Ensemble methods

Random Forest

Ensemble methods

- Use multiple models to obtain better predictive performance
- Combine multiple learners to produce a strong learner
- Typically much more computation, since you are training multiple learners
- Typically combine multiple fast learners (like decision trees)
- Tend to overfit
- Tend to get better results since there is deliberately introduced significant diversity among models

Bagging: Bootstrap aggregating

- Each model in the ensemble votes with equal weight
- Train each model with a random training set

Boosting

- Incremental
- Build new models that try to do better on previous model's mis-classifications
  - Can get better accuracy
  - Tends to overfit
- Adaboost is canonical boosting algorithm

Random forest

- **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.
- The term came from **random decision forests** that was first proposed by Tin Kam Ho of Bell Labs in 1995.
- The method combines Breiman's "bagging" idea and the random selection of features.

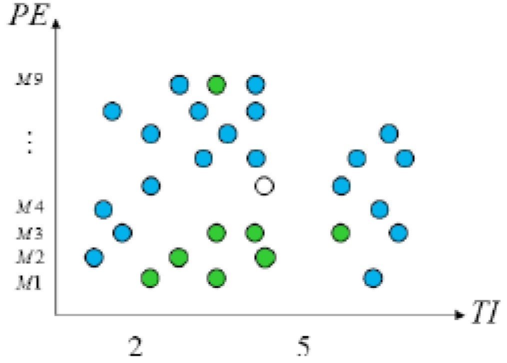
Decision trees

- Decision trees are individual learners that are combined. They are one of the most popular learning methods commonly used for data exploration.
- One type of decision tree is called CART… classification and regression tree.
- CART … greedy, top-down binary, recursive partitioning, that divides feature space into sets of disjoint rectangular regions.
  - Regions should be pure wrt response variable
  - Simple model is fit in each region – majority vote for classification, constant value for regression.

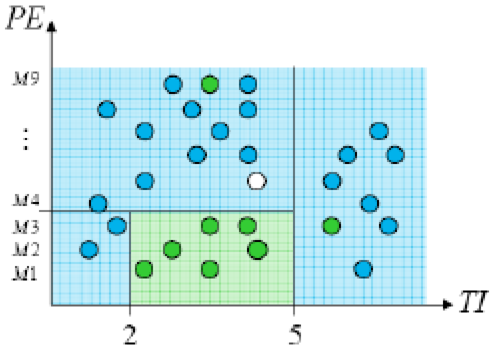
Decision tress involve greedy, recursive partitioning.

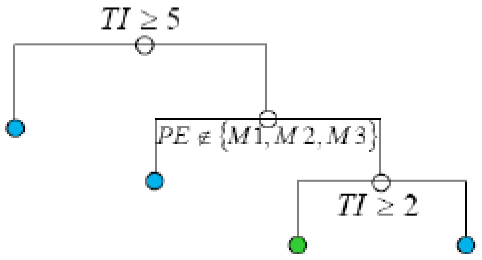
- Simple dataset with two predictors

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



- Greedy, recursive partitioning along TI and PE





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 has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data.