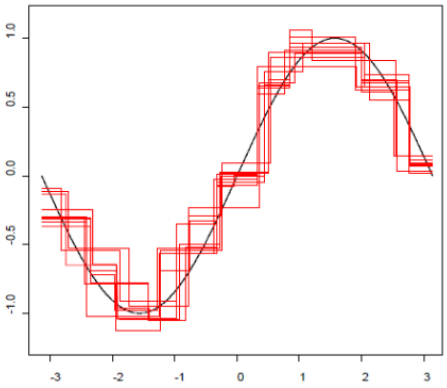
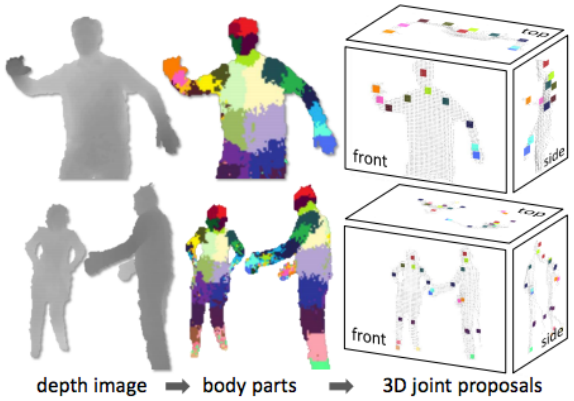
Tufts COMP 135: Introduction to Machine Learning
https://www.cs.tufts.edu/comp/135/2020f/

Random Forests

and other Ensembles of Independent Predictors





Prof. Mike Hughes

Many slides attributable to:
Liping Liu and Roni Khardon (Tufts)
T. Q. Chen (UW),
James, Witten, Hastie, Tibshirani (ISL/ESL books)

Ensembles: Unit Objectives

Big idea: We can improve performance by aggregating decisions from MANY predictors

- Today: Predictors are **Independently** Trained
  - Using bootstrap samples of examples: "Bagging"
  - Using random subsets of features
  - Exemplary method: Random Forest / ExtraTrees
- Next class: Predictors are **Sequentially** Trained
  - Each successive predictor "boosts" performance
  - Exemplary method: XGBoost

Motivating Example

3 binary classifiers

- Model predictions as independent random variables
- Each one is correct 70% of the time

What is chance that majority vote is correct?

Motivating Example

5 binary classifiers

- Model predictions as independent random variables
- Each one is correct 70% of the time

What is chance that majority vote is correct?

Motivating Example

101 binary classifiers

- Model predictions as independent random variables
- Each one is correct 70% of the time

What is chance that majority vote is correct?

Key Idea: Diversity

- Vary the **training data**

Bootstrap Sampling

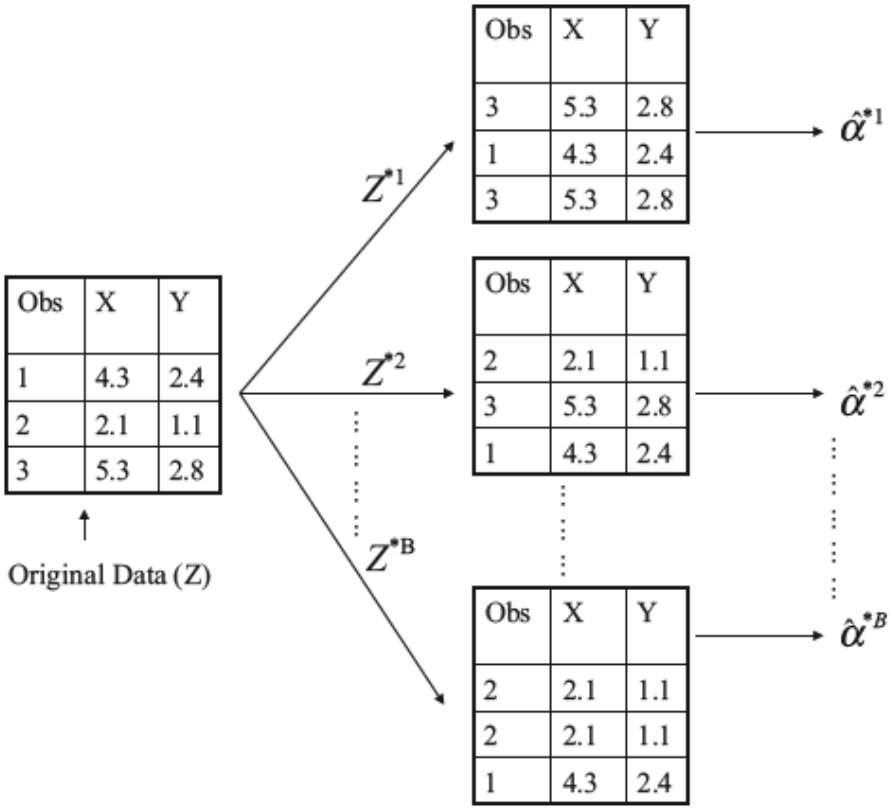


FIGURE 5.11. *A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α.*

Bootstrap Sampling in Python

```
def bootstrap_sample(x_NF, random_state=np.random):
    N = x_NF.shape[0]
    row_ids = random_state.choice(np.arange(N), size=N, replace=True)
    return x_NF[row_ids].copy()
```

```
In [9]: x_NF
Out[9]:
array([[ 4.3,  2.4],
       [ 2.1,  1.1],
       [ 5.3,  2.8]])

In [10]: bootstrap_sample(x_NF)
Out[10]:
array([[ 2.1,  1.1],
       [ 2.1,  1.1],
       [ 2.1,  1.1]])
```

```
In [11]: bootstrap_sample(x_NF)
Out[11]:
array([[ 5.3,  2.8],
       [ 5.3,  2.8],
       [ 5.3,  2.8]])

In [12]: bootstrap_sample(x_NF)
Out[12]:
array([[ 5.3,  2.8],
       [ 5.3,  2.8],
       [ 4.3,  2.4]])
```

Bootstrap Aggregation: BAgg-ing

- Draw B "replicas" of training set
  - Use bootstrap sampling with replacement

- Make prediction by **averaging**

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

Regression Example: 1 tree

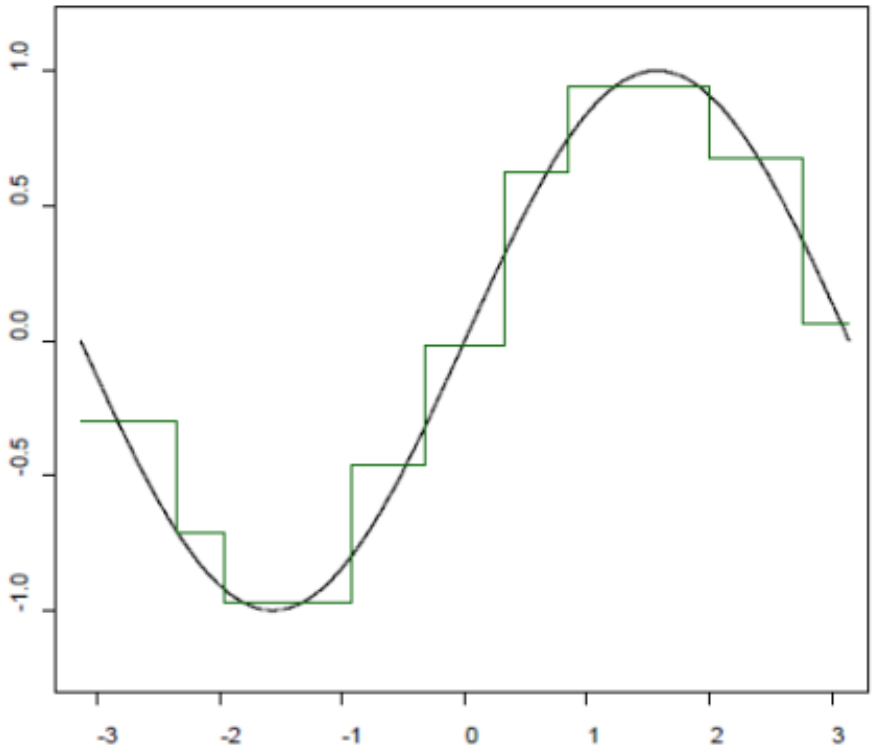
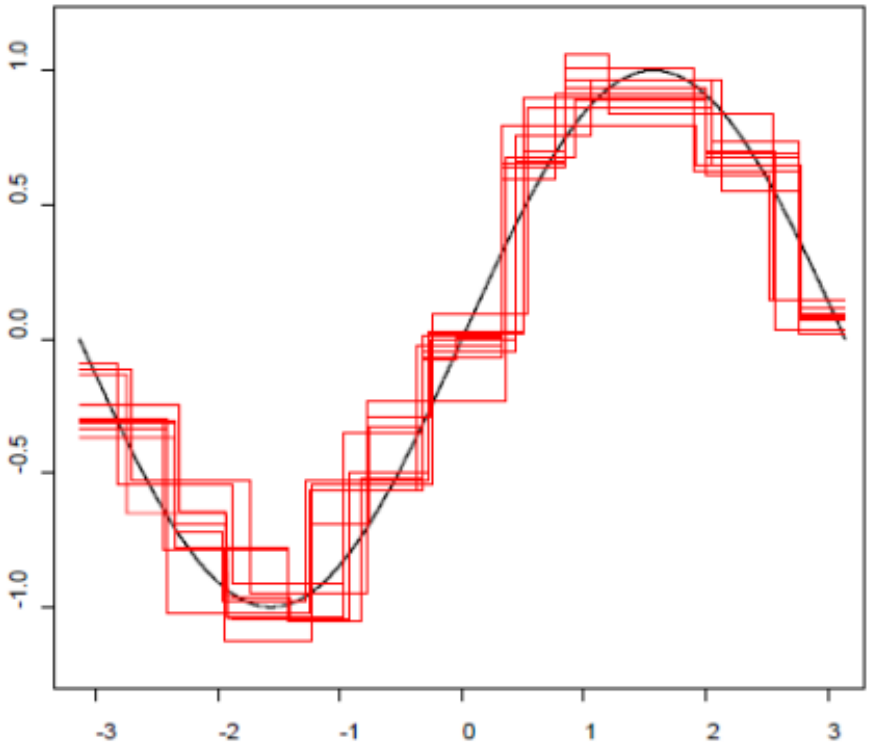


Image Credit: Adele Cutler's slides

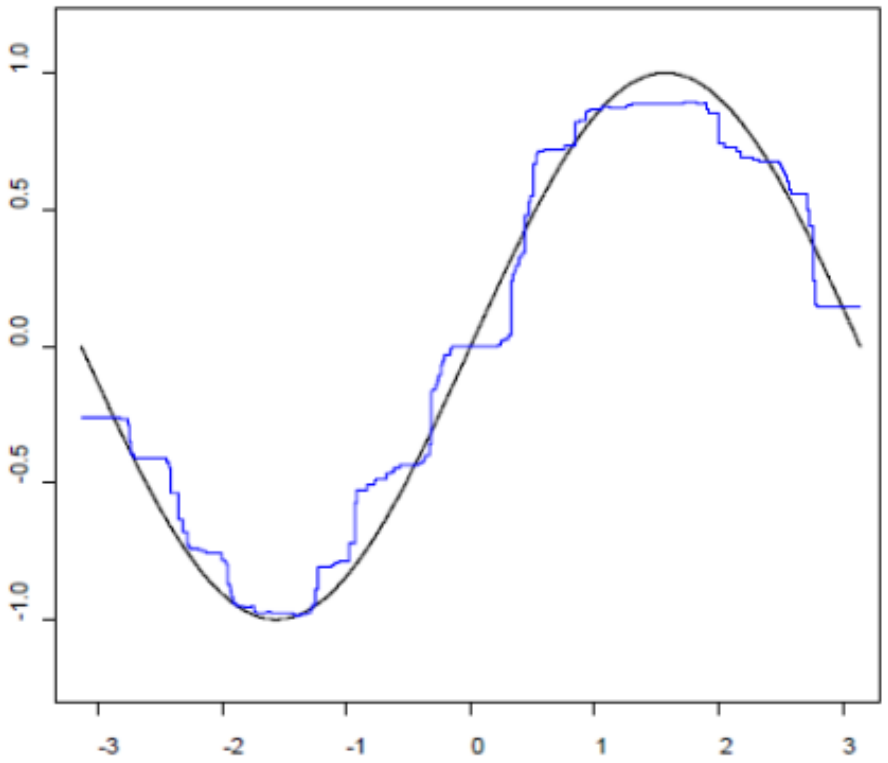
Regression Example: 10 trees



The solid black line is the ground-truth,
Red lines are predictions of single regression trees

Image Credit: Adele Cutler's slides

Regression Average of 10 trees



The solid black line is the ground-truth,
The blue line is the prediction of the average of 10 regression trees

Image Credit: Adele Cutler's slides

Binary Classification

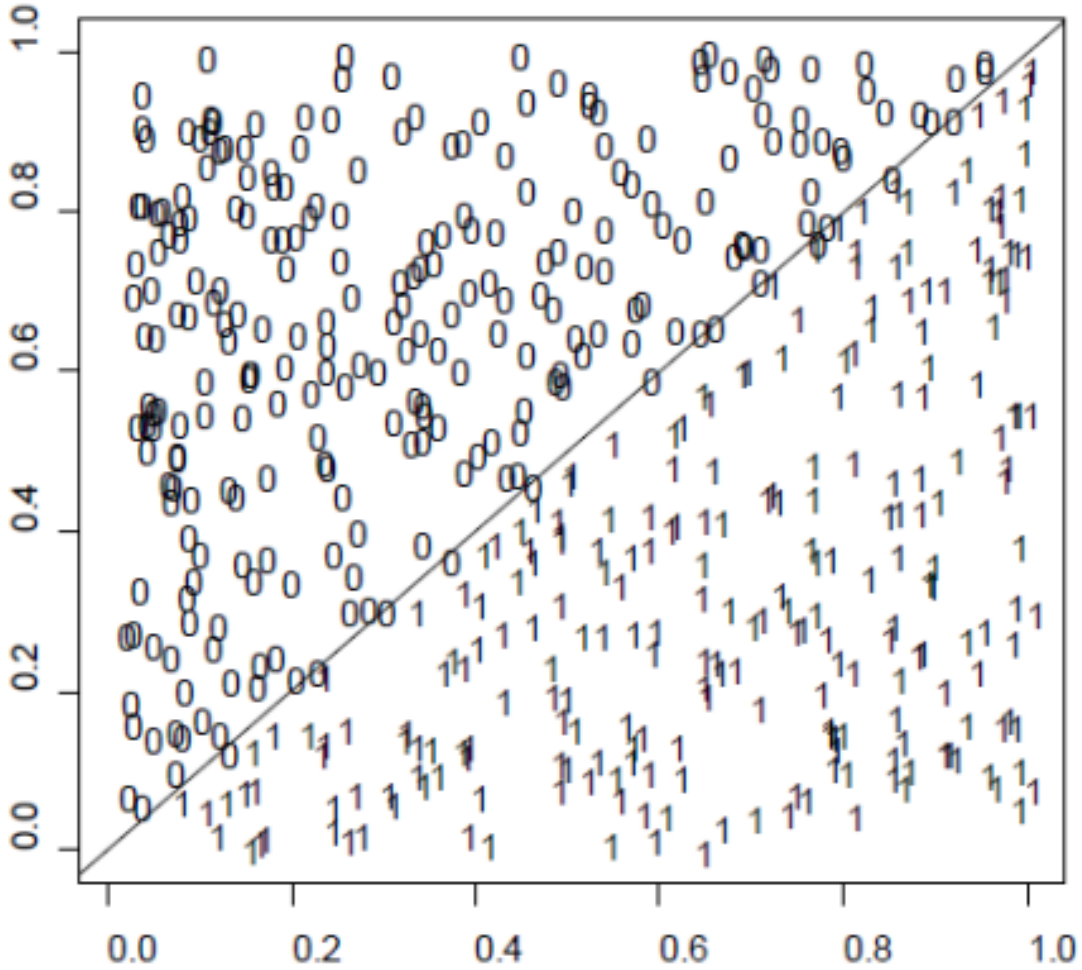


Image Credit: Adele Cutler's slides

Decision Boundary: 1 tree

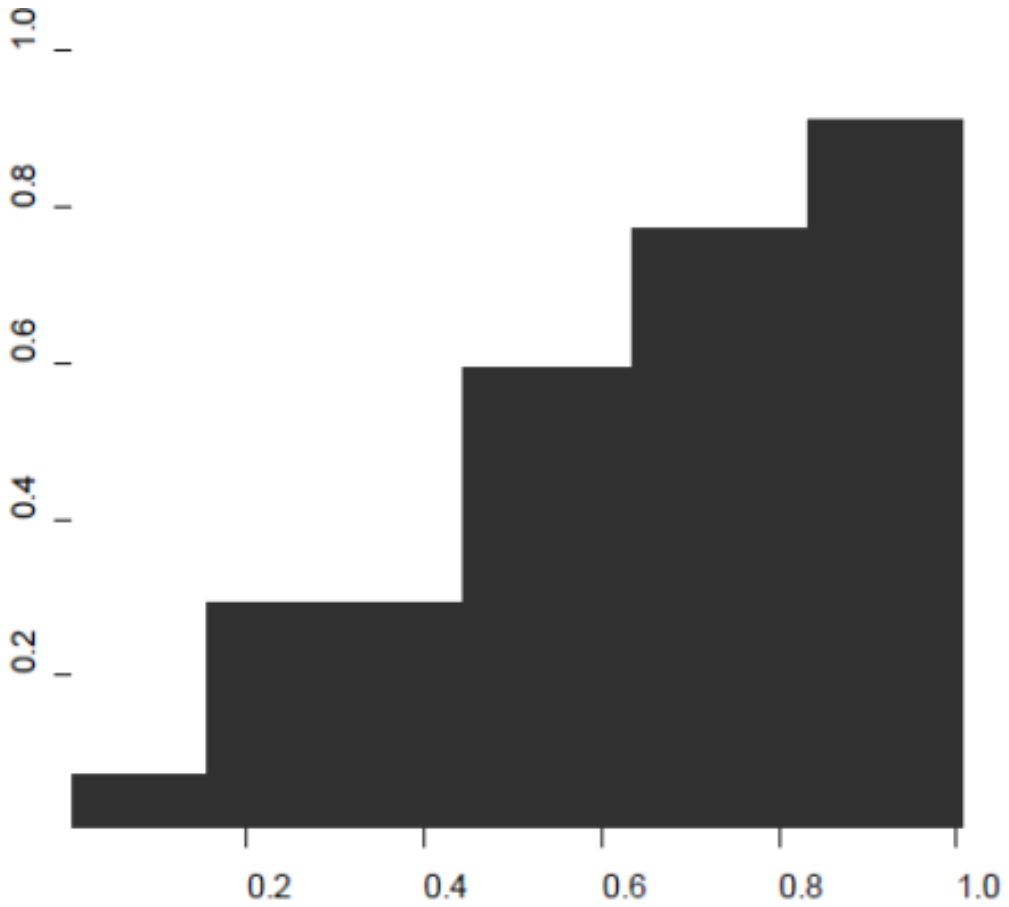


Image Credit: Adele Cutler's slides

Decision boundary: 25 trees

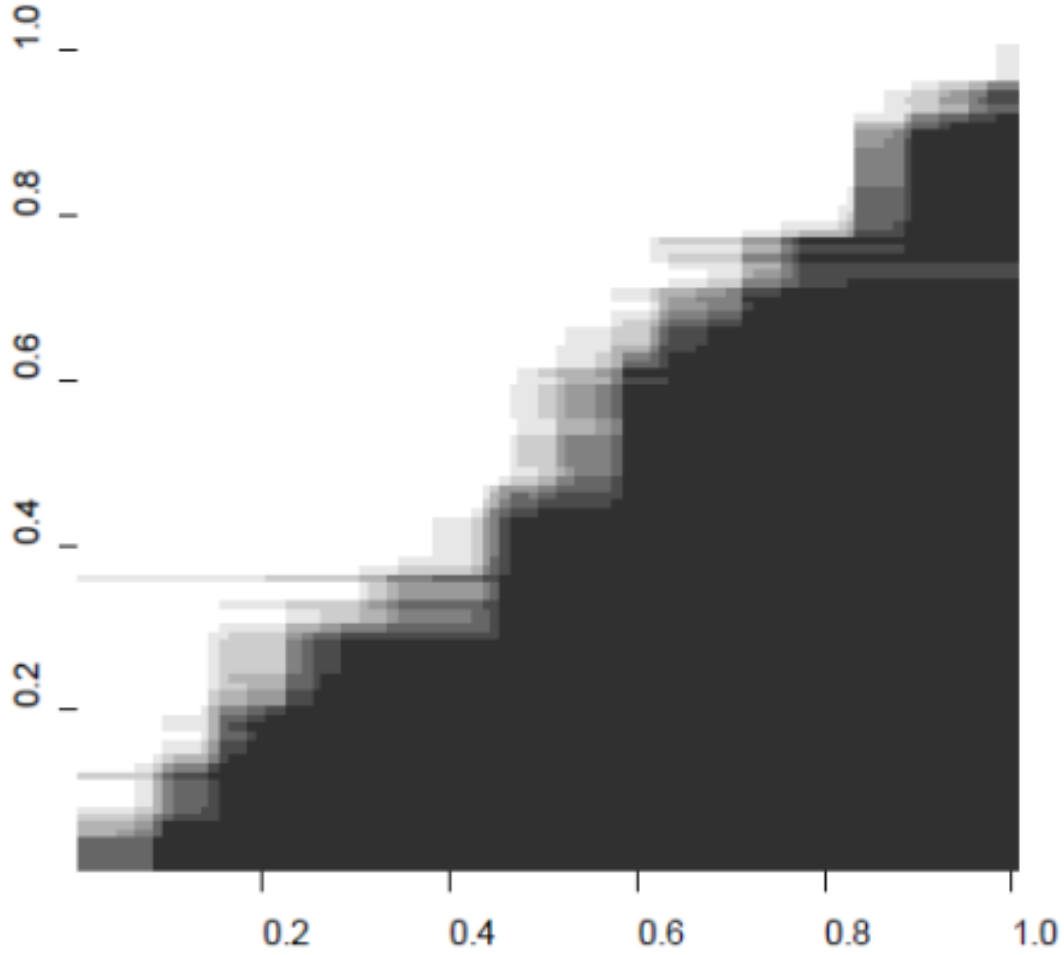


Image Credit: Adele Cutler's slides

Average over 25 trees

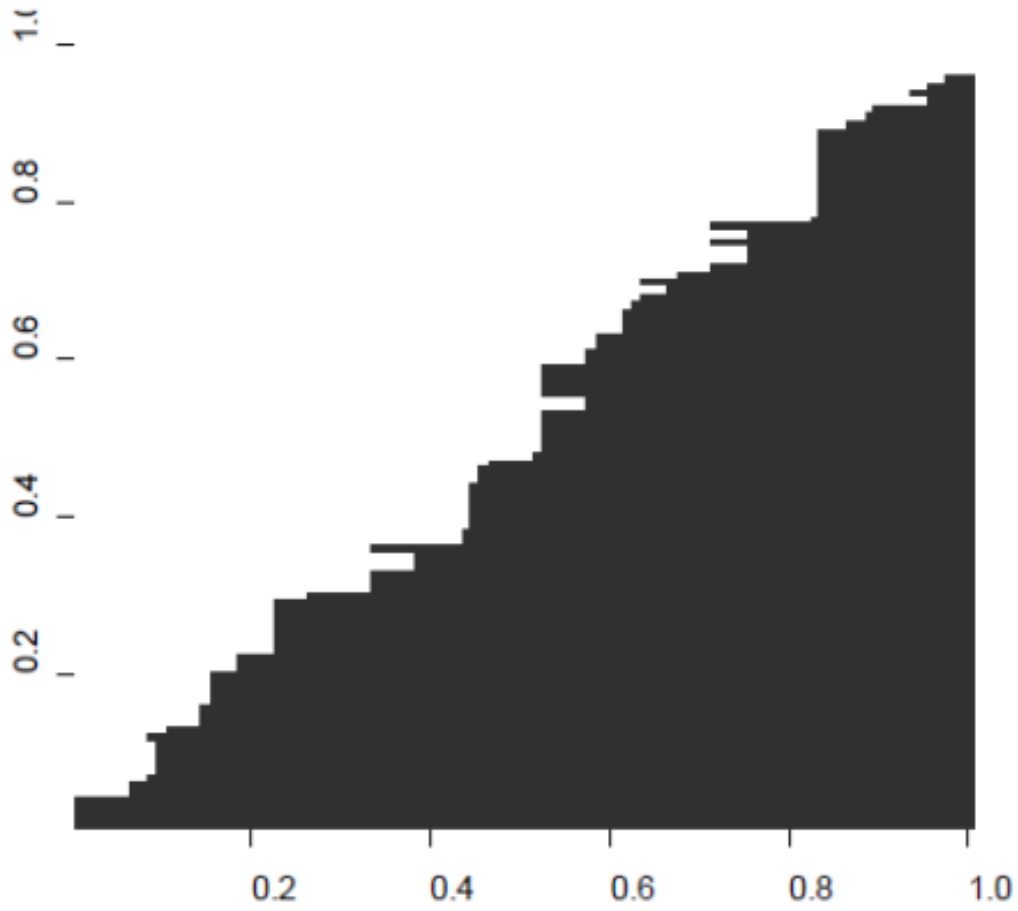


Image Credit: Adele Cutler's slides

Variance of averages

- Given B independent observations

$$z_1, z_2, \ldots z_B$$

  - Each one has variance v

- Compute the mean of the B observations

$$\bar{z} = \frac{1}{B} \sum_{b=1}^{B} z_b$$

  - What is variance of this estimator?

Why Bagging Works: Reduce Variance!

- Flexible learners applied to small datasets have high variance w.r.t. the data distribution
  - Small change in training set -> big change in predictions on heldout set

- Bagging decreases heldout error by decreasing the variance of predictions

- Bagging can be applied to **any** base classifiers/regressors

Another Idea for Diversity

- Vary the **features**

Random Forest

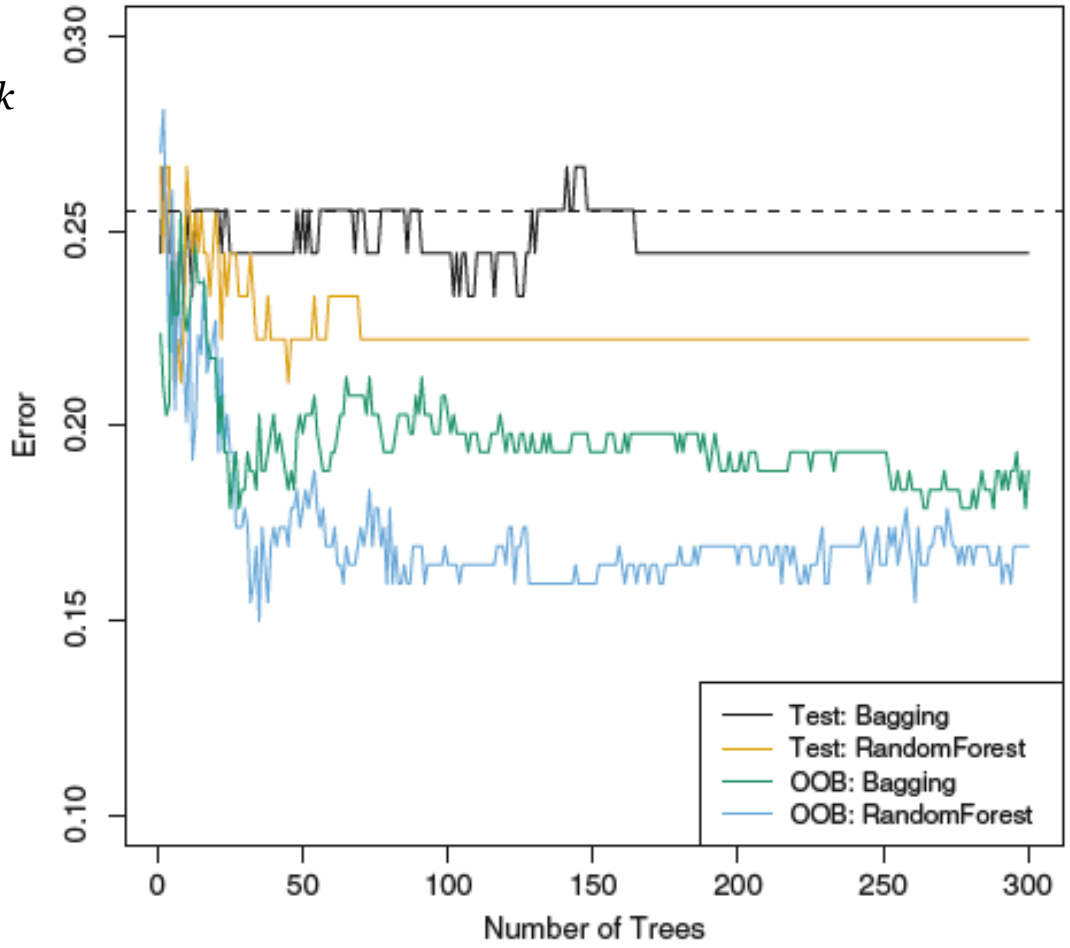
Combine example diversity AND feature diversity

For t = 1 to T (# trees):
Draw independent **bootstrap sample** of training set.
Greedy train tree on **random subsample** of features

For each node within a **maximum depth**:
Randomly select ***M*** features from ***F*** features
Find the best split among these ***M*** features

Average the trees to get predictions for new data.

Credit: ISL textbook



Single tree

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.*

Extremely Randomized Trees aka "ExtraTrees" in sklearn

***Speed**, example diversity, and feature diversity*

For t = 1 to T (# trees):

Draw independent **bootstrap sample** of training set.

Greedy train tree on **random subsample** of features

For each node within a **maximum depth**:

Randomly select ***m*** features from ***F*** features

~~Find the best split among ***M*** variables~~

Try 1 random split at each of M variables,

then select the best split of these

```
>>> from sklearn.model_selection import cross_val_score
>>> from sklearn.datasets import make_blobs
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.ensemble import ExtraTreesClassifier
>>> from sklearn.tree import DecisionTreeClassifier

>>> X, y = make_blobs(n_samples=10000, n_features=10, centers=100,
...     random_state=0)

>>> clf = DecisionTreeClassifier(max_depth=None, min_samples_split=2,
...     random_state=0)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores.mean()
0.98...

>>> clf = RandomForestClassifier(n_estimators=10, max_depth=None,
...     min_samples_split=2, random_state=0)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores.mean()
0.999...

>>> clf = ExtraTreesClassifier(n_estimators=10, max_depth=None,
...     min_samples_split=2, random_state=0)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores.mean() > 0.999
True
```

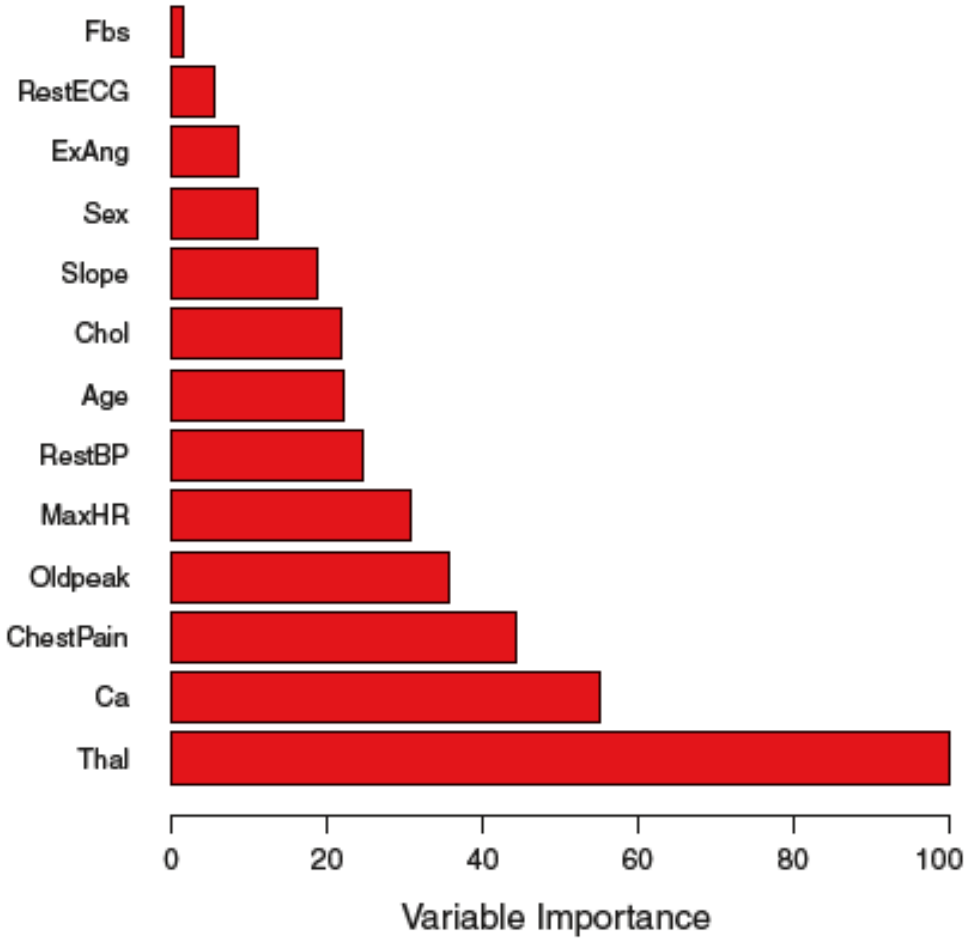
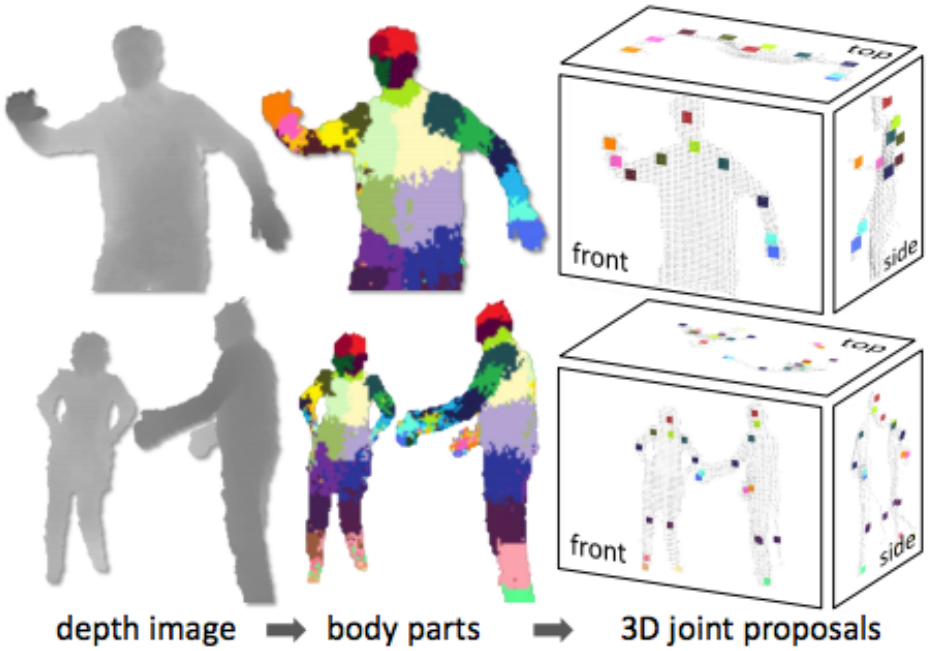


FIGURE 8.9. *A variable importance plot for the* `Heart` *data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.*

Applications of Random Forest in Industry

Real-Time Human Pose Recognition in Parts from Single Depth Images

Jamie Shotton Andrew Fitzgibbon Mat Cook Toby Sharp Mark Finocchio
Richard Moore Alex Kipman Andrew Blake
Microsoft Research Cambridge & Xbox Incubation



Microsoft Kinect RGB-D camera

How does the Kinect classify each pixel into a body part?

Real-Time Human Pose Recognition in Parts from Single Depth Images

Jamie Shotton Andrew Fitzgibbon Mat Cook Toby Sharp Mark Finocchio
Richard Moore Alex Kipman Andrew Blake
Microsoft Research Cambridge & Xbox Incubation

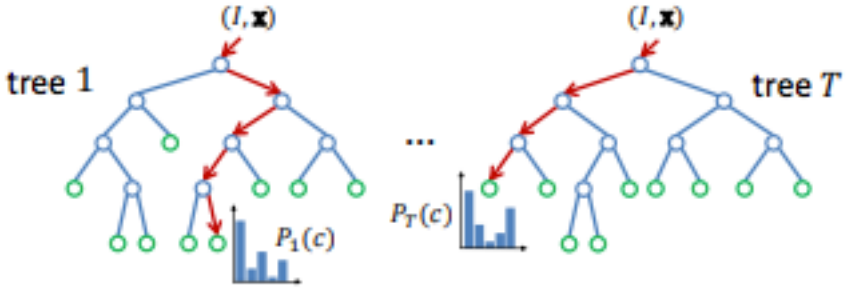


Figure 4. **Randomized Decision Forests.** A forest is an ensemble of trees. Each tree consists of split nodes (blue) and leaf nodes (green). The red arrows indicate the different paths that might be taken by different trees for a particular input.

3.3. Randomized decision forests

Randomized decision trees and forests [35, 30, 2, 8] have proven fast and effective multi-class classifiers for many tasks [20, 23, 36], and can be implemented efficiently on the GPU [34]. As illustrated in Fig. 4, a forest is an ensemble of T decision trees, each consisting of split and leaf nodes. Each split node consists of a feature f_θ and a threshold τ. To classify pixel $\mathbf{x}$ in image I, one starts at the root and repeatedly evaluates Eq. 1, branching left or right according to the comparison to threshold τ. At the leaf node reached in tree t, a learned distribution $P_t(c|I,\mathbf{x})$ over body part labels c is stored. The distributions are averaged together for all trees in the forest to give the final classification

$$P(c|I,\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I,\mathbf{x}) \,. \qquad (2)$$

Training. Each tree is trained on a different set of randomly synthesized images. A random subset of 2000 example pixels from each image is chosen to ensure a roughly even distribution across body parts. Each tree is trained using the following algorithm [20]:

1. Randomly propose a set of splitting candidates $\phi = (\theta, \tau)$ (feature parameters θ and thresholds τ).
2. Partition the set of examples $Q = \{(I,\mathbf{x})\}$ into left and right subsets by each ϕ:

$$Q_l(\phi) = \{\, (I,\mathbf{x}) \mid f_\theta(I,\mathbf{x}) < \tau \,\} \qquad (3)$$
$$Q_r(\phi) = Q \setminus Q_l(\phi) \qquad (4)$$

Summary: Ensembles of Independent Base Classifiers

- Average over independent base predictors
- Why it works: Reduce variance
- PRO
  - Often better heldout performance than base model
- CON
  - Training B separate models is expensive, but can be parallelized