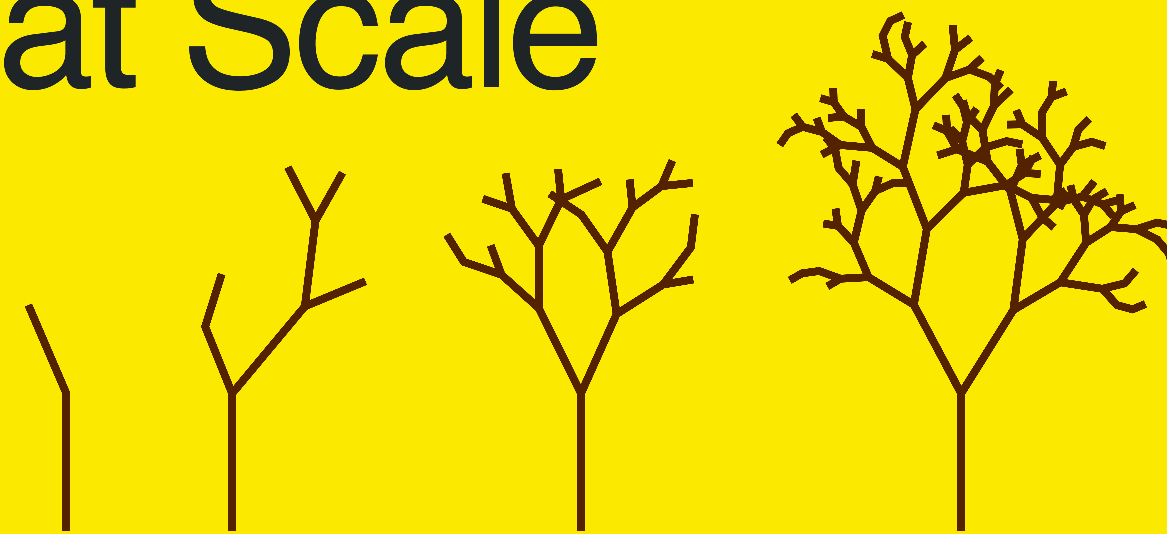
0xdata

Building Random Forest at Scale



Michal Malohlava

@mmalohlava

@hexadata

BrightTalk
5/20/2014

Who am I?

Background

- PhD in CS from Charles University in Prague, 2012
- 1 year PostDoc at Purdue University experimenting with algos for large computation
- 1 year at 0xdata helping to develop H_2O engine for big data computation

Experience with domain-specific languages, distributed system, software engineering, and big data.

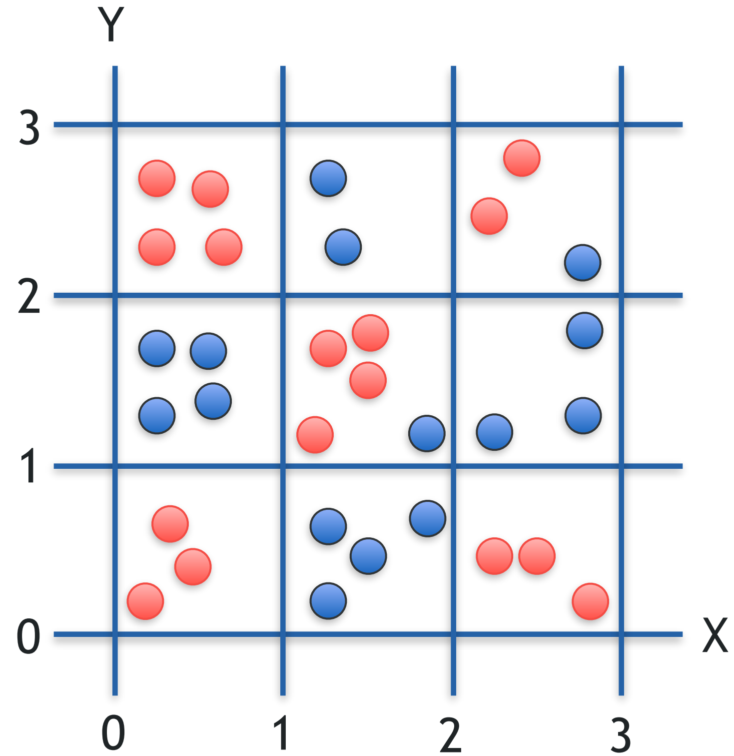
Overview

1. A little bit of theory
2. Random Forest observations
3. Scaling & distribution of Random Forest
4. Q&A

Tree Planting

0xdata

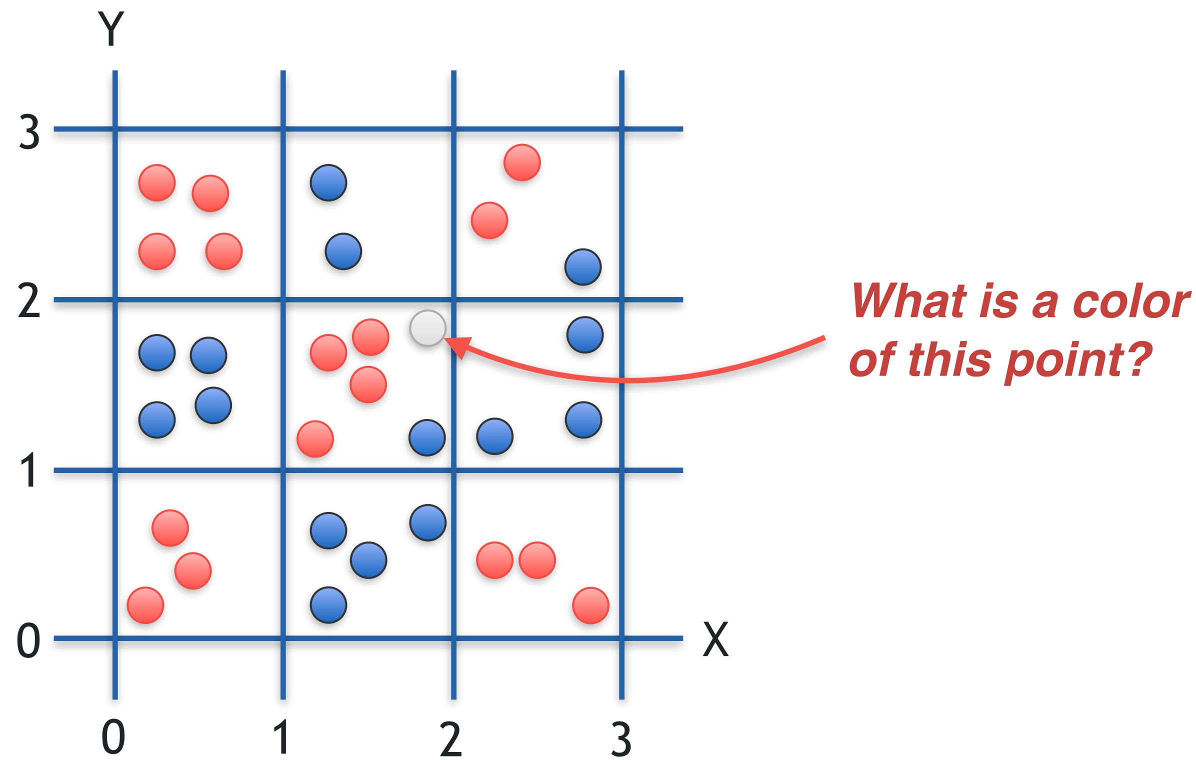
What is a model for this data?



- Training sample of points covering area [0,3] x [0,3]
- Two possible colors of points

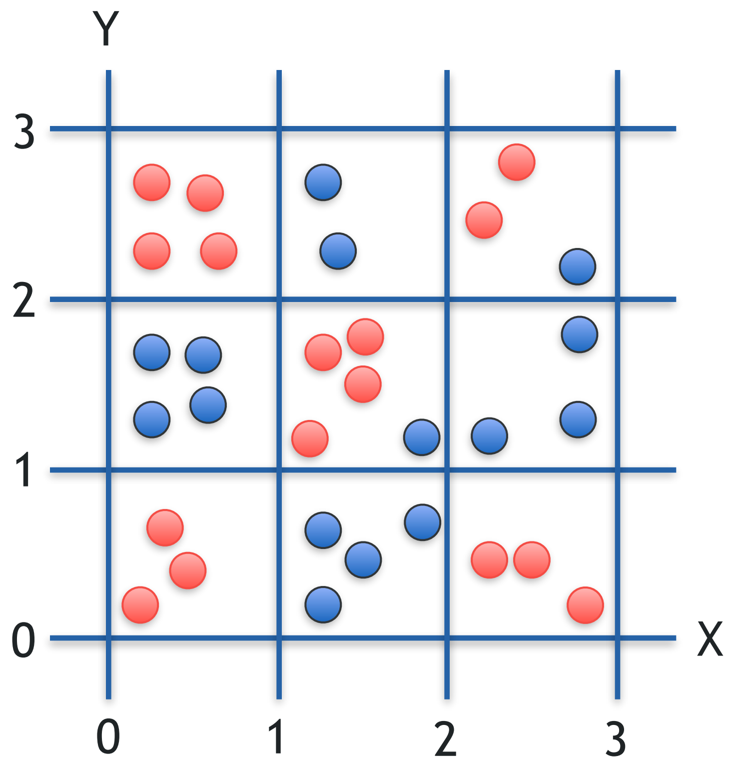
0xdata

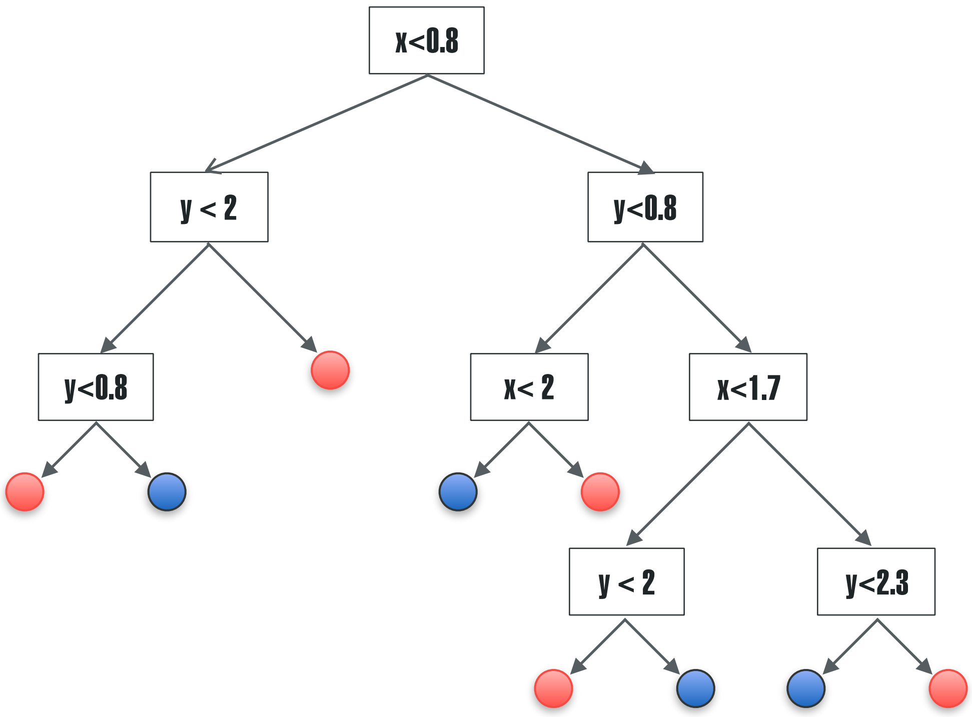
What is a model for this data?



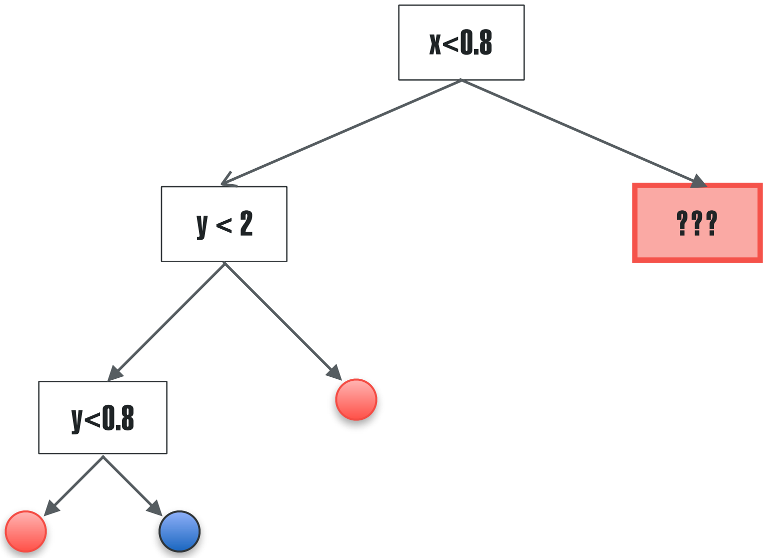
The model should be able to predict a color of a new point

Decision tree





How to grow a decision tree?

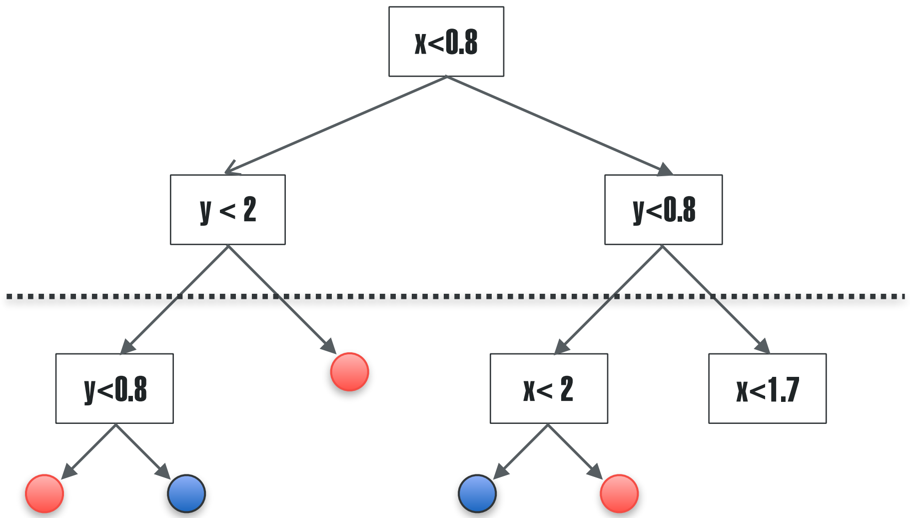


Split rows in a given node into two sets with respect to impurity measure

- The smaller, the more skewed is distribution
- Compare impurity of parent with impurity of children

A. Possible impurity measures
    - Gini, entropy, RSS
B. Respect type of feature - nominal, ordinal, continuous

When to stop growing the tree?



1. Build full tree

or

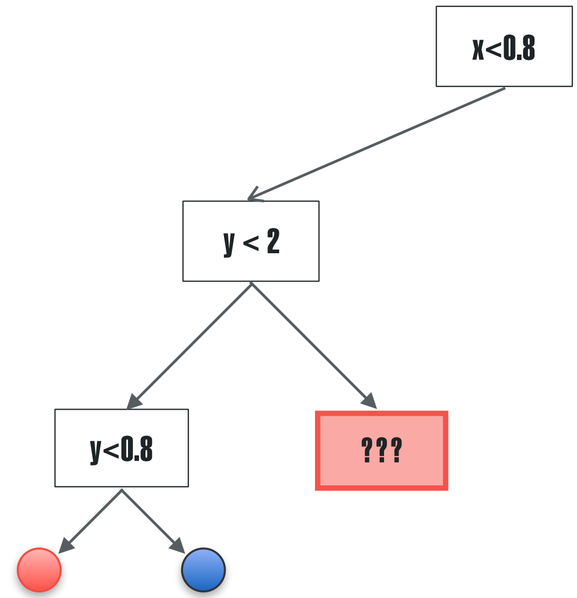
2. Apply stopping criterion - limit on:
    - Tree depth, or
    - Minimum number of points in a leaf

How to assign leaf value?

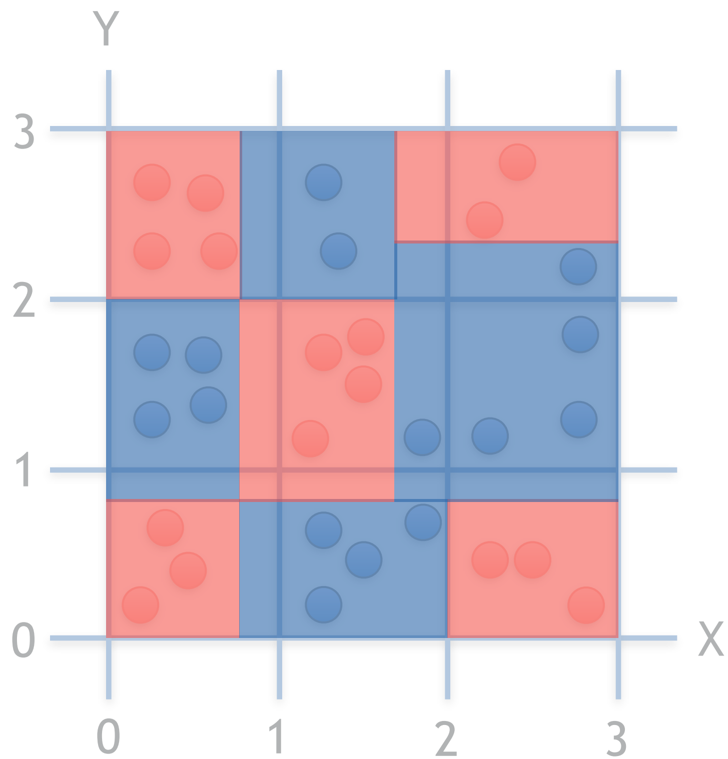
The leaf value is

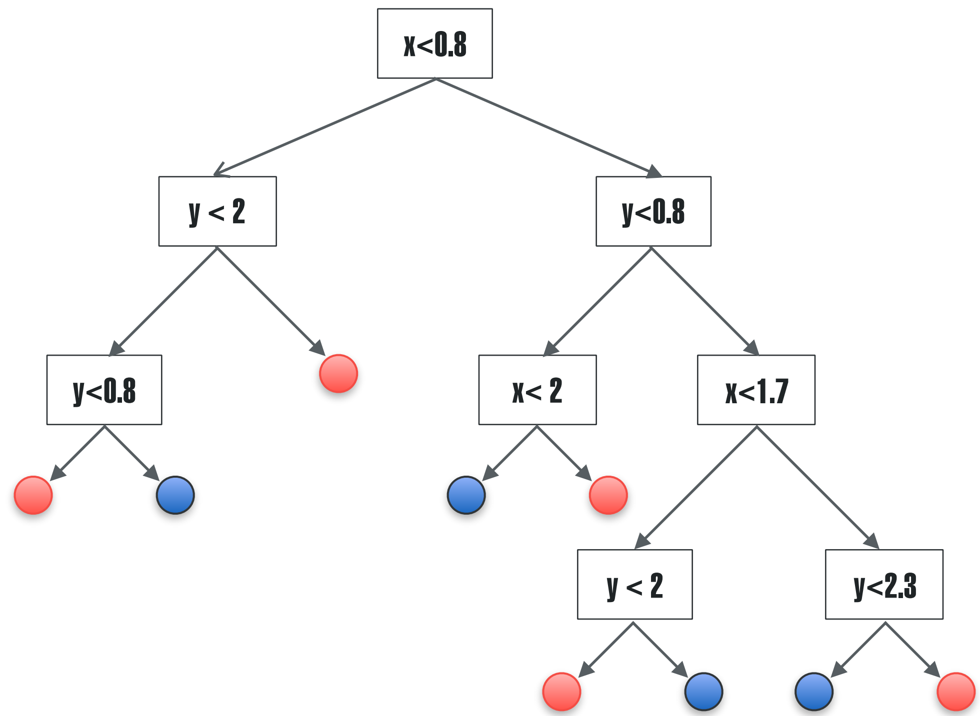
- If leaf contains only one point then its color represents leaf value

- Else majority color is picked, or color distribution is stored



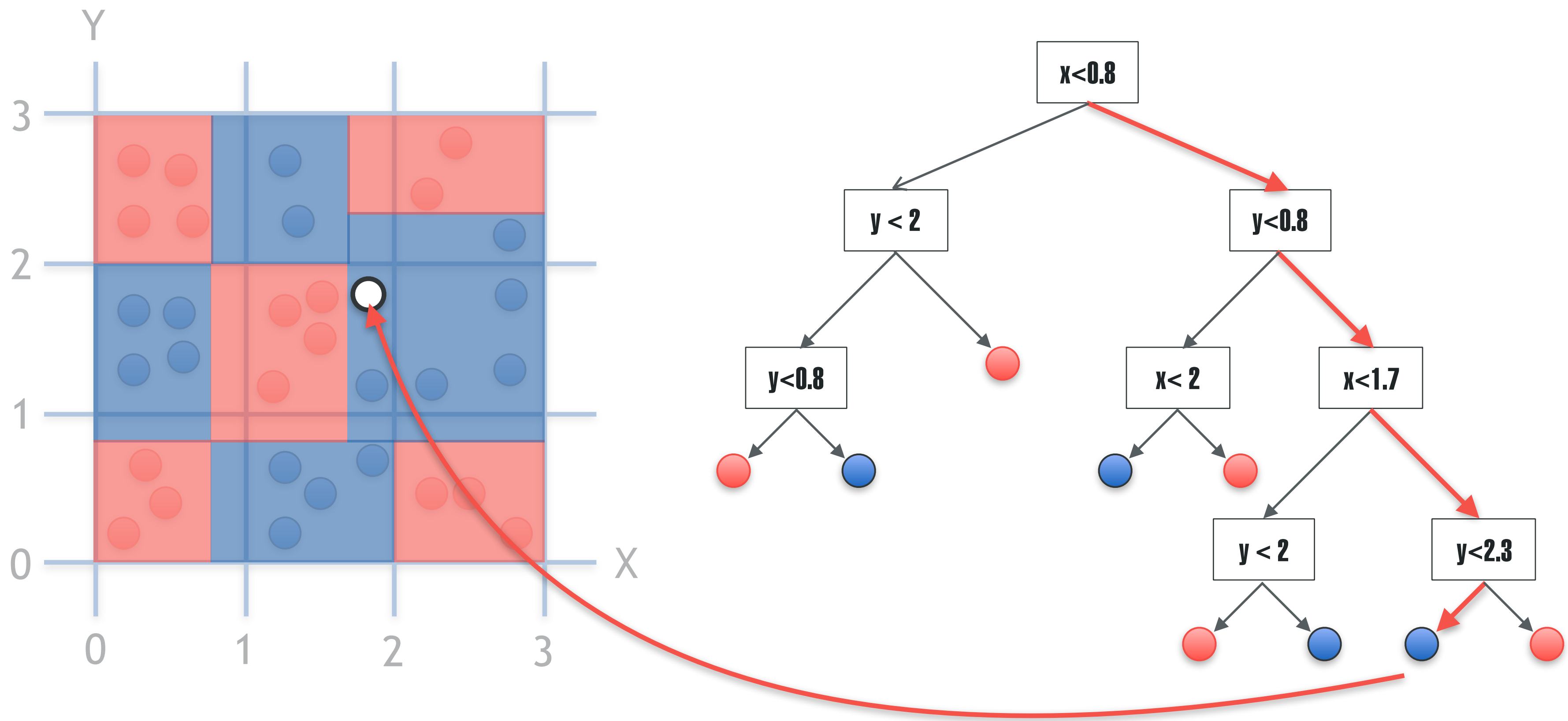
Decision tree





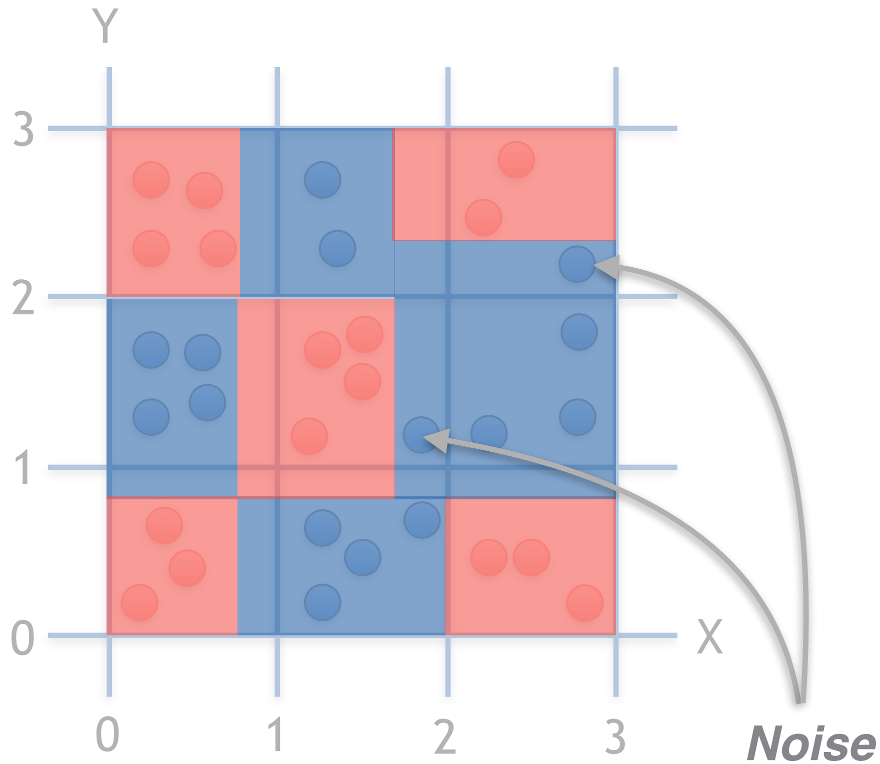
Tree covered whole area by rectangles predicting a point color

Decision tree scoring



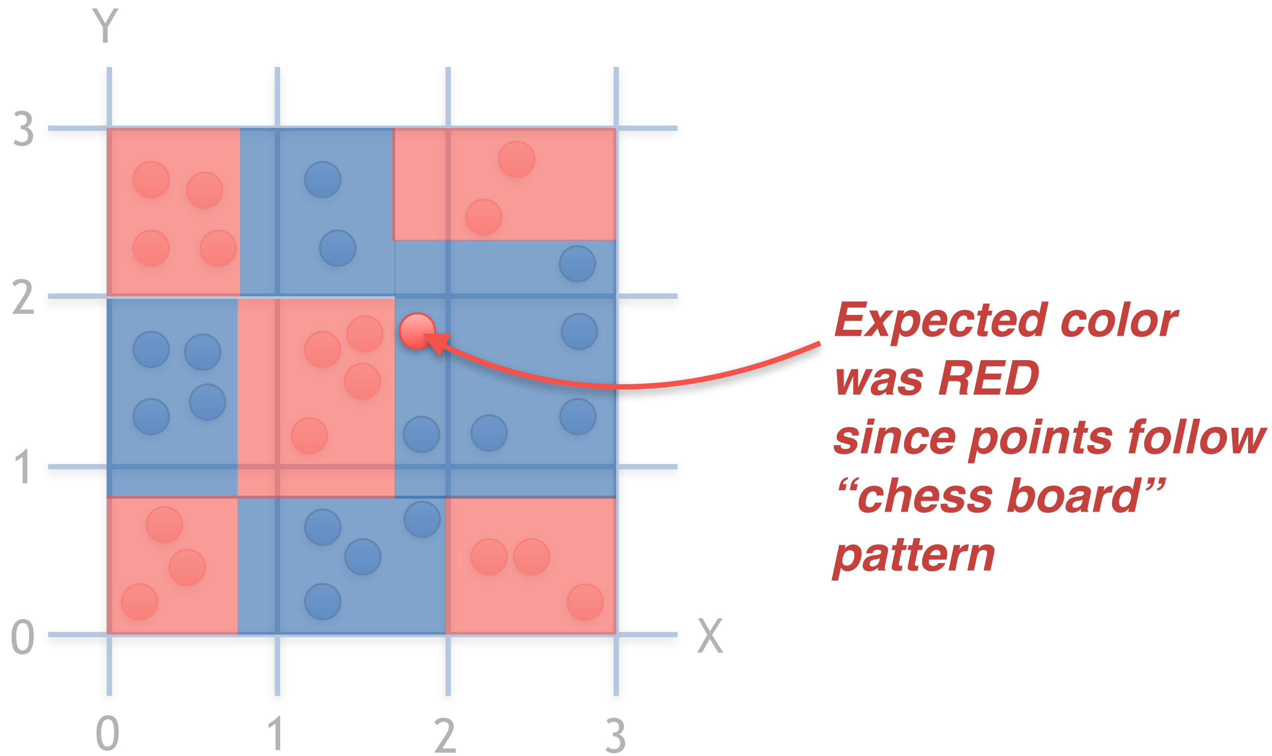
The model can predict a point color based on its coordinates.

Overfitting



Tree perfectly represents training data (0% training error), but also learned about noise!

Overfitting



And hence poorly predicts a new point!

Handle overfitting

"Model should have low training error but also generalization error!"

Pre-pruning via stopping criterion

Post-pruning: decreases complexity of model but helps with model generalization

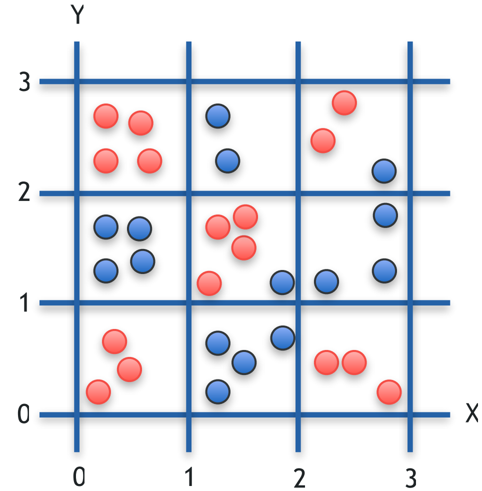
Random Forest idea

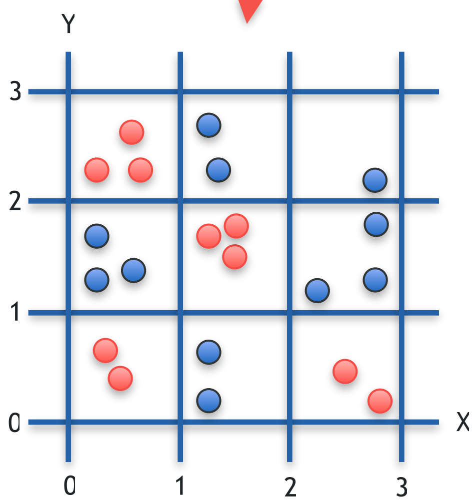
Randomize tree building and combine trees together

0xdata

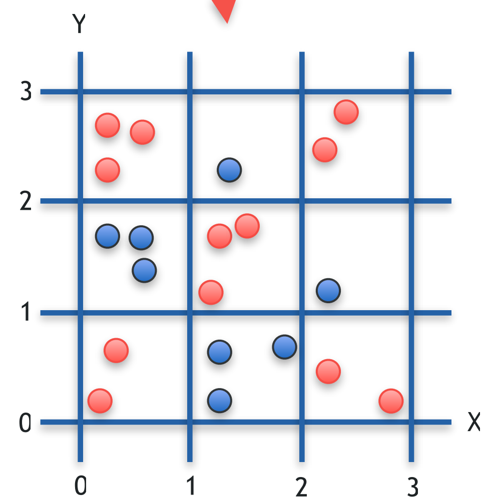
Breiman, L. (2001). *Random forests.* Machine Learning, 5–32.
http://link.springer.com/article/10.1023/A:1010933404324

Randomize #1
Bagging

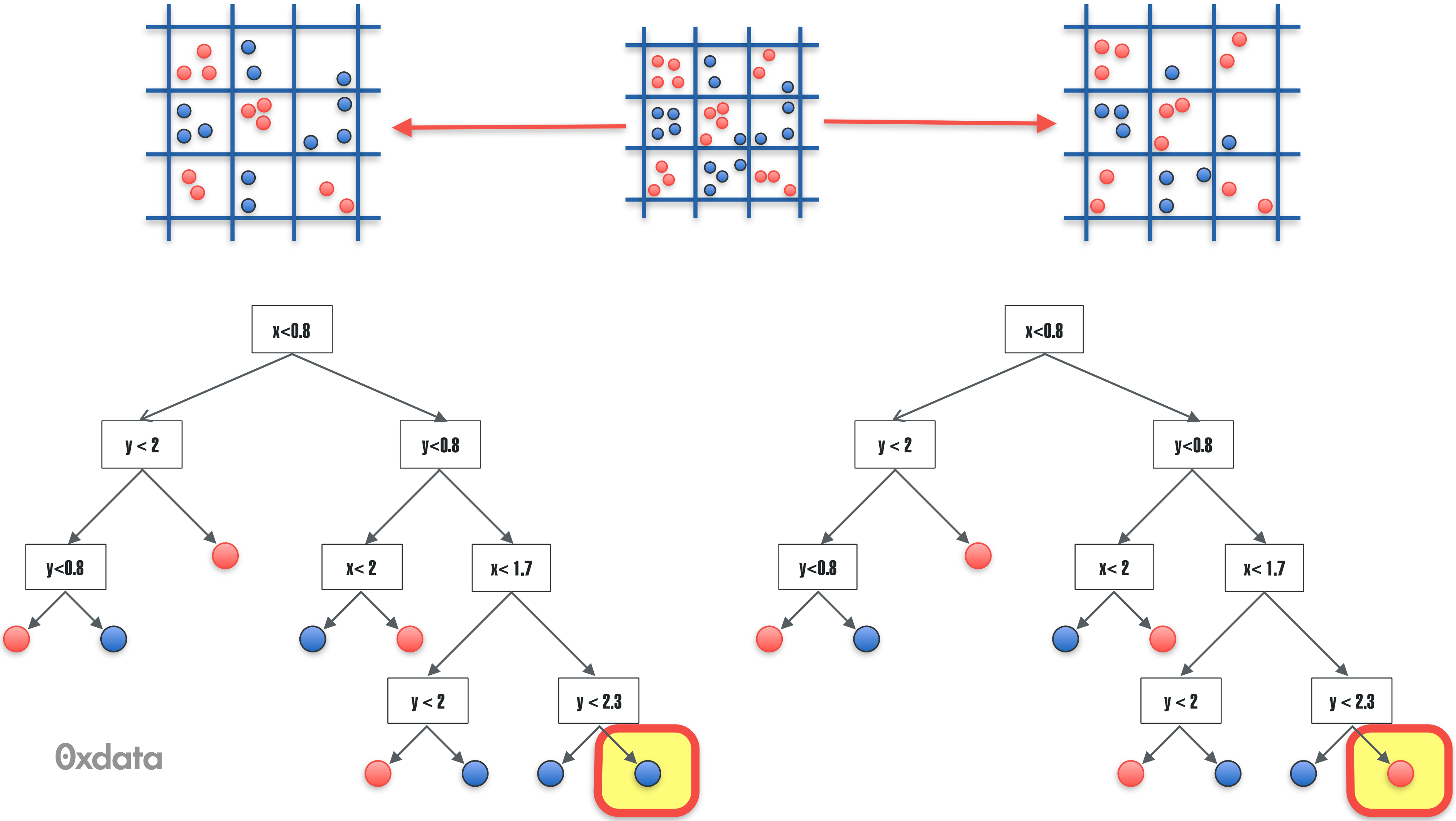




Prepare bootstrap sample for each tree by sampling with replacement



Randomize #1
Bagging



0xdata

Randomize #1
Bagging

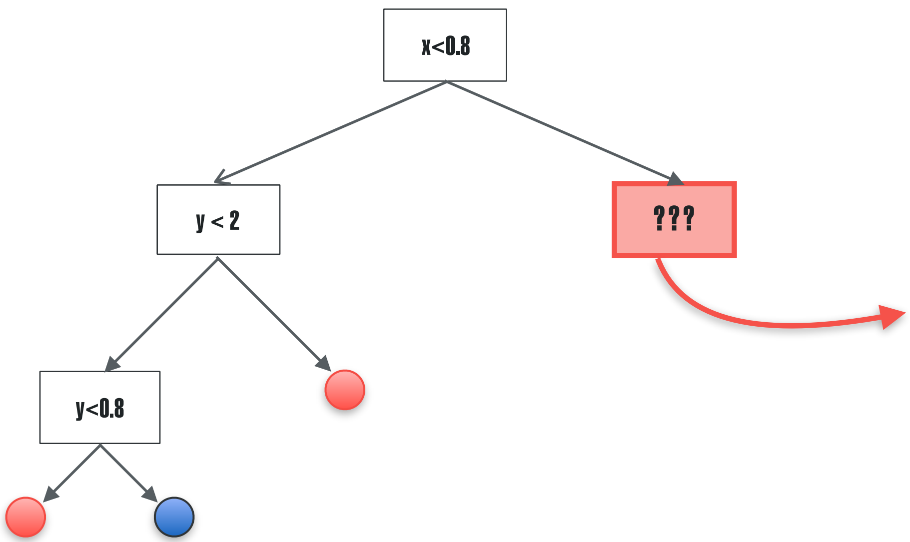
Each tree sees only sample of training data and captures only a part of the information.

Build multiple weak trees which vote together to give resulting prediction

- voting is based on majority vote, or weighted average

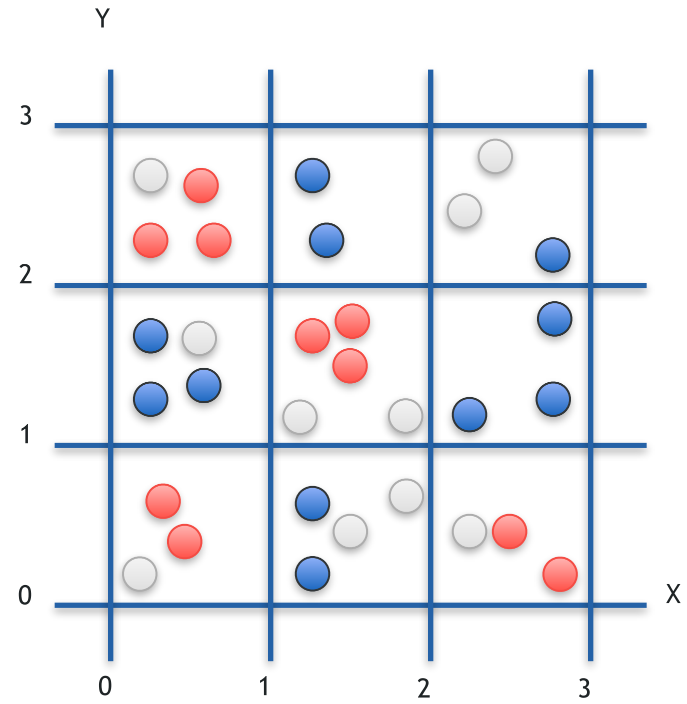
Randomize #2
Feature selection

Randomized split selection



- Select randomly subset of features of size sqrt(#features)
- Select the best split only using the subset

Out of bag points and validation



Each tree is built over a sample of training points.

Remaining points are called "out-of-bag" (OOB).

These points are used for validation as a good approximation for generalization error. Almost identical as N-fold cross validation.

Advantages of Random Forest

Independent trees which can be built in parallel

The model does not overfit easily

Produces reasonable accuracy

Brings more features to analyze data - variable importance, proximities, missing values imputation

0xdata

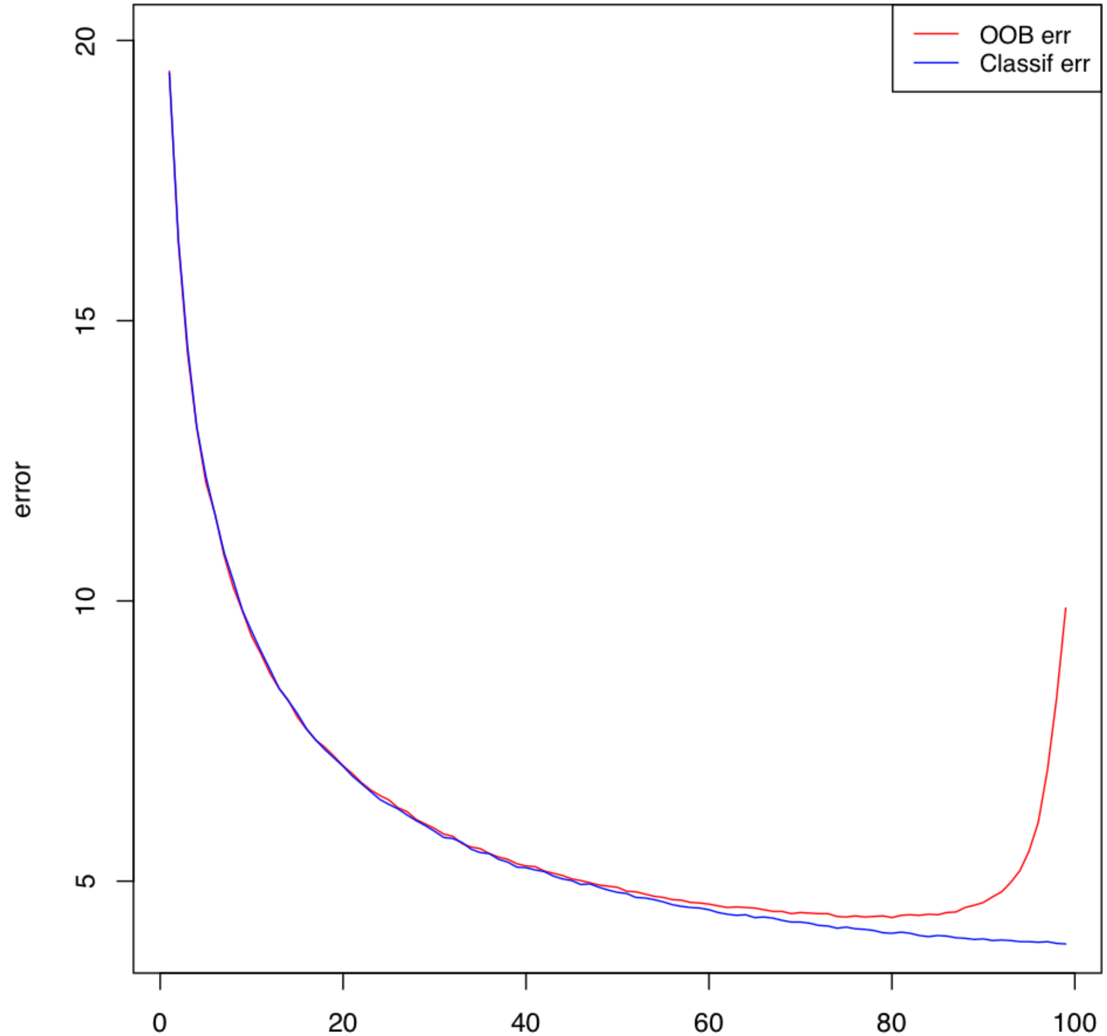
Breiman, L., Cutler, A. *Random Forest.* http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

A Few Observations

Covtype dataset

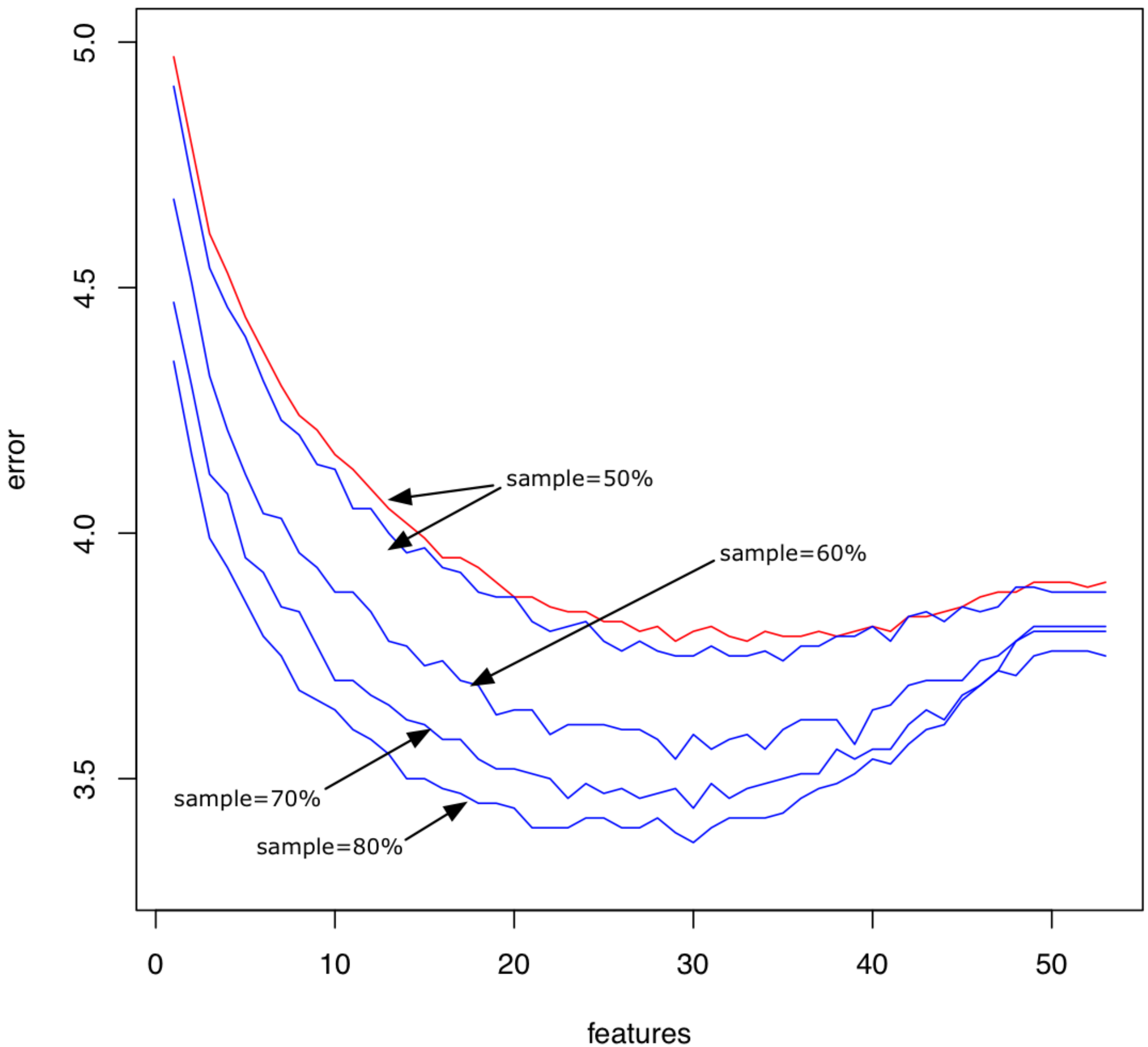
Dataset	Features	Predictor	Instances (train/test)	Imbalanced	Missing observations
Iris	4	3 classes	100/50	NO	0
Vehicle	18	4 classes	564/282	NO	0
Stego	163	3 classes	3,000/4,500	NO	0
Spam	57	2 classes	3,067/1,534	YES	0
Credit	10	2 classes	100,000/50,000	YES	29,731
Intrusion	41	2 classes	125,973/22,544	NO	0
Covtype	54	7 classes	387,342/193,672	YES	0

Sampling rate impact

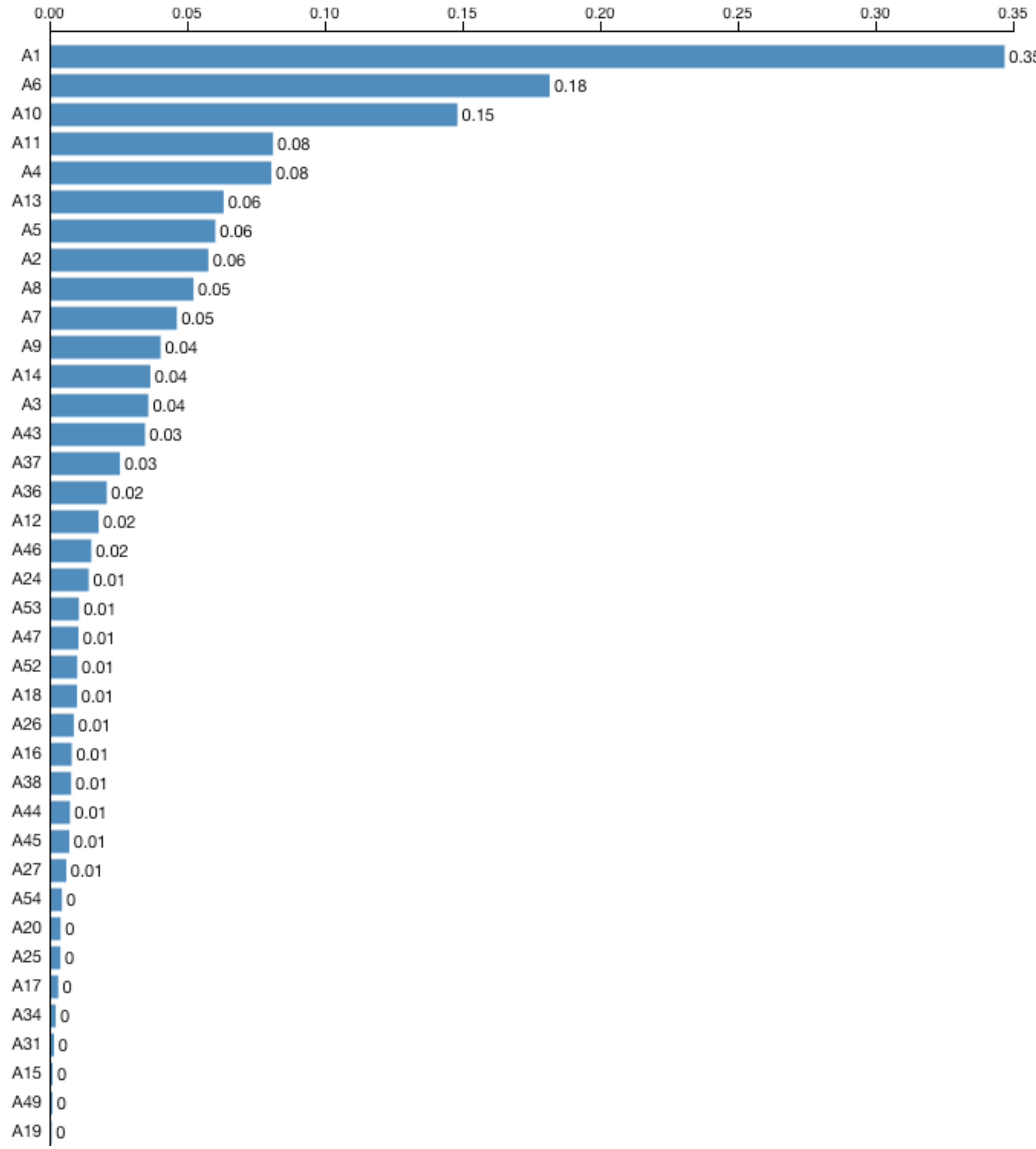


0xdata

Number of split features



Variable importance

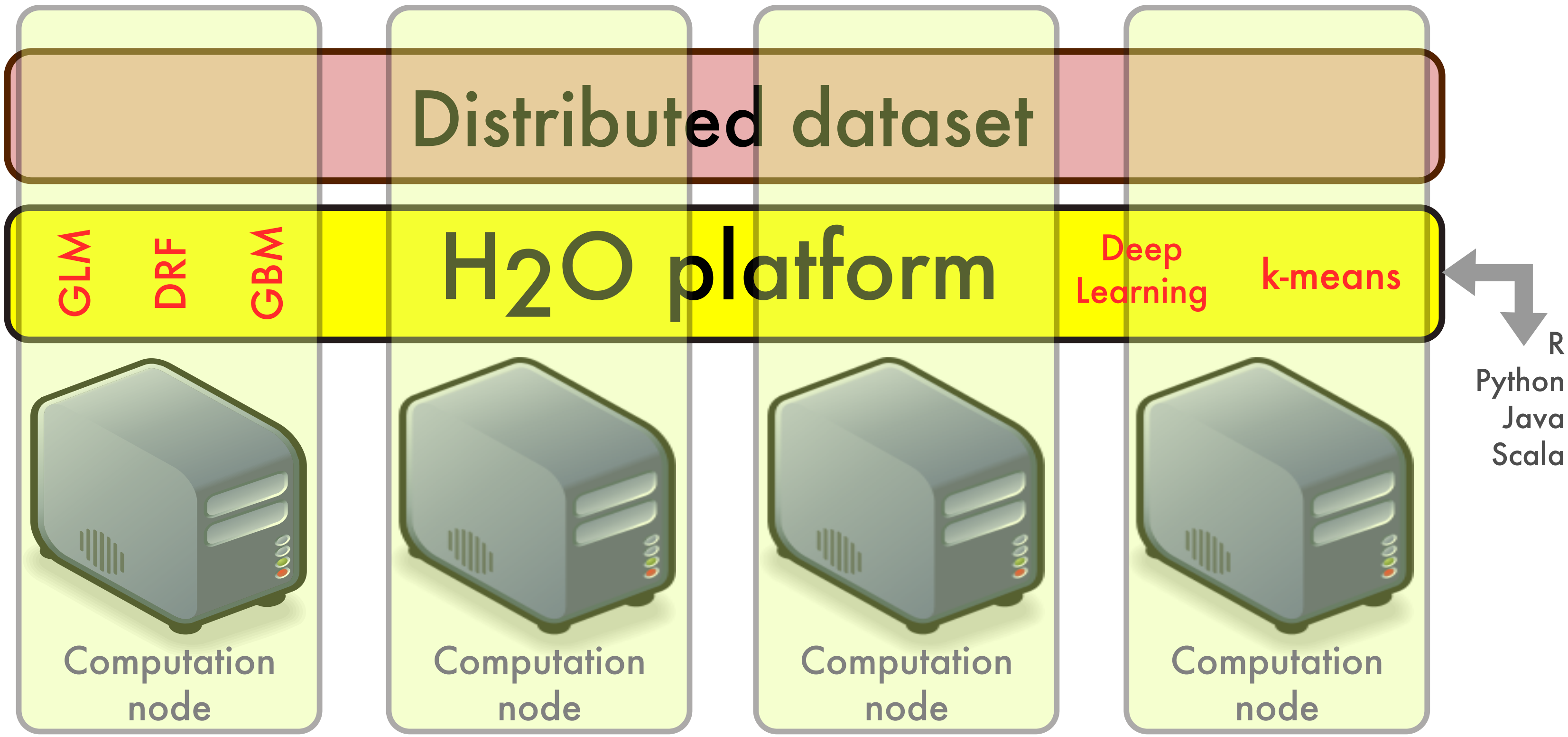


0xdata

Building Forests with H_2O

0xdata

H_2O platform



Challenges

Parallelize and distribute Random Forest algorithm

- Preserve computation with data
- Minimize data transfers

Preserve Random Forest properties

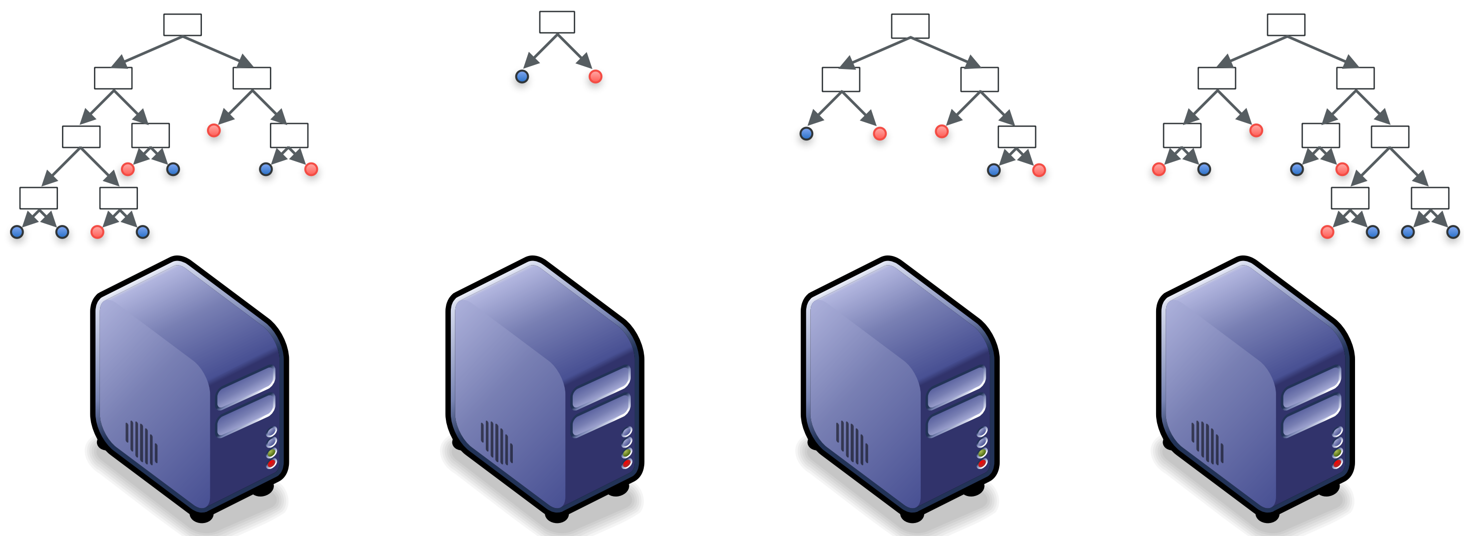
- Split nodes in an efficient way
- Sample and preserve track of OOB samples

Handle large trees

Implementation #1

Build independent trees per machine local data

- RVotes approach
- Each node builds a subset of forest



0xdata

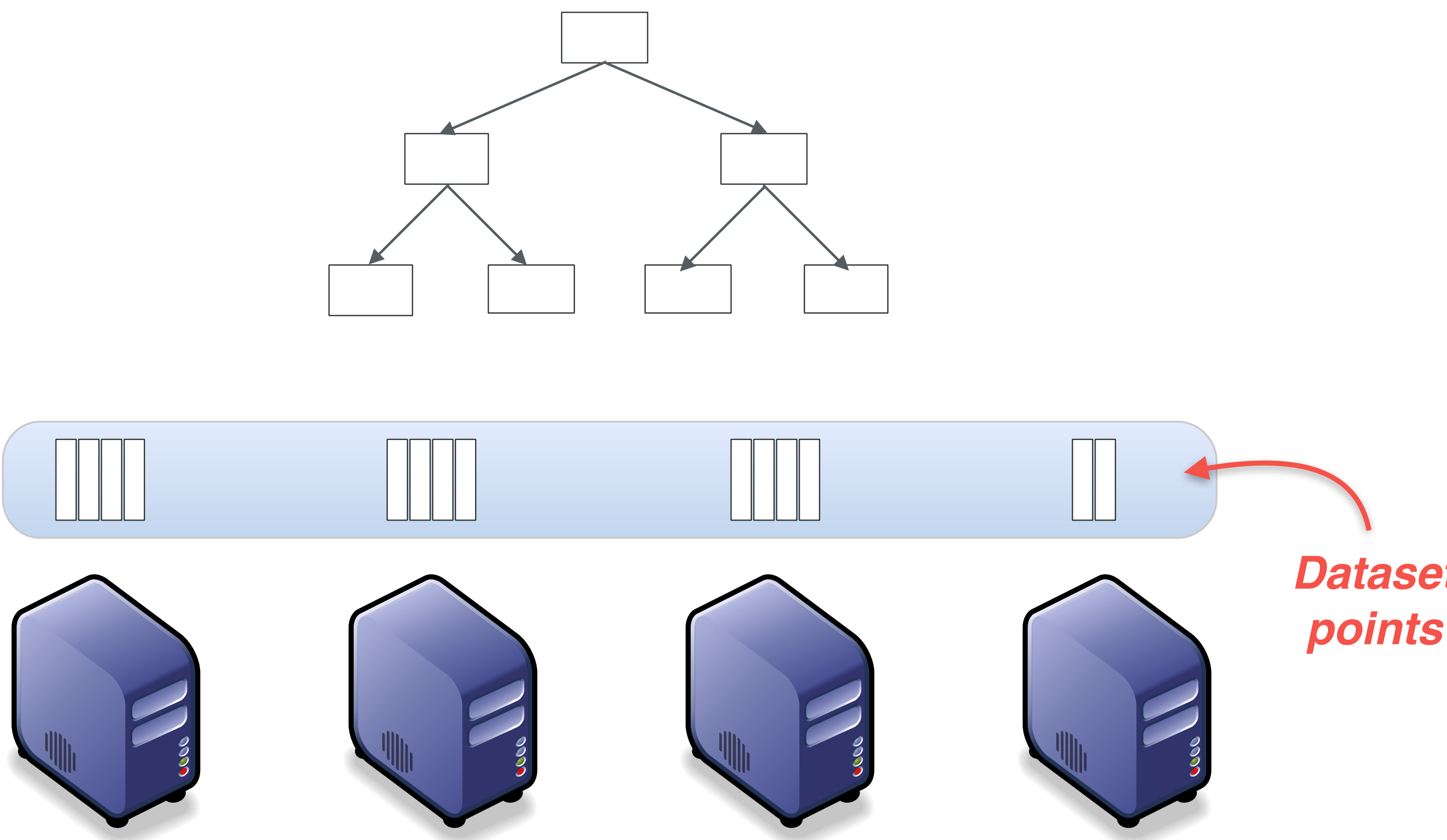
Chawla, N., & Hall, L. (2004). *Learning ensembles from bites: A scalable and accurate approach*. The Journal of Machine Learning Research, 5, p421–451.

Implementation #1

- ✔ Fast - trees are independent and can be built in parallel
- 🚫 Data have to fit into memory
- 🚫 Possible accuracy decrease if each node can see only subset of data

Implementation #2

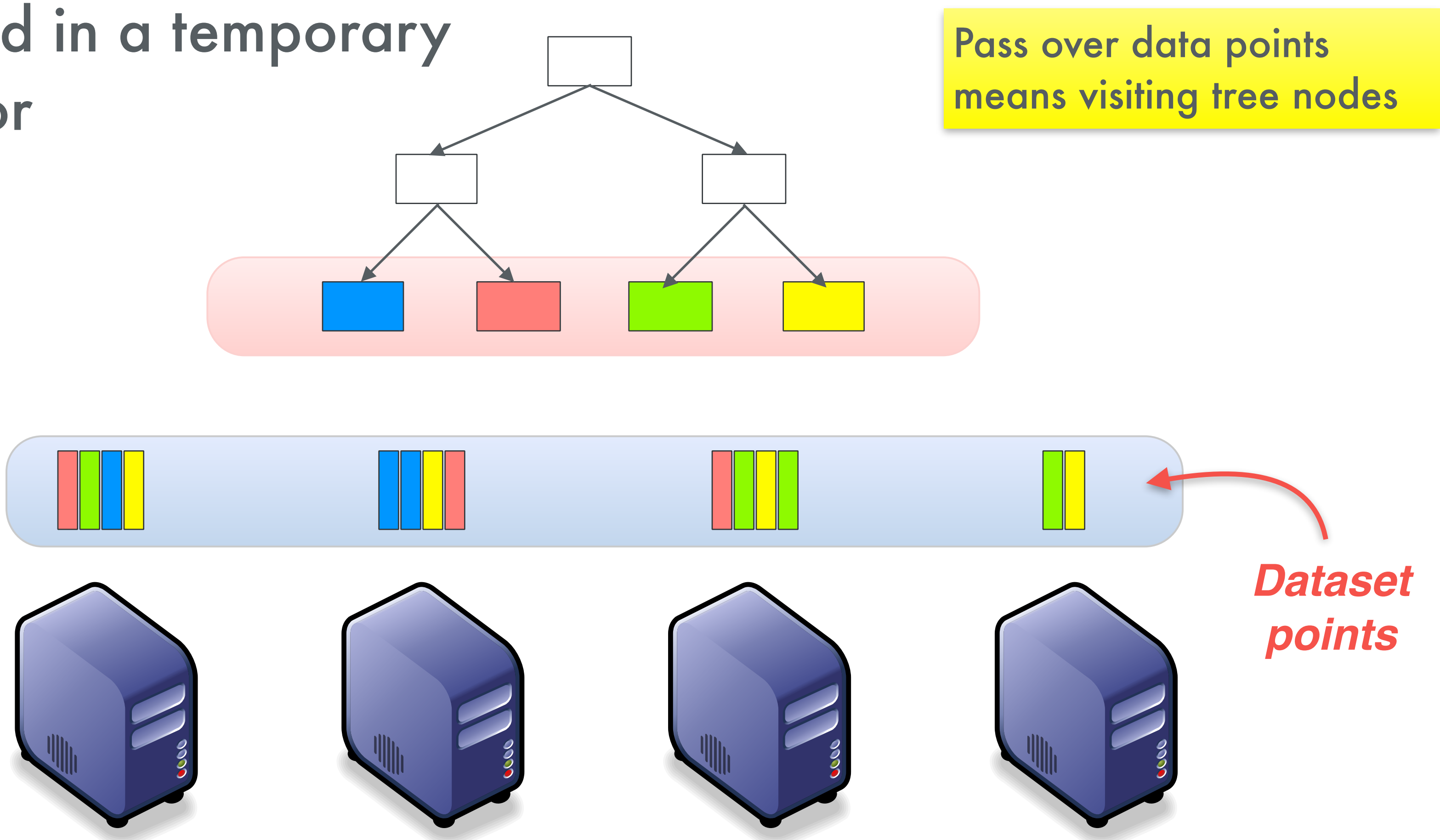
Build a distributed tree over all data



Implementation #2

Each data point has assigned a tree node

- stored in a temporary vector



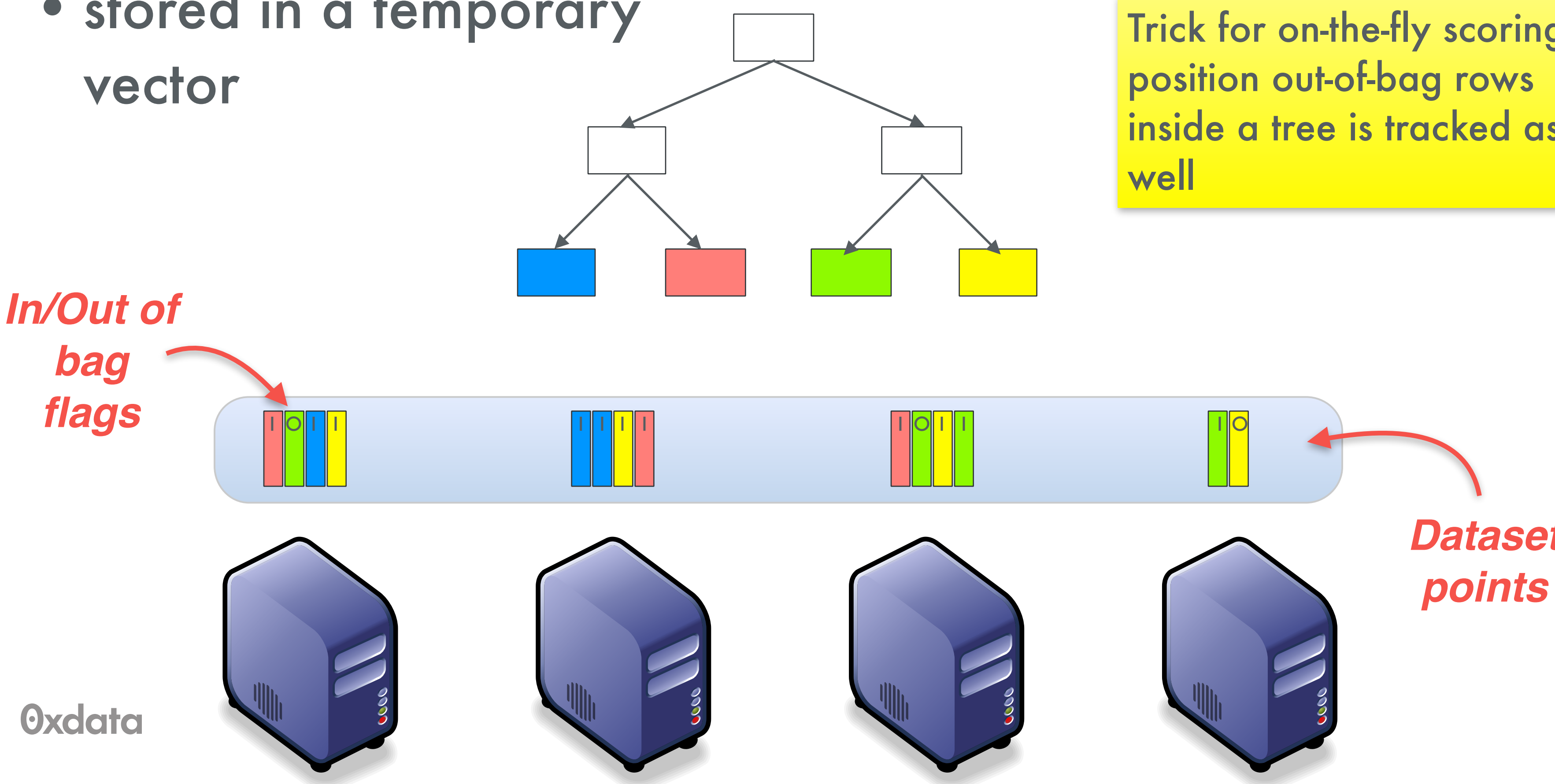
0xdata

Implementation #2

Each data point has in/out of bag flag

- stored in a temporary vector

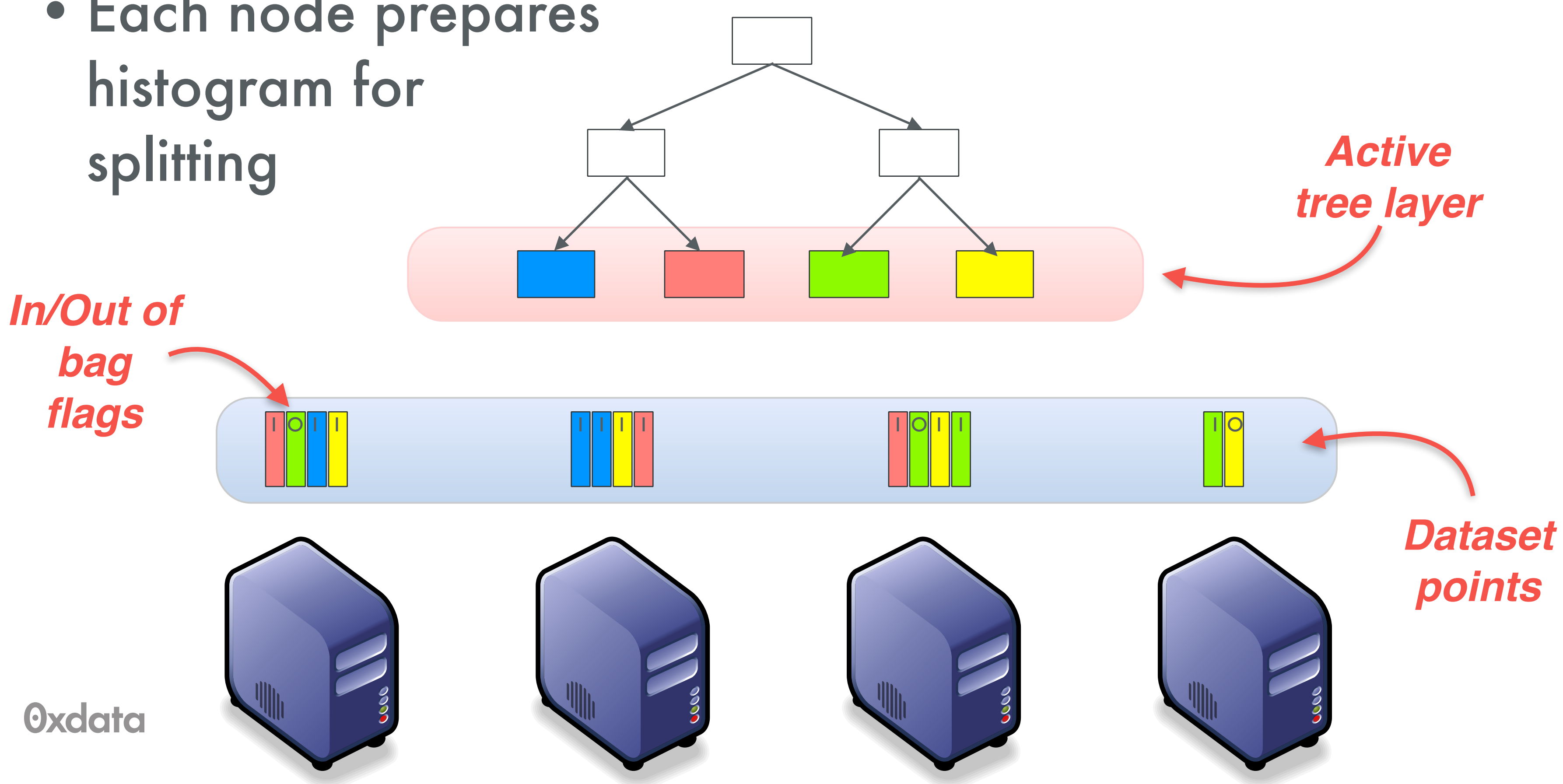
Trick for on-the-fly scoring: position out-of-bag rows inside a tree is tracked as well



Implementation #2

Tree is built per layer

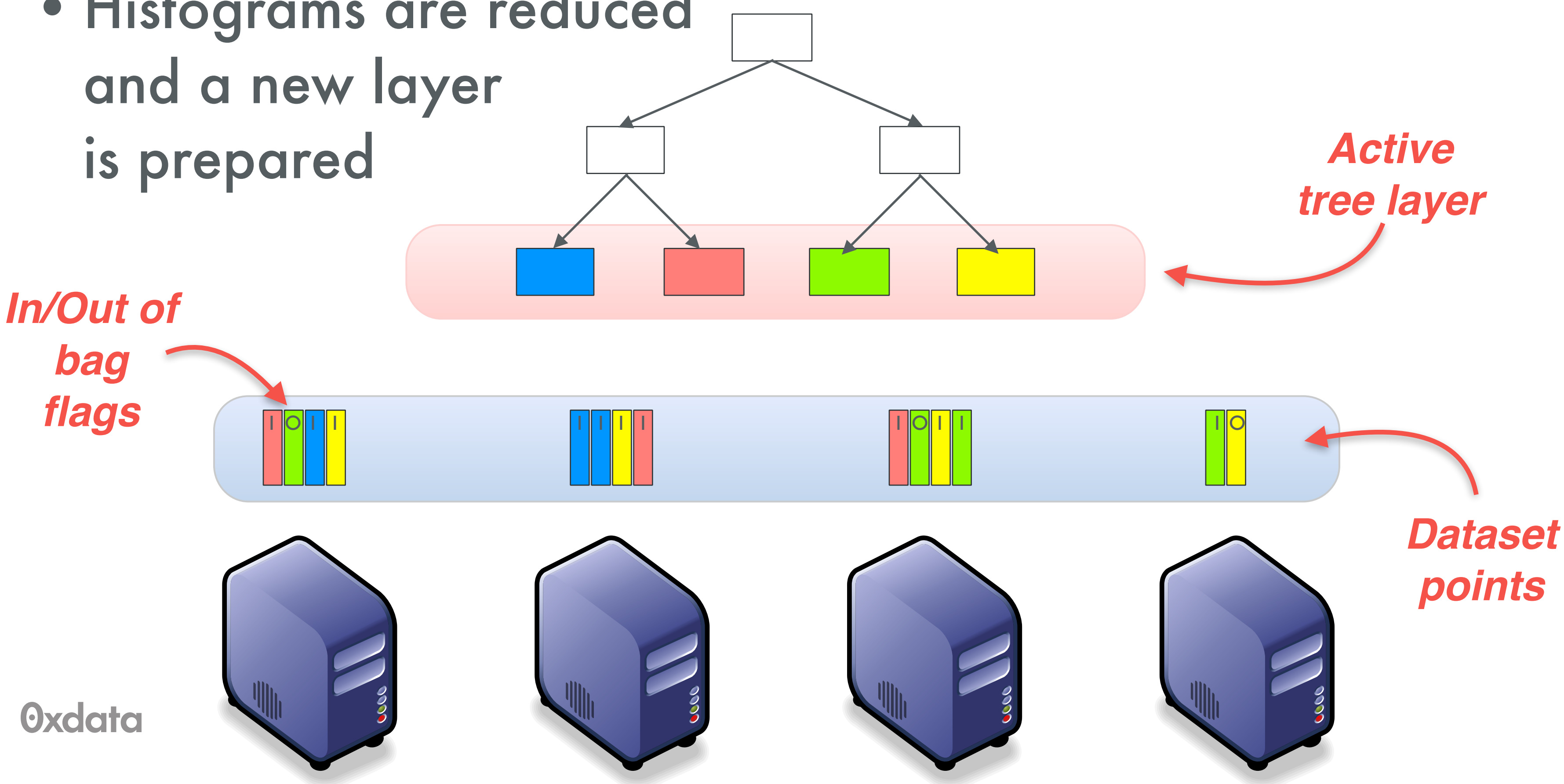
- Each node prepares histogram for splitting



Implementation #2

Tree is built per layer

- Histograms are reduced and a new layer is prepared



Implementation #2

- Exact solution - no decrease of accuracy
- Elegant solution merging tree building and OOB scoring
- More data transfers to exchange histograms
- Can produce huge trees (since tree size depends on data)

Tree representation

Internally stored in compressed format as a byte array

- But it can be pretty huge (>10MB)

Externally can be stored as code

```
class Tree_1 {
  static final float predict(double[] data) {
    float pred =  ((float) data[3 /* petal_wid */] < 0.8200002f ? 0.0f
      : ((float) data[2 /* petal_len */] < 4.835f
        ? ((float) data[3 /* petal_wid */] < 1.6600002f ? 1.0f : 0.0f)
        : ((float) data[2 /* petal_len */] < 5.14475f
          ? ((float) data[3 /* petal_wid */] < 1.7440002f
            ? ((float) data[1 /* sepal_wid */] < 2.3600001f ? 0.0f : 1.0f)
            : 0.0f)
          : 0.0f)));
      return pred;
  }
}
```

Lesson learned

☑ Preserving deterministic computation is crucial!

☑ Trees need to be sent around the cloud for validation which can be expensive!

☑ Tracking out-of-bag points can be tricky!

☑ Clever data binning is a key trick to decrease memory consumption

Thank you!

Time for questions

0xdata

Thank you!

Learn more about H_2O at
0xdata.com
or
git clone https://github.com/0xdata/h2o

Follow us at @hexadata

References

- 0xdata, H_2O: https://github.com/0xdata/h2o/
- Breiman, L. (1999). *Pasting small votes for classification in large databases and on-line*. Machine Learning, Vol 36. Kluwer, p85–103.
- Breiman, L. (2001). *Random forests*. Machine Learning, 5–32. Retrieved from http://link.springer.com/article/10.1023/A:1010933404324
- Breiman, L., Cutler, A. *Random Forest*. http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
- Chawla, N., & Hall, L. (2004). *Learning ensembles from bites: A scalable and accurate approach*. The Journal of Machine Learning Research, 5, p421–451.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. cs.yale.edu. Retrieved from http://link.springer.com/content/pdf/10.1007/978-0-387-84858-7.pdf