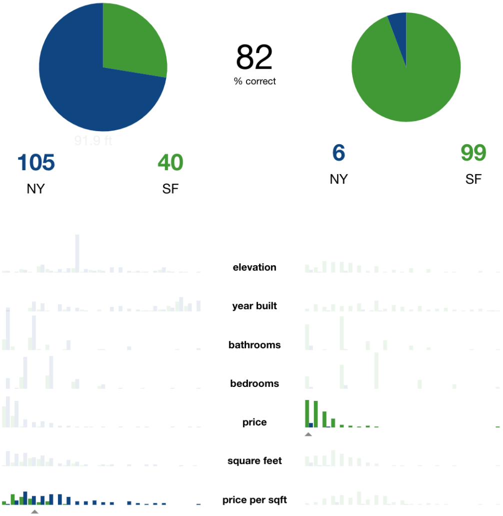
How to build a (classification) tree?

For each branch, we can select another feature to split on.



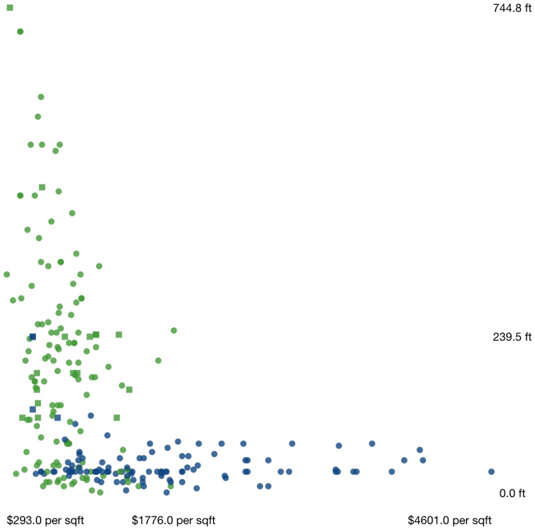
How to build a (classification) tree?

For each branch, we can select another feature to split on.



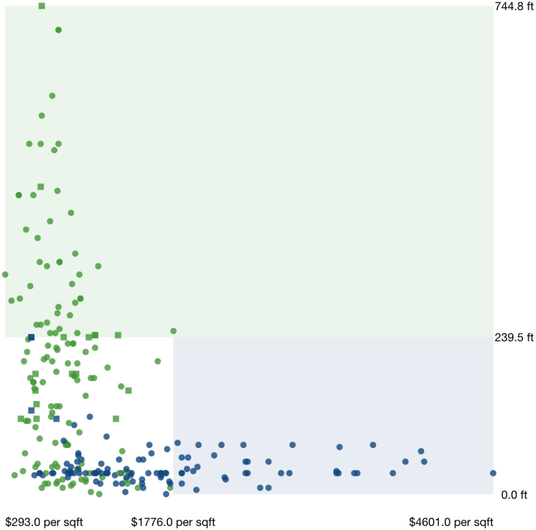
How to build a (classification) tree?

Alternate visualization for 2D:



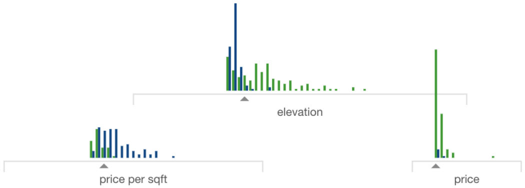
How to build a (classification) tree?

Alternate visualization for 2D:



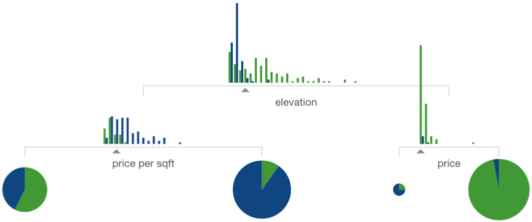
How to build a (classification) tree?

Recurse: For each branch, select another feature to split on until all leaves are pure.



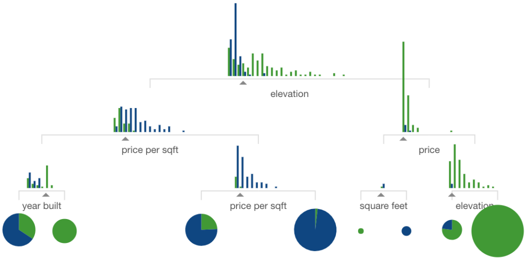
How to build a (classification) tree?

Recurse: For each branch, select another feature to split on until all leaves are pure.



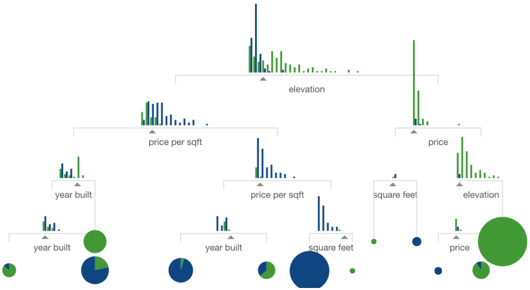
How to build a (classification) tree?

Recurse: For each branch, select another feature to split on until all leaves are pure.



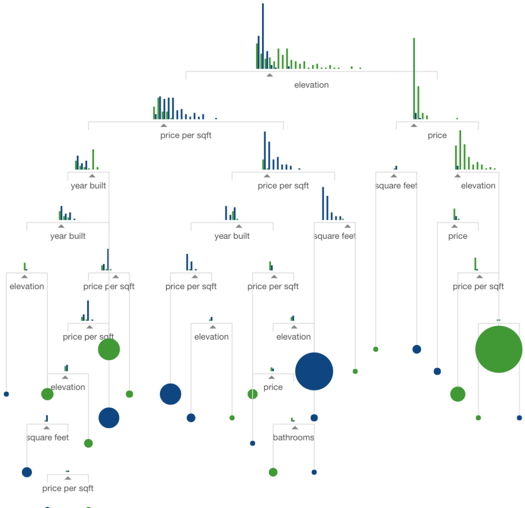
How to build a (classification) tree?

Recurse: For each branch, select another feature to split on until all leaves are pure.



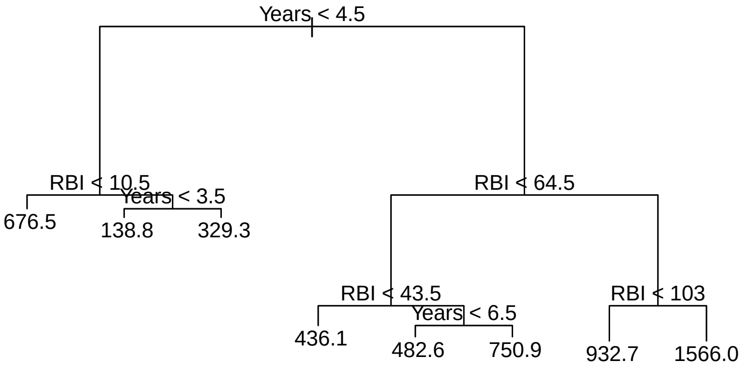
How to build a (classification) tree?

Recurse: For each branch, select another feature to split on until all leaves are pure.



Regression tree example

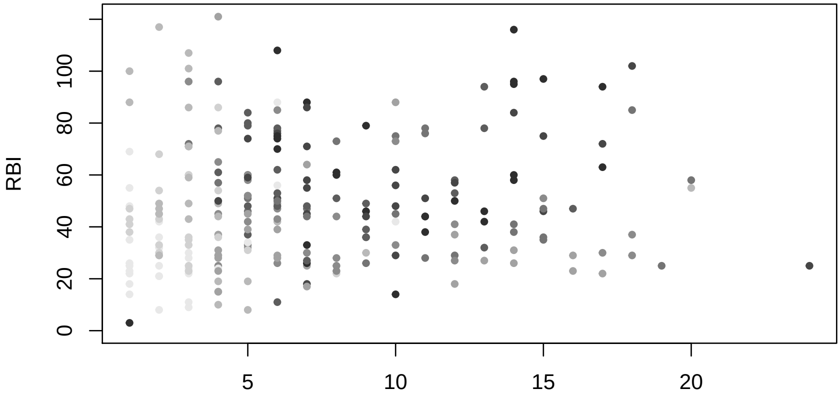
Baseball hitter salaries (1987) (in $1,000s):



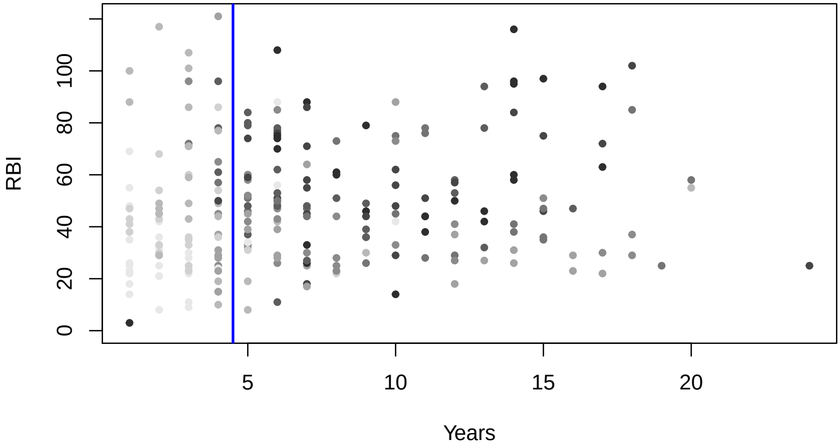
How to build a (regression) tree?

Let's look at the 2D version:

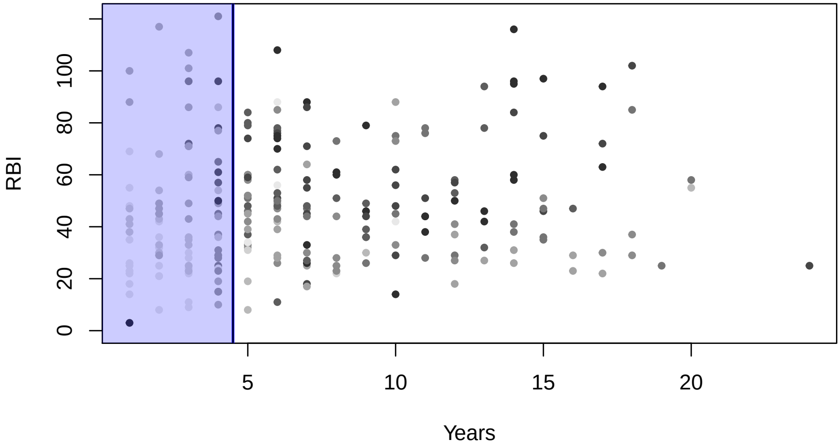
Where can we draw a horizontal or vertical line that best splits the data into two homogeneous parts?



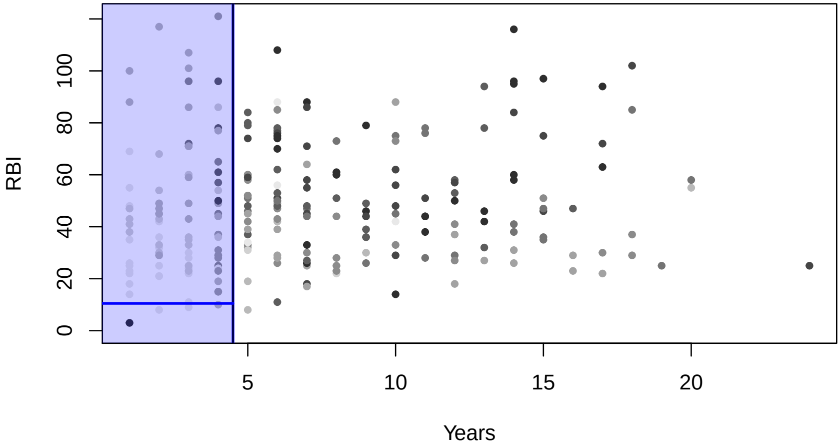
How to build a (regression) tree?



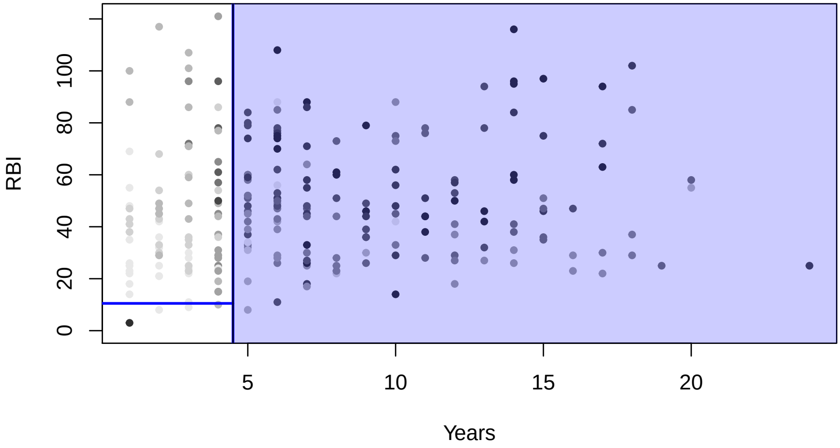
How to build a (regression) tree?



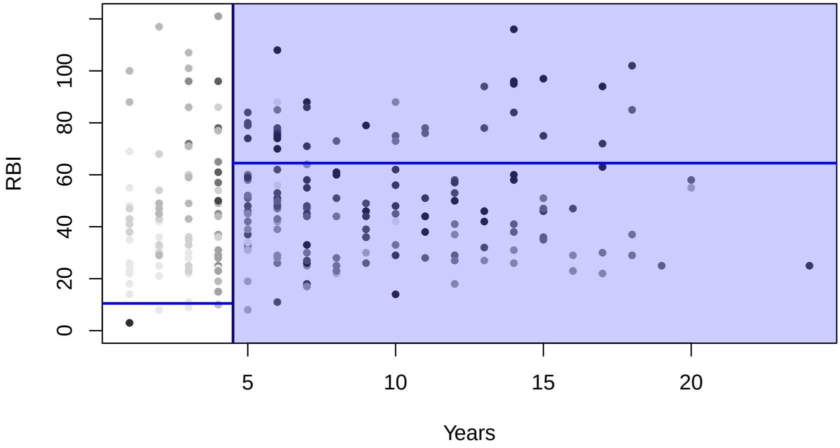
How to build a (regression) tree?



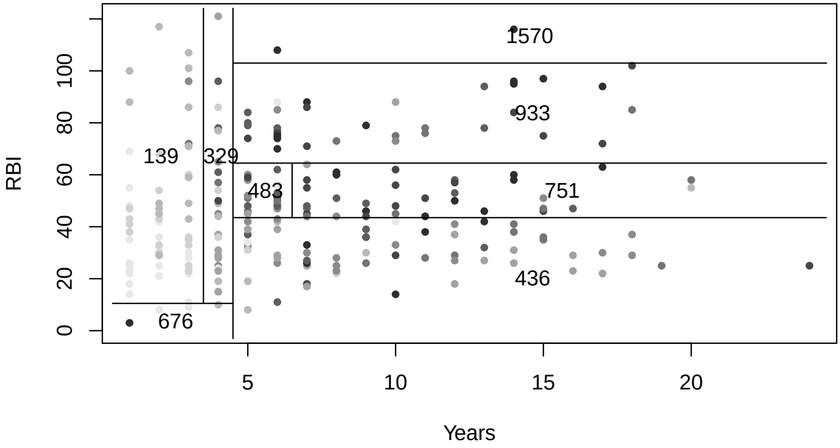
How to build a (regression) tree?



How to build a (regression) tree?



How to build a (regression) tree?



How to build a (regression) tree?

More formally...

We again want to choose $R_1, \ldots, R_J$ to minimize error. In this case, the residual sum of squares:

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

How to build a (regression) tree?

More formally...

We again want to choose $R_1, \ldots, R_J$ to minimize error. In this case, the residulal sum of squares:

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

To build a full tree: continue until every test point is in its own leaf ($RSS = 0$).

How to build a (regression) tree?

More formally...

We again want to choose $R_1, \ldots, R_J$ to minimize error. In this case, the residulal sum of squares:

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

To build a full tree: continue until every test point is in its own leaf ($RSS = 0$).

This is called a **stopping criterion** – we will consider others later.

Tree growth algorithm (general)

For each leaf node v that has not hit the stopping criterion:

Tree growth algorithm (general)

For each leaf node v that has not hit the stopping criterion:

1. For each feature X_k for $k = 1, \ldots, p$:

Tree growth algorithm (general)

For each leaf node v that has not hit the stopping criterion:

1. For each feature X_k for $k = 1, \ldots, p$:
   - Consider cutpoints s (i.e., unique values of X_k) that divide the training points in region R_v into two parts. E.g.:

$$R_{v1}(k, s) = \{i \mid X_{ik} < s\} \quad \text{and} \quad R_{v2}(k, s) = \{i \mid X_{ik} \geq s\}$$

Tree growth algorithm (general)

For each leaf node v that has not hit the stopping criterion:

1. For each feature X_k for $k = 1, \ldots, p$:
   - Consider cutpoints s (i.e., unique values of X_k) that divide the training points in region R_v into two parts.
   - Evaluate metric $Q_k(s)$ for each. E.g., for regression trees:

$$Q_k(s) = \sum_{i:i \in R_1(k,s)} (y_i - \bar{y}_{R_1})^2 + \sum_{i:i \in R_2(k,s)} (y_i - \bar{y}_{R_2})^2$$

Tree growth algorithm (general)

For each leaf node v that has not hit the stopping criterion:

1. For each feature X_k for $k = 1, \ldots, p$:
    - Consider cutpoints s (i.e., unique values of X_k) that divide the training points in region R_v into two parts.
    - Evaluate metric $Q_k(s)$ for each.
    - Find the value of s that minimizes $Q_k(s)$. Call this s_k.

Tree growth algorithm (general)

For each leaf node v that has not hit the stopping criterion:

1. For each feature X_k for $k = 1, \ldots, p$:
   - Consider cutpoints s (i.e., unique values of X_k) that divide the training points in region R_v into two parts.
   - Evaluate metric $Q_k(s)$ for each.
   - Find the value of s that minimizes $Q_k(s)$. Call this s_k.
2. Find the predictor X_k with the minimum $Q_1(s_1), Q_2(s_2), \ldots, Q_p(s_p)$.

Tree growth algorithm (general)

For each leaf node v that has not hit the stopping criterion:

1. For each feature X_k for $k = 1, \ldots, p$:
   - Consider cutpoints s (i.e., unique values of X_k) that divide the training points in region R_v into two parts.
   - Evaluate metric $Q_k(s)$ for each.
   - Find the value of s that minimizes $Q_k(s)$. Call this s_k.
2. Find the predictor X_k with the minimum $Q_1(s_1), Q_2(s_2), \ldots, Q_p(s_p)$.

Make a partition for v along predictor X_k at cut point s_k.

Tree growth algorithm (general)

For each leaf node v that has not hit the stopping criterion:

1. For each feature X_k for $k = 1, \ldots, p$:
   - Consider cutpoints s (i.e., unique values of X_k) that divide the training points in region R_v into two parts.
   - Evaluate metric $Q_k(s)$ for each.
   - Find the value of s that minimizes $Q_k(s)$. Call this s_k.
2. Find the predictor X_k with the minimum $Q_1(s_1), Q_2(s_2), \ldots, Q_p(s_p)$.

Make a partition for v along predictor X_k at cut point s_k.

Repeat until all leaves have met stopping criterion.

Categorical predictors

What if X_k is categorical?

Categorical predictors

What if X_k is categorical?

Suppose X_k takes values in $\{A, B, C, D, E\}$.

Then, the possible splits include:

- $\{D\}$ vs. $\{A, B, C, E\}$
- $\{D, B\}$ vs. $\{A, C, E\}$
- $\ldots$

Categorical predictors

What if X_k is categorical?

Suppose X_k takes values in $\{A, B, C, D, E\}$.

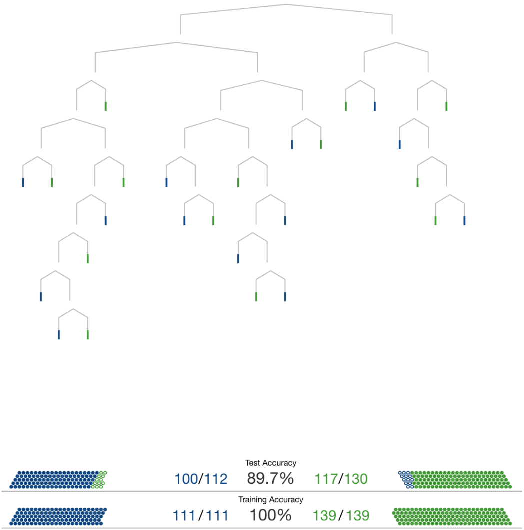
Then, the possible splits include:

- $\{D\}$ vs. $\{A, B, C, E\}$
- $\{D, B\}$ vs. $\{A, C, E\}$
- $\ldots$

Every possible partition of the set into 2 subsets are considered, and we proceed as before.

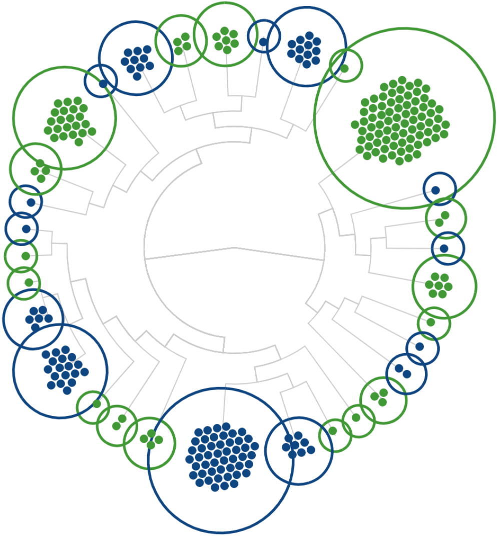
Perfect Trees?

Using the above we built perfect trees... **for the training data.**
In reality:



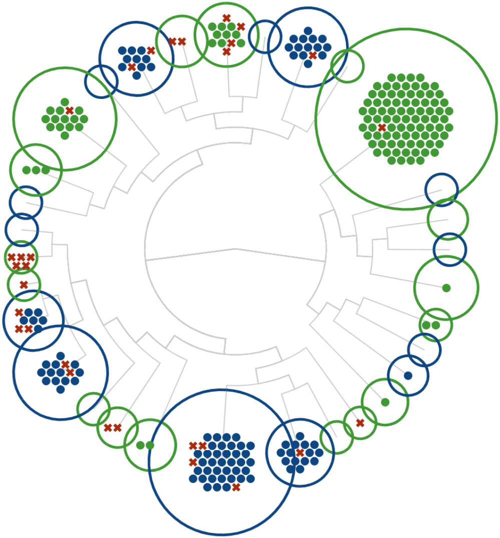
Errors: Variance

By building a full tree, we have created a lot of **variance.**



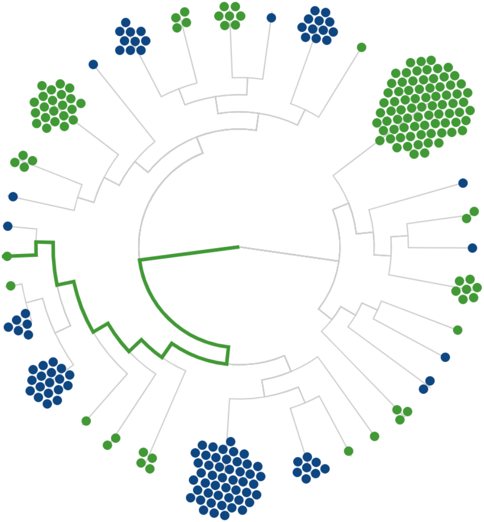
Errors: Variance

By building a full tree, we have created a lot of **variance.**



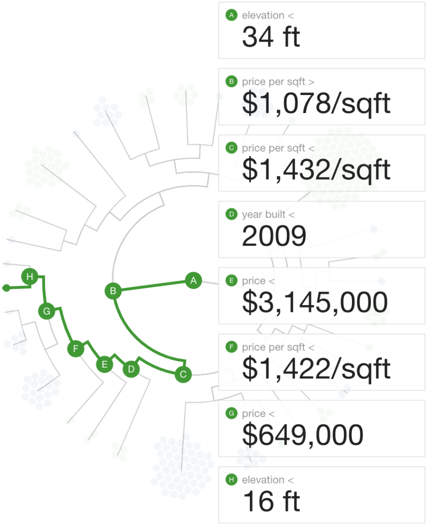
Errors: Variance

By building a full tree, we have created a lot of **variance.**



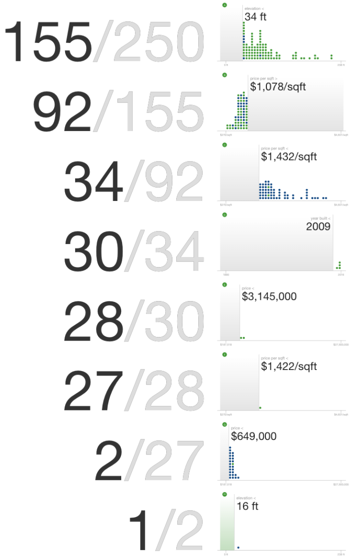
Errors: Variance

By building a full tree, we have created a lot of **variance.**



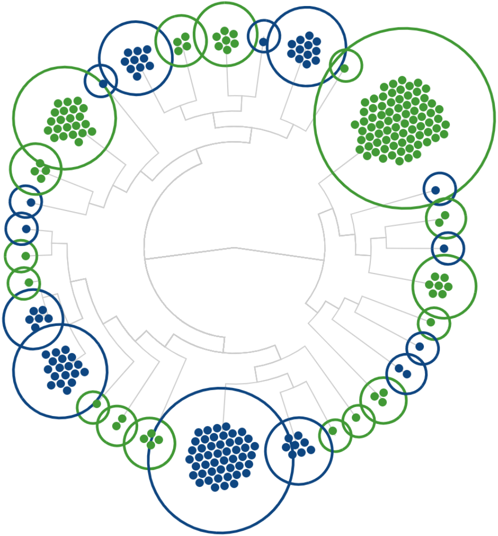
Errors: Variance

By building a full tree, we have created a lot of **variance.**



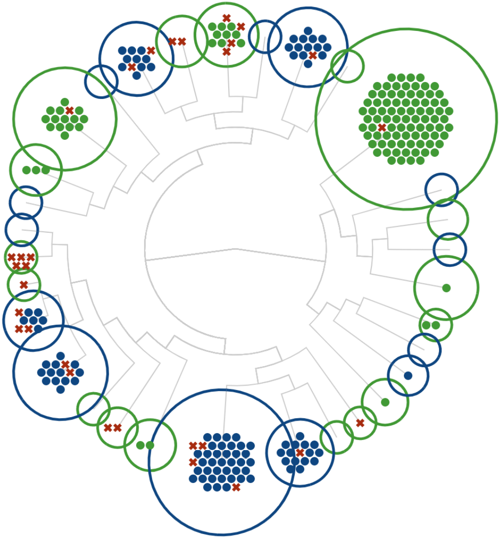
Errors: Variance

By building a full tree, we have allowed for significant **variance.**



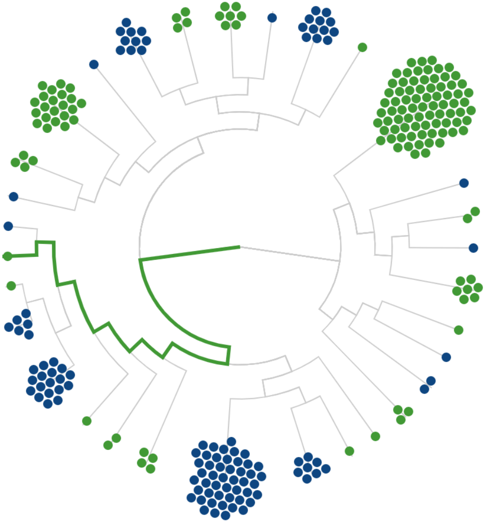
Errors: Variance

By building a full tree, we have allowed for significant **variance.**



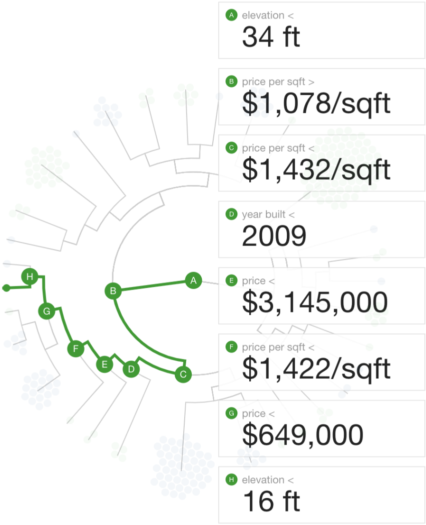
Errors: Variance

By building a full tree, we have allowed for significant **variance.**



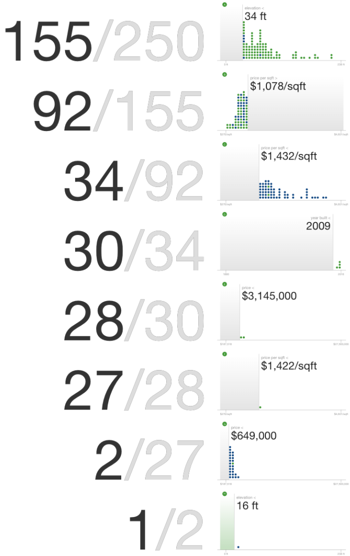
Errors: Variance

By building a full tree, we have allowed for significant **variance.**



Errors: Variance

By building a full tree, we have allowed for significant **variance.**



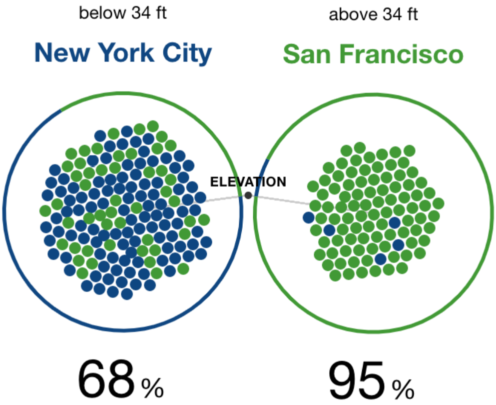
Errors: Variance

By building a full tree, we have allowed for significant **variance.**

What if we had a shorter tree?

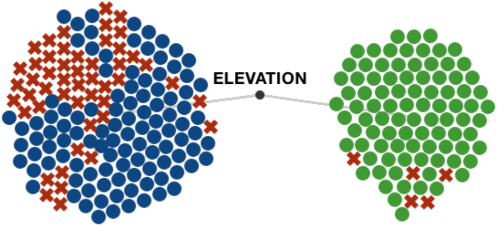
Errors: Bias

What if we had a shorter tree?

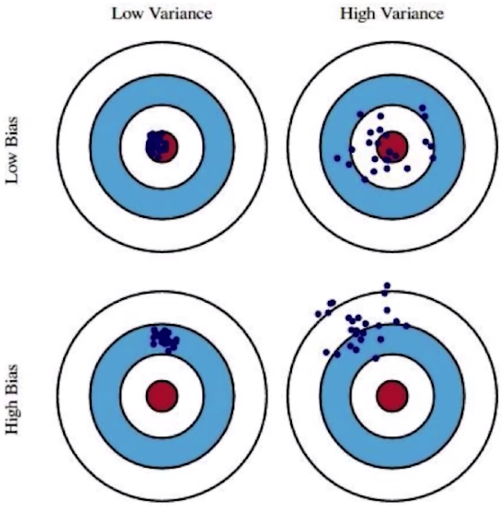


Errors: Bias

What if we had a shorter tree?



Bias vs Variance



Bias vs Variance

- As tree is grown deeper, bias decreases
- But the variance increases
- How to choose the right size of tree?

Bias vs Variance

- As tree is grown deeper, bias decreases
- But the variance increases
- How to choose the right size of tree?

Option 1: Change the stopping criterion.

Stopping criterion

Some possibilities:

- number of observations in a node has reached a minimum

Stopping criterion

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum

Stopping criterion

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum
- grow until no further splits can reduce RSS (or improve error rate) by some amount ϵ

Stopping criterion

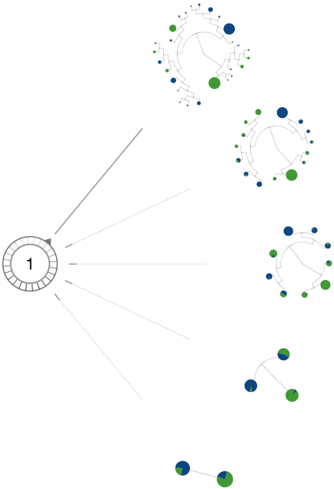
Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum
- grow until no further splits can reduce RSS (or improve error rate) by some amount ϵ

Many options – often resulting in tuning parameters that may be hard to deal with.

Stopping criterion

Example: Number of observations in a node has reached a minimum.

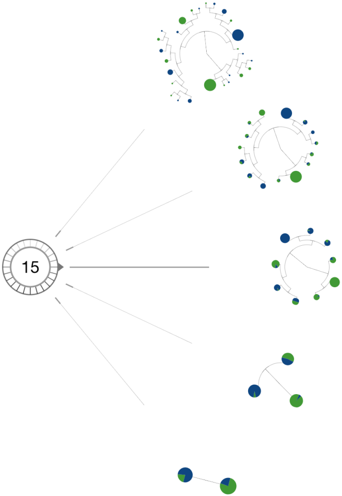


Stopping criterion

Example: Number of observations in a node has reached a minimum.

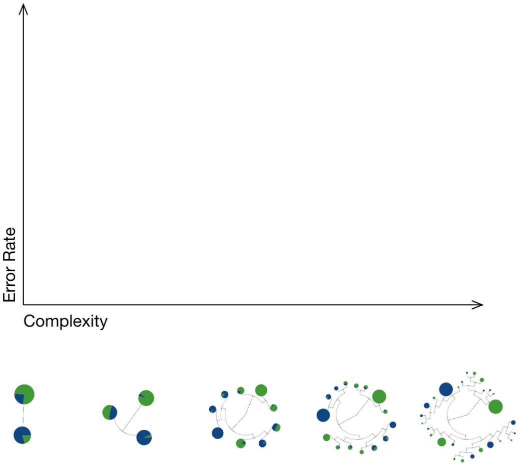
Stopping criterion

Example: Number of observations in a node has reached a minimum.



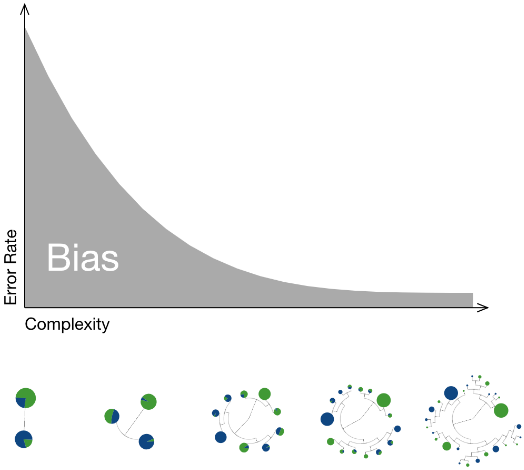
Bias vs Variance

Have to tune this parameter:



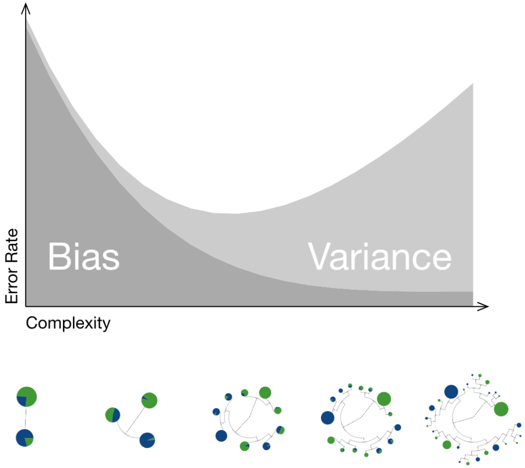
Bias vs Variance

Have to tune this parameter:



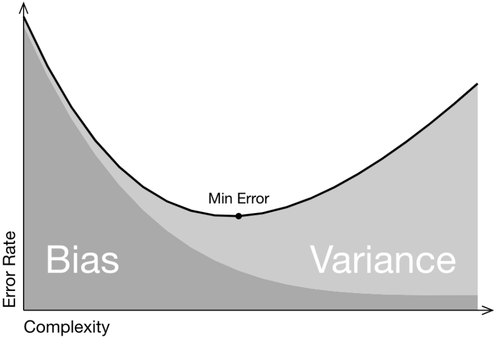
Bias vs Variance

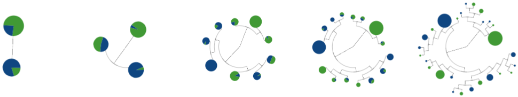
Have to tune this parameter:



Bias vs Variance

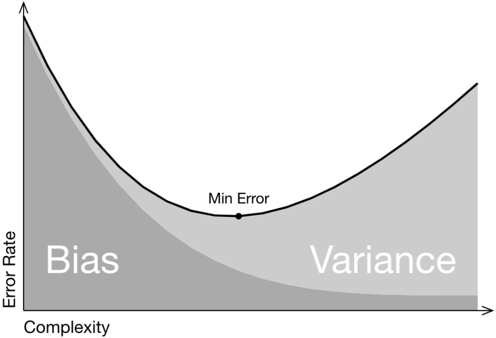
Have to tune this parameter:

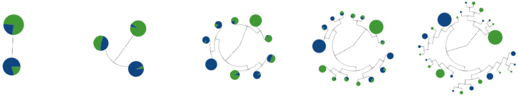




Bias vs Variance

Have to tune this parameter:





In general, these are coarse rules and not always ideal even when tuned.

Tree pruning

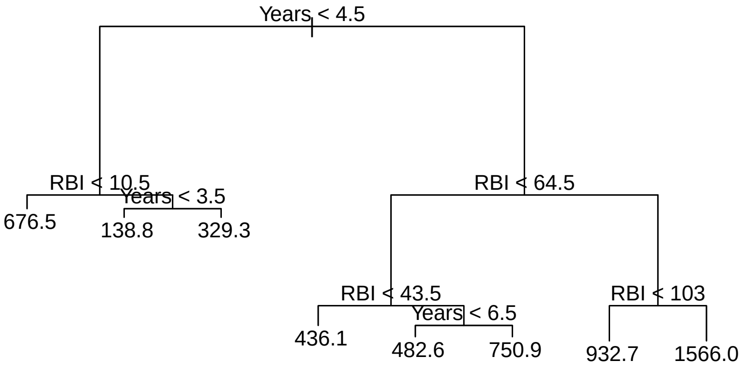
Option 2: Grow a large tree and then **prune** it back.

I.e., look at subtrees of the fully-grown tree, and comparing how well they perform.

Tree pruning

Option 2: Grow a large tree and then **prune** it back.

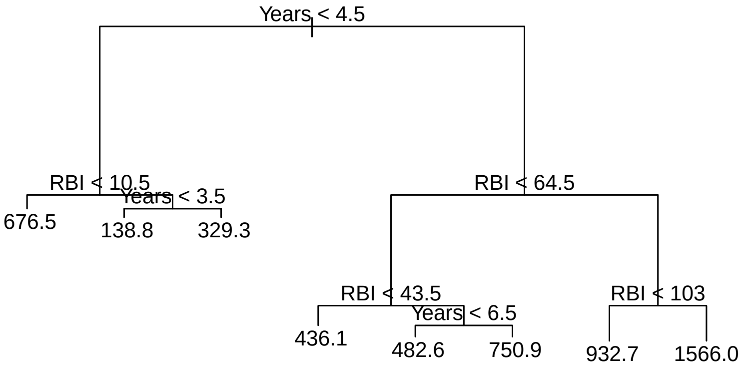
I.e., look at subtrees of the fully-grown tree, and comparing how well they perform.



Tree pruning

Option 2: Grow a large tree and then **prune** it back.

I.e., look at subtrees of the fully-grown tree, and comparing how well they perform.



Cost-complexity pruning

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

α is a tuning parameter that controls for the complexity of the model, and $|T|$ is the number of regions (leaves) of the tree.

Cost-complexity pruning

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

α is a tuning parameter that controls for the complexity of the model, and $|T|$ is the number of regions (leaves) of the tree.

- $\alpha = 0$ implies the full tree

Cost-complexity pruning

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

α is a tuning parameter that controls for the complexity of the model, and $|T|$ is the number of regions (leaves) of the tree.

- $\alpha = 0$ implies the full tree
- Larger α implies higher penalty for complexity of model

Cost-complexity pruning

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

α is a tuning parameter that controls for the complexity of the model, and $|T|$ is the number of regions (leaves) of the tree.

- $\alpha = 0$ implies the full tree
- Larger α implies higher penalty for complexity of model

Cost-complexity pruning

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

α is a tuning parameter that controls for the complexity of the model, and $|T|$ is the number of regions (leaves) of the tree.

- $\alpha = 0$ implies the full tree
- Larger α implies higher penalty for complexity of model

The set of trees corresponding to the various α form a sequence of nested subtrees!

Tree pruning

1. Grow a big tree on a *training set*.
2. Obtain the optimal set of nested subtrees $T_l \subset \cdots \subset T_2 \subset T_1 \subset T_0$ corresponding to the cost complexity minimization problem.
3. Identify the subtree/α that does best.

Nice properties of trees

- Interpretable.
- Inbuilt feature selection – irrelevant covariates won't be used as often as splits.
- Simple, fast implementation – (mostly) performs well with large datasets.

Problems with trees

- Instability – tree topology can change dramatically with slight changes to data. Interpretability kind of an illusion.
- Greedy – cannot guarantee to find the globally optimal decision tree.
- Lack of smoothness – the splits lead to a “jagged” decision boundary.
- Difficulty capturing additive structure – if the actual model is additive, this may not be captured by the tree with limited data.

What did we learn today?

- Trees are a nonparametric method
- Gives interpretable decision rules
- Shallow trees have high bias and low variance, deep trees have low bias, high variance
- Trees are grown greedily to the full, then pruned back