Lecture 13
Decision Trees and Random Forest

EL-GY 6143: INTRODUCTION TO MACHINE LEARNING

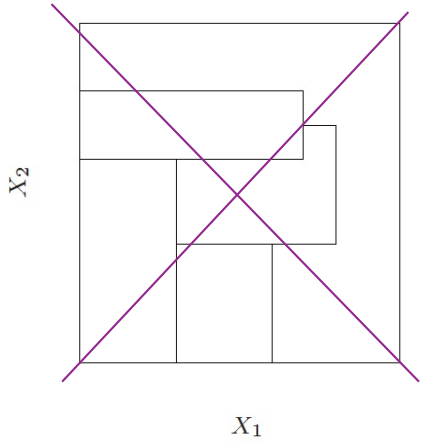
PROF. PEI LIU

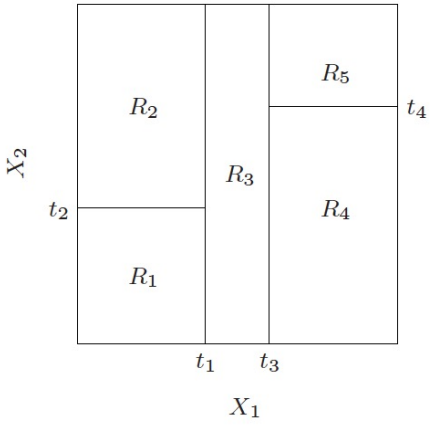
Outline

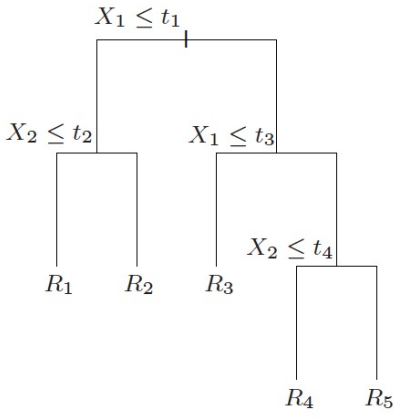
- ❑ Decision tree as constrained space partition
- ❑ Regression tree design
- ❑ Decision tree pruning
- ❑ Classification tree design
- ❑ Bagging
- ❑ Random Forest
- ❑ Feature ranking from random forest

Decision tree as constrained space partitioning

- Each region is regressed/classified to the same value
- The partition can be specified sequentially by splitting the range of one feature at a time.
- The splitting rule can be described by a tree.
  - Each leaf node = One region
  - Size of tree $|T|$= number of leaf nodes
- The partition is constrained: only rectangles in the 2D case.
  - The top left partition cannot be realized by a decision tree.







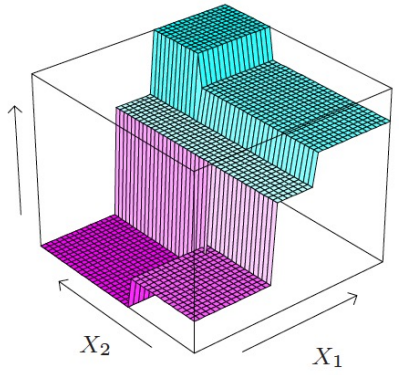


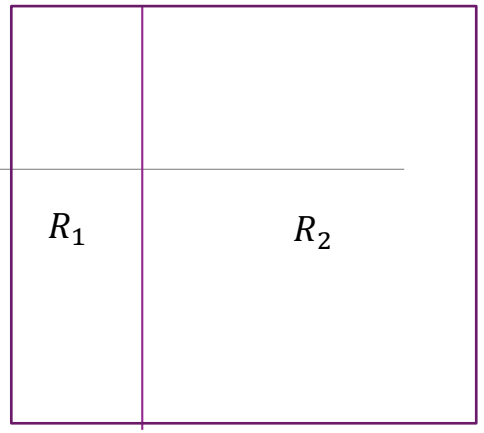
Fig. 9.2 in ESL

How to build a decision tree? (Regression Case)

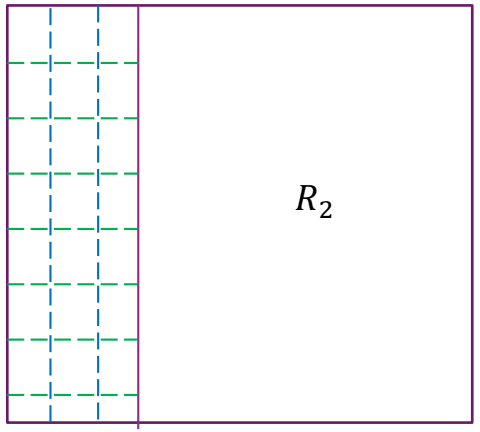
- Goal: minimize RSS

$$L = \sum_{m=1}^{|T|} \sum_{x_n \in R_m} (y_n - \bar{y}_m)^2$$

- Greedy algorithm:
- Start with a single region (entire space) and iterate:
- For each region R_m, select a feature x_j, and a splitting threshold s, such that splitting R_k with the criterion $x_j < s$ produces the largest decrease in RSS in R_m
  - Exhaustive search: for each x_j, try all possible s in the current range of x_j in R_m
- Stop splitting a region if it contains <= N_{min} samples



All possible splits of R_1:



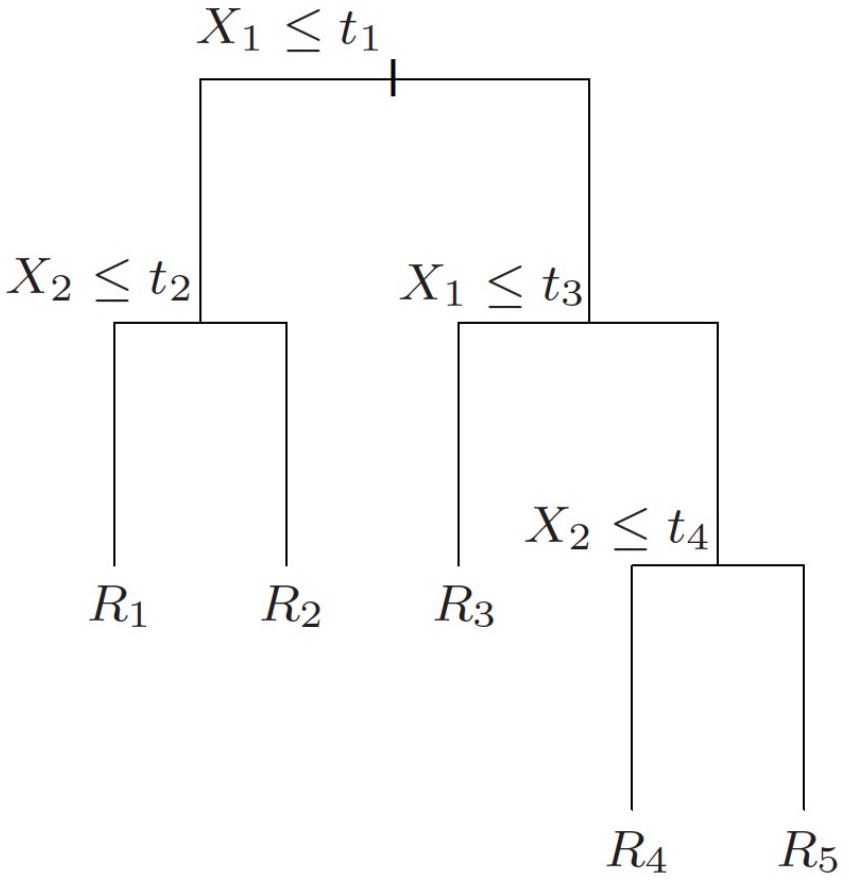
Overfitting

- Decision tree is very prone to overfitting
- Can exactly represent any function defined by the training set by having as many regions (or leaf nodes) as needed (Fully grown tree)
- How to control overfitting?
  - Find optimal subtree (with a certain constraint on the minimum number of samples in the leaf nodes or maximum depth) by cross validation: too many possibilities
  - Stop growing once RSS stop decreasing by a threshold with any new cut:
    - Not good because we use greedy search. It is possible to find a good cut after a bad one.
  - Better idea: grow a full tree first, then prune the tree.

Weakest link pruning

- Starting with with the initial full tree T0 , merge two adjacent leave nodes (daughter nodes) to a single leaf node (mother). Select which nodes to merge by minimizing error increase. This produces a tree with one less region (or node)
- Repeat to merge another two nodes, until the minimum size tree is reached (e.g. a stump with 2 nodes)
- Generate a sequence of trees

  T0, T1, T2, T3, …
- Which one to choose?



Cost complexity pruning

- Minimize a complexity regularized loss, over all possible trees T0, T1, T2, …

$$L(T,\alpha)=\sum_{m=1}^{|T|}\sum_{x_n\in R_m}(y_n-\bar{y}_m)^2+\alpha|T|$$

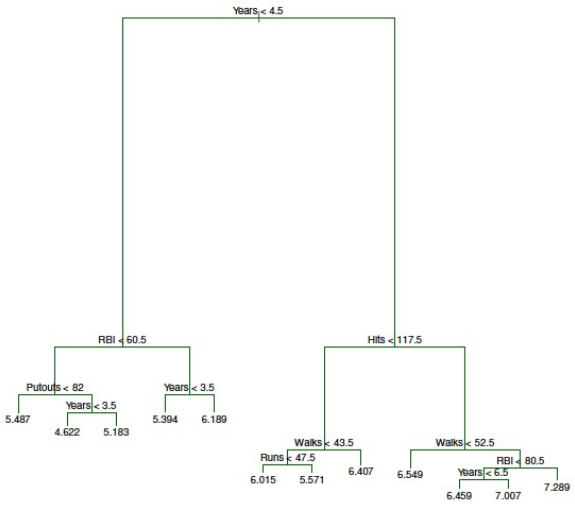
Number of leaf nodes (regions)

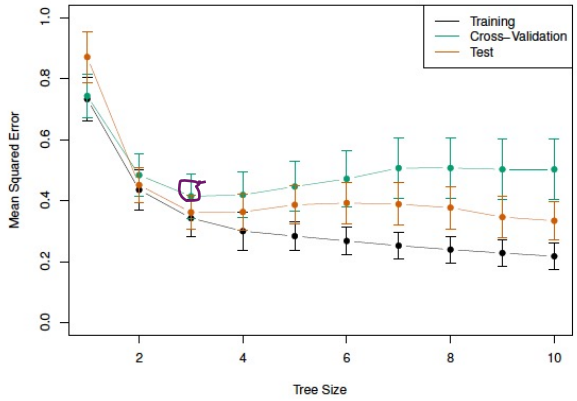
  - α=0: Full tree, $\alpha=\infty$: minimum sized tree

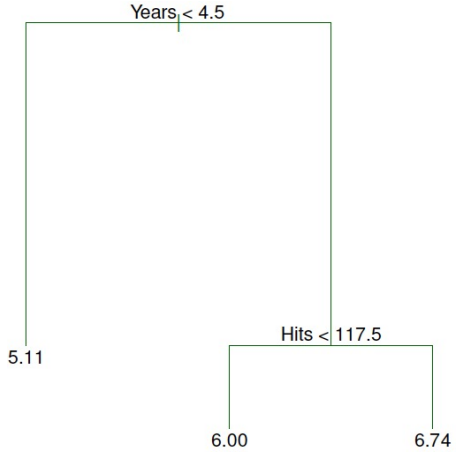
- How to choose α? Cross validation!
  - For each α
    - For each validation fold:
      - build a sequence of trees using the training set, and finding the RSS on the testing set for each candidate tree. Find a tree that minimizes $L(T,\alpha)$
    - Find average $L(T,\alpha)$ over all validation folds

- When dataset is very large, can just pick one tree that has minimal RSS for the testset.

Example: Predicting baseball player salaries





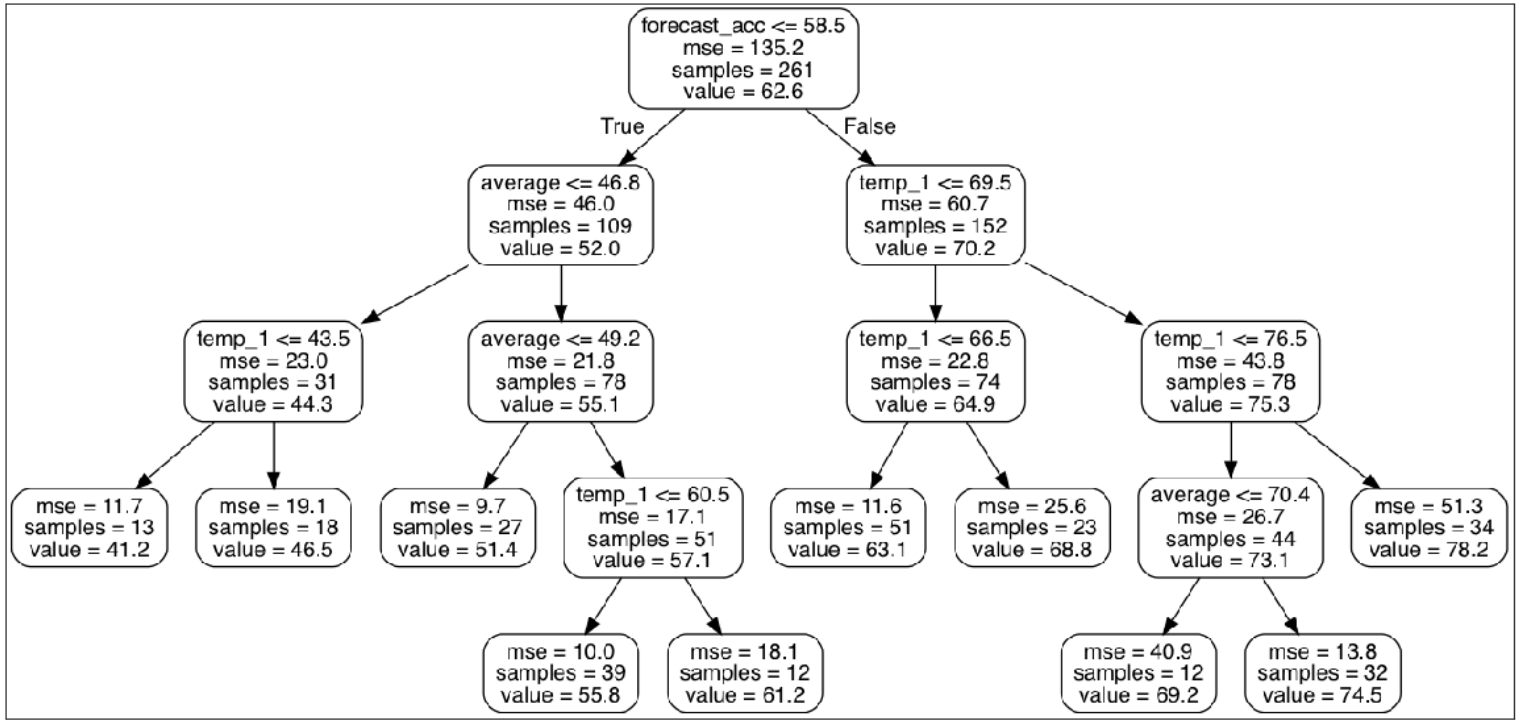


From http://web.stanford.edu/class/stats202/content/lec19.pdf

Feature importance

- For each feature, find all splits where this feature was used as the split variable and add up the loss reduction at all such splits
- The sum reflects the importance of this feature!

Demo: weather prediction using decision tree



What about classification?

- The predicted class for each region = the majority class of training samples in the region
- How to design the tree?
  - Can use the same greedy algorithm
  - Split each region (by picking a feature and a threshold) to minimize a loss
  - What loss functions to use?

Classification loss

- Misclassification rate

$$L = \sum_{m=1}^{|T|} \sum_{x_n \in R_m} 1(y_n \neq \bar{y}_m) \quad , \bar{y}_m = \text{majority class of } R_m$$

- Gini index

$$L = \sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{m,k}\left(1 - \hat{p}_{m,k}\right), \ \hat{p}_{m,k} = \text{numbers of samples in class k in } R_m$$

  - Expected error rate if we randomly pick an index, with probability $\hat{p}_{m,k}$ and error rate $1 - \hat{p}_{m,k}$

- Cross entropy

$$L = -\sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{m,k} \log \hat{p}_{m,k} \, ,$$

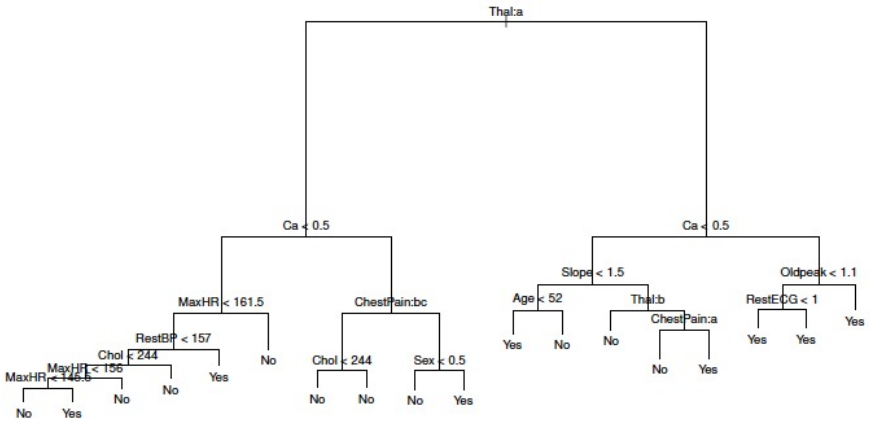
  - Smaller entropy means less uniform distribution (the region is more pure!)

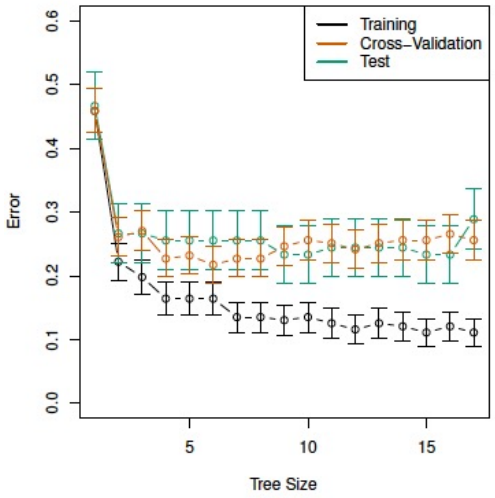
- Gini and cross entropy loss lead to more ”Pure” regions, with 1 dominate class in each region.

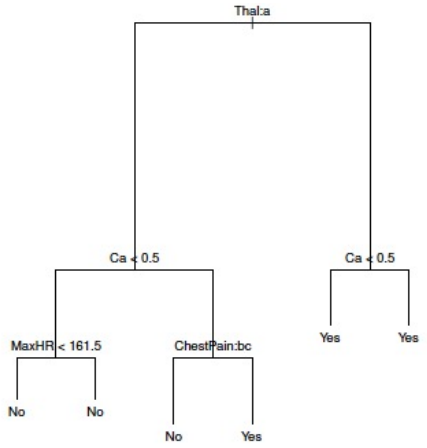
Performance metric and pruning

- After a tree is designed, the performance is still measured by the misclassification rate or accuracy
- It is typical to use the Gini index or cross entropy when growing a tree, but use the misclassification rate for pruning a tree

Example: Classifying heart disease







From http://web.stanford.edu/class/stats202/content/lec19.pdf

Advantage of decision tree

- Easy to interpret: Doctors like them
- Closer to human decision making
- Feature importance can be derived during training
- Can easily handle mixed type of features (numerical and categorical) and missing features in some samples
  - Did not discuss here
- Problem:
  - To reduce bias, needs to grow the tree deeper
  - Deeper trees tend to overfit the training data (Large variance among different training data)
  - How to overcome ?

Bagging (Bootstrap Aggregating)

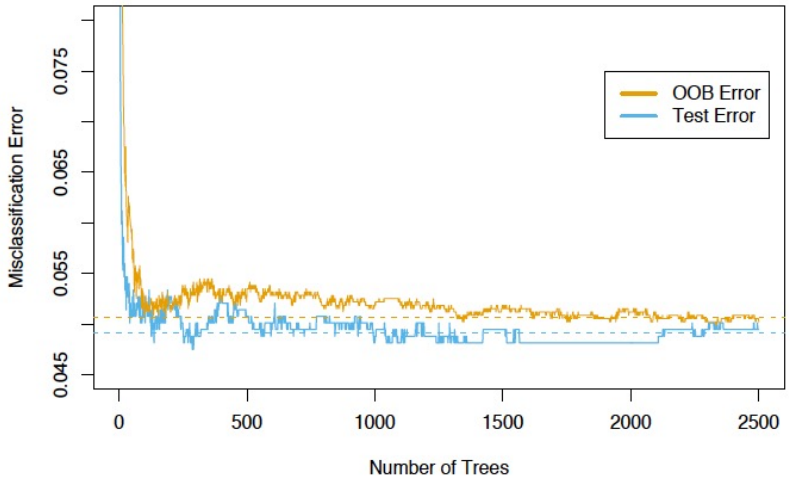
- Idea: Generate multiple trees from different training sets, and apply all models to each test sample and take average (or majority) of the results from all the trees
- How to generate different training sets giving a dataset?
- Cross validation: using a subset of data each time for training and the remaining for testing
- Bootstrap sampling: Sampling by replacement, each sampling contains the same number of samples as the original dataset, but some samples are replicated, others were not included
- Bagging: Generate B models from B bootstrap samplings
  - Regression: Average the prediction results from B models
  - Classification: Take the majority class index
- Apply to other regressors/classifiers as well.

Out of bag (OOB) error

- Each time we draw a bootstrap sampling, we only use ~63% of the samples
  - Probability that a sample is chosen among N samples in each bootstrap sampling

$$1-\left(1-\frac{1}{N}\right)^{N} \sim 1-e^{-1}=0.632$$

- We can use the remaining samples for testing
- OOB Error
  - For each sample x_n, find the models generated by samplings which do not contain x_n. There are about 0.37B of models. Average predictions by these models for x_n.
  - Compute the regression/classification error for x_n
  - Average the error over all samples
- We can use OOB error as an estimate of the test error.
- Does not require design multiple models for multiple folds as in cross validation. OOB can be estimated from one pass of designing multiple trees.



From ESL Fig. 15.4

Why bagging?

- When a regressor or classifier has tendency to overfit (i.e. sensitive to the training set), bagging reduces the variance of the prediction
  - Reduce the test error
  - Particularly useful for decision trees
- When the sample number N in a given dataset is large
  - The empirical distribution is similar to the true distribution
  - Each bootstrap sampling is similar to an independent realization of the true distribution
  - Bagging amounts to averaging the fits from many identically distributed datasets

Problems with bagging?

- Trees generated by different samplings can be very similar
- Test error reduces slowly as *B* increases

  - $f_b(x)$: prediction by tree b for test sample x
  - Assume $f_b(x)$ for all b have the same mean μ and variance σ^2
  - Assume these predictions have pair-wise correlation ρ
  - The variance of the average prediction $f(x) = \frac{1}{B}\sum_b f_b(x)$: (Shown on board)

$$\sigma_B^2 = \rho\,\sigma^2 + \frac{1}{B}(1-\rho)\sigma^2$$

Random Forest

- As with Bagging: fit a different tree for each bootstrap sampling
- Recall that when growing a tree, at each current node (region), we split the region by choosing a particular feature and a threshold. The feature and the threshold are chosen among all P features to minimize a certain loss.
- With random forest, randomly choose among a subset of features (P'<P) for splitting each node
- The resulting trees are more different
- Rule of thumb: $P' = \sqrt{P}$ (but should be turned using test error or OOB error)

Bagging vs. RF

- Bagging:

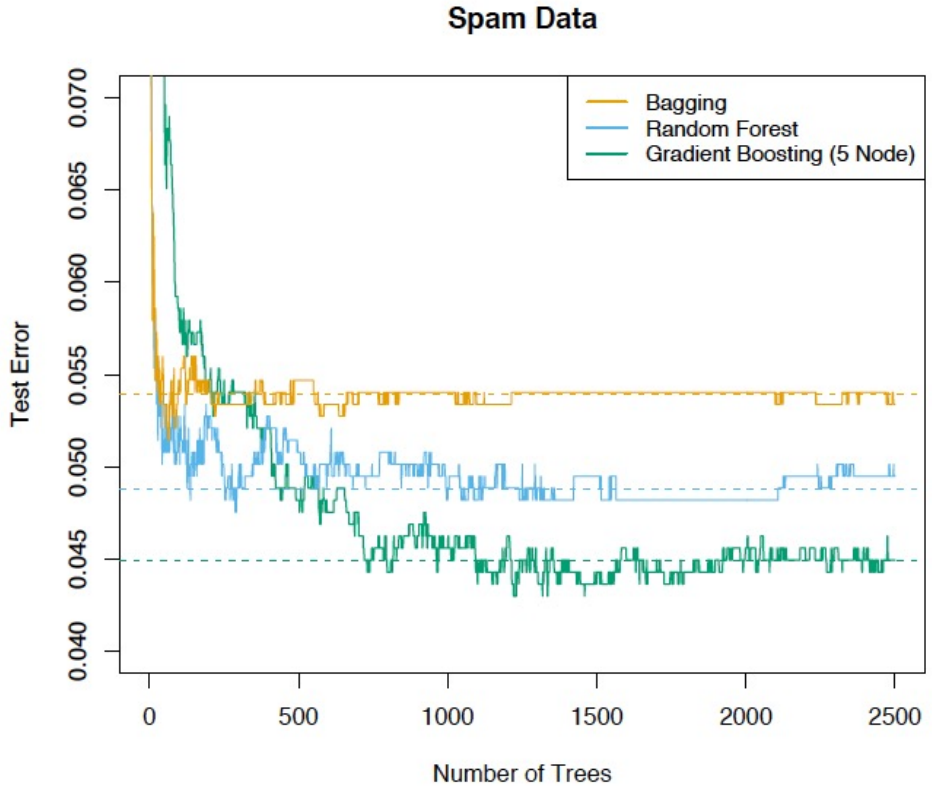
$$\sigma_B^2 = \rho\,\sigma^2 + \frac{1}{B}(1-\rho)\sigma^2$$

- Random forest (assuming $\rho = 0$):

$$\sigma_B^2 = \frac{1}{B}\sigma^2$$

- Recall:

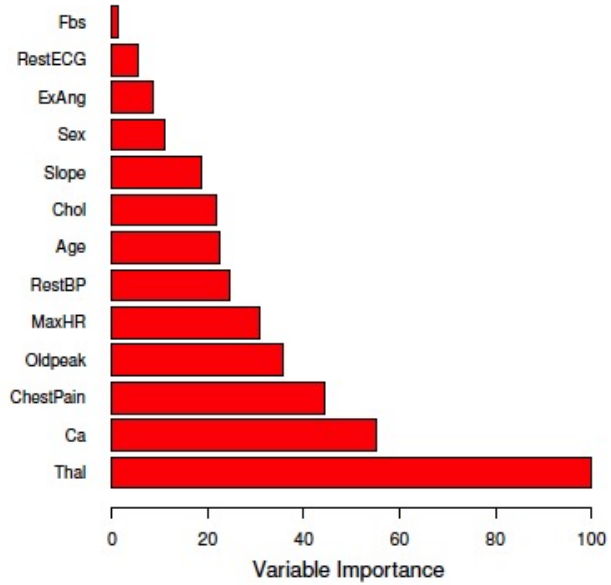
Test error = bias^2+ Variance +Noise Variance



From ESL, Fig. 15.1

Feature importance

- For each feature, add up the loss reduction at splits where this feature was used over all trees.



Demo: Random forest

Problem with bagging and random forest

- Resulting model has many trees!
- Lose interpretability!
- Related methods (not covered):
  - Boosting
  - Gradient boosting

What you should know from this lecture

- How to use/interpret decision tree ?
- How to train a decision tree ?
  - Loss function for regression
  - Loss function for classification
- How to reduce overfitting ?
- What does bagging mean ?
- How to train and use a random forest ?
- How to determine feature importance?

References

- [ESL] Hastie, T., & Tibshirani, R. & Friedman, J.(2008). The Elements of Statistical Learning; Data Mining, Inference and Prediction. Sec. 9. (Decision tree), Sec. 15.2 (Random forest)