CHAPTER 8

REGRESSION TREES AND RULE-BASED MODELS

APPLIED PREDICTIVE MODELING BY KUHN & JOHNSON
COLLATED BY PROF. CHING-SHIH (VINCE) TSOU (PH.D.)
CENTER FOR APPLICATIONS OF DATA SCIENCE (CADS)
GRADUATE INSTITUTE OF INFORMATION AND DECISION SCIENCES (GIIDS)
NATIONAL TAIPEI UNIVERSITY OF BUSINESS (NTUB)
CHINESE ACADEMY OF R SOFTWARE (CARS)
DATA SCIENCE & BUSINESS APPLICATIONS (DSBA) ASSOCIATION OF TAIWAN

AGENDA

- Introduction
- Basic Regression Trees
- Regression Model Trees
- Rule-Based Models
- Bagged Trees
- Random Forests
- Boosting
- Cubist
- Computing

INTRODUCTION – NESTED IF-THEN STATEMENTS

- Tree-based models consist of one or more nested if-then statements for the predictors that partition the data. Within these partitions, a model is used to predict the outcome.
- For example, a very simple tree could be defined as

```
if Predictor A >= 1.7 then
|   if Predictor B >= 202.1 then Outcome = 1.3
|   else Outcome = 5.6
else Outcome = 2.5
```


INTRODUCTION – PREDICTORS VALUES & MODEL FORMULA IN TERMINAL NODES

- In the terminology of tree models, there are two splits of the data into three terminal nodes or leaves of the tree.
- To obtain a prediction for a new sample, we would follow the **if-then** statements defined by the tree *using values of that sample's predictors until* we come to *a terminal node*.
- The *model formula in the terminal node* would then be used to generate the prediction.

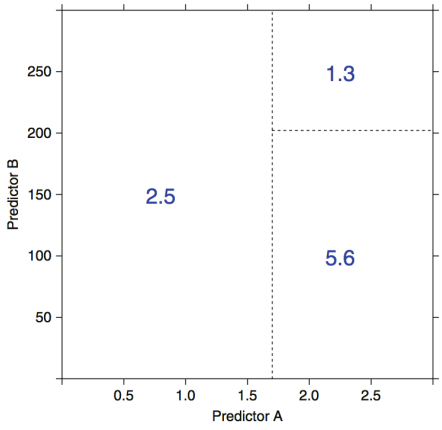
INTRODUCTION – RULES, RULE SET AND PRUNING

- Notice that the if-then statements generated by a tree define *a unique route* to one terminal node for any sample.
- A rule is *a set of if-then conditions* that have been collapsed into independent conditions.
- For the example above, there would be three rules:

```
if Predictor A >= 1.7 and Predictor B >= 202.1 then Outcome = 1.3
if Predictor A >= 1.7 and Predictor B <  202.1 then Outcome = 5.6
if Predictor A <  1.7 then Outcome = 2.5
```

- Rules can be *simplified or pruned* in a way that samples are covered by multiple rules.

INTRODUCTION – DIVIDE AND CONQUER ILLUSTRATION IN 2-D SPACE



- In this case, two-dimensional predictor space is cut into three regions, and, within each region, the outcome is predicted by a single number.
- This figure presents these rules in the predictor space.

INTRODUCTION – CHARACTERISTICS FOR REAL-LIFE MODELING

- *Tree-based* and *rule-based* models are popular modeling tools for a number of reasons.
- They generate a set of conditions that are *highly interpretable* and are *easy to implement*.
- Because of the logic of their construction, they can *effectively handle many types of predictors without the need to pre-process them*.
- These models can *effectively handle missing data* and *implicitly* conduct *feature selection*, characteristics that are desirable for many real-life modeling problems.

INTRODUCTION – WEAKNESS AND ENSEMBLE

- Models based on single trees or rules, however, do have particular weaknesses.
- Two well-known weaknesses are :
  - Model *instability* (i.e., slight changes in the data can drastically change the structure of the tree or rules and, hence, the interpretation).
  - *Less-than-optimal* predictive performance.
- If the relationship between predictors and the response *cannot* be adequately defined by *rectangular subspaces* of the predictors, then tree-based or rule-based models will have *larger prediction error* than other kinds of models.
- To combat these problems, researchers developed *ensemble methods* that combine many trees into one model.

BASIC REGRESSION TREES – THREE FACTORS

- Basic regression trees partition the data into smaller groups that are *more homogenous* with respect to the *response*.
- To achieve outcome homogeneity, regression trees determine:
  - The *predictor to split* on and value of the split.
  - The *depth or complexity* of the tree.
  - The *prediction equation* in the *terminal* nodes.

BASIC REGRESSION TREES - ALGORITHMS

- In this section, we focus on techniques where the model in the terminal nodes are *simple constants*.
- There are many techniques for constructing regression trees.
- One of the oldest and most utilized is the *classification and regression tree (CART)* methodology of Breiman et al. (1984).
- Categorical outcome: ID3 (categorical predictors), C4.5 (categorical predictors), CHAID (categorical predictors)
- Categorical/Numeric outcome: CART (categorical/numeric predictors)

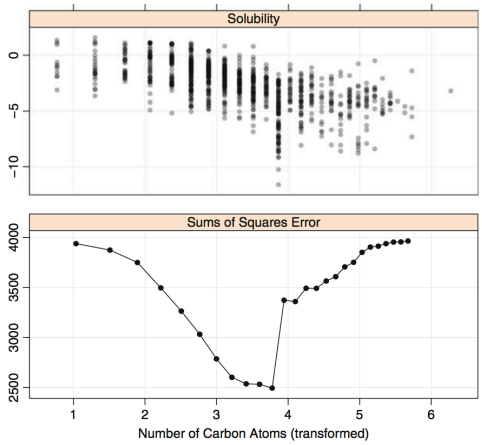
BASIC REGRESSION TREES – PARTITION AND OBJECTIVE

- For regression, the model begins with the entire data set, S, and searches every distinct value of every predictor to find the predictor and split value that *partitions the data into two groups* (S1 and S2) such that the overall sums of squares error are minimized:

$$\text{SSE} = \sum_{i \in S_1} (y_i - \bar{y}_1)^2 + \sum_{i \in S_2} (y_i - \bar{y}_2)^2$$

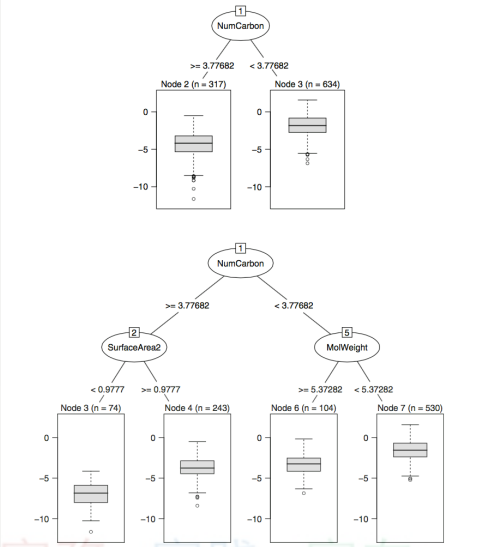
- where y1 bar and y2 bar are the averages of the training set outcomes within groups S_1 and S_2, respectively.
- Then within each of groups S_1 and S_2, this method searches for the predictor and split value that *best reduces SSE*.

BASIC REGRESSION TREES – OPTIMAL SPLIT



- Using the regression tree approach, the optimal split point for this variable is *3.78*.
- The reduction in the SSE associated with *this split is compared to the optimal values for all of the other predictors* and the split corresponding to the absolute minimum error is used to form subsets S_1 and S_2.

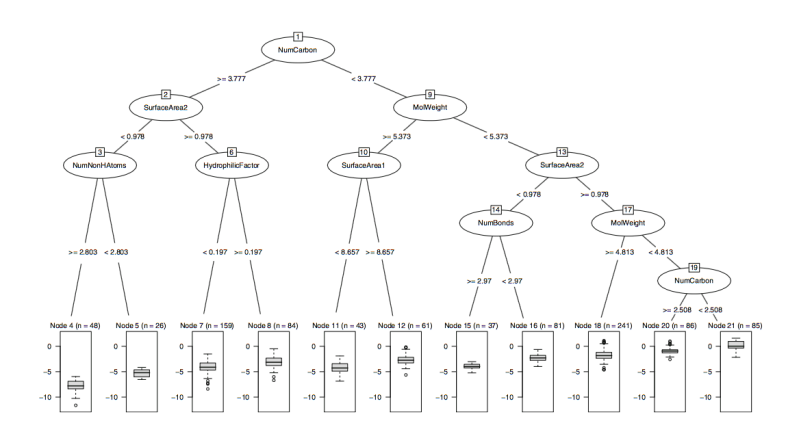
BASIC REGRESSION TREES – TREE GROWING STEP



- If the process were stopped at this point, all sample with values for this predictor less than 3.78 would be predicted to be −1.84 and samples above the splits all have a predicted value of −4.49:

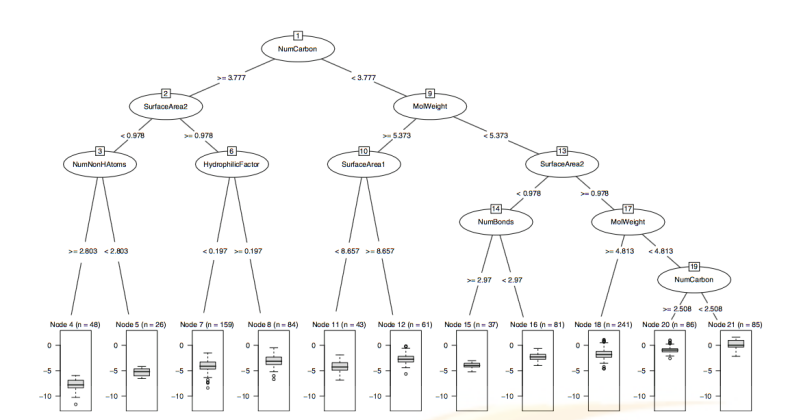
```
if the number of carbon atoms >= 3.78 then  Solubility = -4.49
else Solubility = -1.84
```

BASIC REGRESSION TREES – MINIMUM NUMBER OF SAMPLE



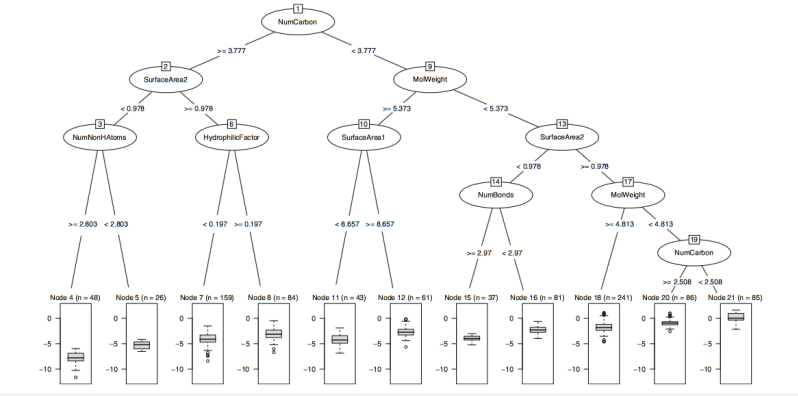
- In practice, the process then continues within sets S_1 and S_2 until the *number of samples* in the splits falls *below some threshold* (such as 20 samples).
- This would conclude the *tree growing step*.

BASIC REGRESSION TREES – COST-COMPLEXITY TUNING & 1-SE RULE



- Using the one-standard-error rule, the regression tree built on the solubility data had 11 terminal nodes (c_p= 0.01) and the cross-validation estimate of the RMSE was 1.05.

BASIC REGRESSION TREES – IMPORTANT PREDICTOR AND ITS REUSE



- All of the splits retained in the model involve the continuous or count predictors and several paths through the tree *use some of the same predictors more than once.*

BASIC REGRESSION TREES – PENALIZED METHODS AGAIN

- Once the full tree has been grown, the tree may be very large and is likely to over-fit the training set.
- The tree is then *pruned* back to a potentially *smaller depth*.
- The processed used by Breiman et al. (1984) is *cost–complexity tuning*.
- The goal of this process is to find a "right-sized tree" that has the smallest error rate. To do this, we penalize the error rate using the size of the tree:

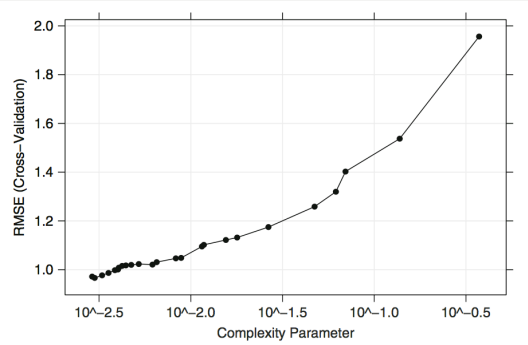
$$\text{SSE}_{c_p} = \text{SSE} + c_p \times (\#\ \text{Terminal Nodes})$$

- where c_p is called the complexity parameter. For a specific value of the complexity parameter, we find the smallest pruned tree that has the lowest penalized error rate.

BASIC REGRESSION TREES

- As with other regularization methods previously discussed, smaller penalties tend to produce *more complex models*, which, in this case, result in *larger trees*.
- *Larger values* of the complexity parameter may result in a tree with *one split* or, perhaps, even *a tree with no splits*.
- The latter result would indicate that no predictor adequately explains enough of the variation in the outcome at the chosen value of the complexity parameter.

BASIC REGRESSION TREES



- The model can be tuned by choosing the value of the complexity parameter associated with the smallest possible RMSE value.
- The cross-validation profile is shown in this figure.
- In this case, the tuning process chose a larger tree with a c_p value of 0.003 and 25 terminal nodes.
- The estimated RMSE from this model was 0.97.

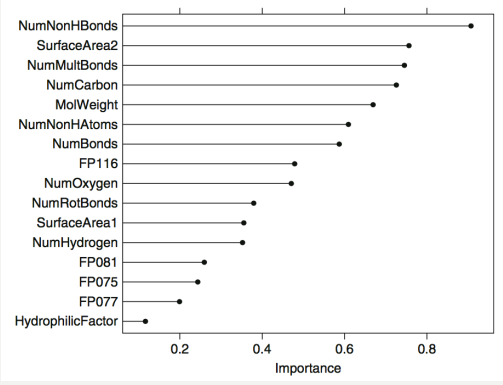
BASIC REGRESSION TREES – MISSING DATA AND SURROGATE SPLIT

- When building the tree, missing data are ignored.
- For each split, a variety of alternatives are evaluated.
- A surrogate split is one whose results are similar to the original split actually used in the tree.
- If a surrogate split approximates the original split well, it can be used when the predictor data associated with the original split are not available. (i.e. it is missing.)
- In practice, several surrogate splits may be saved for any particular split in the tree.

BASIC REGRESSION TREES - RELATIVE IMPORTANCE OF THE PREDICTORS

- Once the tree has been finalized, we begin to assess the relative importance of the predictors to the outcome.
- One way to compute an aggregate measure of importance is to keep track of the overall reduction in the optimization criteria for each predictor (Breiman et al. 1984).
- An advantage of tree-based models is that, when the tree is *not large*, the model is *simple and interpretable*.
- This type of tree can be computed quickly (despite using multiple exhaustive searches).

BASIC REGRESSION TREES



- If SSE is the optimization criteria, then the reduction in the SSE for the training set is aggregated for each predictor.
- Intuitively, predictors that appear higher in the tree or those that appear multiple times in the tree will be more important than predictors that occur lower in the tree or not at all.
- This figure shows the importance values for the 16 predictors in the more complex final solubility model.

BASIC REGRESSION TREES – FEATURE SELECTION

- Tree models intrinsically conduct feature selection.
- If a predictor is never used in a split, the prediction equation is independent of these data.
- This advantage is weakened when there are highly correlated predictors. (collinear data)
- It is possible that the small difference between these predictors is strongly driving the choice between the two, but it is more likely to be due to small, random differences in the variables.
- Because of this, *more predictors may be selected than actually needed*. (collinearity issue again!)

BASIC REGRESSION TREES

- In addition, the variable importance values are affected.
- If the solubility data *only contained one of the surface area predictors*, then this predictor would have likely been *used twice* in the tree, therefore *inflating its importance value*.
- Instead, including *both surface area predictors* in the data causes their importance to have *only moderate values*.

BASIC REGRESSION TREES - WEAKNESS

- While trees are highly interpretable and easy to compute, they do have some noteworthy disadvantages.
- First, single regression trees are more likely to have sub-optimal predictive performance compared to other modeling approaches.
- By construction, tree models partition the data into rectangular regions of the predictor space.

BASIC REGRESSION TREES

- If the relationship between predictors and the outcome is not adequately described by these rectangles, then the predictive performance of a tree will not be optimal.
- The number of possible predicted outcomes from a tree is finite and is determined by the number of terminal nodes.
- For the solubility data, the optimal tree has 11 terminal nodes and consequently can only produce 11 possible predicted values.

BASIC REGRESSION TREES

- However, that the training set data falling within this path of the tree vary across several log units of data.
- If new data points are consistent with the training data, many of the new samples falling along this path will not be predicted with a high degree of accuracy.
- If the data are slightly altered, *a completely different set of splits* might be found. (unstability again!)
- While this is a disadvantage, *ensemble methods* exploit this characteristic to create models that tend to have *extremely good* performance.

BASIC REGRESSION TREES – SELECTION BIAS

- Finally, these trees suffer from selection bias: predictors with a higher number of *distinct values* are favored over *more granular* predictors (Loh and Shih 1997; Carolin et al. 2007; Loh 2010). Loh and Shih (1997) remarked that

"The danger occurs when a data set consists of a mix of informative and noise variables, and the noise variables have many more splits than the informative variables. Then there is a high probability that the noise variables will be chosen to split the top nodes of the tree. Pruning will produce either a tree with misleading structure or no tree at all."

- Also, as the number of missing values increases, the selection of predictors becomes more biased (Carolin et al. 2007).

BASIC REGRESSION TREES – UNBIASED GUIDE

- There are several *unbiased* regression tree techniques.
- For example, Loh (2002) proposed the generalized, unbiased, interaction detection and estimation (*GUIDE*) algorithm which solves the problem by decoupling the process of selecting the split variable and the split value.
- This algorithm *ranks* the predictors using *statistical hypothesis testing* and then finds the *appropriate split value* associated with *the most important factor*.

BASIC REGRESSION TREES – UNBIASED CONDITIONAL TREE

- Another approach is *conditional inference trees* of Hothorn et al. (2006).
- They describe a unified framework for *unbiased* tree-based models for *regression*, *classification*, and *other scenarios*.
- In this model, *statistical hypothesis tests* are used to do an *exhaustive search* across the *predictors* and their *possible split points*.
- For a candidate split, a statistical test is used to evaluate the *difference between the means of the two groups* created by the split and a *p*-value can be computed for the test.

BASIC REGRESSION TREES

- Utilizing the test statistic p-value has several advantages.
  - First, predictors that are on disparate scales can be compared since the p-values are on the same scale.
  - Second, multiple comparison corrections (Westfall and Young 1993) can be applied to the raw *p*-values within a predictor to reduce the bias resulting from a large number of split candidates.
- These corrections attempt to reduce the number of false-positive test results that are incurred by conducting a large number of statistical hypothesis tests.
- Thus, predictors are increasingly penalized by multiple comparison procedures as the number of splits (and associated *p*-values) increases.

BASIC REGRESSION TREES – ABOUT PRUNING

- This algorithm does *not use pruning*.
- As the data sets are further split, the *decrease in the number of samples reduces* the *power of the hypothesis tests*.
- This results in higher p-values and a lower likelihood of a new split.
- However, *statistical hypothesis tests are not directly related to predictive performance.*
- Because of this, it is still advisable to choose the complexity of the tree on the basis of performance.

REGRESSION MODEL TREES – LIMITATION OF REGRESSION TREES AND MODEL TREE

- One limitation of simple regression trees is that each terminal node uses the *average* of the training set outcomes in that node for prediction.
- As a consequence, these models may not do a good job predicting samples whose true outcomes are extremely high or low.
- One approach to dealing with this issue is to use a different estimator in the terminal nodes. Here we focus on the *model tree* approach described in Quinlan (1992) called M5, which is similar to regression trees except:
  - The splitting criterion is different.
  - The terminal nodes predict the outcome using *a linear model* (as opposed to the simple average).
  - When a sample is predicted, it is often *a combination of the predictions* from *different models along the same path through the tree*.

REGRESSION MODEL TREES – STD. DEV. INSTEAD OF SSE FOR TREE GROWING

- The initial split is found using an exhaustive search over the predictors and training set samples, but, unlike those models, the expected reduction in the node's error rate is used.
- Let S denote the entire set of data and let $S_1, \ldots, S_p$ represent the P subsets of the data after splitting.
- The split criterion would be

$$\text{reduction} = \text{SD}(S) - \sum_{i=1}^{P} \frac{n_i}{n} \times \text{SD}(S_i)$$

- where SD is the *standard deviation* and n_i is the number of samples in partition i.

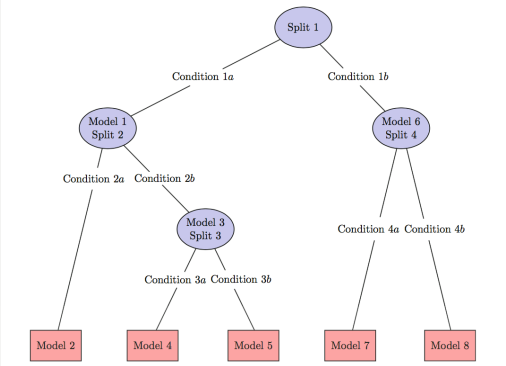
REGRESSION MODEL TREES

- This metric determines if the total variation in the splits, weighted by sample size, is lower than in the presplit data.
- This scheme is *similar to the methodology for classification trees discussed in Quinlan* (1993b).
- The split that is associated with the largest reduction in error is chosen and a linear model is created within the partitions using the split variable in the model.
- For subsequent splitting iterations, this process is repeated: an initial split is determined and a linear model is created for the partition using the current split variable and all others that preceded it.

REGRESSION MODEL TREES

- The error associated with each linear model is used in place of SD(*S*) in Eq. 8.2 to determine the expected reduction in the error rate for the next split.
- The tree growing process continues along the branches of the tree until there are no further improvements in the error rate or there are not enough samples to continue the process.
- Once the tree is fully grown, there is a linear model for every node in the tree.

REGRESSION MODEL TREES – LINEAR MODELS FITTING



- This figure shows an example of a model tree with *four splits* and *eight linear regression models*.
- *Model 5*, for instance, would be created using all the predictors that were in *splits 1–3* and with the *training set data points satisfying conditions 1a, 2b, and 3b*.

REGRESSION MODEL TREES – SIMPLIFY EACH MODEL BY WEIGHTED ABS. RESIDUALS

- Once the complete set of linear models have been created, each undergoes a simplification procedure to potentially drop some of the terms.
- For a given model, an adjusted error rate is computed.
- First, the absolute differences between the observed and predicted data are calculated then multiplied by a term that penalizes models with large numbers of parameters:

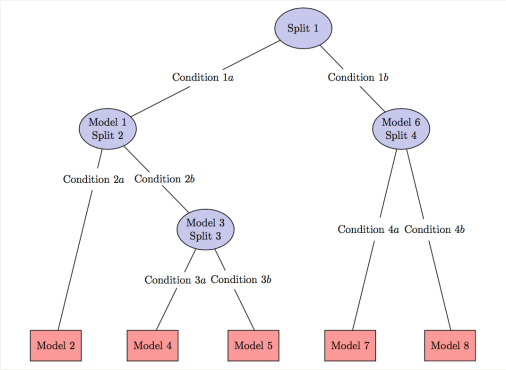
$$\text{Adjusted Error Rate} = \frac{n^* + p}{n^* - p} \sum_{i=1}^{n^*} |y_i - \hat{y}_i|$$

- where n^* is the number of training set data points that were used to build the model and p is the number of parameters.

REGRESSION MODEL TREES

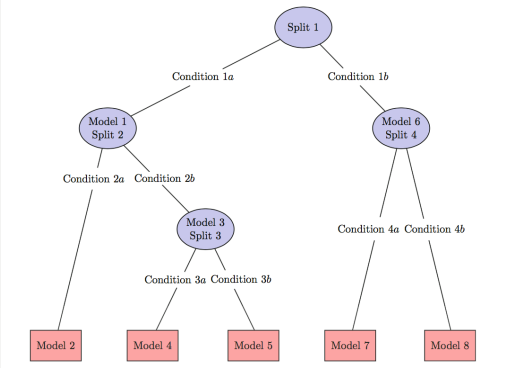
- Each model term is dropped and the adjusted error rate is computed.
- *Terms are dropped* from the model *as long as the adjusted error rate decreases*.
- In some cases, the linear model may be simplified to having only an intercept. This procedure is independently applied to each linear model.

REGRESSION MODEL TREES – SMOOTHING TO PREVENT OVERFITTING



- Model trees also incorporate a type of *smoothing* to decrease the potential for over-fitting. The technique is based on the "*recursive shrinking*" methodology of Hastie and Pregibon (1990).
- When *predicting*, the *new sample* goes *down the appropriate path* of the tree, and *moving from the bottom up*, the *linear models along that path are combined*.
- The tree generates a prediction for this sample *using Model 5* as well as the linear model in the parent node (*Model 3* in this case).

REGRESSION MODEL TREES

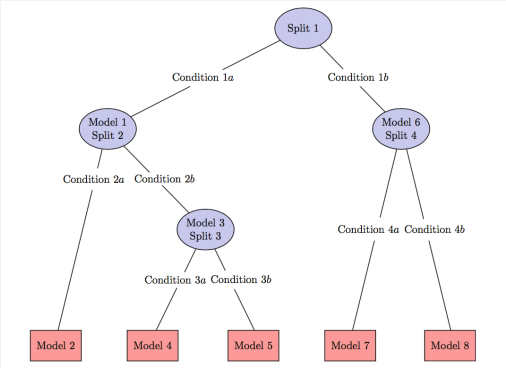


- These two predictions are combined using

$$\hat{y}_{(p)} = \frac{n_{(k)}\,\hat{y}_{(k)} + c\,\hat{y}_{(p)}}{n_{(k)} + c}$$

- where $\hat{y}_{(k)}$ is the prediction from the *child node* (Model 5), n(k) is the number of training set data points in the child node,
- $\hat{y}_{(p)}$ is the prediction from the *parent node*, and c is a constant with a default value of 15.

REGRESSION MODEL TREES



- Once this combined prediction is calculated, it is similarly combined with the next model along the tree (*Model 1*) and so on.
- For our example, the new sample falling under conditions 1a, 2b, and 3b would use *a combination of three linear models*. Note that the smoothing equation is a relatively simple linear combination of models.

REGRESSION MODEL TREES

- This type of smoothing can have a significant positive effect on the model tree when the linear models across nodes are very different.
- There are several possible reasons that the linear models may produce very different predictions.
  - Firstly, the number of training set samples that are available in a node will decrease as new splits are added.
    - This can lead to nodes which model very different regions of the training set and, thus, produce very different linear models.
    - This is especially true for small training sets.
  - Secondly, the linear models derived by the splitting process may suffer from significant collinearity.
    - Suppose two predictors in the training set have an extremely high correlation with one another.

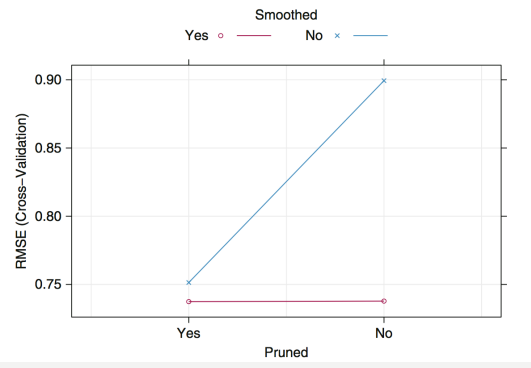
REGRESSION MODEL TREES

- In this case, the algorithm may choose between the two predictors randomly.
- If both predictors are eventually used in splits and become candidates for the linear models, there would be two terms in the linear model for effectively one piece of information.
- As discussed in previous chapters, this can lead to substantial instability in the model coefficients.
- Smoothing using several models can help reduce the impact of any unstable linear models.

REGRESSION MODEL TREES

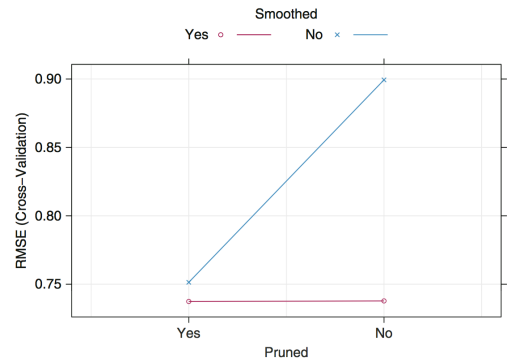
- Once the tree is fully grown, it is pruned back by finding inadequate sub-trees and removing them. Starting at the terminal nodes, the adjusted error rate with and without the sub-tree is computed.
- If the sub-tree does not decrease the adjusted error rate, it is pruned from the model. This process is continued until no more sub-trees can be removed.
- Model trees were built on the solubility data under the conditions of with and without pruning and with and without smoothing.

REGRESSION MODEL TREES – SMOOTHING EFFECT ON PRUNED OR UNPRUNED



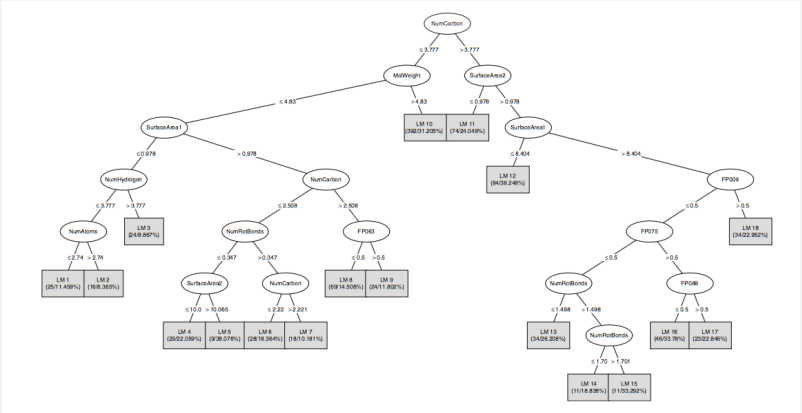
- This figure shows a plot of the cross-validation profiles for these data.
- The unpruned tree has 159 paths through the tree, which may over-fit the training data.
- When the tree is not pruned, model smoothing significantly improves the error rate.

REGRESSION MODEL TREES



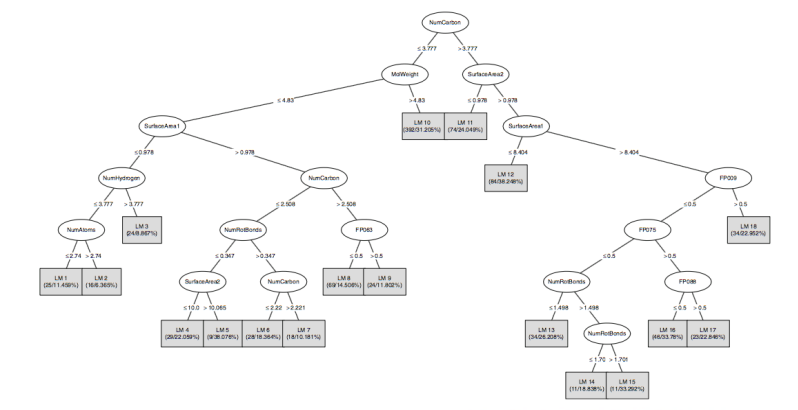
- For these data, the effect of pruning on the model was also substantial: the number of paths through the tree dropped from 159 to 18.
- For the pruned trees, smoothing produced a slight gain in performance, and, as a result, the optimal model used pruning and smoothing.

REGRESSION MODEL TREES



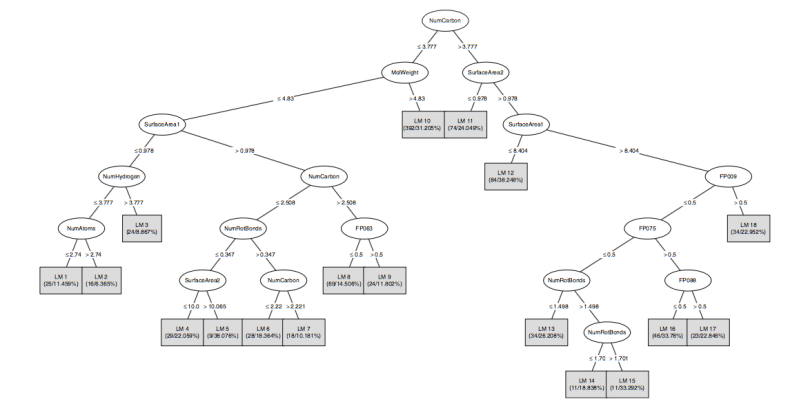
- The resulting model tree shows that many of the splits involve the same predictors, such as the number of carbons.

REGRESSION MODEL TREES



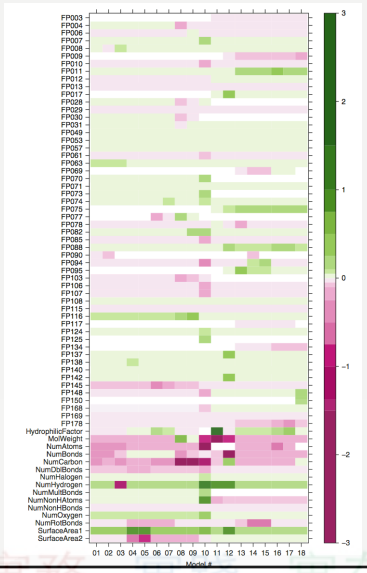
- The splits tend to favor the *continuous predictors* instead of the *fingerprints*.

REGRESSION MODEL TREES



- Splits based on the SSE and the error rate reduction produce almost identical results.

REGRESSION MODEL TREES – HEATMAP FOR COEFFICIENTS IN 18 MODELS



- The details of the linear models are shown in this figure (the model coefficients have been normalized to be on the same scale).
- We can see from this figure that the majority of models use many predictors, including a large number of the fingerprints.
- However, the *coefficients of the fingerprints* are *small relative* to the *continuous predictors*.
- Many of the models that are shown in this figure have *opposite signs for these two variables*. (SurfaceArea1 and SurfaceArea2)

REGRESSION MODEL TREES

- Additionally, this model can be used to demonstrate issues with collinearity.
- In the figure 8.10, linear model 5 (in the lower left of the tree) is associated with the following conditions:

```
NumCarbon <= 3.777 &
MolWeight <= 4.83 &
SurfaceArea1 > 0.978 &
NumCarbon <= 2.508 &
NumRotBonds <= 0.347 &
SurfaceArea2 >  10.065
```

REGRESSION MODEL TREES

- After model reduction and smoothing, there were *57 coefficients* in the corresponding linear model, including both surface area predictors.
- In the training set, these two predictors are highly correlated (0.96).
- We would expect severe collinearity as a result.
- The two scaled coefficients for these predictors are almost complete opposites: *0.9 for SurfaceArea1* and *−0.8 for SurfaceArea2*.
- Since the two predictors are almost identical, there is a contradiction: increasing the surface area equally increases and decreases the solubility.

REGRESSION MODEL TREES - CONCLUSION

- Despite this, the performance for this model is fairly competitive; *smoothing the models has the effect of minimizing the collinearity issues*.
- Removing the correlated predictors would produce a model that has *less in-consistencies and is more interpretable.*
- *However, there is a measurable drop in performance* by using the strategy.

RULE-BASED MODELS – FROM TREE TO RULES

- A rule is defined as *a distinct path through a tree*. *(Regression or Model Trees)*
- Consider the model tree shown in the last section and the path to get to linear model 15 in the lower right of figure 8.10:

```
NumCarbon >  3.777 &
SurfaceArea2 >  0.978 &
SurfaceArea1 >  8.404 &
FP009 <= 0.5 &
FP075 <= 0.5 &
NumRotBonds >  1.498 &
NumRotBonds >  1.701
```

- For the model tree shown in figure 8.10, there are a total of *18 rules*. (*18 terminal models*)
- For the tree, a new sample can only travel down a single path through the tree defined by these rules.
- The number of samples affected by a rule is called its *coverage*.

RULE-BASED MODELS – RULES PRUNING FOR FURTHER SIMPLIFY MODEL TREE

- In addition to the pruning algorithms described in the last section, the complexity of the model tree can be further reduced by *either removing entire rules* (in the entire rule set) or *removing some of the conditions* that define the rule (just simplify one rule).
- In the previous rule, note that the *number of rotatable bonds is used twice*.
- This occurred because another path through the tree determined that modeling the data subset where the number of rotatable bonds is *between 1.498 and 1.701* was important.
- However, when viewed in isolation, the rule above is *unnecessarily complex* because of this redundancy.
- Also, it may be *advantageous to remove other conditions* in the rule because they do not contribute much to the model.

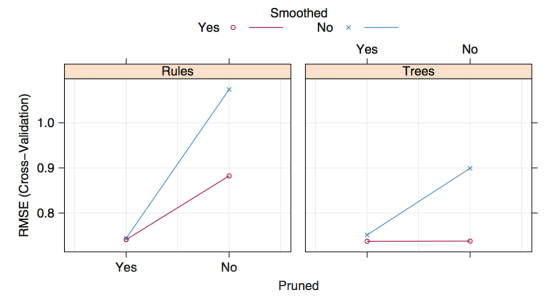
RULE-BASED MODELS – ALGORITHMS TO GENERATE RULES FROM TREE(S)

- Quinlan (1993b) describes methodologies for simplifying the *rules generated from classification trees*.
- Similar techniques can be applied to *model trees* to create a more simplistic set of rules from an initial model tree.
- Another approach to creating rules from model trees is outlined in Holmes et al. (1993) that uses the "*separate and conquer*" strategy.

RULE-BASED MODELS

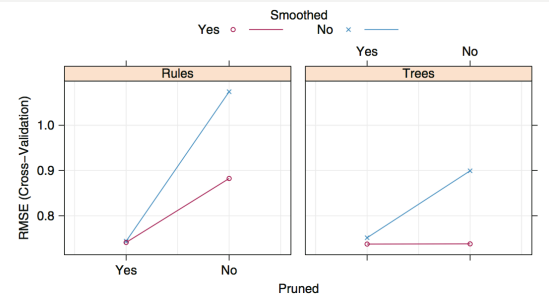
- This procedure derives rules from many different model trees instead of from a single tree.
  - First, an initial model tree (*we can have many rules, but just pick the most coverage one.*) is created (they recommend using unsmoothed model trees).
    - However, only the rule with the *largest coverage* is saved from this model.
    - The samples covered by the rule are *removed from the training set* and another model tree is *created with the remaining data*.
  - Again, only the rule with *the maximum coverage* is retained.
    - This process repeats until all the training set data have been *covered by at least one rule*.
    - A new sample is predicted by determining which rule(s) it falls under then applies the linear model associated with the largest coverage.

RULE-BASED MODELS - COMPARISON FOR PRUNING, SMOOTHING, RULES/TREES



- For the solubility data, a rule-based model was evaluated. Similar to the model tree tuning process, *four models* were fit using all combinations *for pruning and smoothing*.
- The *same resampling data sets* were used in the model tree analysis, so *direct comparisons* can be made.

RULE-BASED MODELS



- The right panel is the same as Fig. 8.9 (page 187) while the *left panel* shows the results when the *model trees are converted to rules*.
- For these data, when *smoothing and pruning are used*, the *model tree and rule-based version had equivalent error rates*.
- As with the model trees, pruning had a large effect on the model and *smoothing had a larger impact on the unpruned models*.

RULE-BASED MODELS

- The *best* fitting *model tree* was associated with a cross-validated RMSE of 0.737*.
- The *best rule-based model* resulted in an RMSE value of 0.741.
- Based on this alone, the model tree would be used for prediction. However, for illustration, the rule-based model will be examined in more detail.

RULE-BASED MODELS

- In all, nine rules were used to model these data, although the final rule has no associated conditions.
- The conditions for the rules are

```
Rule 1: NumCarbon <= 3.777 & MolWeight > 4.83
Rule 2: NumCarbon > 2.999
Rule 3: SurfaceArea1 > 0.978 & NumCarbon > 2.508 & NumRotBonds > 0.896
Rule 4: SurfaceArea1 > 0.978 & MolWeight <= 4.612 & FP063 <= 0.5
Rule 5: SurfaceArea1 > 0.978 & MolWeight <= 4.612
Rule 6: SurfaceArea1 <= 4.159 & NumHydrogen <= 3.414
Rule 7: SurfaceArea1 > 2.241 & FP046 <= 0.5 & NumBonds > 2.74
Rule 8: NumHydrogen <= 3.414
```

RULE-BASED MODELS – REDUNDANT CONDITIONS

- Looking back at the full model tree in Fig.8.10, the rule corresponding to Model 10 has the largest coverage using the conditions ***NumCarbon*** ≥ 3.77 and **MolWeight** > 4.83.
- This rule was preserved as the first rule in the new model.
- The next model tree was created using the remaining samples.
- Here, the rule with the largest coverage has a condition similar to the previous rule: ***NumCarbon*** > 2.99.
- In this case, *a sample with* ***NumCarbon*** > 2.99 *would be covered by at least two rules*.

RULE-BASED MODELS – REPEATED PREDICTORS & CONCLUSIONS

- The other rules used *many* of the *same predictors*:
  - **SurfaceArea1** (*five* times), **MolWeight** (*three* times), and **NumCarbon** (also *three* times).
- Figure 8.13 (*refer to the heatmap on p.193*) shows the coefficients of the linear models for each rule (similar to Fig. 8.11 for the full model tree).
- Here, the linear models are *more sparse*; the *number of terms in the linear models decreases as more rules* are created.
- This *makes sense* because there are *fewer data points to construct deep trees*.

BAGGED TREES – BOOTSTRAPPING + AGGREGATING REGRESSION MODELS

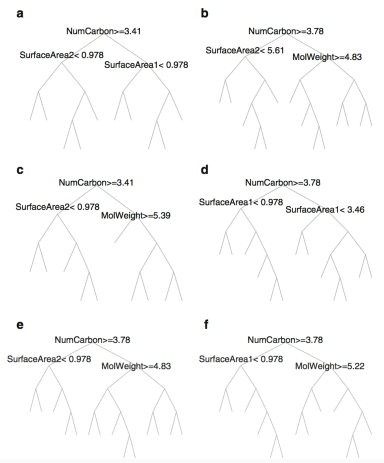
- Bagging, short for *bootstrap aggregation (aggregating)*, was originally proposed by Leo Breiman and was one of the earliest developed ensemble techniques (Breiman 1996a).
- Bagging is a general approach that uses *bootstrapping* in conjunction with *any regression model* to construct an ensemble.
- The method is fairly simple in structure and consists of the steps in Algorithm 8.1.

```
for i = 1 to m do
    Generate a bootstrap sample of the original data
    Train an unpruned tree model on this sample
end
```


BAGGED TREES

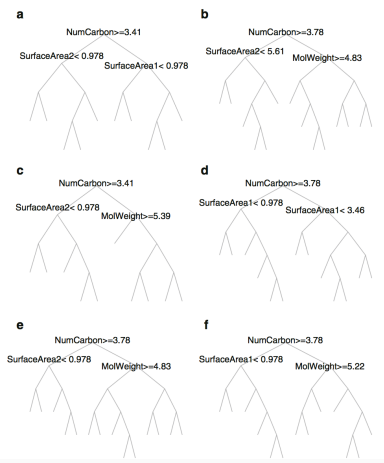
- Each model in the ensemble is then used to generate a prediction for a new sample and these m predictions are averaged to give the bagged model's prediction.
- Bagging models provide several advantages over models that are not bagged.
  - First, bagging effectively reduces the variance of a prediction through its aggregation process.
- For models that produce an unstable prediction, like regression trees, aggregating over many versions of the training data actually reduces the variance in the prediction and, hence, makes the prediction more stable.

BAGGED TREES – AVERAGING AND LOW VARIANCE



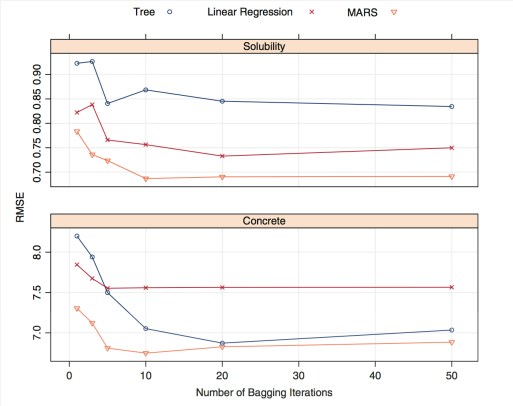
- In this example, six bootstrap samples of the solubility data were generated and a tree of maximum depth was built for each sample.
- These trees *vary in structure*, and hence the prediction for samples will vary from tree to tree.
- When the predictions for a sample are *averaged across all of the single trees*, the average prediction has *lower variance* than the variance across the individual predictions.

BAGGED TREES



- This means that if we were to generate a different sequence of bootstrap samples, build a model on each of the bootstrap samples, and average the predictions across models.
- Then we would likely get a very similar predicted value for the selected sample as with the previous bagging model.
- This characteristic also improves the predictive performance of a bagged model over a model that is not bagged.
- If the goal of the modeling effort is to find the best prediction, then bagging has a distinct advantage.

BAGGED TREES - CAVEAT

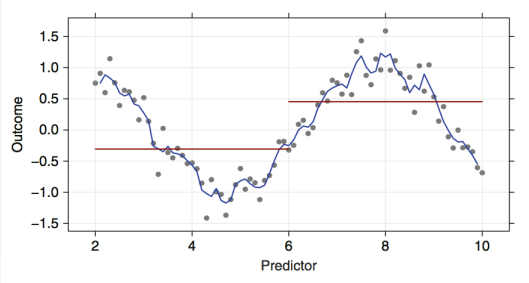


- *Bagging stable, lower variance models* like linear regression and MARS, on the other hand, offers *less improvement* in predictive performance.
- For each set of data, the test set performance based on RMSE is plotted by *number of bagging iterations*.
- For the solubility data, the decrease in RMSE across iterations is similar for trees, linear regression, and MARS, which is not a typical result.

BAGGED TREES

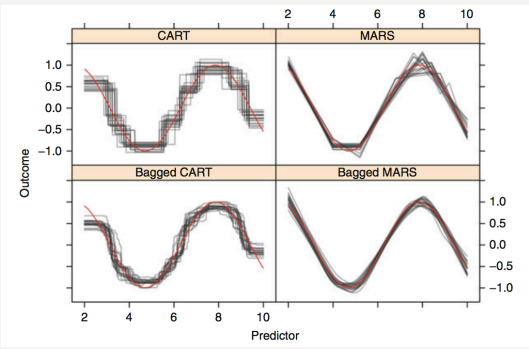
- This suggests that either the model predictions from linear regression and MARS have some inherent instability for these data which can be improved using a bagged ensemble or that trees are less effective at modeling the data.
- Bagging results for the concrete data are more typical, in which linear regression and MARS are least improved through the ensemble, while the predictions for *regression trees* are *dramatically improved*.

BAGGED TREES – CART VS. MARS



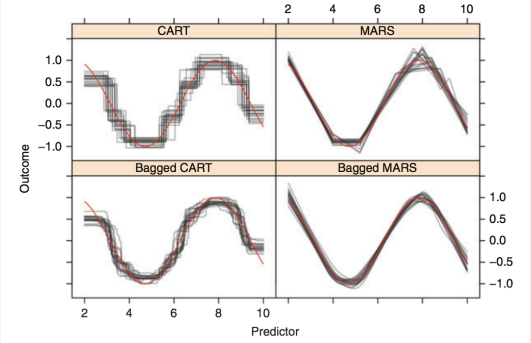
- As a further demonstration of bagging's ability to reduce the variance of a model's prediction, consider the simulated sin wave in this figure (figure 5.2).
- The red lines in the panels show the true trend while the multiple black lines show the predictions for each model.
- Note that the CART panel has more noise around the true *sin* curve than the MARS model, which only shows variation at the change points of the pattern.

BAGGED TREES



- This illustrates the high variance in the regression tree due to model instability.
- The bottom panels of the figure show the results for 20 bagged regression trees and MARS models.
- The variation around the true curve is greatly reduced for regression trees, and, for MARS.

BAGGED TREES



- The variation is only reduced around the curvilinear portions on the pattern.
- Using a simulated test set for each model, the average reduction in RMSE by bagging the tree was 8.6 % while the more stable MARS model had a corresponding reduction of 2 %

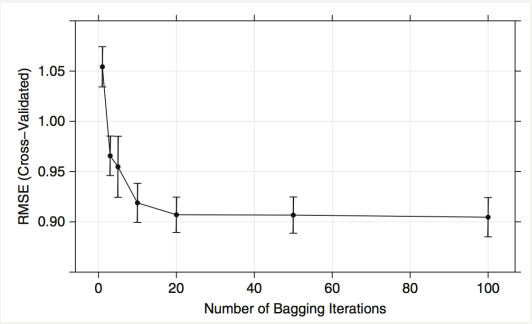
BAGGED TREES – OOB SAMPLES FOR ESTIMATION

- Another advantage of bagging models is that they can provide their own internal estimate of predictive performance that *correlates well* with either cross-validation estimates or test set estimates.
- Here's why: when constructing a bootstrap sample for each model in the ensemble, certain samples are left out.
- Every model in the ensemble generates a measure of predictive performance courtesy of the *out-of-bag samples*.

BAGGED TREES

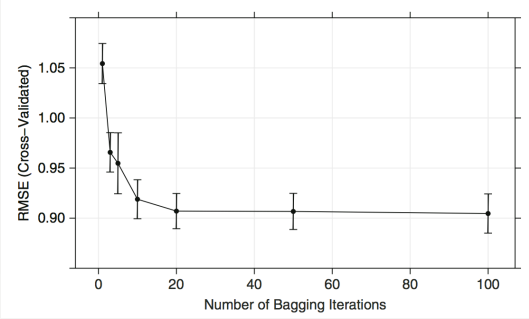
- The *average* of the out-of-bag performance metrics can then be used to gauge the predictive performance of the entire ensemble.
- And this value usually correlates well with the assessment of predictive performance we can get with either cross-validation or from a test set.
- This error estimate is usually referred to as the out-of-bag estimate.

BAGGED TREES – PARAMETER TUNING



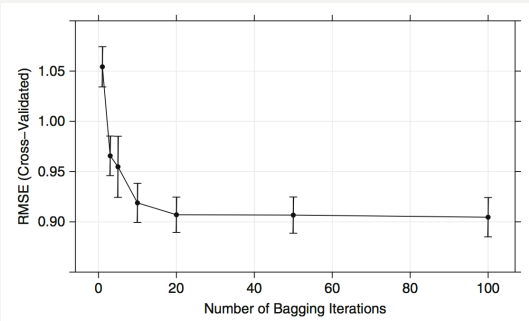
- In its basic form, the user has one choice to make for bagging: the *number of bootstrap samples to aggregate*, *m*.
- Often we see an exponential decrease in predictive improvement as the number of iterations increases; the most improvement in prediction performance is obtained with a small number of trees ($m < 10$).
- To illustrate this point, consider this figure which displays predictive performance (RMSE) for varying numbers of bootstrapped samples for CART trees.

BAGGED TREES



- To illustrate this point, consider this figure which displays predictive performance (RMSE) for varying numbers of bootstrapped samples for CART trees.
- Notice predictive performance improves through ten trees and then *tails off with very limited improvement beyond that point*.

BAGGED TREES – OTHER ENSEMBLES



- In our experience, *small improvements* can still be made using bagging ensembles *up to size 50*.
- *If performance is not at an acceptable level after 50 bagging iterations*, then we suggest trying *other more powerfully predictive ensemble methods* such as *random forests and boosting* which will be described the following sections.

BAGGED TREES - CAVEATS

- Although bagging usually improves predictive performance for unstable models, there are a few caveats.
  - First, computational costs and memory requirements increase as the number of bootstrap samples increases.
    - This disadvantage can be mostly mitigated if the modeler has access to *parallel computing* because the bagging process can be easily parallelized.
    - Recall that each bootstrap sample and corresponding model is independent of any other sample and model.
- This means that each model can be built separately and all models can be brought together in the end to generate the prediction.

BAGGED TREES

- Another disadvantage to this approach is that a bagged model is *much less interpretable* than a model that is not bagged.
- *Convenient rules* that we can get from a single regression tree *cannot be attained*. (*No rule from ensembles.*)
- However, measures of variable importance can be constructed *by combining measures of importance from the individual models across the ensemble*.
- More about variable importance will be discussed in the next section when we examine random forests.

RANDOM FORESTS – ISSUE IN BAGGED TREES

- Generating bootstrap samples introduces a random component into the tree building process, which induces a distribution of trees, and therefore also a distribution of predicted values for each sample.
- The trees in bagging, however, are *not completely independent of each other since all of the original predictors are considered at every split of every tree.*
- One can imagine that if we start with a sufficiently large number of original samples and a relationship between predictors and response that can be adequately modeled by a tree.
- Then trees from *different* bootstrap samples may have *similar structures* to each other due to the underlying relationship.

RANDOM FORESTS

- Despite taking bootstrap samples, each tree starts splitting on the number of carbon atoms at a scaled value of approximately 3.5.
- The second-level splits vary a bit more but are restricted to both of the surface area predictors and molecular weight.
- While each tree is ultimately unique—no two trees are exactly the same—they all begin with a similar structure and are consequently related to each other.
- Therefore, the variance reduction provided by bagging could be improved.

RANDOM FORESTS - DE-CORRELATING TREES

- From a statistical perspective, *reducing correlation among predictors* can be done *by adding randomness to the tree construction process*.
- After Breiman unveiled bagging, several authors tweaked the algorithm by adding randomness into the learning process.
- Because trees were a popular learner for bagging, Dietterich (2000) developed the idea of random split selection, where trees are built using a random subset of the top k predictors at each split in the tree.

RANDOM FORESTS

```
Select the number of models to build, m
for i = 1 to m do
    Generate a bootstrap sample of the original data
    Train a tree model on this sample
    for each split do
        Randomly select k (< P) of the original predictors
        Select the best predictor among the k predictors and
        partition the data
    end
    Use typical tree model stopping criteria to determine when a
    tree is complete (but do not prune)
end
```

- Another approach was to build entire trees based on *random subsets of descriptors* (Ho 1998; Amit and Geman 1997).
- Breiman (2000) also tried adding noise to the response in order to perturb tree structure.
- After carefully evaluating these generalizations to the original bagging algorithm, Breiman (2001) constructed a unified algorithm called random forests.
- A general random forests algorithm for a tree-based model can be implemented as shown in Algorithm 8.2.

RANDOM FORESTS

- Each model in the ensemble is then used to generate a prediction for a new sample and these m predictions are averaged to give the forest's prediction.
- Since the algorithm randomly selects predictors at each split, tree correlation will necessarily be lessened.
- As an example, the first splits for the first six trees in the random forest for the solubility data are **NumNonHBonds**, **NumCarbon**, **NumNonHAtoms**, **NumCarbon**, **NumCarbon**, and **NumCarbon.**
- We can see the different from the trees illustrated in Fig. 8.14.

RANDOM FORESTS – TUNING PARAMETER

- The practitioner must also specify the number of trees for the forest.
- Breiman (2001) proved that random forests is protected from over-fitting; therefore, the model will not be adversely affected if a large number of trees are built for the forest.
- Practically speaking, the larger the forest, the more computational burden we will incur to train and build the model. As a starting point, we suggest using at least 1,000 trees.
- If the cross-validation performance profiles are still improving at 1,000 trees, then incorporate more trees until performance levels off.

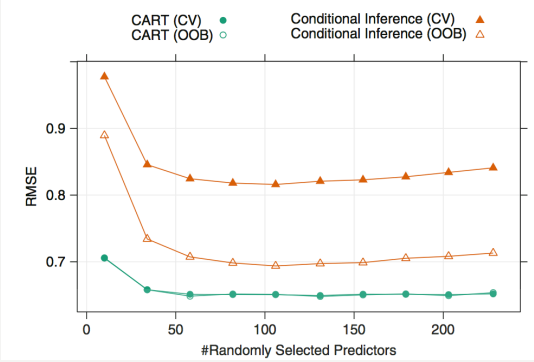
RANDOM FORESTS - ROBUSTNESS

- A random forest model achieves this variance reduction by selecting strong, complex learners that exhibit low bias.
- This ensemble of many *independent*, *strong learners* yields an improvement in error rates.
- Because each learner is selected independently of all previous learners, random forests is *robust* to a noisy response.

RANDOM FORESTS – COMPUTATIONAL EFFICIENCY

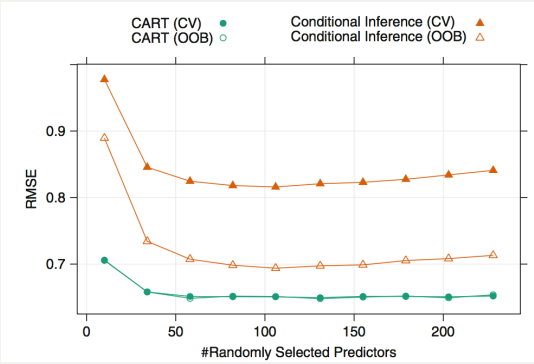
- Compared to bagging, random forests is *more computationally efficient on a tree-by-tree basis* since the tree building process only needs to evaluate *a fraction of the original predictors* at each split.
- Although more trees are usually required by random forests.

RANDOM FORESTS – COMPARISONS ON CART & CTREE



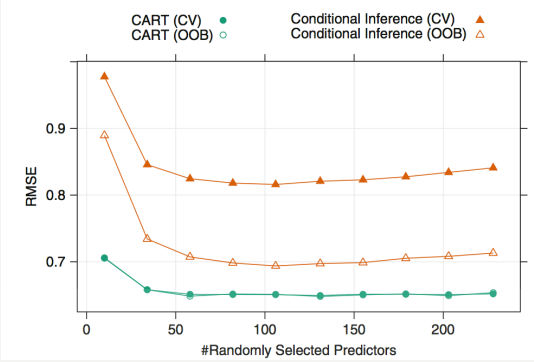
- Like bagging, CART or conditional inference trees can be used as the base learner in random forests.
- Both of these base learners were used, as well as 10-fold cross-validation and out-of-bag validation, to train models on the solubility data.
- The m_{try} parameter was evaluated at ten values from 10 to 228. The RMSE profiles for these combinations are presented in this figure.

RANDOM FORESTS



- Contrary to bagging, CART trees have better performance than conditional inference trees at all values of the tuning parameter.
- Each of the profiles shows a flat plateau between $m_{try} = 58$ and $m_{try} = 155$.
- The CART-based random forest model was numerically optimal at $m_{try} = 131$ regardless of the method of estimating the RMSE.

RANDOM FORESTS



- Notice that random forest models built with CART trees had extremely similar RMSE results with the out-of-bag error estimate and cross-validation.
- It is unclear whether the pattern seen in these data generalizes, especially under different circumstances such as small sample sizes.

RANDOM FORESTS

- Using the out-of-bag error rate would *drastically decrease the computational time* to tune random forest models.
- For forests created using conditional inference trees, the out-of-bag error was much more optimistic than the cross-validated RMSE.
- Again, the reasoning behind this pattern is unclear.

RANDOM FORESTS

- In these data, the only real difference in the RMSE comes when the smallest value is used.
- It is often the case that such a small value is not associated with optimal performance.
- However, we have seen rare examples where small tuning parameter values generate the best results.
- To get a quick assessment of how well the random forest model performs, the default tuning parameter value for regression ($m_{try} = P/3$) tends to work well.
- If there is a desire to maximize performance, tuning this value may result in a slight improvement.

RANDOM FORESTS – IMPORTANCE OF PREDICTORS

- The ensemble nature of random forests makes it impossible to gain an understanding of the relationship between the predictors and the response.
- However, because trees are the typical base learner for this method, it is *possible to quantify the impact of predictors in the ensemble*.
- The difference in predictive performance between the non-permuted sample and the permuted sample for each predictor is recorded and aggregated across the entire forest.

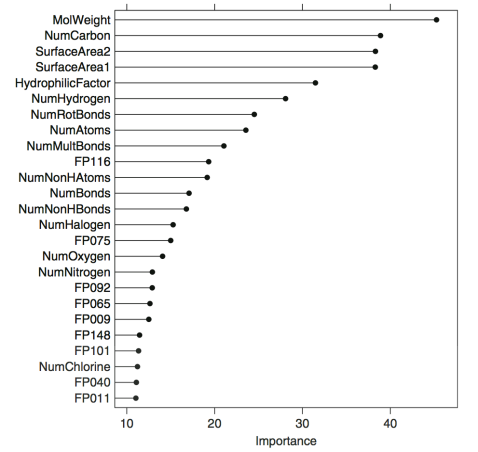
RANDOM FORESTS

- Another approach is to measure the improvement in node purity based on the performance metric for each predictor at each occurrence of that predictor across the forest.
- These individual improvement values for each predictor are then aggregated across the forest to determine the overall importance for the predictor.
- Another impact of between-predictor correlations is to dilute the importance of key predictors.

RANDOM FORESTS

- Strobl et al. (2007) developed an alternative approach for calculating importance in random forest models that takes between-predictor correlations into account.
- Their methodology reduces the effect of between-predictor redundancy.
- It does not adjust for the aforementioned dilution effect.

RANDOM FORESTS



- Random forest variable importance values for the top 25 predictors of the solubility data are presented in this figure.
- For this model, **MolWeight**, **NumCarbon**, **SurfaceArea2**, and **SurfaceArea1** percolate to the top of the importance metric, and importance values begin to taper with fingerprints.
- Importance values for fingerprints 116 and 75 are top fingerprint performers for importance, which may indicate that the structures represented by these fingerprints have an impact on a compound's solubility.

RANDOM FORESTS

- The importance orderings are much different.
- For example **NumNonHBonds** is the top predictor for a CART tree but ends up ranked 14th for random forests; random forests identify **MolWeight** as the top predictor, whereas a CART tree ranks it 5th.
- These differences should not be disconcerting; rather they emphasize that a single tree's greediness prioritizes predictors differently than a random forest.

BOOSTING - HISTORY

- For completeness of this section, we will give a history of boosting to provide a bridge from boosting's original development in classification to its use in the regression context.
- This history begins with the AdaBoost algorithm and evolves to Friedman's stochastic gradient boosting machine, which is now widely accepted as the boosting algorithm of choice among practitioners.

BOOSTING

- Boosting, especially in the form of the AdaBoost algorithm, was shown to be a powerful prediction tool.
- Usually outperforming any individual model. Its success drew attention from the modeling community and its use became widespread with applications in gene expression (Dudoit et al. 2002; Ben-Dor et al. 2000), Chemometrics (Varmuza et al. 2003), and music genre identification (Bergstra et al. 2006), to name a few.

BOOSTING – BASIC PRINCIPLES

- The basic principles of gradient boosting are as follows: given *a loss* and *a weak learner*, the algorithm seeks to find *an additive model* that *minimizes the loss function*.
- The algorithm is typically initialized with the *best guess of the response*.
- The gradient is calculated, and a model is then fit to the residuals to minimize the loss function. The current model is added to the previous model, and the procedure continues for a user-specified number of iterations.

BOOSTING – REQUIREMENT & WHY TREES

- Boosting requires a weak learner, *almost any technique with tuning parameters can be made into a weak learner*. Trees, as it turns out, make an excellent base learner for boosting for several reasons.
  - First, they have the flexibility to be weak learners *by simply restricting their depth*.
  - Second, *separate trees can be easily added together*, much like individual predictors can be added together in a regression model, to generate a prediction. And third, trees can be generated very quickly.
- Hence, results from individual trees can be directly aggregated, thus making them inherently suitable for an additive modeling process.

BOOSTING - ADABOOST

1 Select tree depth, D, and number of iterations, K
2 Compute the average response, $\overline{y}$, and use this as the initial predicted value for each sample
3 **for** $k = 1$ *to* K **do**
4 Compute the residual, the difference between the observed value and the *current* predicted value, for each sample
5 Fit a regression tree of depth, D, using the residuals as the response
6 Predict each sample using the regression tree fit in the previous step
7 Update the predicted value of each sample by adding the previous iteration's predicted value to the predicted value generated in the previous step
8 **end**

- When regression tree are used as the base learner, simple gradient boosting for regression has two tuning parameters: tree depth and number of iterations.
- Tree depth in this context is also known as interaction depth, since each subsequential split can be thought of as a higher-level interaction term with all of the other previous split predictors.

BOOSTING

1 Select tree depth, D, and number of iterations, K
2 Compute the average response, $\overline{y}$, and use this as the initial predicted value for each sample
3 **for** $k = 1$ *to* K **do**
4 Compute the residual, the difference between the observed value and the *current* predicted value, for each sample
5 Fit a regression tree of depth, D, using the residuals as the response
6 Predict each sample using the regression tree fit in the previous step
7 Update the predicted value of each sample by adding the previous iteration's predicted value to the predicted value generated in the previous step
8 **end**

- If squared error is used as the loss function, then a simple boosting algorithm using these tuning parameters can be found in this algorithm.
- Clearly, the version of boosting presented in this algorithm has similarities to random forests: the final prediction is based on an ensemble of models, and trees are used as the base learner.

BOOSTING

1 Select tree depth, D, and number of iterations, K
2 Compute the average response, $\bar{y}$, and use this as the initial predicted value for each sample
3 **for** $k = 1$ *to* K **do**
4 Compute the residual, the difference between the observed value and the *current* predicted value, for each sample
5 Fit a regression tree of depth, D, using the residuals as the response
6 Predict each sample using the regression tree fit in the previous step
7 Update the predicted value of each sample by adding the previous iteration's predicted value to the predicted value generated in the previous step
8 **end**

- A regularization strategy can be injected into the final line of the loop.
- Instead of adding the predicted value for a sample to previous iteration's predicted value, only a fraction of the current predicted value is added to the previous iteration's predicted value.

BOOSTING – COMPARISON TO RF

- However, the way the ensembles are constructed differs substantially between each method.
- In *random forests*, *all trees are created independently*, each tree is created to have *maximum depth*, and each tree *contributes equally* to the final model.
- The *trees in boosting*, however, are *dependent on past trees*, have *minimum depth*, and *contribute unequally* to the final model.
- Despite these differences, *both* random forests and boosting *offer competitive predictive performance*.

BOOSTING - OVERFITTING

- Computation time for boosting is often *greater than for random forests*, since *random forests can be easily parallel processed* given that the *trees are created independently*.
- Despite using weak learners, boosting still employs the greedy strategy of choosing the optimal weak learner at each stage.
- Although this strategy generates an optimal solution at the current stage, it has the drawbacks of not finding the optimal global model as well as over-fitting the training data.

BOOSTING – SHRINKAGE

- This fraction is commonly referred to as the learning rate and is parameterized by the symbol, λ. This parameter can take values between 0 and 1 and becomes another tuning parameter for the model.
- Ridgeway (2007) suggests that small values of the learning parameter (< 0.01) work best.
- But he also notes that the value of the parameter is inversely proportional to the computation time required to find an optimal model, because more iterations are necessary.
- Having more iterations also implies that more memory is required for storing the model.

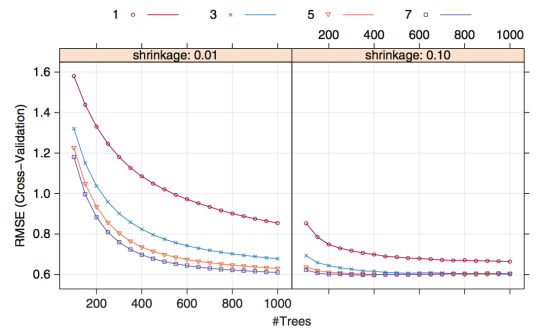
BOOSTING – STOCHASTIC GRADIENT BOOSTING MACHINES

- The random sampling nature of bagging offered a reduction in prediction variance for bagging.
- Friedman updated the boosting machine algorithm with a random sampling scheme and termed the new procedure stochastic gradient boosting.
- Friedman inserted the following step before line within the loop: randomly select a fraction of the training data.
- The residuals and models in the remaining steps of the current iteration are based only on the sample of data.

BOOSTING

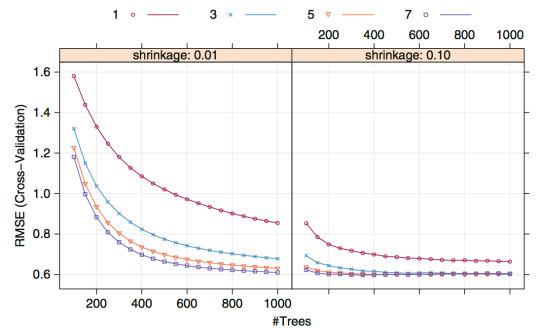
- The fraction of training data used, known as the bagging fraction, then becomes another tuning parameter for the model.
- It turns out that this simple modification improved the prediction accuracy of boosting while also reducing the required computational resources.
- Friedman suggests using a *bagging fraction of around 0.5*; this value, however, can be tuned like any other parameter.

BOOSTING



- This figure presents the cross-validated RMSE results for boosted trees across tuning parameters of tree depth (1–7), number of trees (100–1,000), and shrinkage (0.01 or 0.1); the bagging fraction in this illustration was fixed at 0.5.
- When examining this figure, the larger value of shrinkage (right-hand plot) has an impact on reducing RMSE for all choices of tree depth and number of trees.

BOOSTING

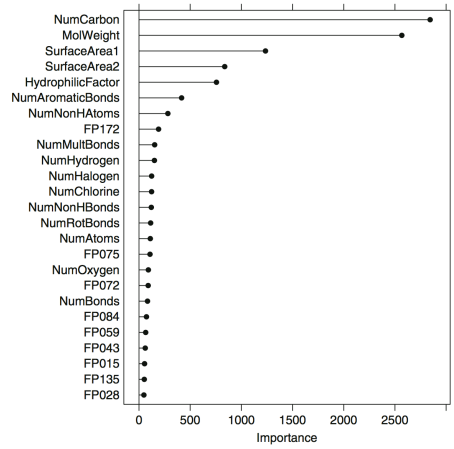


- Also, RMSE decreases as tree depth increases when shrinkage is 0.01.
- The same pattern holds true for RMSE when shrinkage is 0.1 and the number of trees is less than 300.
- Using the one-standard-error rule, the optimal boosted tree has depth 3 with 400 trees and shrinkage of 0.1.
- These settings produce a cross-validated RMSE of 0.616.

BOOSTING

- The improvement in squared error due to each predictor is summed within each tree in the ensemble.
- The improvement values for each predictor are then averaged across the entire ensemble to yield an overall importance value.
- The importance profile for boosting has a much steeper importance slope than the one for random forests.
- This is due to the fact that the trees from boosting are dependent on each other and hence will have correlated structures as the method follows by the gradient.

BOOSTING- IMPORTANCE OF PREDICTORS



- The top 25 predictors for the model are presented in this figure.
- **NumCarbon** and **MolWeight** stand out in this example as most important followed by SurfaceArea1 and SurfaceArea2.
- Importance values tail off after about 7 predictors.
- Comparing these results to random forests we see that both methods identify the same top 4 predictors, albeit in different order.

BOOSTING

- Therefore many of the same predictors will be selected across the trees, increasing their contribution to the importance metric.
- Differences between variable importance ordering and magnitude between random forests and boosting should not be disconcerting.
- Instead, one should consider these as two different perspectives of the data and use each view to provide some understanding of the gross relationships between predictors and the response.

CUBIST

- Cubist is a rule-based model that is an amalgamation of several methodologies published some time ago (Quinlan 1987, 1992, 1993a) but has evolved over this period.
- Our description of the model stems from the open-source version of the model.
- Some specific differences between Cubist and the previously described approaches for model trees and their rule-based variants are:
  - The specific techniques used for linear model smoothing, creating rules, and pruning are different.
  - An optional boosting—like procedure called *committees*.
  - The predictions generated by the model rules can be adjusted using nearby points from the training set data.

CUBIST

- In Cubist, the models are still combined using a linear combination of two models:

$$\hat{y}_{\text{par}} = a \times \hat{y}_{(k)} + (1 - a) \times \hat{y}_{(p)}$$

- where yˆ(k) is the prediction from the current model and yˆ(p) is from parent model above it in the tree.
- Compared to model trees, Cubist calculates the mixing proportions using a different equation.

CUBIST

- The smoothing procedure first determines the covariance between the two sets of model residuals (denoted as Cov[$e_{(p)}$, $e_{(k)}$]).
- This is an overall measure of the linear relation between the two sets of residuals.
- If the covariance is large, this implies that the residuals generally have the same sign and relative magnitude, while a value near 0 would indicate no (linear) relationship between the errors of the two models.
- Cubist also calculates the variance of the difference between the residuals

$$a = \frac{\text{Var}[e_{(p)}] - \text{Cov}[e_{(k)}, e_{(p)}]}{\text{Var}[e_{(p)} - e_{(k)}]}$$

CUBIST

- The first part of the numerator is proportional to the parent model's RMSE.
- If variance of the parent model's errors is larger than the covariance, the smoothing procedure tends to weight the child more than the parent.
- Conversely, if the variance of the parent model is low, that model is given more weight.
- In the end, the model with the smallest RMSE has a higher weight in the smoothed model.
- When the models have the same RMSE, they are equally weighted in the smooth procedure.

CUBIST

- Unlike the previously discussed "separate and conquer" methodology, the final model tree is used to construct the initial set of rules.
- Cubist collects the sequence of linear models at each node into a single, smoothed representation of the models so that there is one liner model associated with each rule.
- Starting with splits made near the terminal nodes, each condition of the rules is tested using the adjusted error rate for the training set.
- If the deletion of a condition in the rule does not increase the error rate, it is dropped.

CUBIST

- This can lead to entire rules being removed from the overall model.
- Once the rule conditions have been finalized, a new sample is predicted using the average of the linear models from the appropriate rules.
- Model committees can be created by generating a sequence of rule-based models. Like boosting, each model is affected by the result of the previous models.
- Recall that boosting uses new weights for each data point based on previous fits and then fits a new model utilizing these weights.

CUBIST

- Commit- tees function differently.
- The training set outcome is adjusted based on the prior model fit and then builds a new set of rules using this pseudo-response.
- Specifically, the *m*th committee model uses an adjusted response:

$$y^*_{(m)} = y - (\hat{y}_{(m-1)} - y)$$

CUBIST

- Basically, if a data point is underpredicted, the sample value is increased in the hope that the model will produce a larger prediction in the next iteration.
- Similarly, over-predicted points are adjusted so that the next model will lower its prediction.
- Once the full set of committee models are created, new samples are predicted using each model and the final rule-based prediction is the simple average of the individual model predictions

CUBIST

- When predicting a new sample, the K most similar neighbors are determined from the training set.
- Suppose that the model predicts the new sample to be yˆ and then the final prediction would be

$$\frac{1}{K}\sum_{\ell=1}^{K} w_\ell \left[t_\ell + (\hat{y} - \hat{t}_\ell)\right]$$

- where t_l is the observed outcome for a training set neighbor, t_lˆ is the model prediction of that neighbor, and w_l is a weight calculated using the distance of the training set neighbors to the new sample.

CUBIST

- As the difference between the predictions of the new sample and its closest neighbor increases, the adjustment becomes larger.
- There are several details that must be specified to enact this process.
  - First, a distance metric to define the neighbors is needed.
    - The implementation of Cubist uses Manhattan (a.k.a. city block) distances to determine the nearest neighbors.
  - Also, neighbors are only included if they are "close enough" to the prediction sample.
    - To filter the neighbors, the average pairwise distance of data points in the training set is used as a threshold.

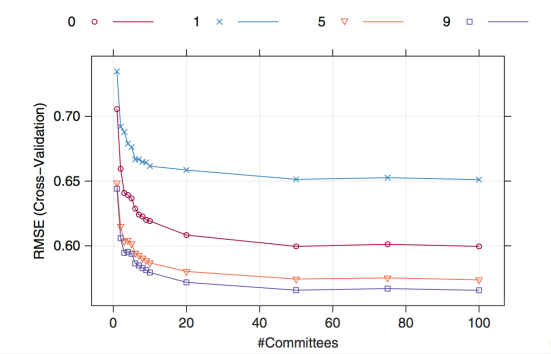
CUBIST

- If the distance from the potential neighbor to the prediction samples is greater than this average distance, the neighbor is excluded.
- The weights w_l also use the distance to the neighbors. The raw weights are computed as

$$w_\ell = \frac{1}{D_\ell + 0.5}$$

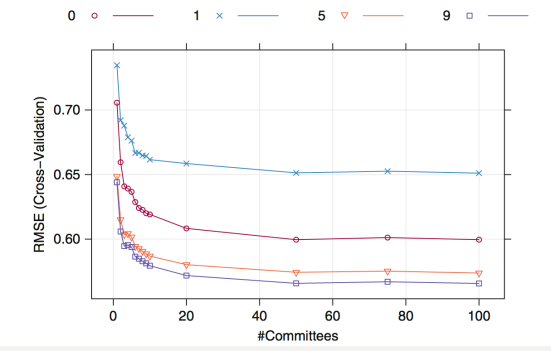
- where D_l is the distance of the neighbor to the prediction sample.

CUBIST



- Independent of the number of neighbors used, there is a trend where the error is significantly reduced as the number of committees is increased and then stabilizes around 50 committees.
- The use of the training set to adjust the model predictions is interesting: a purely rule-based model does better than an adjustment with a single neighbor, but the error is reduced the most when nine neighbors are used.

CUBIST



- In the end, the model with the lowest error (0.57 log units) was associated with 100 committees and an adjustment using nine neighbors, although fewer committees could also be used without much loss of performance.
- For the final Cubist model, the average number of rules per committee was 5.1 but ranged from 1 to 15.

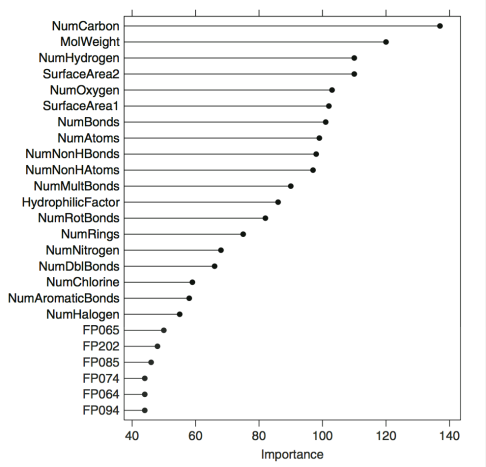
CUBIST

- We can compare the Cubist model with a single committee member and no neighbor adjustment to the previous rule-based model.
- The M5 rule-based model had an estimated cross-validation error of 0.74 whereas the corresponding Cubist model had error rate of 0.71.
- Based on the variation in the results, this difference is slightly statistically significant (p-value: 0.0485).
- This might indicate that the methodological differences between the two methods for constructing rule-based models are not large, at least for these data.

CUBIST

- Each linear model has a corresponding slope for each predictor, but, as previously shown, these values can be gigantic when there is significant collinearity in the data.
- A metric that relied solely on these values would also ignore which predictors were used in the splits.
- This approach ignores the neighbor-based correction that is sometimes used by Cubist.
- The modeler can choose how to weight the counts for the splits and the linear models in the overall usage calculation.

CUBIST



- For the solubility data, predictor importance values were calculated for the model with 100 committees and correct the prediction using the 9-nearest neighbors.
- Like many of the other models discussed for these data, the continuous predictors appear to have a greater impact on the model than the fingerprint descriptors.
- Unlike the boosted tree model, there is a more gradual decrease in importance for these data; there is not a small subset of predictors that are dominating the model fit.

THANK YOU