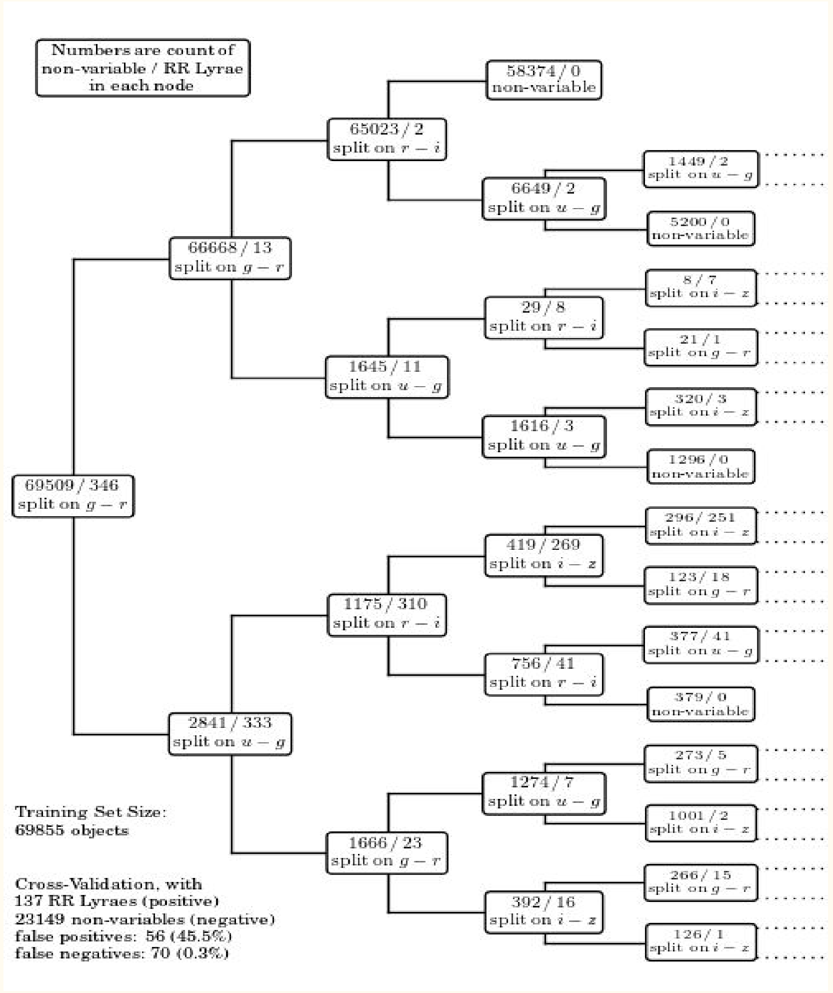
Decision Trees

Chase Hatcher
19 October 2016
ASTR 503/703

What Are Decision Trees?

- Classification method that progressively splits data at a decision boundary
- Goal: put every data point into a leaf node on the tree which provides a probability indicating its class



Training Set Size:
69855 objects

Cross-Validation, with
137 RR Lyraes (positive)
23149 non-variables (negative)
false positives: 56 (45.5%)
false negatives: 70 (0.3%)

Why Are They Useful?

- Simple to implement
- Intuitive and easy to interpret
- Relatively accurate, relatively fast
- Works with any number of classifications
- Works with any number of parameters, both continuous and/or discrete

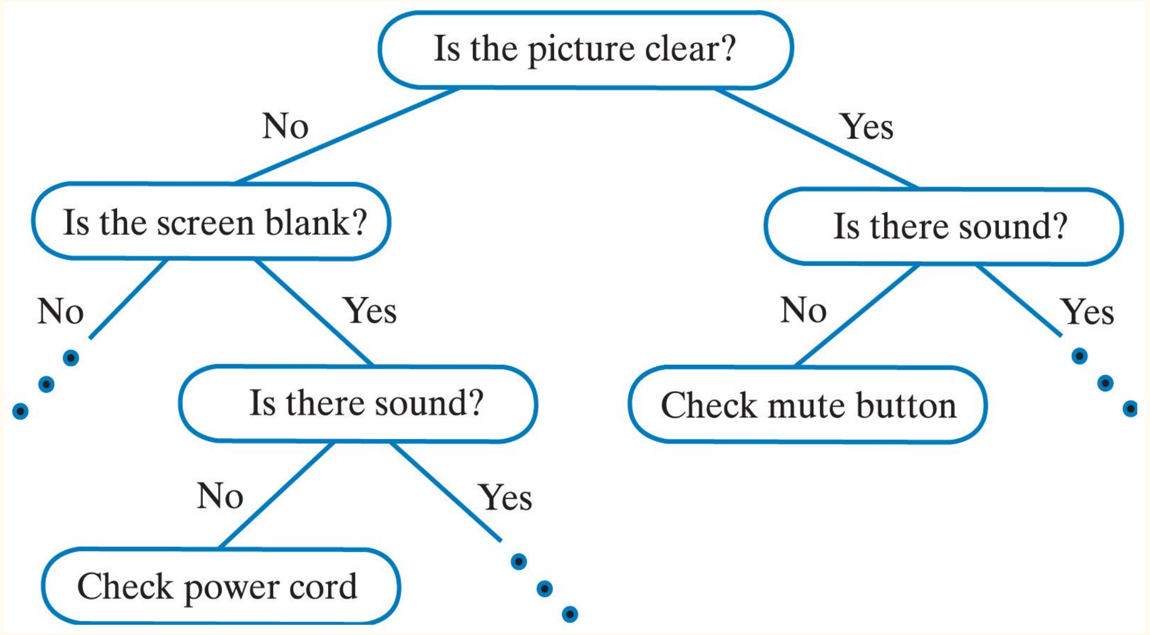


Image credit-http://www.sfs.uni-tuebingen.de/~vhenrich/ss12/java/homework/hw7/decisionTrees.html

Decision Boundaries

- Decision boundaries tell you where to split the data
- First we find the entropy of the data
  - Entropy is the information content of a pdf
- Then we find the information gain
  - Information gain is the difference between the entropy of the parent node and the sum of the entropies of the two children nodes. Here we have IG for a binary class decision:

$$E(x) = -\sum_i p_i(x)\ln(p_i(x)),$$

$$IG(x|x_i) = E(x) - \sum_{i=0}^{1} \frac{N_i}{N} E(x_i),$$

Decision Boundaries

- To choose the parameter on which to split, test all parameters and select the one that yields the largest IG
- To choose the value of the parameter at which to split, test multiple split points and select the one that yields the largest IG
- The best split point *s* for a given feature is defined as follows:

$$IG(x|s) = E(x) - \arg\max_{s} \left(\frac{N(x|x < s)}{N} E(x|x < s) - \frac{N(x|x \geq s)}{N} E(x|x \geq s) \right)$$

Errors and Accuracy

- Misclassification error - probability that a randomly selected data point will be misclassified
- Gini coefficient - estimate of the probability that a random data point would be incorrectly classified if it was classified randomly from the distribution of classifications of the data set

$$MC = 1 - \max_i(p_i).$$

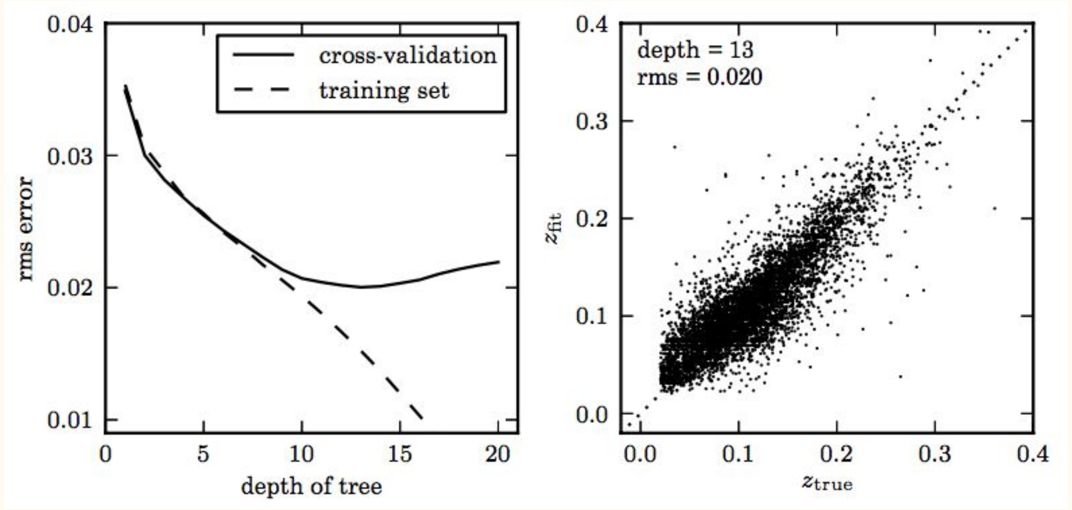
$$G = \sum_i^k p_i(1 - p_i),$$

Stopping the Tree

- When a node has only one class of data in it, it is a leaf node
- When no splitting will increase the IG of a node, it is a leaf node
- When a node has a predefined minimum number of data points, it is a leaf node

Overfitting

- You could keep splitting until there's one data point per node, but that would be overfitting
  - This means your tree would be too deep and the correlation of the data would get drowned out by the noise in the training set
- In general, as tree depth increases, error decreases *up to a point*
  - We use cross validation to find this point



Controlling Overfitting

- Obviously, we could just limit the depth of the tree to the optimized point indicated by the cross validation
- A more accurate approach is to grow the tree until each node reaches a predefined minimum number of points, then go back and terminate (prune) those nodes at which we find that terminating improves the accuracy of the classifications

Visualizing

- First step splits the data
- Next step splits each side of the first split
- Next steps continue splitting each section as needed
- At the end, a single class is attributed to each section (leaf node)

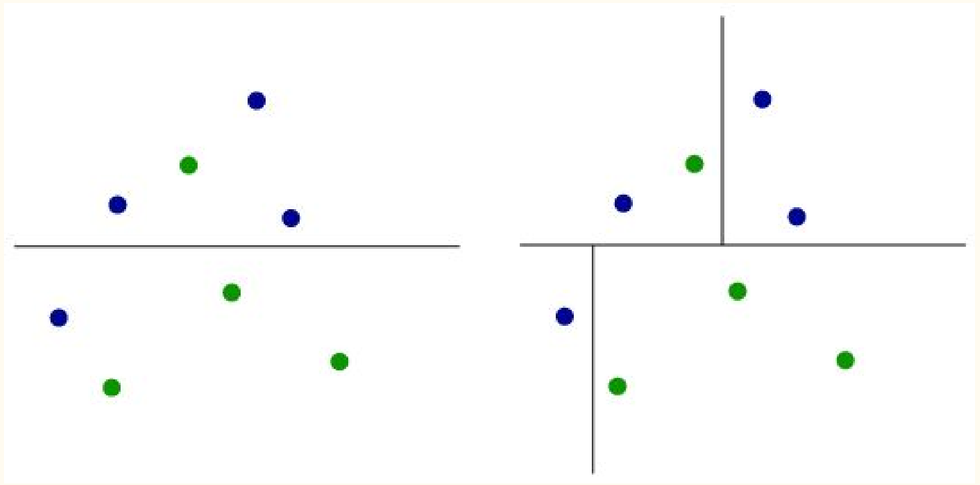
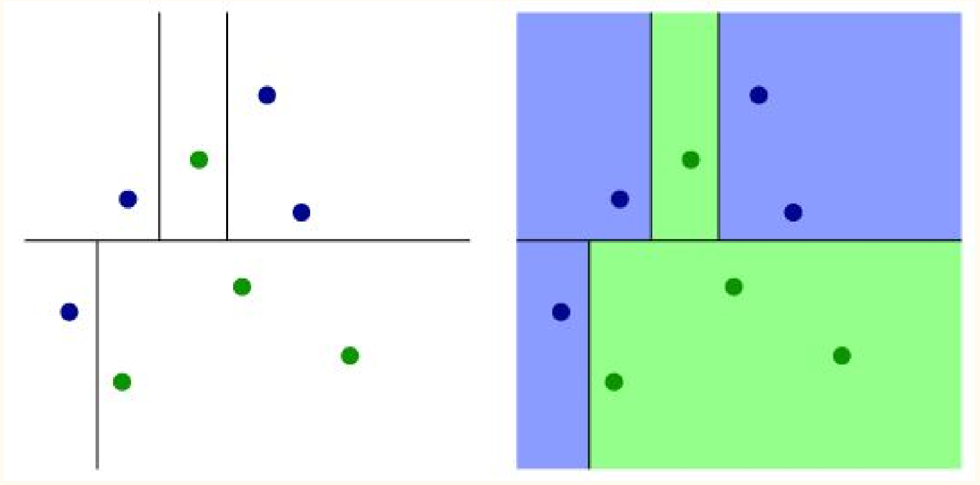
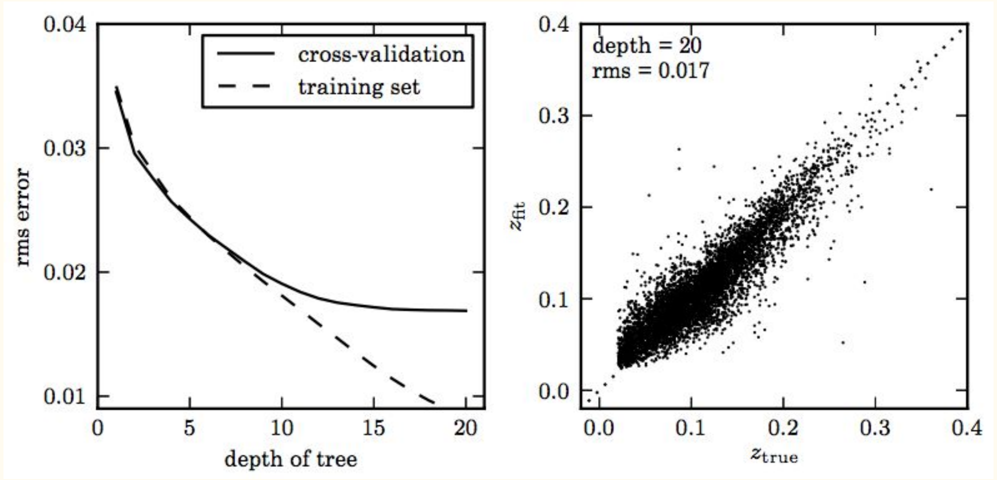


Image source - https://shapeofdata.wordpress.com/2013/07/02/decision-trees/



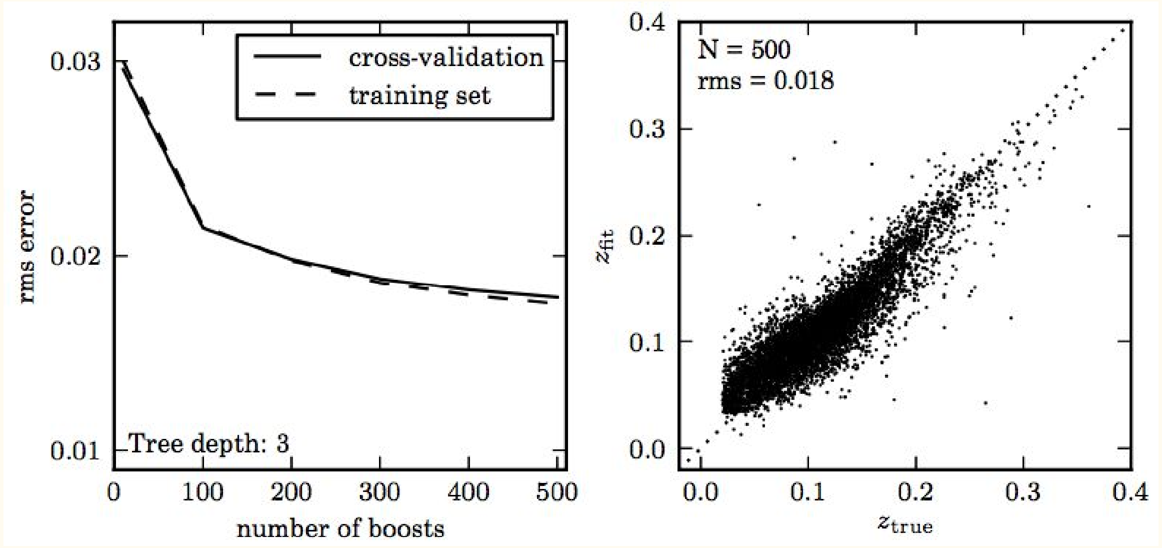
Random Forests

- Random forests make a set of decision trees from bootstrap samples on a training set and average their classifications
  - Define the number of trees N to be generated and the number of parameters M (less than total # parameters) to consider splitting on at each level
  - Randomly select M different parameters to split on at each level, find the one that yields the largest IG, and choose to split on that one. Repeat for N trees.
  - Average the results of N trees

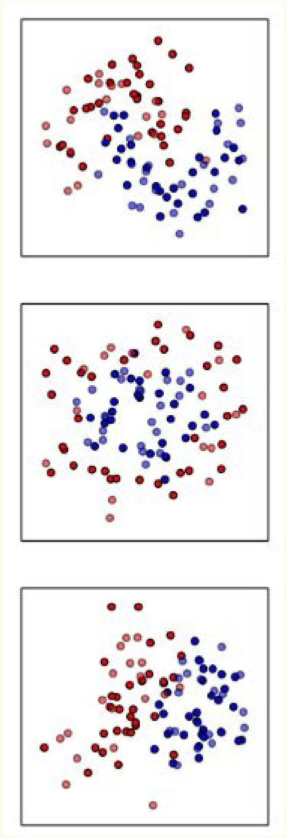


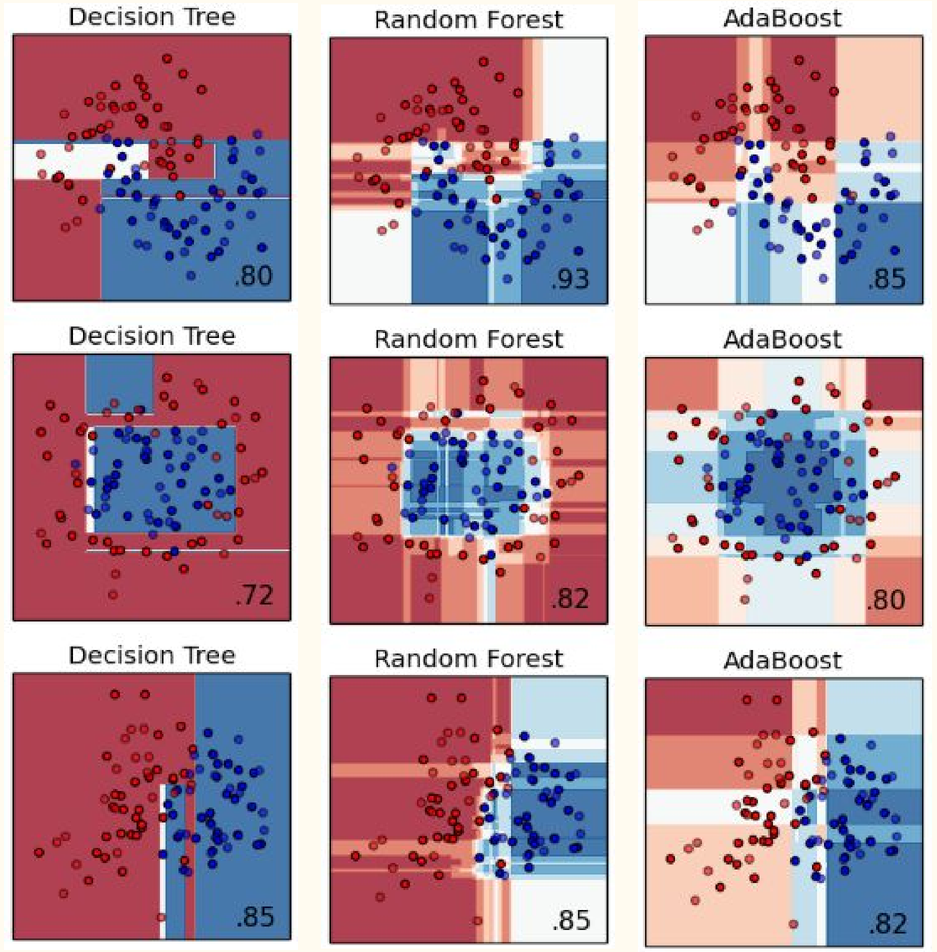
Boosting

- Boosting involves running the classifier many times and reweighting data at each iteration based on the accuracy of classifications at the previous iteration
  - An iteration is weighted more heavily in the final classification if it provides a more accurate classification



Method Comparison



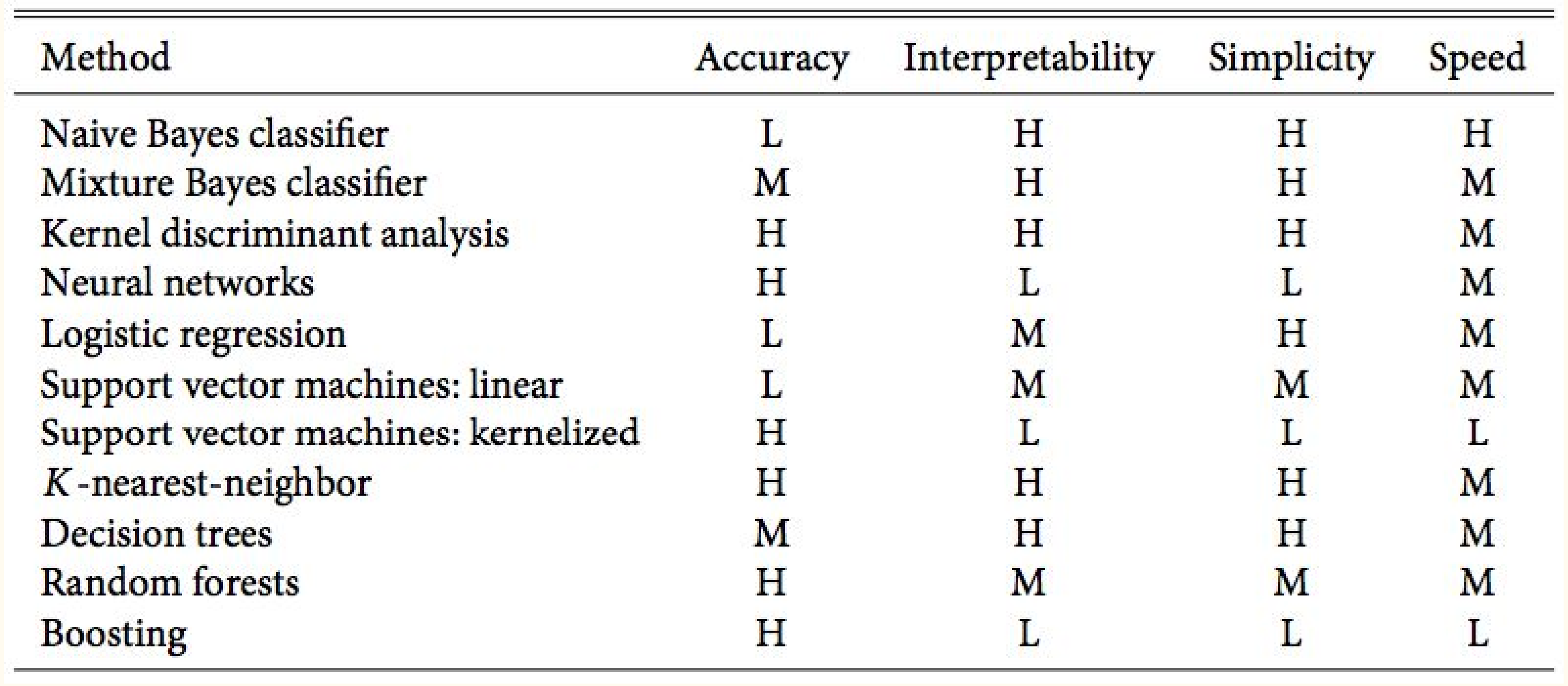


http://scikit-learn.org/0.15/_images/plot_classifier_comparison_0011.png

Computations

- In general, decision trees require $O(N\log N)$ to build and $O(\log N)$ to classify
    - Increasing complexity (depth of the tree, number of additional methods) increases the number of computations
- Random forests and boosting methods increase accuracy, but they can increase the number of computations
    - Random forests can be implemented in parallel, so they don't increase the number of computations excessively, but they're not as simple to implement or interpret
    - Boosting cannot be implemented in parallel because each iteration relies on information from the previous iteration, making it very computationally expensive. It is not as practical for very large data sets. It is also not as easy to interpret or implement.

Comparison of Classification Methods



Method	Accuracy	Interpretability	Simplicity	Speed
Naive Bayes classifier	L	H	H	H
Mixture Bayes classifier	M	H	H	M
Kernel discriminant analysis	H	H	H	M
Neural networks	H	L	L	M
Logistic regression	L	M	H	M
Support vector machines: linear	L	M	M	M
Support vector machines: kernelized	H	L	L	L
K-nearest-neighbor	H	H	H	M
Decision trees	M	H	H	M
Random forests	H	M	M	M
Boosting	H	L	L	L

Python Packages

- DecisionTreeClassifier in sklearn.tree
- RandomForestClassifier in sklearn.ensemble
- AdaBoostClassifier in sklearn.ensemble

References

- Chapter 9 in Ivezic et al. - all images and equations unless otherwise noted
- Hamilton et al. http://dms.irb.hr/tutorial/tut_dtrees.php
- https://en.wikipedia.org/wiki/Random_forest
- http://scikit-learn.org/stable/modules/ensemble.html
- https://shapeofdata.wordpress.com/2013/07/02/decision-trees/