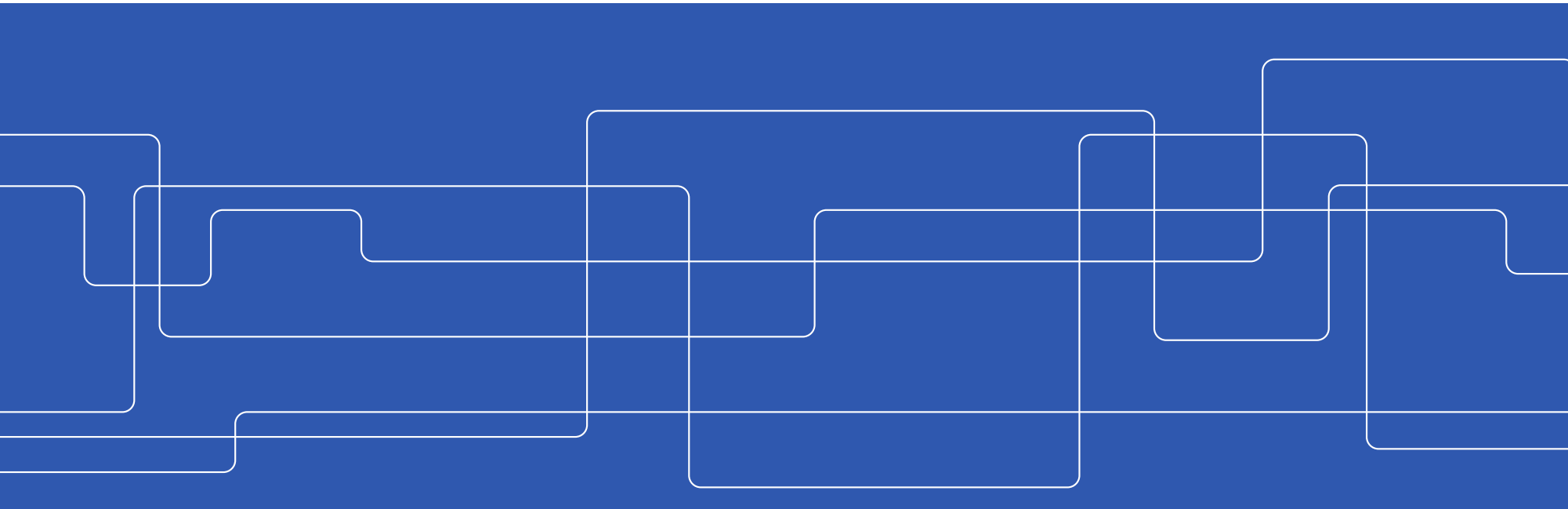
ID2214/FID3214 2021

Programming for Data Science – Combining Models

Henrik Boström

Prof. of Computer Science - Data Science Systems
Division of Software and Computer Systems
Department of Computer Science
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
bostromh@kth.se



Outline

Condorcet's jury theorem

Bagging

Randomization

Random forests

Boosting

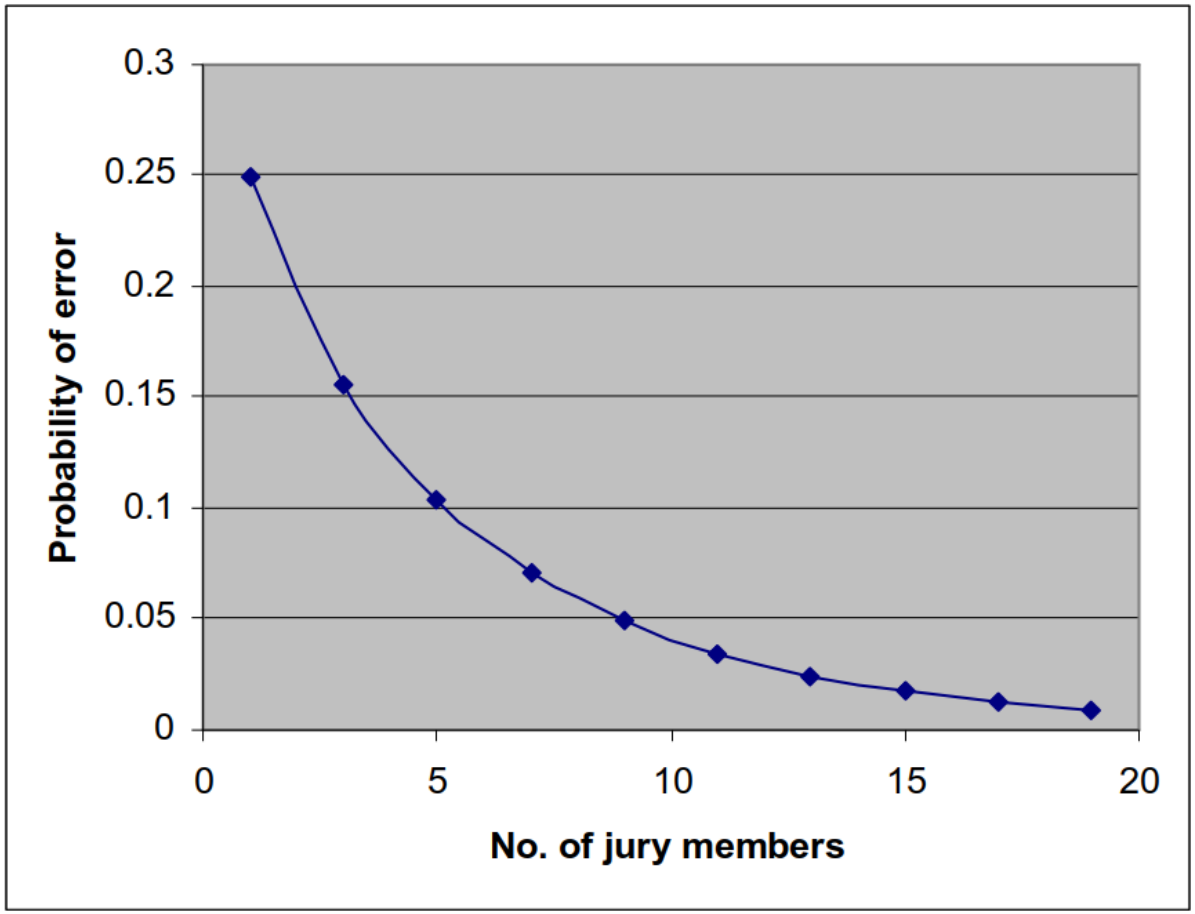
Stacking

Condorcet's jury theorem

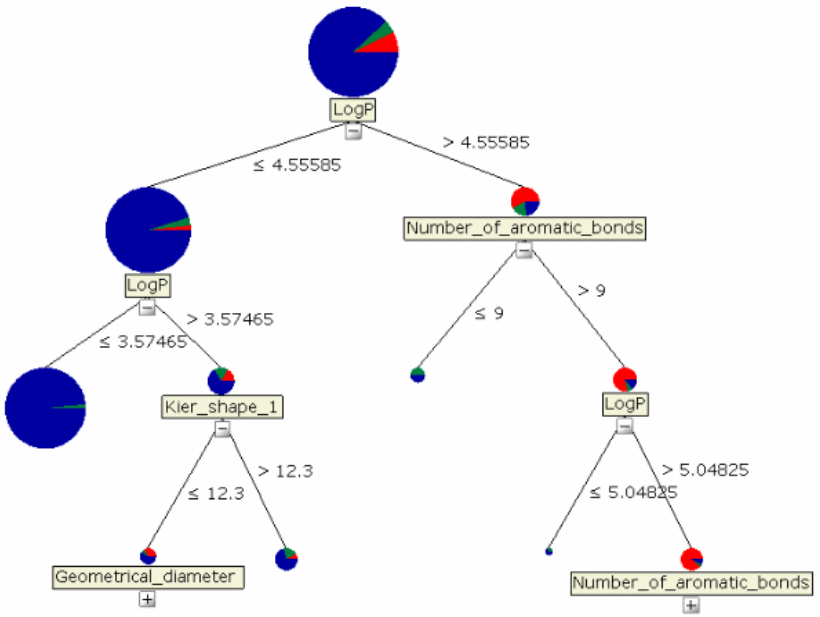
If each member of a jury is more likely to be right than wrong, then the majority of the jury, too, is more likely to be right than wrong; and the probability that the right outcome is supported by a majority of the jury is a (swiftly) increasing function of the size of the jury, converging to 1 as the size of the jury tends to infinity.

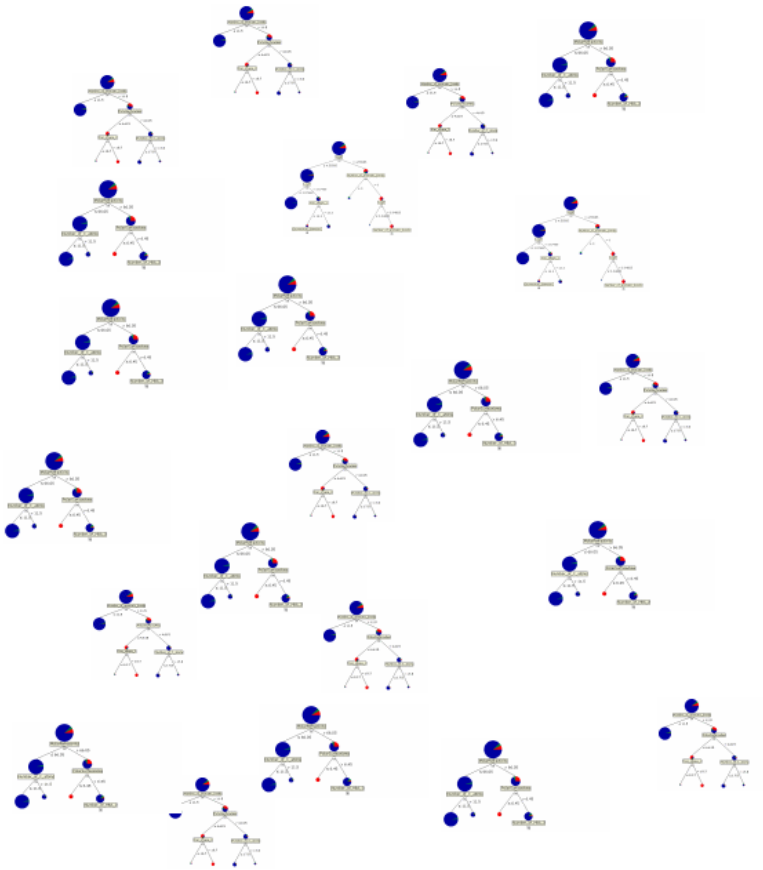
Condorcet, 1785

Condorcet's jury theorem (example)



Single trees vs. forests





Bagging

- A bootstrap sample B of a set of instances I is created by randomly selecting $n = |I|$ instances from I *with replacement*
- The probability of an instance in I appearing in B is

$$1 - (1 - \frac{1}{n})^n = 1 - \frac{1}{e} \approx 0.632$$

- Instances in $I \setminus B$ are said to be *out-of-bag*

L. Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140

Bootstrap sample (example)

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e1	yes	no	no	yes	some	yes
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e5	yes	no	yes	no	none	no
e6	no	yes	no	yes	some	yes

Bootstrap sample (example)

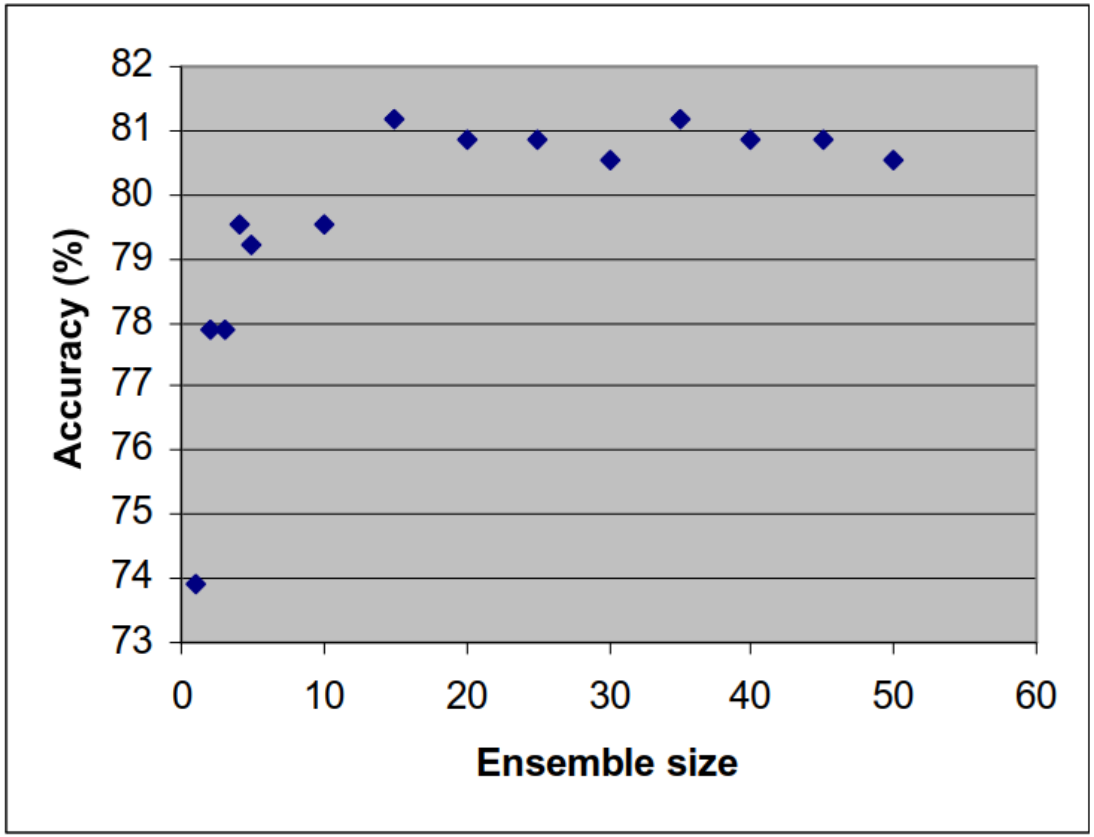
Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	some	yes

Bagging

```
Input: instances I, learner L, iterations m
Output: a combined model M

   for i = 1 to m:
        B = bootstrap sample of I
        M_i = L(B)
    M = average({M_1, ..., M_m})
```

For classification, average can be majority vote (the mode) or the mean of the class probability distributions, while for regression it could be the mean of the predicted values.

Bagging (example)



- Heart-disease dataset from UCI repository
- Stratified 10-fold cross-validation
- Bagged decision trees

Bagging in practice

- Any type of base learner may be used, but better results can often be obtained with more brittle methods, e.g., decision trees, neural networks
- Works in the same way for classification and regression
- For decision trees, pruning often has a detrimental effect, i.e., the more variance of each individual model the better
- Predictive performance can be measured without a separate validation/test set; each training instance may be predicted using ensemble members for which the instance is out-of-bag. This leaves more instances for model generation and all instances will be used for testing.

Randomization

- Repeated application of a base learner that contains some stochastic element (or that can be adjusted to contain stochastic elements) such as
  - the setting of initial weights in a neural network
  - sampling of grow and prune data
  - choice of attribute to split on in decision trees
- Randomization may be combined with other techniques

Random forests

Random forests (Breiman 2001) are generated by combining two techniques:

- bagging (Breiman 1996)
- the random subspace method (Ho 1998)
  - consider only a random sample of the features; either one sample for each tree or one sample for each split. The latter is employed in random forests.

L. Breiman. 2001. Random forests. *Machine Learning*, 45(1):532

T. K. Ho. 1998. The random subspace method for constructing decision forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832-844

Random Forest (example)

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	some	yes

e2 e2 e3 e4 e4 e6

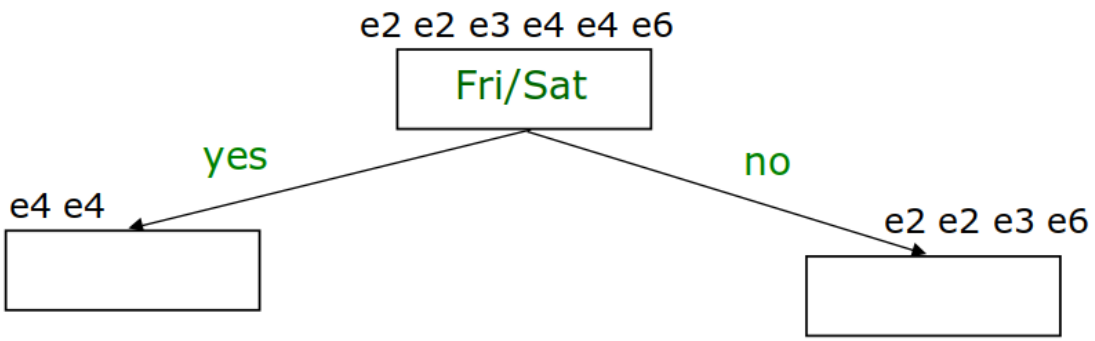
Random Forest (example)

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	some	yes

e2 e2 e3 e4 e4 e6

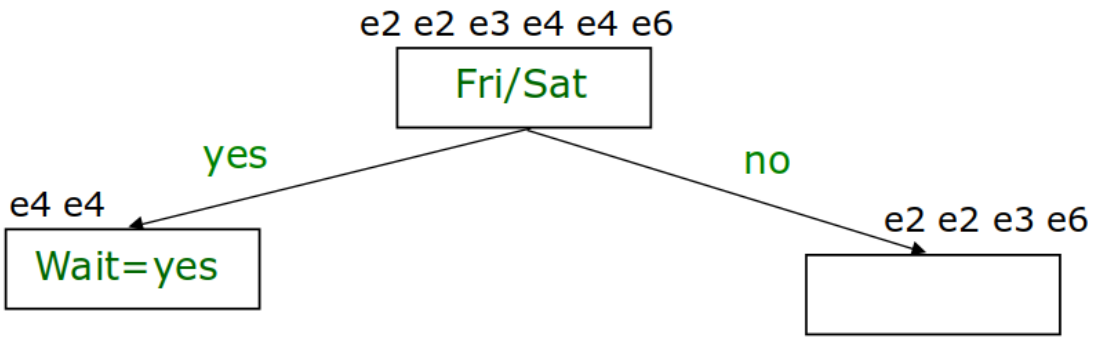
Random Forest (example)

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	some	yes



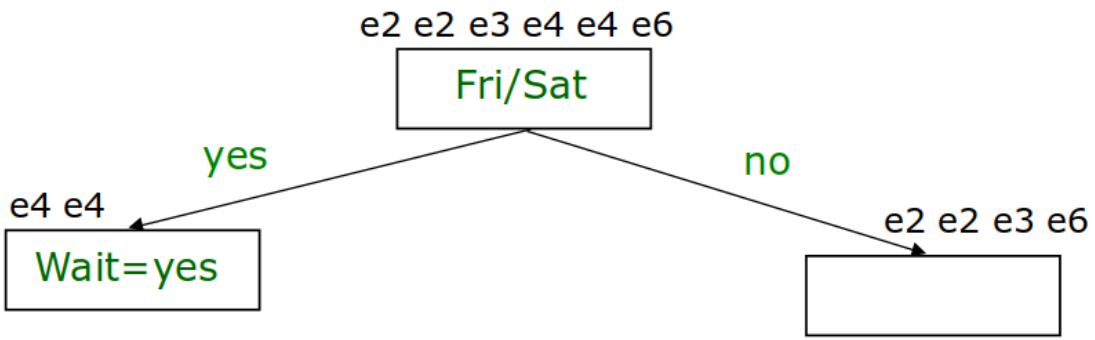
Random Forest (example)

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	some	yes



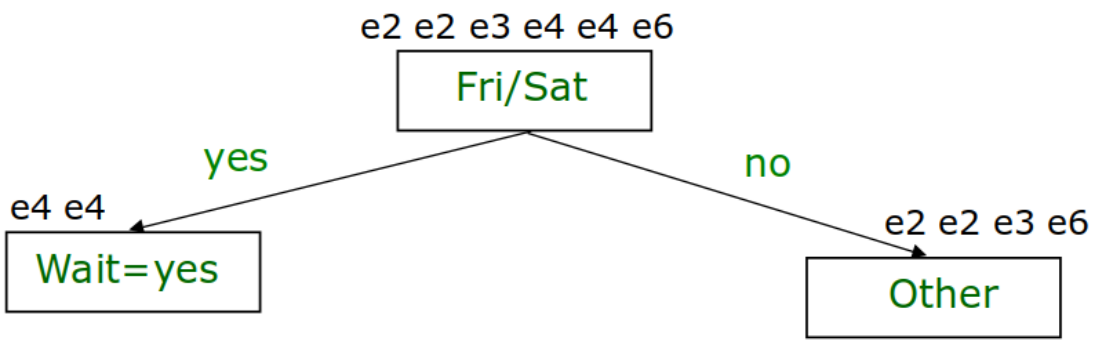
Random Forest (example)

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	some	yes



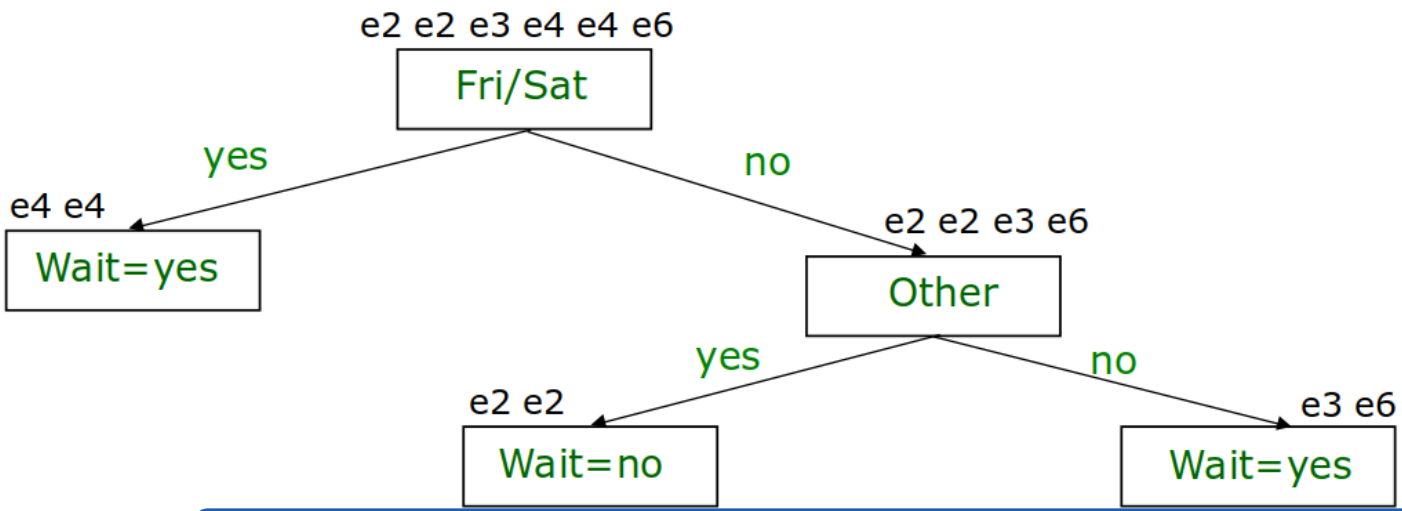
Random Forest (example)

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	some	yes



Random Forest (example)

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	some	yes



Random forests = robust performance

In (Fernandez-Delgado et al 2014), an empirical investigation was presented using:

- 179 classifiers from 17 families, incl. all standard approaches
- 121 datasets, incl. all of UCI except the largest ones
- The random forest versions ranked the highest and they obtained near to the best accuracy for almost all the data sets

It was concluded that:

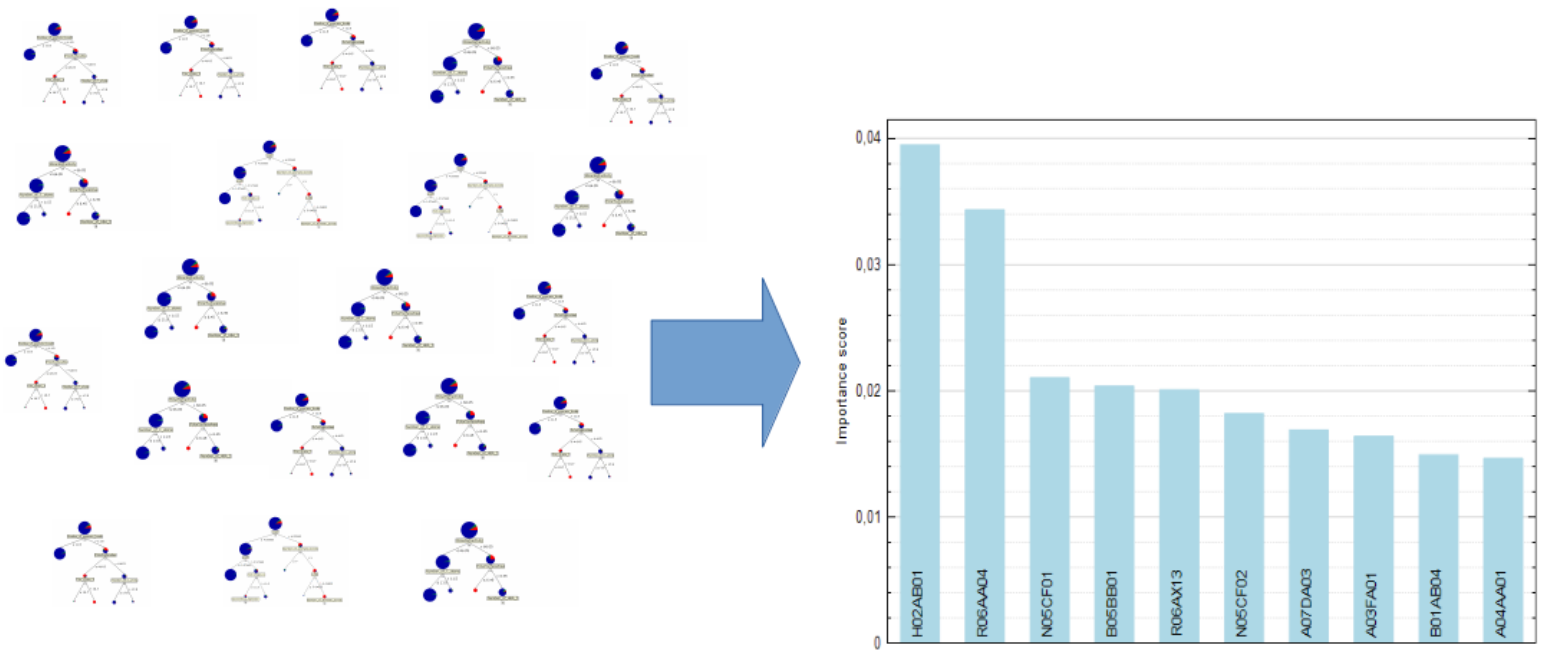
> *The classifiers most likely to be the best are the random forest (RF) versions*

M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, 15(1), pp. 3133-3181, 2014.

Interpretability

- Can we interpret the model/understand the predictions?
  - To be discussed ...

Variable Importance



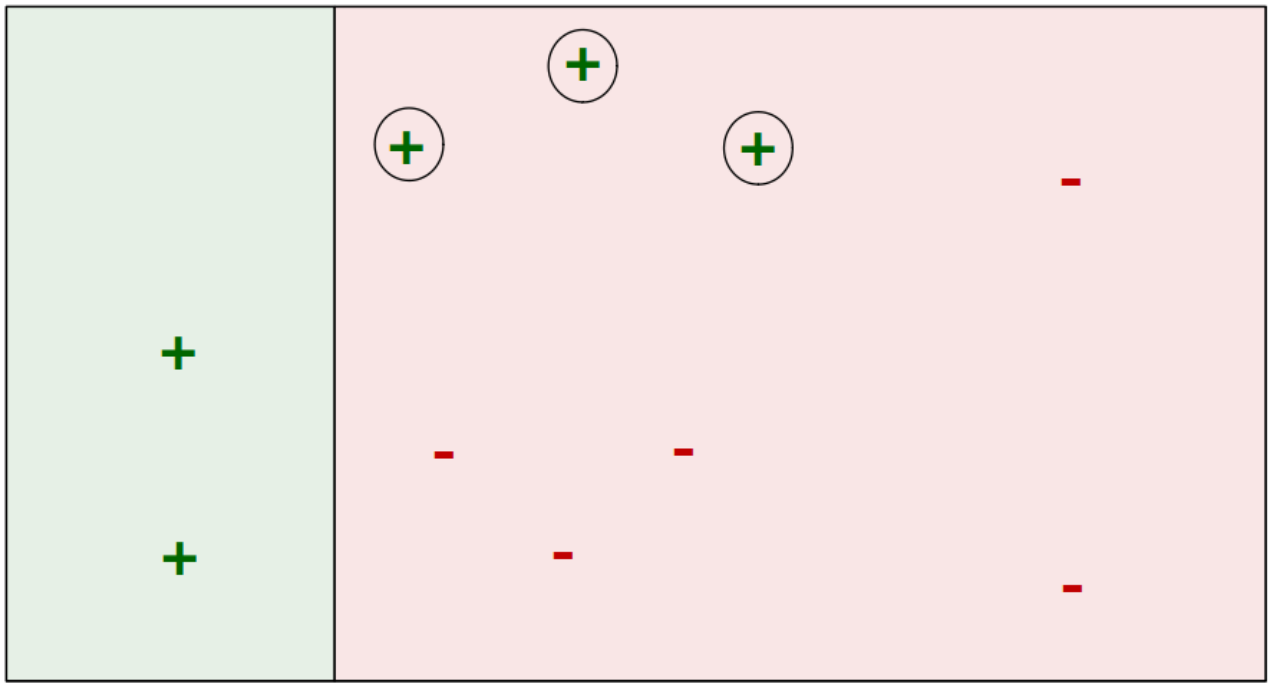
- Breiman (2001) proposed an approach to estimating variable importance by measuring the relative performance degradation (on OOB predictions) when permuting the values of each feature in turn.

Boosting: AdaBoost[1]

```
Input: instances {Z_1,..., Z_n}, learner L, iterations m
Output: a set of model-weight pairs M

w_1, ..., w_n = 1
M = {}
for i = 1 to m:
   M_i = L({(Z_1,w_1), ...,(Z_n,w_n)})
   Err = (w_1*err(M_i,z_1)+...+w_m*err(M_i,z_n))/
         (w_1+...+w_n)
   if Err = 0 or Err > 0.5 then break
   for j = 1 to n:
      if err(M_i,Z_j) = 0 then w_j = w_j*Err/(1-Err)
   M = M + {(M_i,-log Err/(1-Err))}
```

[1] Y. Freund and R. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* 55(1): 119-139

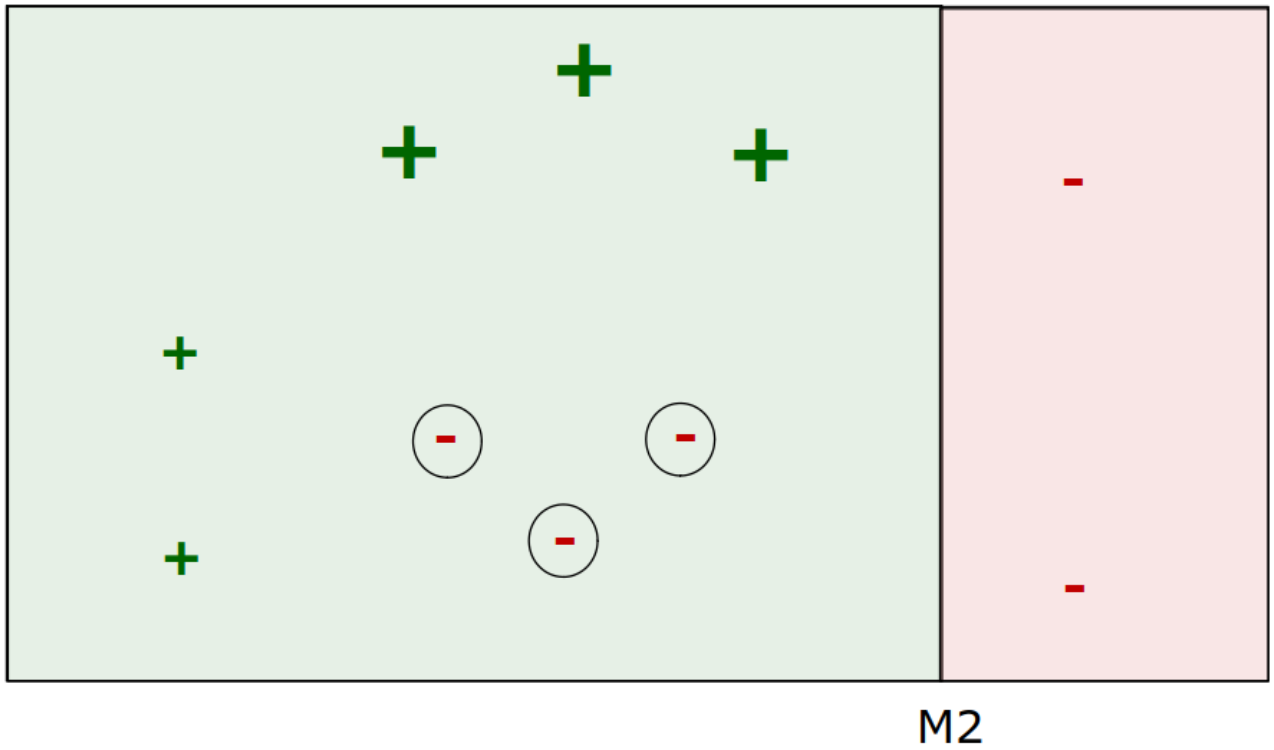
AdaBoost (example)



M1

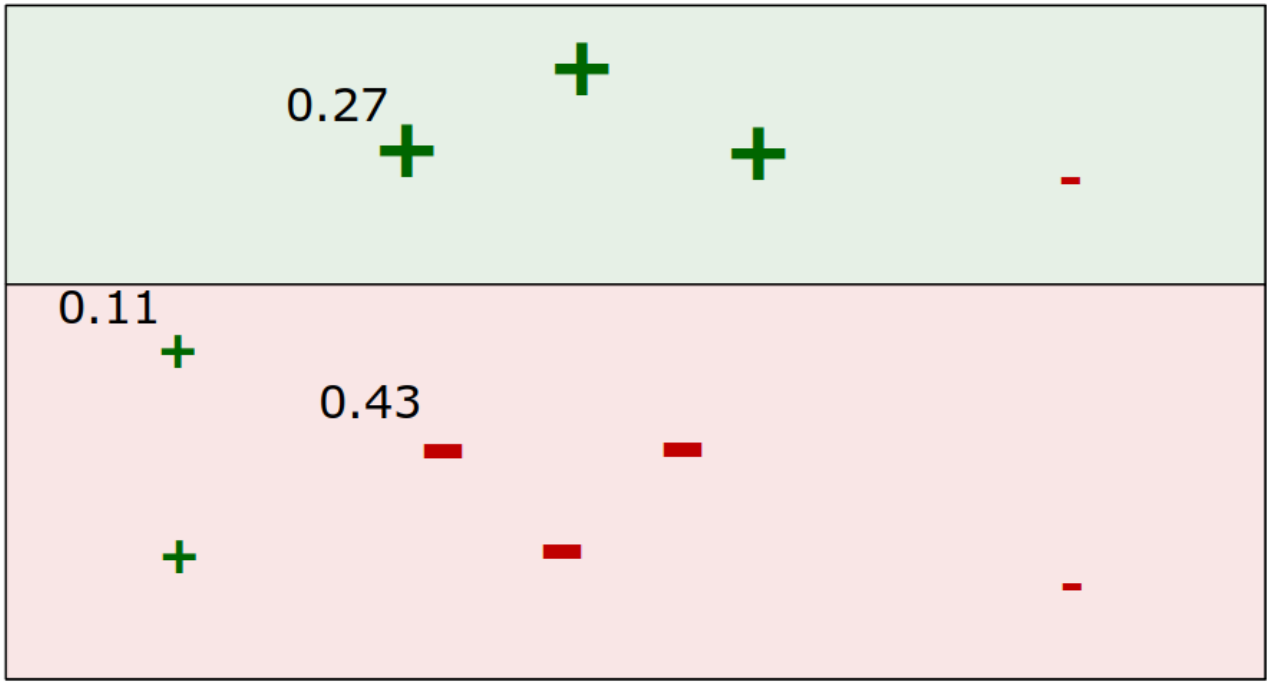
$$Err = 0.3 \qquad \frac{Err}{1 - Err} \approx 0.43$$

AdaBoost (example, cont.)



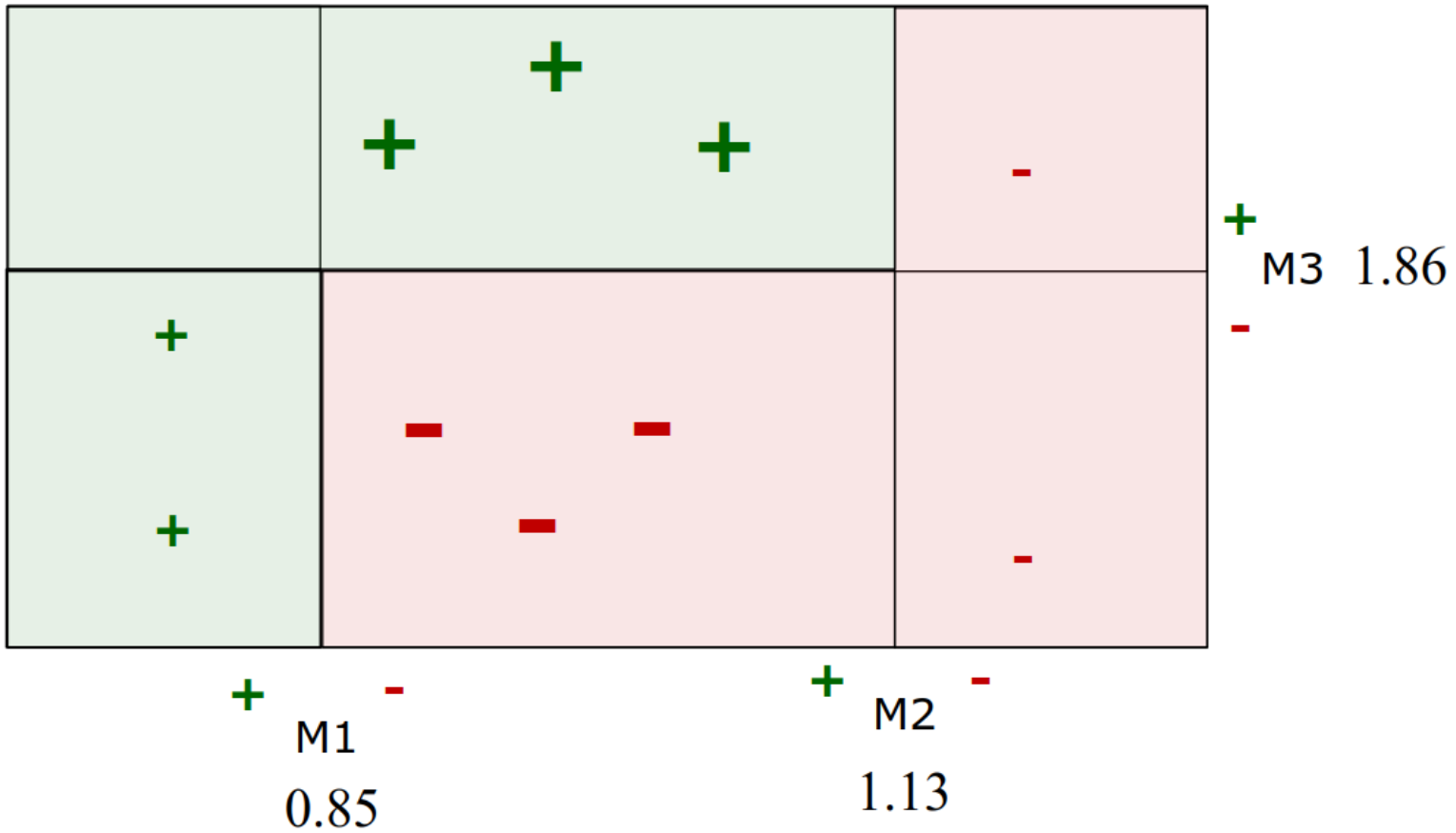
$$Err = 0.21 \qquad \frac{Err}{1 - Err} \approx 0.27$$

AdaBoost (example, cont.)



$$Err \approx 0.13 \qquad \frac{Err}{1 - Err} \approx 0.16$$

AdaBoost (example, cont.)



Gradient boosting machine (regression)[1]

```
Input: instances {(X_1,y_1), ..., (X_n,y_n)},
       learner L, iterations m
Output: a sequence of models M_0, ..., M_m

M_0 = (y_1 + ... + y_n)/n
for i = 1 to m:
   for j = 1 to n:
      y_j =  y_j-M_i-1(X_j)
   M_i = L({(X_1,y_1), ..., (X_n,y_n)})
```

[1] J. Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp.1189-1232

Gradient boosting machine (example)

```
Input: {(0, y_1=1), (1, y_2=1), (2, y_3=2),
        (3, y_4=2), (4, y_5=3), (5, y_6=3)},
       L = decision stump, m = 3

M_0 = (y_1 + ... + y_6)/6 = 2

Iteration 1:      Iteration 2:          Iteration 3:
y_1 -= 2 = -1     y_1 -= -0.5 = -0.5    y_1 -= -0.5 = 0
y_2 -= 2 = -1     y_2 -= -0.5 = -0.5    y_2 -= -0.5 = 0
y_3 -= 2 = 0      y_3 -= -0.5 = 0.5     y_3 -= 0.25 = 0.25
y_4 -= 2 = 0      y_4 -= -0.5 = 0.5     y_4 -= 0.25 = 0.25
y_5 -= 2 = 1      y_5 -= 1 = 0          y_5 -= 0.25 = -0.25
y_6 -= 2 = 1      y_6 -= 1 = 0          y_6 -= 0.25 = -0.25

M_1: 1 if x > 3   M_2: -0.5 if x < 2    M_3: -0.25 if x > 3
     else -0.5          else 0.25             else 0.125
```

Boosting in practice

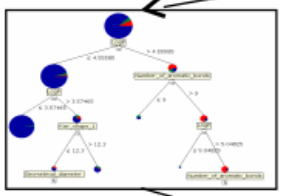
- For multi-class problems, it may be difficult for AdaBoost to reduce the error below 50% with weak base learners, e.g., decision stumps
  - More powerful base learners may be employed
  - The problem may be transformed into multiple binary classification problems
  - Specific multi-class versions of AdaBoost have been developed
- Various loss functions may be used for the Gradient boosting machine (GBM), including log likelihood for classification
- GBM requires careful tuning, in particular of tree depth and *learning rate*
- XGBoost, well-known for winning many competitions, see e.g., `www.kaggle.com`, implements GBM
- Boosting can be sensitive to label noise, e.g., erroneously labeled training examples

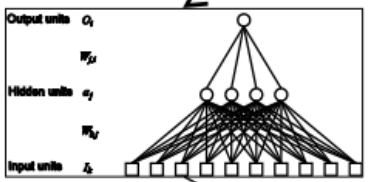
Stacking

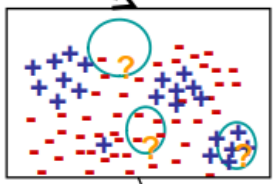
```
Input: instances I, learner L, learners L_1, ..., L_m
Output: a model M, a set of models M_1, ..., M_m

divide I into a training and test set

for each learner L_i:
   generate a model M_i from the training set
   and apply it to the test set

I' = {(y_1,x_11,...,x_1m), ..., (y_n,x_n1,...,x_nm)},
     where y_i is the true label for the ith
     test instance, and x_ij is the prediction
     of model M_j for that instance

M = L(I')
```

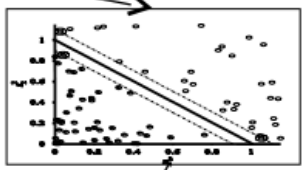
Stacking (example)

Name	Solu-bility	Frac. of rotatable bonds	Geom diam.	Log P	Zagreb group index	Topol diam.	No. C atoms	No. heavy bonds
methylpentane	good	0.40	3.46	2.44	19	4	6	5
methylcyclohexene	good	0	3.00	2.51	31	4	7	7
nonene	med.	0.75	6.93	3.53	28	8	9	8
hexadiene	good	0.60	4.36	2.14	16	5	6	5
butadiene	good	0.33	2.65	1.36	8	3	4	3
naphthalene	good	0	3.61	2.84	57	5	10	11
acenaphthylene	good	0	3.58	3.32	83	5	12	14
pyrene	poor	0	5.00	4.58	117	7	16	19
dimethylanthracene	poor	0	5.29	4.61	108	7	16	18
hexahydropyrene	med.	0	5.00	3.82	117	7	16	19
triphenylene	poor	0	5.00	5.15	126	7	18	21
benzo(e)pyrene	poor	0	5.29	5.64	152	7	20	24









Class	M1	M2	M3	M4
good	good	good	good	med.
good	med.	good	good	good
med.	med.	good	med.	poor
poor	good	poor	good	good

IF M2 = good & M3 = good THEN Class = good
IF M2 = poor THEN Class = poor
...

Stacking in practice

- Linear models have been shown to be effective when learning the combination function
- Predictive performance can be improved by combining class probability estimates rather than class labels generated by the base models

Summary

- We have considered various ways of combining models; bagging, randomization, boosting and stacking (as well as some combinations of these)
- Compared to the base learning algorithms, the predictive performance can be substantially improved; sometimes leading to state-of-the-art performance
- The increased predictive power comes, however, at the cost of reduced interpretability
- The combination strategies may not necessarily lead to increased computational cost; and this cost can often be further reduced by parallelization