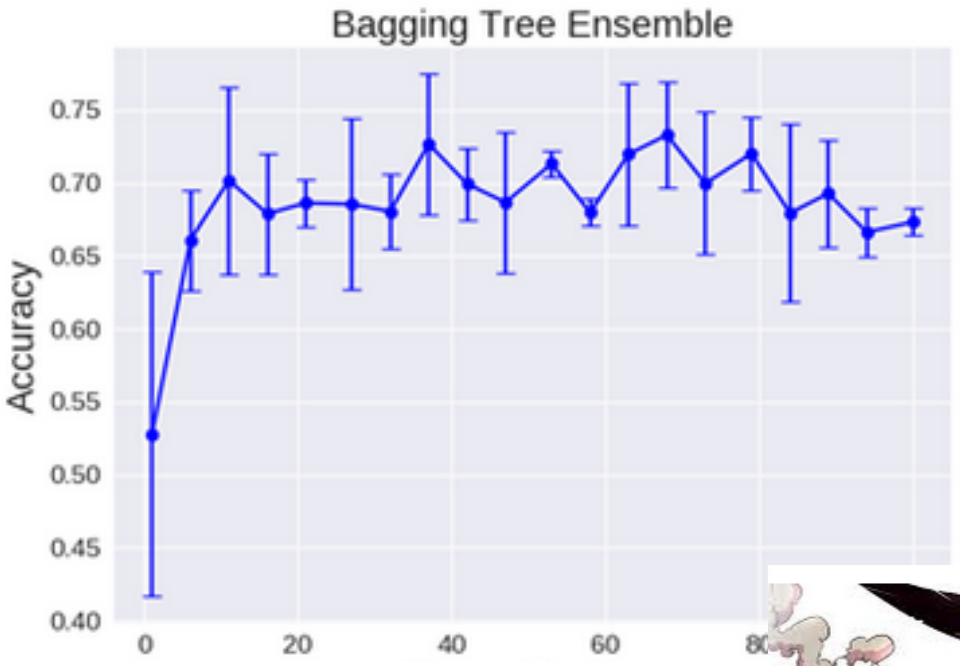
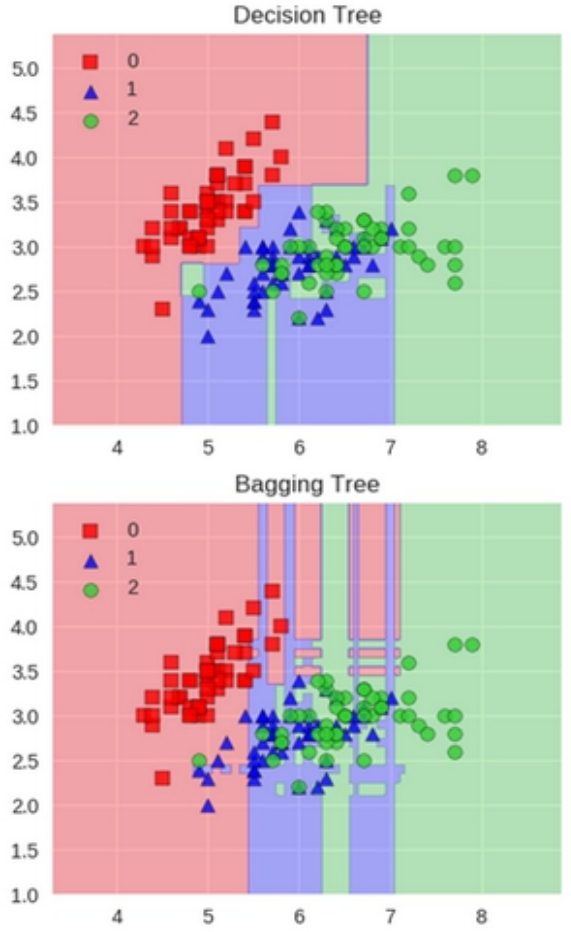
Ensemble Learning

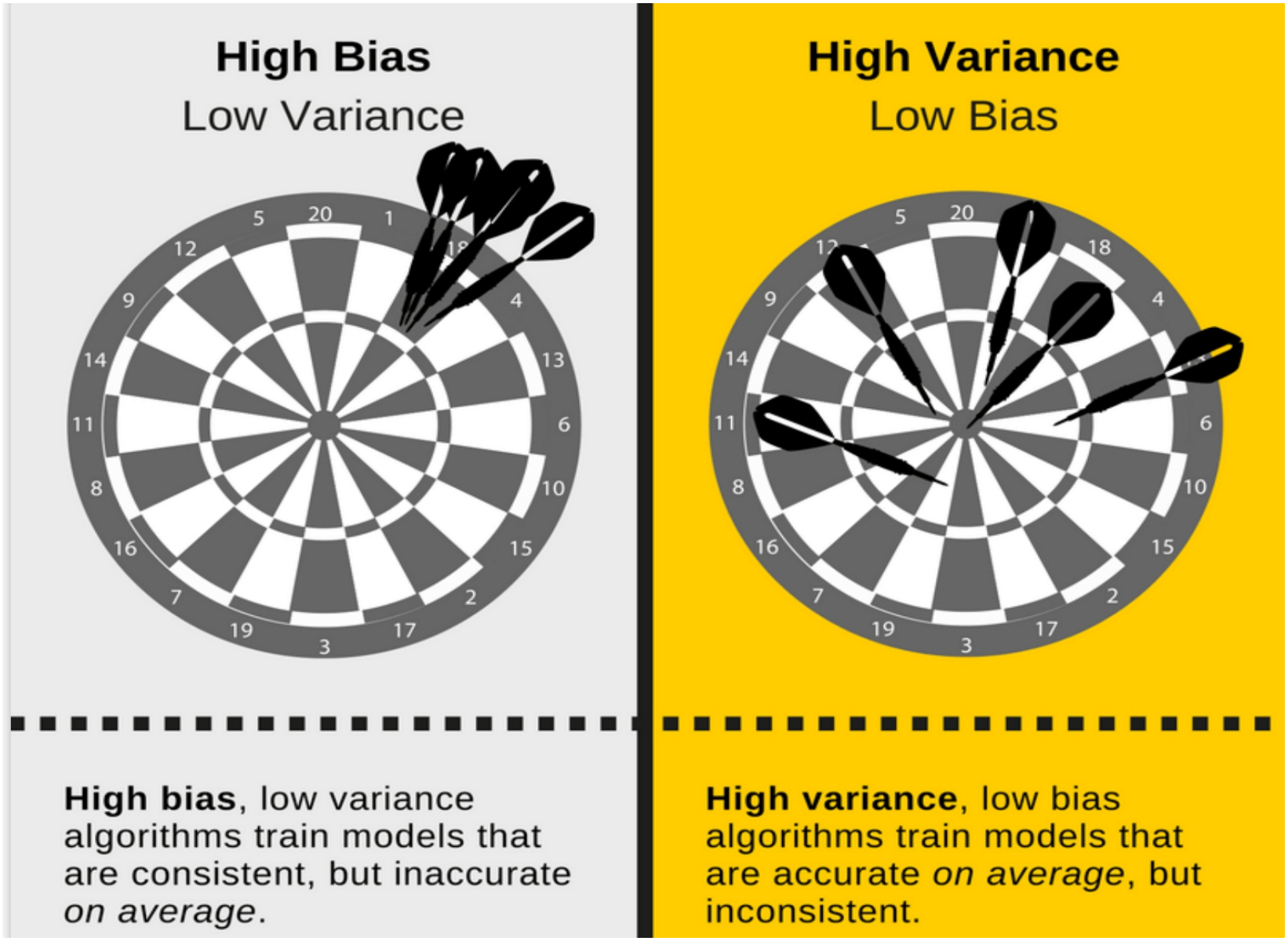
Ronnie Alves (alvesrco@gmail.com)

https://sites.google.com/site/alvesrco/





Why Ensemble Based Systems?



https://elitedatascience.com/bias-variance-tradeoff

Netflix Prize

http://www.netflixprize.com/index

Since October 2006

Supervised learning task

Training data is a set of users and ratings (1,2,3,4,5 stars) those users have given to movies.

Construct a classifier that given a user and an unrated movie, correctly classifies that movie as either 1, 2, 3, 4, or 5 stars

$1 million prize for a 10% improvement over Netflix's current movie recommender/classifier (MSE = 0.9514)

"Our final solution (**RMSE=0.8712**) consists of **blending 107 individual** results. "

Netflix Prize

Leaderboard

Display top 40 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	--	--	--
Grand Prize - RMSE <= 0.8563				
1	BellKor in BigChaos	0.8613	9.47	2008-11-30 19:01:34
Progress Prize - RMSE <= 0.8625				
2	BigChaos	0.8626	9.33	2008-12-01 22:24:12
3	BellKor	0.8631	9.28	2008-11-30 18:44:54
4	PragmaticTheory	0.8638	9.21	2008-11-28 11:46:23
5	Gravity	0.8654	9.04	2008-11-27 21:18:37
6	My Brain and His Chain	0.8668	8.89	2008-09-30 02:19:47
7	Just a guy in a garage	0.8673	8.84	2008-11-26 17:00:27
8	When Gravity and Dinosaurs Unite	0.8675	8.82	2008-10-05 14:16:53
9	Opera Solutions	0.8676	8.81	2008-12-02 22:08:45
10	acmehill	0.8677	8.80	2008-11-26 10:15:28
11	scientist	0.8677	8.80	2008-12-02 01:10:13
12	Ces	0.8711	8.44	2008-08-25 05:00:23
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell				
13	KorBell	0.8712	8.43	2007-10-01 23:25:23
14	basho	0.8714	8.41	2008-05-21 22:06:00

Two heads are better than one

We consider decisions of **multiple experts** in our daily lives. Why not follow the same strategy in **automated decision making**?

Statistical intuition:

Averaging measurements can lead to a more stable and reliable estimate because we **reduce the influence of random fluctuations** in single models.

The key question:

How to achieve **diversity** between these different models.

Why Ensemble Based Systems?

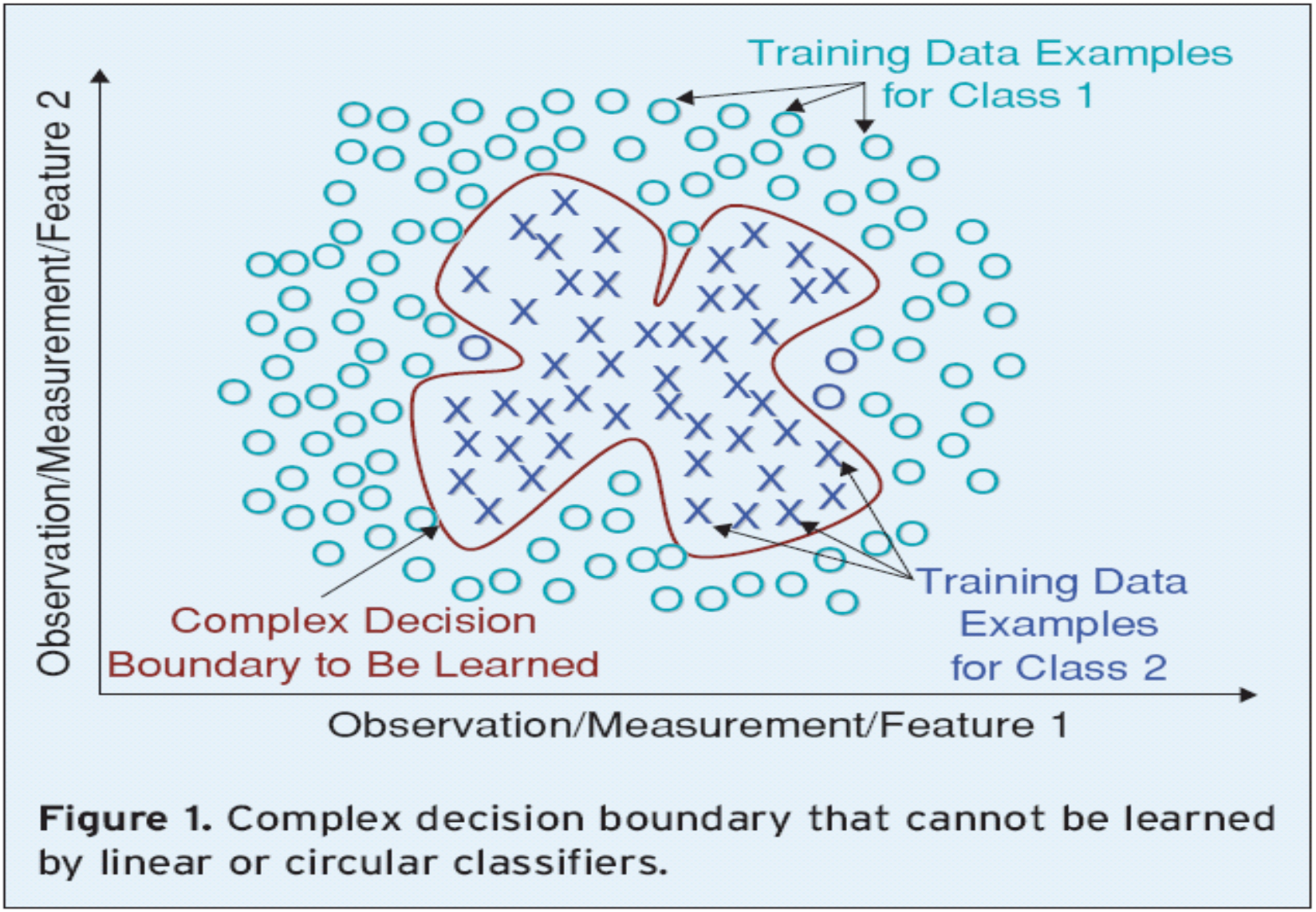


Figure 1. Complex decision boundary that cannot be learned by linear or circular classifiers.

Why Ensemble Based Systems?

Divide data space into smaller & **easier-to-learn partitions**; each **classifier learns** only one of the simpler partitions

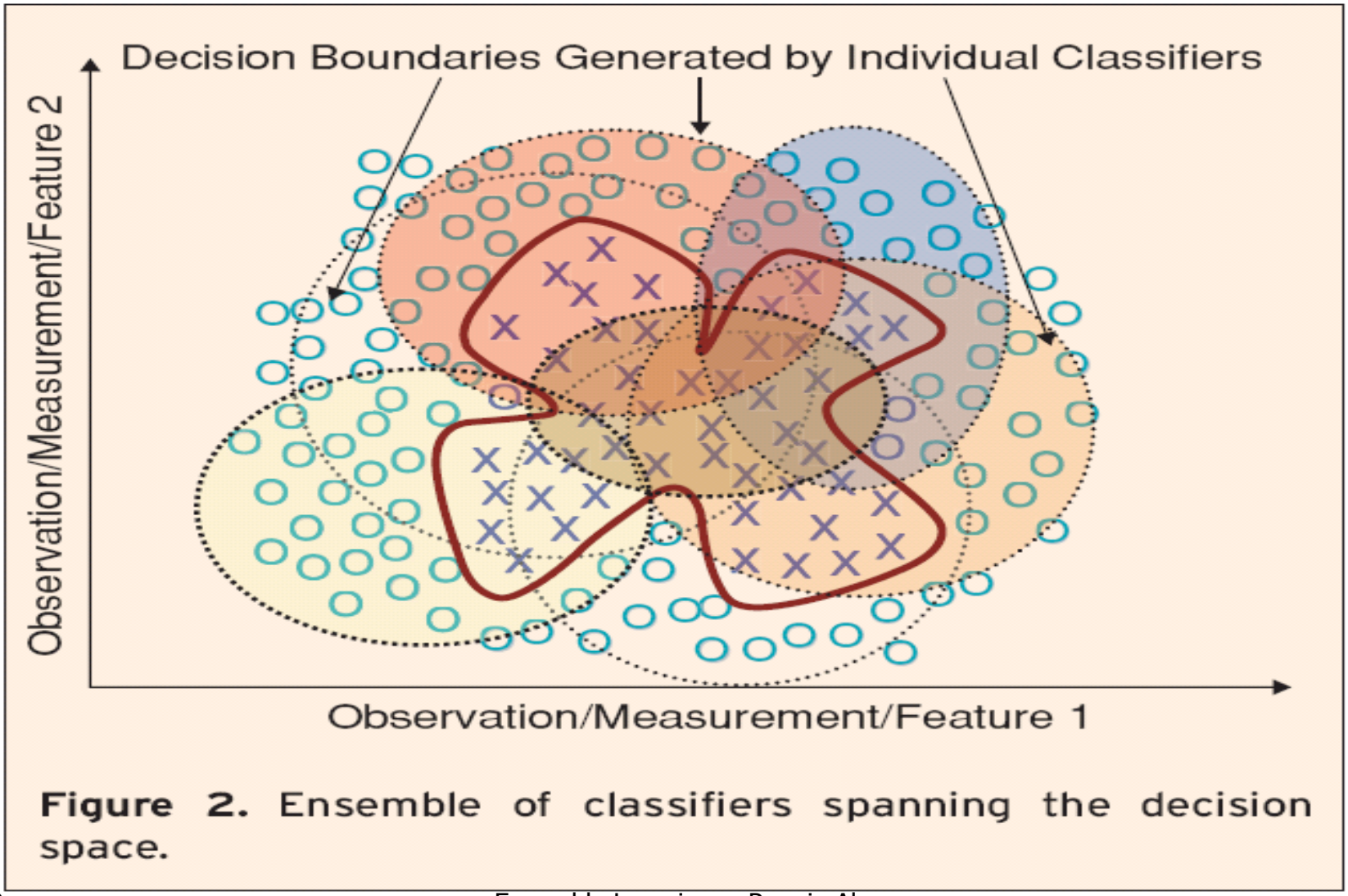


Figure 2. Ensemble of classifiers spanning the decision space.

Diversity of Ensemble

- Intuition: if each **classifier makes different errors**, then their strategic combination can **reduce the total error**!
- Need base classifiers whose **decision boundaries** are **adequately different** from those of others
  - Such a set of classifiers is said to be ***diverse***
- How to achieve classifier diversity?
  - Use **different training sets** to train individual classifiers
  - How to obtain different training sets?
    - **Resampling** techniques: bootstrapping or bagging, training subsets are drawn randomly, usually with replacement, from the entire training set

Sampling with Replacement

- Random & overlapping training sets to train three classifiers;

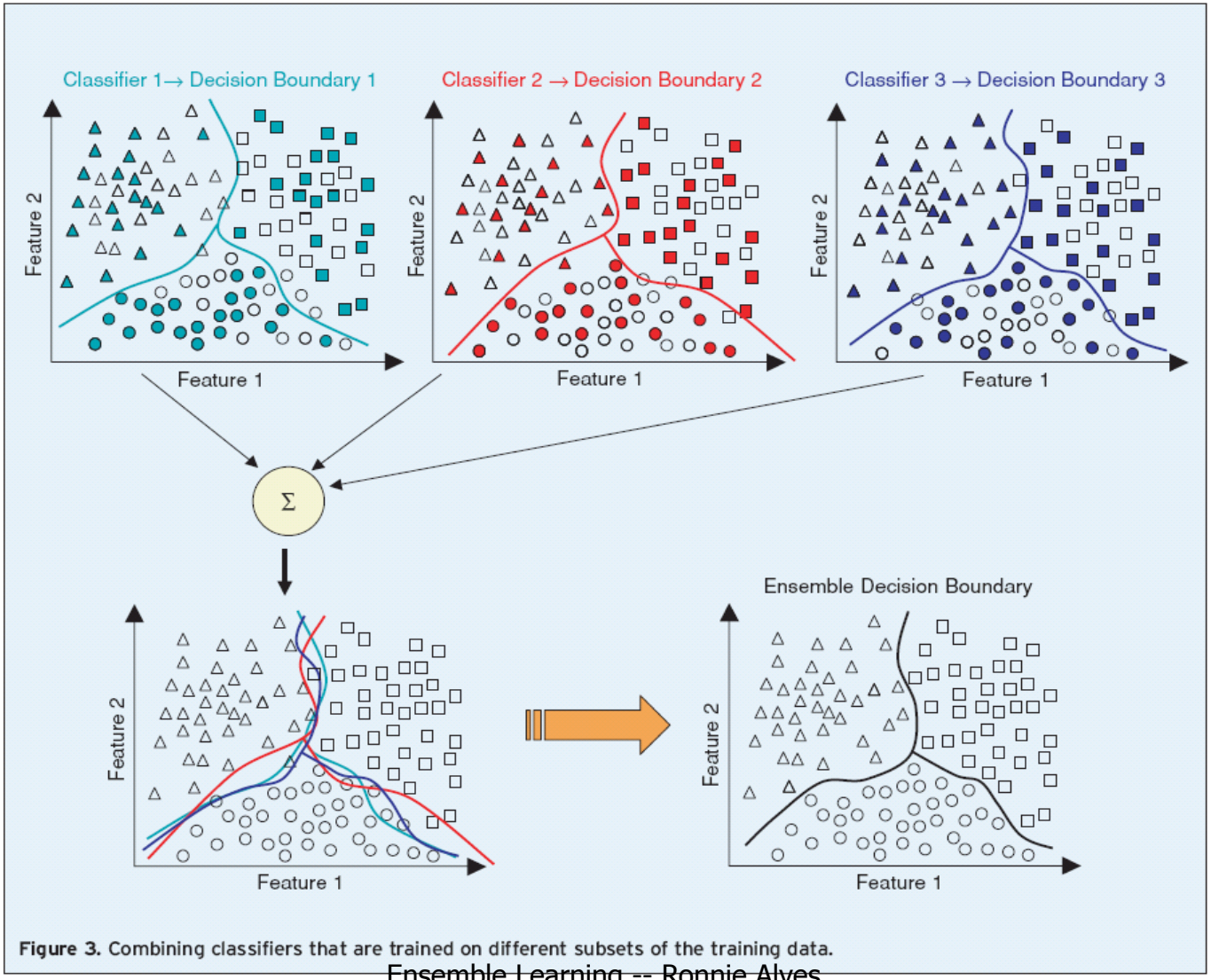


Figure 3. Combining classifiers that are trained on different subsets of the training data.

Sampling without Replacement

- Jackknife or k-fold data split:
  - Entire data is split into k blocks; each **classifier is trained only on different subset** of (k-1) blocks

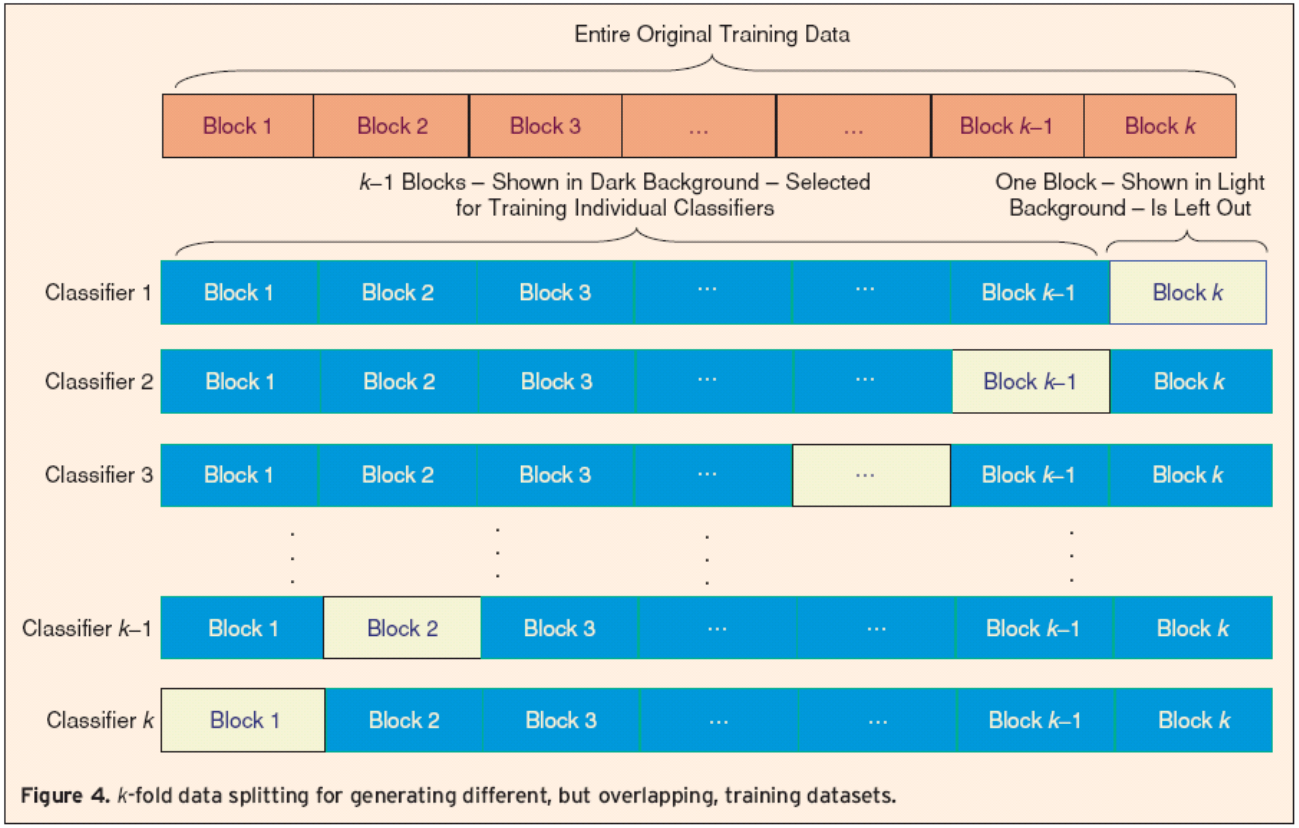


Figure 4. *k*-fold data splitting for generating different, but overlapping, training datasets.

Other Approaches to Achieve Diversity

- Use **different training parameters** for a classifier
  - A series of MLP can be trained using different weight initializations, **number of layers/nodes**, etc.
  - Adjusting these parameters controls the instability of such classifiers (local minima)
  - Similar strategy can be used to generate **different decision trees** for the same problem
- Different types of classifiers (MLPs, decision trees, NN classifiers, SVM) can be combined for added diversity
- Diversity can also be achieved by using **random feature subsets**, called random subspace method

Ensembles improve classifiers's performance

Suppose there are **25 base classifiers**

Each classifier has **error rate**, e = 0.35

Assume **classifiers are independent**

Probability that the ensemble classifier makes a wrong prediction?

Ensembles improve classifiers's performance

Suppose there are 25 base classifiers

Each classifier has error rate, e = 0.35

Assume classifiers are independent

Probability that the ensemble classifier makes a wrong prediction?

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^{i} (1-\varepsilon)^{25-i} = 0.06$$

Bagging

- Bagging, short for **bootstrap aggregating**, is one of the earliest ensemble based algorithms
- It is also one of the most intuitive and simplest to implement, with a surprisingly **good performance**
- Use **bootstrapped replicas** of the training data; large number of (say 200) training subsets are randomly drawn - **with replacement** - from the entire training data
- Each resampled training set is used to **train a different classifier of the same type**
- Individual classifiers are combined by taking a **majority vote** of their decisions
- Bagging is appealing for **small training set**; relatively large portion of the samples is included in each subset

Bagging

Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

Build classifier on each **bootstrap sample**

Each sample has probability $\mathbf{(1 - 1/n)^n}$ of being selected

On average, a bootstrap sample D(i) contains approximately **63% of the original training** data

Bagging

Algorithm

Given a standard training set D of size n

For i = 1 .. M

Draw a sample of size $n^* < n$ from D uniformly and with replacement

Learn classifier C_i

Final classifier is a vote of C_1 .. C_M

Increases classifier stability/reduces variance

Variations of Bagging

Random Forests

- so-called because it is constructed from **decision trees**
- A random forest is created from individual decision trees, **whose training parameters vary randomly**
- Such parameters can be **bootstrapped replicas of the training data**, as in bagging
- But they can also be different feature subsets as in **random subspace methods**

How random Forest works?

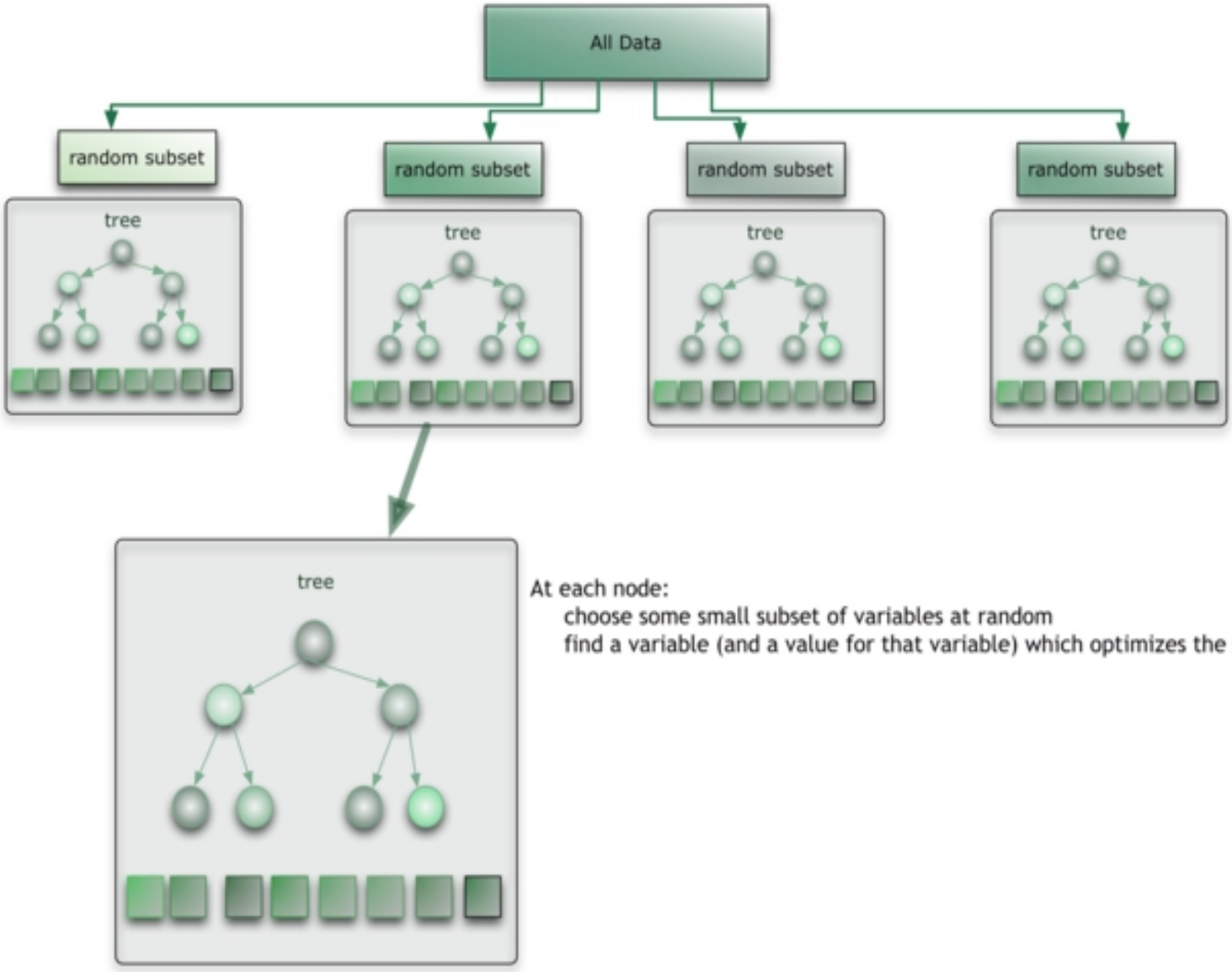
Each tree is grown as follows:

1. **Random Record Selection**: Each tree is trained on roughly 2/3rd of the total training data . Cases are drawn at random with replacement from the original data, this sample will be the training set for growing the tree.
2. **Random variable selection**: Some predictor variables, say m, are selected at random out of all the predictor variables and the **best split on these m is used to split** the node.
3. For each tree, using leftover data, **calculate the misclassification rate – out of bag (OOB)** error rate and aggregate error from all the trees to determine overall the OOB error rate for the classification.

How random Forest works?

4. **Each tree gives a classification** and we say that the tree "votes" for that class. The forest chooses the **classification having the most votes**.

How random Forest works?



Pros and Cons

Then advantages of random forests :

- 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 data sets.
- It can handle thousands of input variables without variable deletion.
- **It gives estimate of what variables are important in the classification.**
- It has an effective method for estimating missing data and maintains when large proportion of the data are missing.
- **It computes proximities between pairs of cases that can be used in clustering, locating outliers.**

Pros and Cons

Disadvantages are:

- **Random forests** have been observed **to over fi**t for some datasets with noisy classification/regression tasks**.**
- For data including categorical variables with **different number of levels**, random forests are **biased** in favor of those attributes with **more labels**.

Parameters

When running random forests there are number of parameters that need to specified. The most common parameters are:

- Input training data including predictor variables.
- The **number of trees** that should be built.
- The **number of predictor variables** to be used to create the binary rule for each split.
- parameters to calculate information related to **error and variable significance.**

Terminologies related to random forest algorithm

Out-of-Bag error rate

- As the forest is built on training data, each tree is **tested on** the **1/3rd of the samples** not used in building that tree. This is the out-of-bag error estimate- an **internal error estimate** of a random forest

Bootstrap sample

- It is a random **with replacement** sampling method.

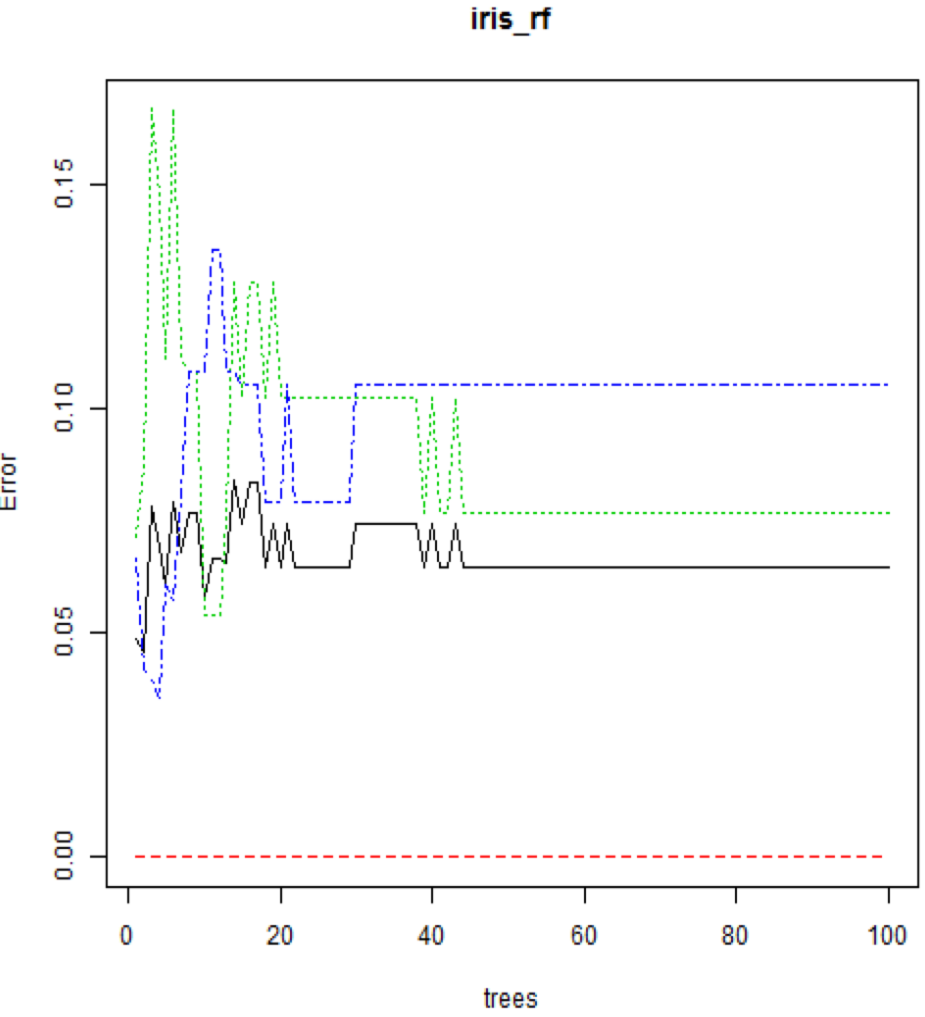
Proximities

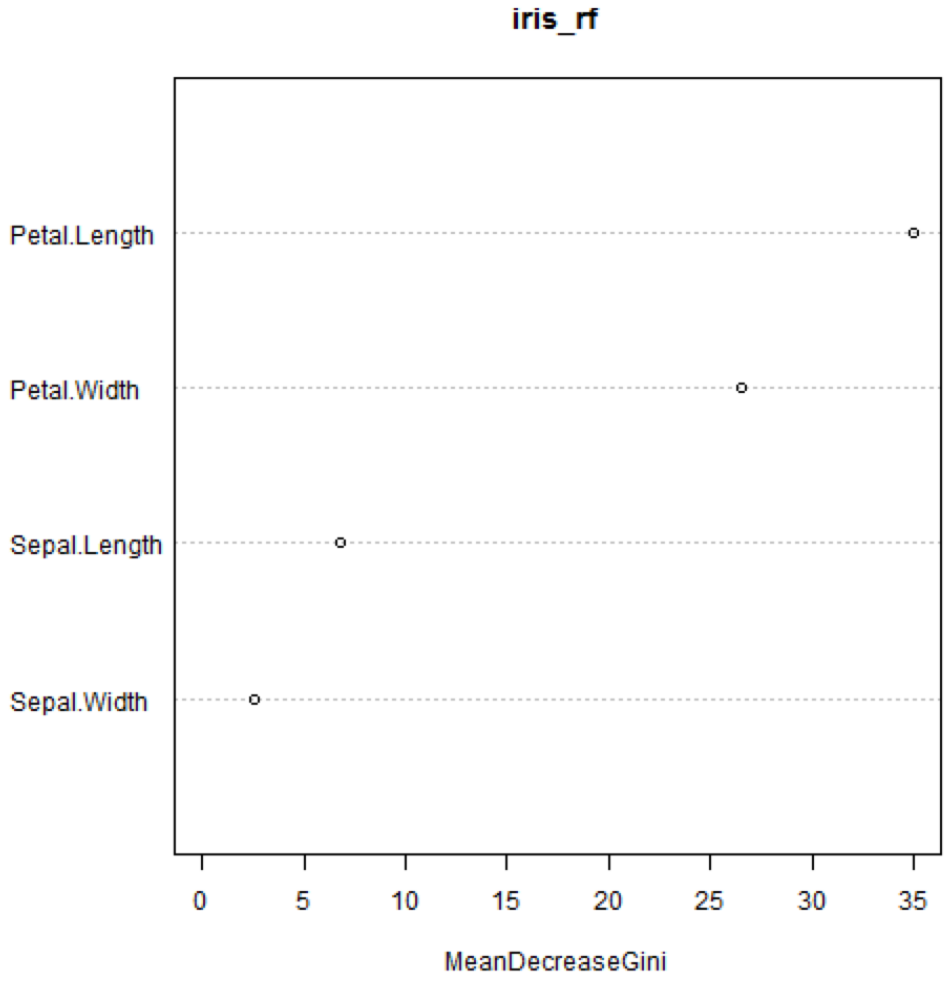
- These are one of the most useful tools in random forests. The proximities originally formed a NxN matrix. After a tree is grown, put all of the data, both training and OOB, down the tree. If **cases** k and n are in the **same terminal node** increase their proximity by one. At the end, **normalize** the proximities by dividing by the **number of trees**.

Variables importance

RF computes two measures of variable importance, one based on a rough-and-ready measure (**Gini for classification**) and the other based on **permutations**.

RF on IRIS





library(randomForest)

iris_rf <- randomForest(Species~.,data=trainData,ntree=100,proximity=TRUE)

Strong and Weak Learners

Strong Learner → Objective of machine learning

Take labeled data for training

Produce a classifier which can be ***arbitrarily accurate***

Weak Learner

Take labeled data for training

Produce a classifier which is more **accurate than random guessing**

Boosting

Weak Learner: only needs to generate a hypothesis with a training accuracy greater than 0.5, i.e., < 50% error over any distribution

Strong learners are very difficult to construct

Constructing weaker Learners is relatively easy

Can a set of weak learners create a single strong learner ?

YES ☺

Boost weak classifiers to a strong learner!

Boosting

- Boost the performance of a **weak learner** to the level of a strong one
- Boosting creates an **ensemble of classifiers** by resampling the data; classifiers combined by **majority voting**
  - resampling is strategically geared to provide the **most informative training data** for each consecutive classifier
- Boosting creates three weak classifiers:
  - First classifier C1 is trained with a random subset of the available training data
  - Training set for second classifier C2 is chosen as the most informative subset, given C1; half of the training data for C2 is correctly classified by C1, other half is misclassified by C1
  - Third classifier C3 is trained on instances on which both C1 & C2 disagree

Boosting

Records that are **wrongly** classified will have their **weights increased**

Records that are classified **correctly** will have their **weights decreased**

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

AdaBoost

- AdaBoost (1997) is a more general version of the boosting algorithm; AdaBoost.M1 can **handle multiclass problems**
- AdaBoost generates a set of hypotheses (classifiers), and combines them through weighted majority voting of the classes predicted by the individual hypotheses
- Hypotheses are generated by training a weak classifier; **samples are drawn from an iteratively updated distribution of the training set**
- This distribution update ensures that instances misclassified by the previous classifier are more likely to be included in the training data of the next classifier
- **Consecutive classifiers are trained on increasingly hard-to-classify samples**

AdaBoost.M1

- AdaBoost algorithm is sequential; classifier (CK-1) is created before classifier CK

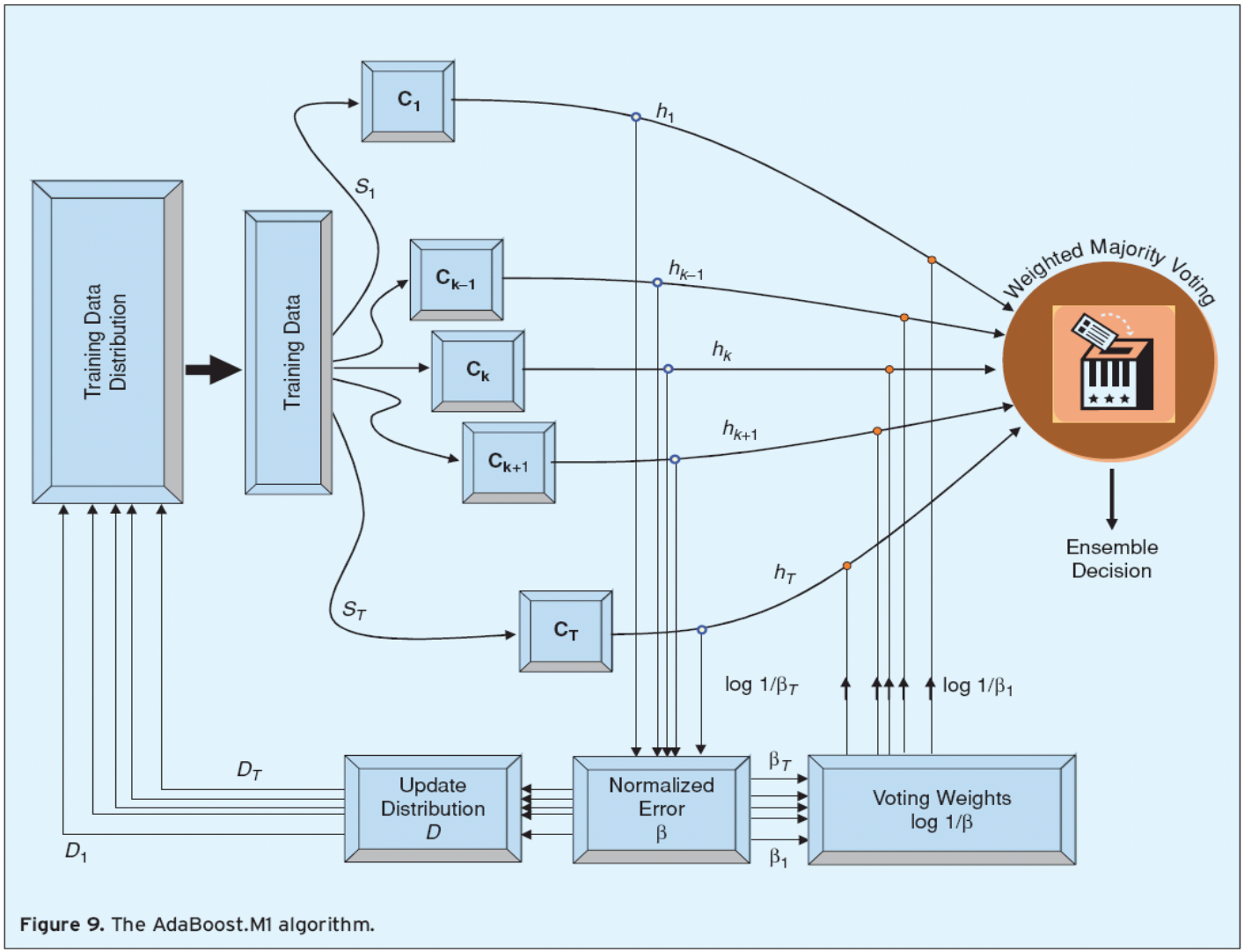


Figure 9. The AdaBoost.M1 algorithm.

Example: AdaBoost

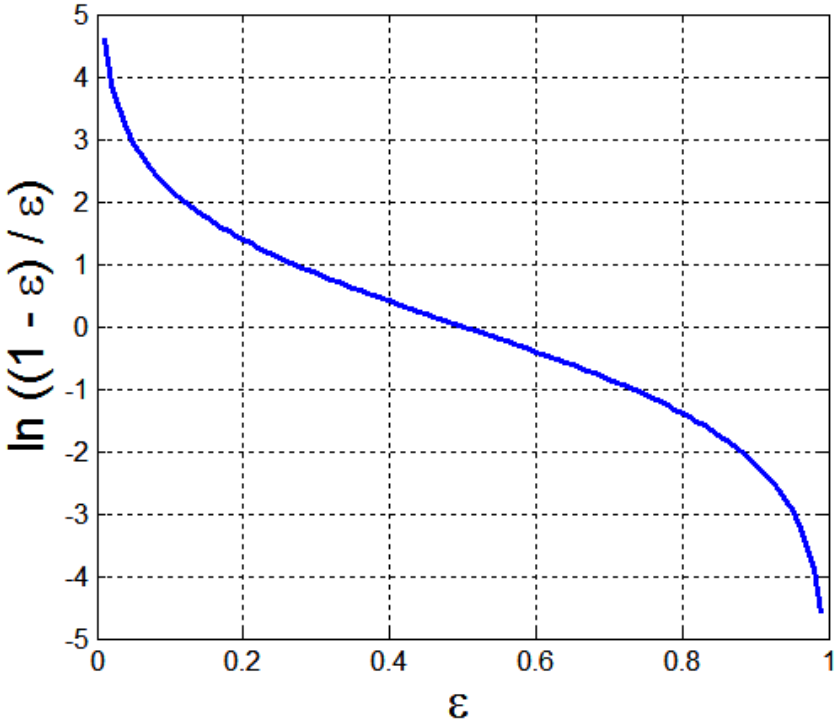
Base classifiers: C_1, C_2, …, C_T

Error rate:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



Example: AdaBoost

Weight update:

$$w_{i^{(j+1)}} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

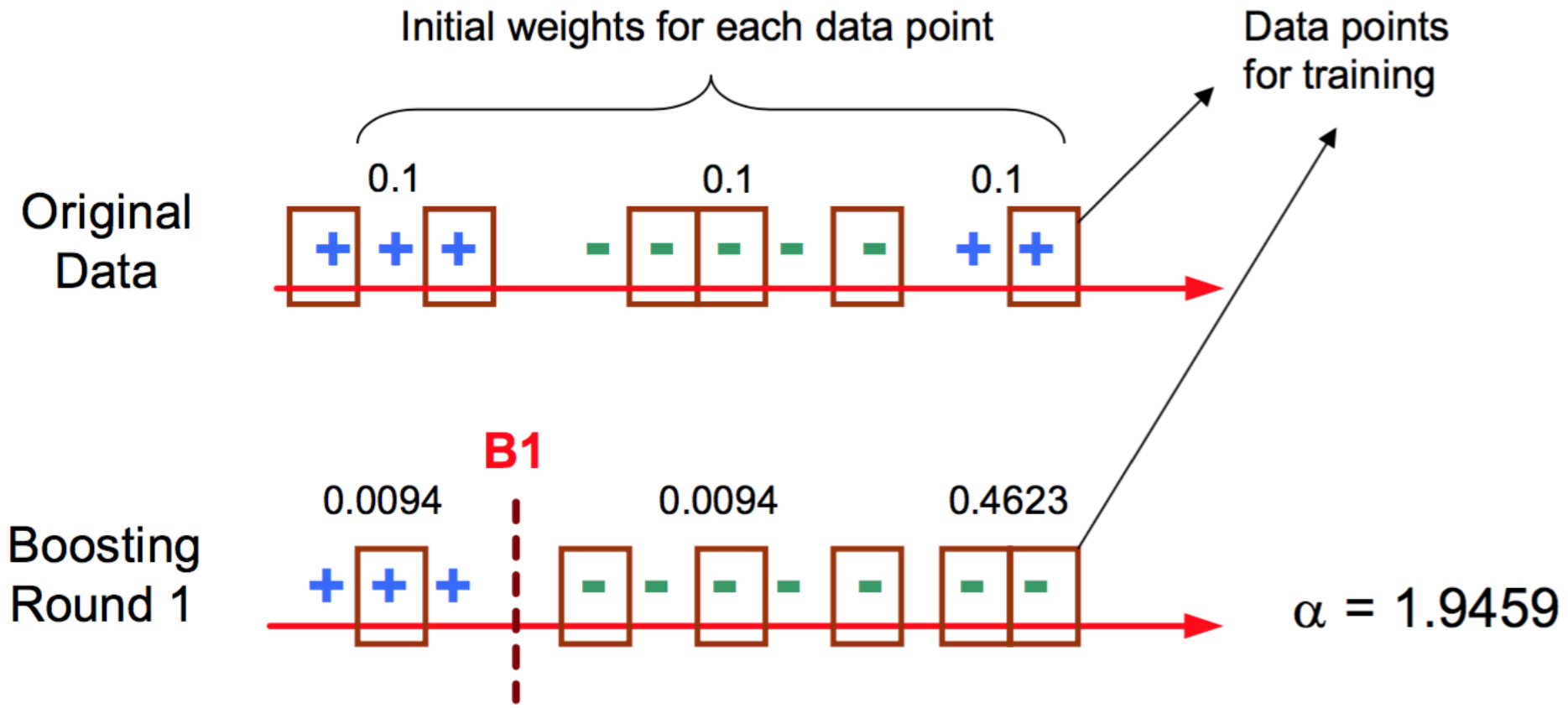
where Z_j is the normalization factor

If any intermediate rounds produce error rate higher than 50%, the **weights are reverted back** to 1/n and the resampling procedure is repeated

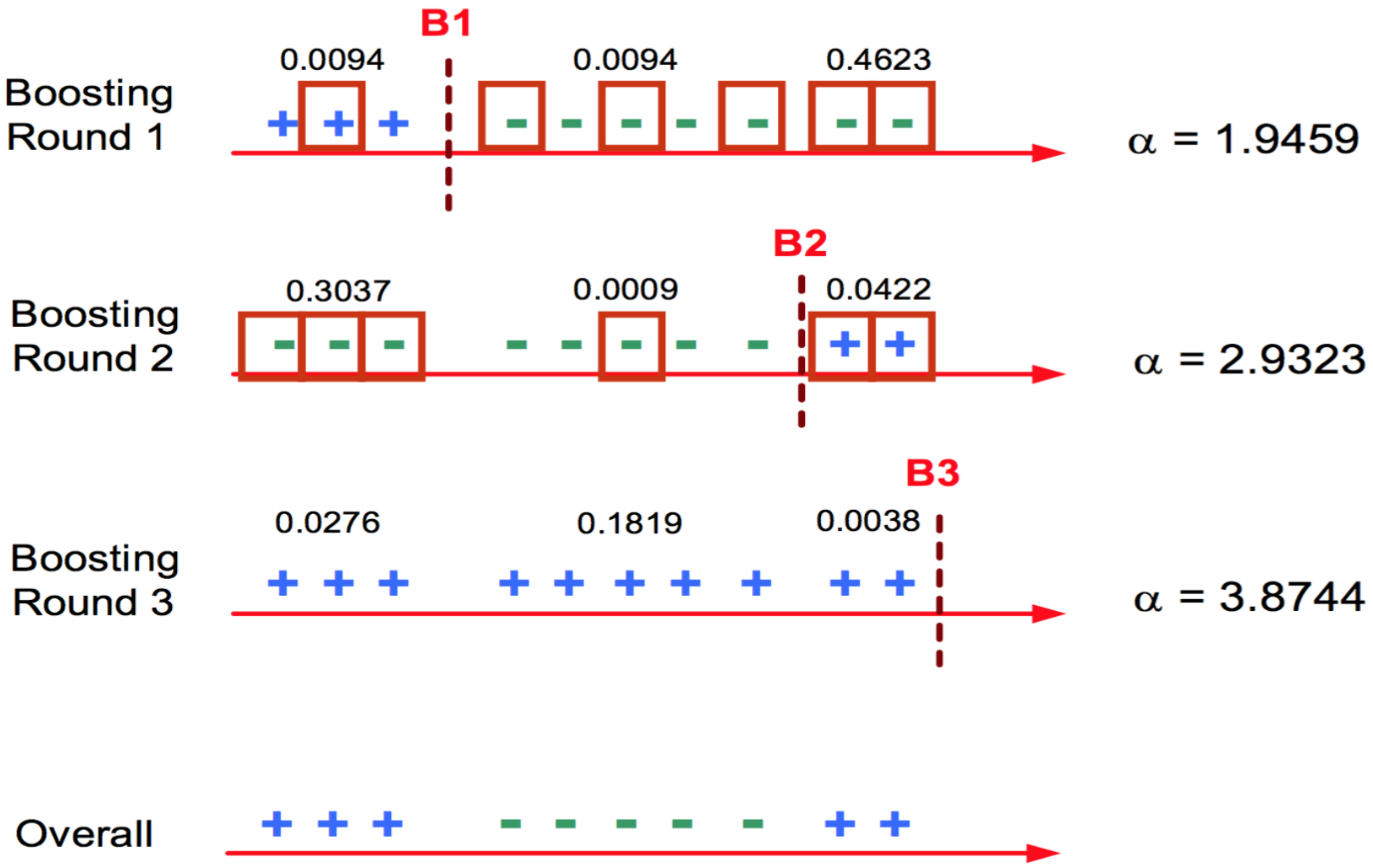
Classification:

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

Illustrating AdaBoost

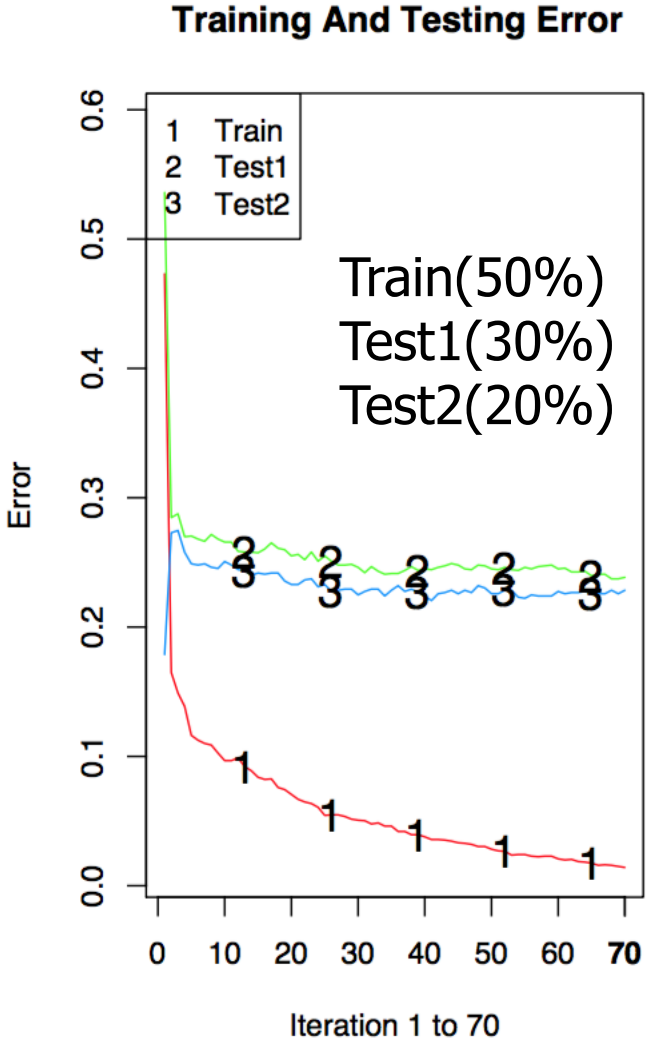


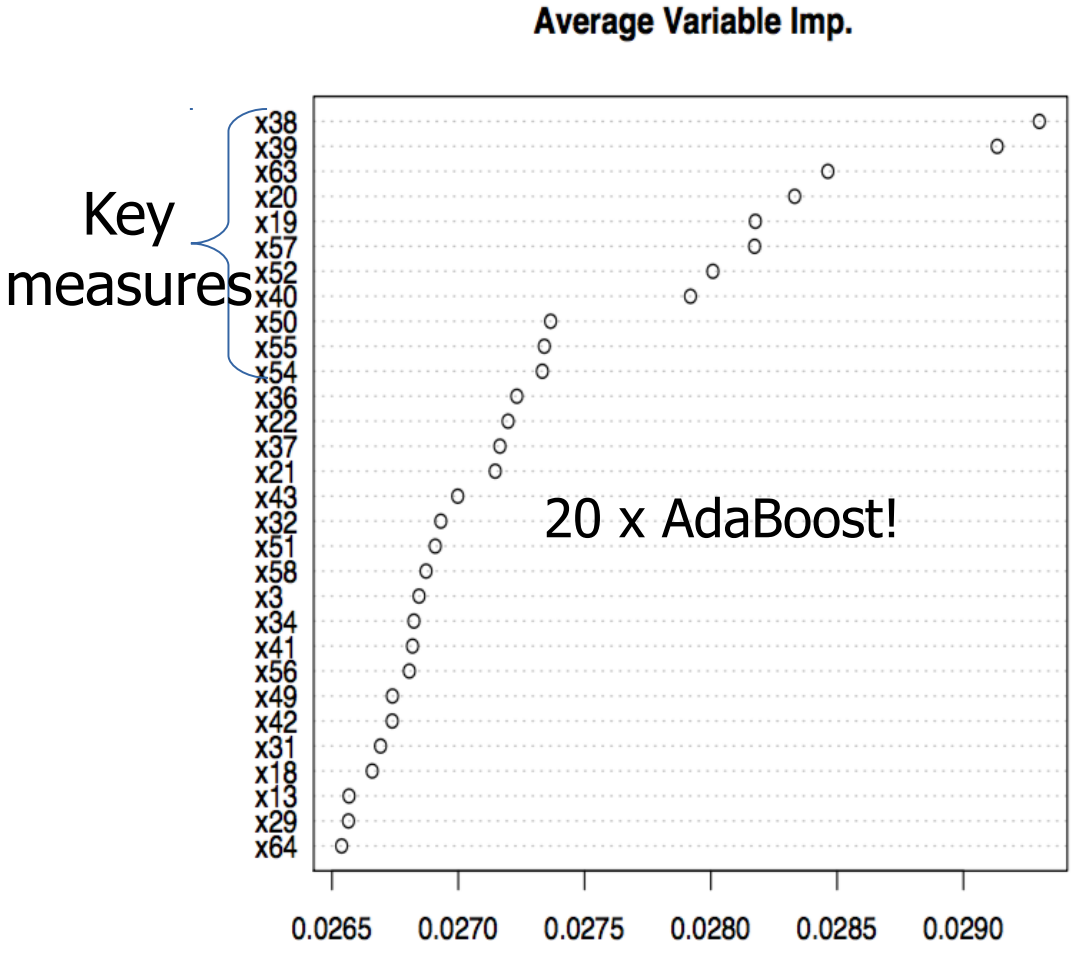
Illustrating AdaBoost



AdaBoost on Solubility data [5631 x 72]

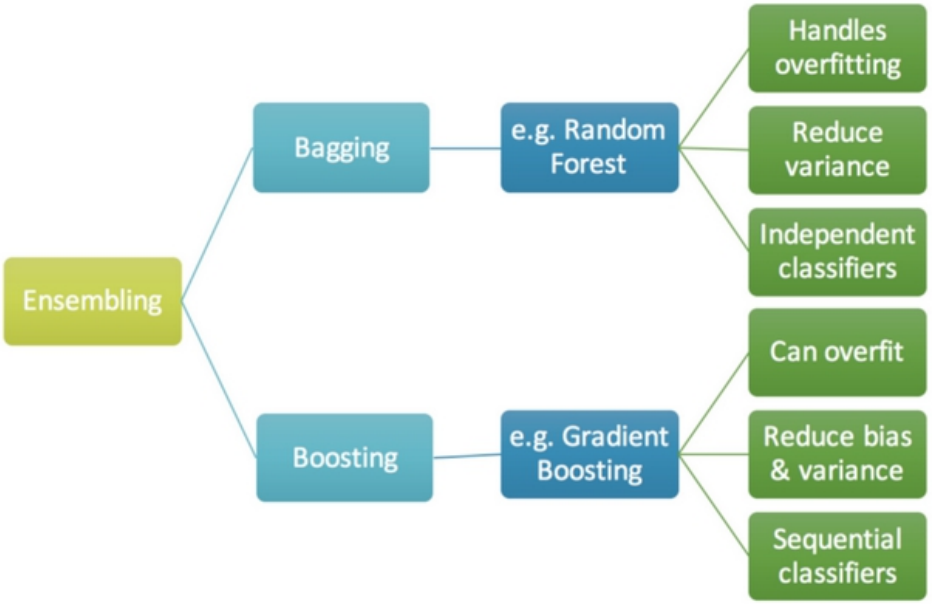
5631 compounds [either insoluble (**n=3493**) or soluble (**p=2138**)]
x 72 continuous, noisy structural and correlated descriptors

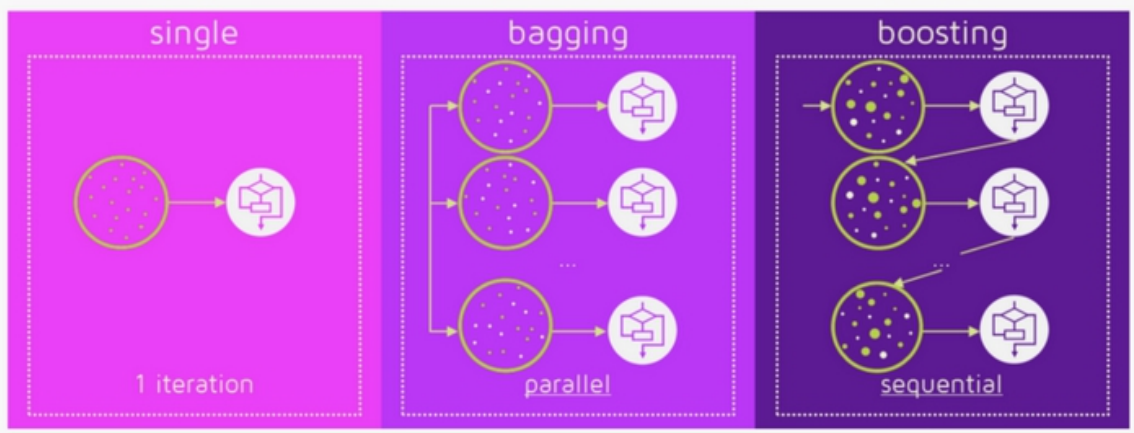




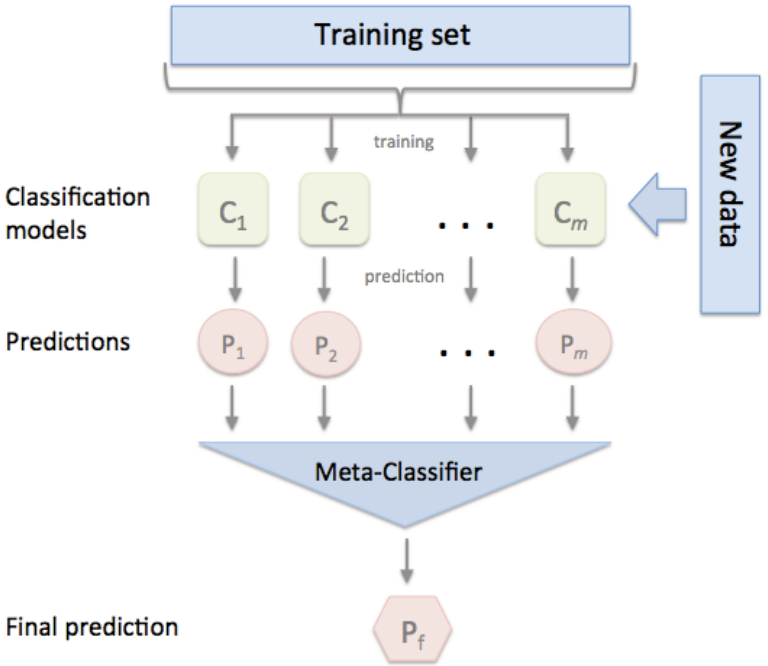

```
ada(y~., data = train, test.x = test[,-73], test.y = test[,73],
+                type = "gentle", control = control, iter = 70)
```

Tree-based ensembles



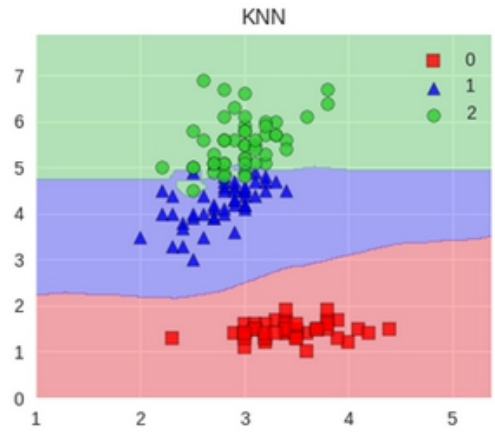


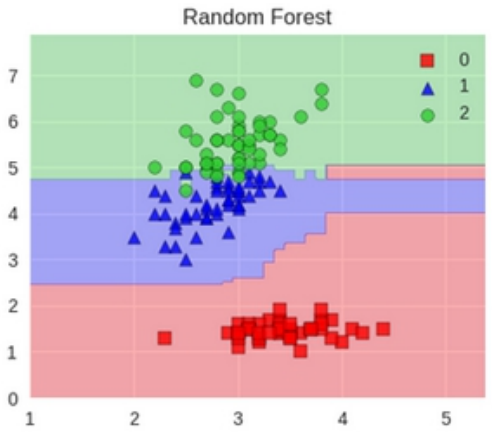
+ stacking

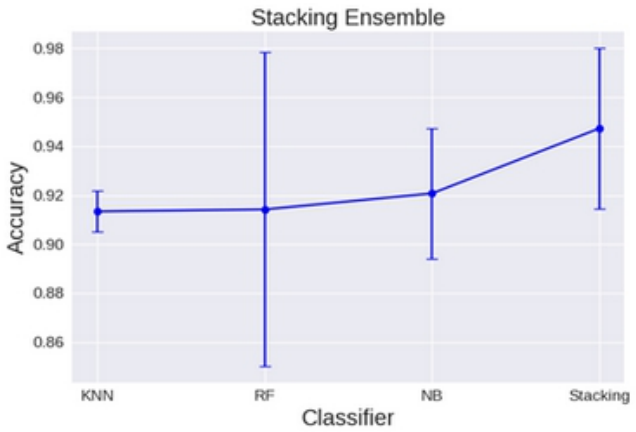


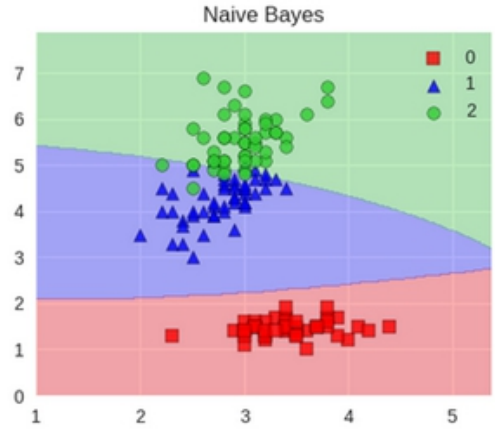
	Bagging	Boosting	Stacking
Partitioning of the data into subsets	Random	Giving mis-classified samples higher preference	Various
Goal to achieve	Minimize variance	Increase predictive force	Both
Methods where this is used	Random subspace	Gradient descent	Blending
Function to combine single models	(Weighted) average	Weighted majority vote	Logistic regression

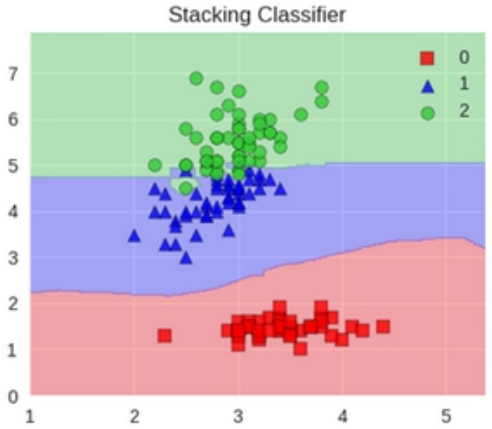
+ stacking

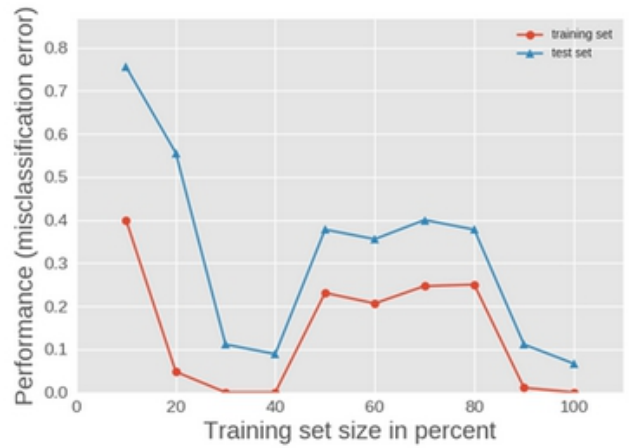






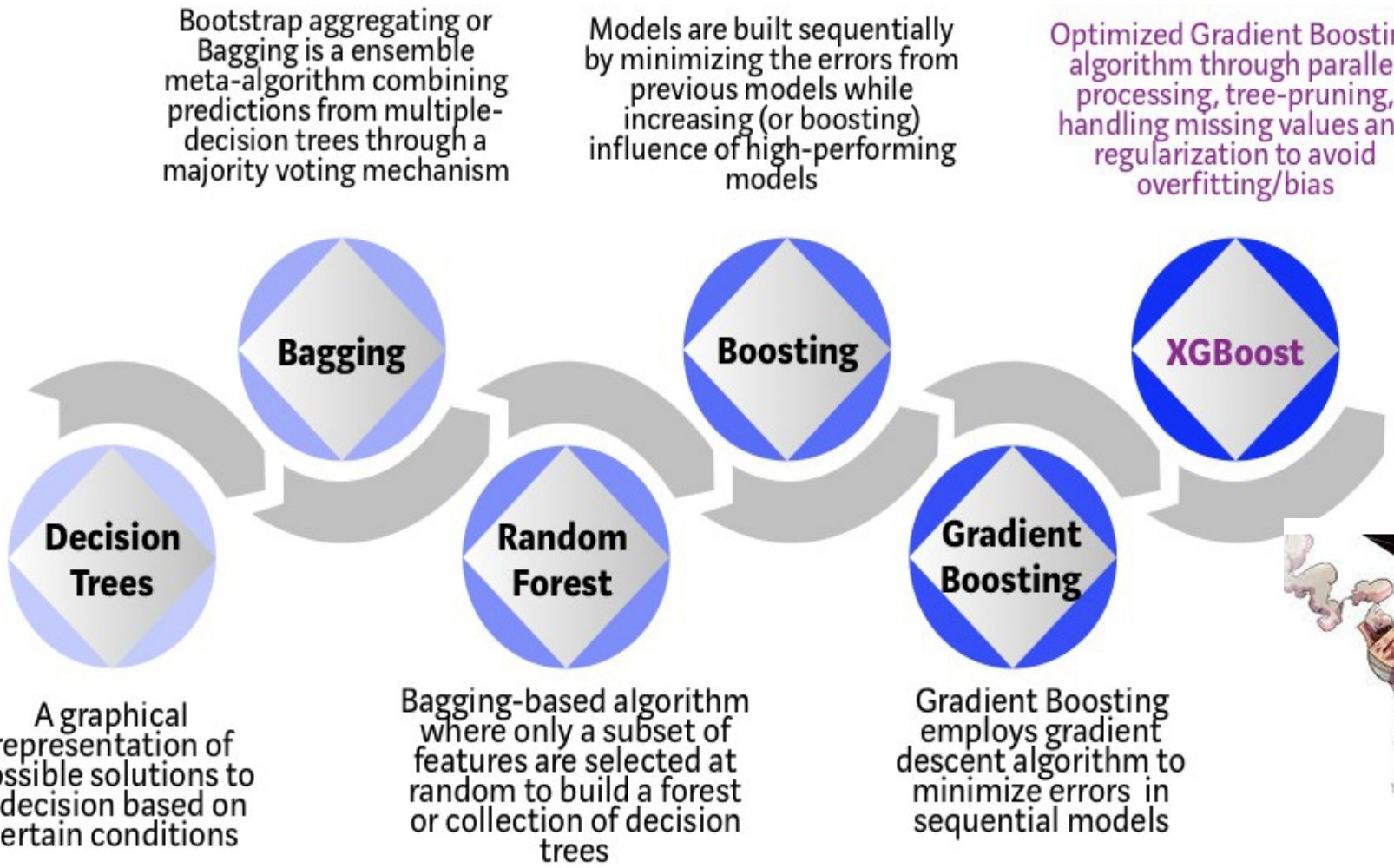






Accuracy: 0.91 (+/- 0.01) [KNN]
Accuracy: 0.91 (+/- 0.06) [Random Forest]
Accuracy: 0.92 (+/- 0.03) [Naive Bayes]
Accuracy: 0.95 (+/- 0.03) [Stacking Classifier]

The rise of the XGBoost [SIGKDD'16]



Summary

Ensemble systems are useful in practice

Diversity of the base classifiers is important

Bagging is predominantly a **variance-reduction technique**, while **boosting** is primarily a **bias-reduction technique**

Effectiveness on real world data depends on the **classifier diversity** and **characteristics of the data**

Given the predictions of some base classifiers as features, learn a **meta-model** that best combines their predictions