Dauphine UNIVERSITÉ PARIS | PSL

→ COMPUTER SCIENCE, DECISION-MAKING, AND DATA

Algorithmic and advanced Programming in Python

Eric Benhamou eric.benhamou@dauphine.eu
Remy Belmonte remy.belmonte@dauphine.eu

Masterclass 8

Outline

1. Introduction to Ensemble
2. Bagging
3. Boosting
4. Discussion around XGBoost and LightGBM

Reminder of the objective of this course

- People often learn about data structures out of context
- But in this course you will learn foundational concepts by building a real application with python and Flask

- To learn the ins and outs of the essential data structure, experiencing in practice has proved to be a much more powerful way to learn data structures

Reminder of previous session

In Master class 7, we discuss about graph traversal

Question: can you summarize the various algorithms seen?

Three major sections for classification

- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative
   directly estimate a decision rule/boundary
   e.g., support vector machine, decision tree, logistic regression,
   e.g. neural networks (NN), deep NN

2. Generative:
   build a generative statistical model
   e.g., Bayesian networks, Naïve Bayes classifier

3. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K nearest neighbors

Ensemble methods principle

In statistics and machine learning, **ensemble methods** use

- multiple learning algorithms
- to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

Its core principle: Together is better than alone as the majority vote cannot go wrong. Averaging over multiple experts should give a better answer!

Three major sections for classification

- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative
   directly estimate a decision rule/boundary
   e.g., support vector machine, decision tree, logistic regression,
   e.g. neural networks (NN), deep NN

2. Generative:
   build a generative statistical model
   e.g., Bayesian networks, Naïve Bayes classifier

3. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K nearest neighbors

Ensemble Core principles

- Framework of Ensemble:
  - 1. Get a set of classifiers $f_1(x), f_2(x), f_3(x), \ldots\ldots$

    They should be diverse.

    How to have different training data sets
    - Re-sampling your training data to form a new set
    - Re-weighting your training data to form a new set
  - 2. Aggregate the classifiers (*properly*)

The different type of Ensemble methods

- Bagging — reduce variance
  - Bagged Decision Tree
  - Random forests:
- Boosting — reduce bias, reduce variance
  - Adaboost
  - Xgboost — Gradient Boosting
- Stacking

Bagging

- Bagging or *bootstrap aggregation*
    - a technique for reducing the variance of an estimated prediction function.

- For instance, for classification, a *committee* of decision trees
    - Each tree casts a vote for the predicted class.

Bootstrap

The basic idea:

randomly draw datasets *with replacement (i.e. allows duplicates)* from the training data, each samples *the same size as the original training set*

$f_1, \quad f_2 \quad \cdots, \quad f_B \quad \# |Tr|$

① change training set $\{S_1, S_2, \ldots, S_B\}$

With Replacement

- Bootstrap with replacement can keep the sampling size the same as the original size for every repeated sampling. The sampled data groups are independent on each other.

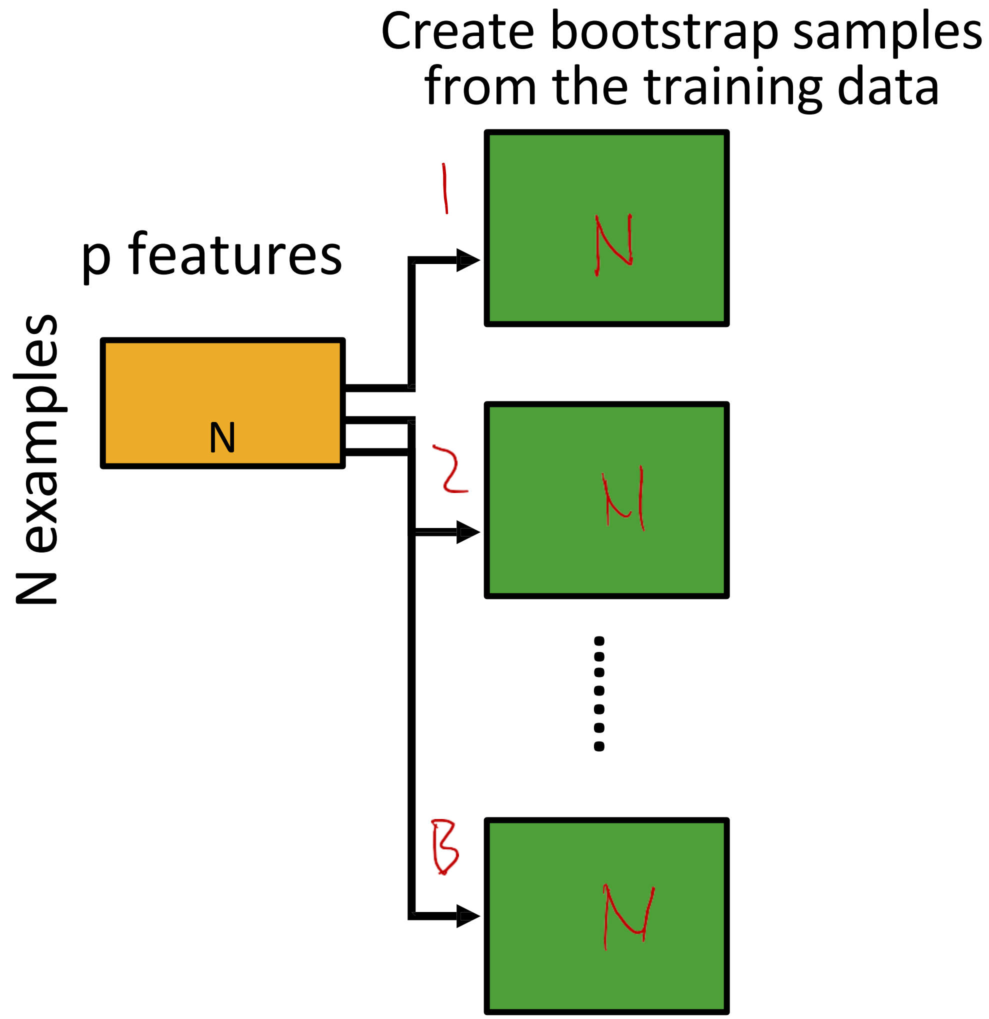
$S_1, S_2, \cdots, S_B$

Or Without Replacement

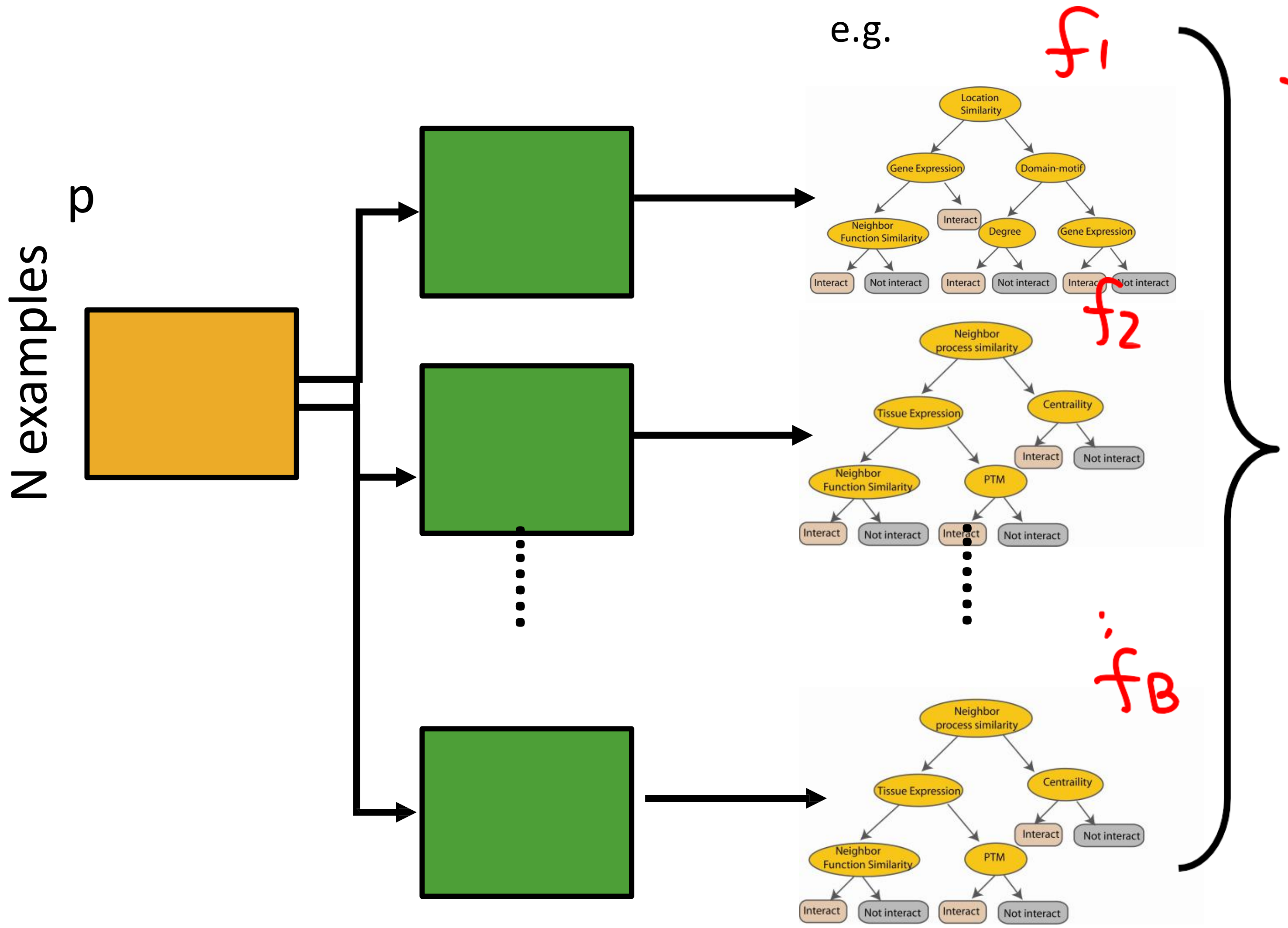
- Bootstrap without replacement cannot keep the sampling size the same as the original size for every repeated sampling. The sampled data groups are dependent on each other.

$S_1, S_2, \cdots, S_B$

Bagging with simple graphs



Bagging of DT Classifiers

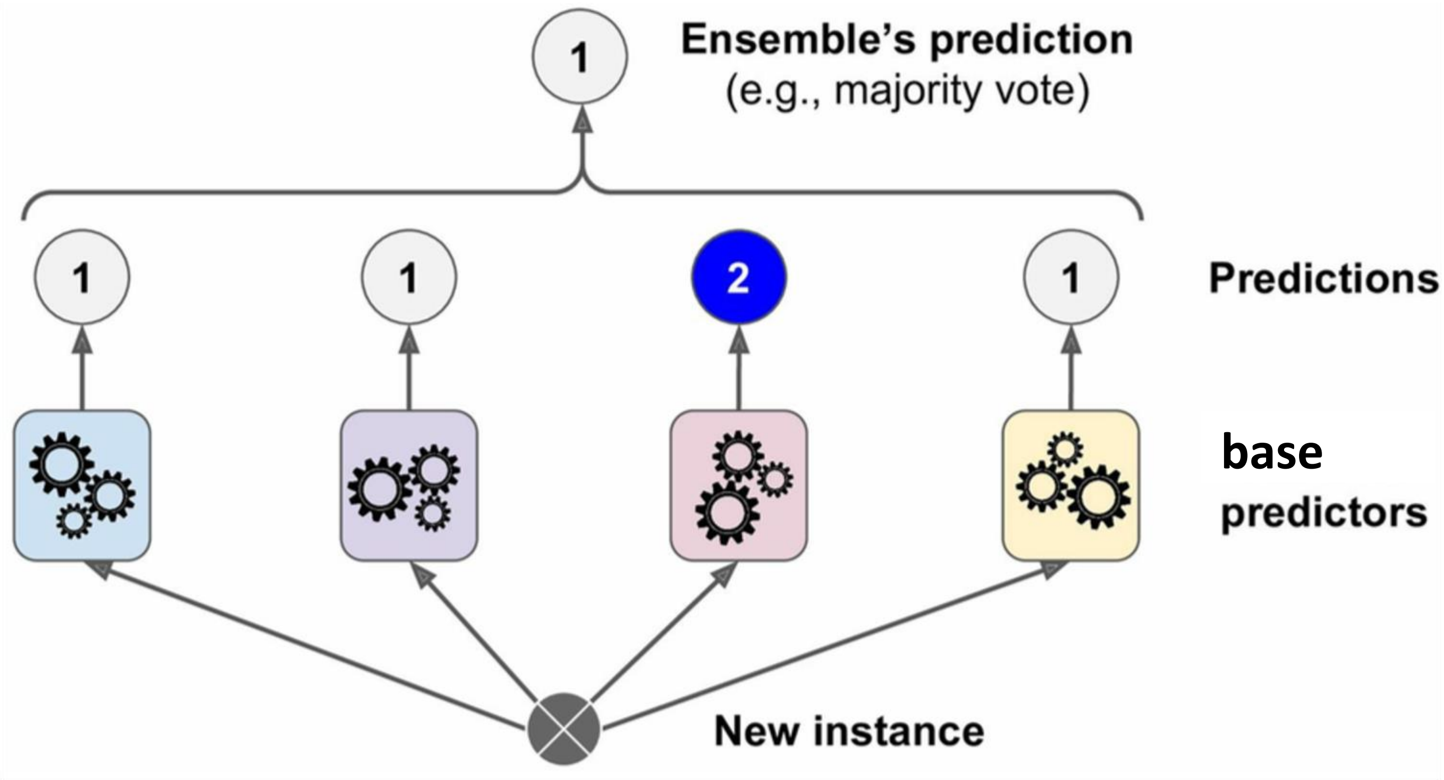


$$f_E = \frac{1}{B}\sum_{i=1}^{B} f_i$$

Take the majority vote

i.e. Refit the model to each bootstrap dataset, and then examine the behavior over the B replications.

E.g., Predict by Hard voting



Decision Boundary Comparison

$Var(f_i)$ vs $Var\left(\frac{1}{B}\sum_{i=1}^{B} f_i\right)$

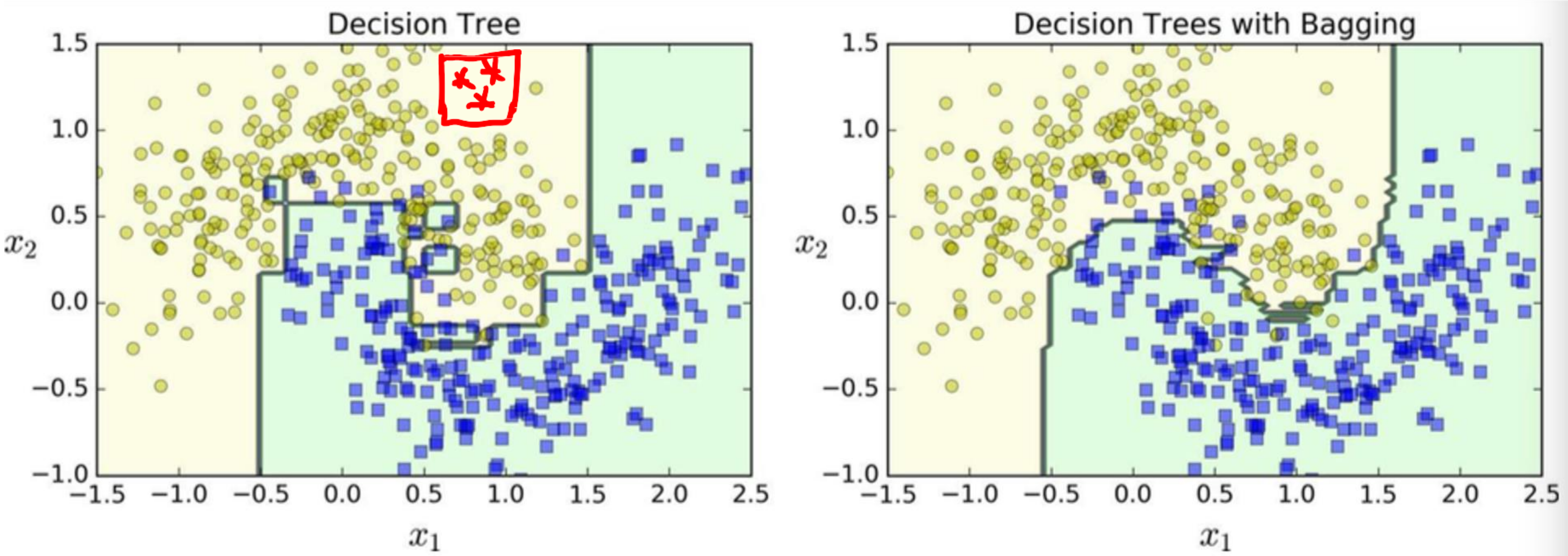
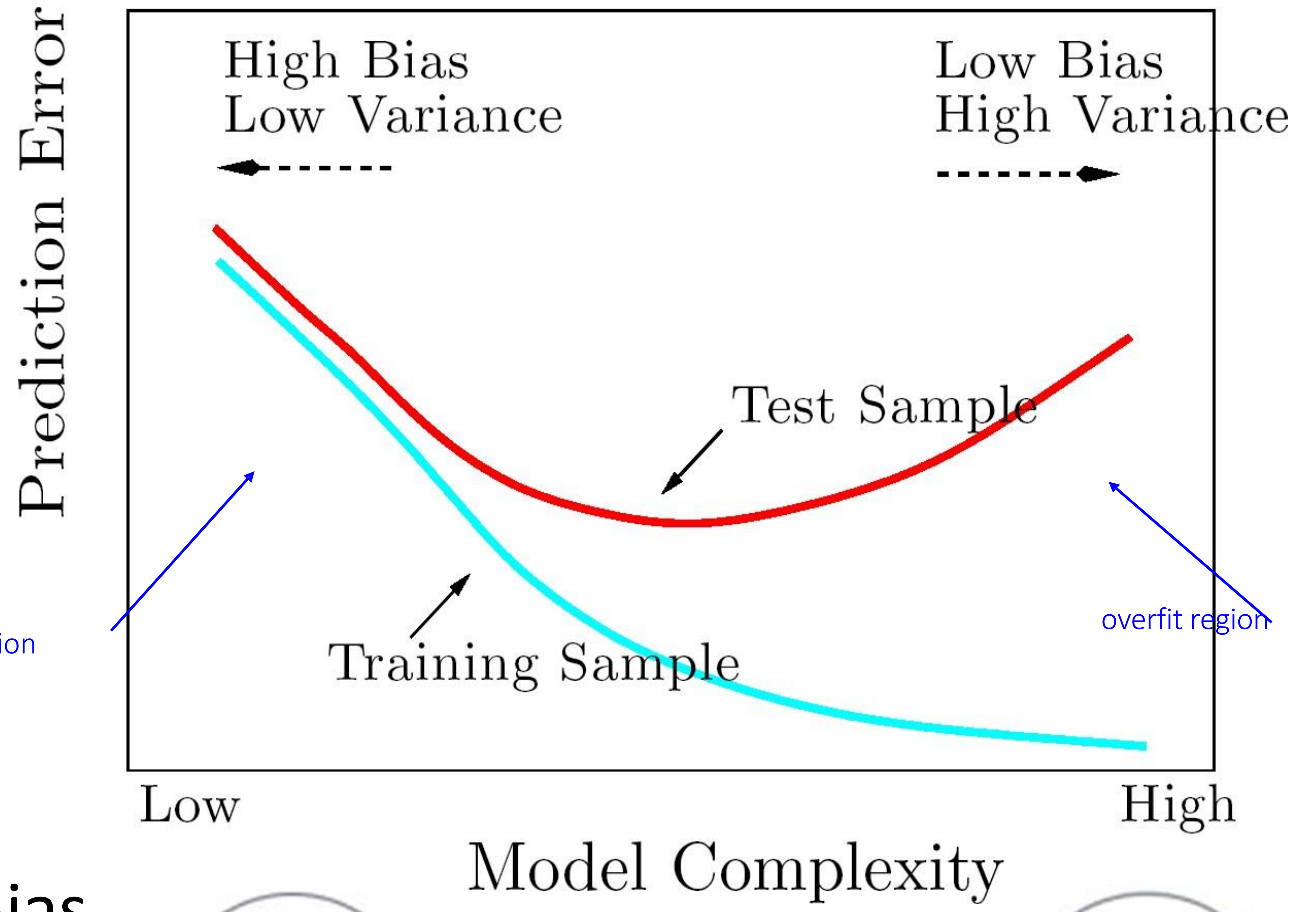


Figure 7-5. A single Decision Tree versus a bagging ensemble of 500 trees

Peculiarities of Bagging

- Model Instability is good when bagging
  - The more variable (unstable) the basic model is, the more improvement can potentially be obtained
  - Low-Variability methods (e.g. LDA) improve less than High-Variability methods (e.g. decision trees)

Bias-Variance Tradeoff / Model Selection

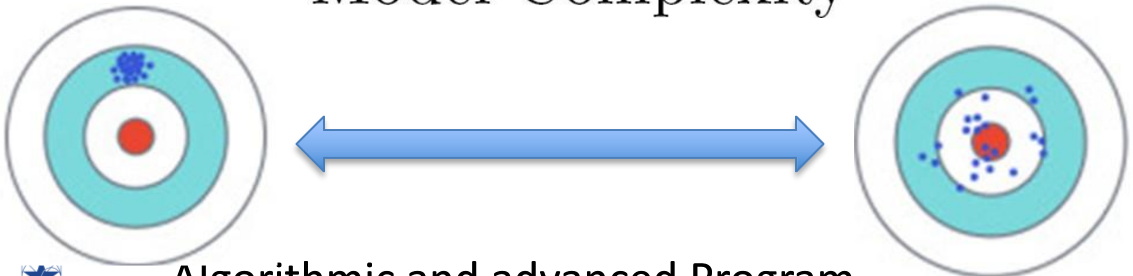


underfit region

overfit region

Large Bias

Small Variance

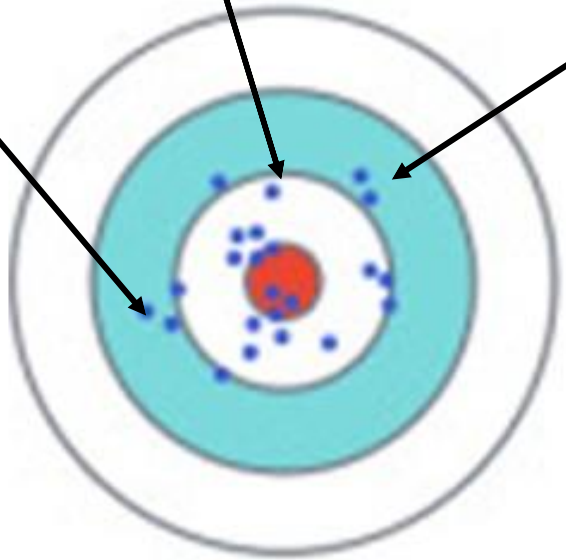


Small Bias Large Variance

In details

classifiers $f_1(x)$, $f_2(x)$, $f_3(x)$, ... $f_B(x)$

S_1 S_2 S_B



A complex model have large variance.

We can average complex models to reduce variance.

If we average all the f_i, is it close to f^*

$$E[f] = f^*$$

Algorithmic and advanced Programming in Python

$$\mathrm{Var}(aX + bY) = a^2\mathrm{Var}(X) + b^2\mathrm{Var}(Y) + 2ab\mathrm{Cov}(X, Y)$$

when $Cov(f_i, f_j) = 0$

$$Var(\hat{f}) = E((\hat{f} - \bar{f})^2) = Var(\frac{1}{B}\sum_{i=1}^{B} \hat{f}_i) = \frac{1}{B^2}\sum_{i=1}^{B} Var(\hat{f}_i) =$$

Base classifiers $f_1(x)$, $f_2(x)$, $f_3(x)$, ... $f_B(x)$

$$Var(\hat{f}) = Var\left(\frac{1}{B}\sum_{i=1}^{B}\hat{f}_i\right)$$

when $\forall i, j$, $Cov(f_i, f_j) = 0$

$$= \frac{1}{B^2}\sum_{i=1}^{B} Var(f_i)$$

, Because $|S_1| = |S_2| = \ldots = |S_B|$

$$= \frac{1}{B} Var(f_i)$$

, $Var(f_i) \simeq Var(f_j)$

EXTRA

Bagging : an extreme case study using simulated data (with correlated features)

N = 300 training samples, $y \in \{0, 1\}$, $x_1, x_2, \dots, x_5$

Y: Two classes and X: p = 5 features,

Each feature N(0, 1) distribution and pairwise correlation .95

Response Y: generated according to:

$$\Pr(Y = 1 | x_1 \leq 0.5) = 0.2 \qquad \Pr(Y = 1 | x_1 > 0.5) = 0.8$$

$y = 0$ $\quad$ $y = 1$

x_1

0.5

Test sample size of 2000

Fit classification trees to training set and bootstrap samples

B = 200

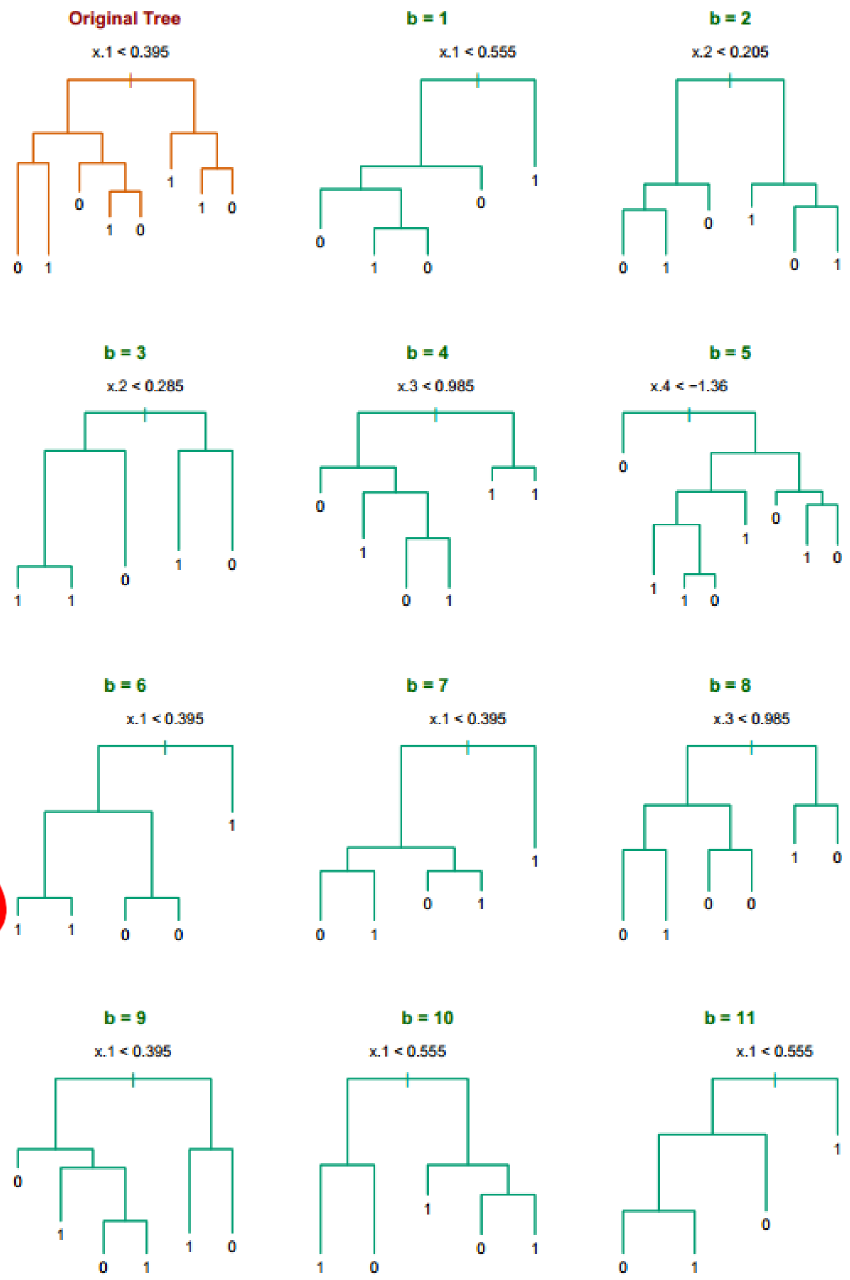
Notice the bootstrap trees look quite different from the original tree

$f_1(X)$

$f_2(X)$

⋮

$f_B(X)$



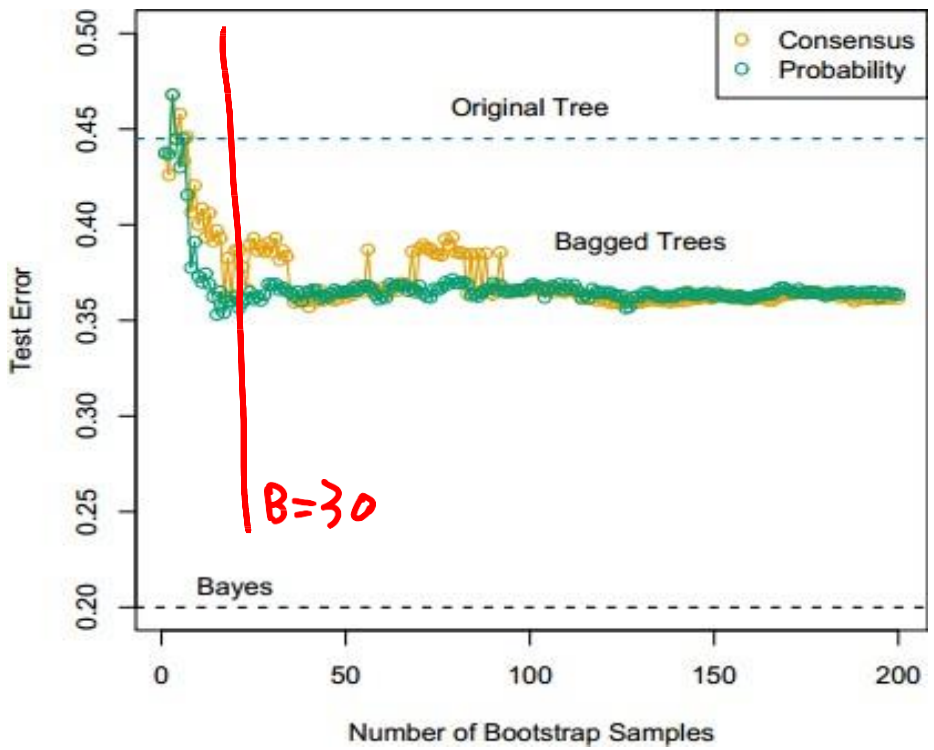
Five features highly correlated with each other

➔ No clear difference with picking up which feature to split

➔ Small changes in the training set will result in different tree

➔ But these trees are actually quite similar wrt output classification

For B>30, more trees do not improve the bagging results



Consensus: Majority vote

Probability: Average distribution at terminal nodes

➔ Since the trees correlate highly to each other and give similar classifications

B $Var\left(\frac{1}{B}\sum f_i\right)$

Bagging

- Slightly increases model complexity
  - Cannot help when greater enlargement of model diversity is needed

- Bagged trees are correlated
  - Use random forest to reduce correlation between trees

$$\mathrm{Var}(aX + bY) = a^2\mathrm{Var}(X) + b^2\mathrm{Var}(Y) + \underline{2ab\mathrm{Cov}(X, Y)}$$

Bagged Decision Tree

Data

Task

Representation

Score Function

Search/Optimization

Models,
Parameters

Tabular

Classification / Regression

multiple (almost) full
decision trees /
bootstrap samples

Split with Purity measure /
e.g. IG / cross-entropy / Gini
/

Greedy like Decision Tree
(e.g. GINI)

Multiple Tree Model
(s), i.e. space partition

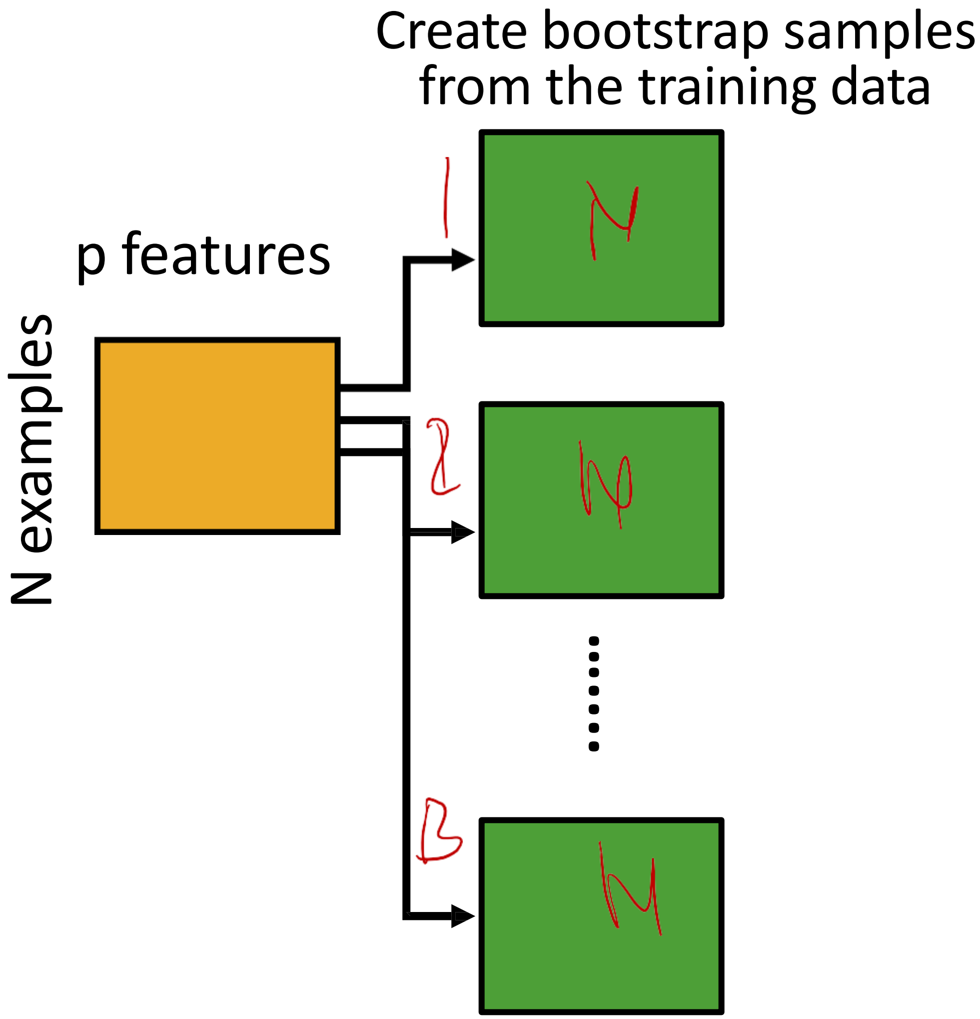
Let us discuss random forest

- Bagging
  - Bagged Decision Tree
  - Random forests
- Boosting
  - Adaboost
  - Xgboost
- Stacking

Random forest classifier

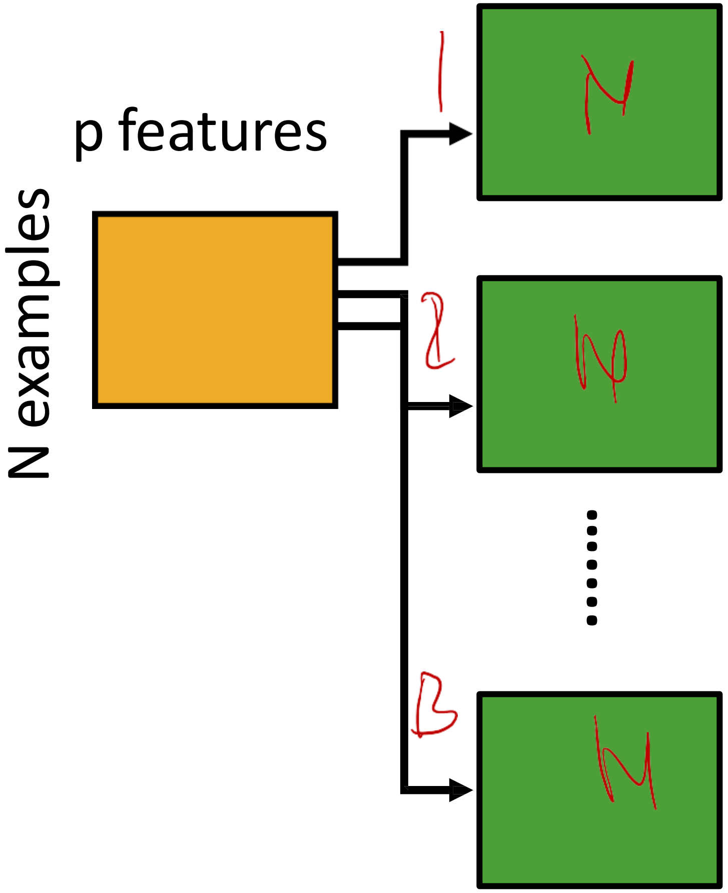
- Random forest classifier,
  - an extension to bagging
  - which uses *de-correlated* trees.

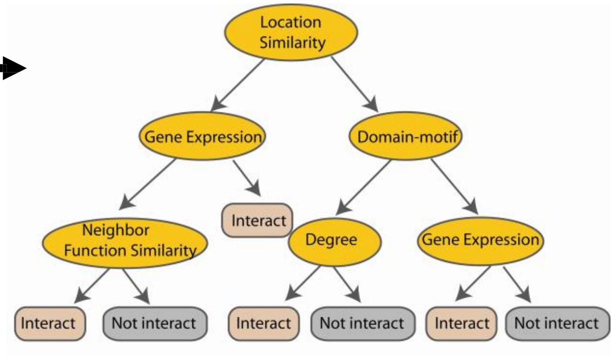
Random Forest Classifier



Random Forest Classifier

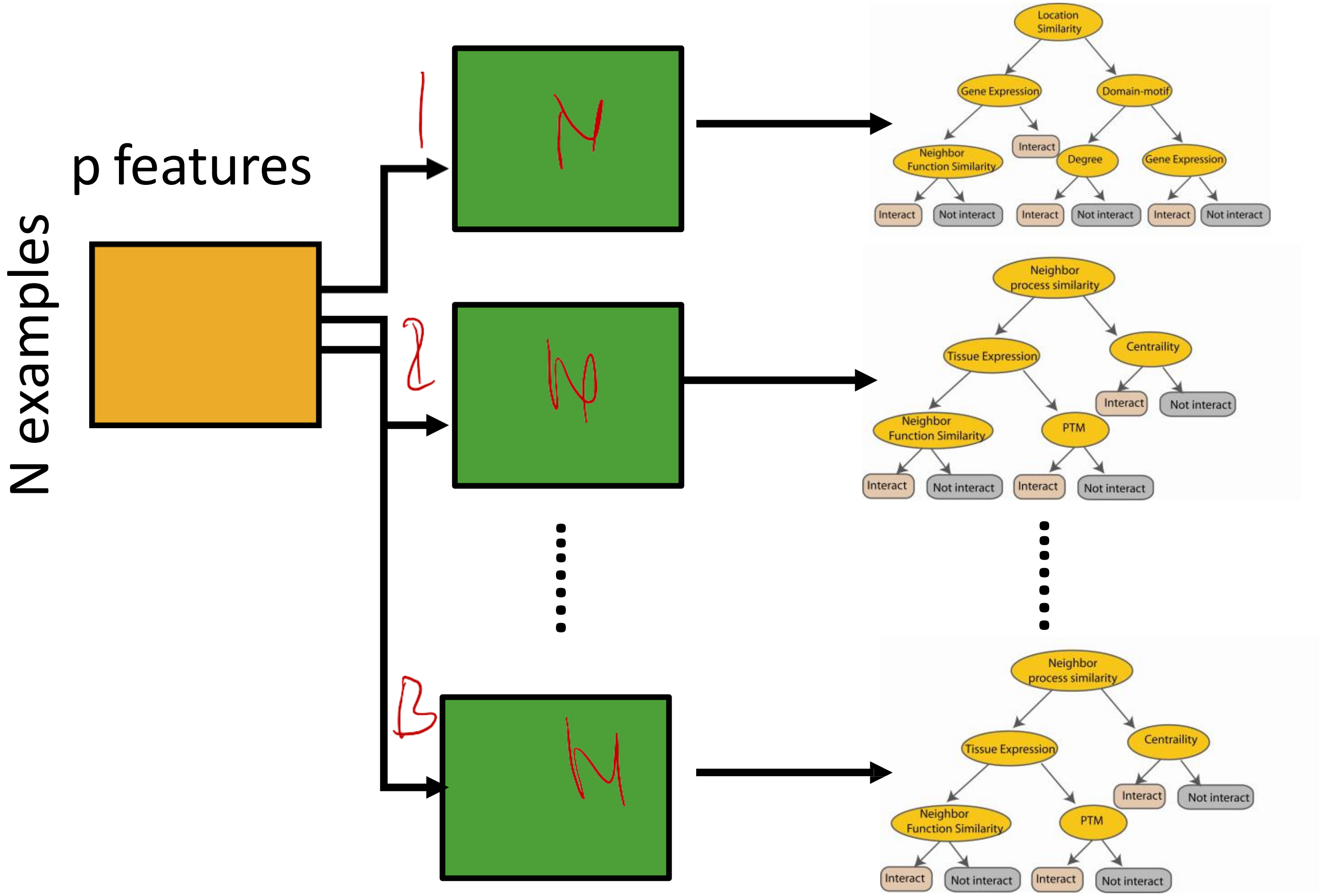
At each node when choosing the split feature
choose only among *m<p* features



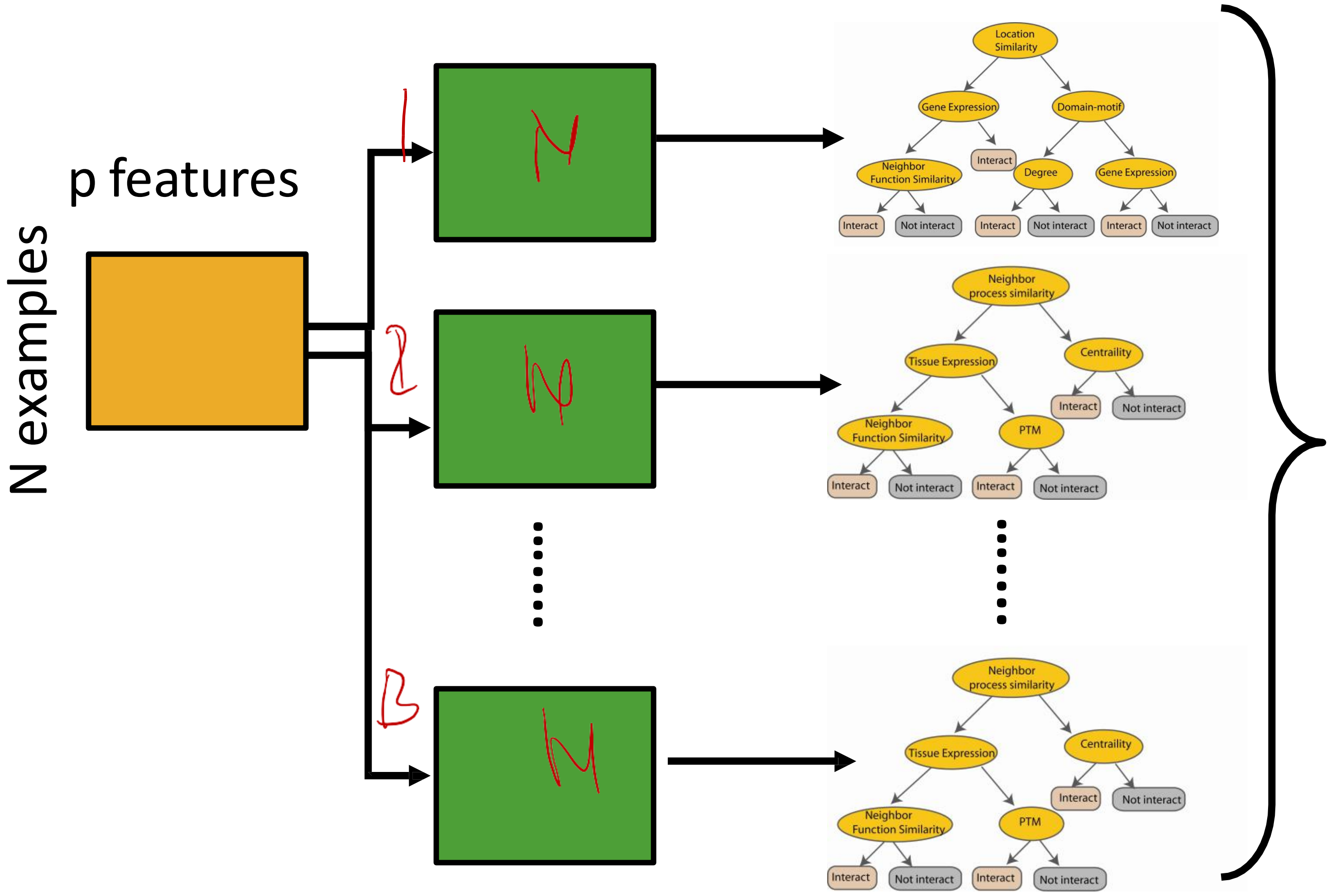


Random Forest Classifier

Create decision tree from each bootstrap sample



Random Forest Classifier



Take he majority vote

Random Forests

For each of our B bootstrap samples

- Form a tree in the following manner
  - i: Given p dimensions, pick m of them
  - ii: Split only according to these m dimensions
    - (we will NOT consider the other p-m dimensions)
  - Repeat the above steps i & ii for each split
    - Note: we pick a different set of m dimensions for each split on a single tree

$\rho = 0,06$

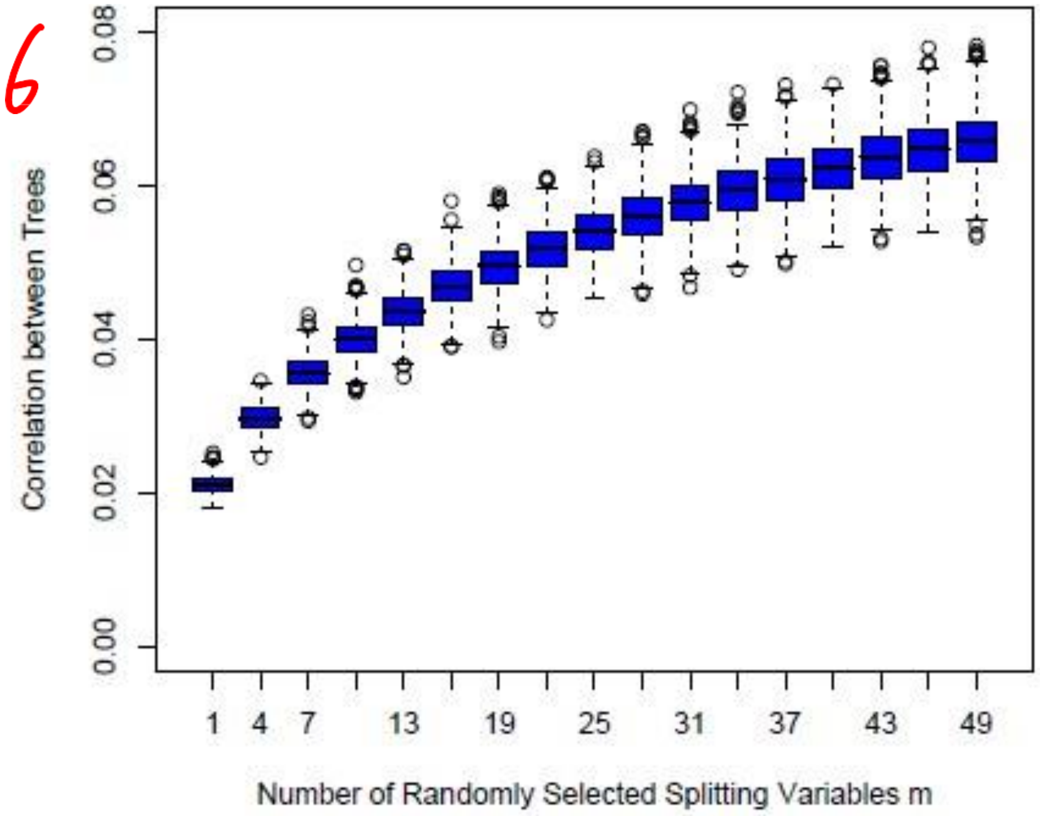


FIGURE 15.9. *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of m. The boxplots represent the correlations at 600 randomly chosen prediction points x.*

Page 598-599 In ESL book

Random Forests

Random forest can be viewed as a refinement of bagging with a tweak of **decorrelating** the trees:

At each tree split, a random subset of **m** features out of all **p** features is drawn to be considered for splitting

Some guidelines provided by Breiman, but be careful to choose m based on specific problem:

m = p amounts to bagging

m = p/3 or log2(p) for regression

m = sqrt(p) for classification

Why correlated trees are not ideal ?

Random Forests try to reduce correlation between the trees.

Why?

Why correlated trees are not ideal ?

Assuming each tree has variance σ^2

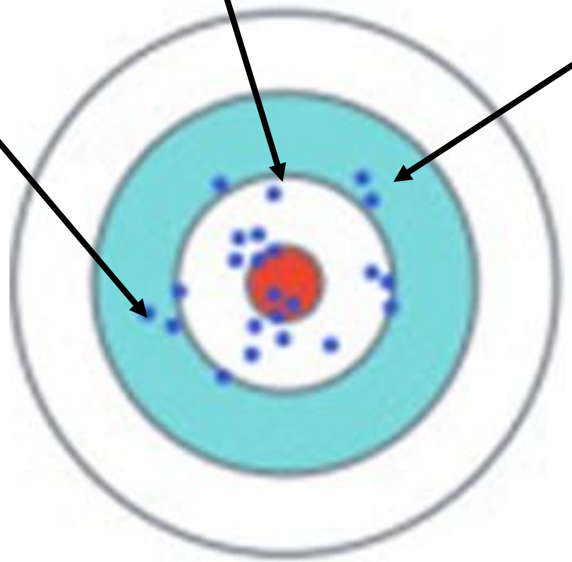
If trees are independently identically distributed, then average variance is σ^2/B

$$\mathrm{Var}(aX + bY) = a^2\mathrm{Var}(X) + b^2\mathrm{Var}(Y) + 2ab\mathrm{Cov}(X, Y)$$

$$Var(\bar{f}) = E((\bar{f} - f\bar{\$}\ ^2) = Var(\frac{1}{B}2 \quad \bar{f}_i) = \frac{1}{B^2}2 \quad \underbrace{(\quad)}_{}^{B\sigma^2} = \frac{1}{B}\sigma^2$$

$$Var \quad \bar{f}_i$$

classifiers $f_1(x)$, $f_\$(x)$, $f_3(x)$, ... $f_B(x)$



A complex model will have large variance.

We can average complex models to reduce variance

If we average all the f_i, is it close to f^*

$$E[\bar{f}] = f^*$$

Why correlated trees are not ideal ?

Assuming each tree has variance σ^2

If simply identically distributed, then average variance is

$\rho_{ij} = \mathrm{Corr}(f_i, f_j) = \rho$

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

As B → ∞, second term → 0

Thus, the pairwise correlation always affects the variance

Why correlated trees are not ideal ?

Assuming each tree has variance σ^2

If simply identically distributed, then average variance is

$$\Rightarrow \rho\sigma^2 + \cancel{\frac{1-\rho}{B}\sigma^2}$$

RAM ×B
trainTim ×B
testTime ×B

As B → ∞, second term → 0

Thus, the pairwise correlation always affects the variance

Why correlated trees are not ideal ?

How to deal?

If we reduce *m* (the number of dimensions we actually consider in each splitting),

then we reduce the pairwise tree correlation

Thus, variance will be reduced.

More about Random Forests

1. Construct subset $(x_1^*, y_1^*), \ldots, (x_n^*, y_n^*)$ by sampling original training set with replacement.
2. Build tree-structured learners $h(x, \Theta_k)$, where at each node, m predictors at random are selected before finding the best split.
   - Gini Criterion.
   - No pruning.
3. Combine the predictions (average or majority vote) to get the final result.

Random Forest

Task

Representation

Score Function

Search/Optimization

Models, Parameters

Classification / Regression

multiple (almost) full decision trees / bootstrap samples / sample features

Split with Purity measure / e.g. IG / cross-entropy / Gini /

Greedy like Decision Tree (e.g. GINI)

Multiple Tree Model (s), i.e. space partition

Let us discuss Boosting

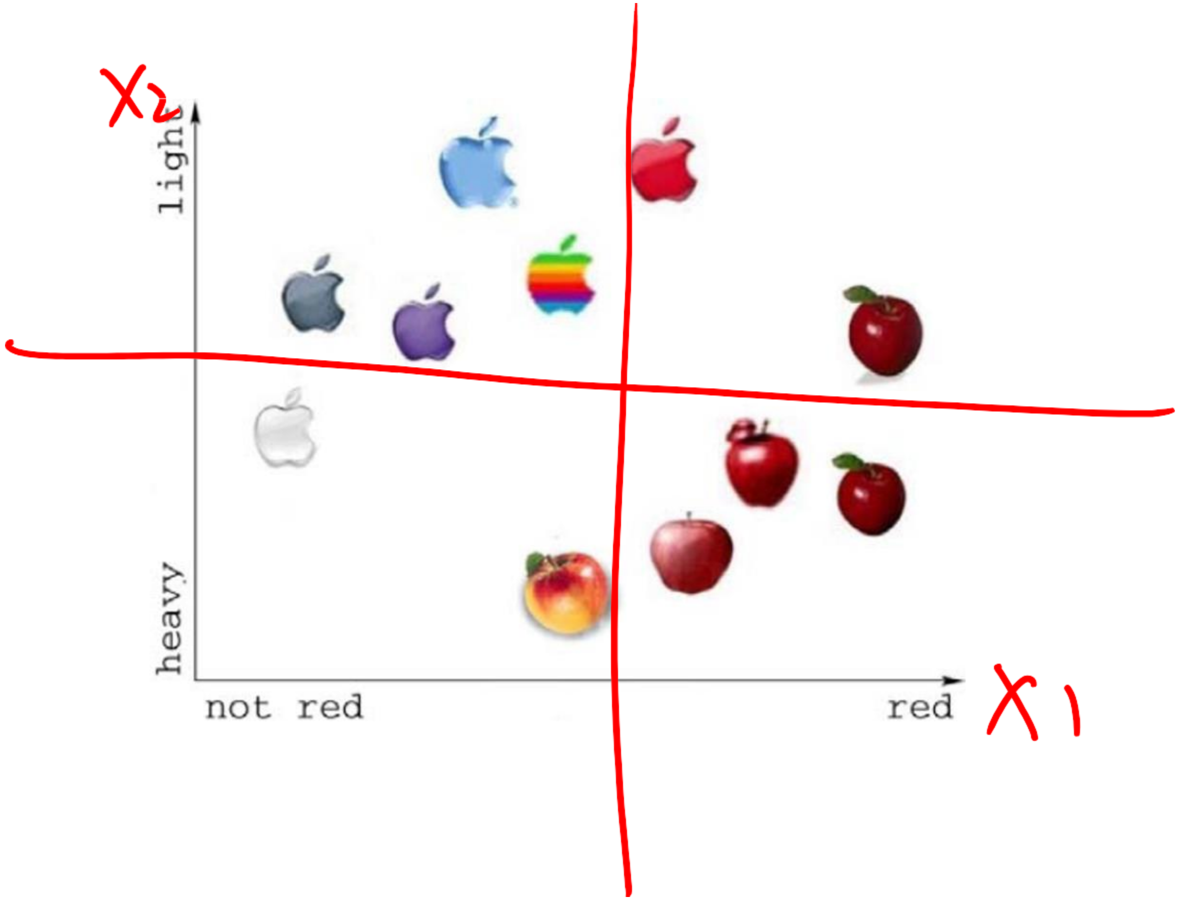
- Bagging
  - Bagged Decision Tree
  - Random forests:
- Boosting
  - Adaboost
  - Xgboost
- Stacking

Boosting Strategies

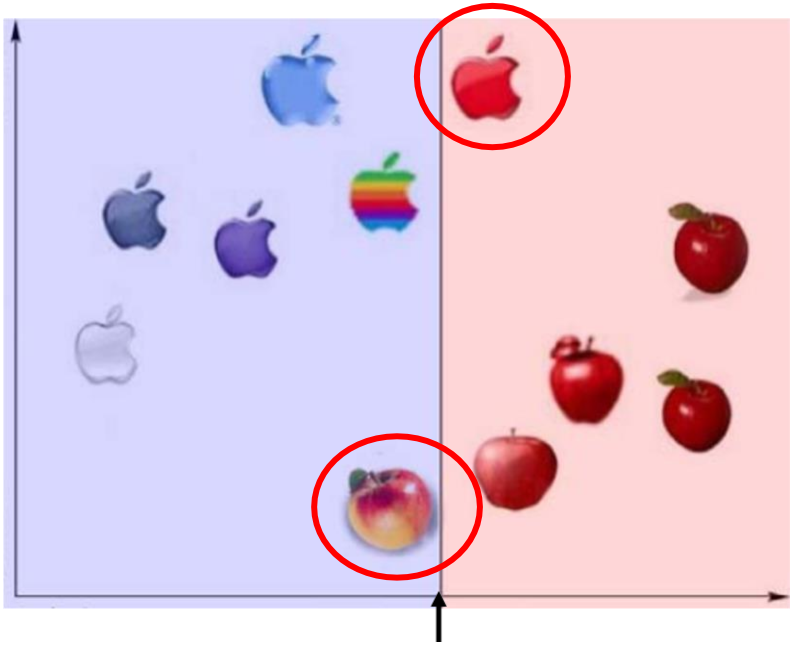
base learners : shallow DT
High bias

1. Have many rules (base classifiers) to **vote** on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules

- Recognizing apples:
- (1) Collect a set of real apples and plastic apples
- (2) Observe some rules to tell them apart based on their characteristics



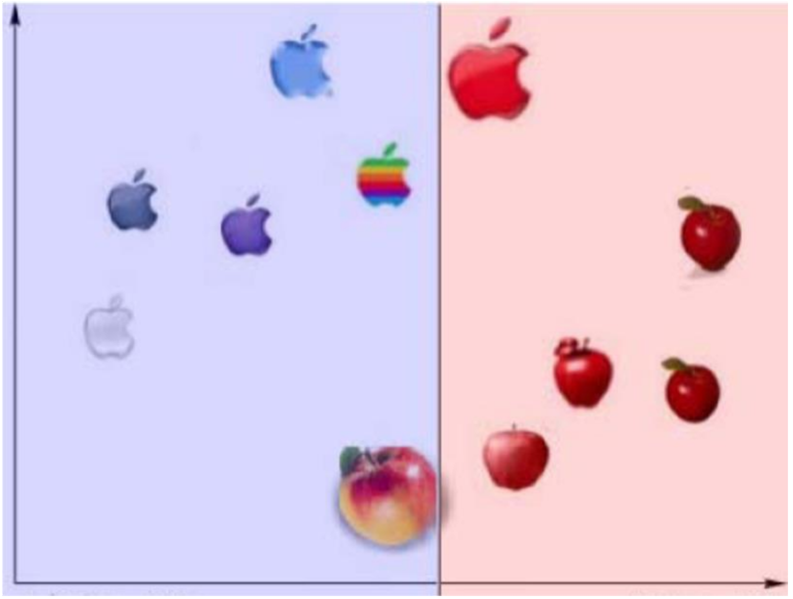
1. Have many rules (base classifiers) to vote on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



1st Simple Rule

1. Have many rules (base classifiers) to vote on the decision

2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



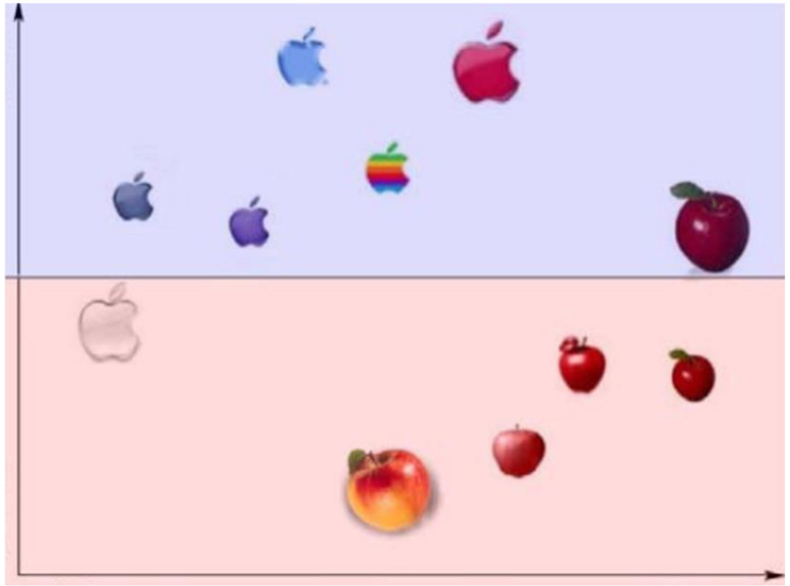
1. Have many rules (base classifiers) to vote on the decision

2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules

2nd Rule

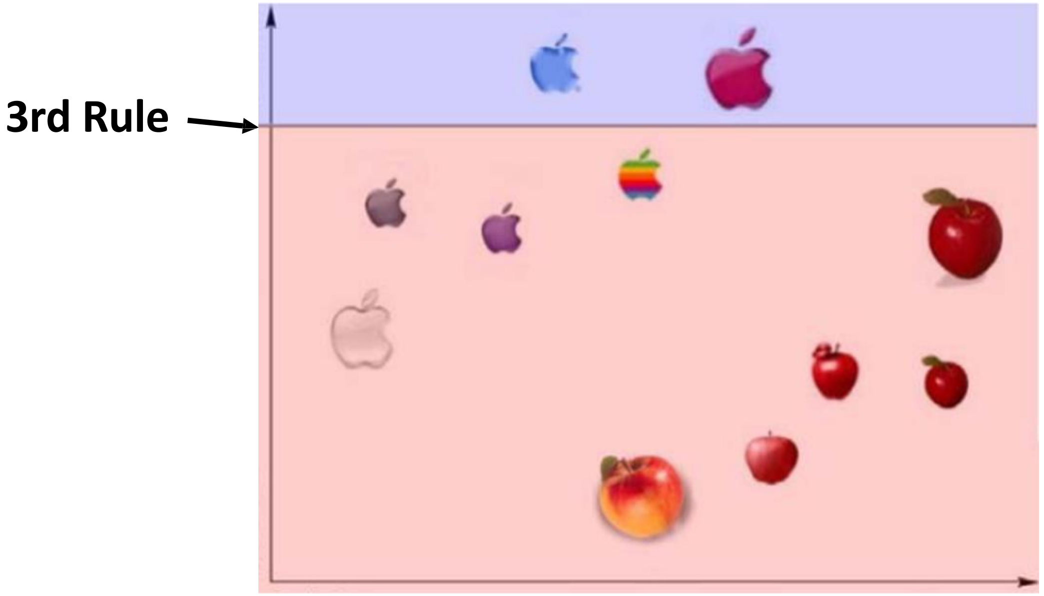
1. Have many rules (base classifiers) to vote on the decision

2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



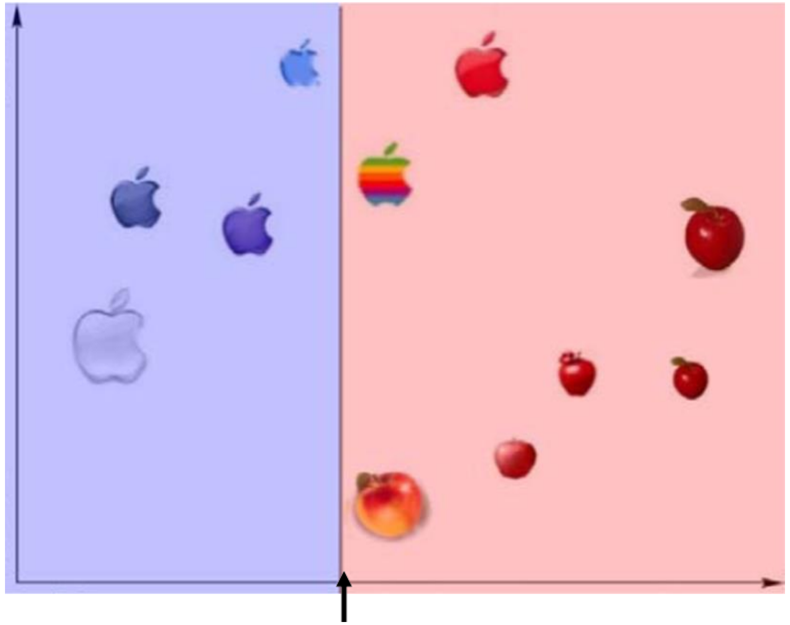
1. Have many rules (base classifiers) to vote on the decision

2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules

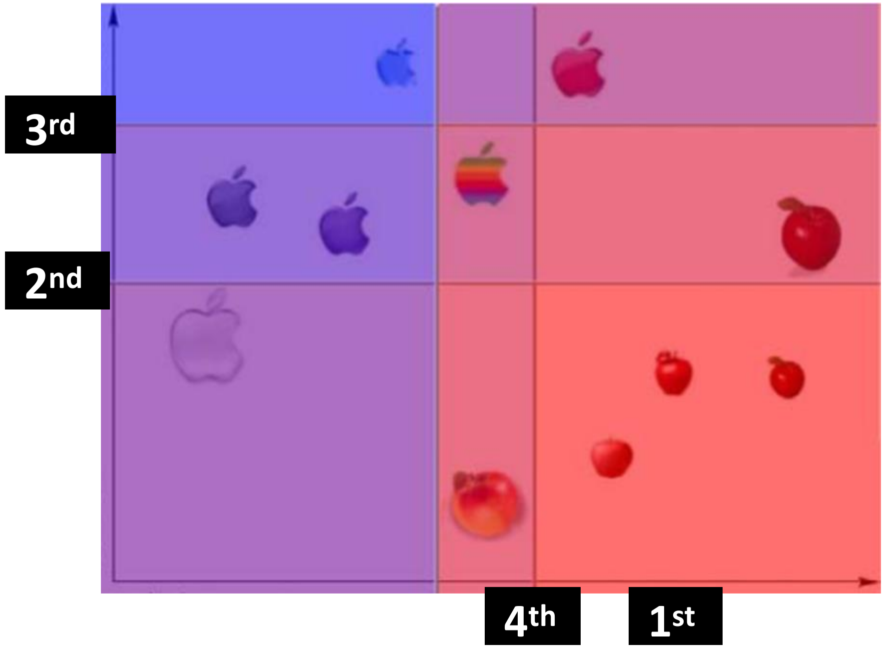


1. Have many rules (base classifiers) to vote on the decision

2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules

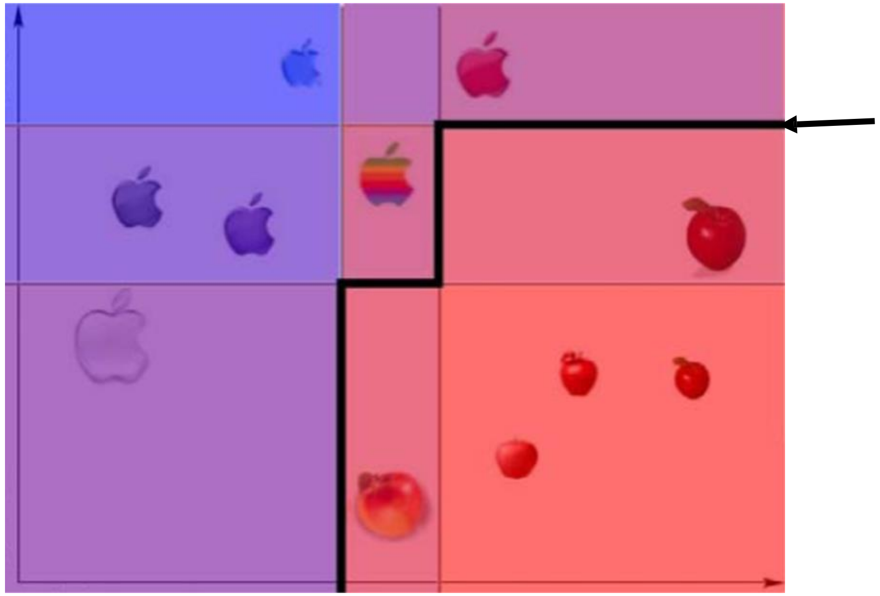


1. Have many rules (base classifiers) to vote on the decision
2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



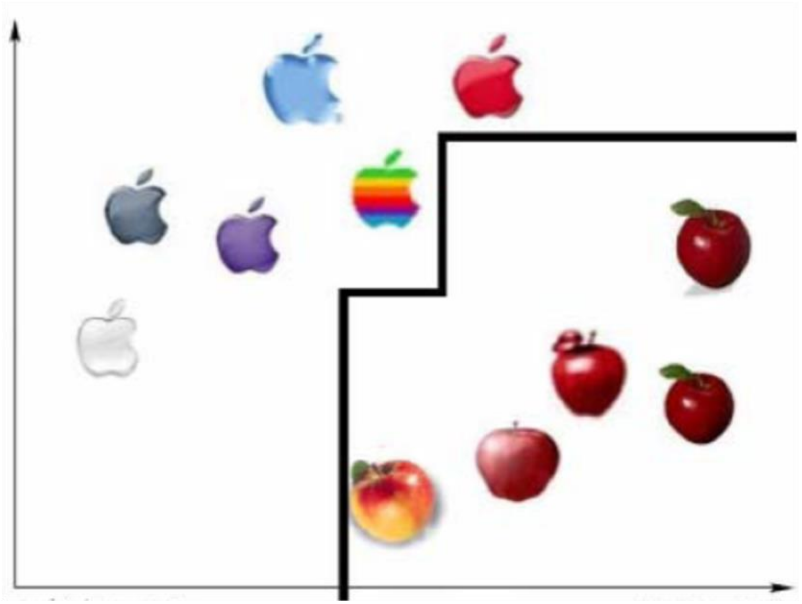
1. Have many rules (base classifiers) to vote on the decision

2. Sequentially train base classifiers that **corrects** mistakes of previous → focus on **hard** examples
3. Give higher **weight** to better rules



A More Complex Rule

Final Classifier is the additive combination of base rules:



Adaboost Algorithm (Proposed by Robert Schapire)

Training Data: $\mathbf{D} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathcal{R}^n, y_i \in \{-1, 1\}, 1 \leq i \leq m\}$

Set uniform example weight $w_i, 1 \leq i \leq m$

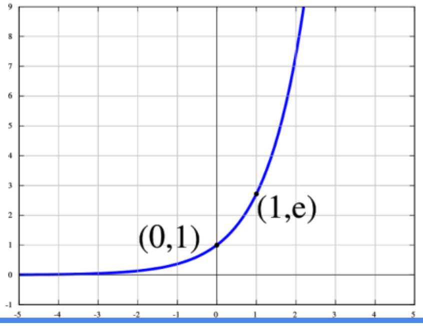
For t = 1 **to** T **iterations:**

Select a base classifier: $h_t(\mathbf{x}_i) = \arg\min(\epsilon_t)$

$$\epsilon_t = \sum_{i=1}^{m} w_i [y_i \neq h_t(\mathbf{x}_i)]$$

Set classifier weight: $\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$

Update example weight: $w_i = w_i e^{-\alpha_t y_i h_t(\mathbf{x}_i)}$



Final Classifier: $\hat{f} = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right)$

Boosting vs. Bagging

- Similar to bagging, boosting combines a weighted sum of many classifiers, thus **it reduces variance**.

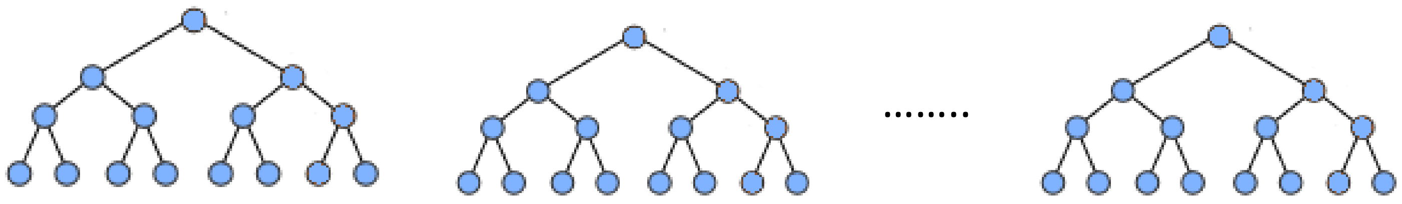
- One key difference: unlike bagging, boosting fit the tree to the entire training set, and adaptively weight the examples.

- Boosting tries to do better at each iteration, (by making model a bit more complex), thus
ıs.

XGBoost

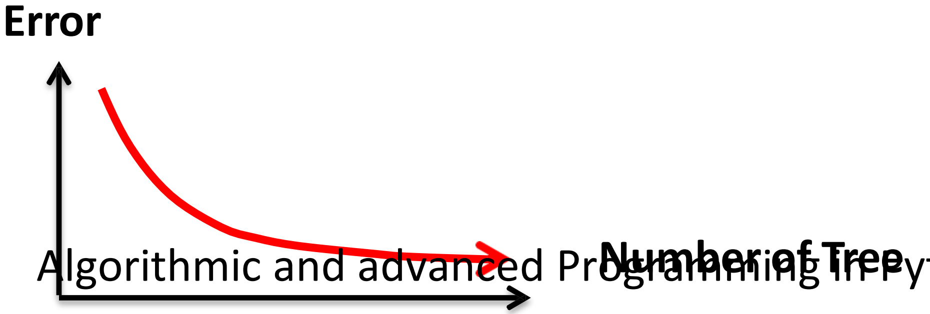
- Additive tree model: add new trees that complement the already-built ones
- Response is the optimal linear combination of all decision trees
- Popular in Kaggle Competitions for efficiency and accuracy

Additive tree model



More in 18c-extraBoosting Slides

Greedy Algorithm



XGBoost

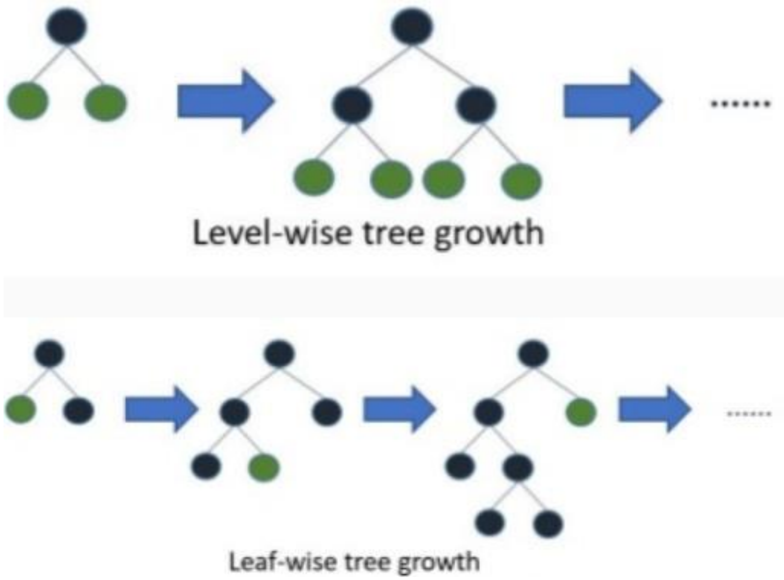
- XGBoost is a very efficient Gradient Boosting Decision Tree implementation with some interesting features:
- **Regularization:** Can use L1 or L2 regularization.
- **Handling sparse data:** Incorporates a sparsity-aware split finding algorithm to handle different types types of sparsity patterns in the data.
- **Weighted quantile sketch:** Uses distributed weighted quantile sketch algorithm to effectively handle weighted data.
- **Block structure for parallel learning:** Makes use of multiple cores on the CPU, possible because of a block structure in its system design. Block structure enables the data layout to be reused.
- **Cache awareness:** Allocates internal buffers in each thread, where the gradient statistics can be stored.
- **Out-of-core computing:** Optimizes the available disk space and maximizes its usage when handling huge datasets that do not fit into memory.

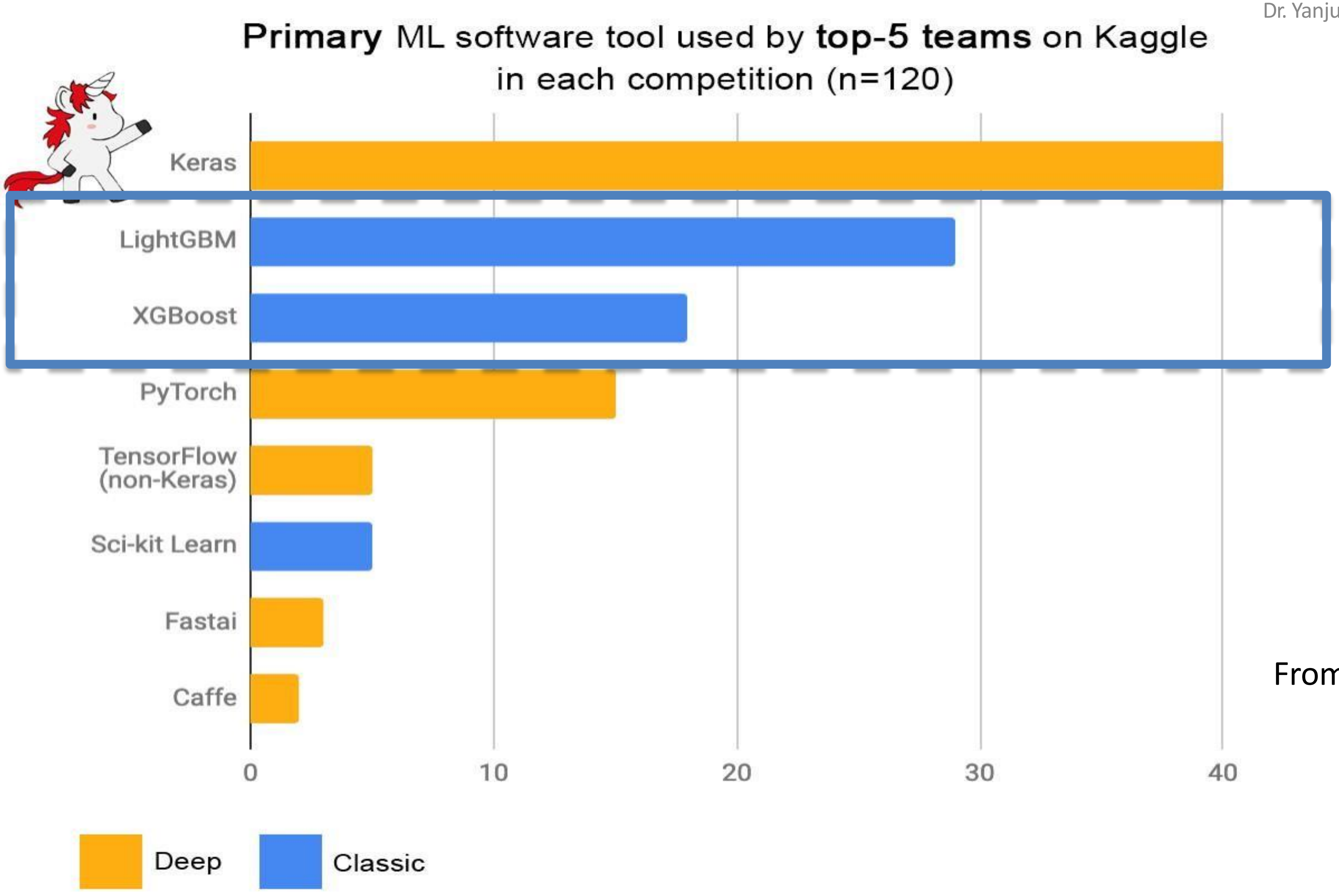
More about History …

- Introduction of Adaboost:
  - Freund; Schapire (1999). "A Short Introduction to Boosting“
- Multiclass/Regression
  - Y. Freund, R. Schapire, “A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting”, 1995.
  - Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory, pages 80–91, 1998.
- Gentle Boost
  - Schapire, Robert; Singer, Yoram (1999). "Improved Boosting Algorithms Using Confidence-rated Predictions".

LGBM

- Stands for Light Gradient Boosted Machines. It is a library for training GBMs developed by Microsoft, and it competes with XGBoost.
- Extremely efficient implementation.
- Usually much faster than XGBoost with low hit on accuracy.
- Main contributions are two novel techniques to speed up split analysis: Gradient based one-side sampling and Exclusive Feature Building.
- Leaf-wise tree growth vs level-wise tree growth of XGBoost.





From: François Chollet 2019

Boosting

Data
↓
Task
↓
Representation
↓
Score Function
↓
Search/Optimization
↓
Models, Parameters

Tabular
↓
Classification / Regression
↓
Weighted sum of a series of shallow decision trees
↓
Exponential Loss
↓
Gradient based search of weights / DT split
↓
Multiple Trees + Tree Weights

Stacking

- Bagging
  - Bagged Decision Tree
  - Random forests
- Boosting
  - Adaboost
  - Xgboost
- Stacking

e.g. Ensembles in practice

Each rating/sample:
+ <user, movie, date of grade, grade>
Training set (100,480,507 ratings) Qualifying set (2,817,131 ratings)➔ winner

Oct 2006 - 2009

- Training data is a set of users and ratings (1,2,3,4,5 stars) those users have given to movies.
- Predict what rating a user would give to any movie

- $1 million prize for a 10% improvement over Netflix's current method (MSE = 0.9514)

Team “Bellkor's Pragmatic Chaos” defeated the team “ensemble” by submitting just 20 minutes earlier! 1 million dollar !

Ensemble in practice

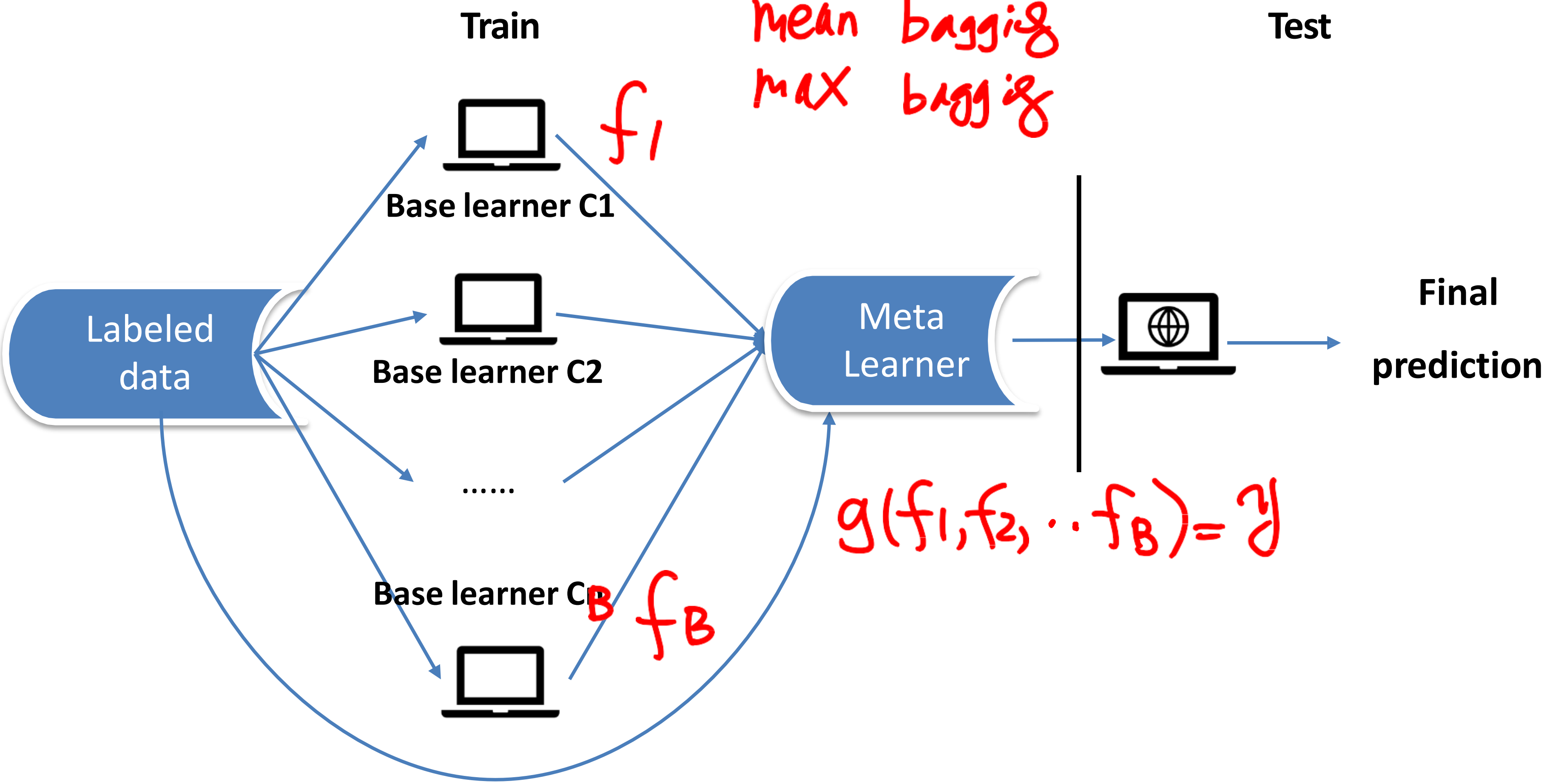
Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

The ensemble team ➔ blenders of multiple different methods

Stacking

- Main Idea: Learn and combine multiple classifiers

mean bagging
max bagging



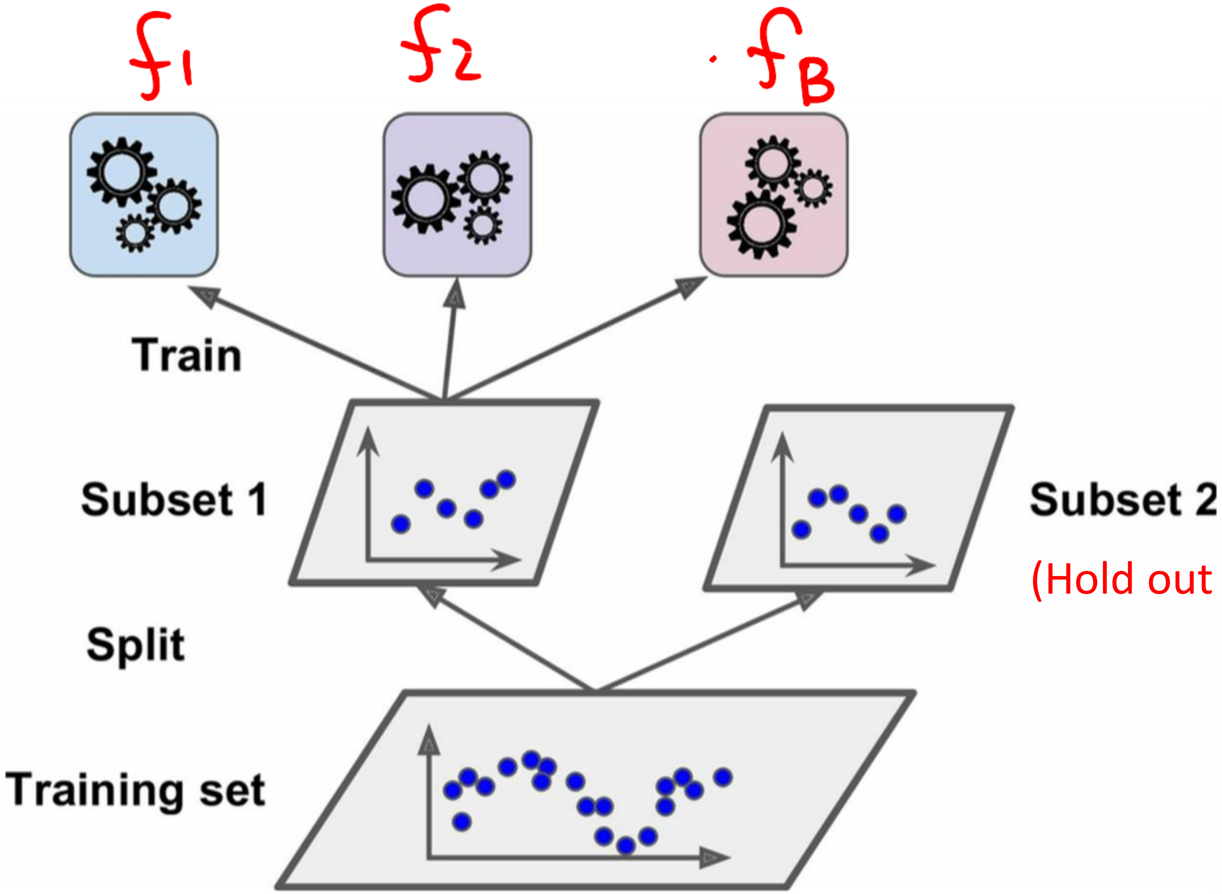
Generating Base and Meta Learners

- **Base model—efficiency, accuracy and diversity**
  - Sampling training examples
  - Sampling features
  - Using different learning models

- **Meta learner**
  - Majority voting
  - Weighted averaging
  - …..
  - Higher level classifier — Supervised (e.g. Xgboost as blender)

Unsupervised

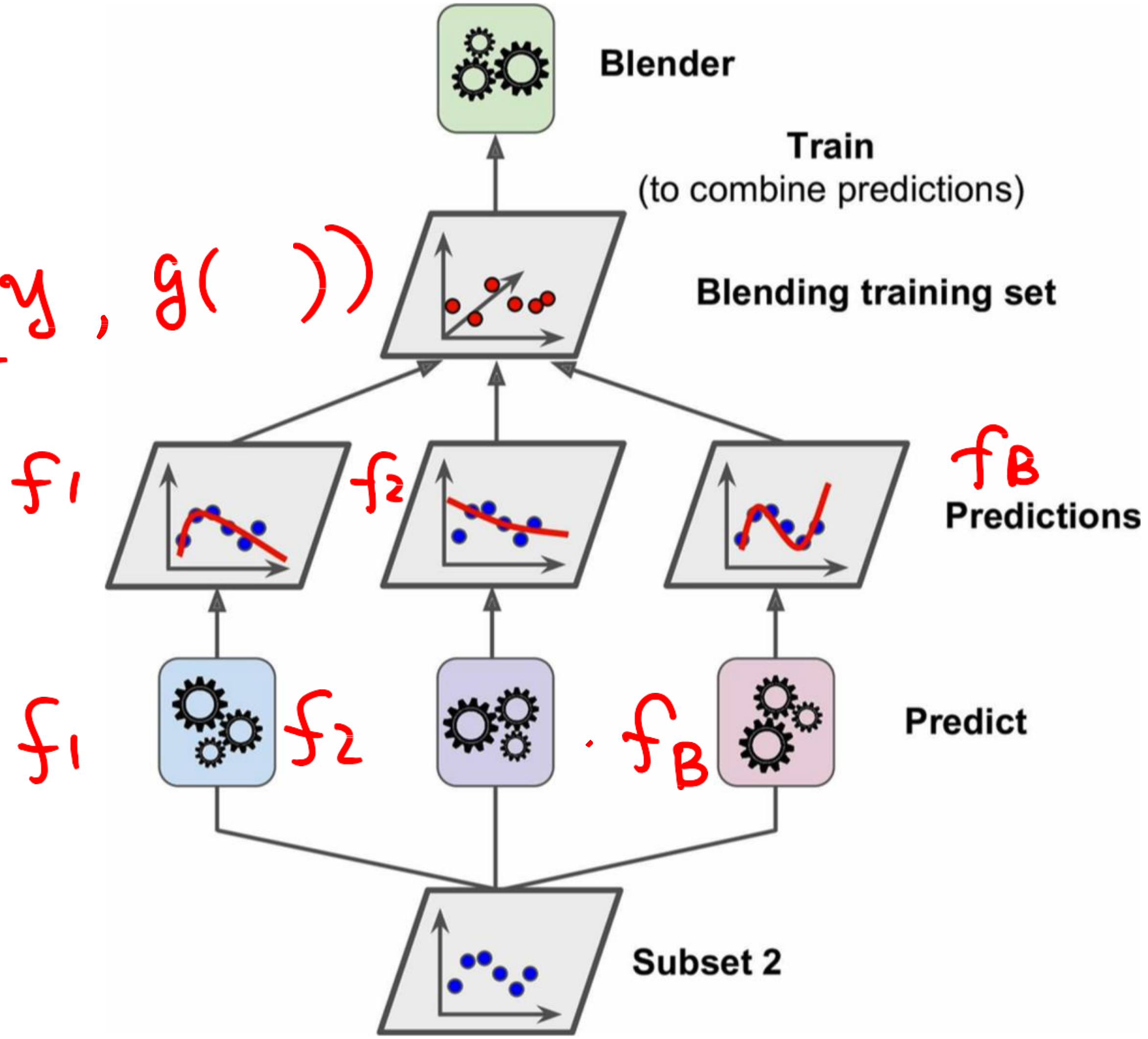
Training the base predictors



(Hold out for later)

Training the meta blender

loss(y, g( ))



In Lab session

You will see how to use XGBoost to do price prediction for houses in Boston

This can be useful for your **FINAL** project

Lab is done by Remy Belmonte