Random Forests

MAS DSE-220

Natasha Balac, Ph.D.

Definition

- Random forest (or random forests) is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees
- Leo Breiman and Adele Cutler and "Random Forests" is their trademark
- The method combines Breiman's "bagging" idea and the random selection of features

Classification and Regression Trees

- Divide and conquer
  - Partition data one variable at a time
  - Recursive over same variable to get highly non-linear combinations of features
- Decision Node: split data depending on value of the attribute
- Leaf Node: label data (or estimate data) according to most likely class (value)

Classification and Regression Trees

- Model:
  - Y=fleaf (X) for decisions on X lead to that leaf partition
- Objective: minimize error/misclassifications
- Algorithm:

  initialize a ROOT node

  for each node

  search all x_i for possible partitions

  choose best x_i

  until leafs are pure or tree is deep enough

Classification and Regression Trees

Parameters:

1.criterion for splitting a node

choose best x_i with regards to the objective function

2.criterion to stop splitting

choose a minimum number of data points that fall to each leaf

choose a maximum tree depth

require a minimal improvement

3.prune tree activate

Classification and Regression Trees

- Issue: deeper tree => less points fall to a leaf
  - too few data points => unreliable partition
  - too many => another split could improve model

Solution: Cross Validation helps determine depth

Solution: Pruning helps avoid overfitting

Issue: Decision boundaries are sharp and perpendicular to input dimensions

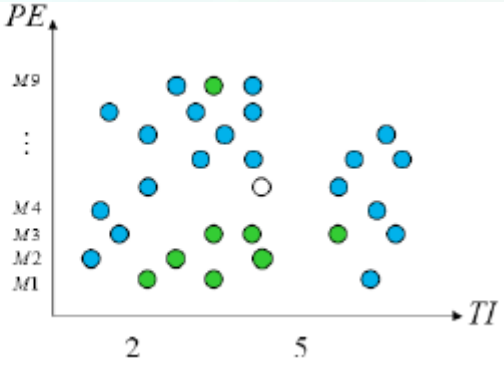
Solution: take more than 1 variable at time

Better: bootstrap and aggregate

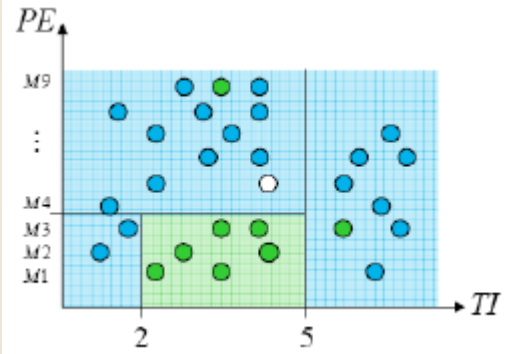
Decision Trees

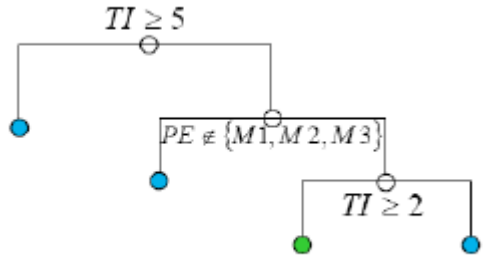
Simple dataset with two predictors

TI	PE	Response
1.0	M2	good
2.0	M1	bad
...	...	...
4.5	M5	?



Greedy, recursive partitioning along

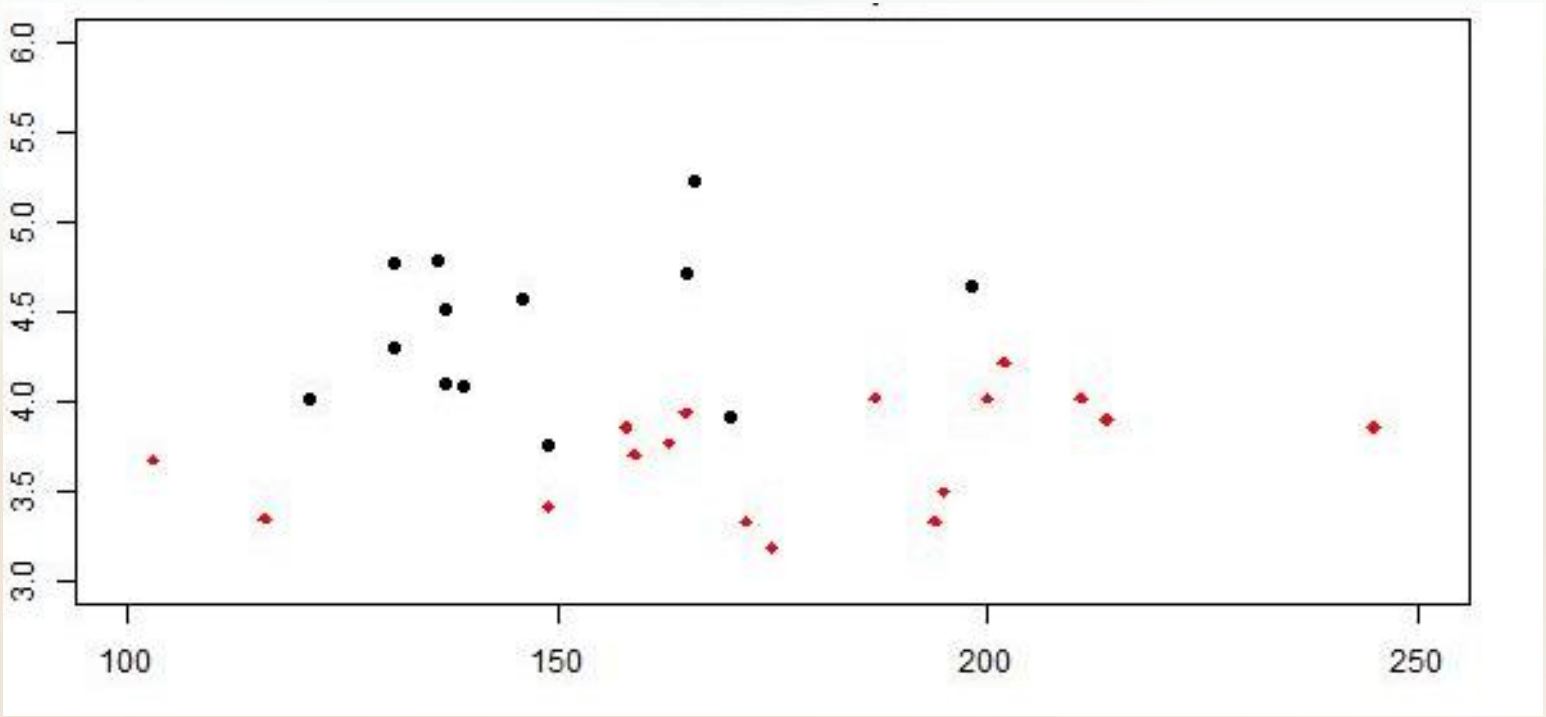




Divide and Conquer

Where to split so for classification or regression?

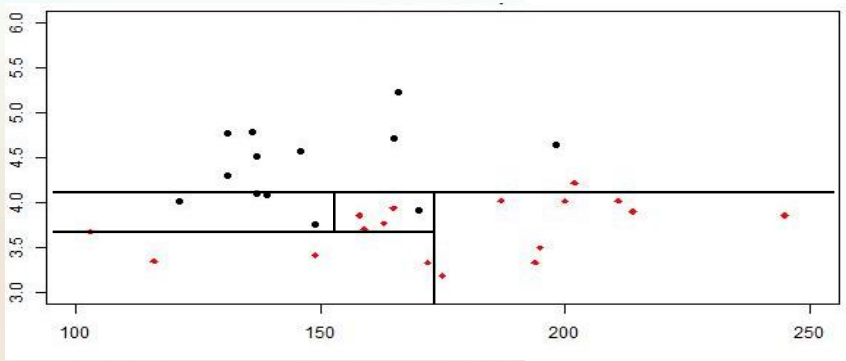
ERA



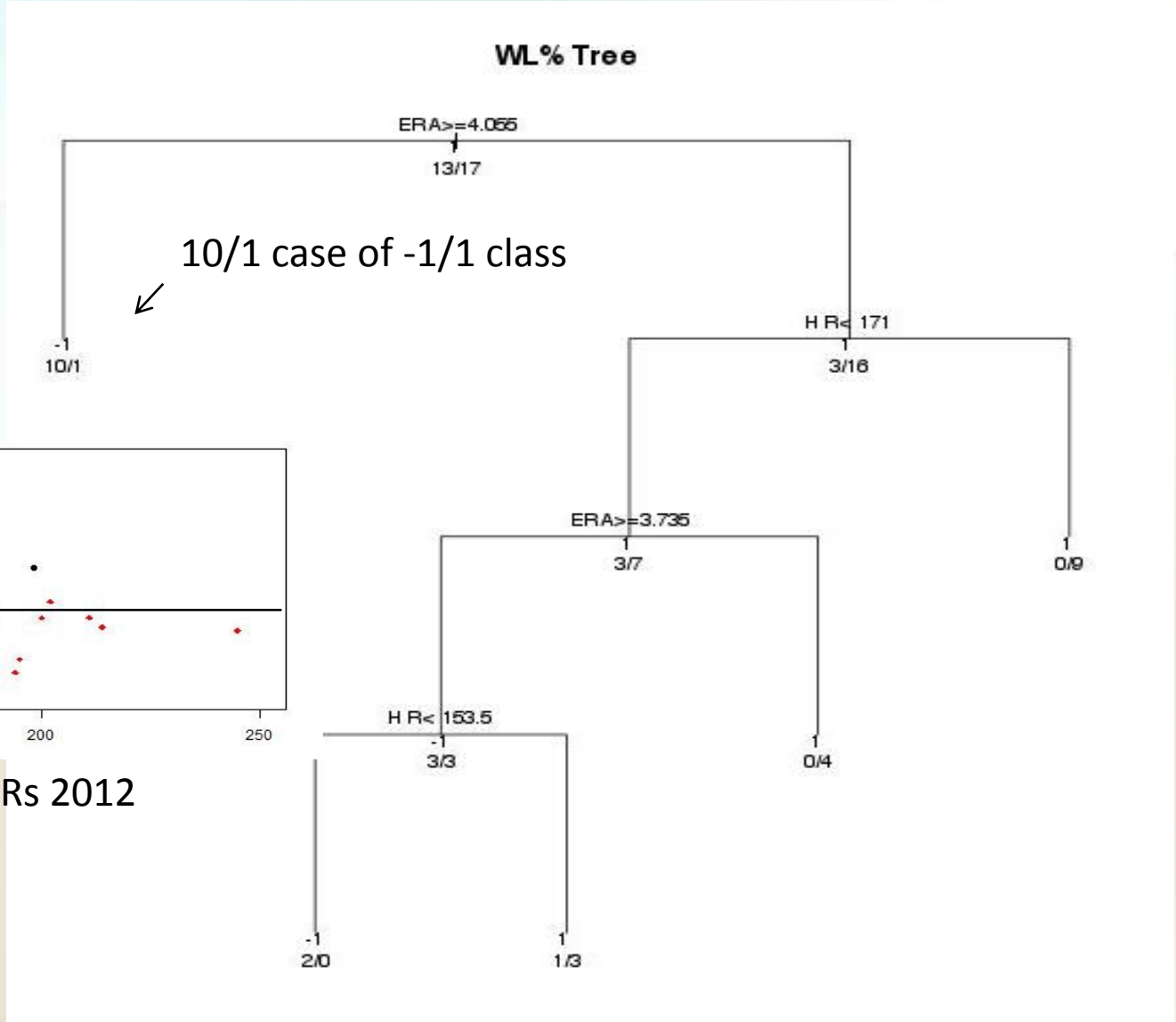
Number of Team HRs

Example: P. Rodroguez, 2014

A Classification Tree on WL%



Number of Team HRs 2012

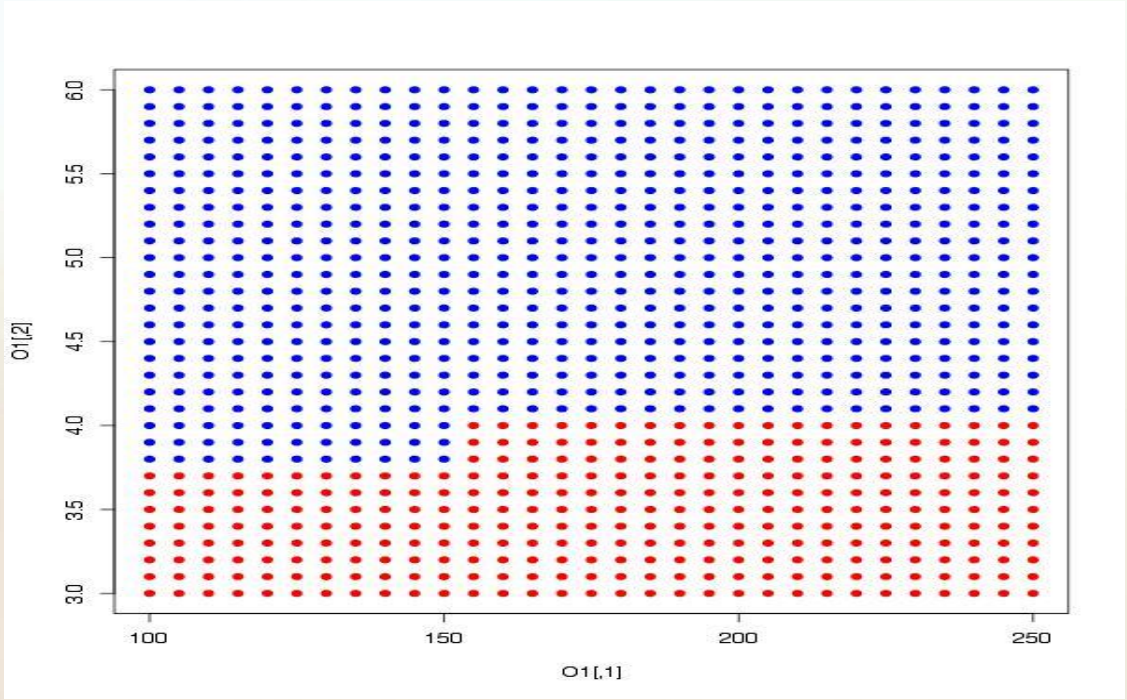


Example: P. Rodroguez, 2014

Decision Tree Issues

- non-linear decision boundary, but with sharp corners (can't take diagonal partitions)

ERA



blue=>point classified as WL% < .50

red=>point classified as WL% > .50

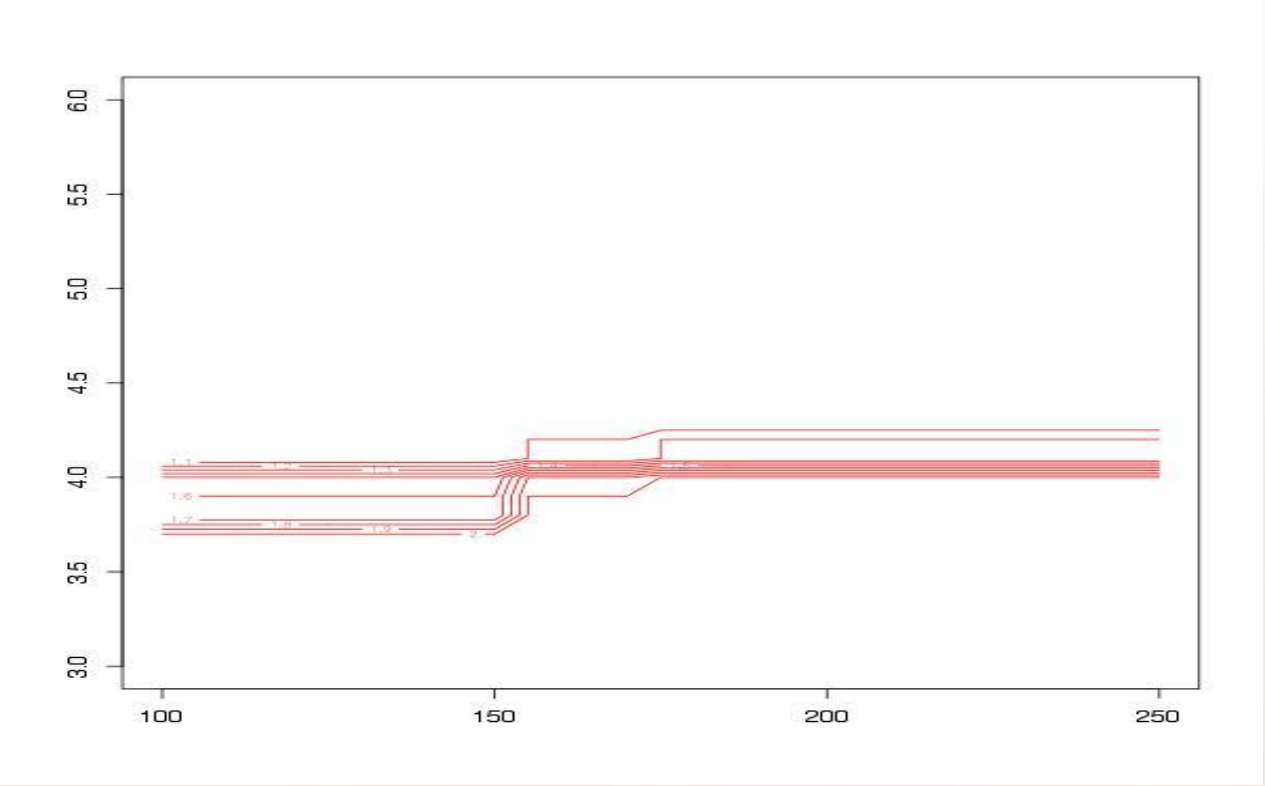
Number of Team HRs

Example: P. Rodroguez, 2014

Classification Tree With Bootstrapping

- 10 samples of data used for 10 trees

ERA



Number of Team HRs

average classification at each grid point,

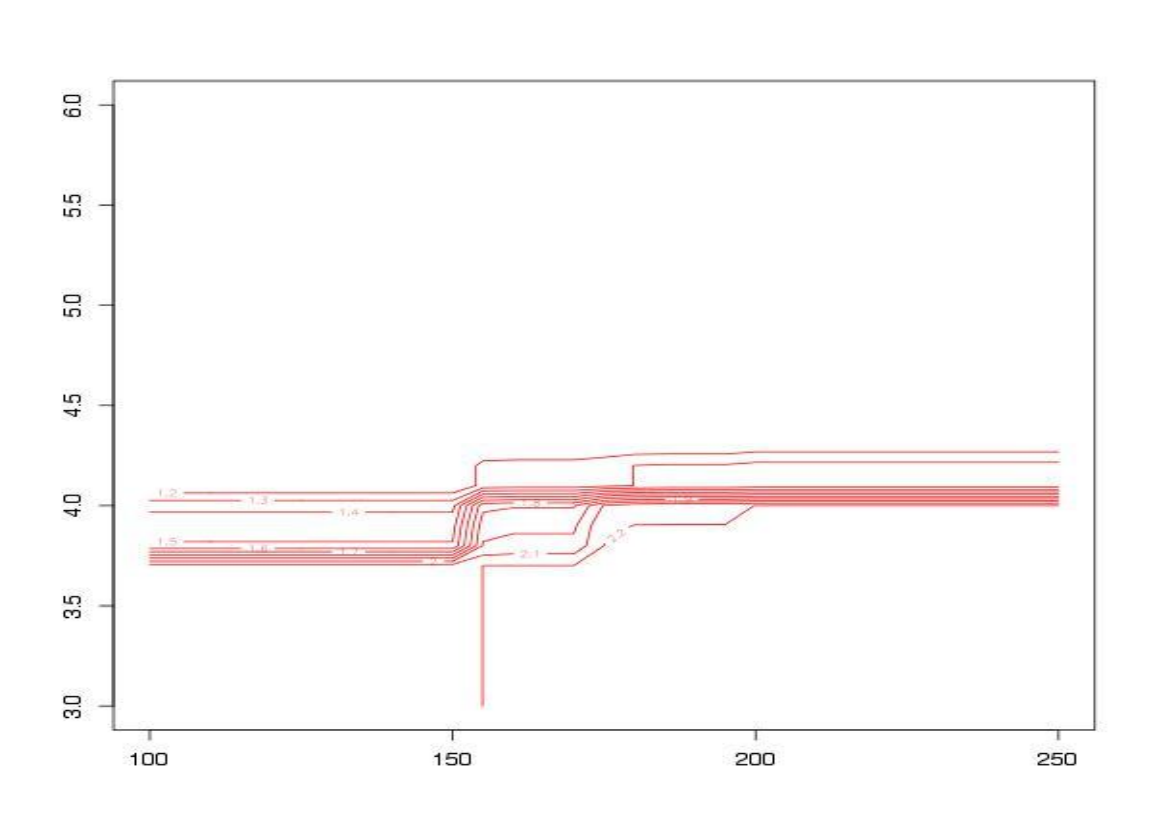
make counter plot of average

Example: P. Rodroguez, 2014

Classification With Bootstraps

- 100 samples => softer boundary

ERA



Number of Team HRs

average classification at each grid point

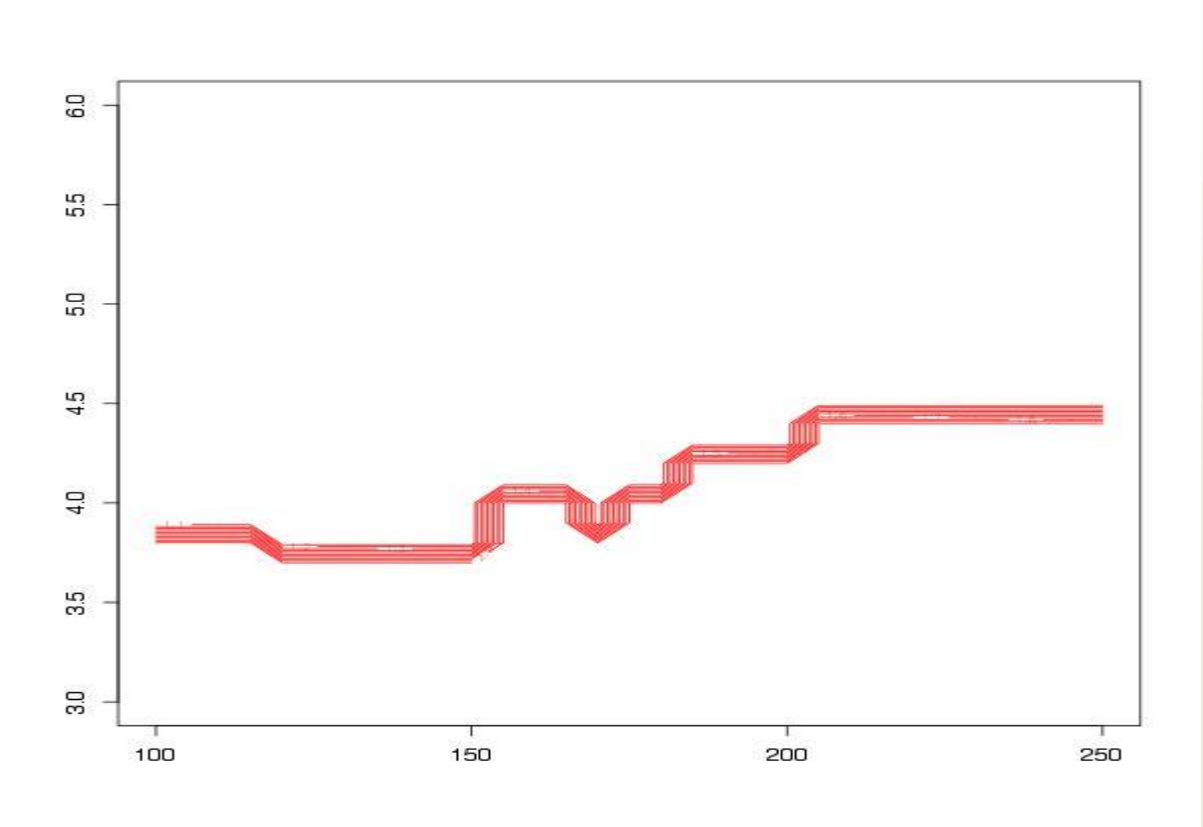
make counter plot of average

Example: P. Rodroguez, 2014

Random Forest

- "Bagging"=bootstrap & aggregate samples, gives soft and highly flexible boundary

ERA

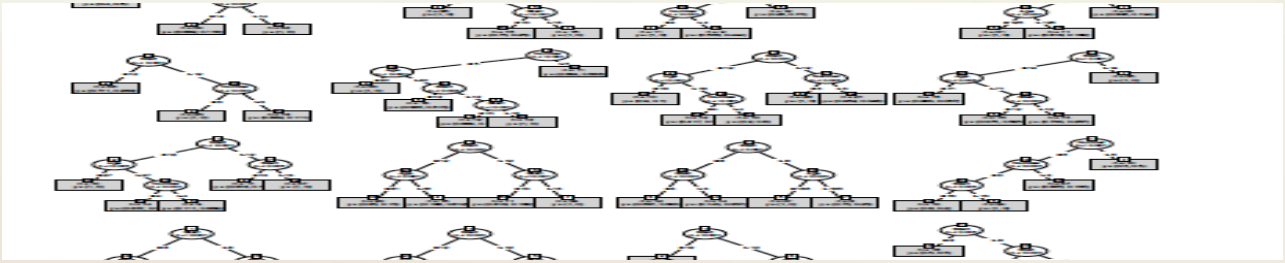


Number of Team HRs 2012

Example: P. Rodroguez, 2014

Construction of Random Forest

- Over and above recursive partitioning do:
  - Take Bootstrap samples of observations (ntree)
  - Fit a regression/classification tree to each sample
  - During construction, choose the best split only from a subset of features (mtry)
  - Result: an ensemble of diverse trees



  - Aggregating over an ensemble reduces prediction variance

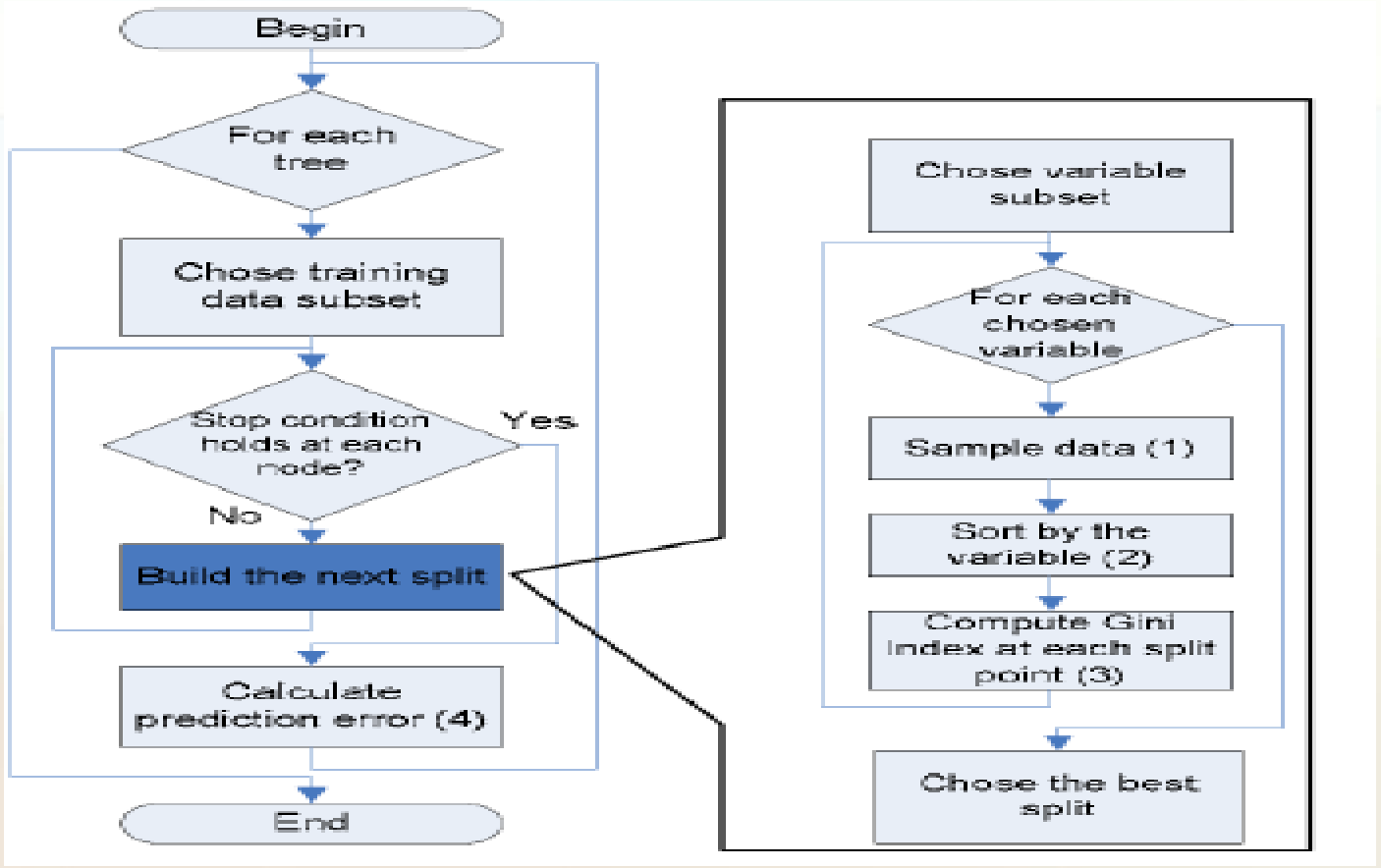
Random Forest Algorithm

- Each tree is constructed using the following algorithm:
  - Let the number of training cases be N, and the number of variables in the classifier be M
  - We are told the number m of input variables to be used to determine the decision at a node of the tree; m should be much less than M
  - Choose a training set for this tree by choosing n times with replacement from all N available training cases (i.e. take a bootstrap sample)
  - Use the rest of the cases to estimate the error of the tree, by predicting their classes
  - For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set
  - Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier)

Random Forest Prediction

- For prediction a new sample is pushed down the tree
- It is assigned the label of the training sample in the terminal node it ends up in
- This procedure is iterated over all trees in the ensemble
- The average vote of all trees is reported as random forest prediction

Random Forest Flow Chart



Practical Consideration

- Splits are chosen according to a purity measure:
  - E.g. squared error (regression), Gini index (classification)
- How to select N?
  - Build trees until the error no longer decreases
- How to select M?
  - Try to recommend defaults, half of them and twice of them and pick the best

New Control Parameter N

- Ntree, n_estimator (number of trees)
  - 1 tree per bootstrap sample
  - Larger is better, but at some point useless
  - Helps avoid overfitting
  - Alleviates need for pruning trees through bagging

New Control Parameter M

- Mtrys, max_feature (size of variable subsets) control diversity
  - As M decreases -> trees are less correlated
    - (different splits, but not all node interactions)
  - As M increases -> trees are more correlated
    - (similar splits, more possible interactions)
  - Essentially, trade off in bias and variance
  - Defaults: $M=\sqrt{P}$ for classification
    - $M= P/3$ for regression

A Tree or a Forest

- Same as Classification/Regression Tree:
  - Need to find splits, build tree
- Different than Tree
  - In principle new parameters maybe easy to set:
    - ntree, can just get large
    - node size (ie bucket size), can just be set low
    - mtry, not obvious but sqrt(P) or P/3 seems good
  - Variable importance is new measure
  - Performance less sensitive to particular points (lower variance), smoother decision thresholds
  - More computation
  - Less interpretable (no final tree to visualize!)
  - Aggregating over an ensemble reduces prediction variance

In R: randomForest

R code:
install.packages("randomForest")library("randomForest")

```
tree_result=randomForest(X,as.factor(Y),
                         ntree=100,        #number of trees
                         mtry=sqrt(P),     #number of variables to sample
                         importance=T,   #estimate importance
                         nodesize=1);    #number of example at each node
```

In Python

- class sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)

- http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Features and Advantages

The advantages of random forest are:

- It is one of the most accurate learning algorithms available
  - For many data sets, it produces a highly accurate classifier
- It runs efficiently on large databases
- It can handle thousands of input variables without variable deletion
- It gives estimates of what variables are important in the classification
- It generates an internal unbiased estimate of the generalization error as the forest building progresses
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing

Disadvantages

- Random forests have been observed to overfit for some datasets with noisy classification or regression tasks
- For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels
  - Therefore, the variable importance scores from random forest are not reliable for this type of data

Additional information

Estimating the test error:

- While growing forest, estimate test error from training samples
- For each tree grown, 33-36% of samples are not selected in bootstrap, called out of bootstrap (OOB) samples
- Using OOB samples as input to the corresponding tree, predictions are made as if they were novel test samples
- Through book-keeping, majority vote (classification), average (regression) is computed for all OOB samples from all trees
- Such estimated test error is very accurate in practice, with reasonable N

Summary

- Extremely fast
  - Fast to build - even faster to predict
  - Practically speaking, not requiring cross-validation alone for model selection significantly speeds training by 10x-100x or more
  - Fully parallelizable
- Automatic predictor selection from large number of candidates
- Resistance to over training
- Ability to handle data without preprocessing
  - data does not need to be rescaled, transformed, or modified
  - resistant to outliers
  - automatic handling of missing values
- Cluster identification can be used to generate tree-based clusters through sample proximity

New Measures of Variable Importance

- Bootstrap samples leave out some data points
- Use these OOB (out of bag) points for testing
  - For each node in each tree:
    - record OOB predictions
    - permute values and record OOB prediction

$$VI_i = AVE_{trees}(\%\text{correct before permuting} - \%\text{correct after permuting})$$

Note: node importance calculated during tree construction is related to first term

Questions?