Insurance analytics

A walk in the forest - tree-based machine learning methods

Katrien Antonio & Bram Wouters
LRisk - KU Leuven and ASE - University of Amsterdam

May 7, 2019

Acknowledgement

- Some of the figures in this presentation are taken from *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

- Some of the figures in this presentation are from *Boosting insights in insurance tariff plans with tree-based machine learning* (available on arxiv, April 2019), written by Roel Henckaerts, Marie-Pier Côté, Katrien Antonio and Roel Verbelen.

Today's mission

- Today's mission is twofold:

Today's mission

- Today's mission is twofold:
  - a general discussion of decision trees, bagging, random forests, gradient boosting machines

Today's mission

- Today's mission is twofold:
  - a general discussion of decision trees, bagging, random forests, gradient boosting machines
  - a discussion of specific considerations to keep in mind when using these predictive modeling techniques with frequency/severity data.

Regression trees

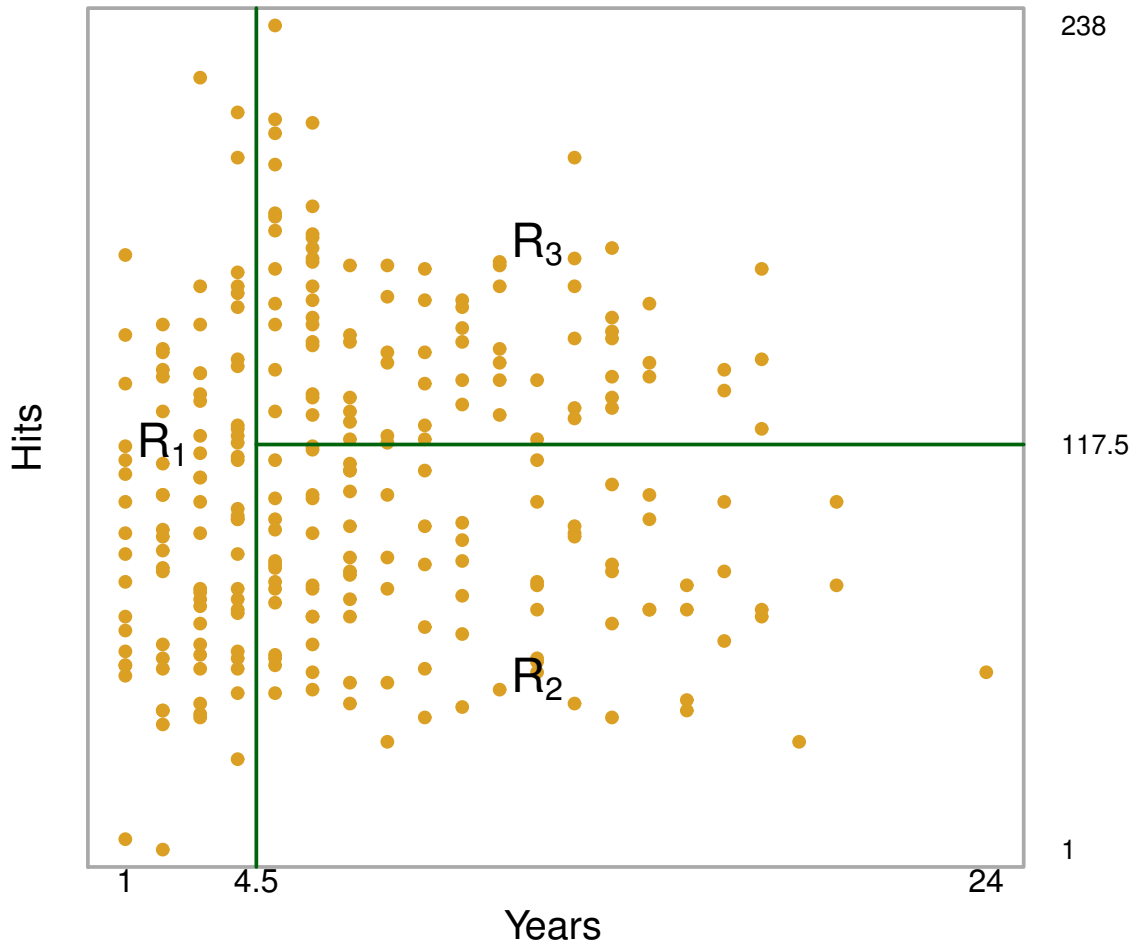
Intro

- CART: Classification And Regression Trees, introduced by Breiman et al. (1984).
- Consider the `Hitters` data to predict a player's `Salary` based on `Years` and `Hits`.

Regression trees

Intro

- The region partition for the `Hitters` data:



Regression trees

Intro

- The process of building a regression tree:

Regression trees

Intro

- The process of building a regression tree:

  1. divide the predictor space into J distinct, non-overlapping regions $R_1, R_2, \ldots, R_J$

     recall: the predictor space is the set of possible values for $X_1, X_2, \ldots, X_p$ (= the covariates).

Regression trees

Intro

- The process of building a regression tree:

  1. divide the predictor space into J distinct, non-overlapping regions $R_1, R_2, \ldots, R_J$

     recall: the predictor space is the set of possible values for $X_1, X_2, \ldots, X_p$ (= the covariates).

  2. for every observation in region R_j we make the same prediction:

     the mean of the response values for the training observations in R_j.

Regression trees

Intro

- The process of building a regression tree:

  1. divide the predictor space into J distinct, non-overlapping regions $R_1, R_2, \ldots, R_J$

     recall: the predictor space is the set of possible values for $X_1, X_2, \ldots, X_p$ (= the covariates).

  2. for every observation in region R_j we make the same prediction:

     the mean of the response values for the training observations in R_j.

- The prediction obtained with a regression tree:

$$f(X_1, \ldots, X_p) = \bar{y}_1 I_{\{\boldsymbol{x} \in R_1\}} + \ldots + \bar{y}_J I_{\{\boldsymbol{x} \in R_J\}},$$

where $\bar{y}_j = \text{ave}\,(y_i | \boldsymbol{x}_i \in R_j)$.

Regression trees

Intro

- We consider now step 1:

how to construct the regions $R_1, \ldots, R_J$?

Regression trees

Intro

- We consider now step 1:

  how to construct the regions $R_1, \ldots, R_J$?

- We divide the predictor space into high-dimensional rectangles, or boxes.

Regression trees

Intro

- We consider now step 1:

  how to construct the regions $R_1, \ldots, R_J$?

- We divide the predictor space into high-dimensional rectangles, or boxes.

- Find boxes $R_1, \ldots, R_J$ that minimize the Residual Sum of Squares (RSS) (actuarial reflections?)

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

  with $\bar{y}_{R_j}$ the mean response for the training observations within the jth box.

Regression trees

Intro

- Computationally infeasible to consider every possible partition of the feature space into J boxes.

Regression trees

Intro

- Computationally infeasible to consider every possible partition of the feature space into J boxes.
- Therefore:

  use a top-down, greedy approach, known as recursive binary splitting.

Regression trees

Intro

- Computationally infeasible to consider every possible partition of the feature space into J boxes.
- Therefore:

  use a top-down, greedy approach, known as recursive binary splitting.
- Motivation:
  - top-down because it begins at top of the tree
  - greedy because at each step the best split is made at that particular step, rather than looking ahead.

Regression trees

Intro

- To perform recursive binary splitting:

Regression trees

Intro

- To perform recursive binary splitting:
  - select the predictor X_j and cutpoint s such that

$$\{X|X_j < s\} \text{ and } \{X|X_j \geq s\},$$

    leads to greatest possible reduction in RSS

Regression trees

Intro

- To perform recursive binary splitting:
  - select the predictor X_j and cutpoint s such that

    $$\{X|X_j < s\} \text{ and } \{X|X_j \geq s\},$$

    leads to greatest possible reduction in RSS
  - thus, for any j and s we define the pair of half-planes

    $$R_1(j,s) \quad = \quad \{X|X_j < s\} \quad \text{and} \quad R_2(j,s) = \{X|X_j \geq s\},$$

    we seek j and s minimizing

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

Regression trees

Intro

- Next, we repeat the process:

Regression trees

Intro

- Next, we repeat the process:

  we look for the best predictor and best cutpoint to split data further,

Regression trees

Intro

- Next, we repeat the process:

  we look for the best predictor and best cutpoint to split data further,

  within each of the resulting regions.

Regression trees

Intro

- Next, we repeat the process:

  we look for the best predictor and best cutpoint to split data further,

  within each of the resulting regions.

- The process continues until a stopping criterion is reached, e.g. no region has > 5 observations.

Regression trees

Intro

- Next, we repeat the process:

  we look for the best predictor and best cutpoint to split data further,

  within each of the resulting regions.

- The process continues until a stopping criterion is reached, e.g. no region has > 5 observations.

- Given regions $R_1, \ldots, R_J$ we predict as follows: (step 2)

Regression trees

Intro

- Next, we repeat the process:

  we look for the best predictor and best cutpoint to split data further,

  within each of the resulting regions.

- The process continues until a stopping criterion is reached, e.g. no region has > 5 observations.

- Given regions $R_1, \ldots, R_J$ we predict as follows: (step 2)

  - take a test observation
  - belongs to which region?
  - use the mean of the training observations in that region.

Regression trees

Loss functions

- What about alternatives for the Residual Sum of Squares (RSS)?

Regression trees

Loss functions

- What about alternatives for the Residual Sum of Squares (RSS)?
- Use a loss function $L(y_i, f(\boldsymbol{x}_i))$ and split the predictor space into $R_1(j,s)$ and $R_2(j,s)$ such that

$$\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)) = \sum_{i:\ \boldsymbol{x}_i \in R_1(j,s)} L(y_i, \bar{y}_{R_1}) + \sum_{i:\ \boldsymbol{x}_i \in R_2(j,s)} L(y_i, \bar{y}_{R_2})$$

is minimized.

Regression trees

Loss functions

- What about alternatives for the Residual Sum of Squares (RSS)?
- Use a loss function $L(y_i, f(\boldsymbol{x}_i))$ and split the predictor space into $R_1(j,s)$ and $R_2(j,s)$ such that

$$\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)) = \sum_{i:\ \boldsymbol{x}_i \in R_1(j,s)} L(y_i, \bar{y}_{R_1}) + \sum_{i:\ \boldsymbol{x}_i \in R_2(j,s)} L(y_i, \bar{y}_{R_2})$$

  is minimized.
- Hereby $\bar{y}_{R_j}$ the 'mean' response for the training observations within the jth box.

Regression trees

Loss functions inspired by GLMs

- Recall the notion of the (scaled) deviance:

$$D(\boldsymbol{y}, \hat{f}(\boldsymbol{x})) \quad = \quad -2 \cdot \ln \left(\frac{\mathcal{L}(\hat{f}(\boldsymbol{x}))}{\mathcal{L}(\boldsymbol{y})} \right),$$

where $\boldsymbol{y}$ is the vector of responses (or: targets) and $\hat{f}(\boldsymbol{x})$ is the vector of fitted values, $\mathcal{L}(\boldsymbol{y})$ is the likelihood of the saturated model and $\mathcal{L}(f(\boldsymbol{x}))$ the model likelihood.

Regression trees

Loss functions inspired by GLMs

- Recall the notion of the (scaled) deviance:

$$D(\boldsymbol{y}, \hat{f}(\boldsymbol{x})) \quad = \quad -2 \cdot \ln \left(\frac{\mathcal{L}(\hat{f}(\boldsymbol{x}))}{\mathcal{L}(\boldsymbol{y})} \right),$$

  where $\boldsymbol{y}$ is the vector of responses (or: targets) and $\hat{f}(\boldsymbol{x})$ is the vector of fitted values, $\mathcal{L}(\boldsymbol{y})$ is the likelihood of the saturated model and $\mathcal{L}(f(\boldsymbol{x}))$ the model likelihood.

- Now, use a loss function $L(y_i, f(\boldsymbol{x}_i))$ such that

$$D(\boldsymbol{y}, \hat{f}(\boldsymbol{x})) \quad = \quad \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)).$$

Regression trees

Loss functions inspired by GLMs

- For example, loss function inspired by Poisson deviance

$$\begin{aligned} D(\boldsymbol{y}, \hat{f}(\boldsymbol{x})) &= 2 \cdot \ln \prod_{i=1}^{n} \exp(-y_i) \frac{y_i^{y_i}}{y_i!} \\ &\quad -2 \cdot \ln \prod_{i=1}^{n} \exp(-\hat{f}(\boldsymbol{x}_i)) \frac{\hat{f}(\boldsymbol{x}_i)^{y_i}}{y_i!} \\ &= 2 \sum_{i=1}^{n} \left(y_i \cdot \ln \frac{y_i}{\hat{f}(\boldsymbol{x}_i)} - (y_i - \hat{f}(\boldsymbol{x}_i)) \right). \end{aligned}$$

The corresponding (weighted) loss function $L(y_i, f(\boldsymbol{x}_i))$ is then

$$L(y_i, f(\boldsymbol{x}_i)) = 2 \cdot w_i \cdot (y_i \cdot \ln y_i - y_i \cdot \ln \hat{f}(\boldsymbol{x}_i) - y_i + \hat{f}(\boldsymbol{x}_i)).$$

When using an exposure measure d_i, $f(\boldsymbol{x}_i)$ is replaced by $d_i \cdot f(\boldsymbol{x}_i)$.

Regression trees

Loss functions inspired by GLMs

- For example, loss function inspired by normal deviance

$$
\begin{aligned}
D(\boldsymbol{y}, \hat{f}(\boldsymbol{x})) &= 2 \cdot \ln \prod_{i=1}^{n} \exp\left(-\frac{1}{2\sigma^2}(y_i - y_i)^2\right) \\
&\quad -2 \cdot \ln \prod_{i=1}^{n} \exp\left(-\frac{1}{2\sigma^2}(y_i - \hat{f}(\boldsymbol{x}_i))^2\right) \\
&= \frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i - \hat{f}(\boldsymbol{x}_i))^2.
\end{aligned}
$$

The corresponding (weighted) loss function $L(y_i, f(\boldsymbol{x}_i))$ is then

$$
L(y_i, f(\boldsymbol{x}_i)) = w_i \cdot (y_i - \hat{f}(\boldsymbol{x}_i))^2,
$$

which is simply the squared error loss.

Regression trees

Finding optimal splits

- Finding the optimal split:

Regression trees

Finding optimal splits

- Finding the optimal split:
  - is straightforward for continuous predictors which can be ordered in a natural way

Regression trees

Finding optimal splits

- Finding the optimal split:
  - is straightforward for continuous predictors which can be ordered in a natural way
  - is easy for binary predictors

Regression trees

Finding optimal splits

- Finding the optimal split:
  - is straightforward for continuous predictors which can be ordered in a natural way
  - is easy for binary predictors
  - is way more complicated for categorical predictors with (e.g.) q levels, since $2^{q-1} - 1$ possible partitions in two groups.

Regression trees

Finding optimal splits

- Finding the optimal split:
  - is straightforward for continuous predictors which can be ordered in a natural way
  - is easy for binary predictors
  - is way more complicated for categorical predictors with (e.g.) q levels, since $2^{q-1} - 1$ possible partitions in two groups.
- Actuarial reflections? Postal code, multi-level factors?

Regression trees

Tree pruning

- Process as described above is likely to overfit the data.

Regression trees

Tree pruning

- Process as described above is likely to overfit the data.
- An alternative:

  build the tree so long as the decrease in the RSS due to each split exceeds some (high) threshold.

Regression trees

Tree pruning

- Process as described above is likely to overfit the data.
- An alternative:

  build the tree so long as the decrease in the RSS due to each split exceeds some (high) threshold.

- Better strategy:

  grow a very large tree T_0 and prune it back to obtain a subtree.

- This is called cost complexity pruning.

Regression trees

Tree pruning

- The strategy:

Regression trees

Tree pruning

- The strategy:
  - consider a sequence of trees indexed by a nonnegative tuning parameter α

Regression trees

Tree pruning

- The strategy:
  - consider a sequence of trees indexed by a nonnegative tuning parameter α
  - for each value of α there is a subtree $T \subset T_0$ such that

$$\sum_{j=1}^{|T|} \sum_{i:\ x_i \in R_j} (y_i - \bar{y}_{R_j})^2 + \alpha |T|$$

    is as small as possible.

Regression trees

Tree pruning

- The strategy:
  - consider a sequence of trees indexed by a nonnegative tuning parameter α
  - for each value of α there is a subtree $T \subset T_0$ such that

$$\sum_{j=1}^{|T|} \sum_{i:\ x_i \in R_j} (y_i - \bar{y}_{R_j})^2 + \alpha |T|$$

    is as small as possible.

- Hereby:
  - $|T|$ is the number of terminal nodes of the tree
  - $\bar{y}_{R_j}$ is the predicted response associated with R_j, the rectangle corresponding to the jth terminal node.

Regression trees

Tree pruning

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

Regression trees

Tree pruning

- Tuning parameter α controls a trade-off between the subtree's complexity and its fit to training data.
- With $\alpha = 0$ the subtree T is equal to T_0.

Regression trees

Tree pruning

- Tuning parameter α controls a trade-off between the subtree's complexity and its fit to training data.
- With $\alpha = 0$ the subtree T is equal to T_0.
- When α increases, there is a price to pay for having a tree with many terminal nodes.

Regression trees

Tree pruning

- Tuning parameter α controls a trade-off between the subtree's complexity and its fit to training data.
- With $\alpha = 0$ the subtree T is equal to T_0.
- When α increases, there is a price to pay for having a tree with many terminal nodes.
- We can select a value of α using a validation set or using cross-validation.

Regression trees

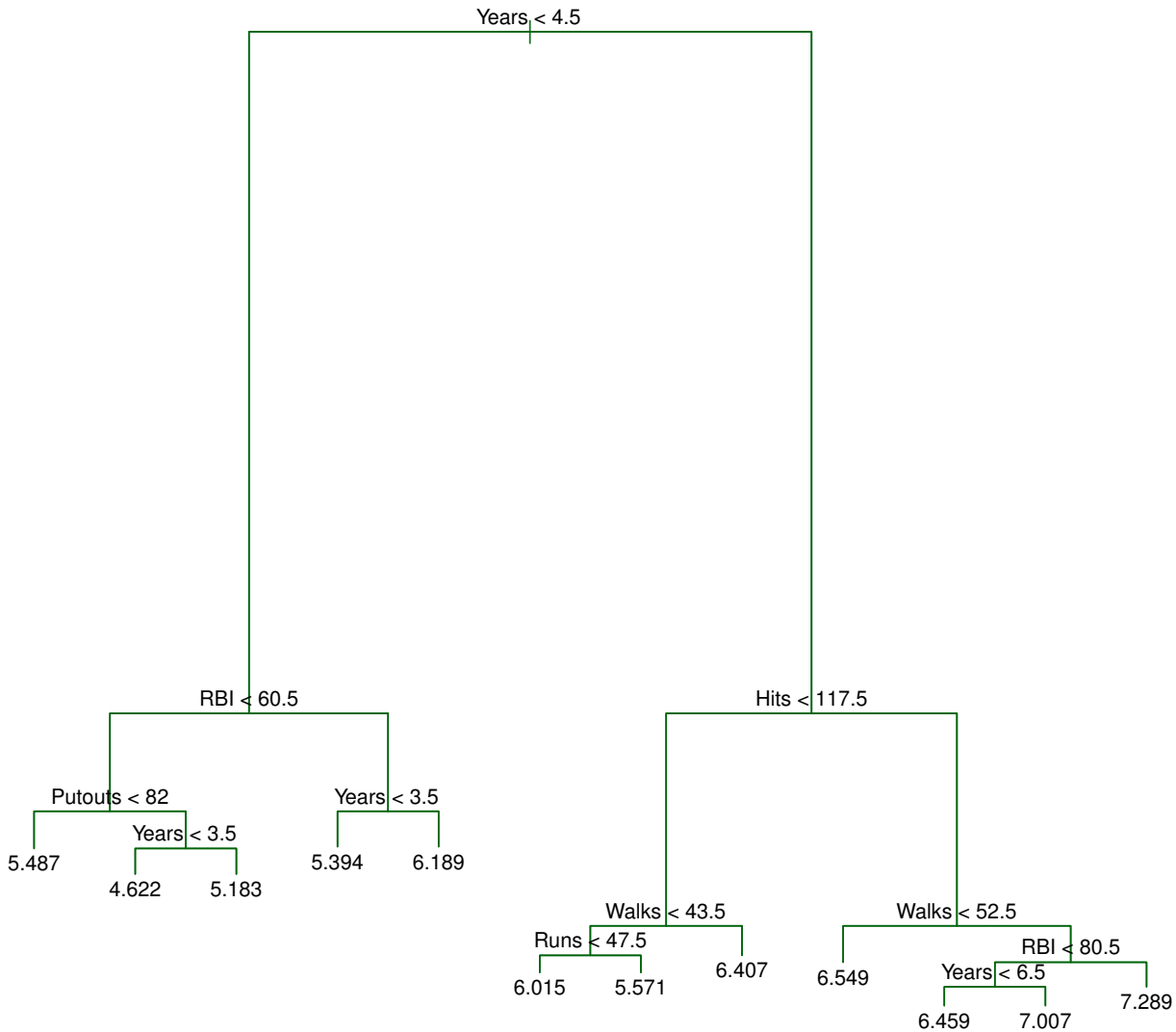
Tree pruning

- Tuning parameter α controls a trade-off between the subtree's complexity and its fit to training data.
- With $\alpha = 0$ the subtree T is equal to T_0.
- When α increases, there is a price to pay for having a tree with many terminal nodes.
- We can select a value of α using a validation set or using cross-validation.
- We return to the full data and obtain the subtree corresponding to α.

Regression trees

Tree pruning

Return to the `Hitters` data set: unpruned tree.



Regression trees

Tree pruning

- Return to the `Hitters` data set: unpruned tree.

Regression trees

Tree pruning

- Return to the `Hitters` data set: unpruned tree.
- Strategy:

Regression trees

Tree pruning

- Return to the `Hitters` data set: unpruned tree.
- Strategy:
  - randomly divide the data in half, yields 132 observations in the training set and 131 in the test set

Regression trees

Tree pruning

- Return to the `Hitters` data set: unpruned tree.
- Strategy:
  - randomly divide the data in half, yields 132 observations in the training set and 131 in the test set
  - build a large tree on the training data and vary α to create subtrees

Regression trees

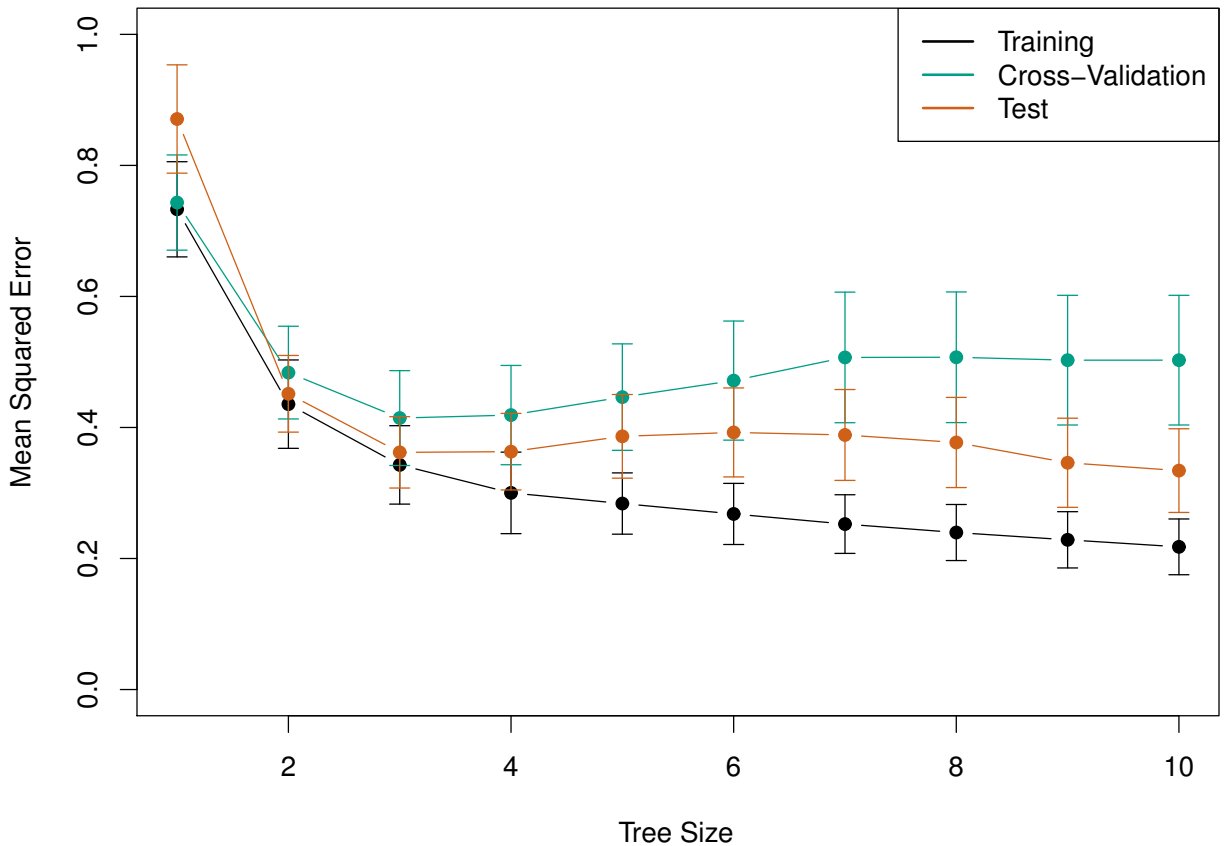
Tree pruning

- Return to the `Hitters` data set: unpruned tree.
- Strategy:
  - randomly divide the data in half, yields 132 observations in the training set and 131 in the test set
  - build a large tree on the training data and vary α to create subtrees
  - perform 6-fold cross validation and estimate cross-validated MSE of the trees as function of α.

Regression trees

Tree pruning

Return to the `Hitters` data set: training, cross-validation and test error.



CV error is minimal for the three-node tree (see earlier).

Regression trees

Tree pruning

- Return to the `Hitters` data set: pruned tree.

Regression trees

Tree in R with `rpart`

- Optimize performance of the tree by minimizing the following quantity:

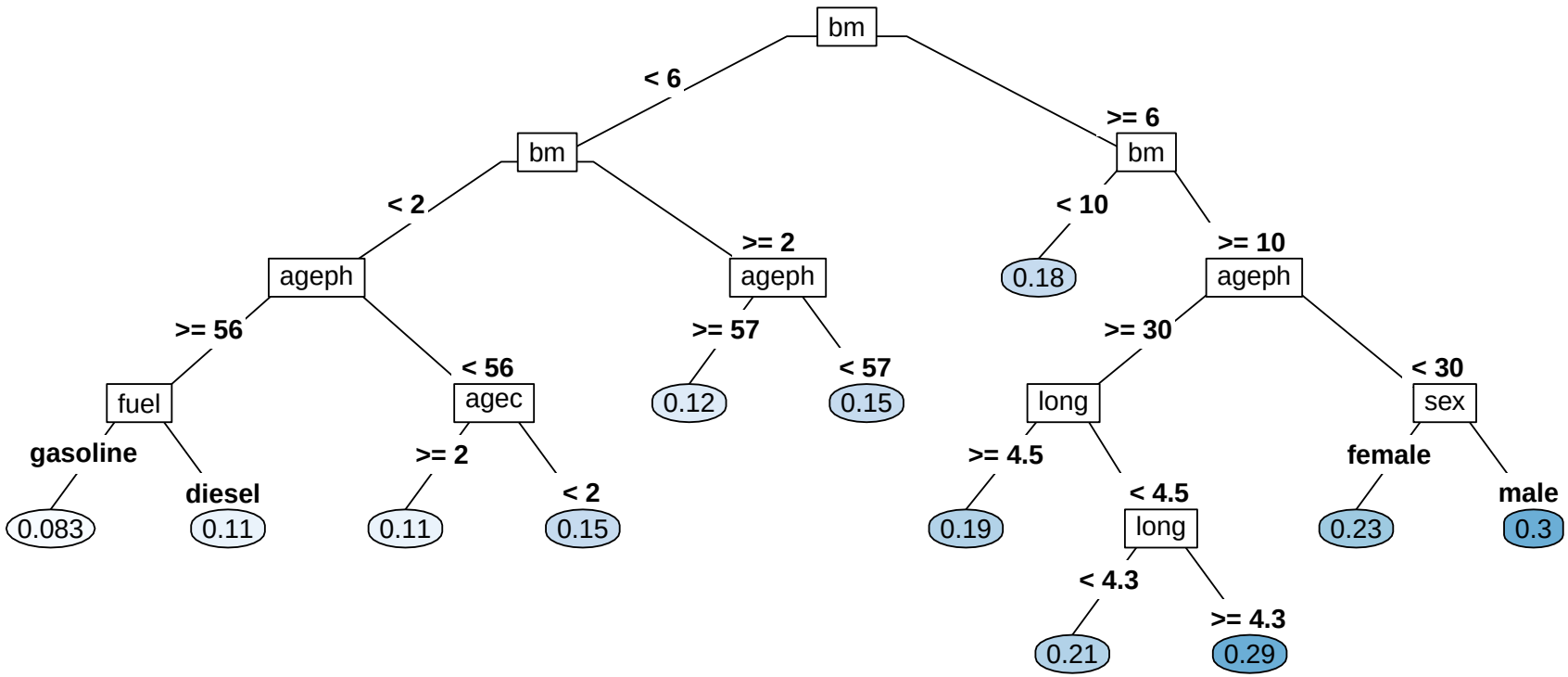
$$\sum_{j=1}^{J} \sum_{i\,:\,\boldsymbol{x}_i \in R_j} L(y_i, \hat{y}_{R_j}) \;+\; J \cdot cp \cdot \sum_{i\,:\,\boldsymbol{x}_i \in R} L(y_i, \hat{y}_R)$$

  - complexity parameter cp will determine the size of the tree
  - $cp = 0$ gives biggest possible tree
  - $cp = 1$ gives root tree without splits
  - cp is an important tuning parameter.

- We employ a tuning strategy and search grid to find the optimal value for cp, e.g. via cross-validation.

Regression trees

Example of a frequency tree with `rpart.plot`



Regression trees

Advantages and disadvantages of trees

Regression trees

Advantages and disadvantages of trees

+ Trees are very easy to explain.

Regression trees

Advantages and disadvantages of trees

+ Trees are very easy to explain.

+ Decision trees closely mirror human decision-making.

Regression trees

Advantages and disadvantages of trees

+ Trees are very easy to explain.

+ Decision trees closely mirror human decision-making.

+ Trees can be displayed graphically, easily interpreted by non-expert.

Regression trees

Advantages and disadvantages of trees

+ Trees are very easy to explain.
+ Decision trees closely mirror human decision-making.
+ Trees can be displayed graphically, easily interpreted by non-expert.
+ Trees can easily handle qualitative predictors.

Regression trees

Advantages and disadvantages of trees

+ Trees are very easy to explain.
+ Decision trees closely mirror human decision-making.
+ Trees can be displayed graphically, easily interpreted by non-expert.
+ Trees can easily handle qualitative predictors.
– Trees generally do not have same level of predictive accuracy as some other predictive modeling techniques discussed today.

Regression trees

Advantages and disadvantages of trees

- By aggregating many decision trees into ensembles of trees, the predictive performance can be substantially improved.
- These techniques are known as bagging, random forests and boosting (see further).

Bagging

- A natural way to reduce the variance and increase the prediction accuracy of a statistical learning method:

Bagging

- A natural way to reduce the variance and increase the prediction accuracy of a statistical learning method:
  - take many training sets from the population

Bagging

- A natural way to reduce the variance and increase the prediction accuracy of a statistical learning method:
  - take many training sets from the population
  - build a separate prediction model using each training set

Bagging

- A natural way to reduce the variance and increase the prediction accuracy of a statistical learning method:
  - take many training sets from the population
  - build a separate prediction model using each training set
  - average the resulting predictions.

Bagging

- A natural way to reduce the variance and increase the prediction accuracy of a statistical learning method:
  - take many training sets from the population
  - build a separate prediction model using each training set
  - average the resulting predictions.
- Thus, calculate $\hat{f}^1(x),\ \hat{f}^2(x), \ldots, \hat{f}^T(x)$ using B separate training sets and average them

$$\hat{f}_{\text{avg}}(x) \quad = \quad \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x).$$

Bagging

- A natural way to reduce the variance and increase the prediction accuracy of a statistical learning method:
  - take many training sets from the population
  - build a separate prediction model using each training set
  - average the resulting predictions.
- Thus, calculate $\hat{f}^1(x),\ \hat{f}^2(x), \ldots, \hat{f}^T(x)$ using B separate training sets and average them

$$\hat{f}_{\text{avg}}(x) \quad = \quad \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x).$$

However, usually we do not have access to multiple training sets.

Bagging

- Instead, we can bootstrap and take repeated samples from the (single) training data set.

Bagging

- Instead, we can bootstrap and take repeated samples from the (single) training data set.
- Thus, we generate B different bootstrapped training data sets.

Bagging

- Instead, we can bootstrap and take repeated samples from the (single) training data set.
- Thus, we generate B different bootstrapped training data sets.
- We train our method on the bth bootstrapped training set and get $\hat{f}^{\star b}(x)$. Finally, we average all the predictions

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{\star b}(x).$$

Bagging

- Instead, we can bootstrap and take repeated samples from the (single) training data set.
- Thus, we generate B different bootstrapped training data sets.
- We train our method on the bth bootstrapped training set and get $\hat{f}^{\star b}(x)$. Finally, we average all the predictions

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{\star b}(x).$$

This is called bagging and goes back to Breiman (1996). It mainly reduces the estimation variance.

Bagging

- Instead, we can bootstrap and take repeated samples from the (single) training data set.
- Thus, we generate B different bootstrapped training data sets.
- We train our method on the bth bootstrapped training set and get $\hat{f}^{\star b}(x)$. Finally, we average all the predictions

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{\star b}(x).$$

  This is called bagging and goes back to Breiman (1996). It mainly reduces the estimation variance.

- Bagging refers to Bootstrap Aggregating.

Bagging

- Bagging with regression trees:

Bagging

- Bagging with regression trees:
  - construct B regression trees using B bootstrapped training sets

Bagging

- Bagging with regression trees:
  - construct B regression trees using B bootstrapped training sets
  - these trees are grown deep, and are not pruned

Bagging

- Bagging with regression trees:
  - construct B regression trees using B bootstrapped training sets
  - these trees are grown deep, and are not pruned
  - we then average the resulting predictions.

Bagging

- Bagging with regression trees:
  - construct B regression trees using B bootstrapped training sets
  - these trees are grown deep, and are not pruned
  - we then average the resulting predictions.
- Bagging with classification trees:
  - for a given test observation, we record the class predicted by each of the B trees and take a majority vote.

Random forests

- Random forests (Breiman, 2001) provide an improvement over bagged trees by way of a small tweak that decorrelates the trees.

Random forests

- Random forests (Breiman, 2001) provide an improvement over bagged trees by way of a small tweak that decorrelates the trees.
- Random forests:

Random forests

- Random forests (Breiman, 2001) provide an improvement over bagged trees by way of a small tweak that decorrelates the trees.
- Random forests:
  - we build a number of decision trees on bootstrapped training samples

Random forests

- Random forests (Breiman, 2001) provide an improvement over bagged trees by way of a small tweak that decorrelates the trees.
- Random forests:
  - we build a number of decision trees on bootstrapped training samples
  - each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates (from the full set of p predictors)

Random forests

- Random forests (Breiman, 2001) provide an improvement over bagged trees by way of a small tweak that decorrelates the trees.
- Random forests:
  - we build a number of decision trees on bootstrapped training samples
  - each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates (from the full set of p predictors)
  - typically, $m \approx \sqrt{p}$.

Random forests

- Rationale:

Random forests

- Rationale:
  - suppose there is a very strong predictor in the data set

Random forests

- Rationale:
  - suppose there is a very strong predictor in the data set
  - in the collection of bagged trees, most or all of the trees will use this predictor as top split

Random forests

- Rationale:
  - suppose there is a very strong predictor in the data set
  - in the collection of bagged trees, most or all of the trees will use this predictor as top split
  - all bagged trees will look quite similar to each other

Random forests

- Rationale:
  - suppose there is a very strong predictor in the data set
  - in the collection of bagged trees, most or all of the trees will use this predictor as top split
  - all bagged trees will look quite similar to each other
  - predictions are highly correlated; no substantial reduction in variance over a single tree.

Random forests

- Rationale:
  - suppose there is a very strong predictor in the data set
  - in the collection of bagged trees, most or all of the trees will use this predictor as top split
  - all bagged trees will look quite similar to each other
  - predictions are highly correlated; no substantial reduction in variance over a single tree.
- Random forests overcome this problem.

Random forests

Details

- The inventors of random forests recommend:

Random forests

Details

- The inventors of random forests recommend:
  - for classification, default value for m is $\lfloor \sqrt{p} \rfloor$ and the minimum node size is one

Random forests

Details

- The inventors of random forests recommend:
  - for classification, default value for m is $\lfloor \sqrt{p} \rfloor$ and the minimum node size is one
  - for regression, default value for m is $\lfloor p/3 \rfloor$ and the minimum node size is five.

Random forests

Details

- The inventors of random forests recommend:
  - for classification, default value for m is $\lfloor \sqrt{p} \rfloor$ and the minimum node size is one
  - for regression, default value for m is $\lfloor p/3 \rfloor$ and the minimum node size is five.
- Instead of using defaults, better practice to use a tuning strategy and search grid.

Random forests

Details

- The inventors of random forests recommend:
  - for classification, default value for m is $\lfloor \sqrt{p} \rfloor$ and the minimum node size is one
  - for regression, default value for m is $\lfloor p/3 \rfloor$ and the minimum node size is five.
- Instead of using defaults, better practice to use a tuning strategy and search grid.
- Possible tuning parameters in a random forest:
  - number of trees B
  - number of sample variables m
  - minimum node size n_{min}

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.
- We focus on boosting with decision trees.

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.
- We focus on boosting with decision trees.
- Trees are grown sequentially; each tree is grown using information from previously grown trees.

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.
- We focus on boosting with decision trees.
- Trees are grown sequentially; each tree is grown using information from previously grown trees.
- Boosting is:
  - iterative method that combines many weak learners into one powerful prediction
  - one of the most powerful shallow machine learning techniques.

Boosting

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach learns slowly.

Boosting

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach learns slowly.
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well.

Boosting

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach learns slowly.
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well.
- The shrinkage parameter λ slows down the process even further, allowing more and different shaped trees to attack the residuals.

Boosting

- Boosting has three tuning parameters:

  1. the number of trees B; boosting can overfit if B is too large. We use cross-validation to select B;
  2. the shrinkage parameter λ, a small positive number, which controls the rate at which boosting learns. Typical values are 0.01 or 0.001;
  3. the number d of splits in each tree, which controls the complexity of the boosted ensemble. $d = 1$ uses stumps (a single split) and often works well. d is the interaction depth.

Boosting: gradient boosting method

- The algorithm *Boosting for regression trees* is one example of the generic gradient tree-boosting algorithm (Friedman, 2001) for regression.
- Available in R in the `gbm` package.
- The generic algorithm (see next page) reduces to Algorithm 3:
  - loss function is $\frac{1}{2}\left[y_i - f(x_i)\right]^2$;
  - minus the gradient $-\partial L(y_i, f(x_i))/\partial f(x_i)$ is $y_i - f(x_i)$.

Boosting

- Link with GLMs?
- When building trees for a count data set, `gbm` uses the negative Poisson likelihood as the loss function:

$$\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x})) \quad = \quad \exp\left(-\exp f(\boldsymbol{x}_i)\right) \cdot \frac{(\exp f(\boldsymbol{x}_i))^{y_i}}{y_i!}$$

  where `gbm` builds the following predictor: $\exp\left(f(\boldsymbol{x}_i)\right)$.

- The loss function used by `gbm` is then:

$$L(y_i, \exp\left(f(\boldsymbol{x}_i)\right)) \quad = \quad -2 \cdot w_i \left(y_i \cdot f(\boldsymbol{x}_i) - \exp f(\boldsymbol{x}_i)\right).$$

  See discussion on
  https://stats.stackexchange.com/questions/300674/understand-the-cost-function-in-the-gbm-implementation-of-r.

The R universe

What is out there?

Model	Poisson	Gamma
Generalized linear model	✓ `stats`	✓ `stats`
Generalized additive model	✓ `mgcv`	✓ `mgcv`
Regression tree	✓ `rpart`	✗ -
Random forest	✗ -	✗ -
Gradient boosting machine	✓ `gbm`	✓ harrysouthworth/`gbm`

The R universe

Filling the gaps

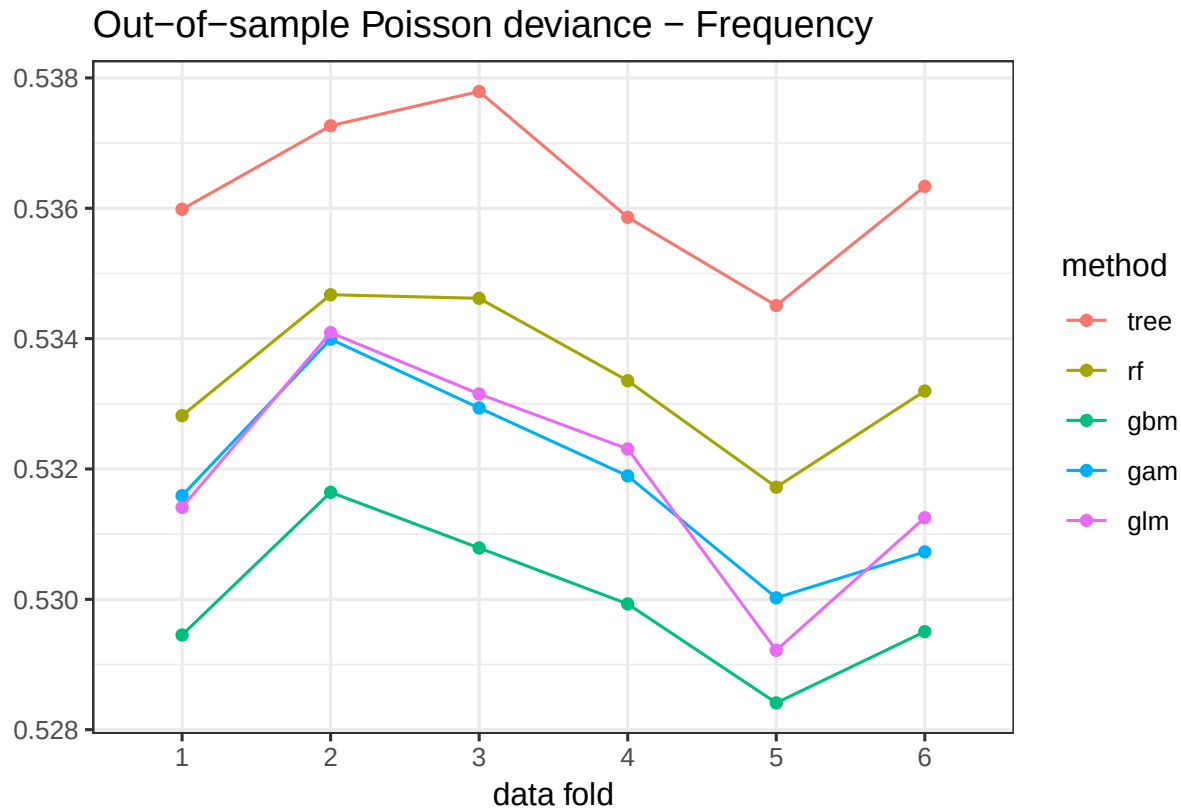
Model	Poisson	Gamma
Generalized linear model	✓ `stats`	✓ `stats`
Generalized additive model	✓ `mgcv`	✓ `mgcv`
Regression tree	✓ `rpart`	✓ `rpart*`
Random forest	✓ `rpart*`	✓ `rpart*`
Gradient boosting machine	✓ `gbm`	✓ harrysouthworth/`gbm`

Interpretation

- Classical statistical methods are highly interpretable:
  - coefficients in a GLM
  - splines in a GAM
- Not necessarily the case for machine learning techniques:
  - \+ regression trees
  - – bagged trees/random forests
  - – boosted trees
- There is a need for interpretation tools. This will be the topic of College 9 ('boosting your model - putting it all together").

Comparison of pricing models

Preview



More comparison tools will be covered in College 9 ('boosting your model - putting it all together").