COMP6915 Machine Learning

Random Forests

Dr. Lourdes Peña-Castillo
Departments of Computer Science and Biology
Memorial University of Newfoundland

Random Forests

- Random forests (Breiman, 2001) is a substantial modification of bagging that builds a large collection of *de-correlated* trees and then combines their prediction.
- On many problems, random forests' performance is comparable to that of boosting, and they are simpler to train and tune. Thus, random forests have become quite popular.

Random Forests

- In bagging all trees are identically distributed (but not necessarily independent). This means that the bias of bagged trees is the same as that of the individual trees, and bagging improvement is due to variance reduction.
- Random forests improve over bagging by reducing the correlation between the trees, without increasing the variance. To do this, in the tree-growing process random selection of the input variables is performed.

Rationale

- Suppose there is a feature in the data that is a very strong predictor of the response, and additionally there is a number of other features that are moderately strong predictors of the response
  - In bagging, most or all of the trees will use this strong predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other and their predictions will be highly correlated.
    - Averaging many highly correlated predictions does not substantially improve over a single tree
  - Random forests overcome this problem by forcing each split to consider only a subset of the features. Before each split, select $m <= p$ of the features at random as candidate for splitting, where p is the number of features
    - On average $(p-m)/p$ of the splits will not even consider the strong predictor and other features will be used more
- **What if a random forest is built with $m = p$?**

Random Forest

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Recommended values:
* For classification
m = floor(sqrt(p)) and n_{min} = 1
* For regression
m = floor(p/3) and n_{min} = 5
Best practice: tune these.

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Setting the *m* parameter

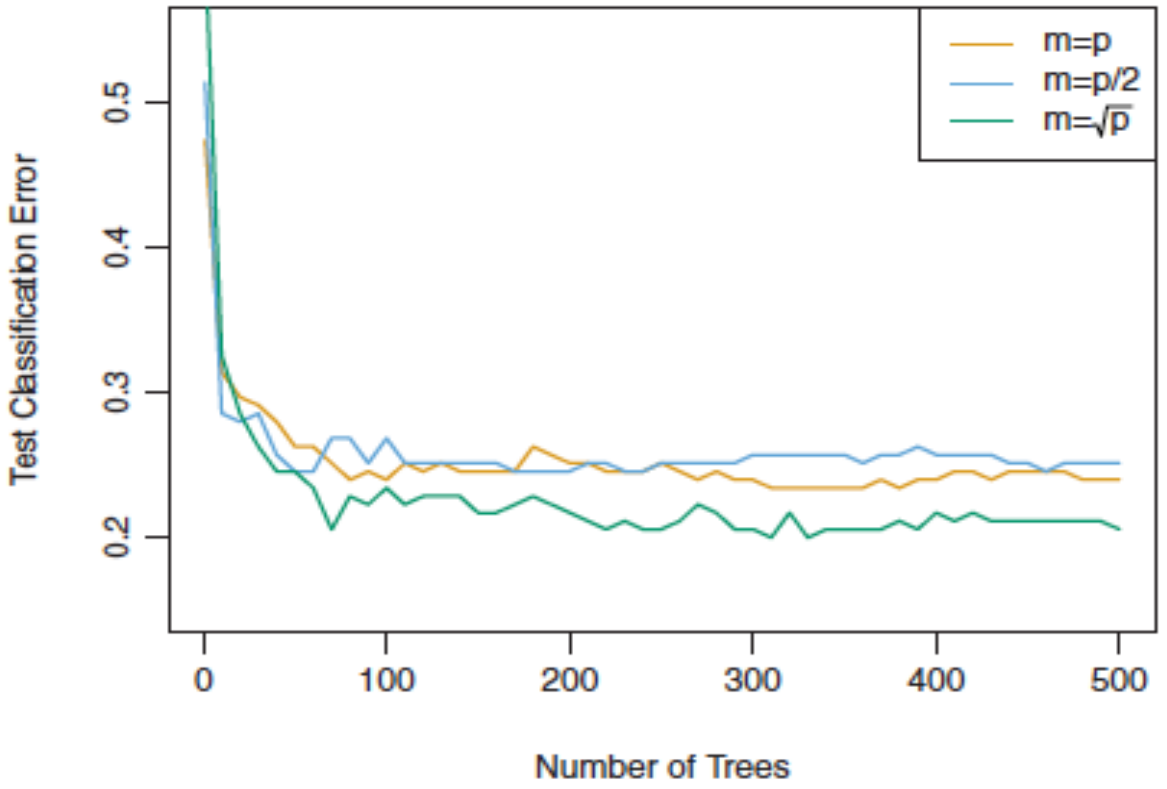

FIGURE 8.10. *Results from random forests for the 15-class gene expression data set with $p = 500$ predictors. The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node. Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$). A single classification tree has an error rate of 45.7%.*

Figure from *An Introduction to Statistical Learning: with Applications in R* by James, Witten, Hastie, and Tibshirani, 2013.

De-correlating trees

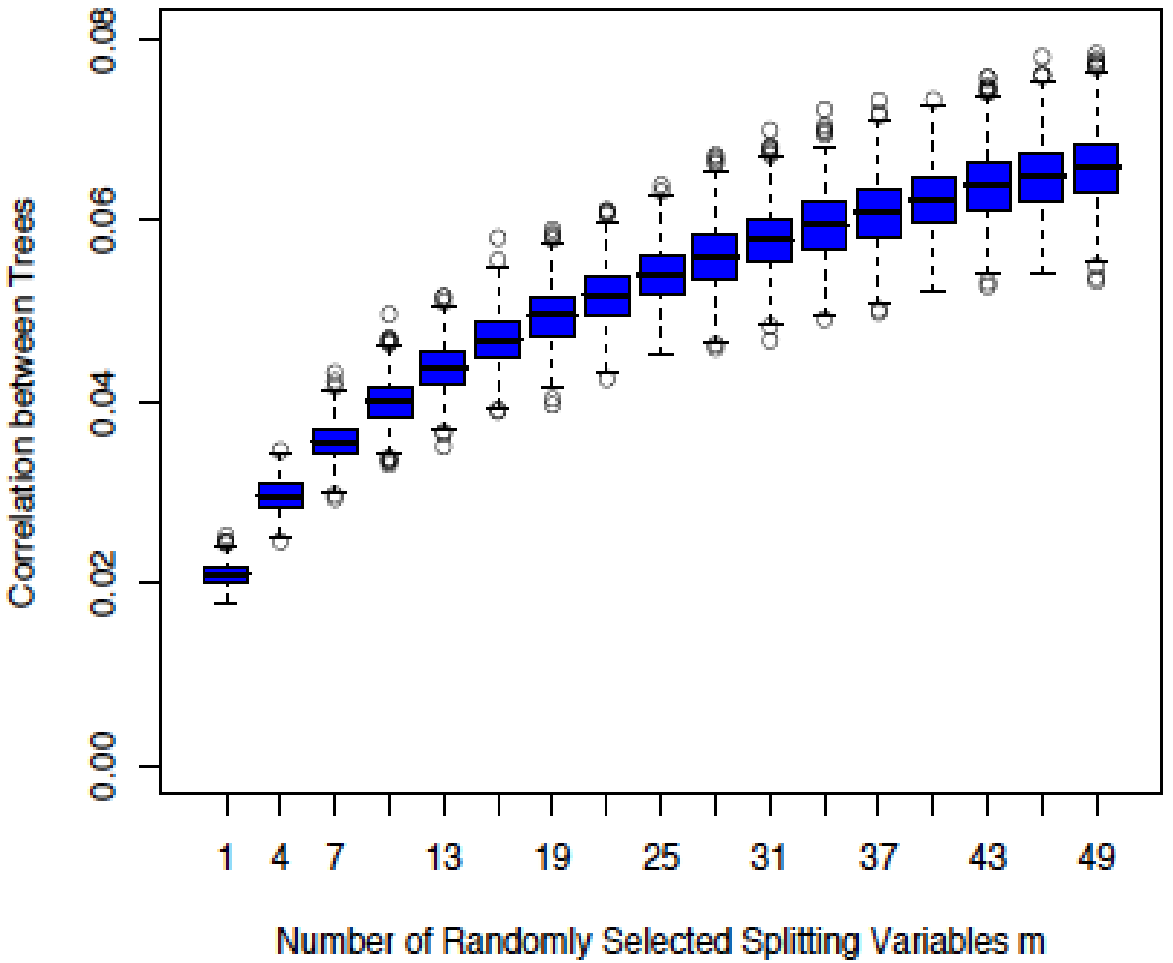

FIGURE 15.9. *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of m. The boxplots represent the correlations at 600 randomly chosen prediction points x.*

--> Reducing m reduces the correlation between any pair of trees in the ensemble.

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Out of Bag (OOB) Error

For each observation x_i, construct its random forest predictor by averaging only those trees corresponding to bootstrap samples in which x_i did not appear.

An OOB error estimate is almost identical to that obtained by N-fold cross-validation.

Hence, random forests can be fit in one sequence and once the OOB error stabilizes, the training can be terminated.

OOB vs Test Error

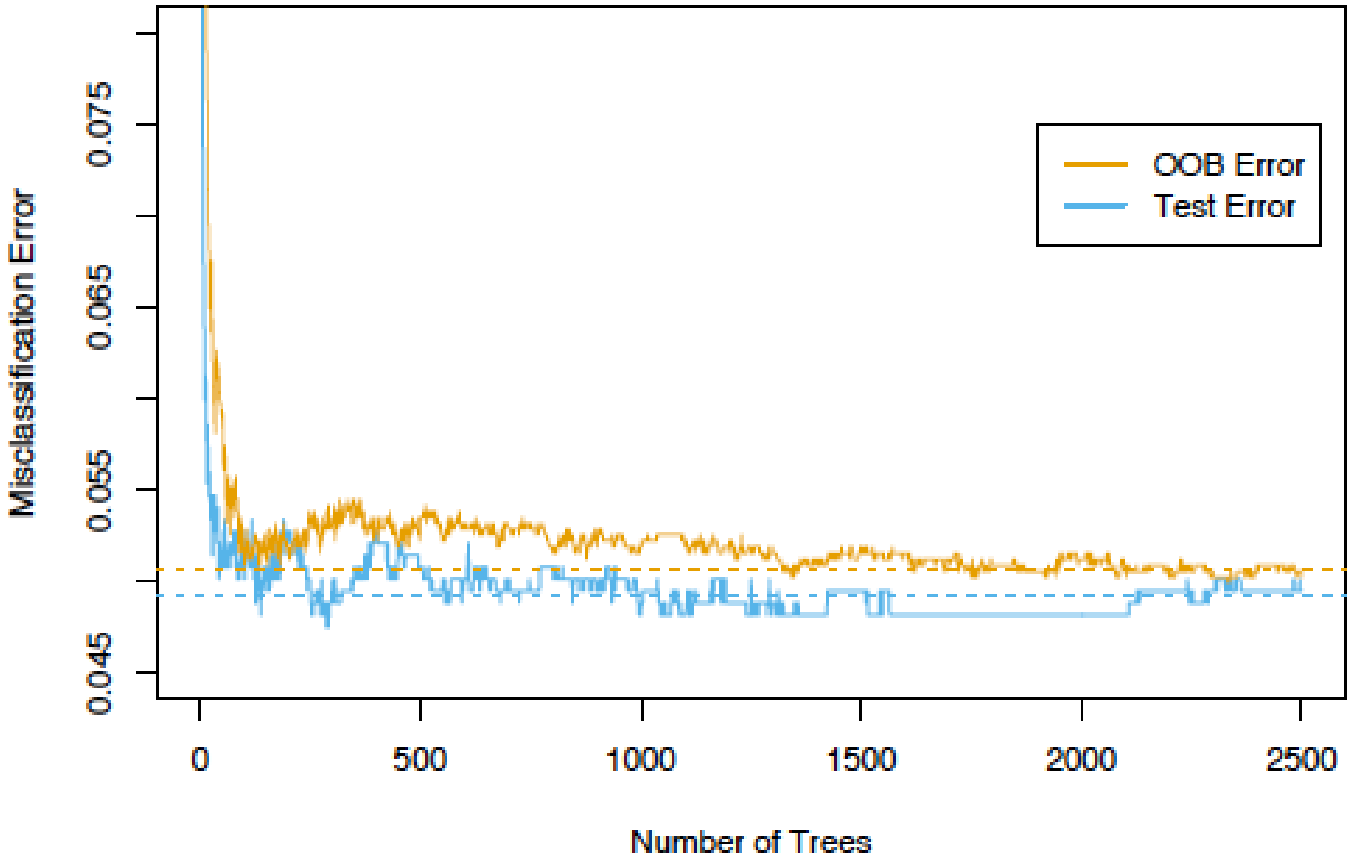

FIGURE 15.4. OOB *error computed on the* spam *training data, compared to the test error computed on the test set.*

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Variable Importance

- Random forests can estimate variable importance in two ways:
  1. [Same as bagging] At each split in each tree, the improvement in the split-criterion is the importance measure attributed to the splitting variable, and is accumulated over all the trees in the forest separately for each variable.
  2. [Using OOB instances] When the *b*th tree is grown, the OOB instances are passed down the tree, and its prediction accuracy is recorded. Then the values for the *j*th variable are randomly permuted in the OOB instances, and the accuracy is again computed. The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of variable *j* in the random forest.

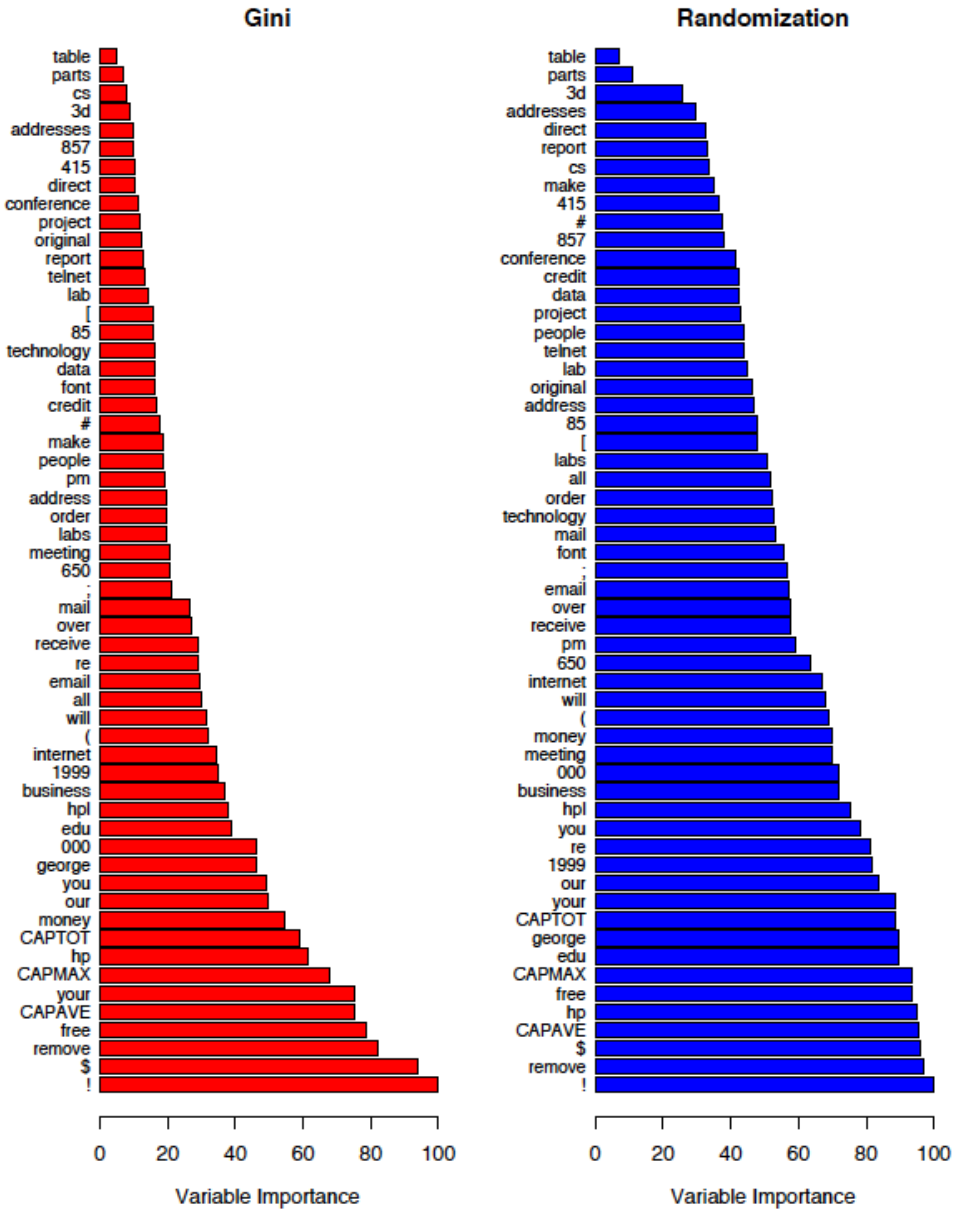

FIGURE 15.5. *Variable importance plots for a classification random forest grown on the* `spam` *data. The left plot bases the importance on the Gini splitting index, as in gradient boosting. The rankings compare well with the rankings produced by gradient boosting (Figure 10.6 on page 354). The right plot uses* OOB *randomization to compute variable importances, and tends to spread the importances more uniformly.*

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Random Forest – noise resistance

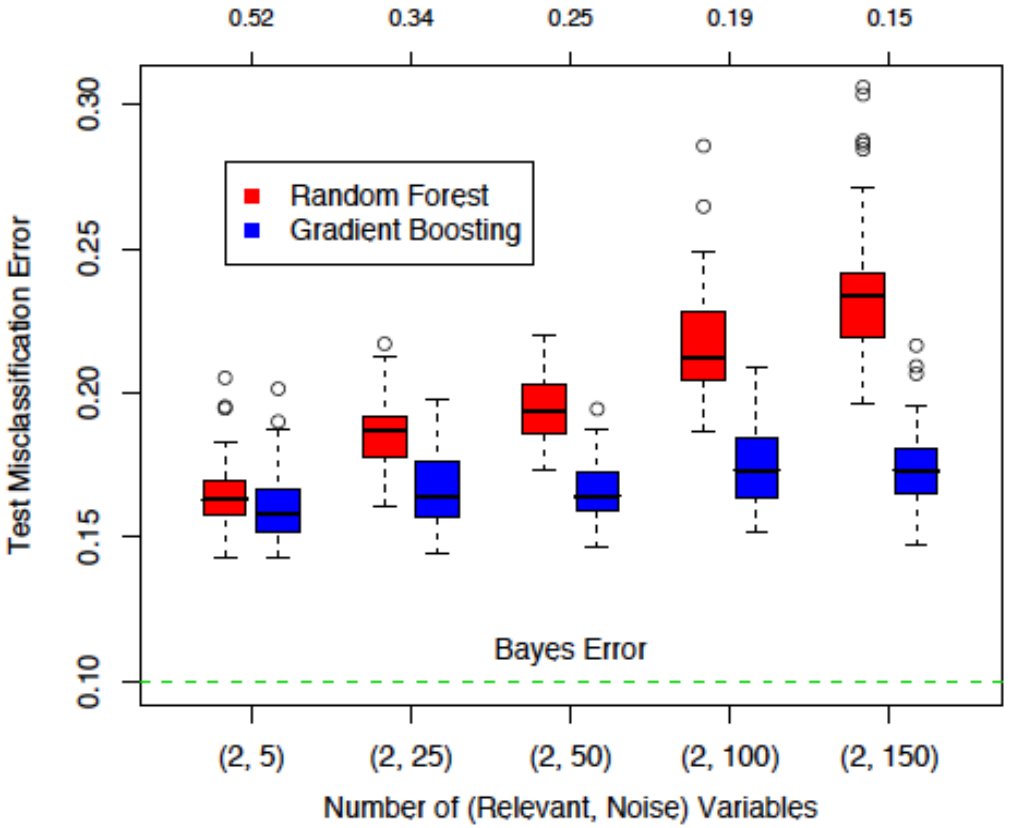

FIGURE 15.7. *A comparison of random forests and gradient boosting on problems with increasing numbers of noise variables. In each case the true decision boundary depends on two variables, and an increasing number of noise variables are included. Random forests uses its default value* $m = \sqrt{p}$. *At the top of each pair is the probability that one of the relevant variables is chosen at any split. The results are based on* 50 *simulations for each pair, with a training sample of* 300, *and a test sample of* 500.

When the number of variables is large, but the fraction of relevant variables small, random forests are likely to perform worse with small *m.*

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Overfitting

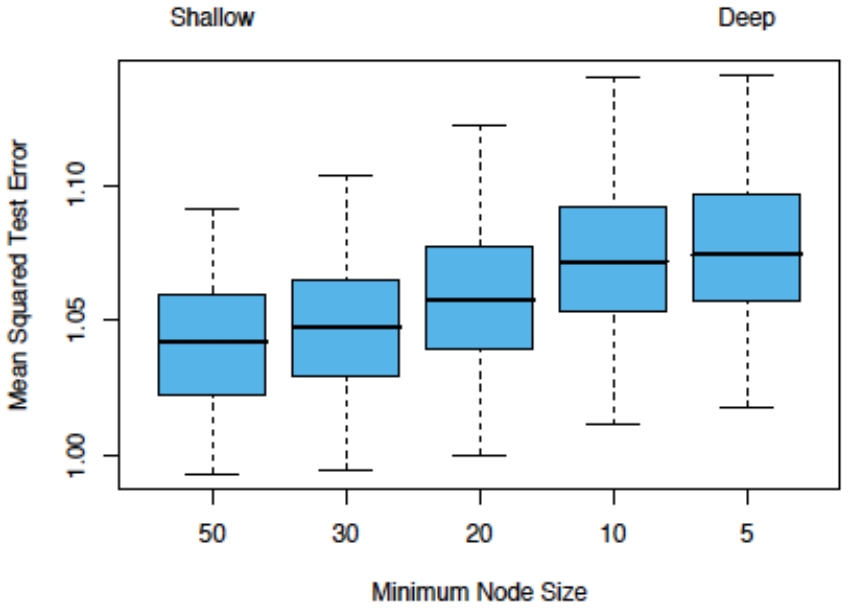

FIGURE 15.8. *The effect of* tree size *on the error in random forest regression. In this example, the true surface was additive in two of the* 12 *variables, plus additive unit-variance Gaussian noise. Tree depth is controlled here by the minimum node size; the smaller the minimum node size, the deeper the trees.*

- Same as bagging, increasing *B* does not cause the random forest to overfit.
- In regression, small gains in performance can be obtained by controlling the depths of the individual trees grown.

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Random Forest vs Bagging vs Boosting

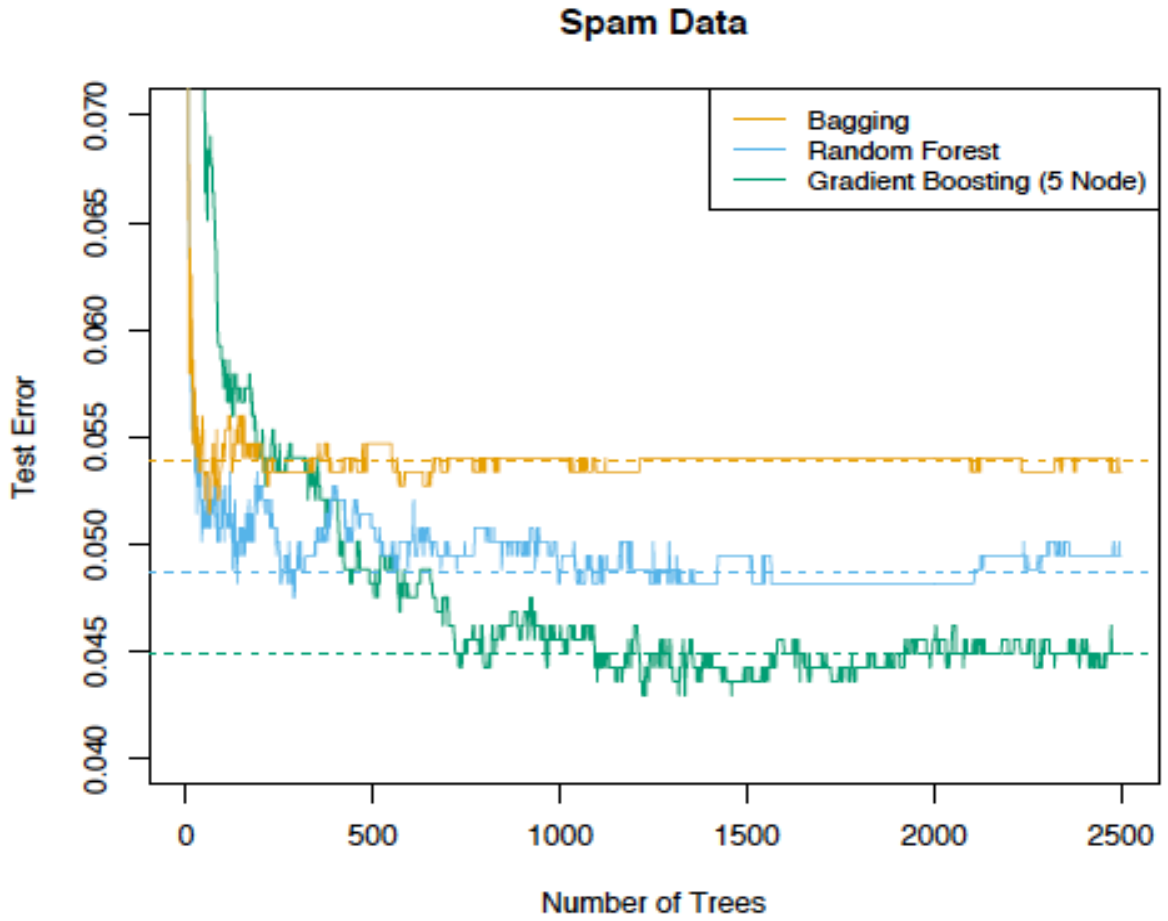

FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of 1536).*

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

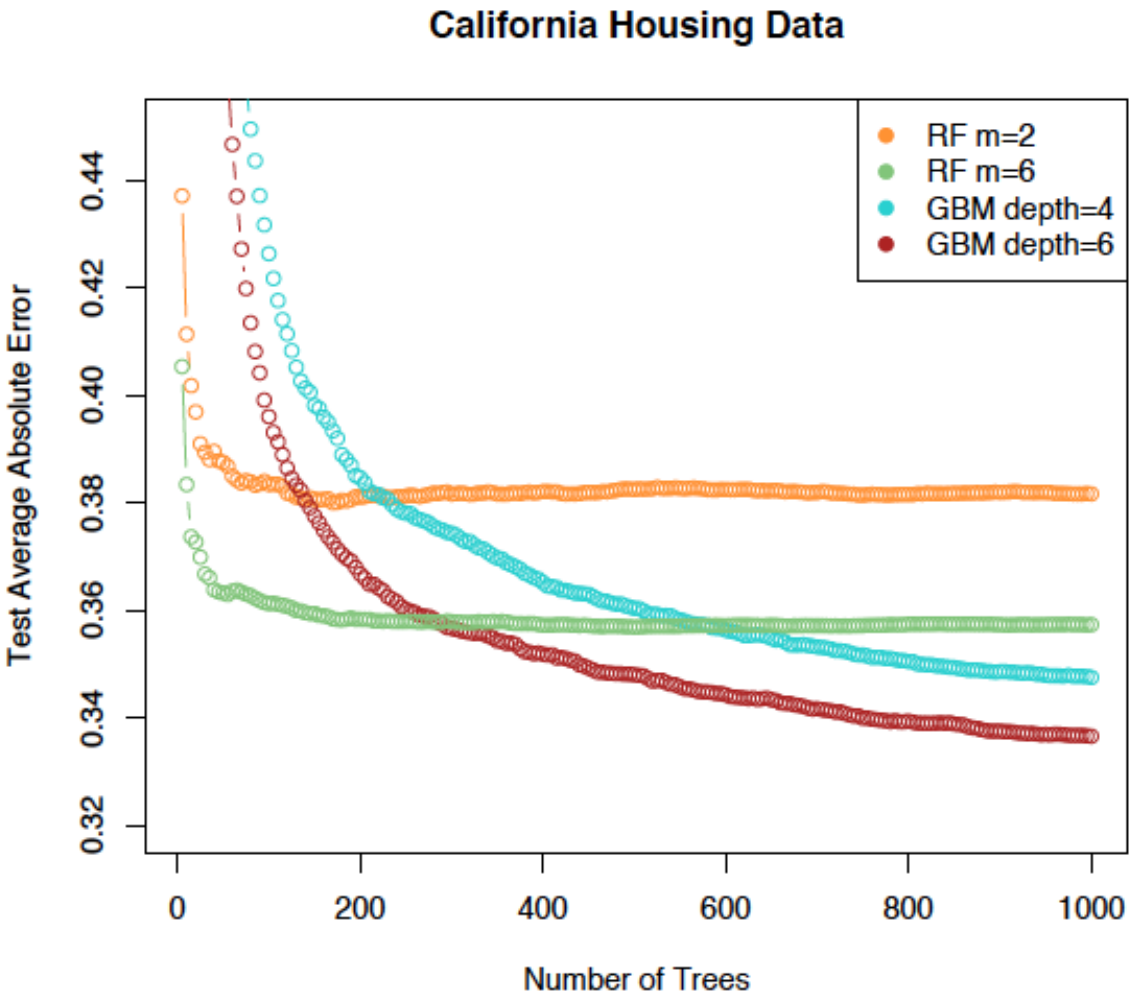

FIGURE 15.3. *Random forests compared to gradient boosting on the California housing data. The curves represent mean absolute error on the test data as a function of the number of trees in the models. Two random forests are shown, with* $m = 2$ *and* $m = 6$*. The two gradient boosted models use a shrinkage parameter* $\nu = 0.05$ *in (10.41), and have interaction depths of* 4 *and* 6*. The boosted models outperform random forests.*

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

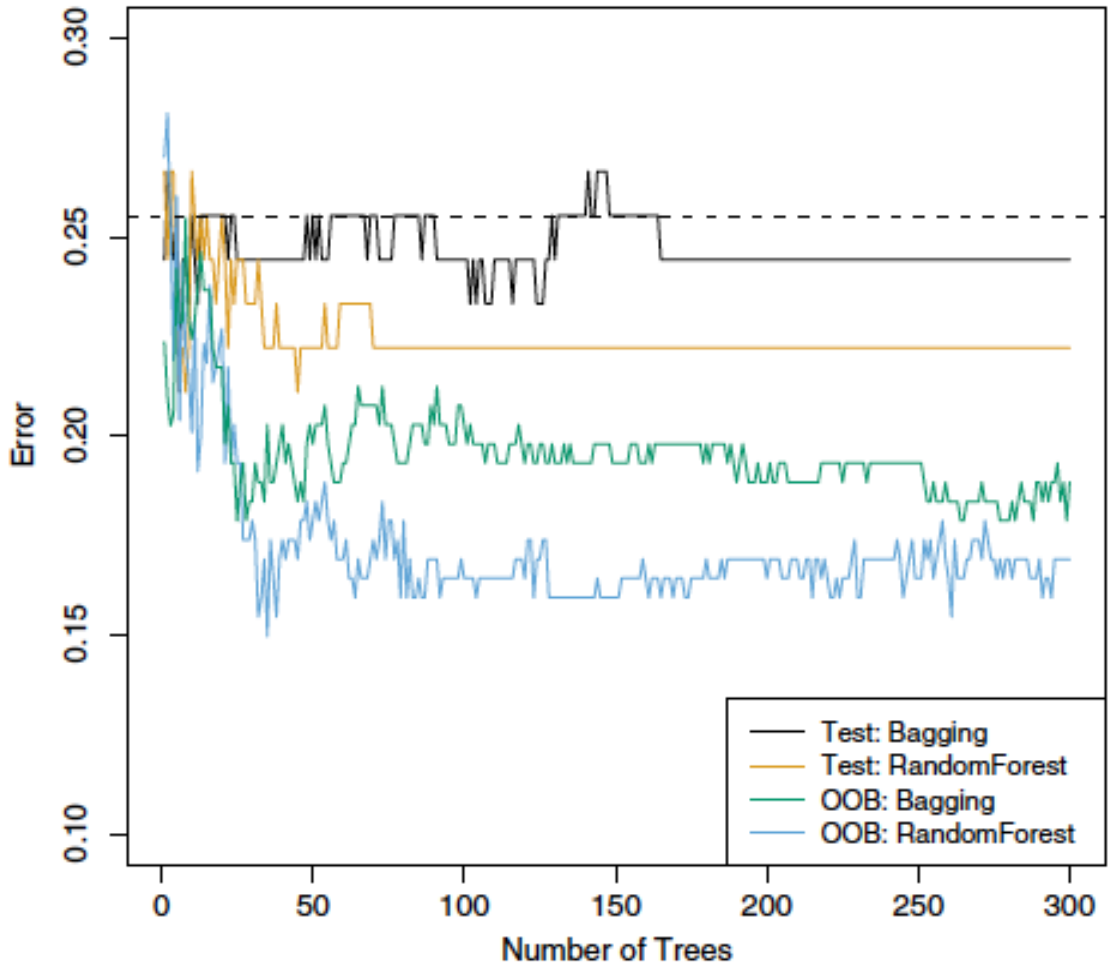

FIGURE 8.8. *Bagging and random forest results for the* `Heart` *data. The test error (black and orange) is shown as a function of* B*, the number of bootstrapped training sets used. Random forests were applied with* $m = \sqrt{p}$*. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.*

On many problems, random forests outperform bagging and are comparable to boosting. But in some problems, boosting outperforms random forests.

Figure from *An Introduction to Statistical Learning: with Applications in R* by James, Witten, Hastie, and Tibshirani, 2013.

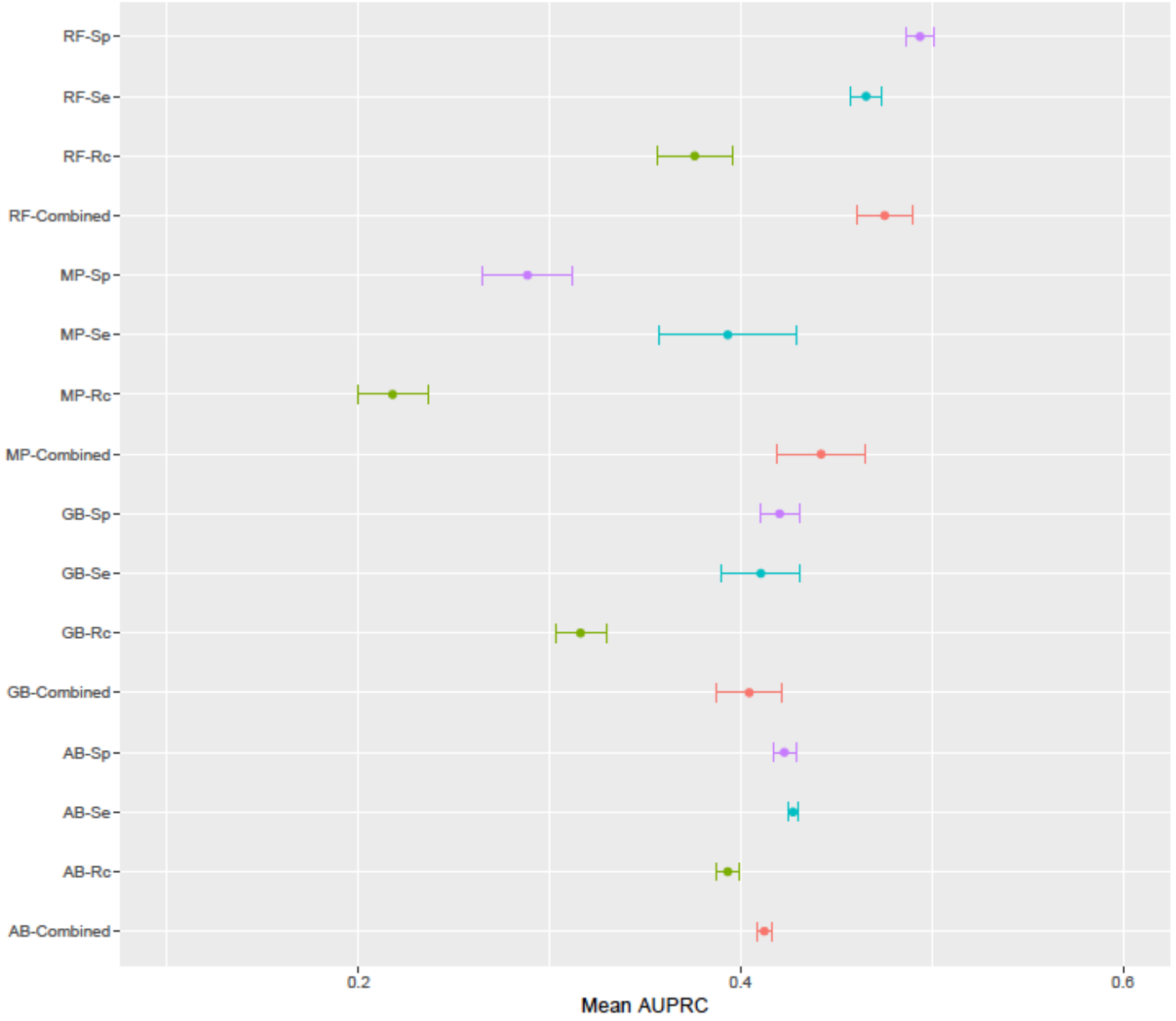

Figure 4 Mean AUPRC per model. The dot represents the mean AUPRC and bars represent standard error. Colour indicates the training data used: Red, Combined data, Green, *R. capsulatus* data; Blue, *S. enterica* data; Purple, *S. pyogenes* data. Classifiers are indicated by AB, adaptive boosting; GB, gradient boosting; MP, multilayer perceptron; RF, random forest.

Random Forest and KNN

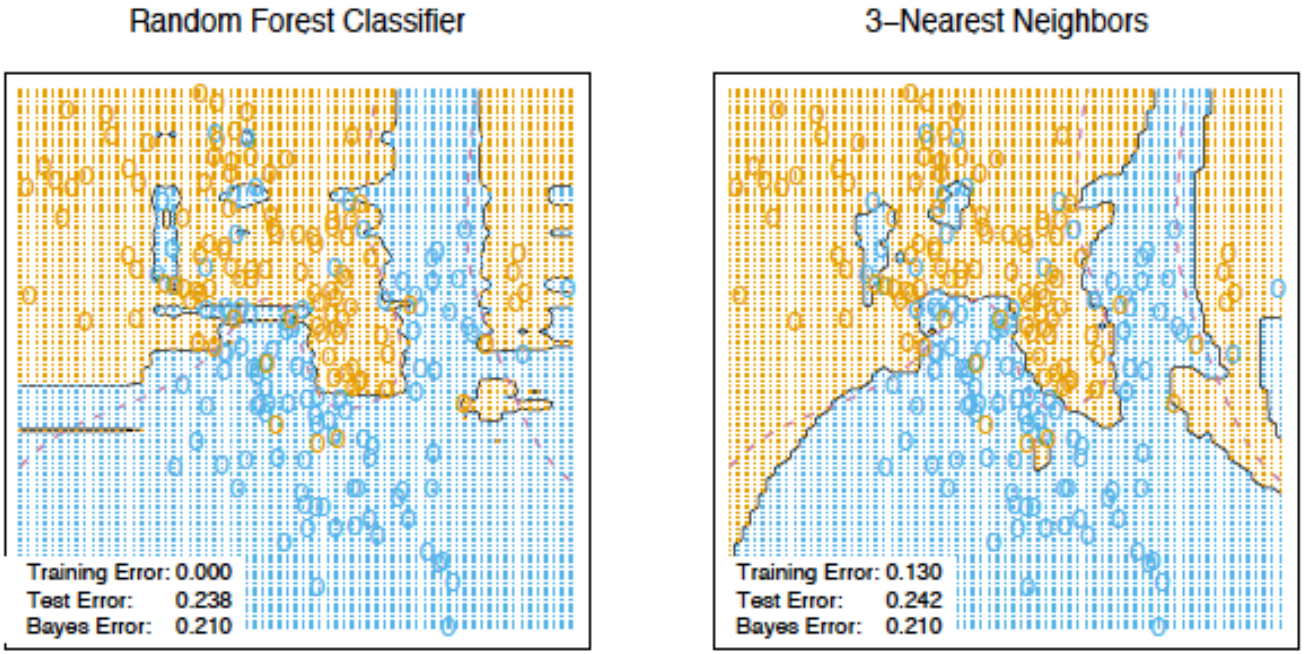

FIGURE 15.11. *Random forests versus 3-NN on the mixture data. The axis-oriented nature of the individual trees in a random forest lead to decision regions with an axis-oriented flavor.*

- Interestingly, random forests learn decision boundaries similar to those learnt by weighted KNN.
- As each tree is grown to maximal size, the tree-growing algorithm finds a "best" path to each observation. The averaging process assigns weights to those observations *close* to the target point.

Figure from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

By now, you should be able to...

- explain the rationale behind random forests, how random forests work, and why they are an improvement over bagging.
- understand the OOB error estimate, and the variable importance calculation.
- list random forests characteristics with respect to overfitting, robustness to noise and parameter tuning.
- discuss the similarity between random forests and KNN