Data Mining

Predictive Modelling

Ensembles

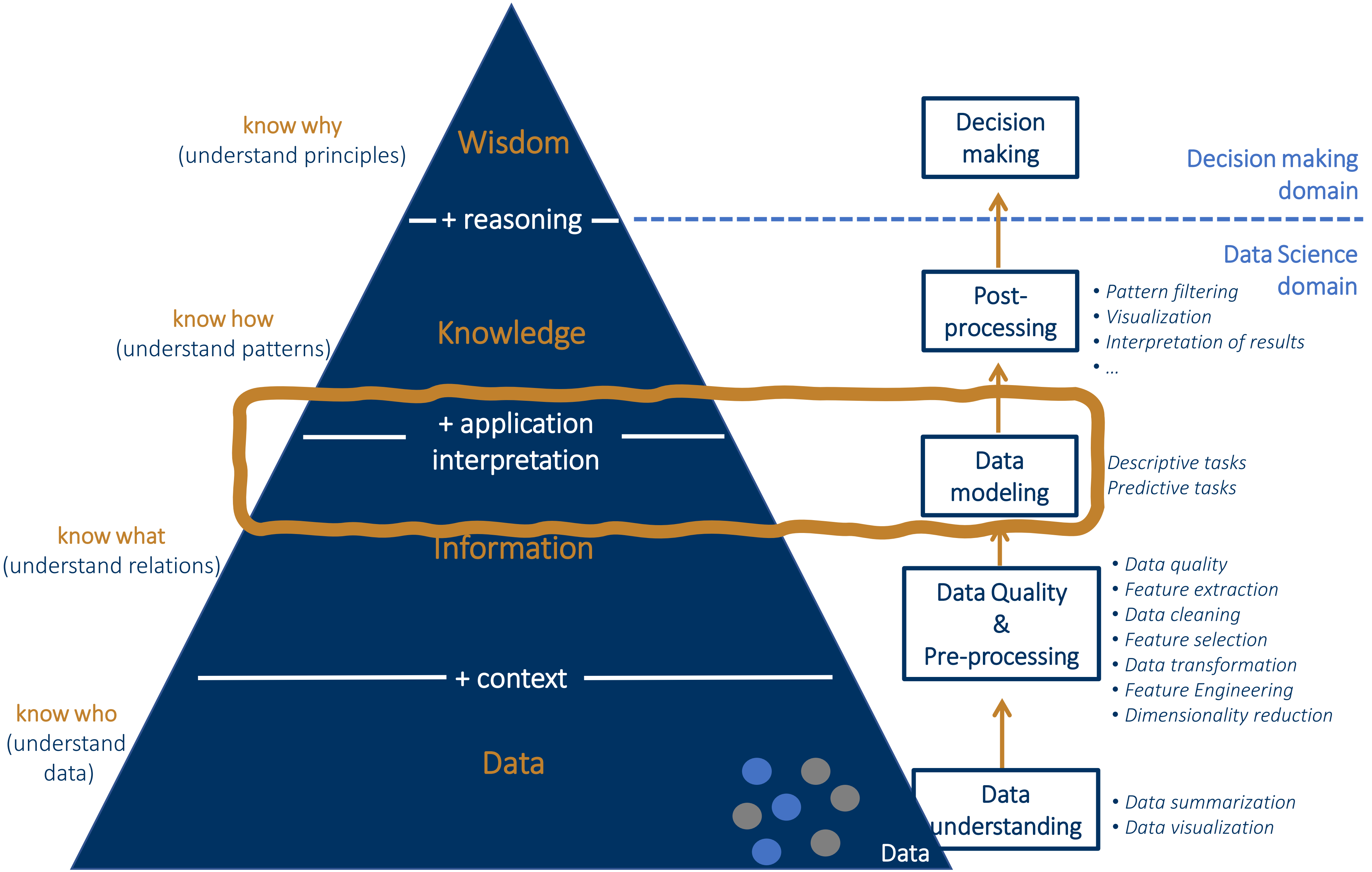
Raquel Sebastião

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

raquel.sebastiao@ua.pt

2022/2023


know why
(understand principles)
Wisdom
+ reasoning
Decision making
Decision making domain
Data Science domain
know how
(understand patterns)
Knowledge
Post-processing
• Pattern filtering
• Visualization
• Interpretation of results
• ...
+ application interpretation
Data modeling
Descriptive tasks
Predictive tasks
know what
(understand relations)
Information
Data Quality & Pre-processing
• Data quality
• Feature extraction
• Data cleaning
• Feature selection
• Data transformation
• Feature Engineering
• Dimensionality reduction
+ context
know who
(understand data)
Data
Data understanding
• Data summarization
• Data visualization
Data

Prediction Models – approaches

Geometric approaches

- **Distance-based:** kNN
- **Linear models:** Fisher's linear discriminant, perceptron, logistic regression, SVM (w. linear kernel)

Probabilistic approaches

- naive Bayes, logistic regression

Logical approaches

- classification or regression trees, rules

Optimization approaches

- neural networks, SVM

Sets of models (ensembles)

- random forests, adaBoost

Contents

Ensemble models

Why?

- For complex problems **it is hard to find a model that** “explains” **all observed data**

What?

- Ensembles are **collections of models** that are used **together** to solve a prediction problem

How?

- Construct a set of base models learned from different samples of the training data
- Use a **combination** of models to **increase accuracy**
    - **Predict** class label/value of new cases by combining the predictions made by multiple models (majority vote/averaging)

Ensemble classifiers

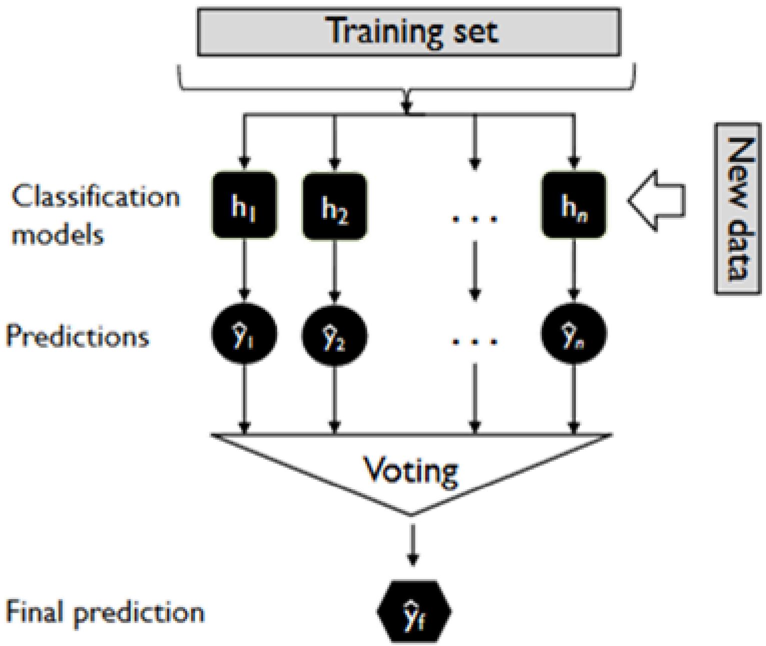


Image credits: Sebastian Raschka

Classification

- Majority vote of the classifiers
- Each classifier has an error rate better than chance. For binary classification:

$$\{\epsilon_1, \epsilon_2, \dots, \epsilon_n\}, \quad \epsilon_t < 0.5$$

If all classifiers are independent (errors are uncorrelated):

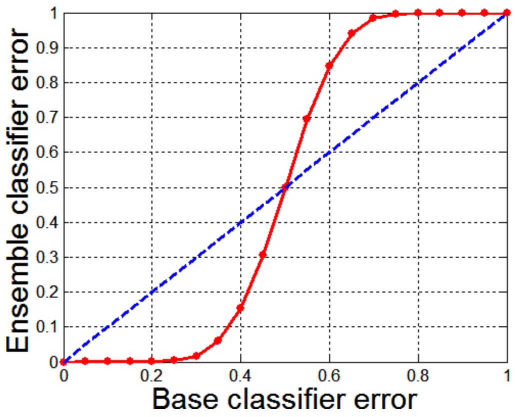
- Error rate of ensemble = probability of having more than half of base classifiers being wrong:

$$\epsilon_{ensemble} = \sum_{i=\lfloor n/2 \rfloor}^{n} \binom{n}{i} \epsilon^i (1-\epsilon)^{n-i}$$

Ensemble classifiers

An ensemble of classifiers improves over individual classifiers iif (Dietterich 2002):

- they perform better than random guess
- they have uncorrelated errors
- they commit errors in different regions of the instance space



Classification error for an ensemble of **25** base classifiers, assuming their errors are uncorrelated

Ensemble methods work best with unstable base classifiers

- Classifiers that are **sensitive to minor perturbations in training set**, due to *high model complexity*
- Examples: Unpruned decision trees, ANNs, …

Ensemble classifiers: construction

- **By manipulating the training set**
  - Homogeneous models (e.g.: Bagging, Boosting)
  - Heterogeneous models (e.g.: Cascading, Stacking)
- **By manipulating the input features**
  - e.g.: random forests
- **By manipulating the class labels**
  - E.g.: error-correcting output coding
- **By manipulating the learning algorithm**
  - Example: injecting randomness in the initial weights of ANN

Bias-variance trade-off

The **generalization error** of a model can be split in two main components:

- the bias and the variance components

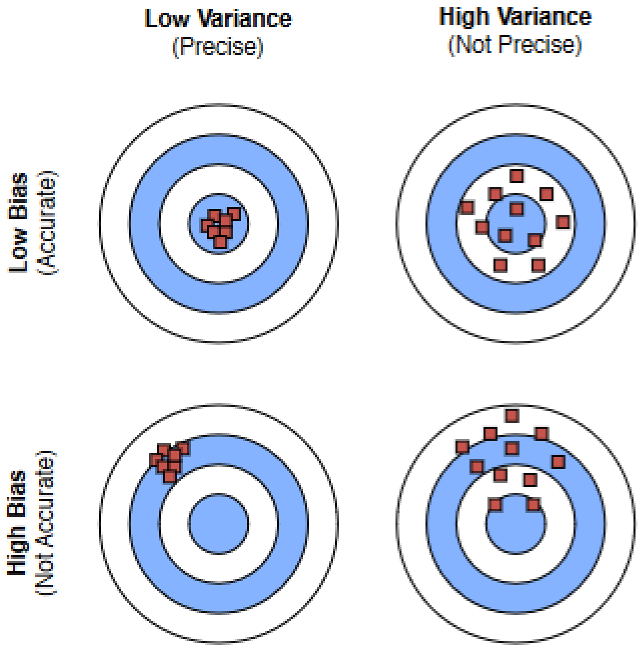


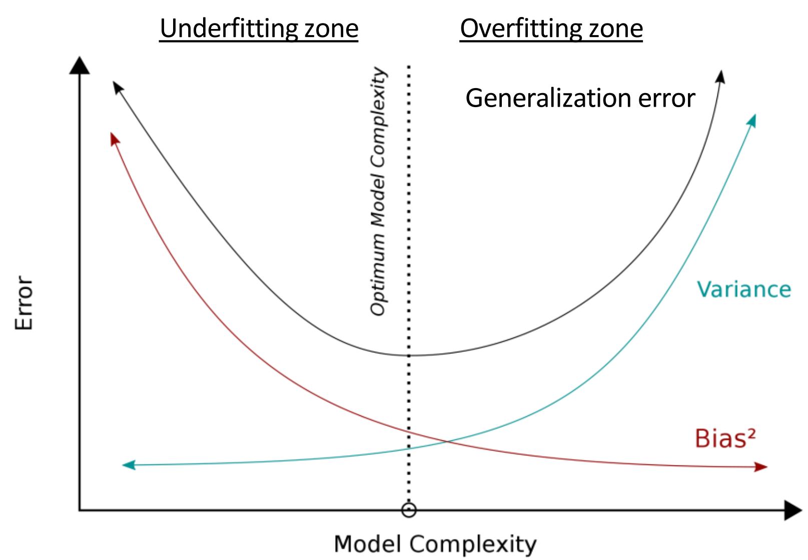
Image credits: Sebastian Raschka

- Bias: error that is due to the **poor ability of the model to fit the seen data**
- Variance: error related to the **sensibility of the model to the given training data**

Bias-variance trade-off

- Decreasing the bias by adjusting more to the training sample → higher variance: over-fitting

- Decreasing the variance by being less sensitive to the given training data → higher bias



en.wikipedia.org

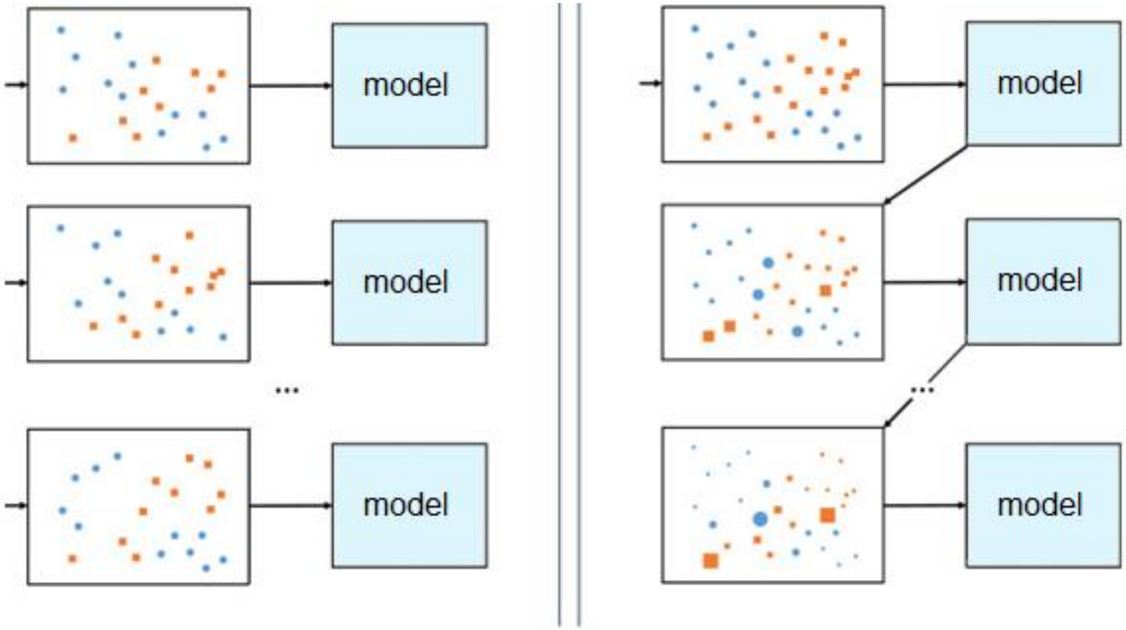
Ensembles can reduce both components of the generalization error!

Contents

- Ensembles
- Types of ensembles
  - Bagging
  - Boosting
- Ensembles methods
- Summary

Types of ensembles

- Independent or Parallel models
- Iterative or Sequential models



Independent or Parallel models **Iterative or Sequential models**

Sergio González, et al., A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities, Information Fusion, 64:205-237, 2020.

Types of ensembles

Independent or Parallel models:

- Models are trained in parallel
- Each model is trained with a bootstrap sample of the data set
- Gobal decision: the class is predicted by voting/averaging of the models

Iterative or Sequential models:

- Models are trained in sequence
- Each model is trained by emphasizing the training samples that previous models misclassified
  - Initially, all N samples are assigned equal weights
  - weights may change at the end of each boosting round
- Gobal decision: the class is predicted by voting/averaging but the models contributions depend in their performance

Types of ensembles: similarities & differences

Similarities

- Ensemble methods to get n models
- Generate several training data sets by random sampling
- Reduce variance and provide higher stability

Differences

- Bagging: they are built independently; Boosting tries to add new models that do well where previous models fail
- Boosting determines weights for the data to increase (boost) performance for the most difficult examples
- Gobal decision – Bagging: **equally weighted voting/averaging of the models**; Boosting: **voting/averaging depending on models's performance** (models with better performance on training data contribute more in predicting)
- **Boosting tries to reduce bia**s
- **Bagging may solve the over-fitting** problem, while **Boosting can increase it**

Contents

Ensembles methods

- **Bagging** (Breiman 1996): Fit many models to bootstrap-resampled versions of the training data, and classify by majority vote

- **AdaBoost** (Freund and Schapire 1996): Fit many weak learners to reweighted versions of the training data while increase (boosting) the efficiency, and classify by weighted majority vote

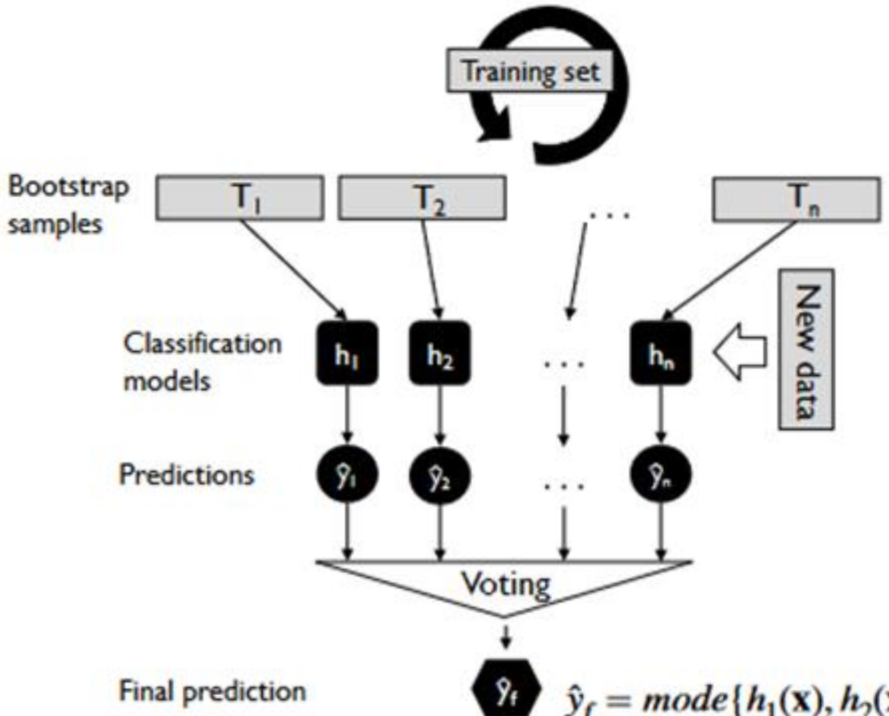
- **Gradient Boosting** (Friedman 2000): Employs Gradient Descent algorithm to minimize errors in sequential models, and classify by weighted majority vote

- **Random Forests** (Breiman 2001): Bagging w. trees + random feature subsets

- **XGBoost** (Chen and Guestrin 2016): It is an ideal combination of hardware and software optimization techniques to achieve superior results by utilizing minimal computing resources in short periods

Ensembles using independent models: Bagging

Bagging or Bootstrap Aggregating (Breiman 1996)



$$\hat{y}_f = mode\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots h_n(\mathbf{x})\}$$

where $h_i(\mathbf{x}) = \hat{y}_i$

Training example indices	Bagging round 1	Bagging round 2	...
1	2	7	...
2	2	3	...
3	1	2	...
4	3	1	...
5	7	1	...
6	2	7	...
7	4	7	...
	h_1	h_2	h_n

Image credits: Sebastian Raschka

If the base learner has a high variance (i.e. very sensitive to variations on the training sample), this procedure ensures diversity among the n models

Ensembles using independent models: Bagging

- Obtains a set of n models using **different bootstrap samples** of the given training data
  - for each model a **sample with replacement** of the same size as the available data is obtained
  - this means that for each model there is a small proportion of the examples that will be different
- **Bagging should be applied to base learners with high variance:**
  - Decision trees, Rule learners, etc
- Easy to implement with any algorithm
- Easy to implement in parallel environments
- The bias-variance argument:
  - **error decreases due to reduction in the variance component**

Ensembles using independent models: Random Forests

Random Forests (Breiman 2001): Bagging w. trees + random feature subsets

- Ensemble classification (and regression) algorithm based on decision trees
- Construct an ensemble of decision trees by manipulating the training set and the input features
  - Use bootstrap sample to train every decision tree (similar to Bagging)
  - At every internal node of DT, randomly sample feature subsets (p attributes) for selecting split criterion
- Easy to implement
- Very effective
- Reduces variance of unstable classifiers without negatively impact the bias (good generalization properties)
- Algorithm outputs **more information than just class label/value**

Ensembles using independent models: Random Forests

Combine weak learners (unpruned trees) and generate a classification algorithm with good performance

- Each weak learner is a decision tree trained in **parallel** with
  - Random training set: a **bootstrap sample**
  - Random **selection of features** at each node
    - Typically, the number of selected features is $m = \sqrt{D}$ or $m = log_2(D)$, where D is the number of features
  - For each tree grown on a bootstrap sample, the error rate for **examples left out of the bootstrap** sample is monitored
    - Out-of-Bag (OOB): test set for testing performance

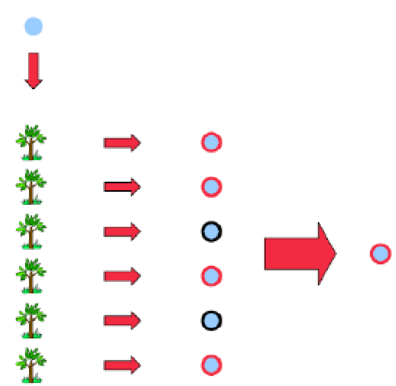
Ensembles using independent models: Random Forests

The examples not included in the bootstrap sample form the Out-Of-BAG (OOB) set

Complete data	Training data	Out of bag
1 2 3 4 5 6 7 8 9 10	2 4 8 9 10 6 1 1 7 6	3 5
1 2 3 4 5 6 7 8 9 10	10 3 5 6 1 8 5 2 4 6	7 9
1 2 3 4 5 6 7 8 9 10	10 1 10 5 4 1 10 7 2 2	3 6 8 9
1 2 3 4 5 6 7 8 9 10	1 10 10 4 1 4 10 1 9 9	2 3 5 6 7 8
1 2 3 4 5 6 7 8 9 10	9 6 1 9 2 3 5 10 9 2	4 7 8
1 2 3 4 5 6 7 8 9 10	10 10 8 5 8 7 9 8 3 8	1 2 4 6
⋮	⋮	⋮

Classification rule:

- each new sample is classified by all trees
- final decision by majority vote

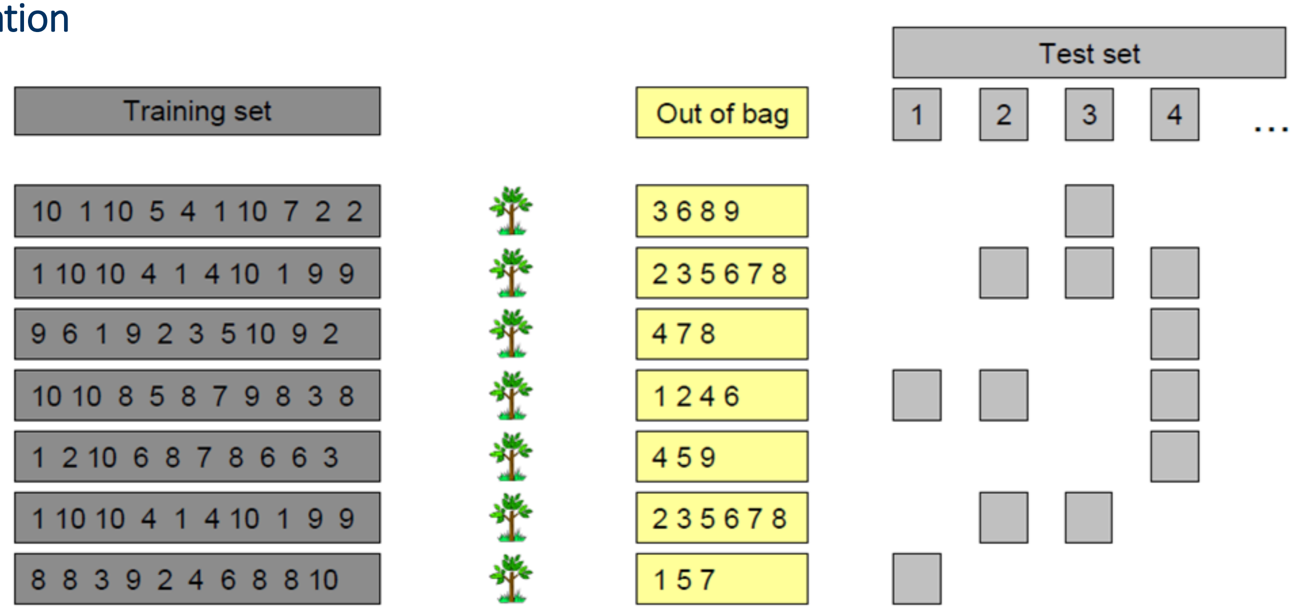


Ensembles using independent models: Random Forests

Out-Of-Bag error:

Estimating the TEST error (during train phase): for each example **x** in data set

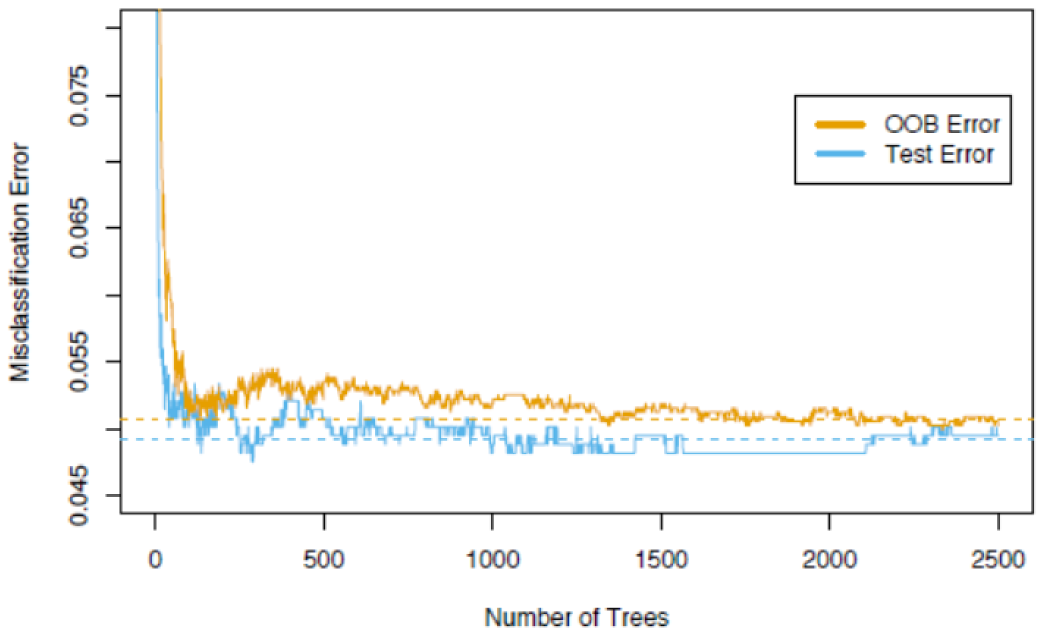
- Predicting the response of the **x** using the trees in which **x** ∈ OOB set
  - Good estimate for the generalization error because the information provided by **x** was not used for building these trees
  - **No hold-out or cross-validation**



Ensembles using independent models: Random Forests

Out-Of-Bag error:

- Misclassification error Out-Of-Bag error versus test error (hold-out method)



Ensembles using independent models: Random Forests

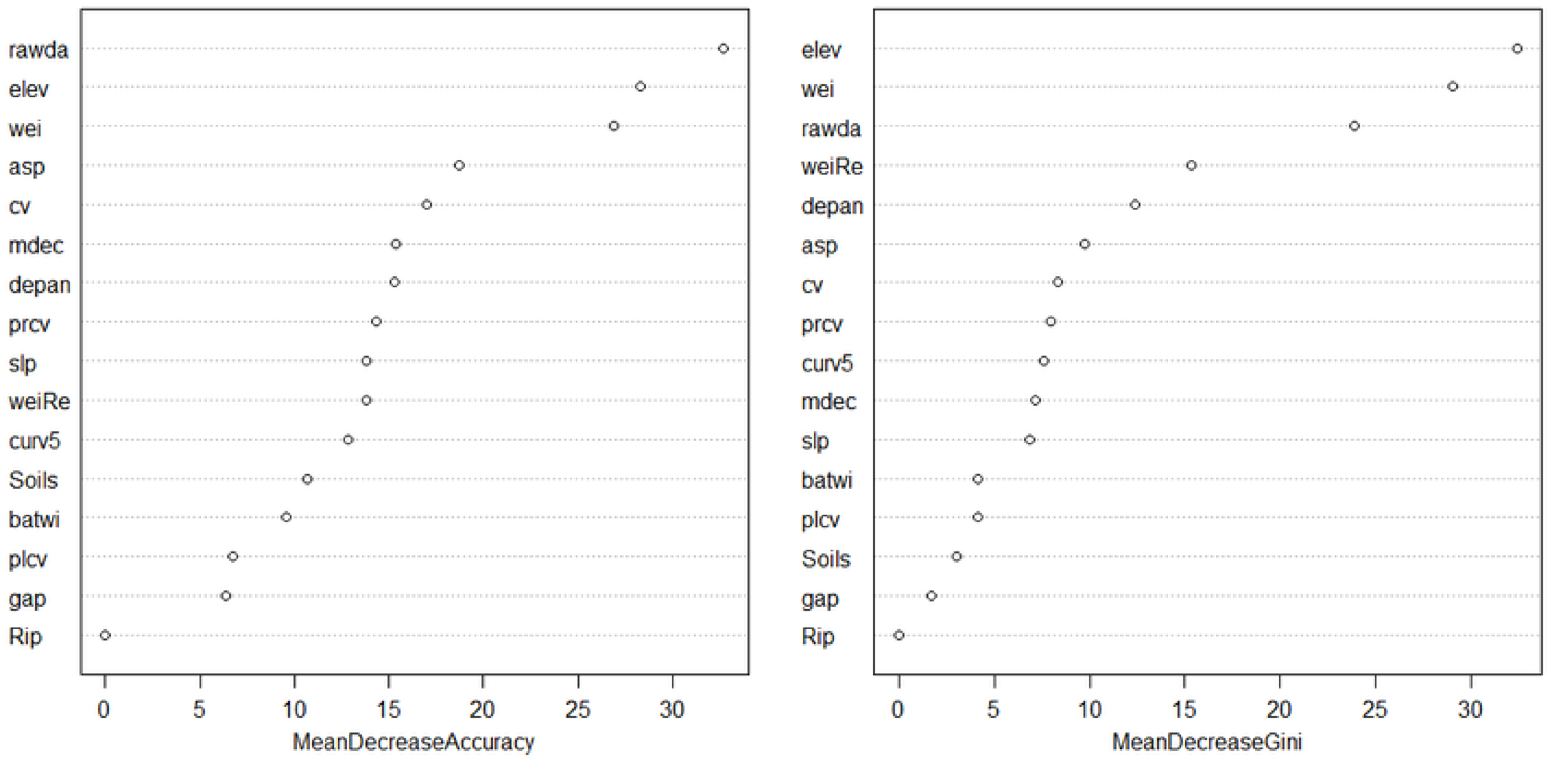
Feature importance

- The use of forests of trees to **evaluate the importance of features** on a classification task:
  - Which variables have the most predictive power?

- The more often used measures are:
  - **Mean Decrease in Impurity:** the average of the decrease in impurity over all nodes (all trees) where the feature is used for a rule
    - how much the impurity decreases when the variable is chosen to split a node?
  - **Mean decrease on the accuracy/mean square-error increases** in out-of-bag subset. After **randomly permutate the feature** in the feature vector the decrease in accuracy is calculated
    - how much the accuracy decreases / mean square error increases when the variable is excluded?

Ensembles using independent models: Random Forests

Feature importance



Sheng-Guo Wang, et al., **Random Forest Classification and Automation for Wetland Identification based on DEM Derivatives**. 2015

Ensembles using independent models: Random Forests

Advantages

- Do **not require elaborate tuning of the hyper-parameters**. Often these can/should be optimized
- The most important parameter to tune is the **number of trees to grow**, typically the larger the best
- Do not need to worry about creating very complex trees
- **Less prone to overfitting** (than DT)
- Slower than DT, but faster than typical bagging or boosting
- **Out-Of-Bag error** (cross-validation is not needed)

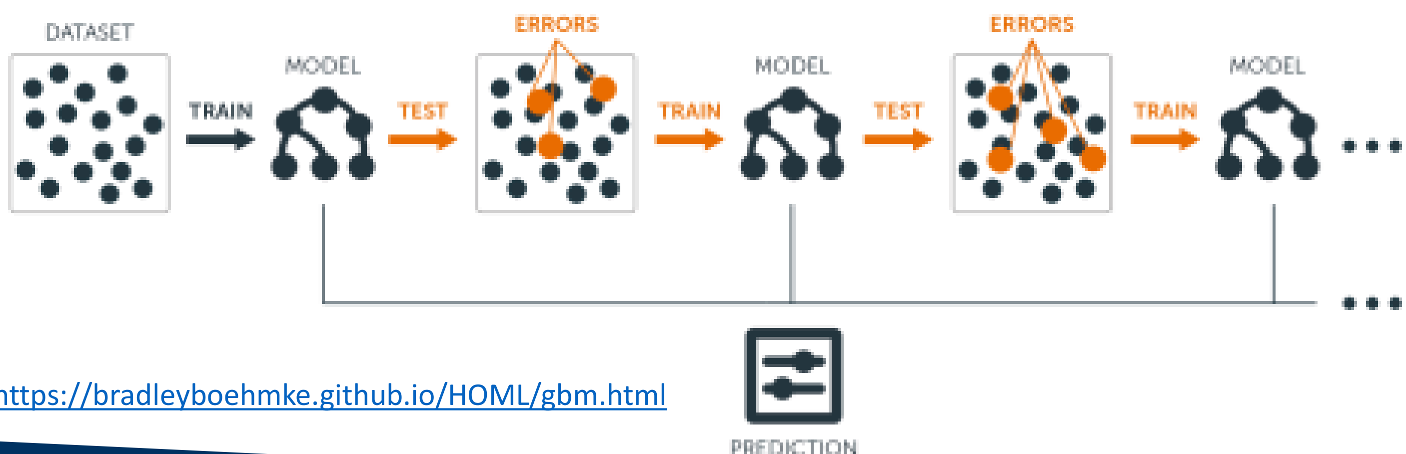
Disadvantages

- Do **not provide the interpretability** level of a Decision Tree

Ensembles using iterative models: AdaBoost

AdaBoost or Adaptive Boosting (Freund and Schapire 1996)

- AdaBoost was the first successful boosting algorithm developed for binary classification
- AdaBoost is best used to boost the performance of decision trees on binary classification problems
- Is used for classification rather than regression
- Can be used to **boost** the performance of any machine learning algorithm



https://bradleyboehmke.github.io/HOML/gbm.html

Ensembles using iterative models: AdaBoost

AdaBoost or Adaptive Boosting (Freund and Schapire 1996)

Ensemble of weak classifiers trained sequentially, $\boldsymbol{f_t} = 1, \dots, n$

Goal: Train each classifier given the performance of previous weak classifiers

- At each step t
  - Modify training sample distribution in order to favor difficult examples (according to previous weak classifiers)
  - Train a new weak classifier $\boldsymbol{f_t}$
  - Select the new weight $\boldsymbol{\alpha_t}$ by optimizing a global criterion
- Stop when impossible to find a weak classifier satisfying the simplest condition (being better than chance)
- Final classifier is the combination (with weights α_t) of all n classifiers
  - Assigns weights to the N classifiers: a classifier with good a classification result on the training data will contribute more than a poor one

Ensembles using iterative models: AdaBoost

Most popular algorithm in the family of boosting algorithms

- Boosting: the performance of simple (weak) classifiers is boosted by combining them iteratively (usually **Decision Tree Stumps**)

- Combination rule

$$g(\mathbf{x}) = \sum_{t=1}^{n} \alpha_t \, f_t(\mathbf{x})$$, where α_t means the importance of classifier f_t

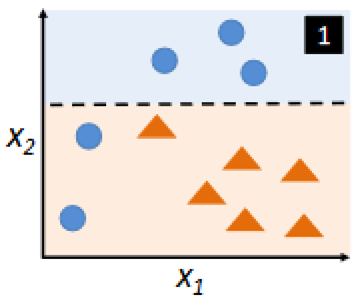
- Simplest framework: binary classification, each $f_1 = \{-1, +1\}$

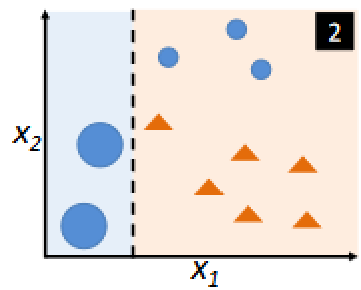
- The following simplest requirement: **each weak classifier f_t should perform better than chance**

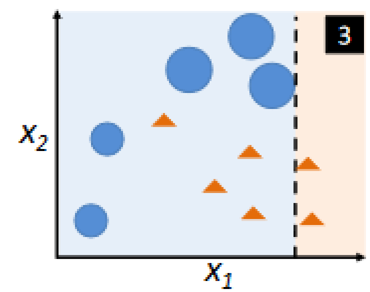
Ensembles using iterative models: AdaBoost

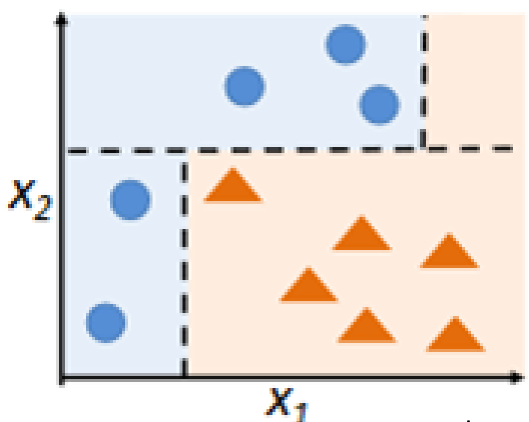
Binary Classification problem: with 3 trained weak learners

$$g(\mathbf{x}) = sign\big(\alpha_1 f_1(\mathbf{x})\big) + sign\big(\alpha_2 f_2(\mathbf{x})\big) + sign\big(\alpha_3 f_3(\mathbf{x})\big)$$

0.92 

+ 0.65 

+ 0.42 



$$f_i(\mathbf{x}) = \begin{cases} +1\,, \mathbf{x} \in blue \\ -1\,, \mathbf{x} \in pink \end{cases}$$

Image credits: Sebastian Raschka

Ensembles using iterative models: GBM

Gradient Boosting Machine (Friedman 2000)

- Boosting problem as an **optimization** problem
- **Sequential** ensemble learning
- Base learners are generated **sequentially** in such a way that the **present base learner is always more effective than the previous one**
- Weights for misclassified outcomes are not incremented
- At each step, adds another weak learner to increase the performance and build a strong learner
- Final classifier is the equally weighted combination of all *n* classifiers, but their predictive capacity is restricted with learning rate to increase accuracy

Ensembles using iterative models: AdaBoost vs GBM

Adaboost	Gradient Boost
An additive model where shortcomings of previous models are identified by high-weight data points.	An additive model where shortcomings of previous models are identified by the gradient.
The trees are usually grown as decision stumps.	The trees are grown to a greater depth usually ranging from 8 to 32 terminal nodes.
Each classifier has different weights assigned to the final prediction based on its performance.	All classifiers are weighed equally and their predictive capacity is restricted with learning rate to increase accuracy.
It gives weights to both classifiers and observations thus capturing maximum variance within data.	It builds trees on previous classifier's residuals thus capturing variance in data.

Ensembles using iterative models: XGBoost

eXtreme Gradient Boosting (XGBoost) (Chen and Guestrin 2016)

- An **advanced** version of Gradient Boosting Method
- Software and hardware optimization
    - a scalable tree boosting system

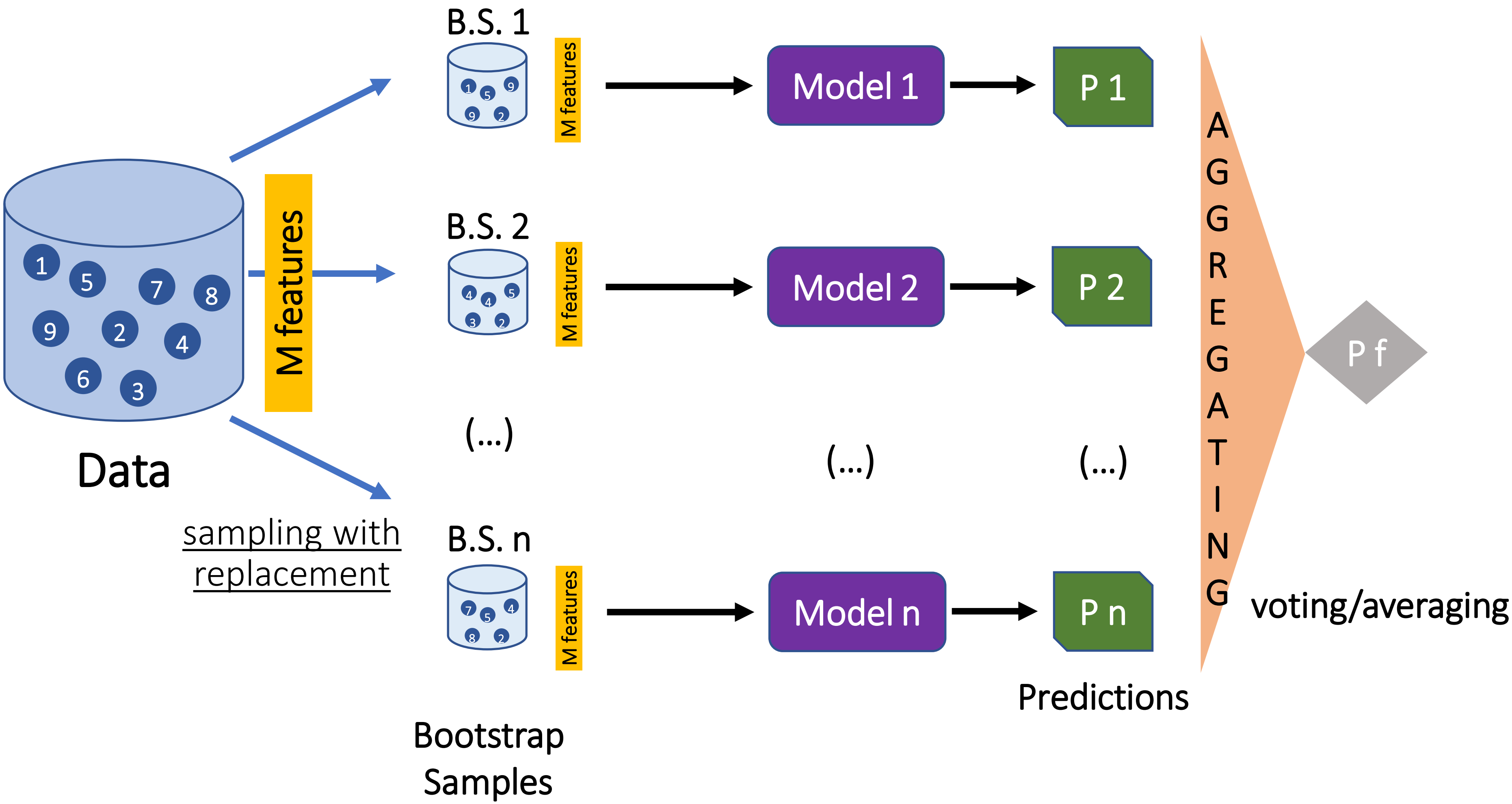
Some features:

- clever penalization of trees: weights of the trees that are calculated with less evidence is shrunk more heavily
- extra randomization parameter to reduce the correlation between the trees
- parallelization, cache optimization, distributed computing, etc

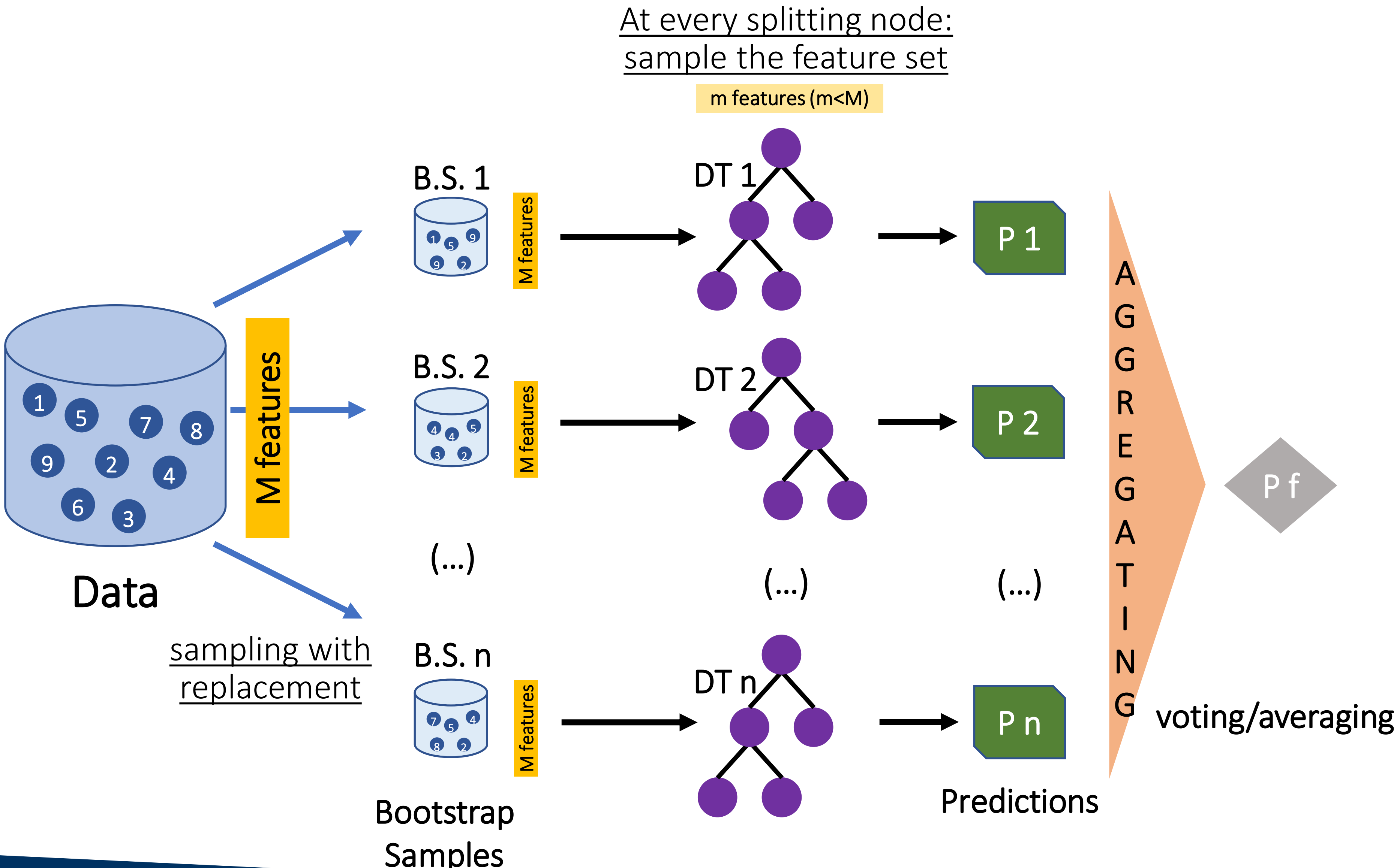
Contents

- Ensembles
- Types of ensembles
- Ensembles methods
- **Summary**

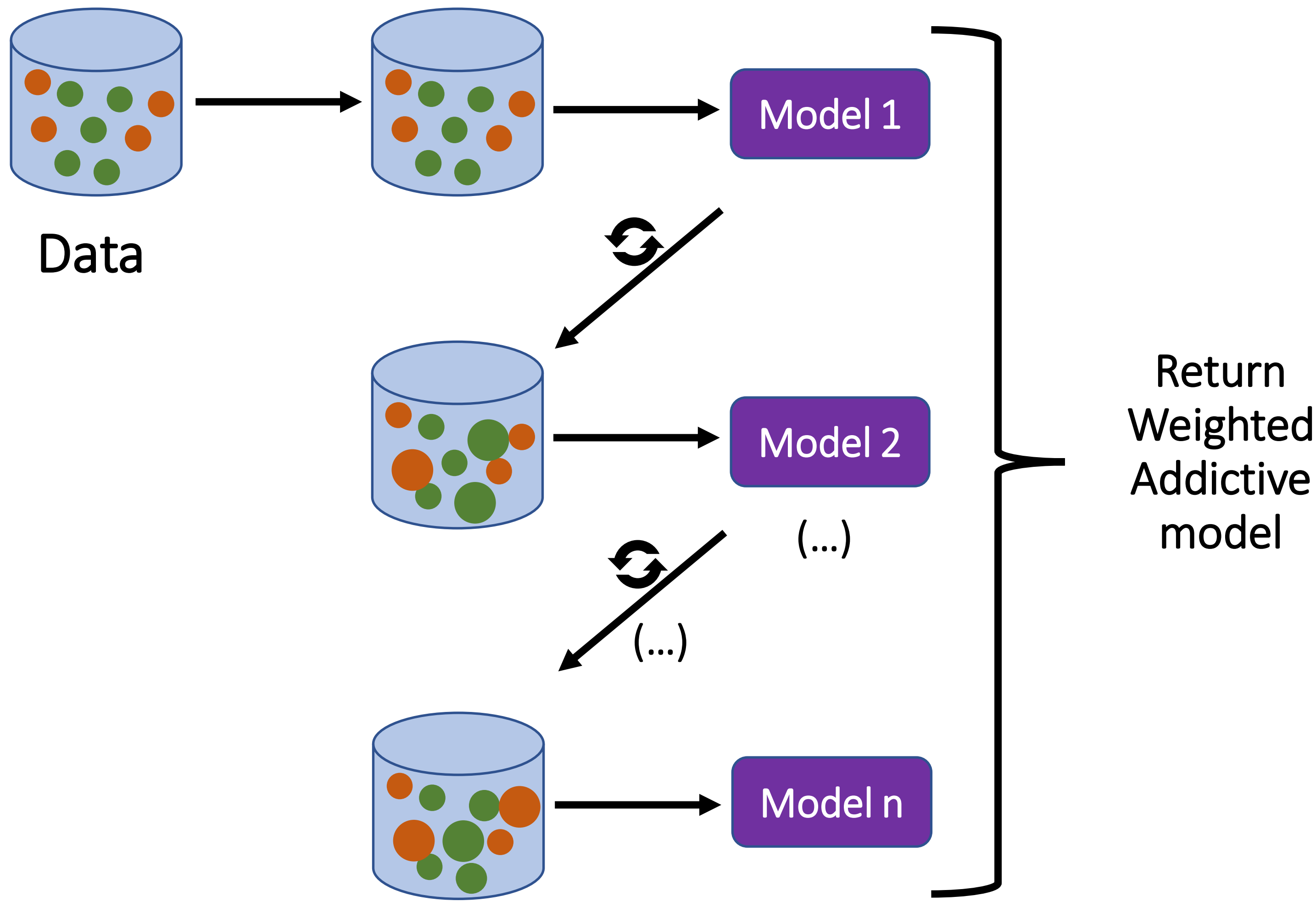
Summary: Bagging



Summary: Random Forests

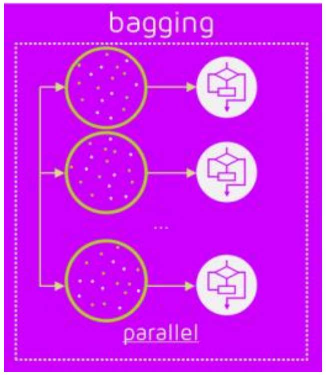


Summary: AdaBoost



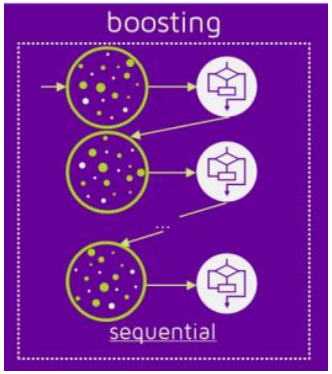
Summary: Bagging & Boosting

Training stage:



Bagging Methods :

- parallel
- **bootstrap samples**



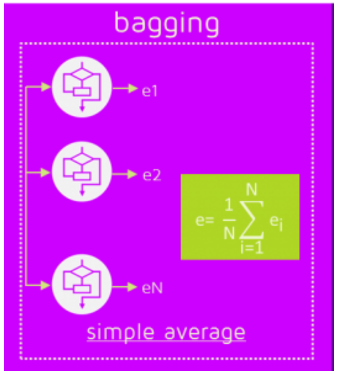
Boosting Methods :

- Sequential
- Increase of the **weights of misclassified data** to emphasize the most difficult example
  - subsequent learners will focus on them during their training

https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

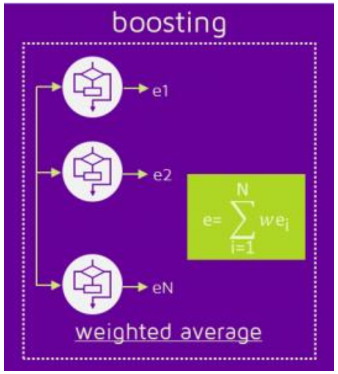
Summary: Bagging & Boosting

Prediction stage: apply the n learners to the new observations



Bagging Methods :

- Prediction is obtained by **equally** voting/averaging the responses of the n learners



Boosting Methods :

- Prediction is obtained by voting/averaging **but the models contributions depend in their performance**

https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

Summary: Bagging & Boosting

Bagging Methods

- Error reduction due to reduction in variance
- Effective with unstable models

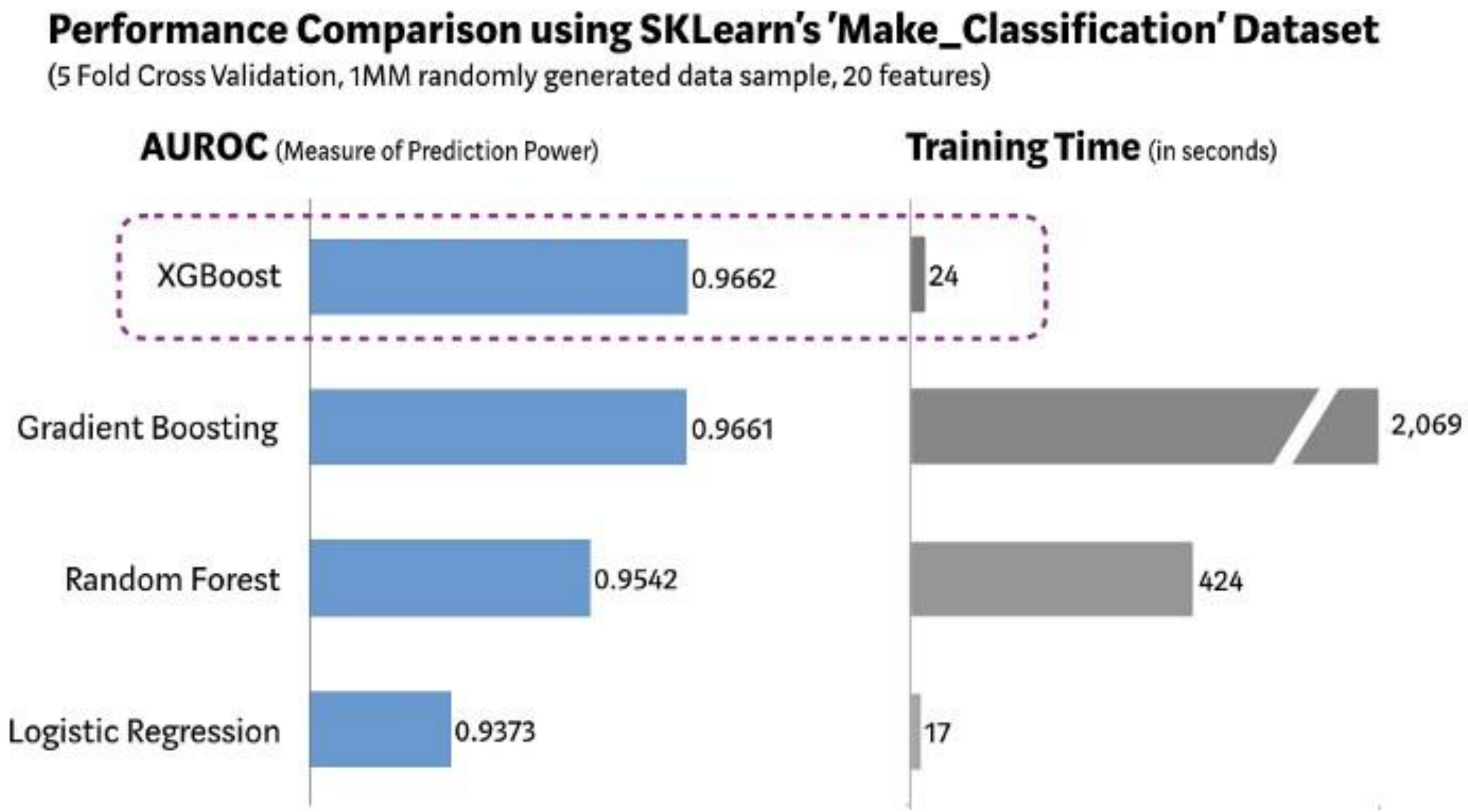
$\Rightarrow$ single model is **over-fitting** → Bagging ensembles can get reduced variance

Boosting Methods

- Error reduction due to reduction in bias and variance
- Risky in problems with noise (increase of the error)
- more prune to over-fitting

$\Rightarrow$ single model with **low performance** → Boosting ensembles can get a **better bias**

Summary: comparision



https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Bibliography

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Pearson*, 2019 (chap 6.10)

Data Mining, the Textbook, Charu C. Aggarwal, *Springer*, 2015 (chap 11)

https://sebastianraschka.com/pdf/lecture-notes/stat451fs20/07-ensembles__slides.pdf

Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140

Breiman, Leo. “Random Forests” Machine learning 45.1 (2001): 5-32.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1), 119-139

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.