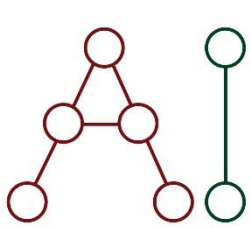


Trees, Weak Learners, and Ensemble Learning

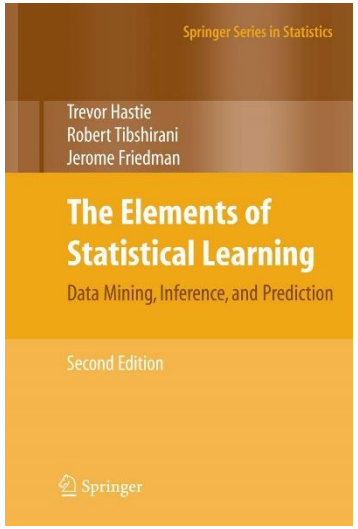
Assoc. Prof. Karl Ezra Pilario, Ph.D.
Process Systems Engineering Laboratory
Department of Chemical Engineering
University of the Philippines Diliman

© 2023 Karl Ezra S. Pilario, PhD.

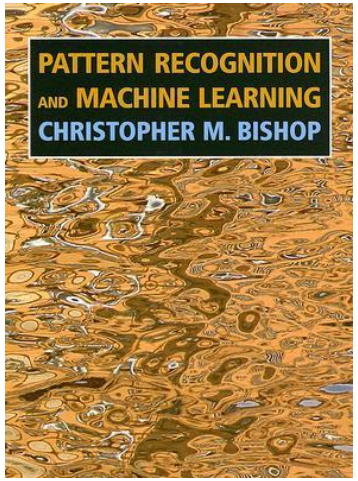
Outline

- Weak Learners
  - Naïve Bayes
  - k-Nearest Neighbors
  - Decision Trees
- Ensemble Learning
  - Bagging
  - Boosting
  - Stacking

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.

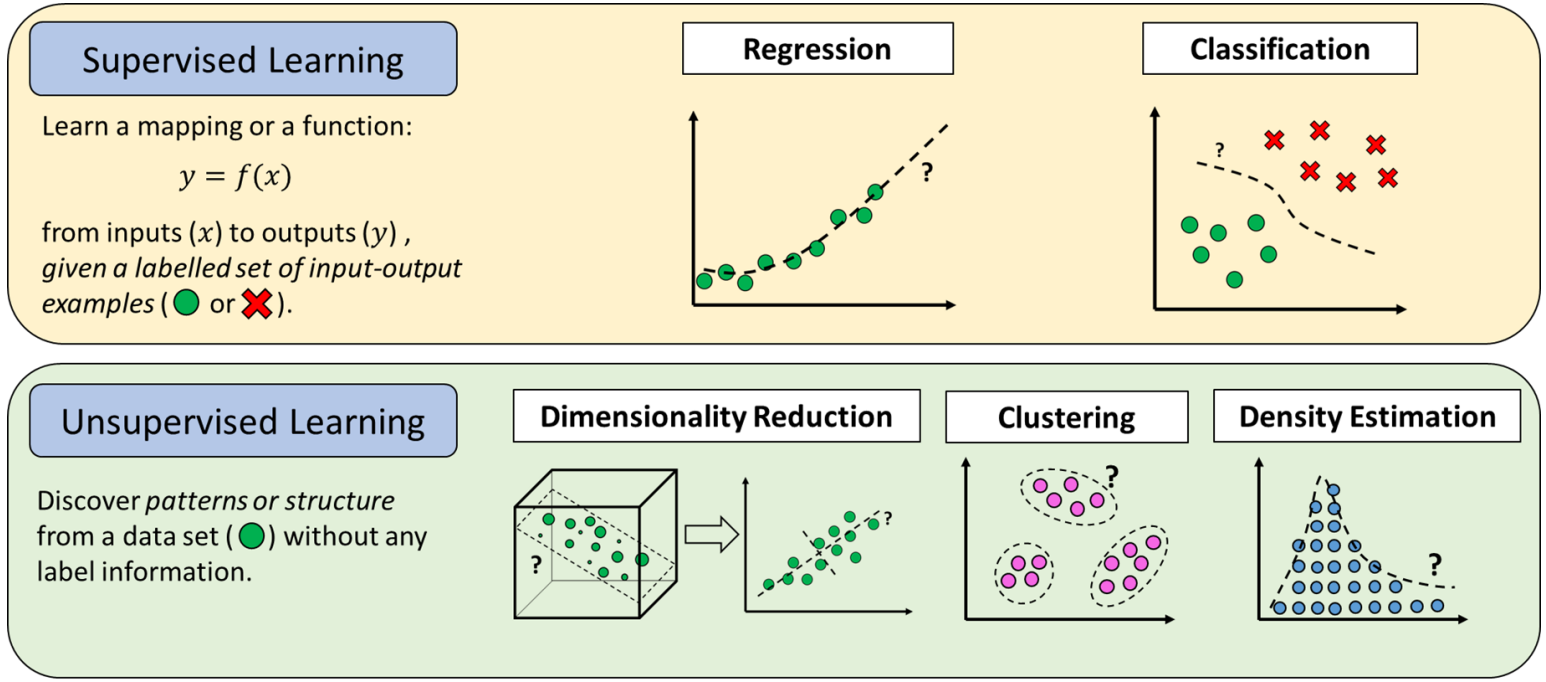


Bishop (2006)
Pattern Recognition and Machine Learning. Springer.



© 2023 Karl Ezra S. Pilario, PhD.

Weak Learners vs. Strong Learners



Among **supervised learning** models[1]:

- **Weak Learners** are models that perform slightly better than *random guessing*.
  - e.g. Naïve Bayes Classifier, k-Nearest Neighbors, Decision Trees
- **Strong Learners** are models that have arbitrarily *good accuracy*.
  - e.g. Support Vector Machine, Neural Networks, Logistic Regression

[1]**Source:** https://machinelearningmastery.com/strong-learners-vs-weak-learners-for-ensemble-learning

© 2023 Karl Ezra S. Pilario, PhD.

Naïve Bayes

- For classification
- *Naively* assumes that each feature is independent of each other. In general, this is not true.
- *Maximum a posteriori* decision rule:

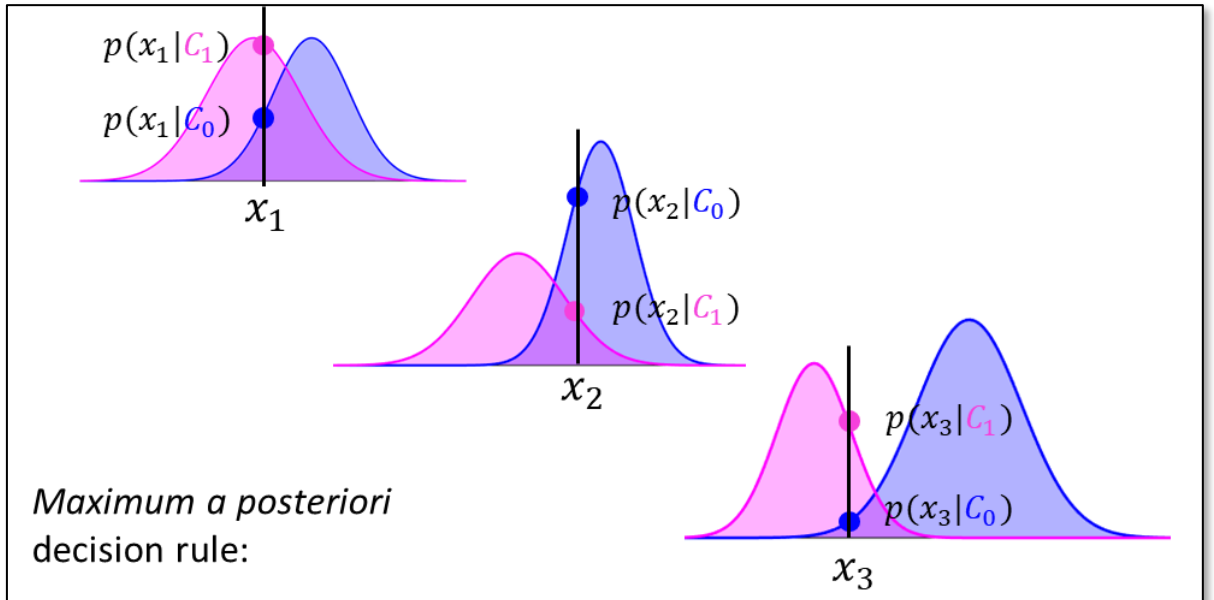
$$y = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(\boldsymbol{x}_i | C_k)$$

$p(C_k)$ = class prior; can be assumed 1/K, i.e. all classes are equally probable to occur.

$p(\boldsymbol{x}_i | C_k)$ = likelihood; how likely it is that $\boldsymbol{x}_i$ belongs to class C_k
= can be assumed as *Gaussian distributed* or a *kernel density estimate*.

n = no. of features (assumed independent)
K = no. of classes

	Feature 1	Feature 2	Feature 3	Class
Sample 1	1.1	0.5	8.1	1
Sample 2	3.2	1.4	2.5	0
Sample 3	5.2	2.2	0.7	1
Sample 4	0.4	9.4	0.8	0
…				



© 2023 Karl Ezra S. Pilario, PhD.

K-Nearest Neighbors

- For classification or regression
- *Instance-based, memory-based, model-free*
- **Algorithm:**
  1. Specify k (hyper-parameter).
  2. Calculate the k-nearest neighbors of a test sample.
  3. Predict the label:

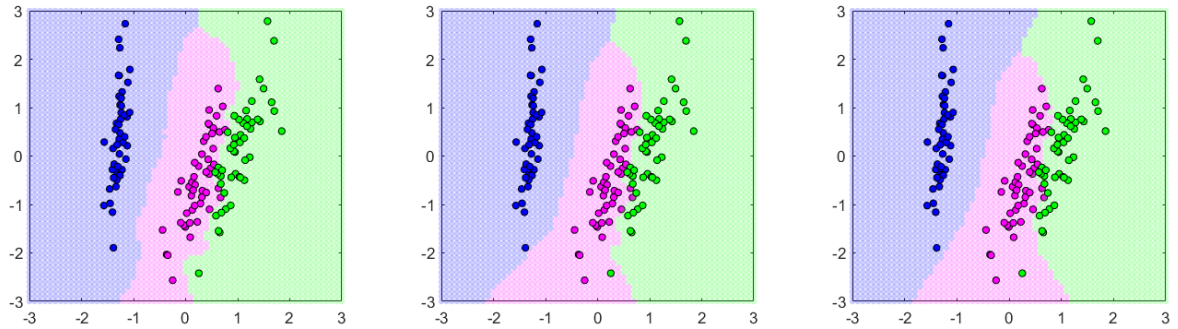
     For classification: $y = \text{mode}(y_i), \quad i \in \mathcal{N}_{\boldsymbol{x}}$

     For regression: $y = \text{mean}(y_i), \quad i \in \mathcal{N}_{\boldsymbol{x}}$

     where $\mathcal{N}_{\boldsymbol{x}}$ is the set of indices of the k nearest neighbors of $\boldsymbol{x}$.

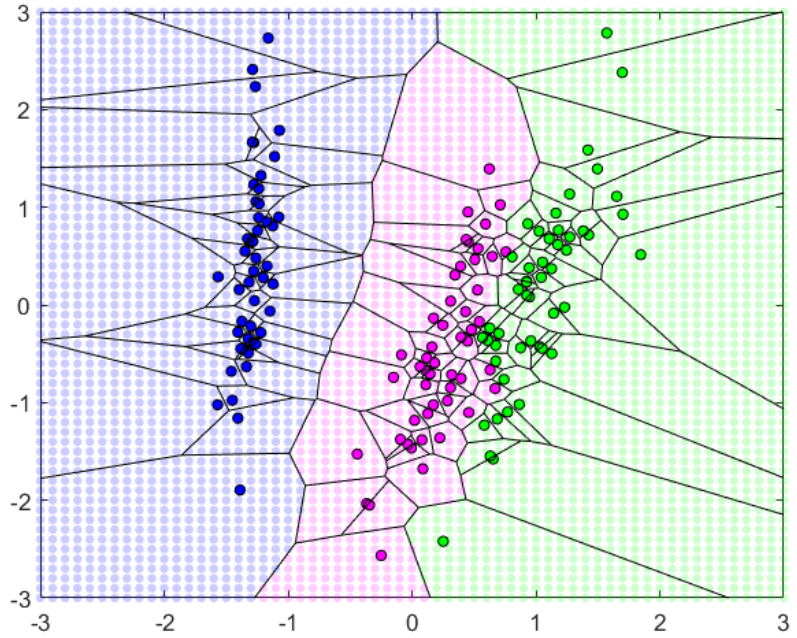
- The "nearest" neighbors can be defined by any metric, not just Euclidean distance.

kNN classification of 2-D Fisher Iris Data after PCA:



$K = 1$ $K = 5$ $K = 10$

The 1-NN result is equivalent to the Voronoi tessellation of the data points.

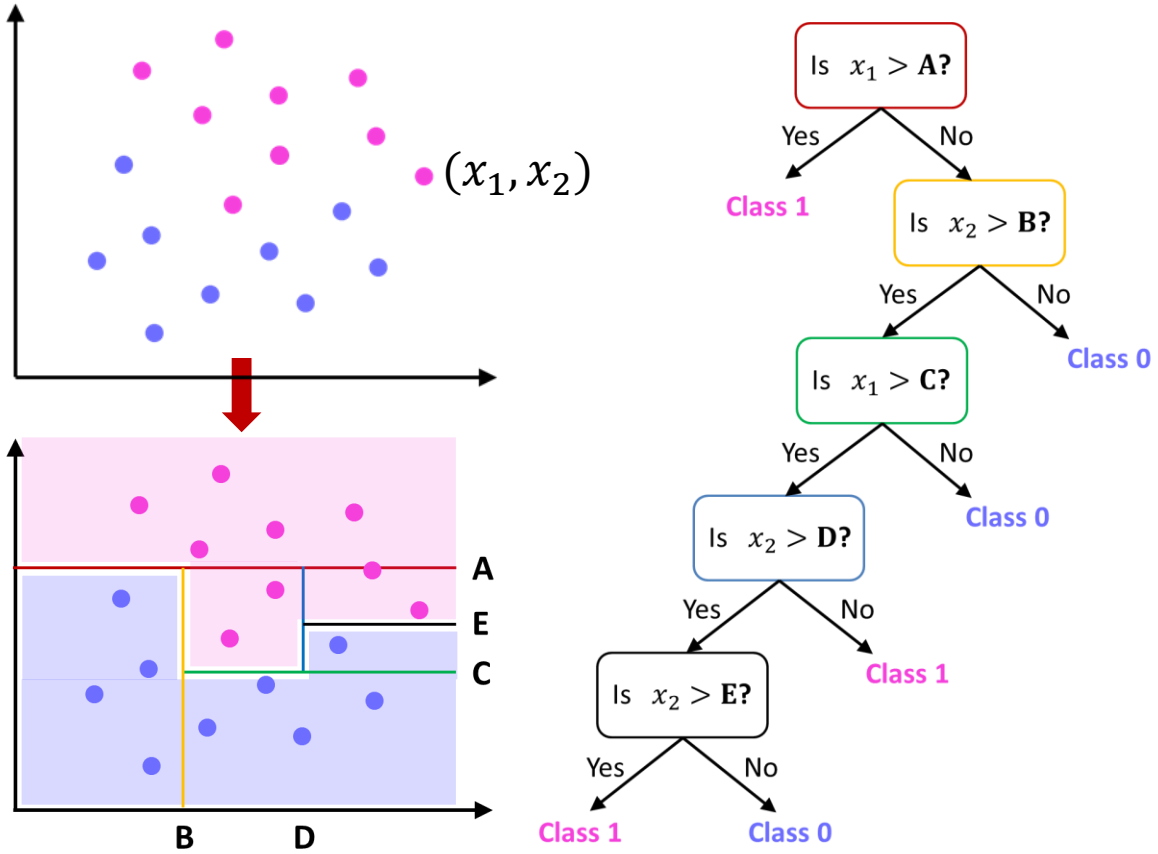


© 2023 Karl Ezra S. Pilario, PhD.

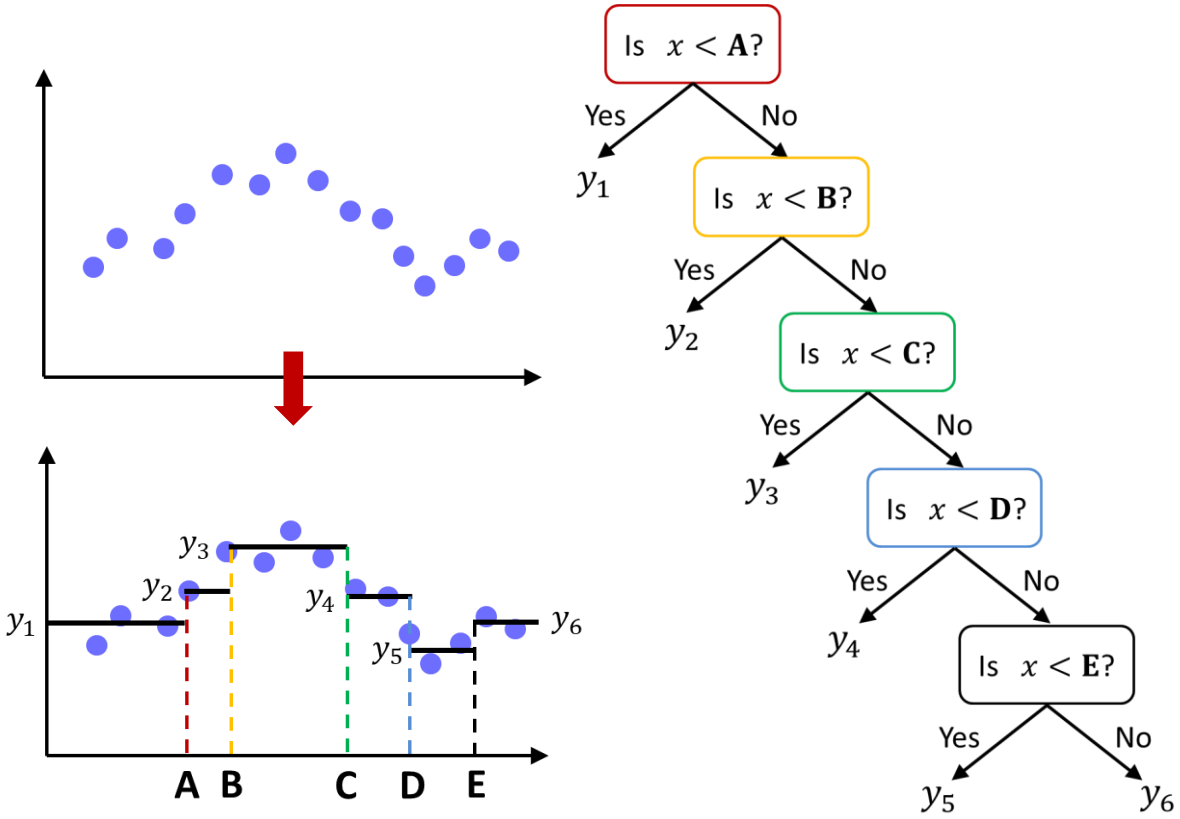
Decision Trees

- *Decision Trees* –decision rules arranged as a binary tree, where each branch leads to a different outcome.

Classification



Regression



© 2023 Karl Ezra S. Pilario, PhD.

Decision Trees: CART

Breiman et al. (1984)

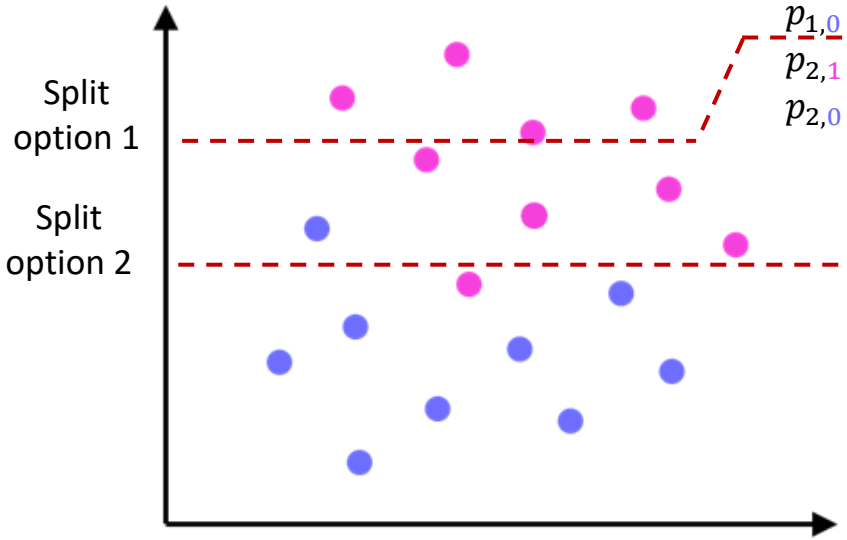
- Recursive Binary Splitting
  - To decide where to split, various measures of *node impurity* can be used:
    - Misclassification error
    - Gini index
    - Cross-entropy or deviance
  - A split is made where the node impurity is minimum, i.e. we desire a *homogeneous* node.

$$p_{m,k} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

$p_{m,k}$ = proportion of class k samples in node m
N_m = number of samples at region R_m
$I(\cdot)$ = indicator function:

$$I(y_i = k) = \begin{cases} 1, & y_i = k \\ 0, & y_i \neq k \end{cases}$$

From our example:



	Measures of Node Impurity		
	Misclassification error	**Gini Index**	**Cross-entropy**
	$1 - \max(p, 1-p)$	$2p(1-p)$	$-p \log_2 p$ $-(1-p)\log_2(1-p)$
Split option 1			
$p_{1,1} = 100\%$, $p_{1,0} = 0\%$ (Class 1)	0	0	0
$p_{2,1} = 35.7\%$, $p_{2,0} = 64.3\%$ (Class 0)	0.357	0.459	0.940
	0.278*	0.357	0.731
Split option 2			
$p_{1,1} = 88.9\%$, $p_{1,0} = 11.1\%$ (Class 1)	0.111	0.197	0.503
$p_{2,1} = 11.1\%$, $p_{2,0} = 88.9\%$ (Class 0)	0.111	0.197	0.503
	0.111	0.197	0.503

*Weighted average of the node impurity per child node:
e.g. 0.278 = (**0***4 + **0.357***14)/18

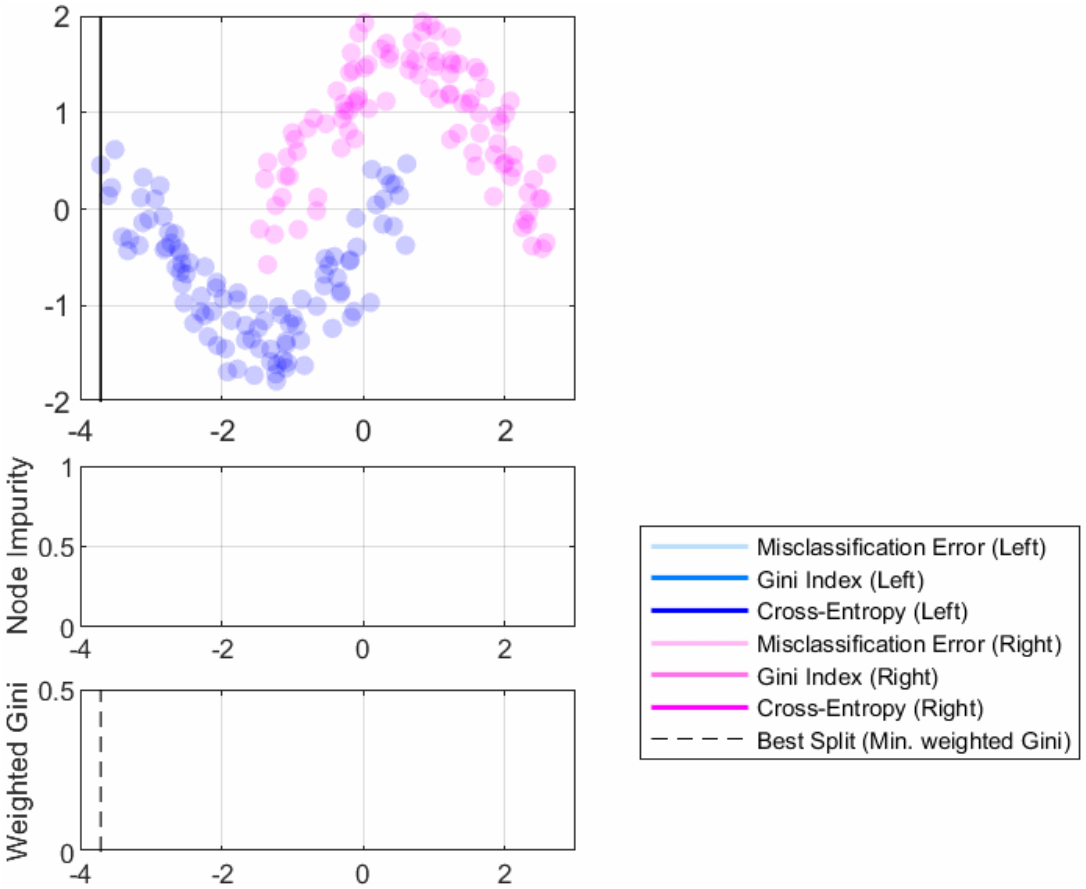
© 2023 Karl Ezra S. Pilario, PhD.

Decision Trees: CART

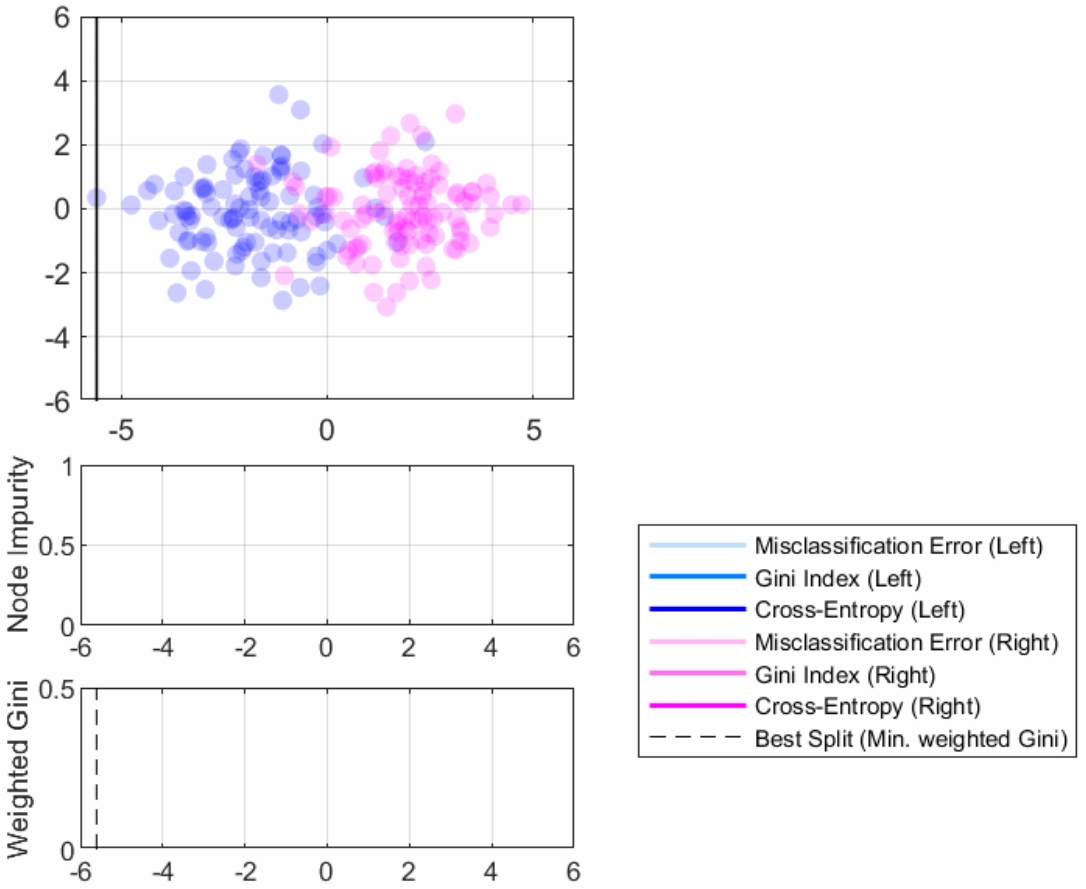
Breiman et al. (1984)

An illustration of how CART finds splits:

Two Moons Data Set



Two Gaussians Data Set



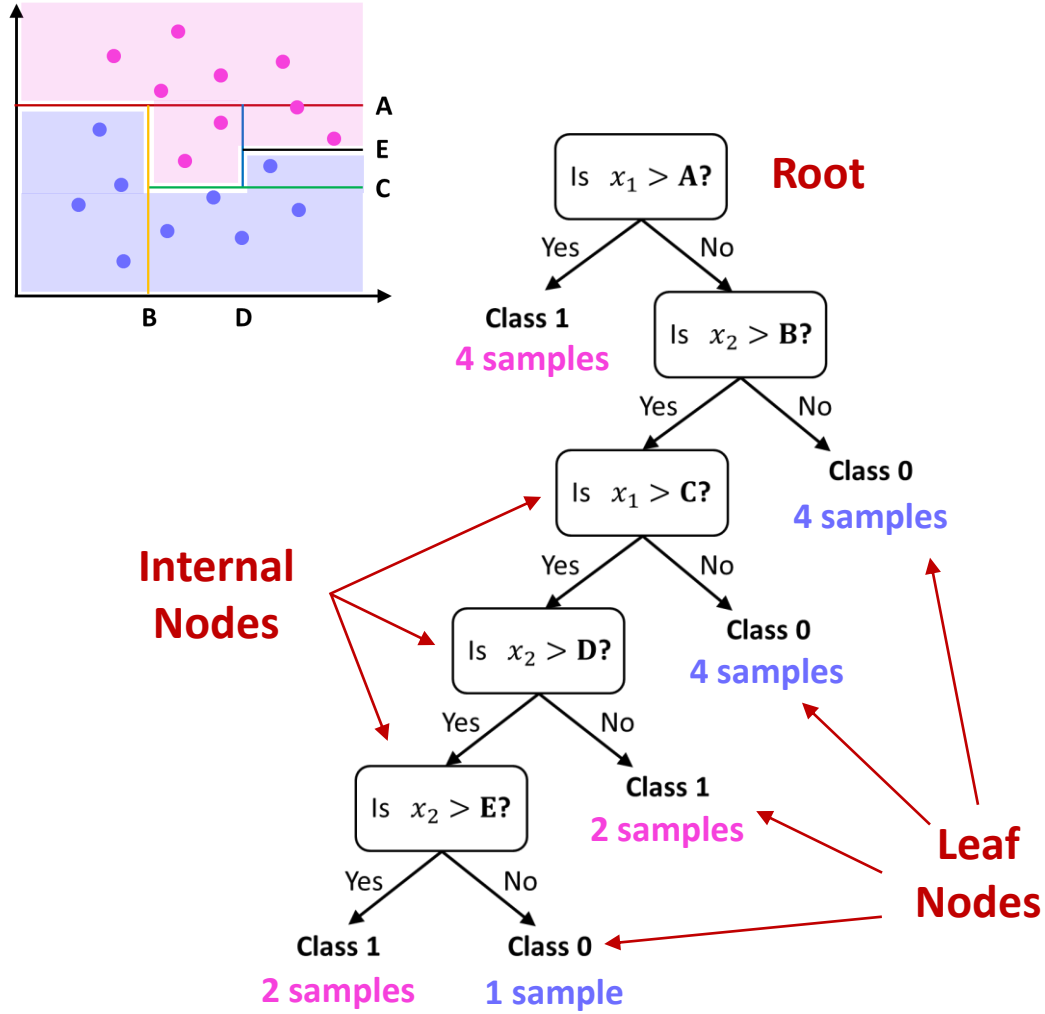
© 2023 Karl Ezra S. Pilario, PhD.

Decision Trees: CART

Breiman et al. (1984)

Common CART (Decision Tree) Hyper-parameters

- **Criterion:** **Classifier:** Gini, Entropy, Log-loss
  Regressor: squared_error, friedman_mse, absolute error, poisson
- **Splitter:** Best: best split (minimum impurity) is chosen
  Random: best split among random splits
- **Max_depth:** maximum depth of tree
  None = tree is grown until all leaves are pure
- **Min_samples_split:** Minimum number of samples required to split an internal node.
- **Min_samples_leaf:** Minimum number of samples required to be at a leaf node.
- **Max_features:** Maximum number of features to consider when looking for a best split.



© 2023 Karl Ezra S. Pilario, PhD.

Decision Trees

Comparison between different Decision Tree algorithms:

	Classification and Regression Trees	*Iterative Dichotomiser 3*	*A version 4.5 of a program written in C??*
	CART	**ID3**	**C4.5**
	Breiman et al. (1984)	*Quinlan (1986)*	*Quinlan (1993)*
Main splitting criteria	Gini Index	Information Gain	Gain Ratio
Continuous variables?	Yes	No	Yes
Missing values?	Yes	No	Yes
Pruning?	Yes	No	Yes
Time Complexity	$O(n * m \log(m))$	$O(n*m^2)$	$O(n * m \log(m))$

sklearn.tree.DecisionTreeClassifier
sklearn.tree.DecisionTreeRegressor

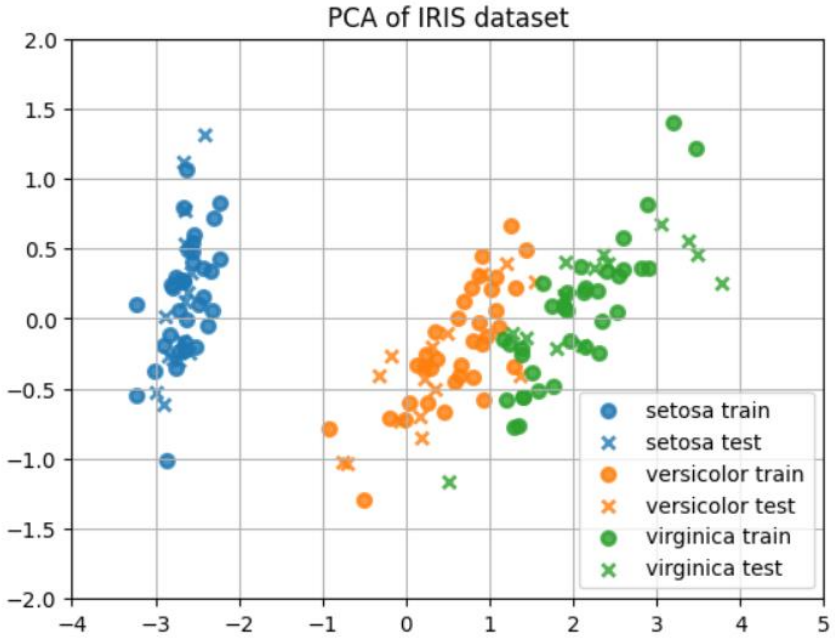
© 2023 Karl Ezra S. Pilario, PhD.

Weak Learners

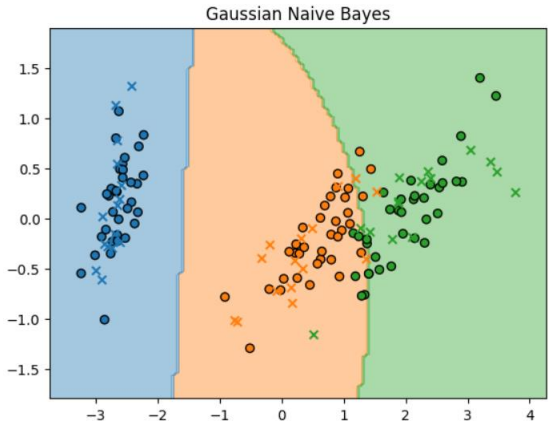
Example:

Perform classification on the following 2-D Fisher Iris Data Set after PCA projection. Use the following weak learners:

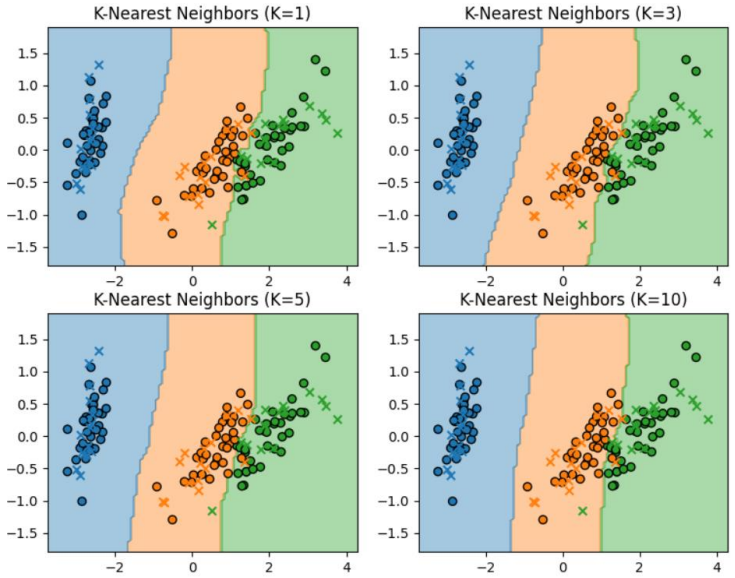
- Gaussian Naïve Bayes Classifier
- K-Nearest Neighbors Classifier
- Decision Tree Classifier



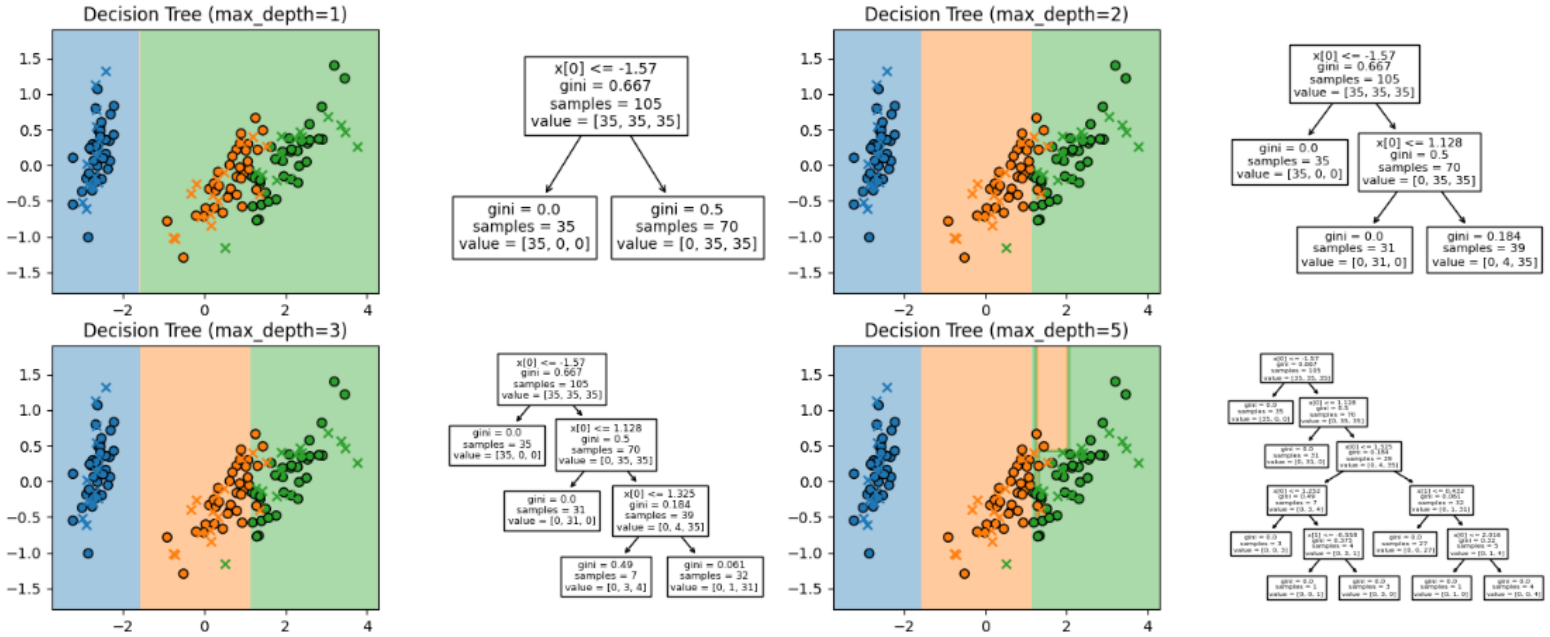
Gaussian Naïve Bayes Classifier



K-nearest Neighbors Classifier



Decision Tree Classifier

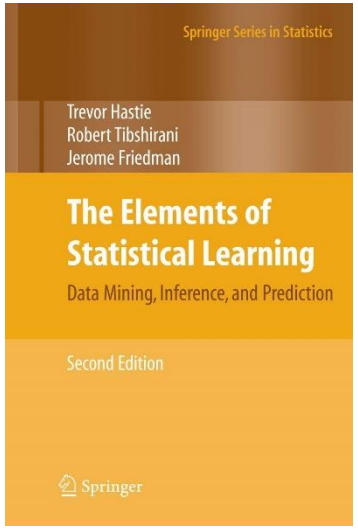


© 2023 Karl Ezra S. Pilario, PhD.

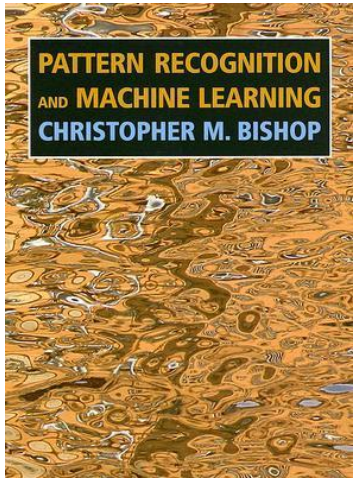
Outline

- Weak Learners
  - Naïve Bayes
  - k-Nearest Neighbors
  - Decision Trees
- **Ensemble Learning**
  - Bagging
  - Boosting
  - Stacking

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.



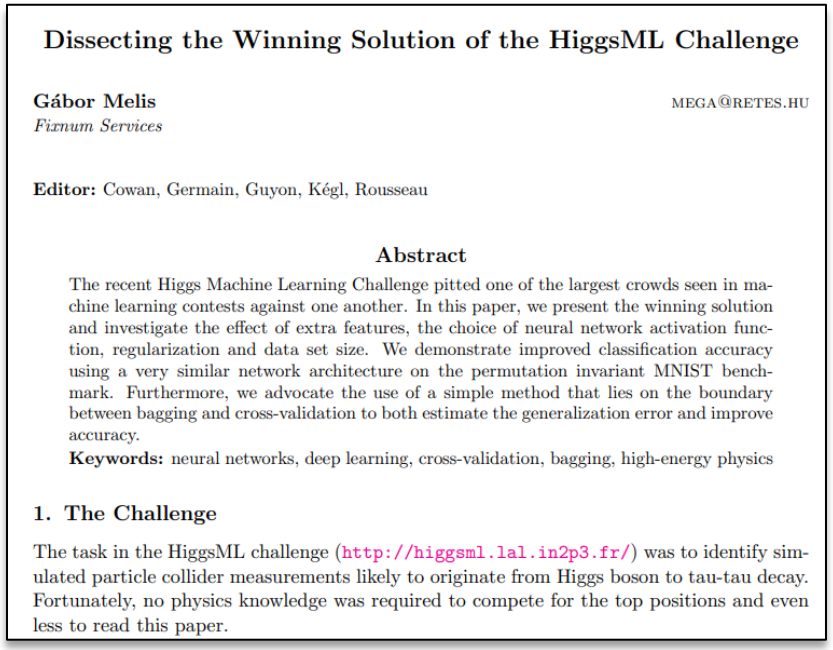
Bishop (2006)
Pattern Recognition and Machine Learning. Springer.



© 2023 Karl Ezra S. Pilario, PhD.

Ensemble Learning

- Combine several weak learners into one strong learner.
- Ensemble learning is the winning solution in most data science competitions.



Dissecting the Winning Solution of the HiggsML Challenge

Gábor Melis MEGA@RETES.HU
Fixnum Services

Editor: Cowan, Germain, Guyon, Kégl, Rousseau

Abstract

The recent Higgs Machine Learning Challenge pitted one of the largest crowds seen in machine learning contests against one another. In this paper, we present the winning solution and investigate the effect of extra features, the choice of neural network activation function, regularization and data set size. We demonstrate improved classification accuracy using a very similar network architecture on the permutation invariant MNIST benchmark. Furthermore, we advocate the use of a simple method that lies on the boundary between bagging and cross-validation to both estimate the generalization error and improve accuracy.

Keywords: neural networks, deep learning, cross-validation, bagging, high-energy physics

1. The Challenge

The task in the HiggsML challenge (http://higgsml.lal.in2p3.fr/) was to identify simulated particle collider measurements likely to originate from Higgs boson to tau-tau decay. Fortunately, no physics knowledge was required to compete for the top positions and even less to read this paper.

- Melis (2015)
- Task: Classification

- Winner: Ensemble of 70 neural networks with 1.1 million parameters per network

The BigChaos Solution to the Netflix Grand Prize

Andreas Töscher and Michael Jahrer

commendo research & consulting
Neuer Weg 23, A-8580 Köflach, Austria
{andreas.toescher,michael.jahrer}@commendo.at

Robert M. Bell*

AT&T Labs - Research
Florham Park, NJ

September 5, 2009

1 Introduction

The team *BellKor's Pragmatic Chaos* is a combined team of *BellKor*, *Pragmatic Theory* and *BigChaos*. *BellKor* consists of Robert Bell, Yehuda Koren and Chris Volinsky. The members of *Pragmatic Theory* are Martin Piotte and Martin Chabbert. Andreas Töscher and Michael Jahrer form the team *BigChaos*. *BellKor* won the Progress Prize 2007 [4]. The Progress Prize 2008 was won by the combined efforts of *BellKor* and *BigChaos* [5][17].

The documentation of the Netflix Grand Prize consists of three parts. In this document we focus on the contribution of *BigChaos* to the combined Grand Prize Solution.

The document is organized as follows: In Section 2 we describe the Netflix dataset and important statistical properties, followed by a detailed explanation of the training procedure of our predictors in Section 3. Section 4 defines the notation, which we use throughout this document. The algorithmic

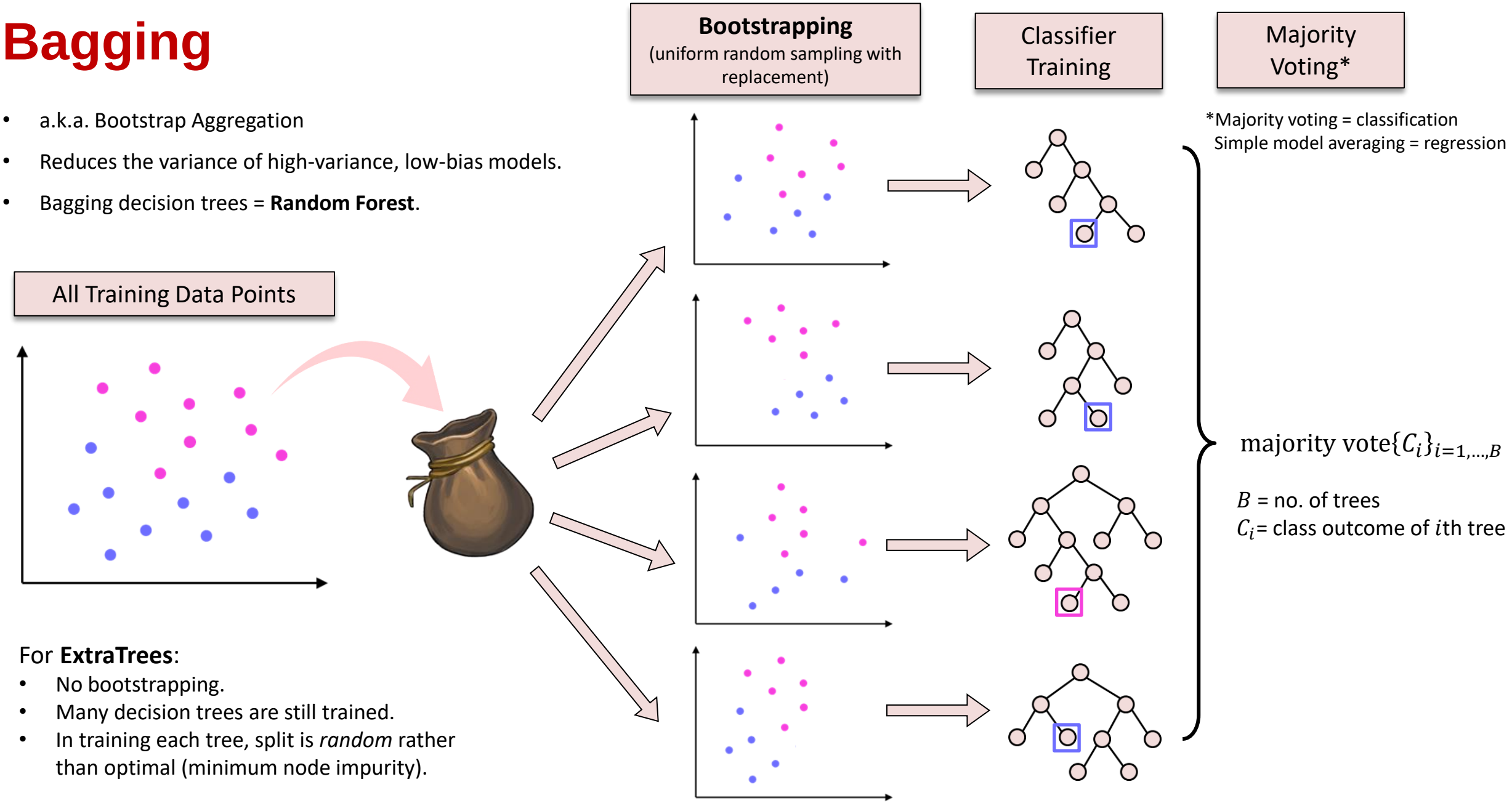
- Toscher and Jahrer (2009)
- Task: Regression

- Winner: Ensemble of 100 learners

© 2023 Karl Ezra S. Pilario, PhD.

Bagging

- a.k.a. Bootstrap Aggregation
- Reduces the variance of high-variance, low-bias models.
- Bagging decision trees = **Random Forest**.



*Majority voting = classification
Simple model averaging = regression

$$\text{majority vote}\{C_i\}_{i=1,\ldots,B}$$

B = no. of trees
C_i = class outcome of ith tree

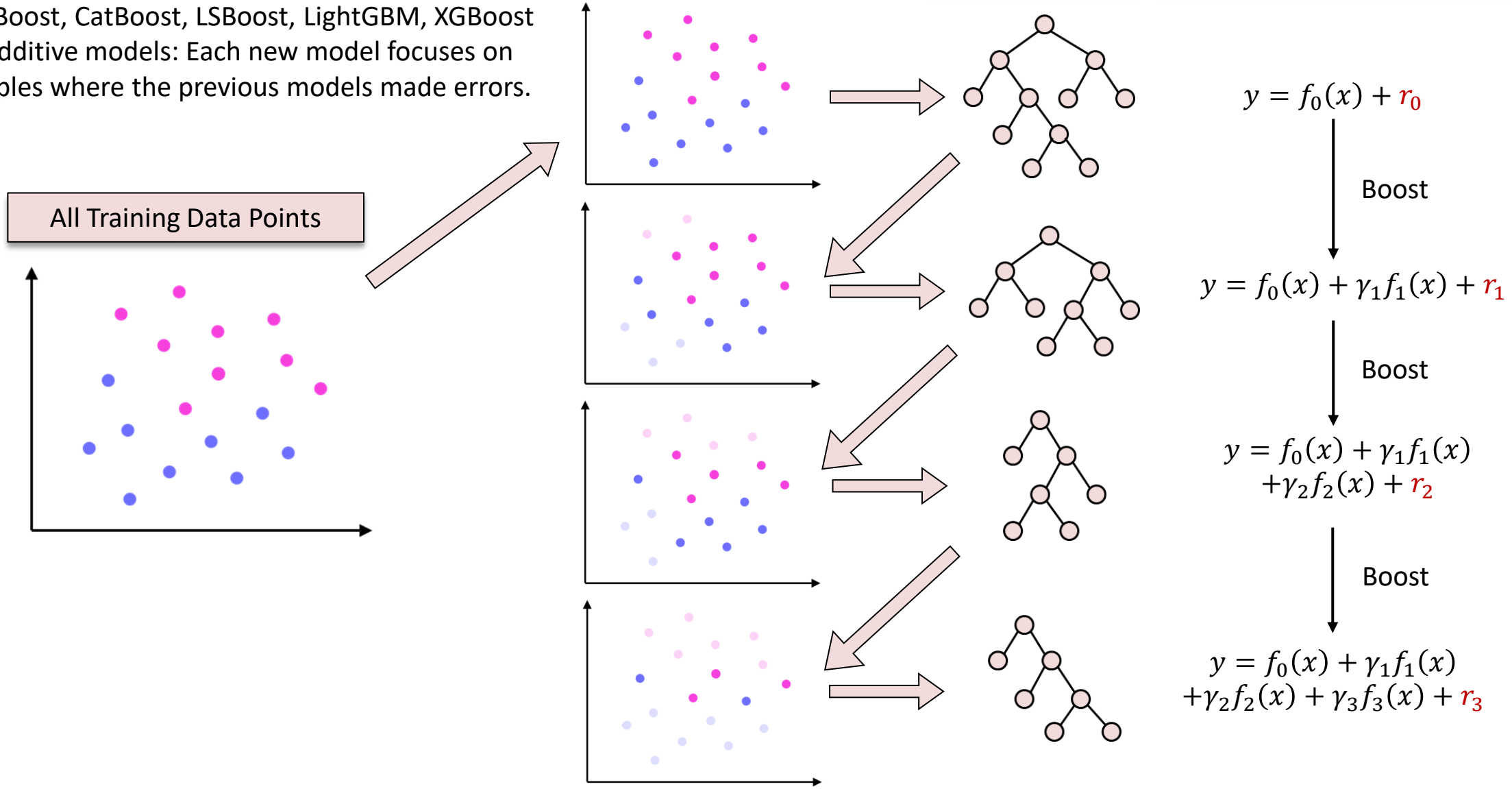
For **ExtraTrees**:

- No bootstrapping.
- Many decision trees are still trained.
- In training each tree, split is *random* rather than optimal (minimum node impurity).

© 2023 Karl Ezra S. Pilario, PhD.

Boosting

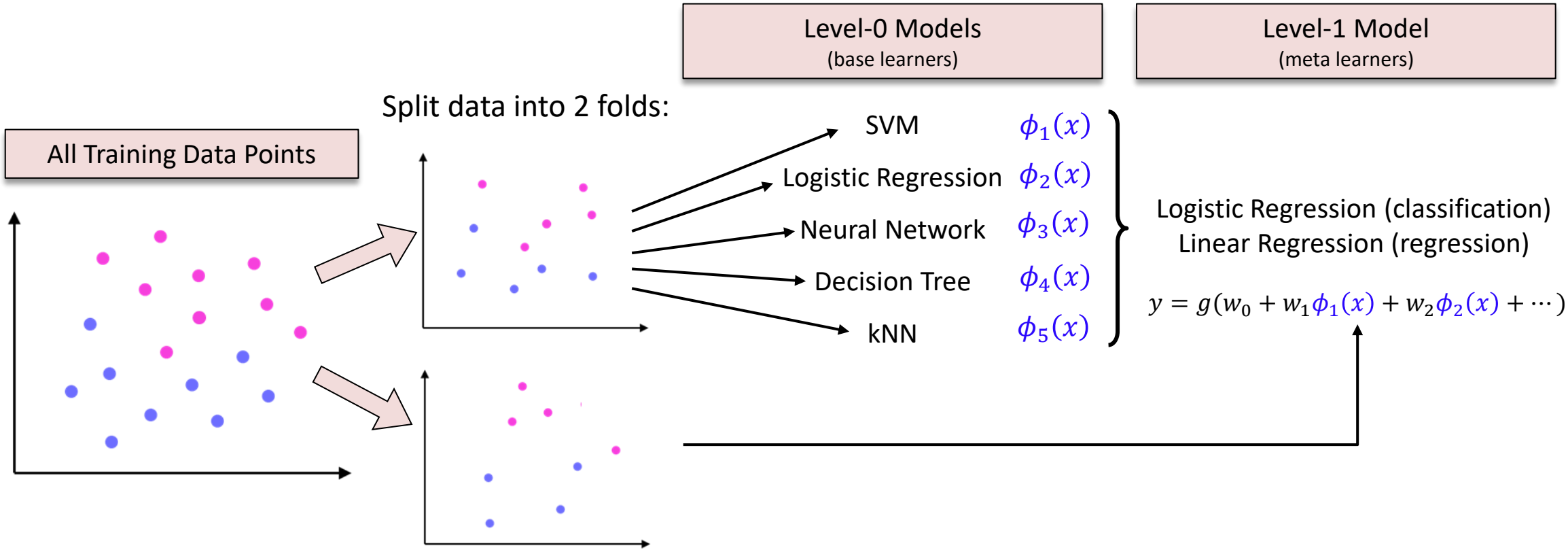
- AdaBoost, CatBoost, LSBoost, LightGBM, XGBoost
- Fit additive models: Each new model focuses on samples where the previous models made errors.



© 2023 Karl Ezra S. Pilario, PhD.

Stacking

- Meta-modeling: Build a model on top of different models.
- Similar to the **linear basis function model** in our previous lecture: $y = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_m\phi_m(x)$

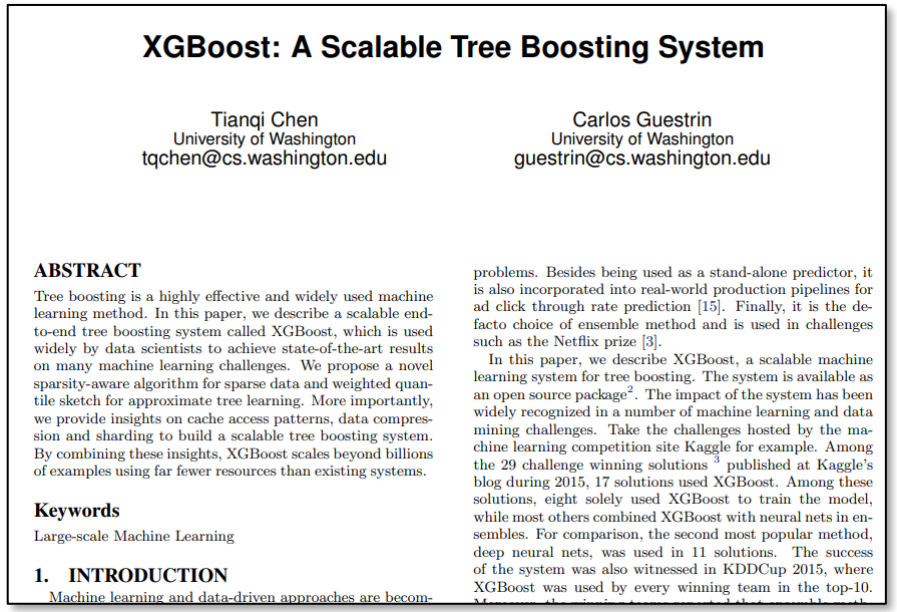


© 2023 Karl Ezra S. Pilario, PhD.

XGBoost

Chen et al. (2006)

- Extreme Gradient Boosting
- **Parallelized** and **carefully optimized** gradient boosting.
- Can be tuned using regularization: L1, L2, etc.
- XGBoost is **scalable**: It can process enormous amounts of data with the help of distributed servers (Hadoop, Spark, etc.)



XGBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

ABSTRACT

Tree boosting is a highly effective and widely used machine learning method. In this paper, we describe a scalable end-to-end tree boosting system called XGBoost, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. We propose a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning. More importantly, we provide insights on cache access patterns, data compression and sharding to build a scalable tree boosting system. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems.

Keywords

Large-scale Machine Learning

1. INTRODUCTION

Machine learning and data-driven approaches are becom-

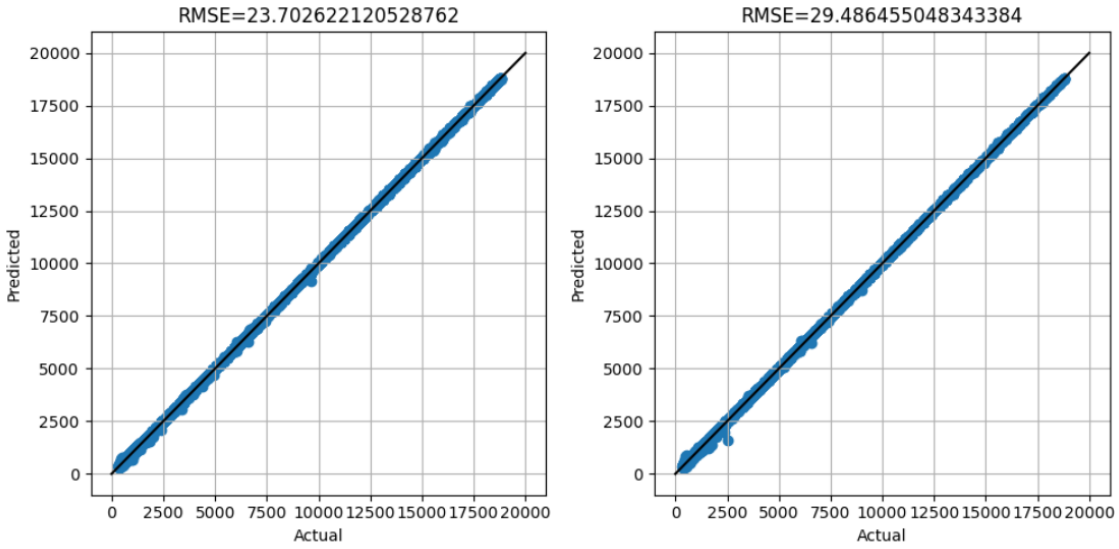
problems. Besides being used as a stand-alone predictor, it is also incorporated into real-world production pipelines for ad click through rate prediction [15]. Finally, it is the de-facto choice of ensemble method and is used in challenges such as the Netflix prize [3].

In this paper, we describe XGBoost, a scalable machine learning system for tree boosting. The system is available as an open source package[2]. The impact of the system has been widely recognized in a number of machine learning and data mining challenges. Take the challenges hosted by the machine learning competition site Kaggle for example. Among the 29 challenge winning solutions [3] published at Kaggle's blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method, deep neural nets, was used in 11 solutions. The success of the system was also witnessed in KDDCup 2015, where XGBoost was used by every winning team in the top-10.

Example:

Use XGBoost Regression with 100 decision trees to predict the price of diamonds in the Diamonds Data Set.

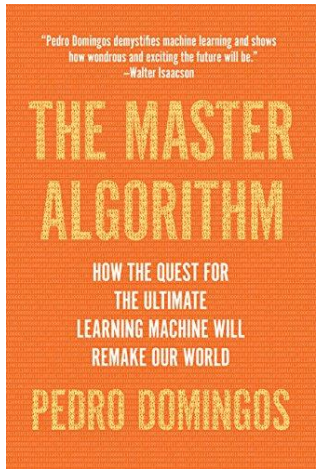
	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75



© 2023 Karl Ezra S. Pilario, PhD.

A Perspective on ML Algorithms

Pedro Domingos (2015)



The Five Tribes of Machine Learning

	Symbolists	Connectionists	Analogizers	Bayesians	Evolutionaries
ORIGINS	Logic, Philosophy	Neuroscience	Psychology	Statistics	Evolutionary Biology
MAIN ALGORITHMS	Decision Trees, Random Forests	Neural Nets, Deep Learning	SVM, kNN, Kernel Machines	Gaussian Processes, Graphical Models	Genetic Programming
PIONEERS	• Tom Mitchell • Steve Muggleton • Ross Quinlan	• Yann LeCun • Geoff Hinton • Yoshua Bengio	• Vladimir Vapnik • Peter Hart • Douglas Hofstadter	• David Heckerman • Judea Pearl • Michael Jordan	• John Holland • John Koza • Hod Lipson

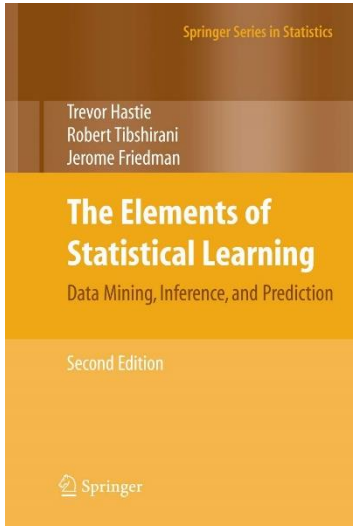
A universal learner, if it exists, must be some combination of these.

© 2023 Karl Ezra S. Pilario, PhD.

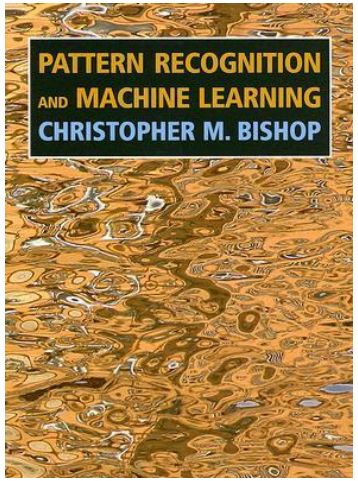
Outline

- Weak Learners
  - Naïve Bayes
  - k-Nearest Neighbors
  - Decision Trees
- **Ensemble Learning**
  - Bagging
  - Boosting
  - Stacking

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.



Bishop (2006)
Pattern Recognition and Machine Learning. Springer.



© 2023 Karl Ezra S. Pilario, PhD.

Further Reading

- https://github.com/benedekrozemberczki/awesome-gradient-boosting-papers
- https://github.com/the-black-knight-01/Data-Science-Competitions
- https://www.datacamp.com/tutorial/xgboost-in-python
- Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2006; ACM Press: New York, NY, USA, 2016; Volume 19, pp. 785–794.
- Melis, G. Dissecting the Winning Solution of the HiggsML Challenge. J. Mach. Learn. Res. Work. Conf. Proc. 2015, 42, 57–67.
- https://neptune.ai/blog/xgboost-everything-you-need-to-know
- Lessons from Kaggle Competitions: https://www.kdnuggets.com/2015/12/harasymiv-lessons-kaggle-machine-learning.html
- Super Learner: https://www.degruyter.com/document/doi/10.2202/1544-6115.1309/html
- https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/

© 2023 Karl Ezra S. Pilario, PhD.