Introduction of Some Ensemble Learning Methods

Random Forest, Boosted Trees and so on

Honglin Yu

March 24, 2014

Outline

Big Picture

In statistics and machine learning, ensemble methods use multiple models to obtain better predictive performance than could be obtained from any of the constituent models.

— Wikipedia

Big Picture

> In statistics and machine learning, ensemble methods use multiple models to obtain better predictive performance than could be obtained from any of the constituent models.
>
> — Wikipedia

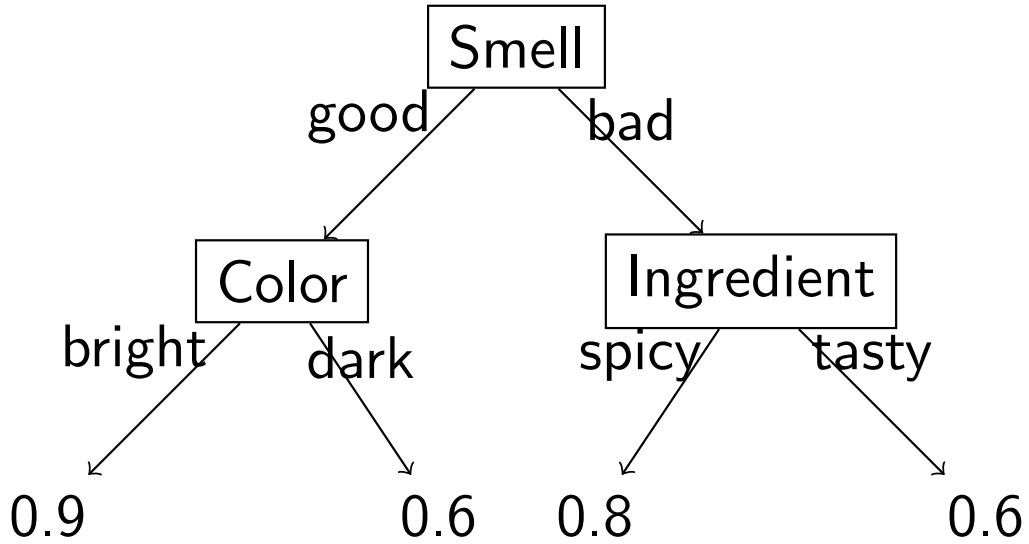
Build **house** (strong classifier) from **bricks** (weak classifiers)

Two questions:

1. What are the bricks? How to make them?
2. How to build house from bricks?

Classification and Regression Tree

Classification and Regression Tree (CART): Example and Inference



Inference:

1. Regression: the predicted response for an observation is given by the mean response of the training observations that belong to the same terminal node.
2. Classification: each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.

Classification and Regression Tree (CART): Learning

How to *learn* or *build* or *grow* a tree?

- Finding the optimal partitioning of the data is NP-complete
- Practical solution: greedy method

Famous Example: C4.5 algorithm

Growing the Tree (1)

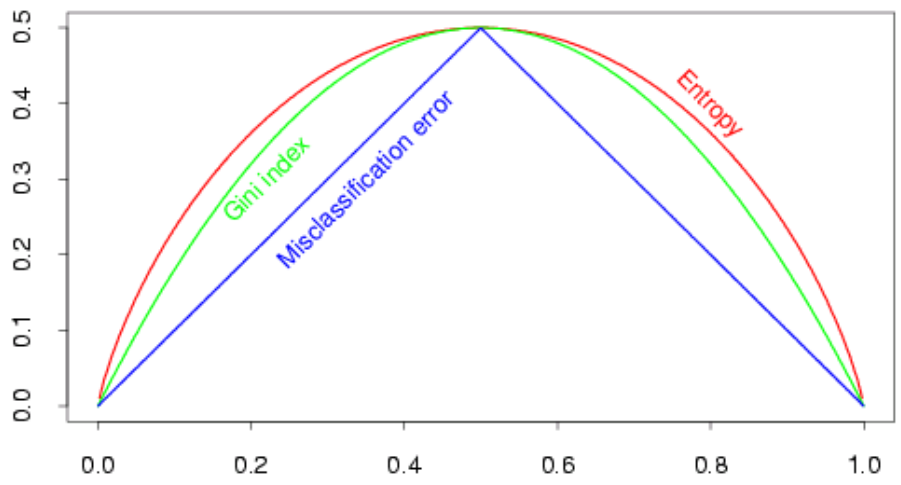
- Main procedure: (similar to building a *kd-tree*) recursively split a node of data $\{x_i\}$ into two nodes $(N_{X_j<s}, N_{X_j \geq s})$ by one dimension of feature X_j and a threshold s (greedy)

Growing the Tree (2)

- Split criterion (selecting X_j and s):
  - Regression: "minimizing RSS"

$$\min_{j,s} \sum_{i:(x_i)_j<s} (y_i - \bar{y}_1)^2 + \sum_{i:(x_i)_j \geq s} (y_i - \bar{y}_2)^2$$

  - Classification: minimizing weighted (by number of data in nodes) sum of
    - Classification error rate of 1 node: $1 - \max_k(p_{mk})$
    - Gini index of 1 node: $\sum_{k=1}^{K} p_{mk}(1 - p_{mk})$
    - Cross-entropy of 1 node: $-\sum_{k=1}^{K} p_{mk} \log p_{mk}$



Growing the Tree (3)

"When building a classification tree, either the Gini index or the cross-entropy are typically used to evaluate the quality of a particular split, since these two approaches are more sensitive to node purity than is the classification error rate. Any of these three approaches might be used when pruning the tree, but the classification error rate is preferable if prediction accuracy of the final pruned tree is the goal."

— ISLR book

Growing the Tree (3)

> "When building a classification tree, either the Gini index or the cross-entropy are typically used to evaluate the quality of a particular split, since these two approaches are more sensitive to node purity than is the classification error rate. Any of these three approaches might be used when pruning the tree, but the classification error rate is preferable if prediction accuracy of the final pruned tree is the goal."
>
> — ISLR book

- Stop criterion: for example, until "each terminal node has fewer than some minimum number of observations"

Pruning the Tree (1)

- Why: The growed tree is often too large
  - Overfitting
  - Sensitive to data fluctuations
- How: *Cost complexity pruning* or *weakest link pruning*

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

where T is a subtree of the full tree T_0. $|T|$ is the number of nodes of T.

Pruning the Tree (2)

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

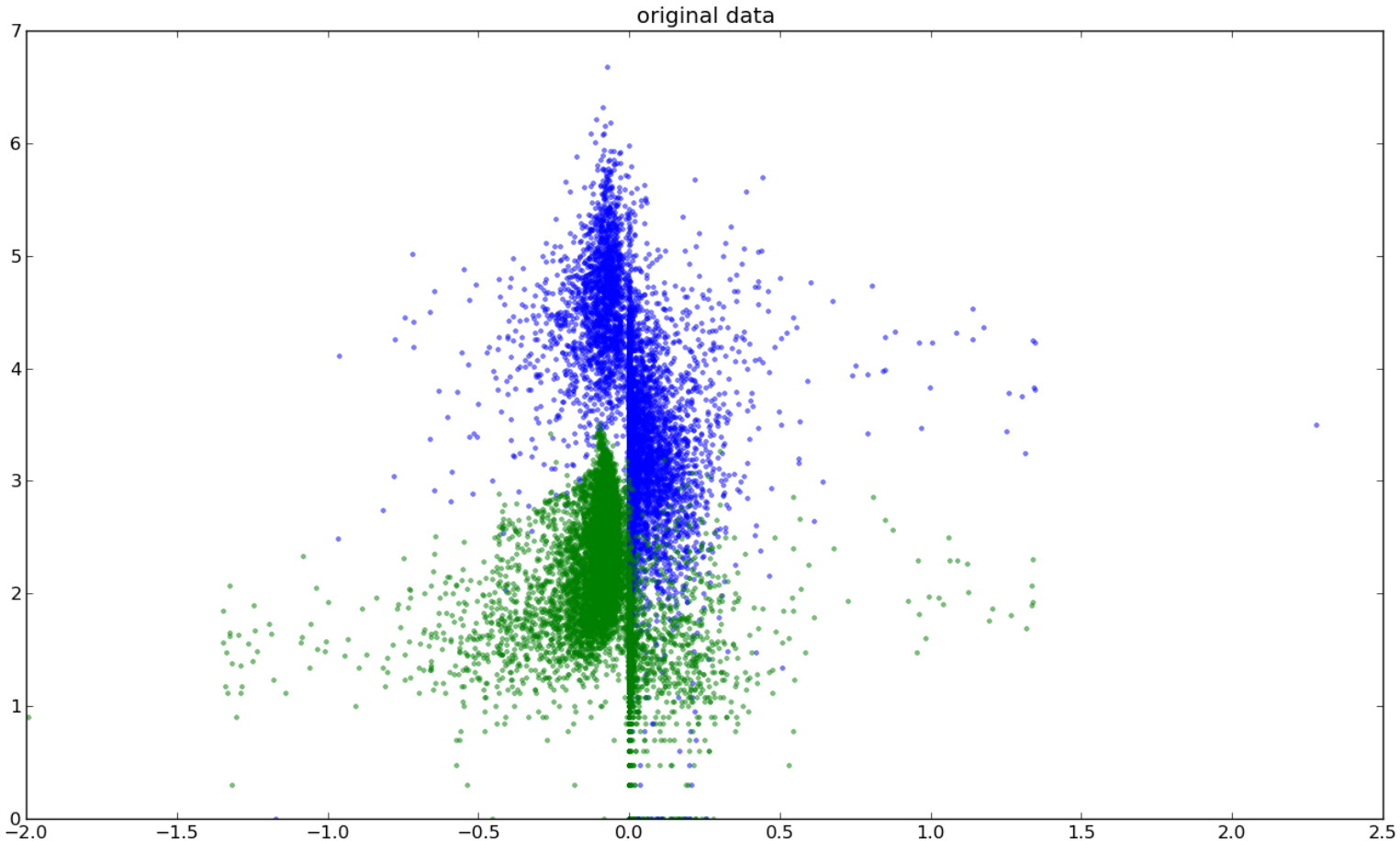
   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

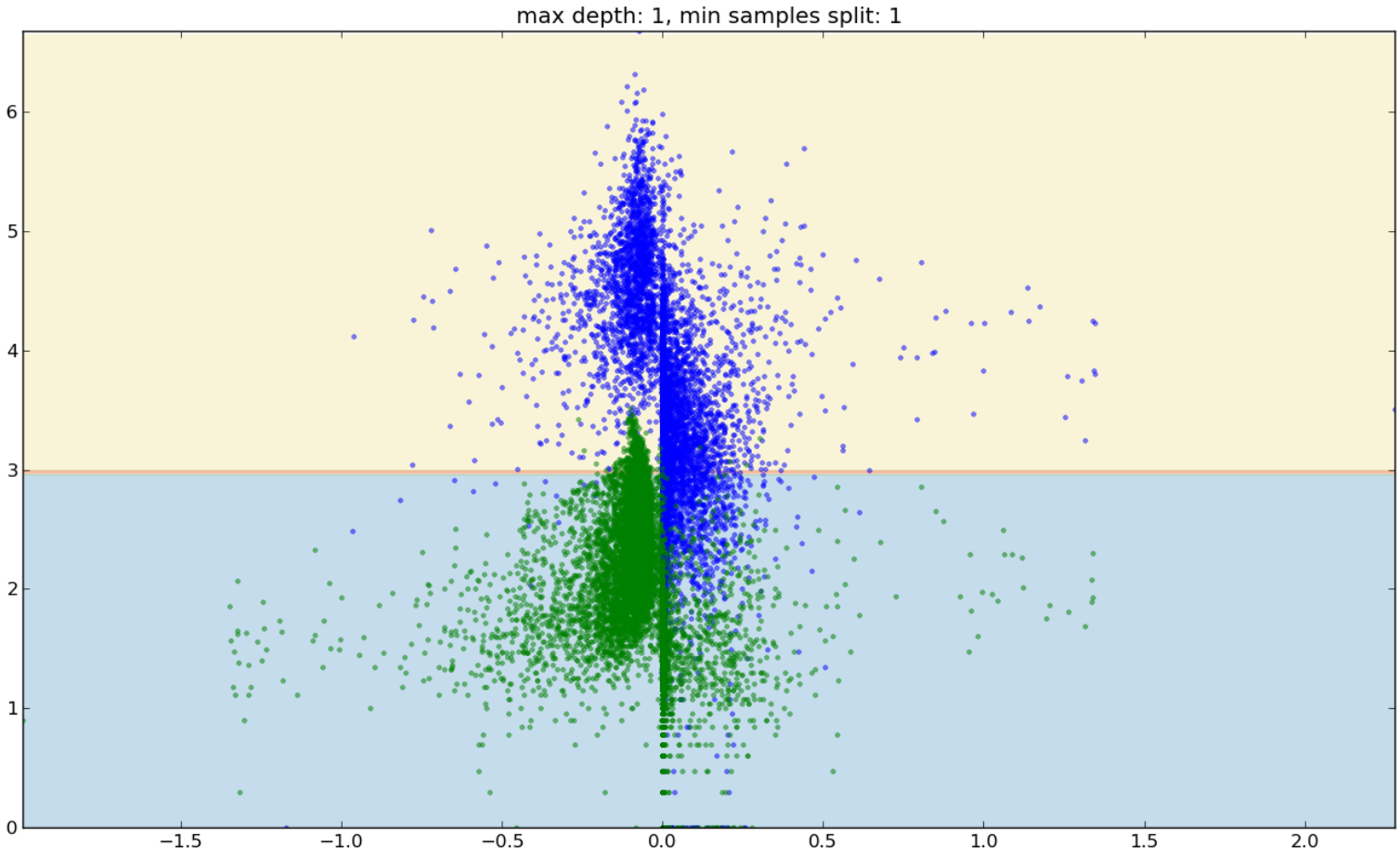
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

— ISLR book

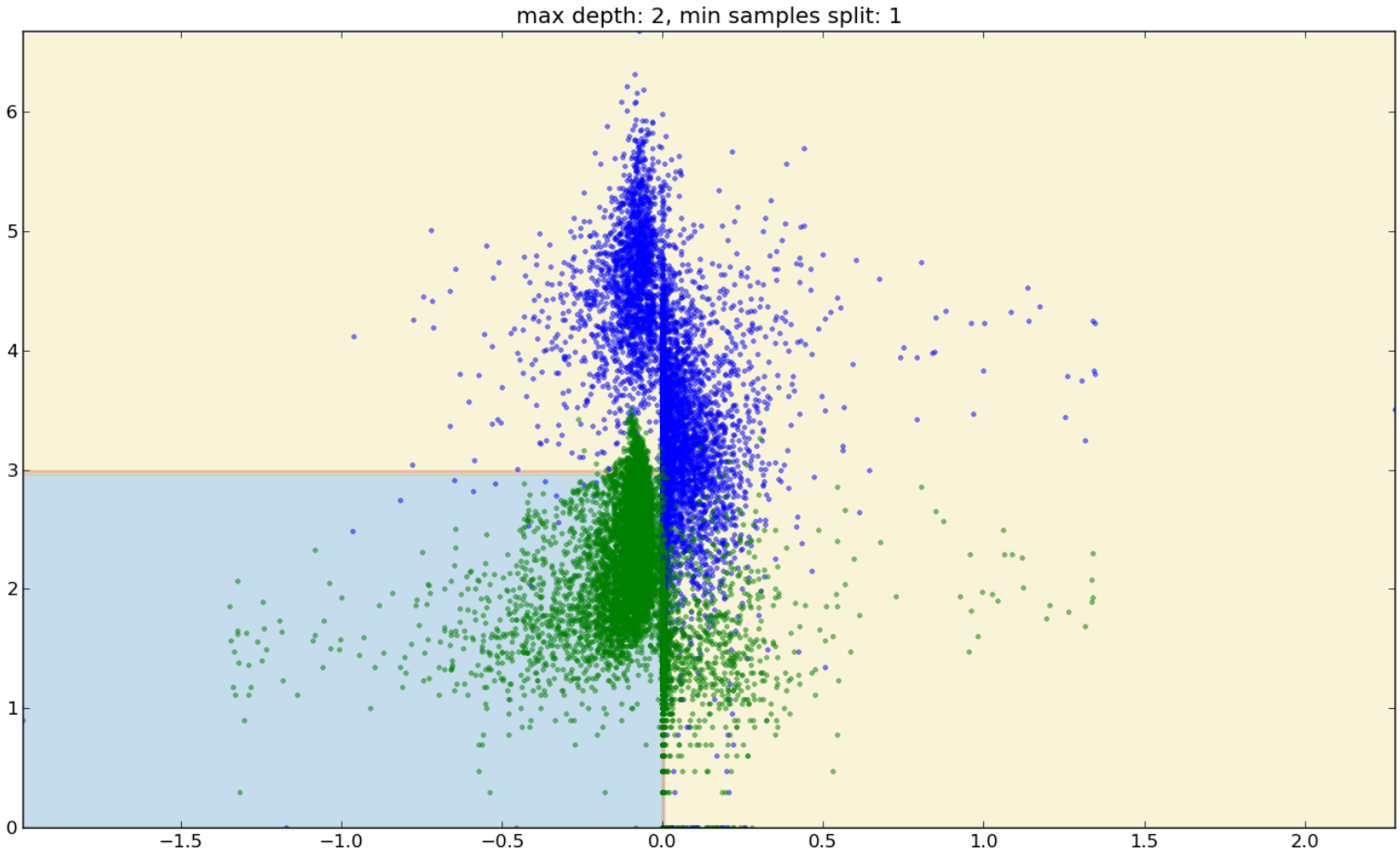
Example



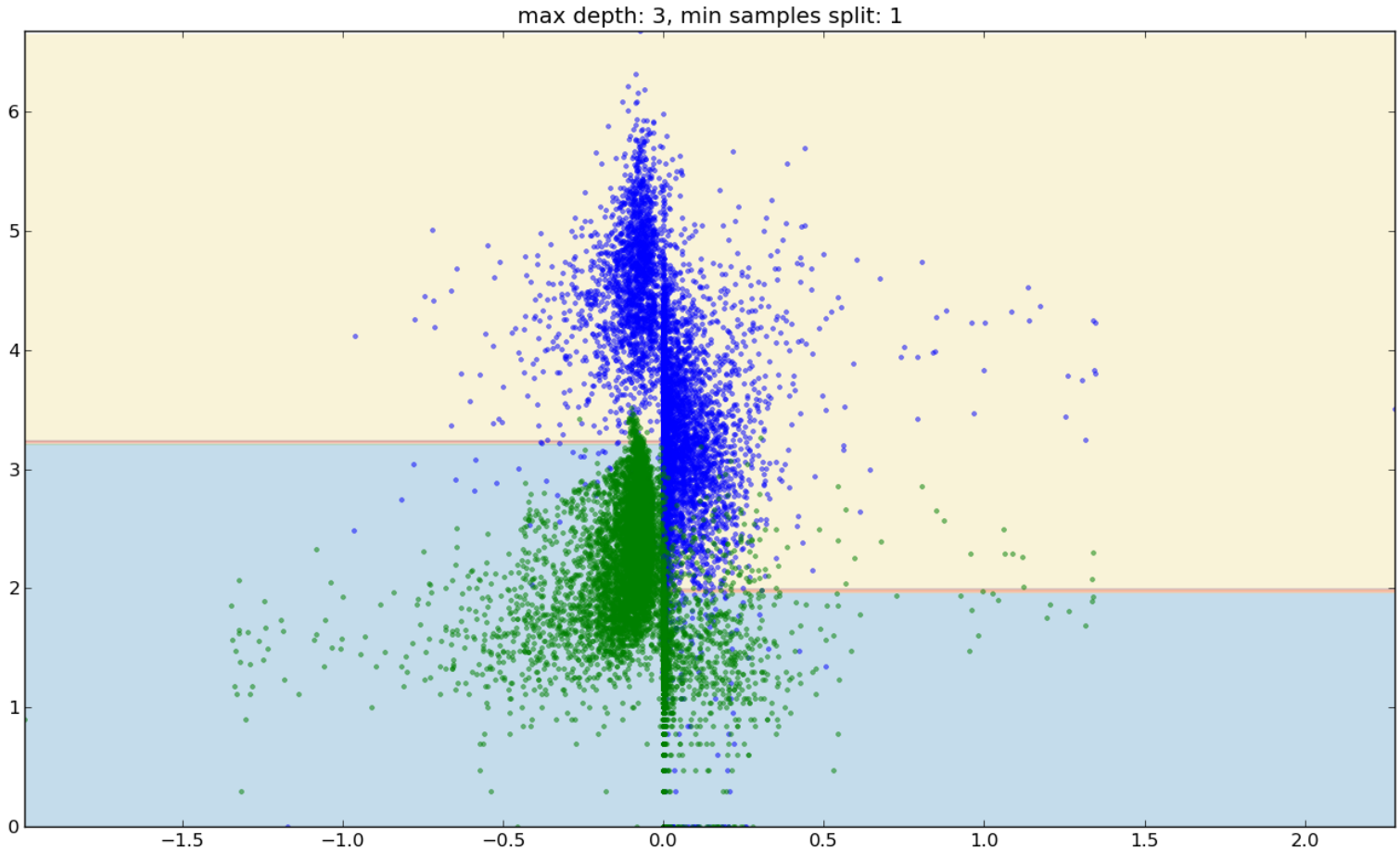
Example



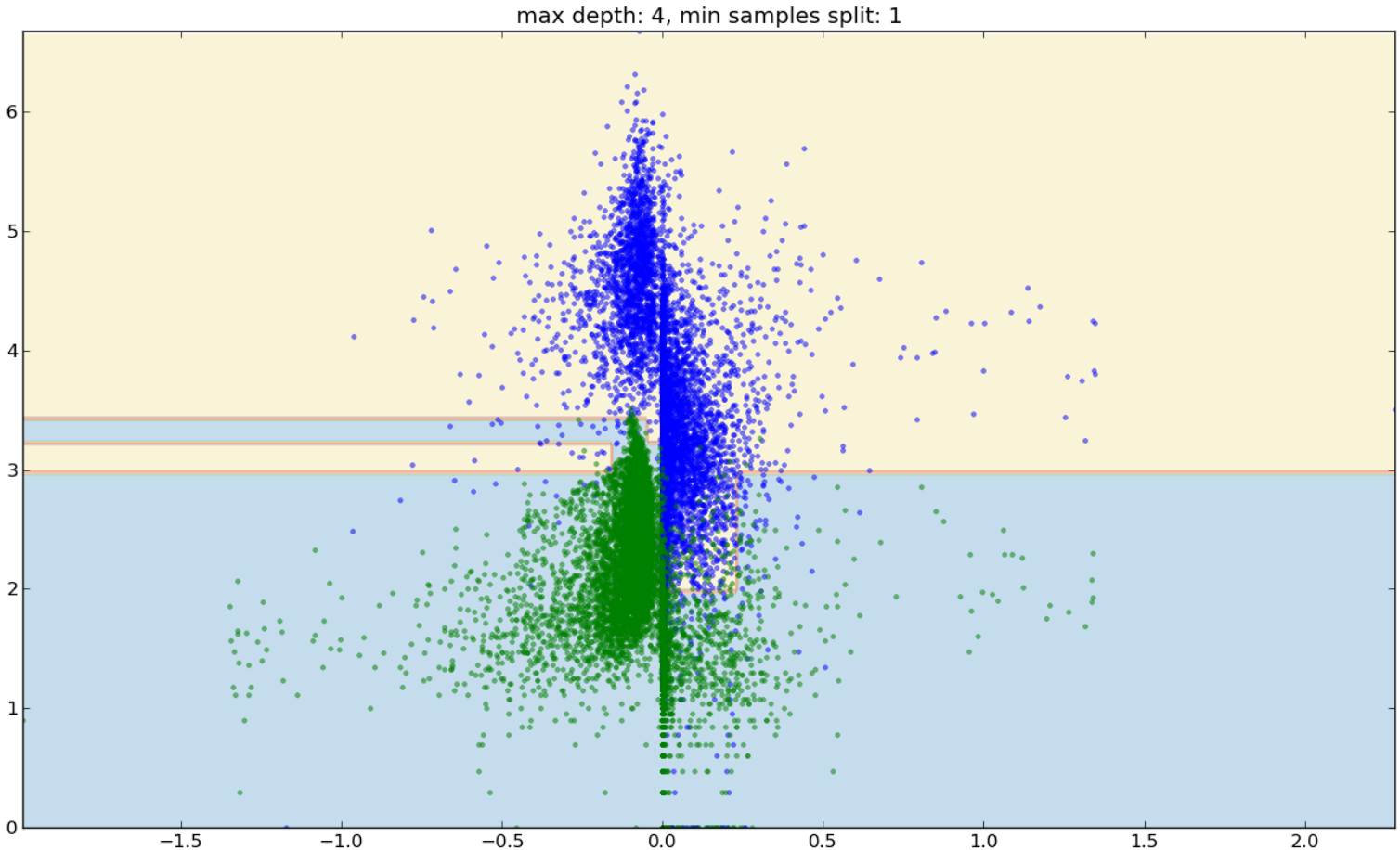
Example



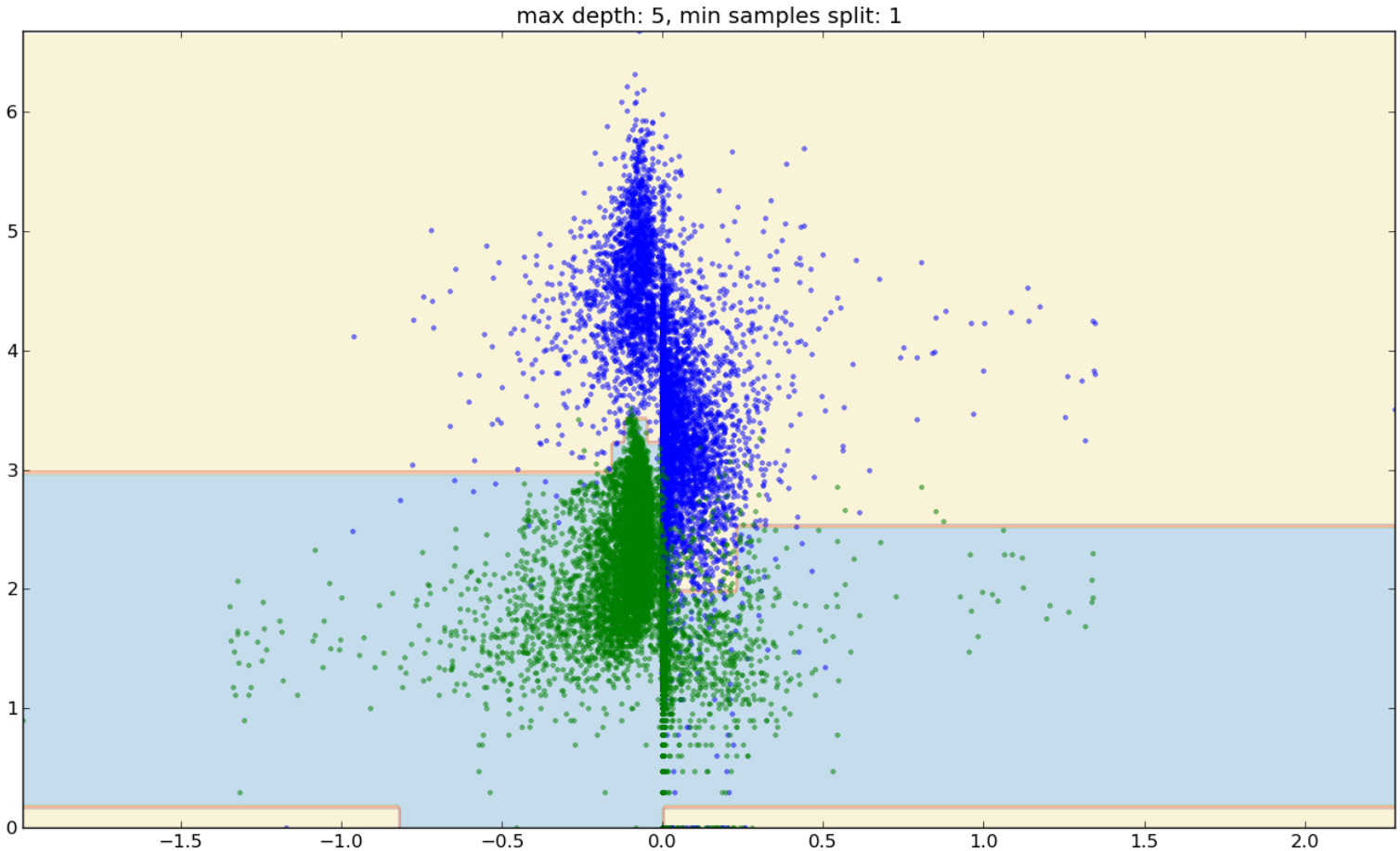
Example



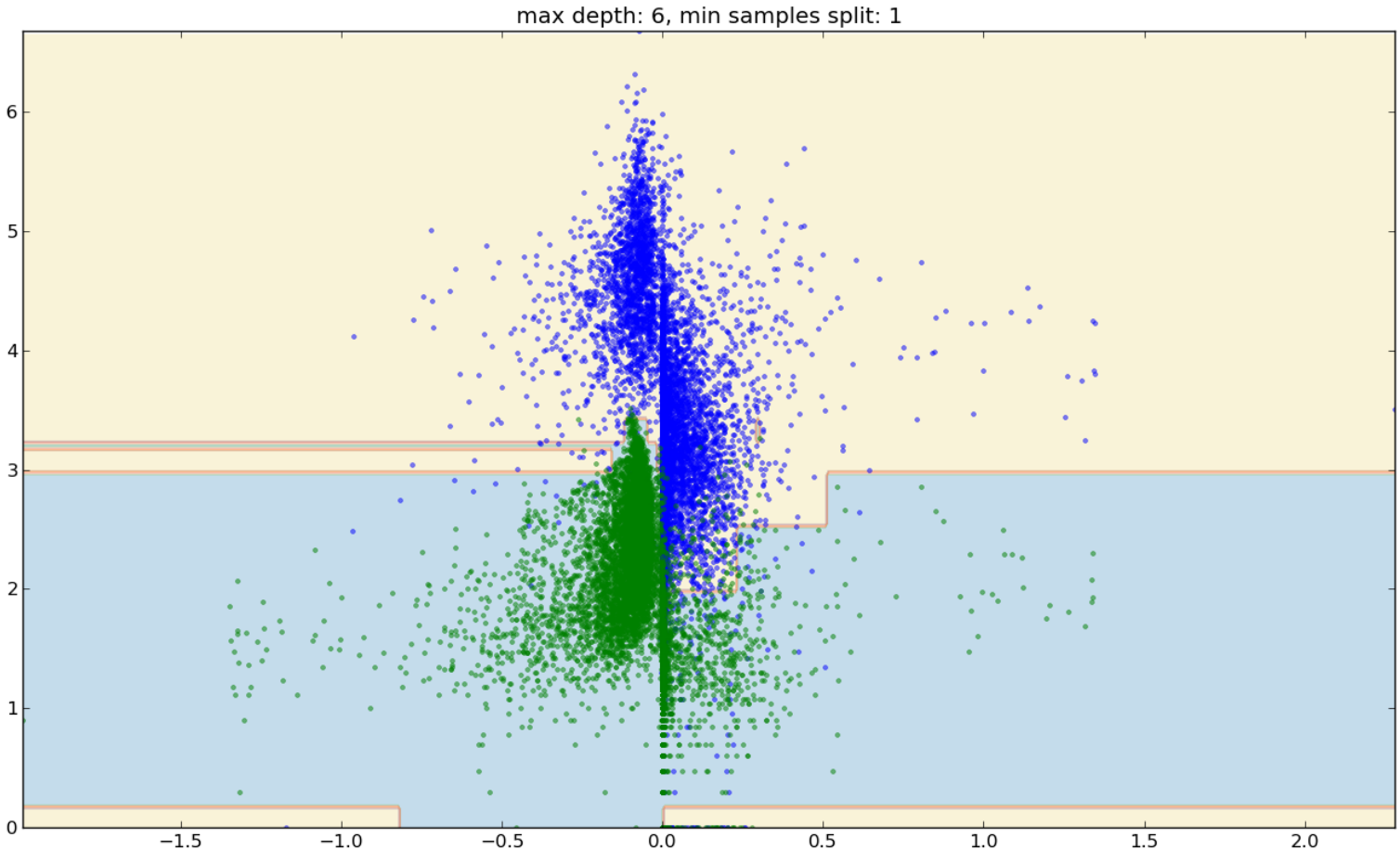
Example



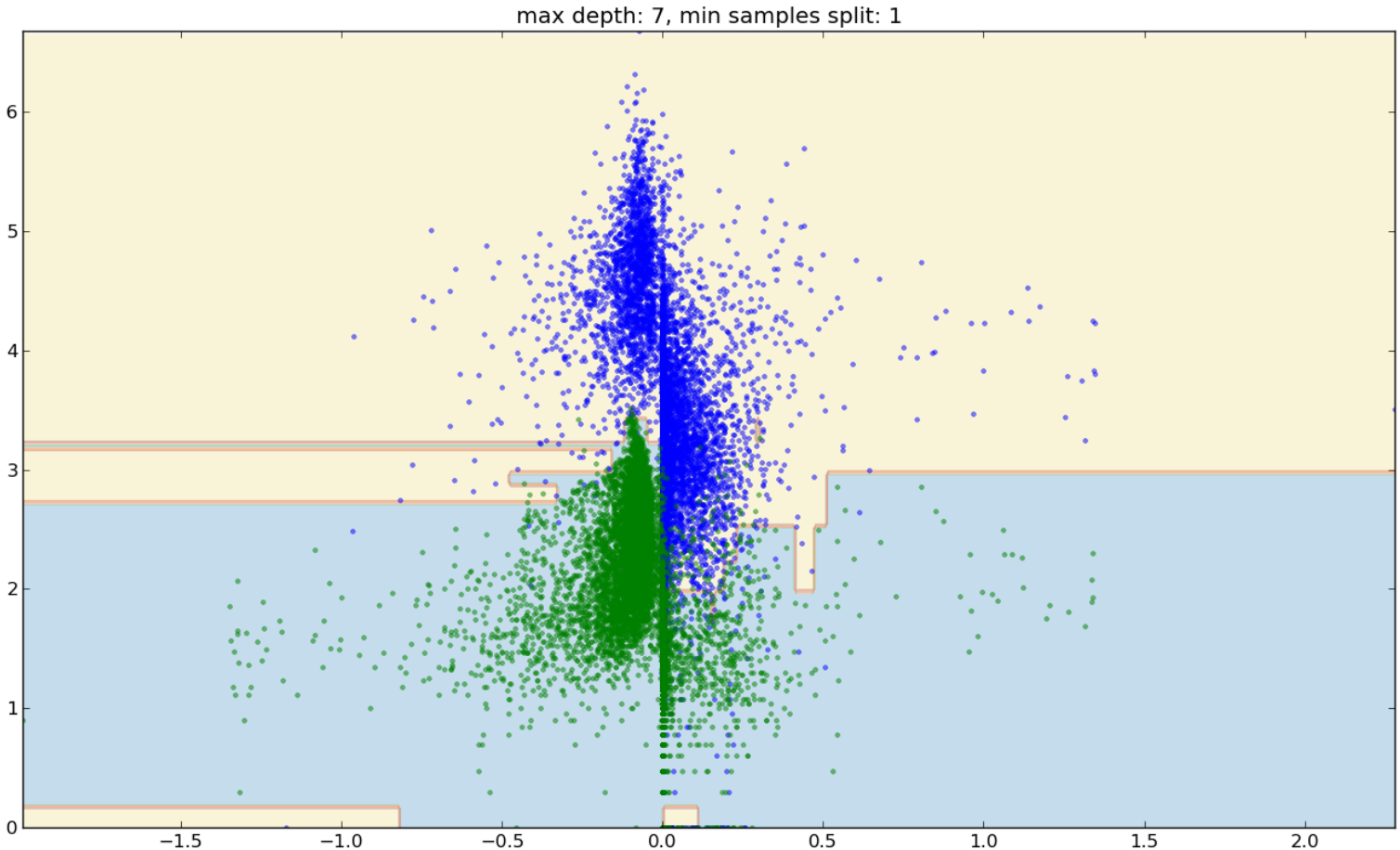
Example



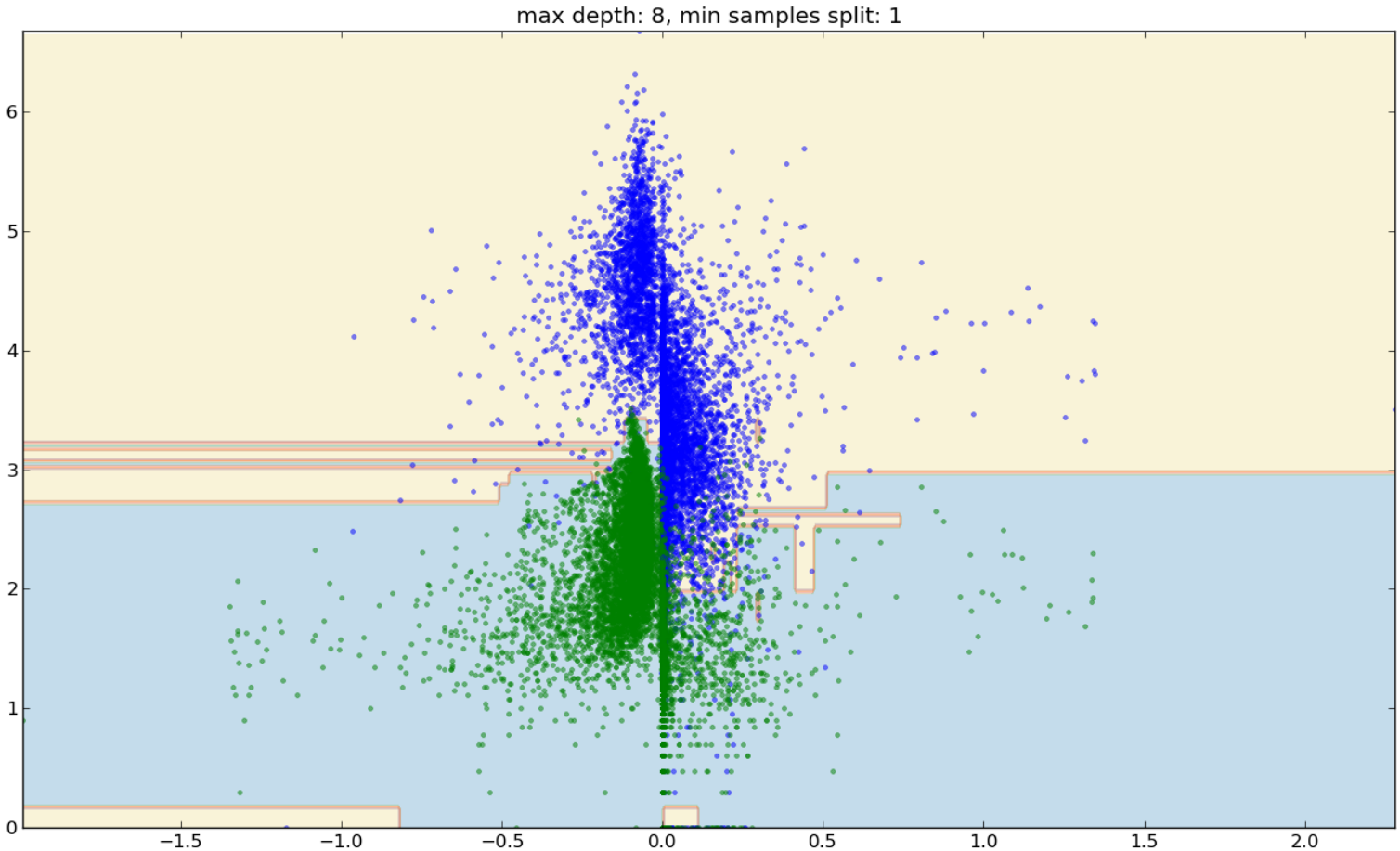
Example



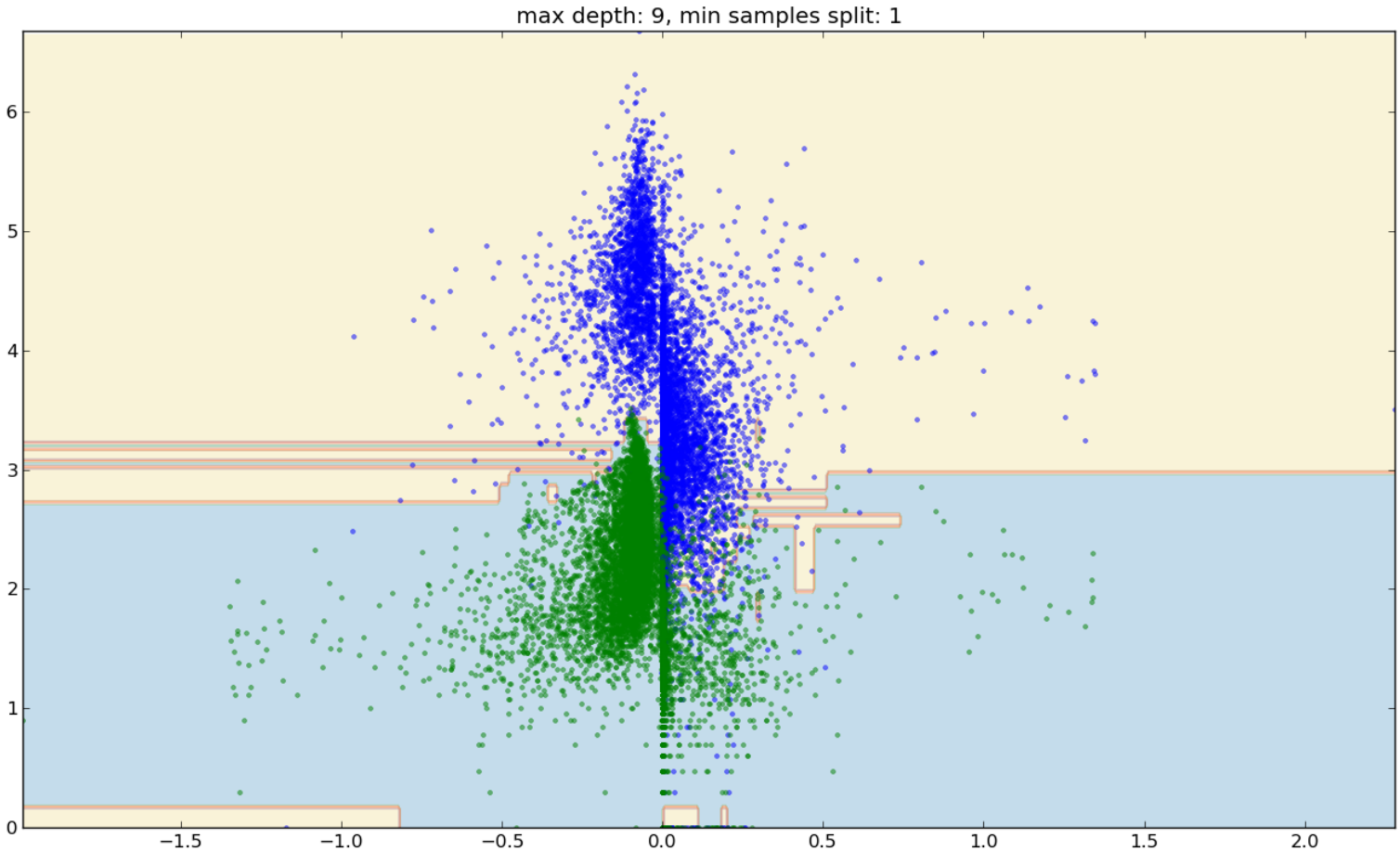
Example



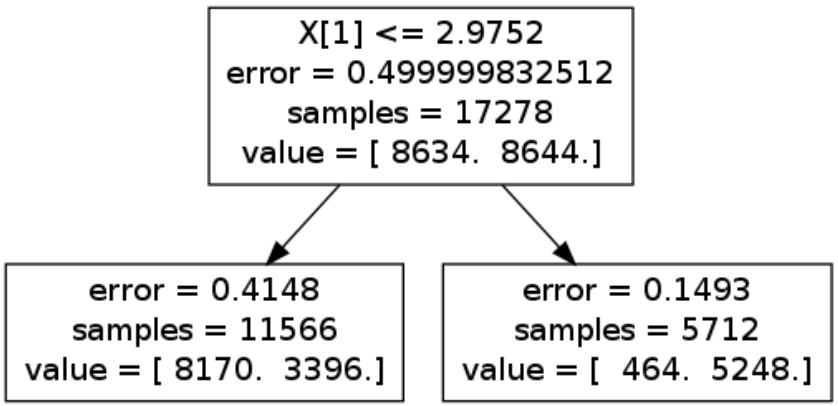
Example



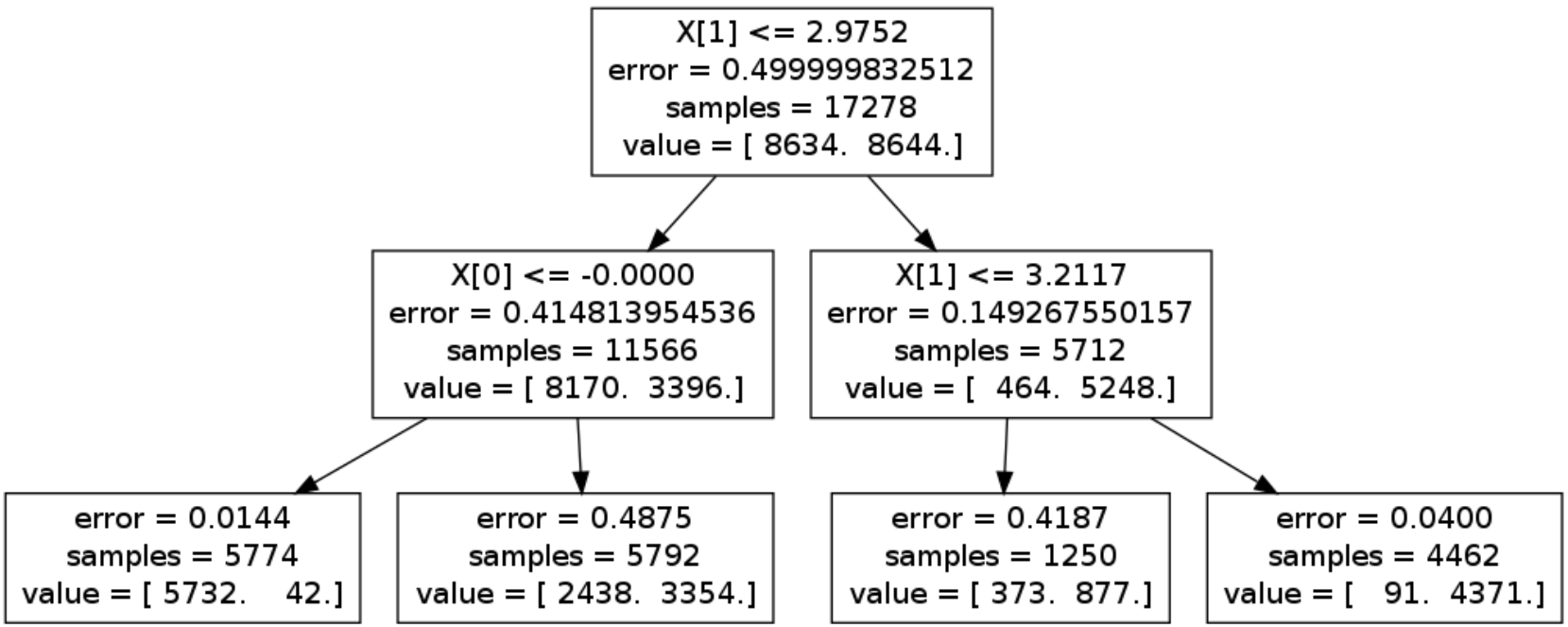
Example



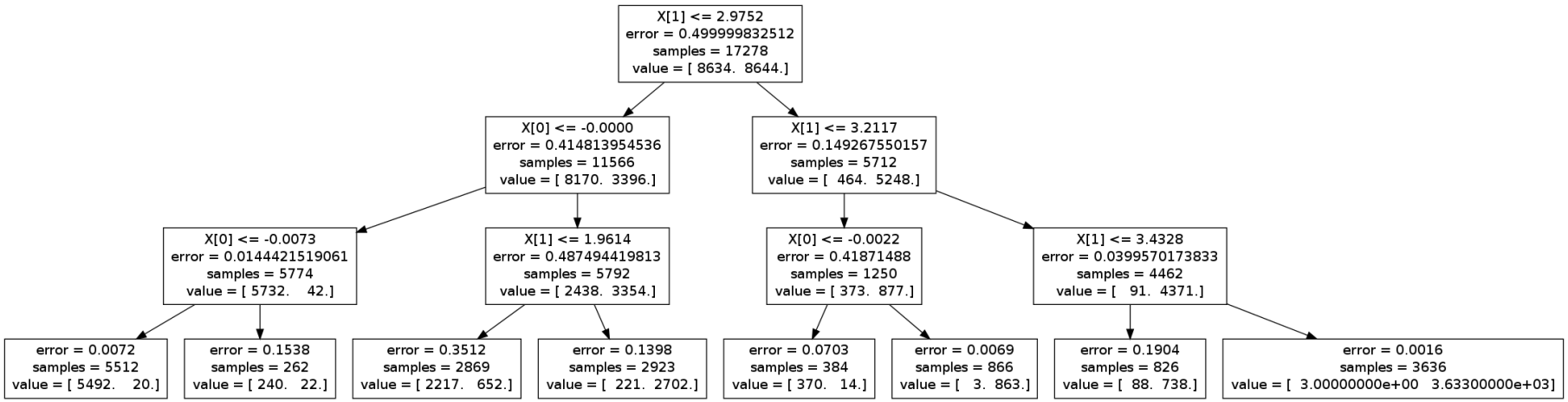
Example


X[1] <= 2.9752
error = 0.499999832512
samples = 17278
value = [8634. 8644.]
error = 0.4148
samples = 11566
value = [8170. 3396.]
error = 0.1493
samples = 5712
value = [464. 5248.]

Example


X[1] <= 2.9752
error = 0.499999832512
samples = 17278
value = [8634. 8644.]
X[0] <= -0.0000
error = 0.414813954536
samples = 11566
value = [8170. 3396.]
X[1] <= 3.2117
error = 0.149267550157
samples = 5712
value = [464. 5248.]
error = 0.0144
samples = 5774
value = [5732. 42.]
error = 0.4875
samples = 5792
value = [2438. 3354.]
error = 0.4187
samples = 1250
value = [373. 877.]
error = 0.0400
samples = 4462
value = [91. 4371.]

Example


X[1] <= 2.9752
error = 0.499999832512
samples = 17278
value = [8634. 8644.]
X[0] <= -0.0000
error = 0.414813954536
samples = 11566
value = [8170. 3396.]
X[1] <= 3.2117
error = 0.149267550157
samples = 5712
value = [464. 5248.]
X[0] <= -0.0073
error = 0.0144421519061
samples = 5774
value = [5732. 42.]
X[1] <= 1.9614
error = 0.487494419813
samples = 5792
value = [2438. 3354.]
X[0] <= -0.0022
error = 0.41871488
samples = 1250
value = [373. 877.]
X[1] <= 3.4328
error = 0.0399570173833
samples = 4462
value = [91. 4371.]
error = 0.0072
samples = 5512
value = [5492. 20.]
error = 0.1538
samples = 262
value = [240. 22.]
error = 0.3512
samples = 2869
value = [2217. 652.]
error = 0.1398
samples = 2923
value = [221. 2702.]
error = 0.0703
samples = 384
value = [370. 14.]
error = 0.0069
samples = 866
value = [3. 863.]
error = 0.1904
samples = 826
value = [88. 738.]
error = 0.0016
samples = 3636
value = [3.00000000e+00 3.63300000e+03]

Pros and Cons of CART

- Pros:
  1. easier to interpret (easy to convert to logic rules)
  2. automatic variable selection
  3. scale well
  4. insensitive to monotone transformations of the inputs (based on ranking the data points), no need to scale the data
- Cons:
  1. performance is not good (greedy)
  2. unstable

You can not expect too much on a brick ...

Qustion: what makes a good brick?

Question: Why *Binary Split*

Question: Why *Binary Split*

No concrete answer

1. Similar to logic
2. Stable (otherwise, more possibilities, errors on the top will affect the splitting below)

Let's begin to build the house!

Bootstrap Aggregation (Bagging): Just average it!

Bagging:

1. Build M decision trees on bootstrap data (randomly resampled with replacement)
2. Average it:

$$f(x) = \frac{1}{M} \sum_{i=1}^{M} f_i(x)$$

Note that, CARTs here do not need pruning (why?)

Two things

1. How many trees to use (M) ? More the better, increase until converge (CART is very fast, can be trained in parallel)
2. What is the size of bootstrap learning set? Breiman chose same size as original data, then every bootstrap set leaves out

$$(1 - \frac{1}{n})^n$$

percentage of data (about 37% when n is large)

Question 1: Why Bagging Works?

Question 1: Why Bagging Works?

1. Bias-Variance Trade off:

$$E[(F(x) - f_i(x))^2] = (F(x) - E[f_i(x)])^2 + E[f_i(x) - Ef_i(x)]^2$$
$$= (Bias[f_i(x)])^2 + Var[f_i(x)]$$

2. $|Bias(f_i(x))|$ is small but $Var(f_i(x))$ is big
3. Then for $f(x)$, if $\{f_i(x)\}$ are mutually independent

$$Bias(f(x)) = Bias(f_i(x))$$
$$Var(f(x)) = \frac{1}{n} Var(f_i(x))$$

Variance is reduced by averaging

Question 2: Why Bagging works NOT quite well?

Question 2: Why Bagging works NOT quite well?

Trees are strongly Correlated (repetition, bootstrap)
To improve : random forest (add more randomness!)

Random Forest

> Random forests are a combination of tree predictors such that each tree depends on the values of a **random** vector sampled independently and with the same distribution for all trees in the forest
>
> — LEO BREIMAN

Random Forest

> Random forests are a combination of tree predictors such that each tree depends on the values of a **random** vector sampled independently and with the same distribution for all trees in the forest
>
> — LEO BREIMAN

Where randomness helps!

Random Forest: Learning

- Where is the "extra" randomness:

> As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a **random sample** of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use **only** one of those m predictors.
>
> — ISLR book

- Give more chance on the "single weak" predictor (features)
- *Decorrelate* the trees $\rightarrow$ reduce the variance of the final hyperthesis

Random Forest

- How many sub-features to choose (not very sensitive, try it (1, half, $int(\log_2 p + 1)$))
- Internal test of Random Forest (OOB error)
- Interpretation

Out of Bag Error (OOB error)

- "By product of bootstraping": Out-of-bag data
- For every training data $(\mathbf{x_i}, y_i)$, there are about 37% of trees that does not train on it.
- In testing, use the 37% trees to predict the $(\mathbf{x_i}, y_i)$. Then we can get a RSS or classification error rate from it.
- "proven to be unbiased in many tests."
- Now need to do Cross-validation.

Random Forest: Interpretation (Variable Importance)

By averaging, "Random Forest" lost the ability of "easy interpretation" of CART. But it is still "interpretable"

Permutation Test (1)

Test correlation by permutation

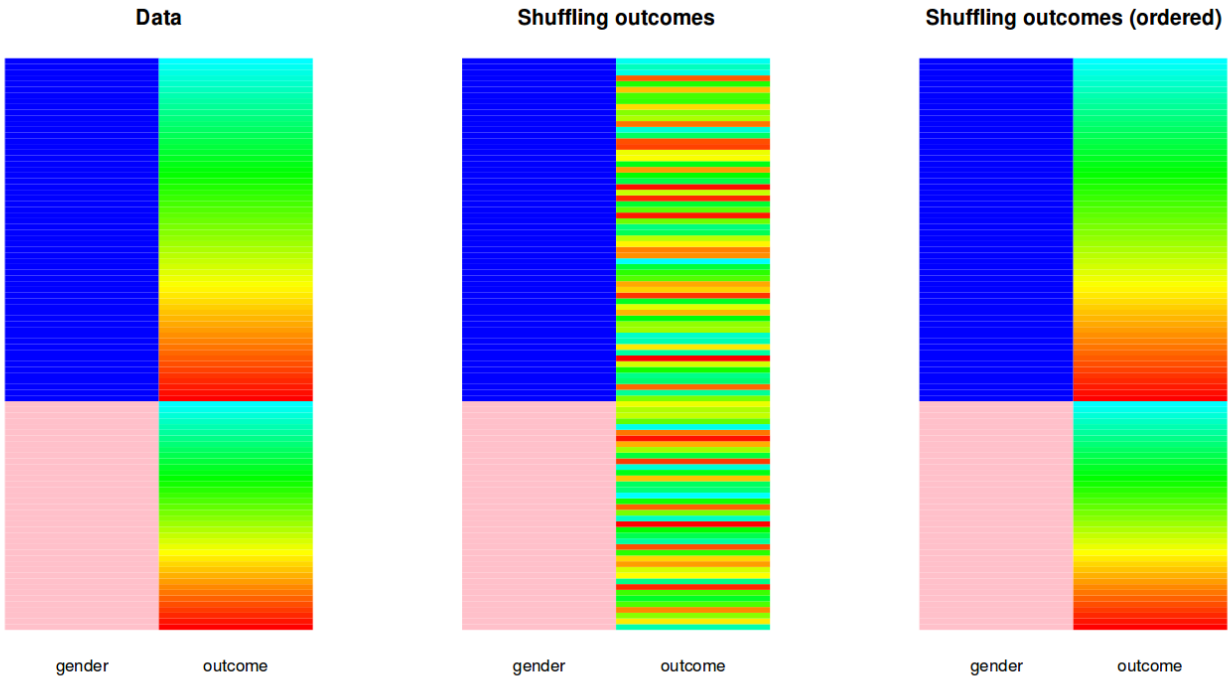


Figure : Does not correlate

— Ken Rice, Thomas Lumley, 2008

Permutation Test (2)

Test correlation by permutation

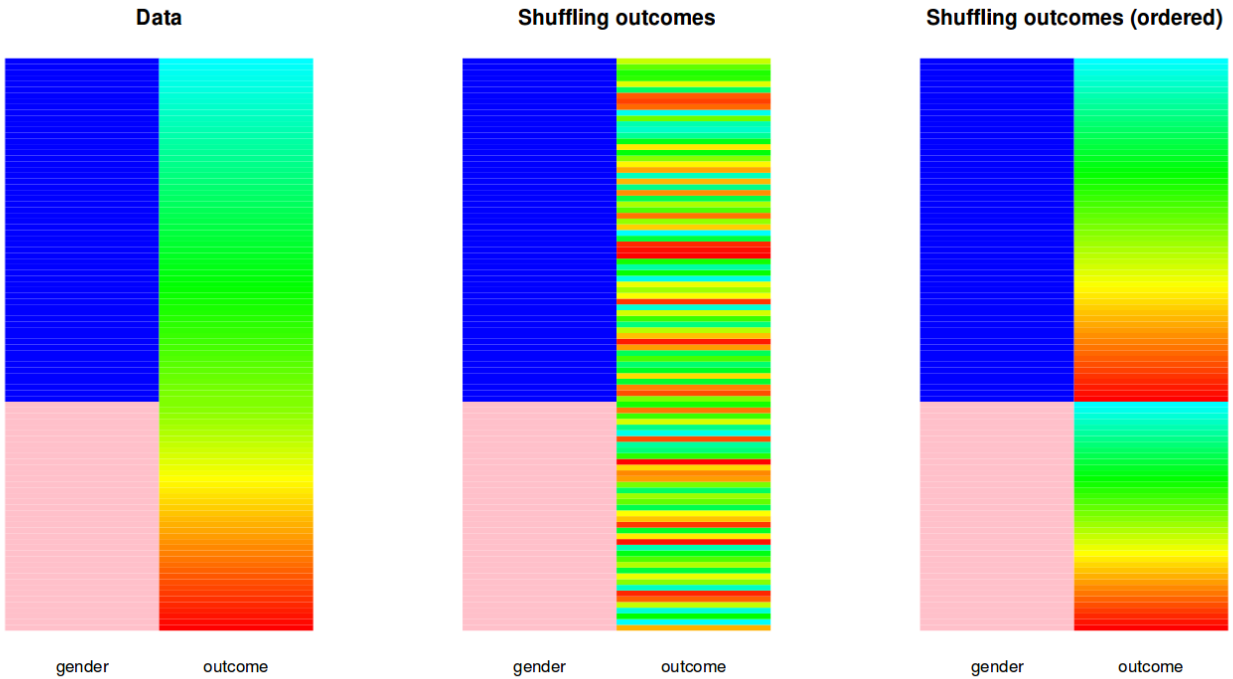
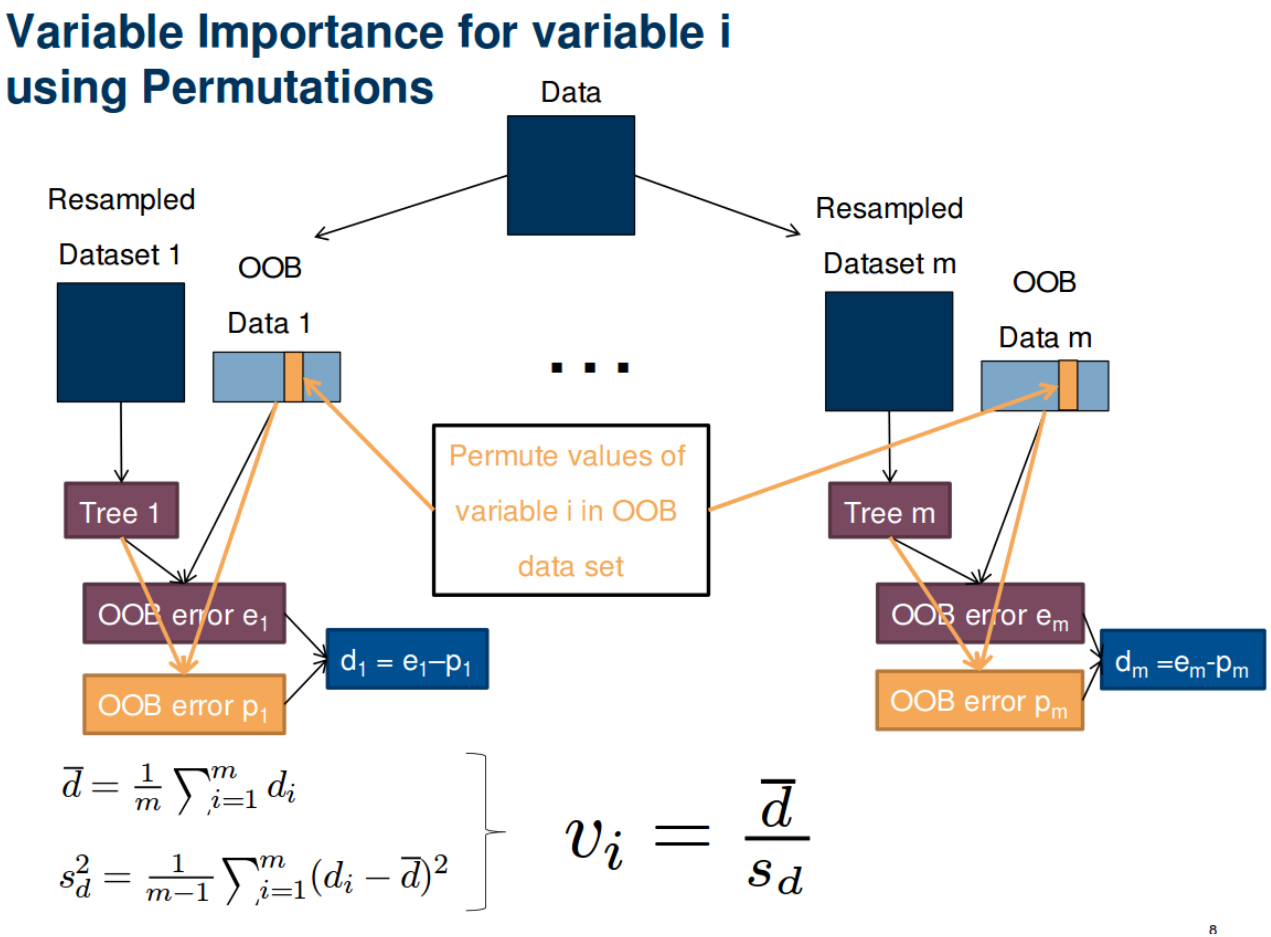


Figure : Correlate

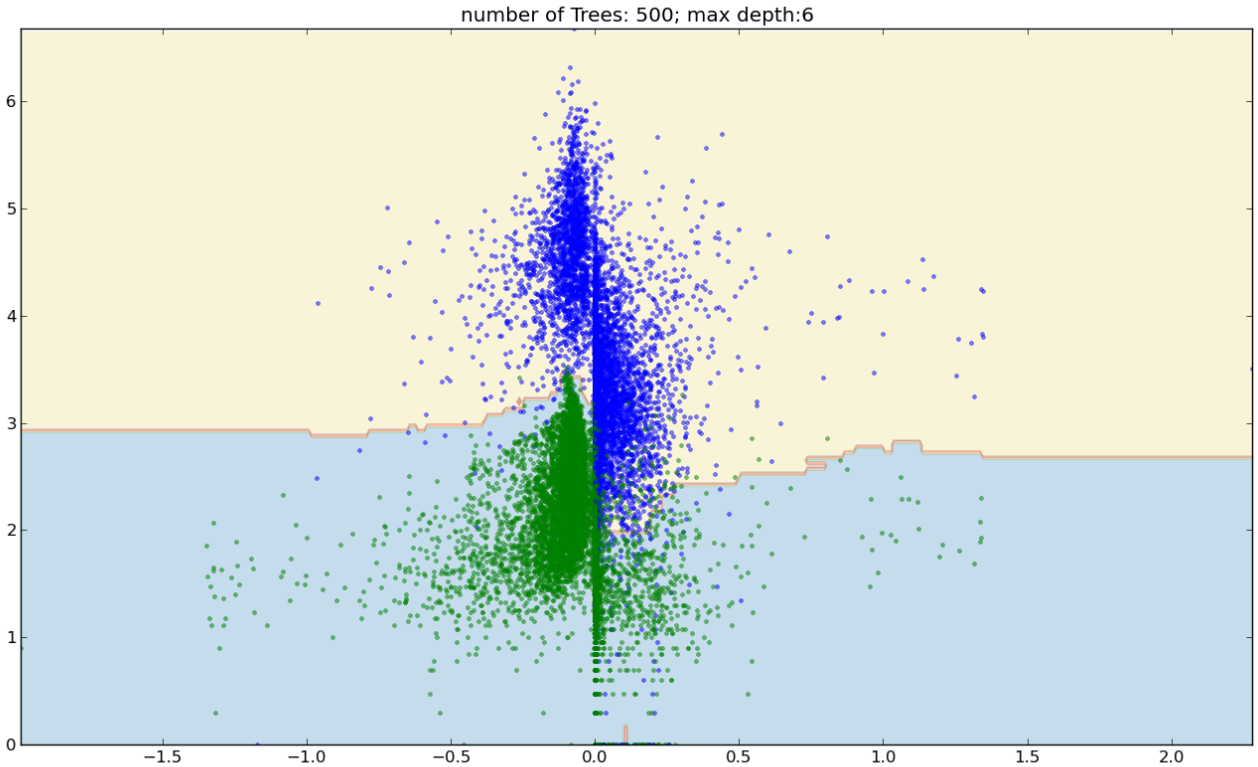
— Ken Rice, Thomas Lumley, 2008

Interpret Random Forest



https://stat.ethz.ch/education/semesters/ss2012/ams/slides/v10.2.pdf

Example



- Feature importance: $x : y = 0.34038216 : 0.65961784$
- OOB error: 0.947563375391

Pros and Cons

- Pros:
  1. Performance is good, a lot of applications
  2. not hard to interpret
  3. Easy to train in parallel
- Cons:
  1. Easy to overfit
  2. (?) worse than SVM in high dimension learning

Boosting

Main difference between Boosting and Bagging?

1. "Bricks" (CART here) are trained sequentially.
2. Every "brick" is trained on the whole data (with modification based on training former "bricks")

Note that, actually, "Boosting" originates before the invention of "Random Forest". Though "Boosting" performs better than "Random forest" and is also more complex than it.

Boosting a Regression Tree

Older trees can be seen as (except shrinkage) being built on the "residuals" of former trees.

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \quad (8.10)$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \quad (8.11)$$
3. Output the boosted model,
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \quad (8.12)$$

1. The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
2. The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
3. The number d of splits in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a *stump*, consisting of a single split. In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable. More generally d is the *interaction depth*, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

— ISLR book

Boosting a Classification Tree

- Train trees sequentially: For new trees, "hard" data from old trees are given more weights.
- Reduce *bias* (early stage) and the *variance* (late stage)

Compared with Random Forest

1. Performance is often better
2. Slow to train and test
3. Easier to overfit

Questions

Is Neural Network also Ensemble Learning?

Questions

What is the relationship between Linear Regression and Random Forest etc.?

Big Picture again

Adaptive basis function models