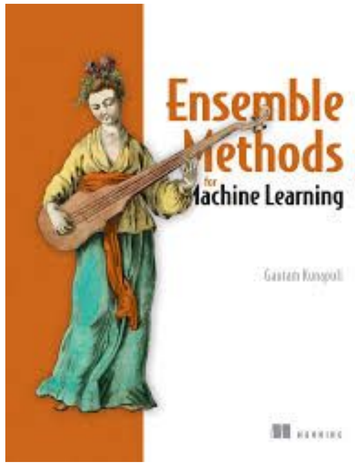
IF3270 Pembelajaran Mesin

Ensemble Methods

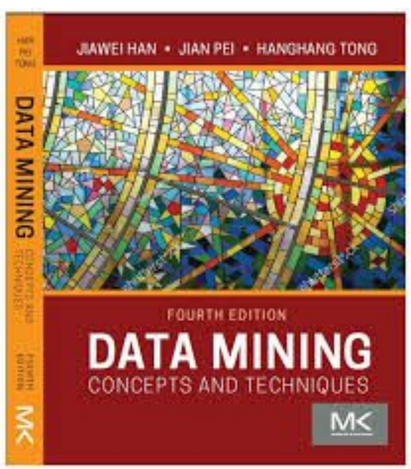
Tim Pengajar IF3270

Reference

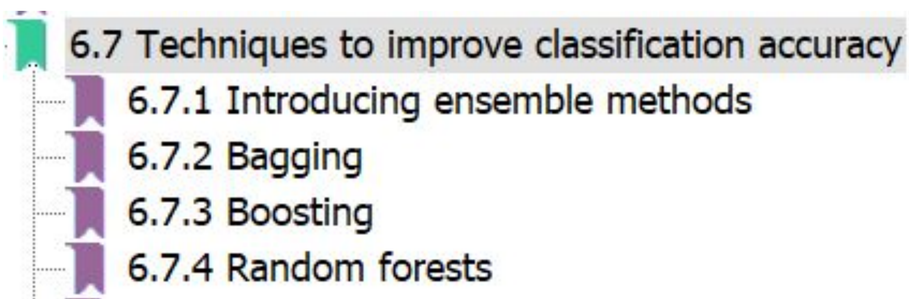


Kunapuli, G. (2023). *Ensemble methods for machine learning*. Simon and Schuster.

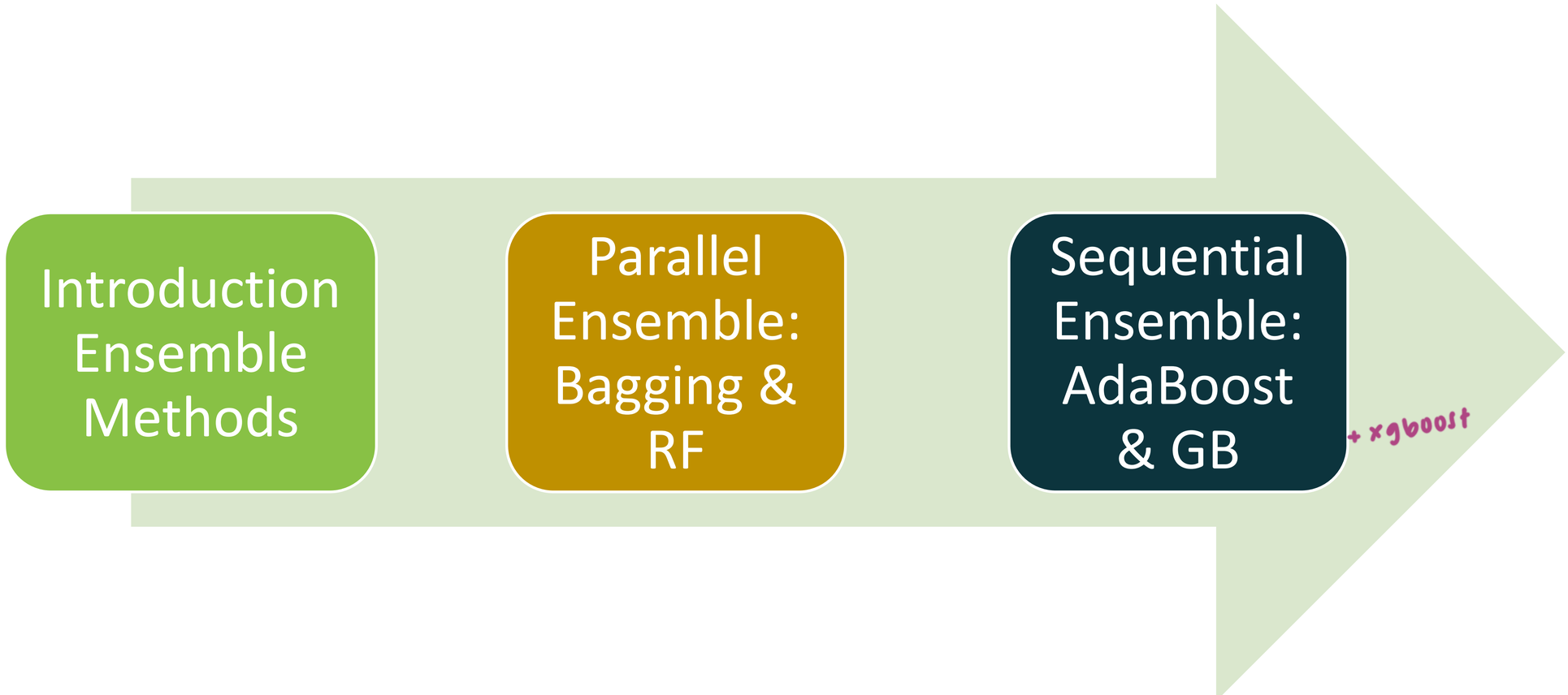
https://livebook.manning.com/book/ensemble-methods-for-machine-learning/chapter-1/19

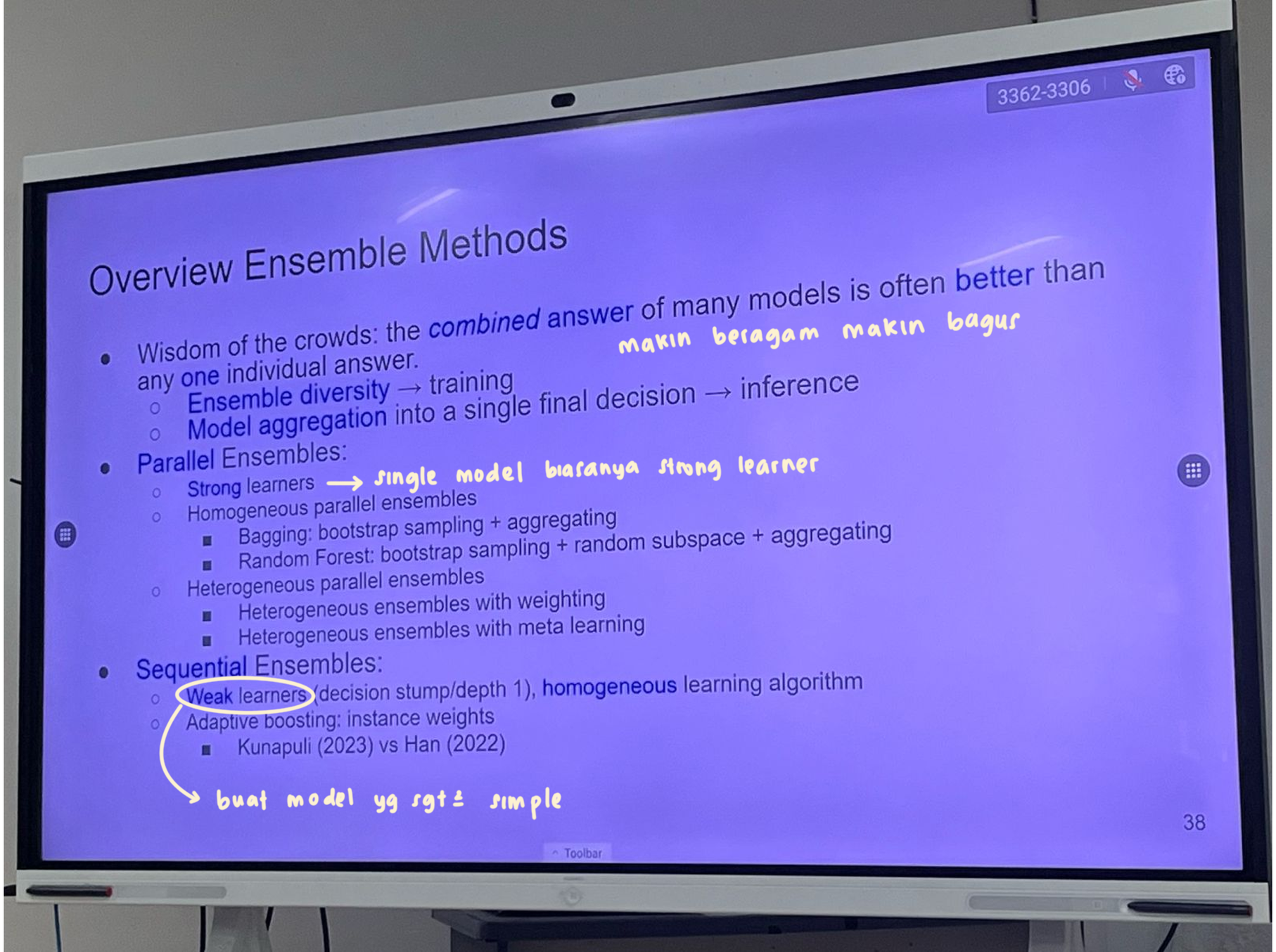


Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques*. Morgan kaufmann.

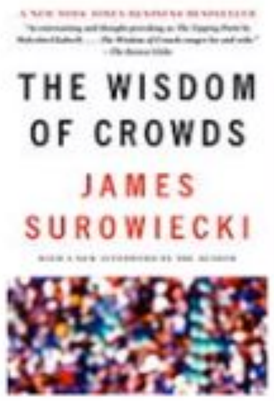


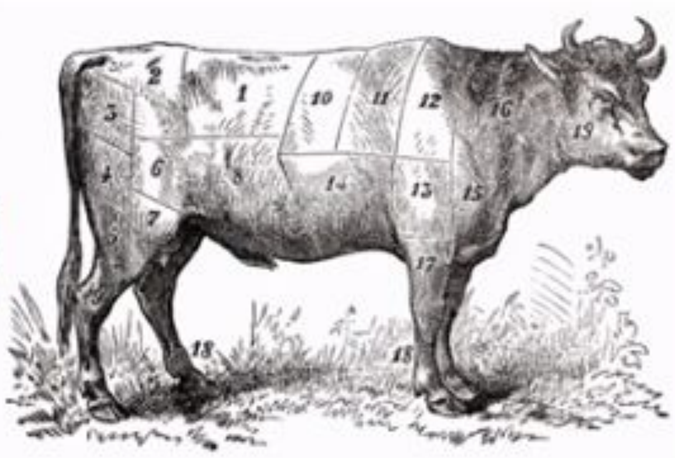
Outline


Introduction Ensemble Methods
Parallel Ensemble: Bagging & RF
Sequential Ensemble: AdaBoost & GB
+ xgboost


3362-3306
Overview Ensemble Methods
Wisdom of the crowds: the combined answer of many models is often better than any one individual answer.
Ensemble diversity → training
Model aggregation into a single final decision → inference
Parallel Ensembles:
Strong learners
Homogeneous parallel ensembles
Bagging: bootstrap sampling + aggregating
Random Forest: bootstrap sampling + random subspace + aggregating
Heterogeneous parallel ensembles
Heterogeneous ensembles with weighting
Heterogeneous ensembles with meta learning
Sequential Ensembles:
Weak learners (decision stump/depth 1), homogeneous learning algorithm
Adaptive boosting: instance weights
Kunapuli (2023) vs Han (2022)
makin beragam makin bagus
single model biasanya strong learner
buat model yg sgt ± simple
38
Toolbar

The Wisdom of Crowds





average of 800 guesses = 1,197
actual weight of the ox = 1,198

The wisdom of the crowd is the process of taking into account the **collective opinion** of a group of individuals rather than a **single expert** to answer a question.

https://vivifychangecatalyst.wordpress.com/2014/07/25/wisdom-of-crowd-crowdsourcing-performance-appraisal/

Ensemble Methods: The Wisdom of the Crowds

Dr. Randy Forrest works at a teaching hospital, and have a team with a diversity skills. Each resident has their own strengths.

Every time Dr. Forrest gets a new case, he solicits the opinions of his residents and then democratically selects the final diagnosis as **the most common one** from among all those proposed.

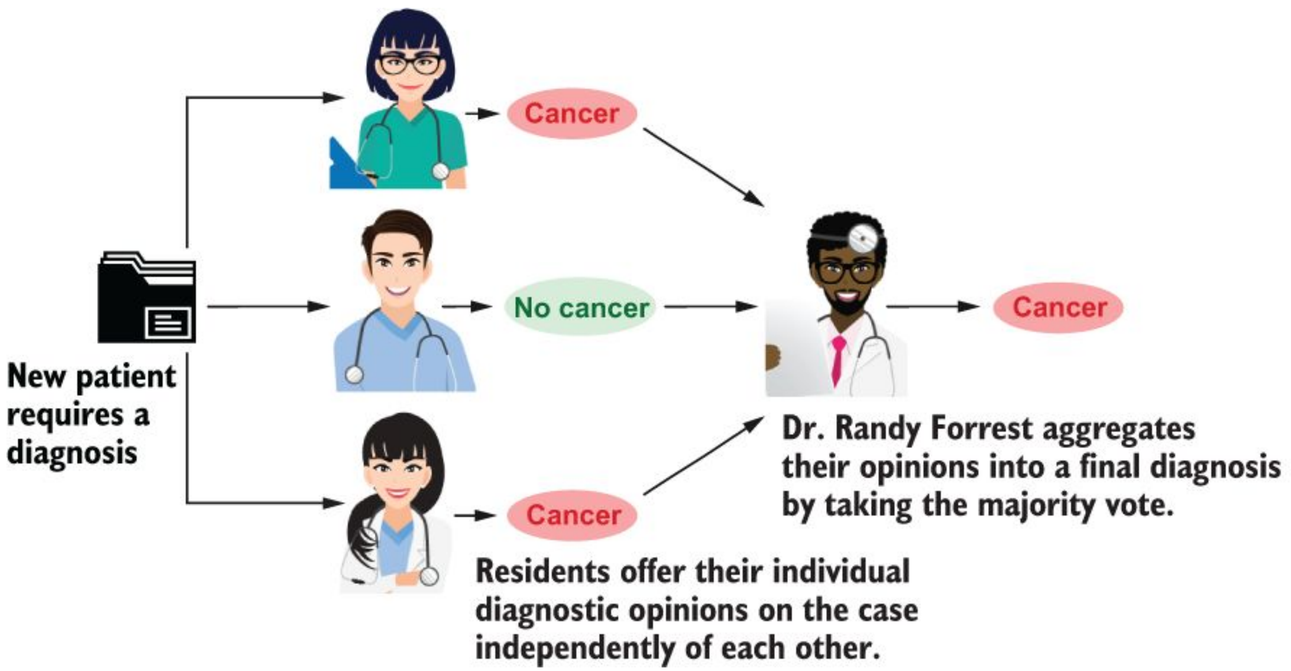
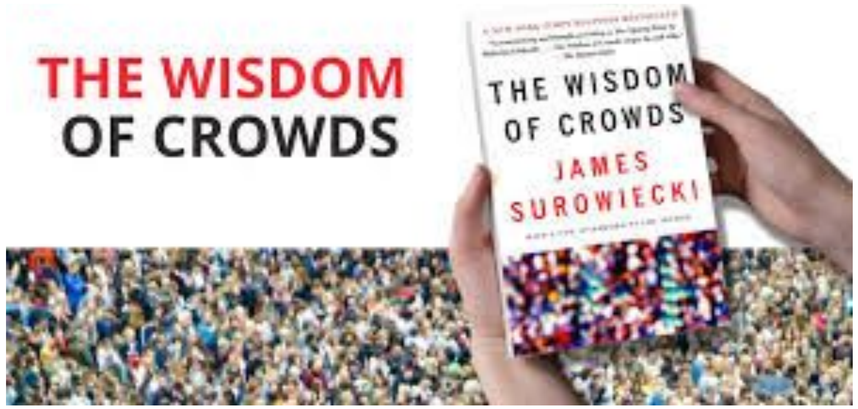


Figure 1.1 The diagnostic procedure followed by Dr. Randy Forrest every time he gets a new case is to ask all of his residents their opinions of the case. His residents offer their diagnoses: either the patient does or does not have cancer. Dr. Forrest then selects the majority answer as the final diagnosis put forth by his team.

Kunapuli, G. (2023). *Ensemble methods for machine learning*. Simon and Schuster.
https://livebook.manning.com/book/ensemble-methods-for-machine-learning/chapter-1/19

Ensemble Methods: The Wisdom of the Crowds

An ensemble method relies on the notion of “wisdom of the crowd”: the *combined* answer of many models is often better than any one individual answer.



The secrets to the success of ensemble methods are:

- Ensemble diversity (variety of opinions to choose from) variasi orgz yg ditanya
- Model aggregation into a single final decision agregasi modelnya

Diversity and independence are important because the best collective decisions are the product of disagreement and contest, not consensus or compromise.

https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

Terminology

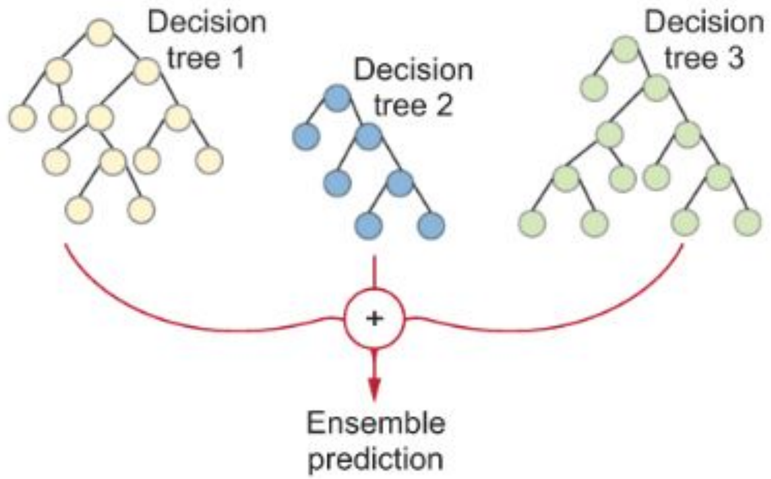
satu model yg jd kolektif dr model²nya

- Base models/learners/estimators: individual machine-learning models
- Parallel vs sequential ensemble method: train each base model independently (or in parallel) vs sequentially in stages.
- Homogeneous vs heterogeneous ensembles: all the base learners are trained using the same vs different base-learning algorithm.

mempengaruhi waktu training

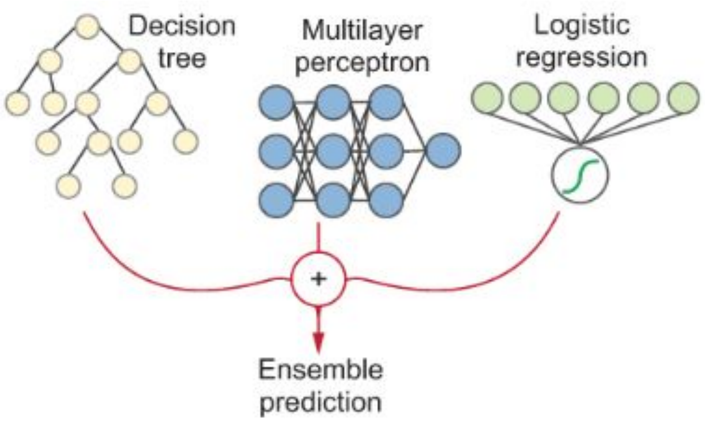
semua pake algoritma yg sama

Homogeneous parallel ensembles



semua pake algoritma yg berbeda

Heterogeneous parallel ensembles



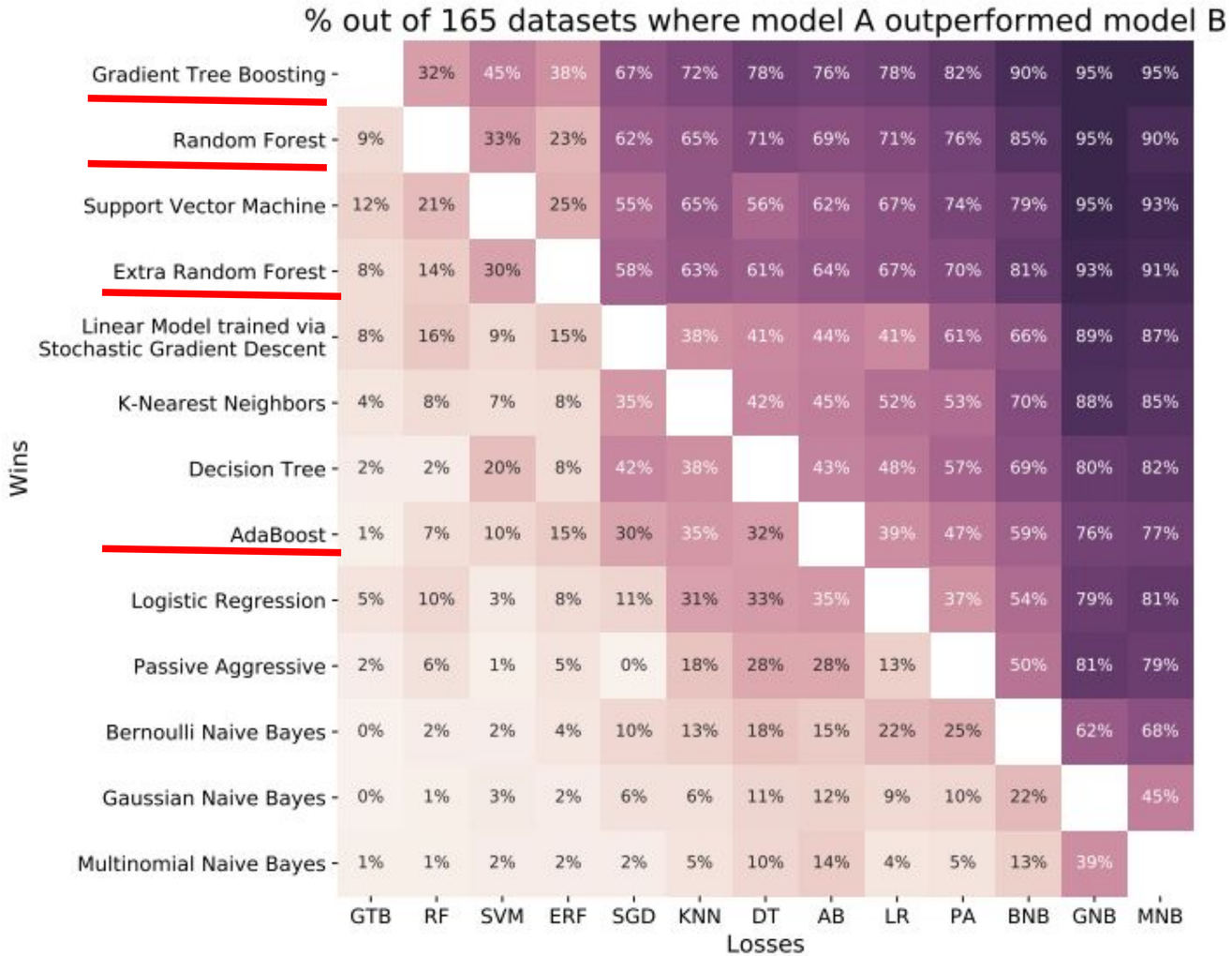
https://livebook.manning.com/book/ensemble-methods-for-machine-learning/chapter-1/125

Why Ensemble Methods ?

ketika data terstruktur & independen
biasanya bagus pakai ensemble

- Ensemble methods have proven to be very successful on structured data.
  - Anthony Goldbloom, CEO of Kaggle, revealed in 2015 that the three most successful algorithms for structured problems were XGBoost, random forest, and gradient boosting, all ensemble methods.
  - Olson et al. (2018) ranked machine-learning algorithm based on their performance on all benchmark data sets. Ensemble methods outperformed other methods, that's why ensemble methods (specifically, tree-based ensembles) are considered state of the art for structured dataset.

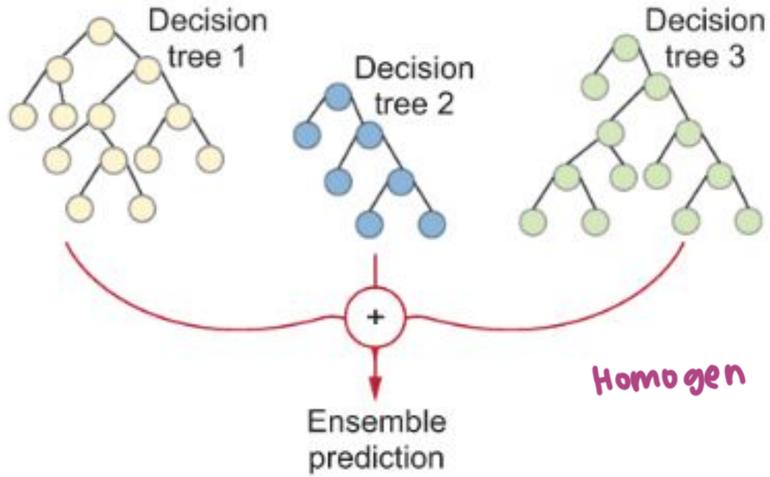
SOTA

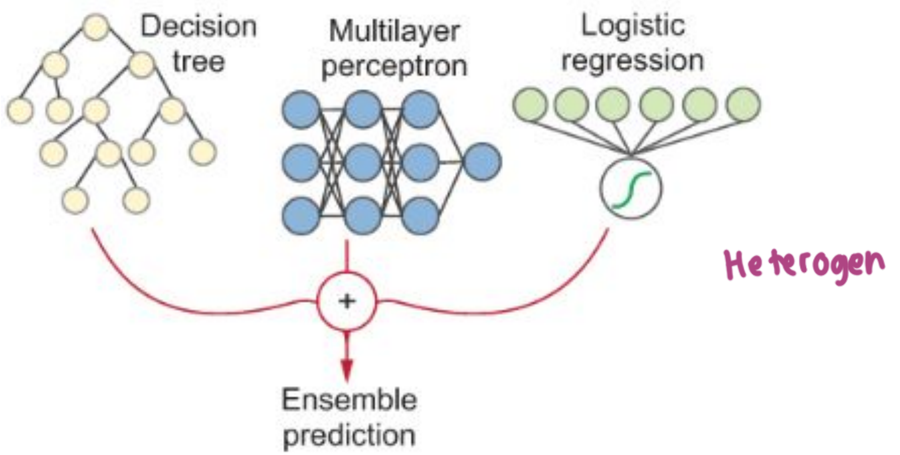


https://livebook.manning.com/book/ensemble-methods-for-machine-learning/chapter-1/125

Parallel Ensembles

exploit the *independence* of each base estimator



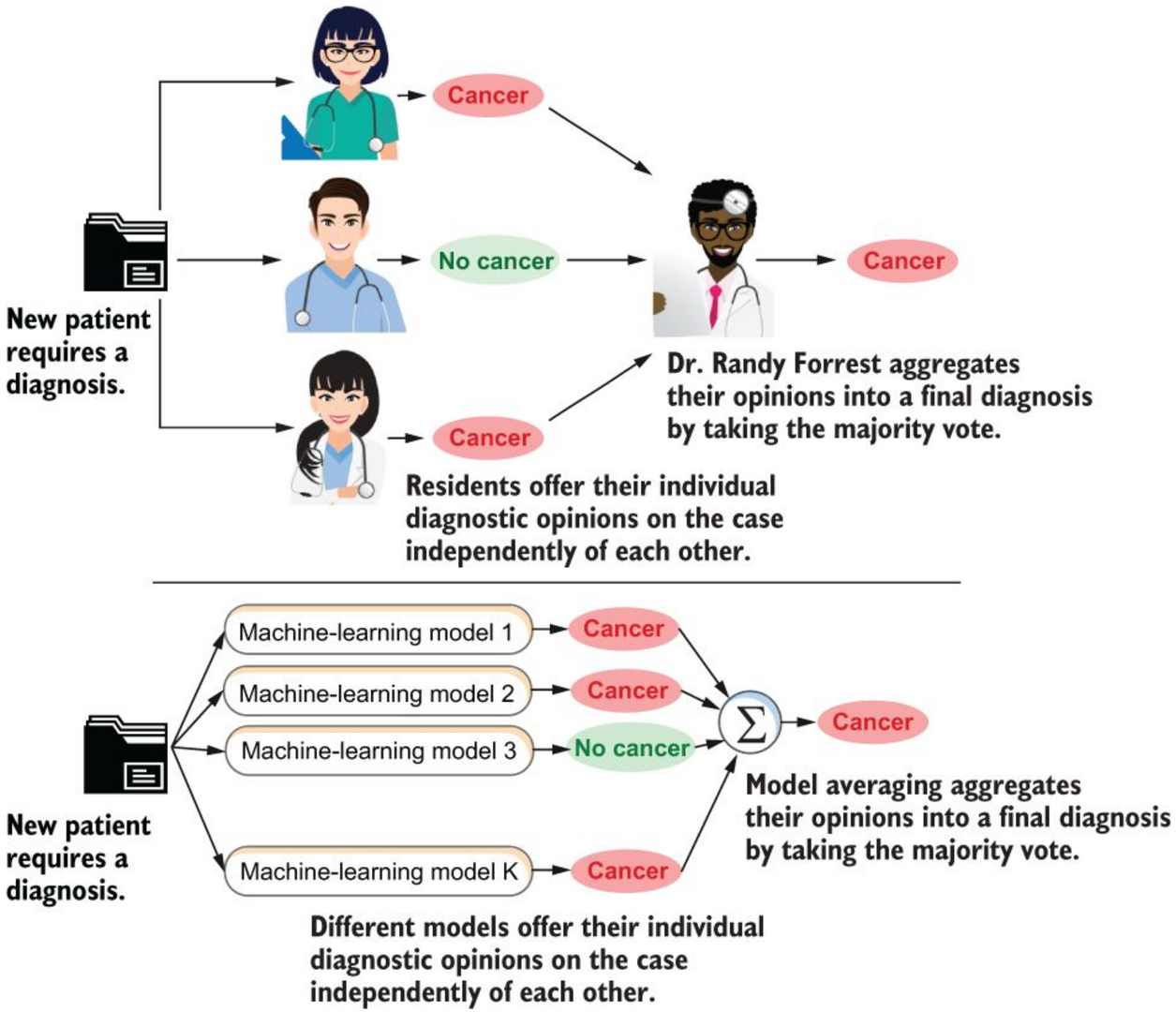


Homogeneous Parallel Ensembles

Homogeneous: component models are all trained using the same machine-learning algorithm

Parallel: train each component base estimator independently of the others (in parallel)

Dr. Randy Forrest's diagnostic process is an analogy of a parallel ensemble method

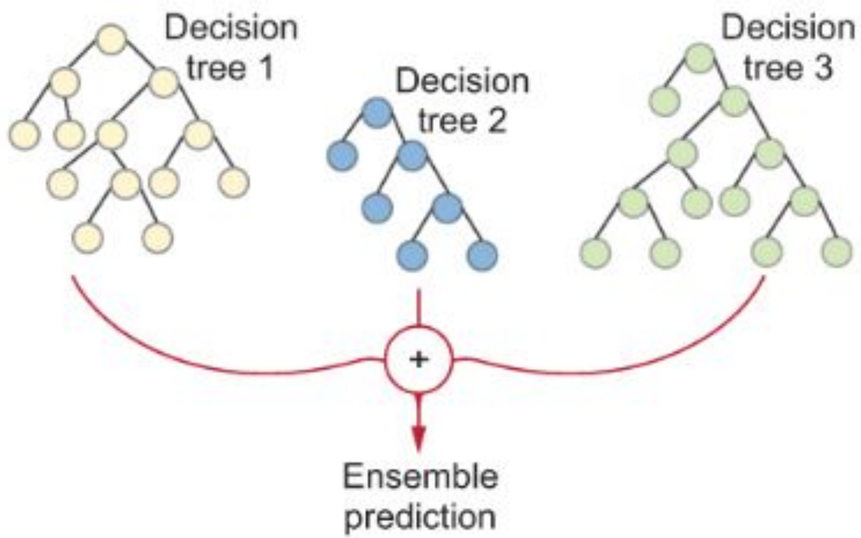


https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

Homogeneous Parallel Ensembles: Ensemble Diversity

We have a single data set, so how do we obtain different base models with the same learning algorithm in homogeneous ensemble ?

Tentuin best attribute yg bs jd node pakai information gain (AI)



biar dpt model tree beda bisa lakuin
- pruning
- random sampling → ini yg oke

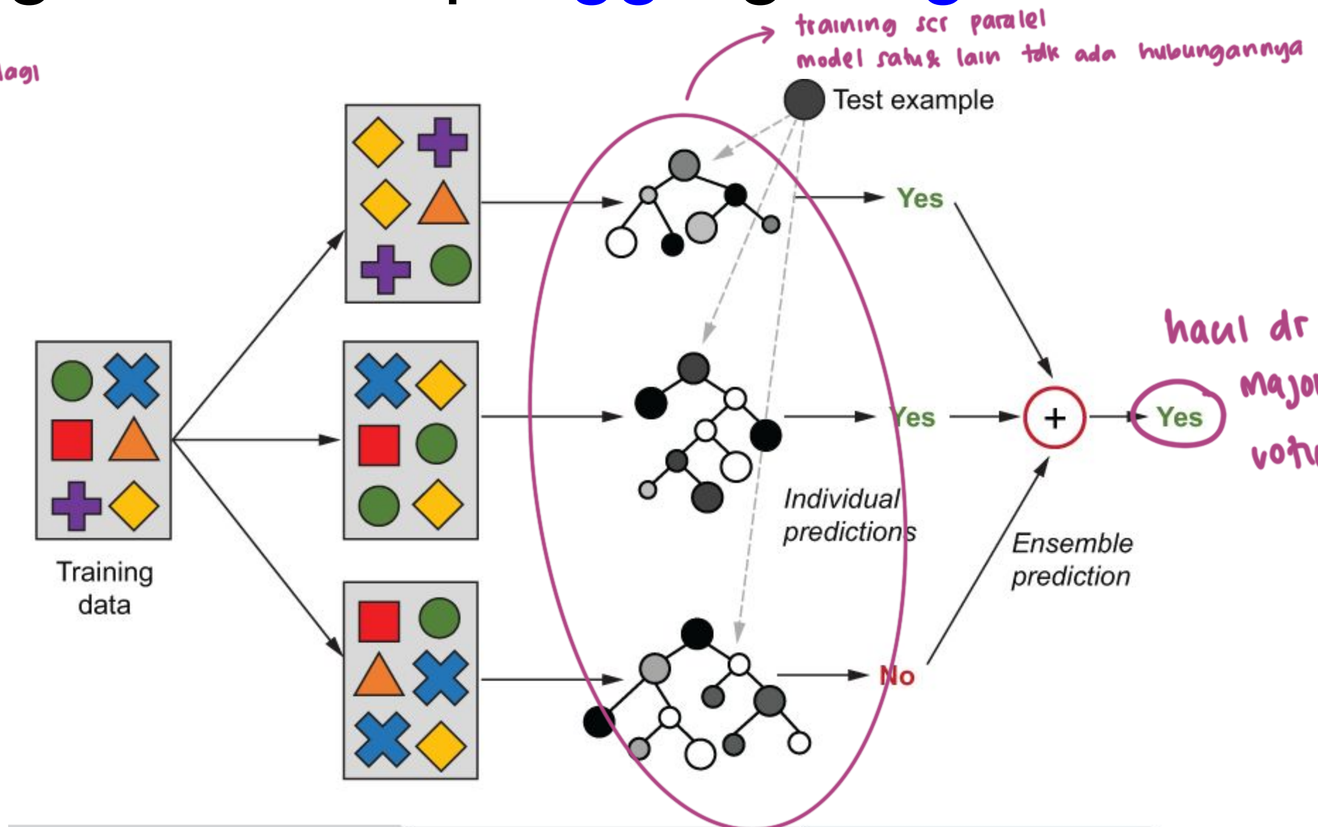
https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

Ensemble Diversity: Bagging - Bootstrap Aggregating

peluang ga berubah
abis diambil lgsg dikembalikin lagi

Training: bootstrap sampling (or sampling with replacement) is used to generate replicates of the training data set that are different from each other but drawn from the original data set. This ensures ensemble diversity.

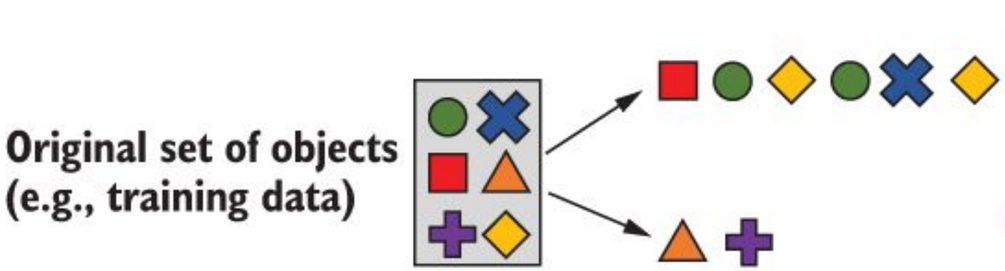
Prediction: model aggregation is used to combine the predictions of the individual base learners into one ensemble prediction (classification: majority voting; regressing: simple averaging).



1 dataset → bagi2 jd bbrp dataset dgn replacement concept → setiap dataset dimodelling secara paralel /brg

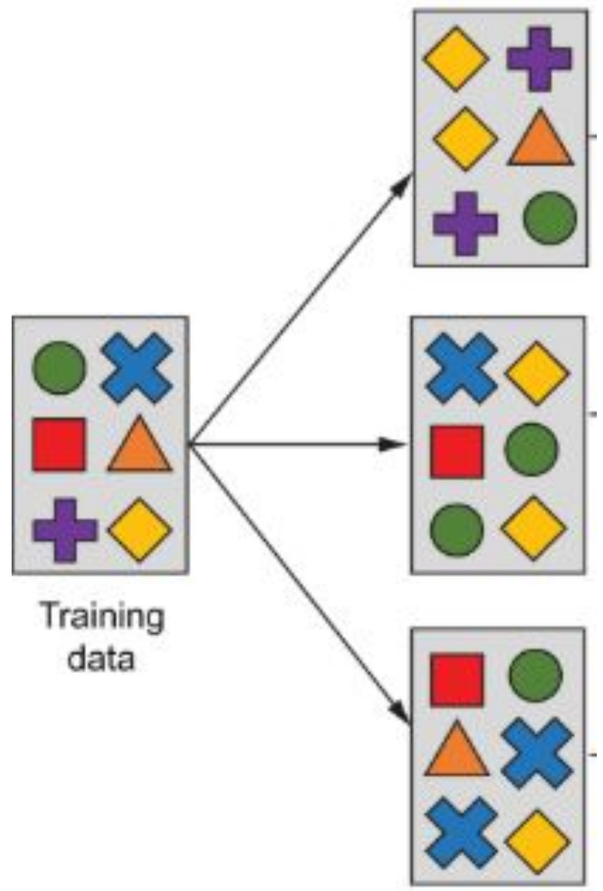
https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

Bootstrap Sampling



Bootstrap sample: Sampling with replacement allows some objects to be selected more than once.

Out-of-bag sample: Sampling with replacement means some objects will not be selected even once.




```
import numpy as np
bag = np.random.choice(range(0, 50), size=50, replace=True)
np.sort(bag)
```

```
array([ 1,  3,  4,  6,  7,  8,  9, 11, 12, 12, 14, 14, 15, 15, 21, 21, 21,
       24, 24, 25, 25, 26, 26, 29, 29, 31, 32, 32, 33, 33, 34, 34, 35, 35,
       37, 37, 39, 39, 40, 43, 43, 44, 46, 46, 48, 48, 48, 49, 49, 49])
```

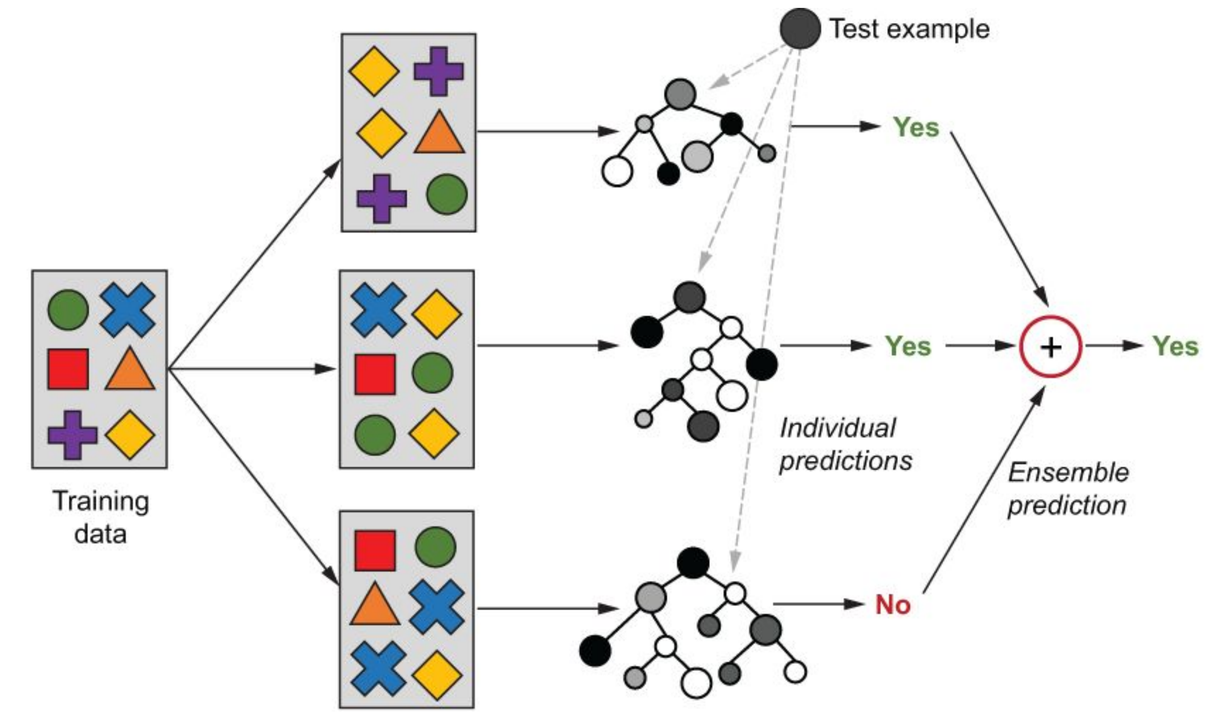
```
oob = np.setdiff1d(range(0, 50), bag)
oob
```

```
array([ 0,  2,  5, 10, 13, 16, 17, 18, 19, 20, 22, 23, 27, 28, 30, 36, 38,
       41, 42, 45, 47])
```

https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

Training with Bootstrap Sampling

Because different bootstrap samples will contain different examples repeating a different number of times, each base estimator will turn out to be somewhat different from the others.



https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

Bagging: Bootstrap Aggregation (Han et al., 2022)

Algorithm: Bagging. The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

Input:

- D, a set of d training tuples;
- k, the number of models in the ensemble;
- a classification learning scheme (e.g., decision tree algorithm, naïve Bayesian, etc.).

Output: The ensemble—a composite model, M^*.
Method:

(1) **for** $i = 1$ to k **do** // create k models:
(2) create bootstrap sample, D_i, by sampling D with replacement;
(3) use D_i and the learning scheme to derive a model, M_i.
(4) **endfor**

To use the ensemble to classify a tuple, X:

let each of the k models classify $\boldsymbol{X}$ and return the majority vote;

The bagged classifier, M∗, counts the votes and assigns the class with the most votes to X.

FIGURE 6.25

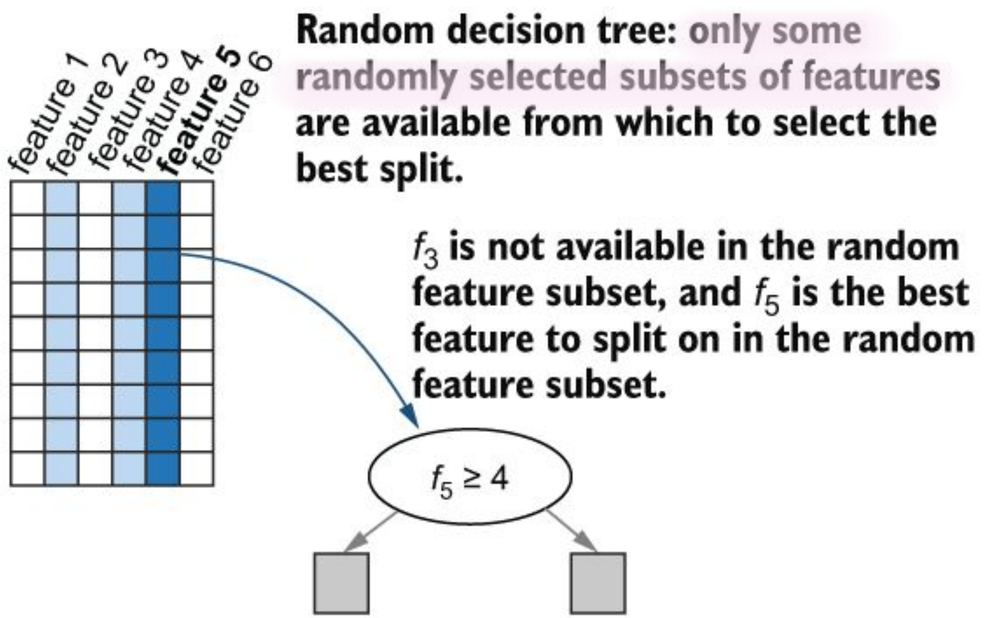
Bagging.

Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques*. Morgan kaufmann.

Random Forest

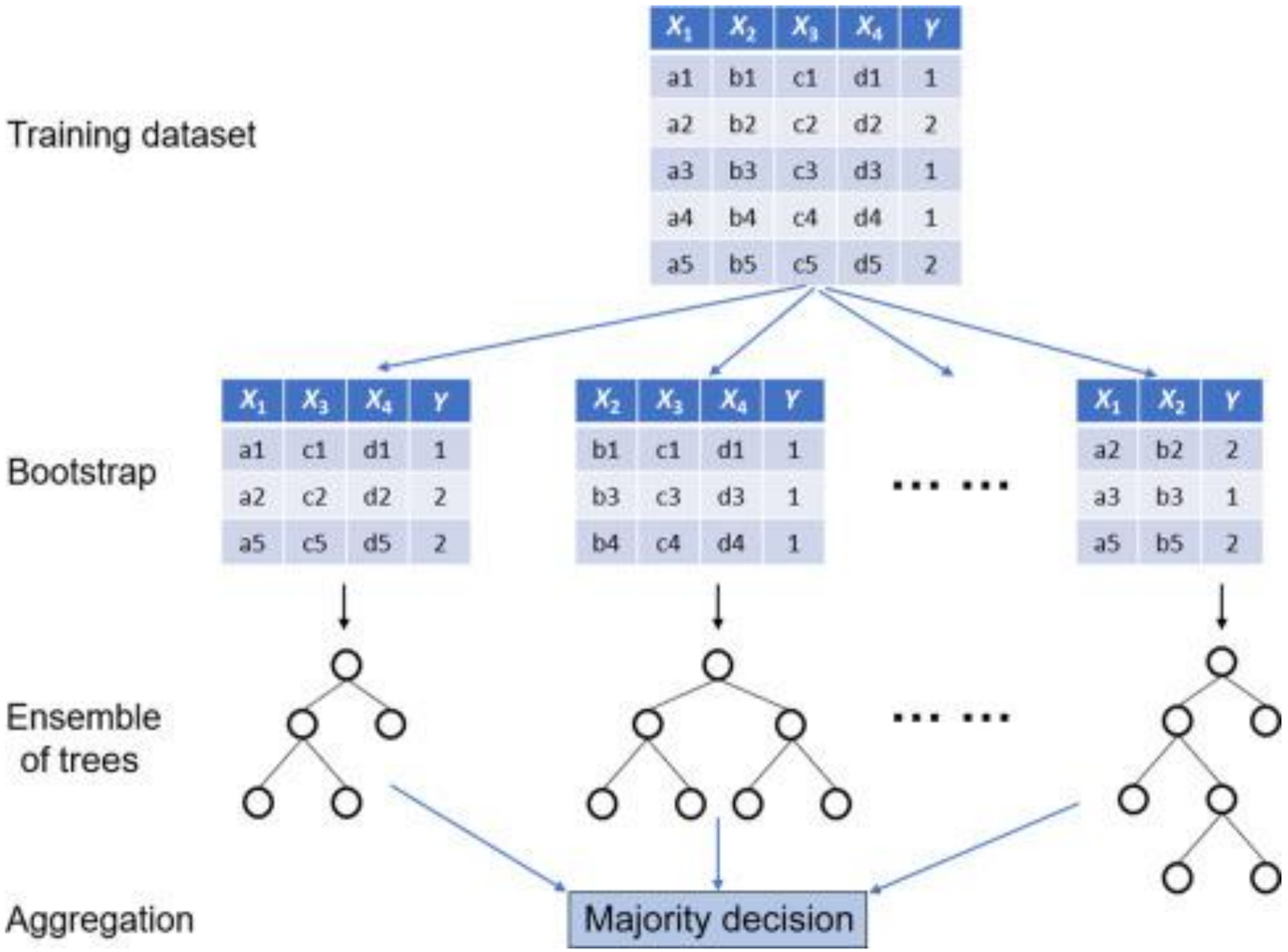
bagging ensemble tp tdk semua featurenya digunakan

- an ensemble of randomized decision trees constructed using a special extension of bagging.
- Randomized decision trees are trained using a modified decision-tree learning algorithm
  - Problem: the same small number of dominant features are repeatedly used in different trees. This makes the ensemble less diverse.
  - Solution: randomly samples features before creating a decision node
- This additional source of randomness increases ensemble diversity and generally leads to better predictive performance.



https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

Random Forest



X_1	X_2	X_3	X_4	Y
a1	b1	c1	d1	1
a2	b2	c2	d2	2
a3	b3	c3	d3	1
a4	b4	c4	d4	1
a5	b5	c5	d5	2

X_1	X_3	X_4	Y
a1	c1	d1	1
a2	c2	d2	2
a5	c5	d5	2

X_2	X_3	X_4	Y
b1	c1	d1	1
b3	c3	d3	1
b4	c4	d4	1

X_1	X_2	Y
a2	b2	2
a3	b3	1
a5	b5	2

Random forest consists of using randomly selected inputs (bootstrap sample) or combinations of inputs at each node (random subspace) to grow each tree.

↳ sample dr featurenya

variasi banyak karena ambil random features & random data

Misra, S., & Li, H. (2019). Noninvasive fracture characterization based on the classification of sonic wave travel times. *Machine Learning for Subsurface Characterization*, 243.

nentuinnya random

Algorithm 1: Pseudo code for the random forest algorithm

To generate c classifiers:
for i = 1 to c **do**
  Randomly sample the training data D with replacement to produce D_i
  Create a root node, N_i containing D_i
  Call BuildTree(N_i)
end for

BuildTree(N):
if N contains instances of only one class **then**
  return
else
  Randomly select x% of the possible splitting features in N
  Select the feature F with the highest information gain to split on
  Create f child nodes of N, N_1,..., N_f, where F has f possible values (F_1, ..., F_f)
  for i = 1 to f **do**
    Set the contents of N_i to D_i, where D_i is all instances in N that match F_i
    Call BuildTree(N_i)
  end for
end if

ga semua atribut dihitung information gainnya, random cm berp % atribut yg dihitung

Misal ada 10 atr → disuru ambil 50%
→ random 5 atr x itung gainnya → baru pilih best atr

https://www.researchgate.net/publication/335375894_Forecasting_mining_capital_cost_for_open-pit_mining_projects_based_on_artificial_neural_network_approach/figures?lo=1

https://scikit-learn.org/stable/modules/ensemble.html#forest

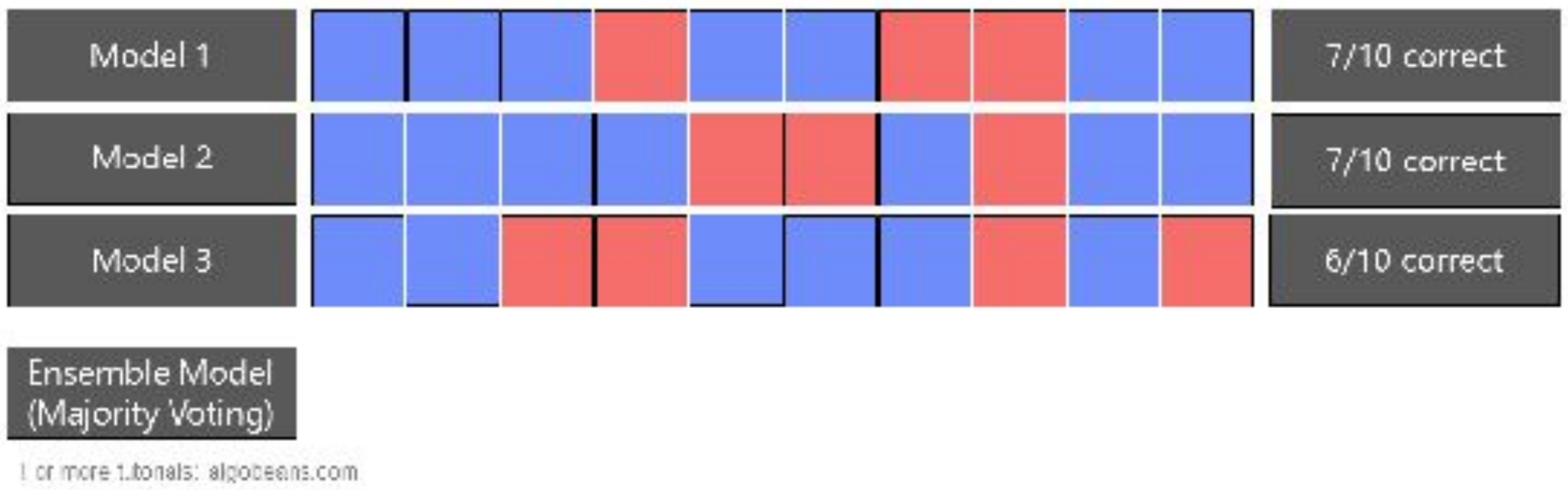
- Each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.
- When splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size `max_features`.
- the scikit-learn implementation combines classifiers by **averaging their probabilistic prediction,** instead of letting each classifier vote for a single class.

bukan majority

Kalo ga ada info di ujian ambil yg kanan

Majority Voting

task regreri → average



The predicted class probabilities of an input sample are computed as the mean predicted class probabilities of the trees in the forest.

https://www.kdnuggets.com/2016/09/reandom-forest-criminal-tutorial.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.predict_proba

Why Random Forest ?

- Ensemble methods have been a very effective tool to improve the performance of multiple existing models (Breiman, 2021)
- Random forest is also well known as one of the most accurate and as a fast learning method independently from the nature of the datasets, as shown by various recent comparative studies (Fernández-Delgado et al. 2014; Caruana and Niculescu-Mizil 2006; Rokach 2016).
- Ensemble learning algorithms like random forests are well suited for medium to large datasets.

data jelek pake ensemble ⟶ hasilnya bagus

Random Forest: d >>n

- When the number of independent variables (d) is larger than the number of observations (n), linear regression and logistic regression algorithms will not run, because the number of parameters to be estimated exceeds the number of observations. Random forest works because not all predictor variables are used at once.

  Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, *20*(1), 3-29.

Random Forest: Summary

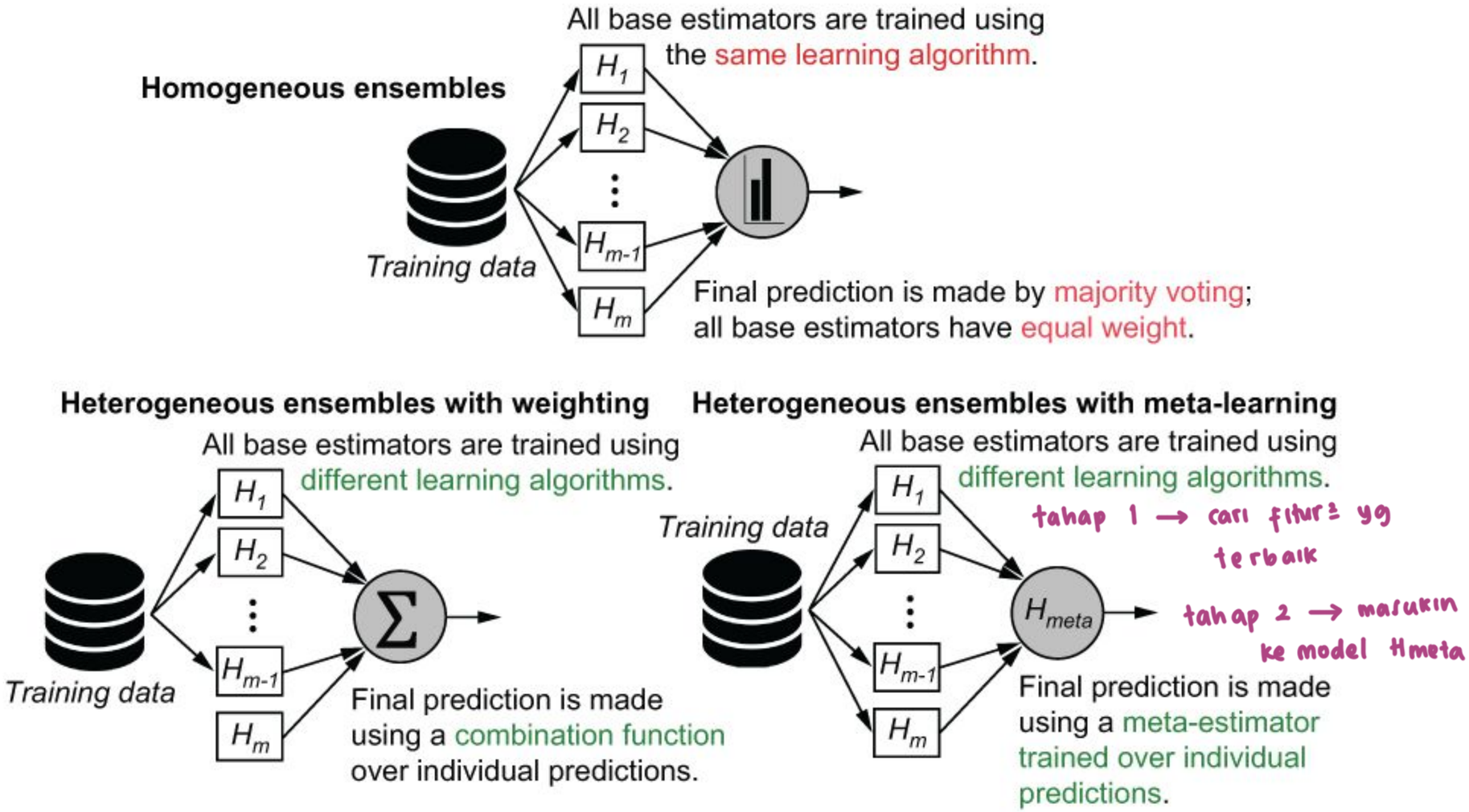
Advantages

- one of the most accurate learning algorithms available
- It runs efficiently on large databases. (parallel)
- It has methods for balancing error in class population unbalanced data sets.

Disadvantages

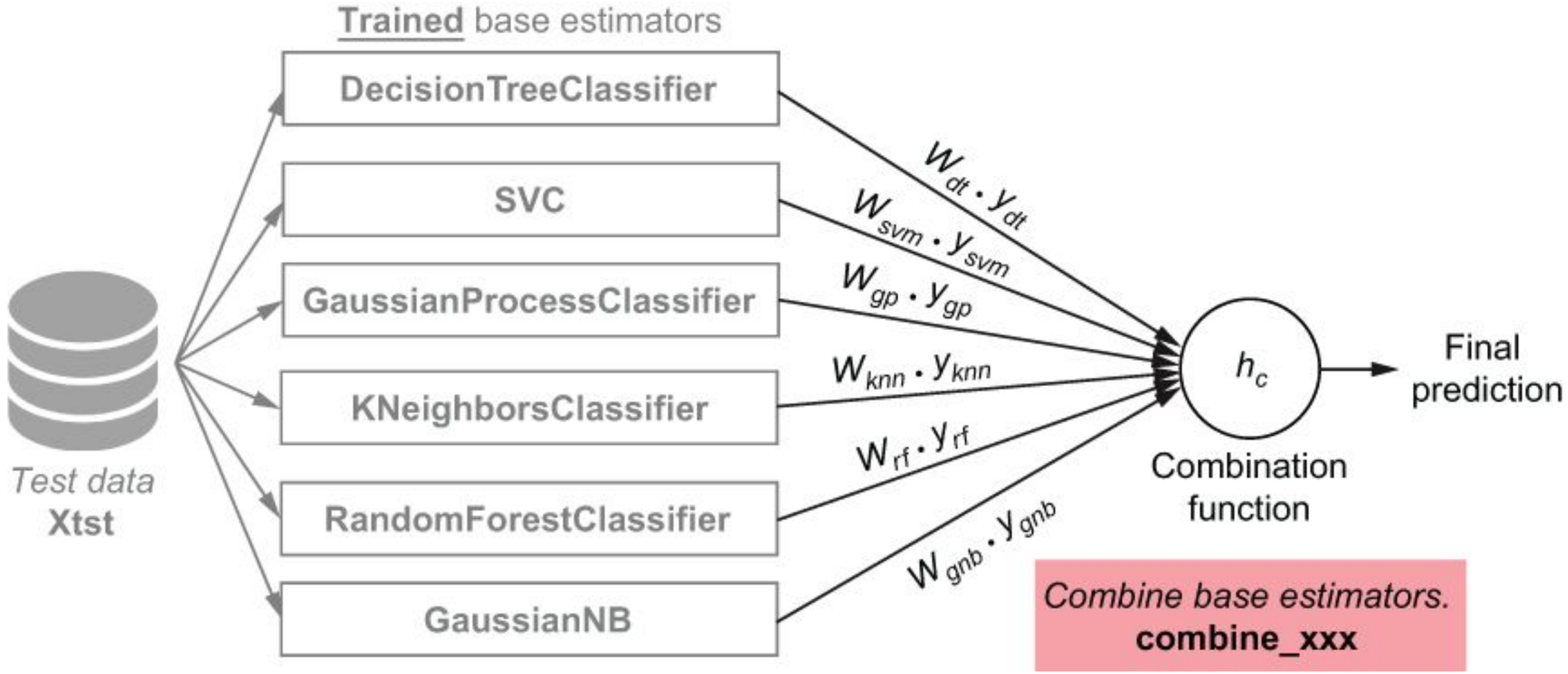
- Random forests have been observed to overfit for some datasets with noisy classification
- Prediction time may be slow

Heterogeneous Parallel Ensembles



https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

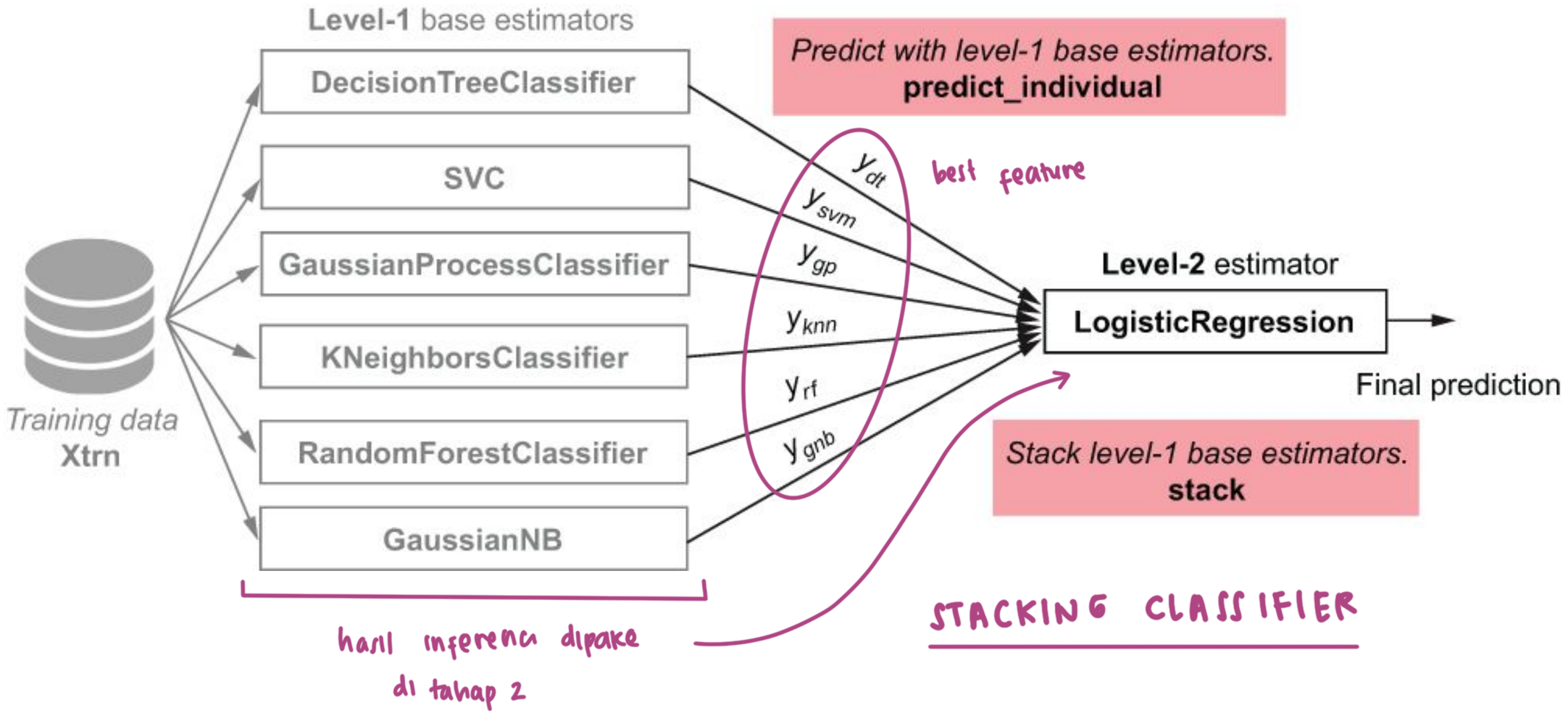
Heterogeneous Ensembles with Weighting



$$y_{final} = w_1 \cdot y_1 + w_2 \cdot y_2 + \cdots + w_m \cdot y_m = \sum_{t=1}^{m} w_t \cdot y_t$$

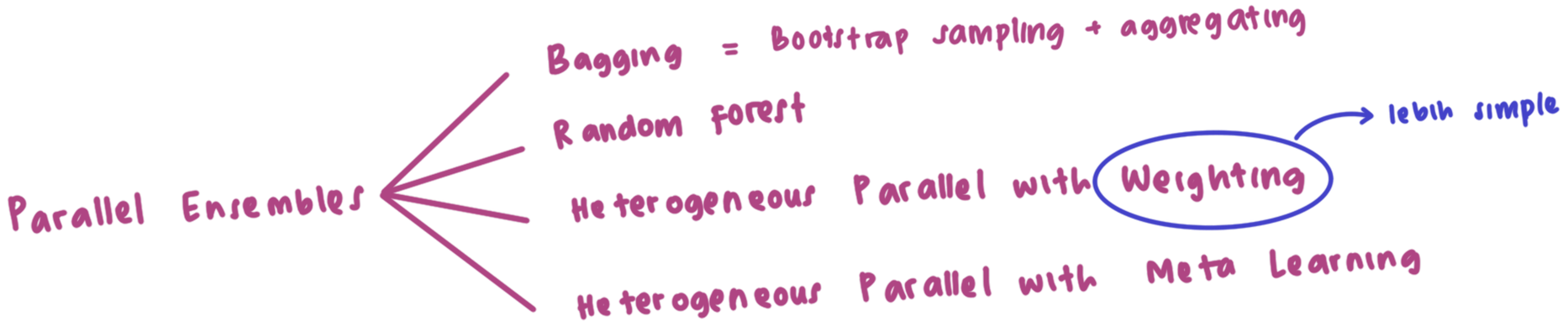
https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

Heterogeneous Ensembles with Meta Learning



https://livebook.manning.com/concept/machine-learning/parallel-ensemble-method

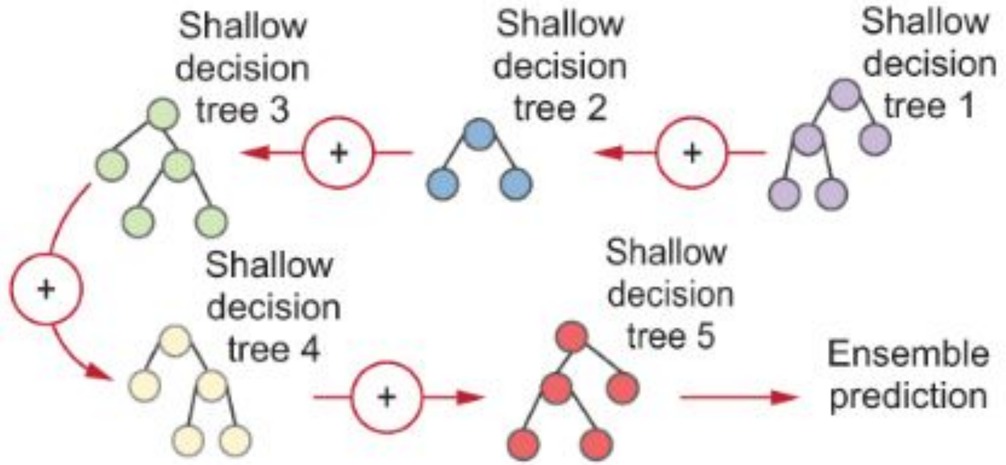
CONCLUSION


Parallel Ensembles
Bagging = Bootstrap sampling + aggregating
Random forest
Heterogeneous Parallel with Weighting
lebih simple
Heterogeneous Parallel with Meta Learning

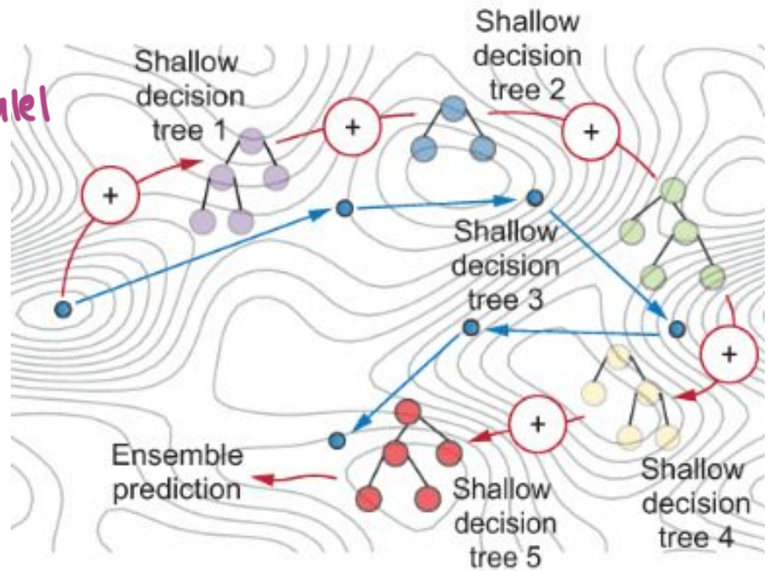
harus weak learner
(yg performanya kurang bagus)

Sequential Ensembles

exploit the *dependence* of base estimators

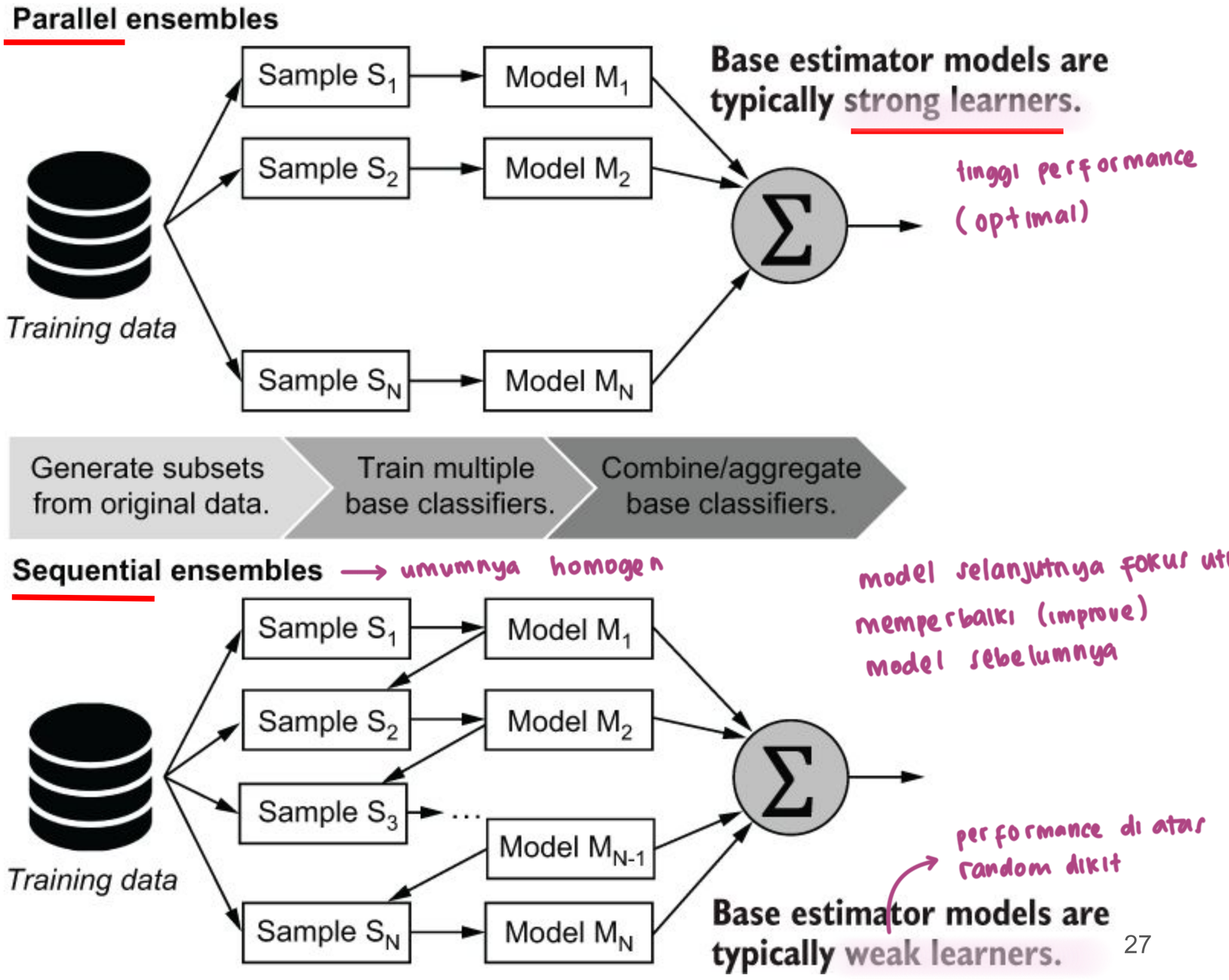


lebih rumit dr paralel



Parallel vs Sequential Ensembles

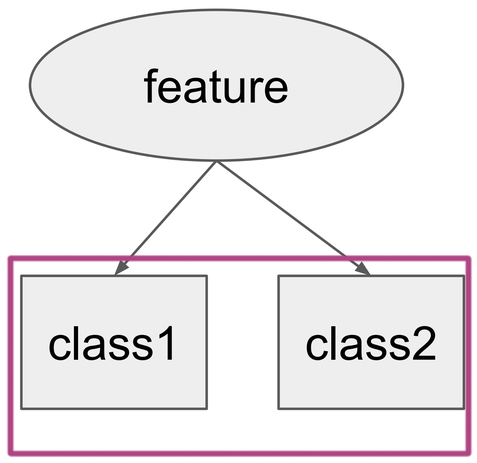
Figure 4.1 Differences between parallel and sequential ensembles: (1) base estimators in parallel ensembles are trained independently of each other, while in sequential ensembles, they are trained to improve on the predictions of the previous base estimator; (2) sequential ensembles typically use weak learners as base estimators.



Boosting

- Sequential ensemble methods such as boosting aim to combine several weak learners into a single strong learner.
  - Boosting literally aims to boost the performance of a collection of weak learners (very simple model that has performance slightly better than random chance)
- Decision trees are often used as base estimators for sequential ensembles. Boosting algorithms typically use decision stumps, or decision trees of depth 1
- Methods: AdaBoost, Gradient Boosting

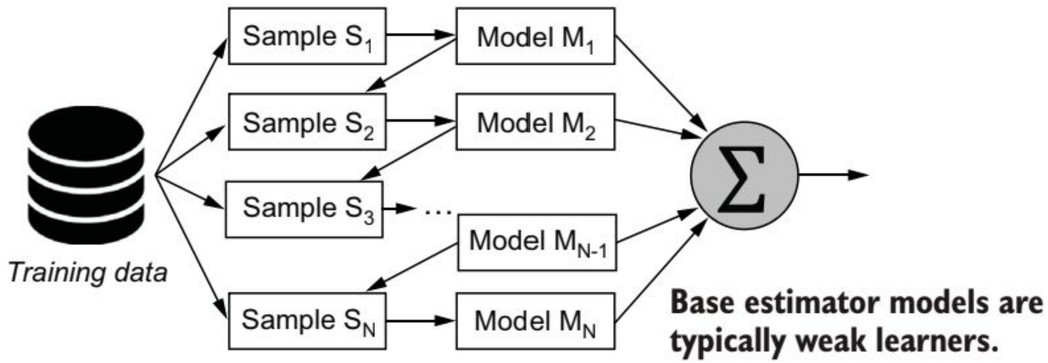
Decision stumps



harusnya ada lg milih fitur
tp lgsg pake majority voting
aja buat pilih kelas

Adaptive Boosting (AdaBoost)

AdaBoost is an adaptive algorithm: at every iteration, it trains a new base estimator h_i that fixes the mistakes made by the previous base estimator h_{i-1} → prioritizes misclassified training examples.



setiap data ada bobotnya

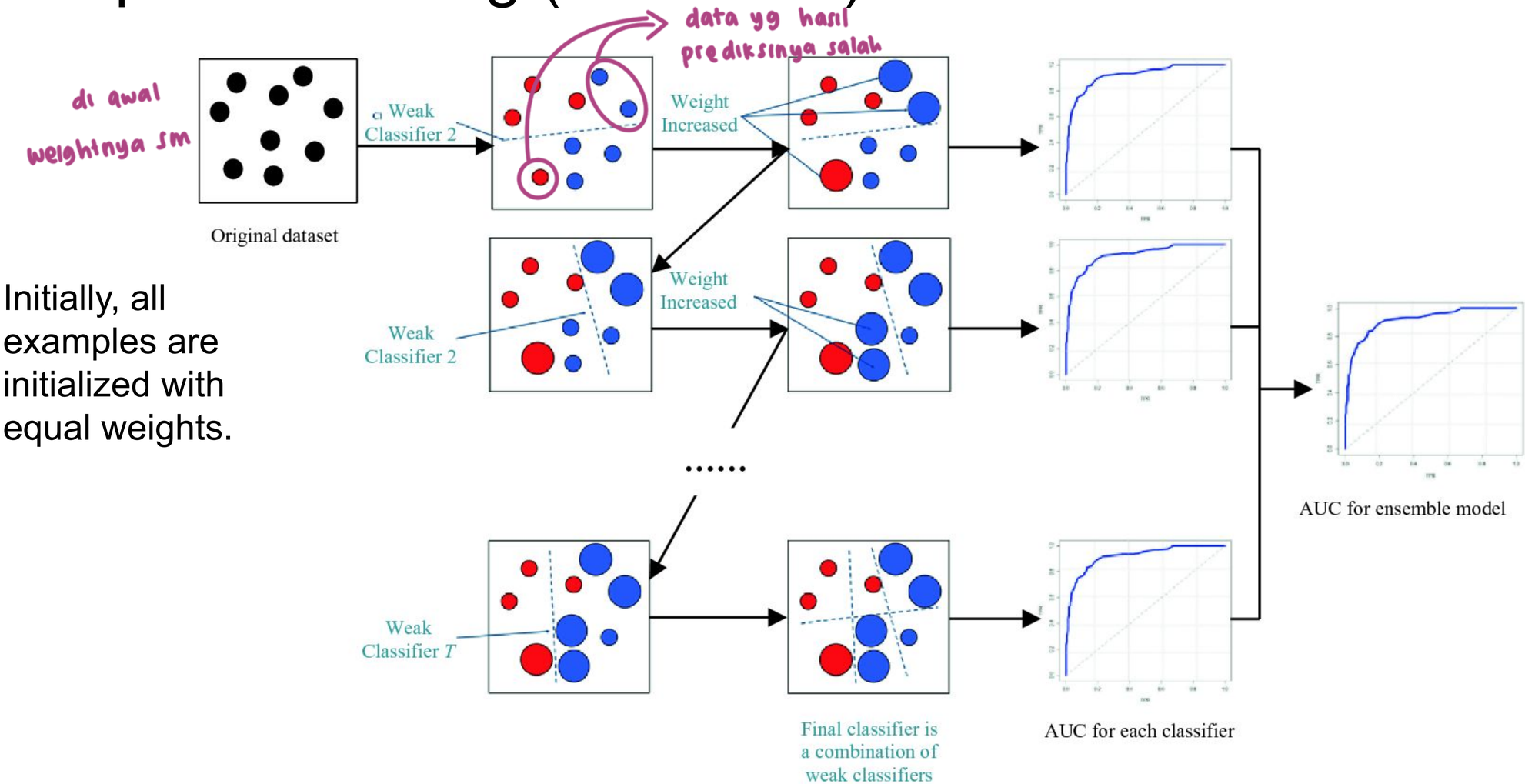
AdaBoost does this by maintaining weights over individual training examples (weights reflect the relative importance of training examples).

The final boosted classifier, H, combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy. This forms a weighted linear combination of base estimators.

Kunapuli, G. (2023). *Ensemble methods for machine learning*. Simon and Schuster.
Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques*. Morgan kaufmann.

Adaptive Boosting (AdaBoost): Illustration

Initially, all examples are initialized with equal weights.



https://datascience.eu/machine-learning/gradient-boosting-what-you-need-to-know/

Instance Weights in Adaptive Boosting: Illustration

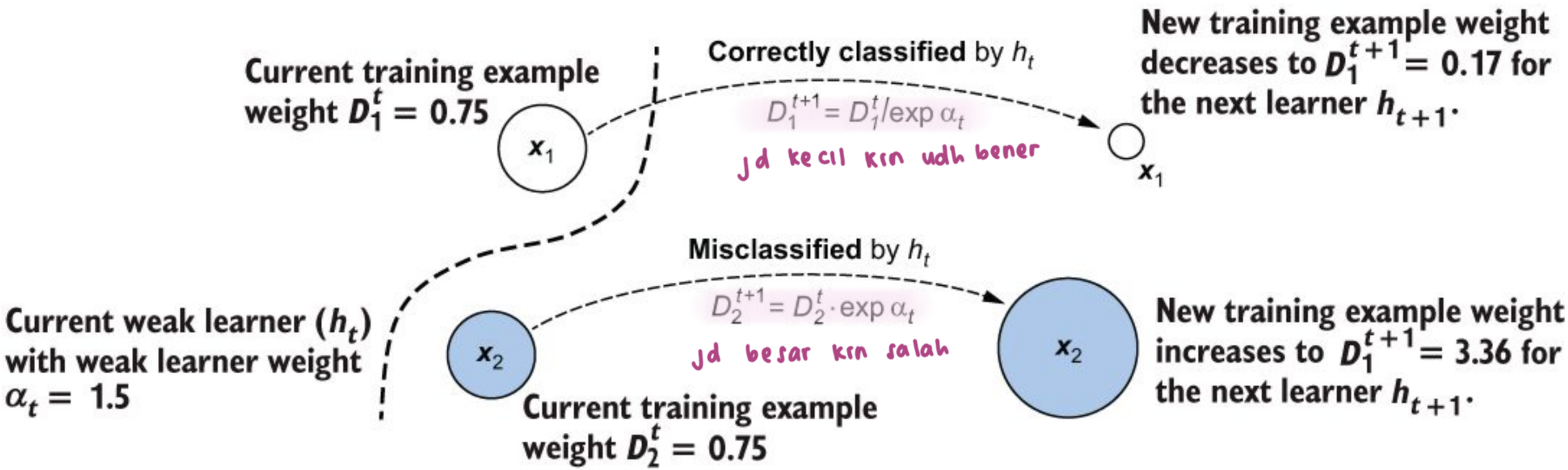


Figure 4.8 In iteration t, two training examples, x_1 and x_2, have the same weights. x_1 is correctly classified, while x_2 is misclassified by the current base estimator h_t. As the goal in the next iteration is to learn a classifier h_{t+1} that fixes the mistakes of h_t, AdaBoost increases the weight of the misclassified example x_2, while decreasing the weight of the correctly classified example x_1.This allows the base-learning algorithm to prioritize x_2 during training in iteration t+1.

Instance Weights in Adaptive Boosting: Illustration

- Iteration t: α_t=1.5
  Weight D_1= D_2=0.75
  $h_t(x_1) = c(x_1)$ #correctly classified, $c(x_1)$: label of x_1
  $h_t(x_2) \neq c(x_2)$ #misclassified, $c(x_2)$: label of x_2

- Update weights:
  $D_1 = D_1/e^{\alpha t} = 0.75/e^{1.5}=0.17$
  $D_2 = D_2.e^{\alpha t} = 0.75.e^{1.5}=3.36$

$$D_i^{t+1} = D_i^t \cdot \begin{cases} e^{\alpha_t}, \text{ if misclassified,} \\ e^{-\alpha_t}, \text{ if correctly classified} \end{cases}$$

AdaBoost: Kunapuli (2023)

Kunapuli, G. (2023). *Ensemble methods for machine learning*. Simon and Schuster.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import numpy as np

def fit_boosting(X, y, n_estimators=10):
    n_samples, n_features = X.shape
    D = np.ones((n_samples, ))
    estimators = []

    for t in range(n_estimators):
        D = D / np.sum(D)

        h = DecisionTreeClassifier(max_depth=1)
        h.fit(X, y, sample_weight=D)

        ypred = h.predict(X)
        e = 1 - accuracy_score(y, ypred,
                               sample_weight=D)
        a = 0.5 * np.log((1 - e) / e)

        m = (y == ypred) * 1 + (y != ypred) * -1
        D *= np.exp(-a * m)

        estimators.append((a, h))

    return estimators
```

Initializes an empty ensemble

Nonnegative weights, initialized to 1

Normalizes the weights so they sum to 1

Trains a weak learner (h_t) with weighted examples

Computes the training error (ε_t) and the weight (α_t) of the weak learner

Updates the example weights: increase for misclassified examples, decrease for correctly classified examples

Saves the weak learner and its weight

gapake sampling with replacement

gada normalisasi

1) Train a weak learner $h_t(\mathbf{x})$ using weighted dataset <xi,yi,Di>*

2) Compute training error e_t dari $h_t(\mathbf{x})$

3) Computer model weight α_t

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

4) Update weight

$$D_i^{t+1} = D_i^t \cdot \begin{cases} e^{\alpha_t}, \text{ if misclassified,} \\ e^{-\alpha_t}, \text{ if correctly classified} \end{cases}$$

Adaptive Boosting (AdaBoost) - Training: Han et al. (2022)

Algorithm: AdaBoost. A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

Input:

- D, a set of d class-labeled training tuples;
- k, the number of rounds (one classifier is generated per round);
- a classification learning scheme.

ada normalisasi

pakai model biasa, ga perlu yg bs handle weighted model

Output: A composite model.

Method:

(1) initialize the weight of each tuple in D to $1/d$;

Initial weights

(2) **for** $i = 1$ to k **do** // for each round:

(3) sample D with replacement according to the tuple weights to obtain D_i;

(4) use training set D_i to derive a model, M_i;

(5) compute $error(M_i)$, the error rate of M_i (Eq. (6.34))

$$error(M_i) = \sum_{j=1}^{d} w_j \times err(\mathbf{X}_j)$$

(6) **if** $error(M_i) > 0.5$ **then**

(7) abort the loop;

(8) **endif**

(9) **for** each tuple in D that was correctly classified **do**

Update weights if correctly classified:
$D_i^{t+1}=D_i^t.error(M_i)/(1-error(M_i))$

(10) multiply the weight of the tuple by $error(M_i)/(1 - error(M_i))$; // update weights

(11) normalize the weight of each tuple.

To normalize a weight, we multiply it by the sum of the old weights, divided by the sum of the new weights.

(12) **endfor**

Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques*. Morgan kaufmann.

Adaptive Boosting (AdaBoost) - Inference: Han et al. (2022)

To use the ensemble to classify tuple, X:

(1) initialize weight of each class to 0;
(2) **for** $i = 1$ to k **do** // for each classifier:
(3) $w_i = log \frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
(4) $c = M_i(X)$; // get class prediction for X from M_i
(5) add w_i to the weight for class c
(6) **endfor**
(7) return the class with the largest weight.

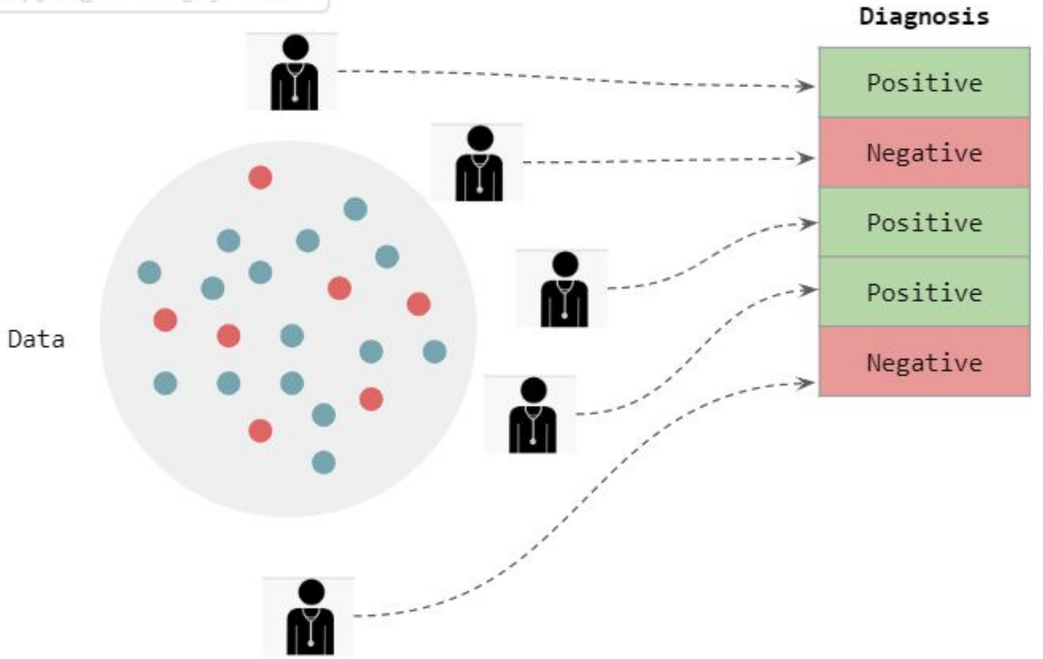
Model weights: $\alpha_t = \log[(1-error(M_i))/error(M_i)]$

ga efisien
hanya nilai inferenci bisa dihitung saat training
kl ini pas train udh slese br hitung nilai inferenci

Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques*. Morgan kaufmann.

AdaBoost Inference: Han et al. (2022)

Base Model	Error(Mi)	wi
M1	e1	log[(1-e1)/e1]
M2	e2	log[(1-e2)/e2]
…	..	…
Mn	en	log[(1-en)/en]



take real weight

M1: + (w1)
M2: - (w2)
M3: + (w3)
M4: + (w4)
M5: - (w5)

To use the ensemble to classify tuple, X:

(1) initialize weight of each class to 0;
(2) **for** $i = 1$ to k **do** // for each classifier:
(3) $w_i = log\frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
(4) $c = M_i(X)$; // get class prediction for X from M_i
(5) add w_i to the weight for class c
(6) **endfor**
(7) return the class with the largest weight.

w+: w1+w3+w4
w-: w2+w5
Class: argmax {w+,w-}

Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques*. Morgan kaufmann.

AdaBoost Inference: Kunapuli (2023)

default pake ini

Base Model	Error(Mi)	ai
M1	e1	a1=log[(1-e1)/e1]
M2	e2	a2=log[(1-e2)/e2]
…	..	…
Mn	en	an=log[(1-en)/en]

M1: + 1
M2: - 1
M3: + 1
M4: + 1
M5: - 1

pred=a1*1+a2*-1+a3*1+a4*1+a5*-1
Class: sign (pred)

Assumption: predict(X) return +1 or -1

weight nya direpresentasi sm +1 & -1

```
def predict_boosting(X, estimators):
    pred = np.zeros((X.shape[0], ))

    for a, h in estimators:
        pred += a * h.predict(X)

    y = np.sign(pred)

    return y
```

Initializes all the predictions to 0

Makes weighted prediction for each example

Converts weighted predictions to –1/1 labels

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

AdaBoost vs Gradient Boosting

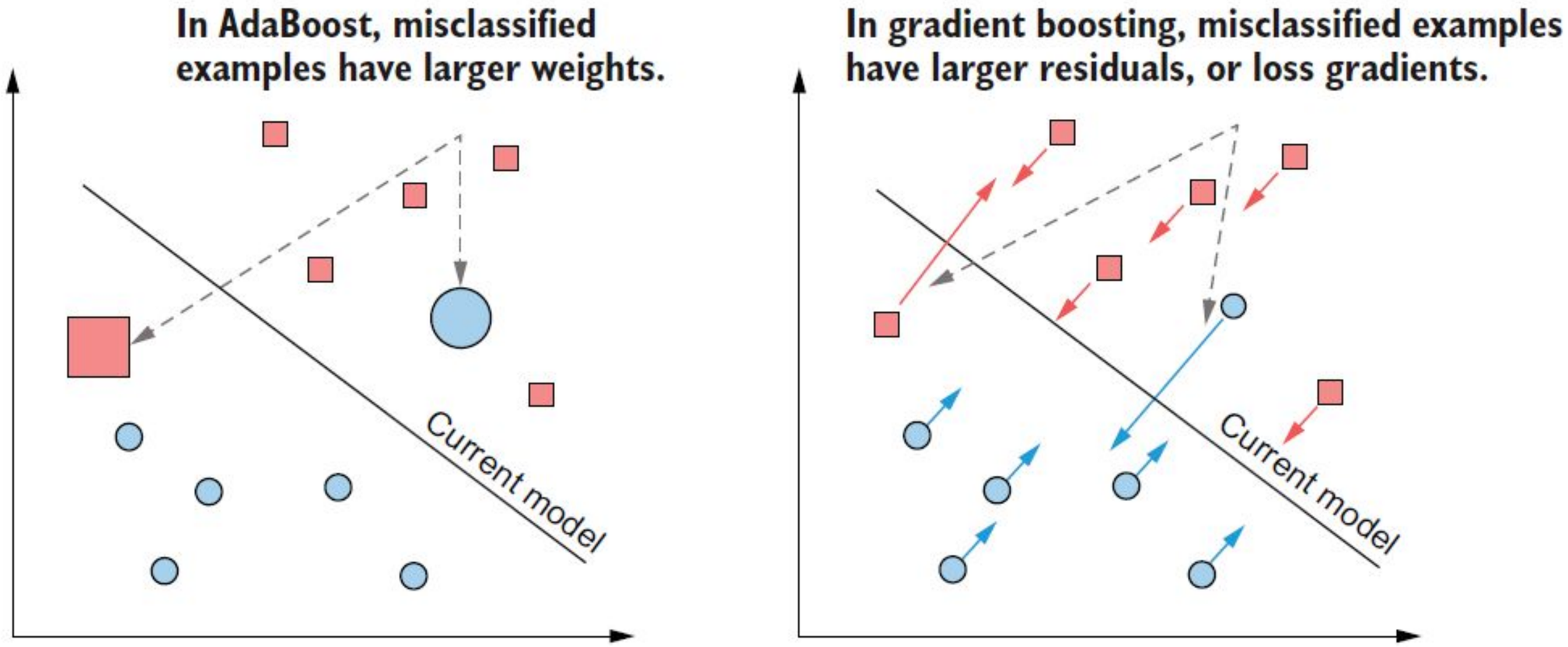


Figure 5.8 Comparing AdaBoost (left) to gradient boosting (right). Both approaches train weak estimators that improve classification performance on misclassified examples. AdaBoost uses weights, with misclassified examples being assigned higher weights. Gradient boosting uses residuals, with misclassified examples having higher residuals. The residuals are nothing but negative loss gradients.

Gradient Boosting
=
Gradient Descent + Boosting

gradient boosting updates the current model with gradient information

gradient boosting trains a weak learner to fix the mistakes made by the previous weak learner.

Hill Climbing Steepest Ascent (naik)	Gradient Descent (turun)
local search	state continuous
value (state) → diskrit	linear regression
better neighbor	back propagation

bisa secara iteratif pake $Loss(\vec{b}) = \sum (y_i - \hat{y}_i)^2$

bisa cari successor based on turunan parsial dgn bobot $<b_0, b_1, \ldots, b_n>$

→ buat koefisien gradien (ga iteratif)

Least square (LS)

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

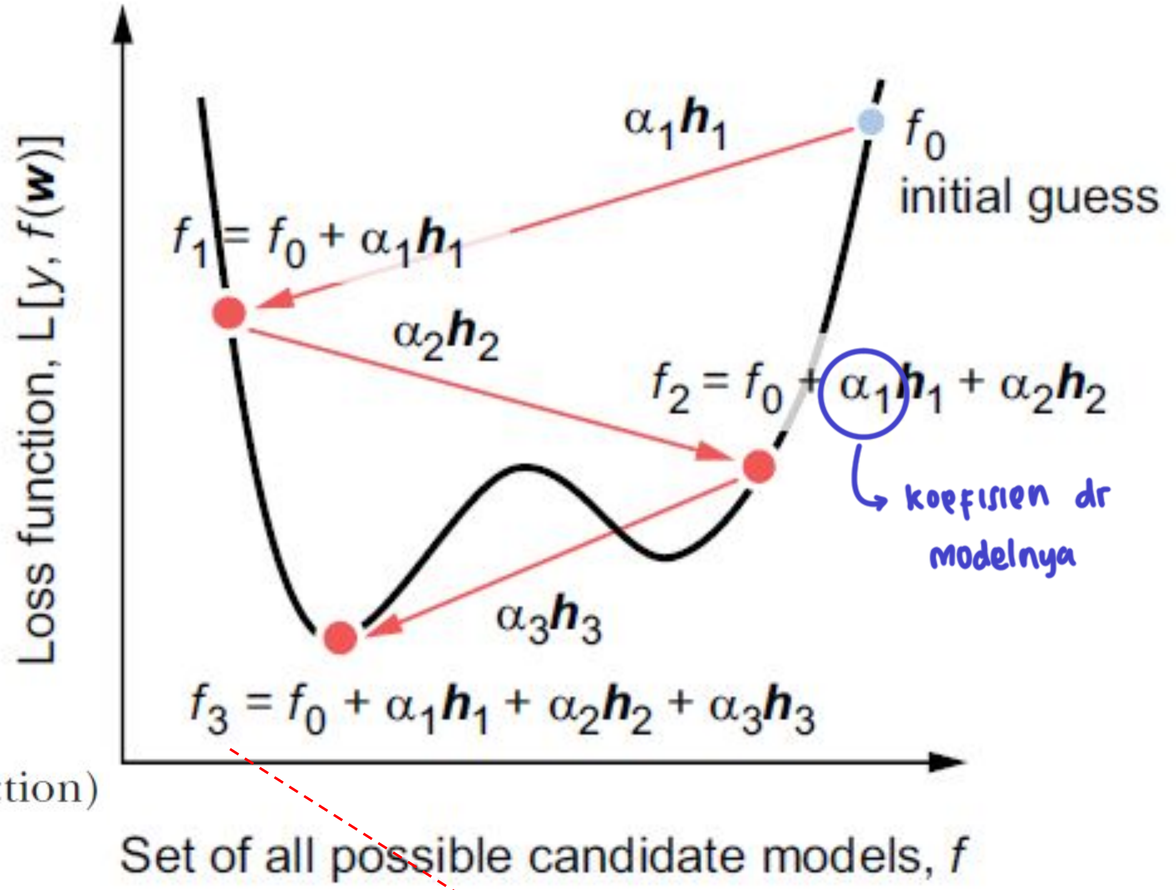
$$\hat{y} = b_0 + b_1 x_1 + \ldots + b_n x_n$$

$$< b_0, b_1, \ldots, b_n >$$

iterative

Gradient Boosting

Each iteration additively updates the current model with the new approximate weak gradient estimate (the regression tree, ht).



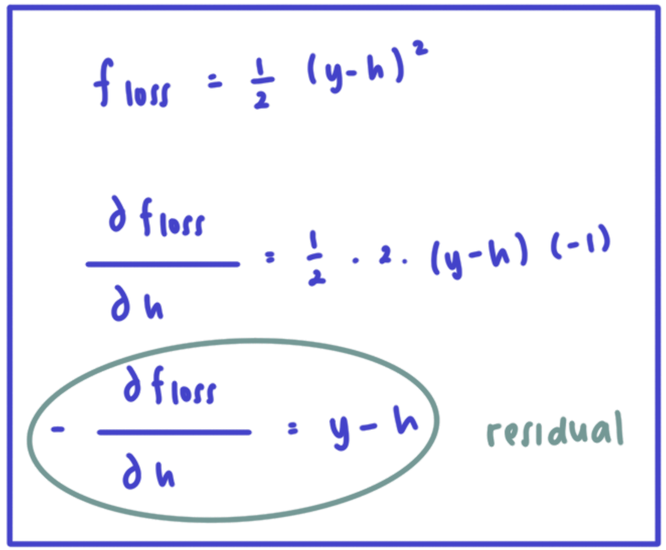
new model = old model + (step length) * (update direction)

$$F_{t+1}(x) = F_t(x) + \alpha_t \cdot h_t(x)$$

$$F_{t+1}(x) = F_0(x) + \alpha_1 \cdot h_1(x) + \alpha_2 \cdot h_2(x) + \cdots + \alpha_{t-1} \cdot h_{t-1}(x) + \alpha_t \cdot h_t(x)$$

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Gradient Boosting: Residual



- residual(y,ŷ)=residual(y,h(x))=y-h(x)
- Squared loss function (SSE) to model h:

  $f_{loss}(y,\hat{y})=(y-h(x))^2/2$

  $gradient(y,\hat{y})=\partial f_{loss}/\partial h\ (y,h(x))=-(y-h(x))$

  $$\text{gradient}(\text{true, predicted}) = \frac{\partial f_{\text{loss}}}{\partial h}(y, h(x)) = -(y - h(x))$$

- Residual is the negative gradient:

  $residual(y,\hat{y})=-\partial f_{loss}/\partial h=y-h(x)$ # for squared loss

Gradient Boosting: Basic Algorithm

initialize: F = f_0, some constant value
for t = 1 to T:
1. compute the negative residuals for each example, $r_i^t = -\frac{\partial L}{\partial F}(\boldsymbol{x}_i)$
2. fit a weak decision tree regressor $h_t(\boldsymbol{x})$ using the training set $(\boldsymbol{x}_i, r_i)_{i=1}^n$
3. compute the step length (α_t) using line search
4. update the model: $F_t = F + \alpha_t \cdot h_t(\boldsymbol{x})$

Gradient Boosting vs Gradient Descent

- instead of using the negative gradient, we use an approximate gradient trained on the negative residuals
- instead of checking for convergence, the algorithm terminates after a finite, maximum number of iterations T.

Gradient Boosting: Training

```
from scipy.optimize import minimize_scalar
from sklearn.tree import DecisionTreeRegressor

def fit_gradient_boosting(X, y, n_estimators=10):
    n_samples, n_features = X.shape
    n_estimators = 10
    estimators = []
    F = np.full((n_samples, ), 0.0)

    for t in range(n_estimators):
        residuals = y - F
        h = DecisionTreeRegressor(max_depth=1)
        h.fit(X, residuals)

        hreg = h.predict(X)
        loss = lambda a: np.linalg.norm(
                              y - (F + a * hreg))**2
        step = minimize_scalar(loss, method='golden')
        a = step.x

        F += a * hreg
        estimators.append((a, h))

    return estimators
```

Gets dimensions of the data set

Initializes an empty ensemble

Predicts the ensemble on the training set

Computes residuals as negative gradients of the squared loss

Fits weak regression tree (h_t) to the examples and residuals

Gets predictions of the weak learner, h_t

Sets up the line search problem

Finds the best step length using the golden section search

Updates the ensemble predictions

Updates the ensemble

Gradient Boosting: Inference

```
def predict_gradient_boosting(X, estimators):
    pred = np.zeros((X.shape[0], ))

    for a, h in estimators:
        pred += a * h.predict(X)

    y = np.sign(pred)

    return y
```

Initializes all the predictions to 0

Aggregates individual predictions from each regressor

Converts weighted predictions to –1/1 labels

Gradient Boosting: Han et al. (2022)

Algorithm: Gradient Tree Boosting for Regression.

Input:

- D, a set of n training tuples $\{(x_1, y_1), ..., (x_n, y_n)\}$, where x_i is the attribute vector of the ith training tuple and y_i is its true target output value;
- k, the number of rounds (one base regression model is generated per round);
- a differential loss function Loss $= \sum_{i=1}^{n} L(y_i, F(x_i))$.

Output: A composite regression model $F(x)$.

Method:

(1) initialize the regression model $F(x) = \frac{\sum_{i=1}^{n} y_i}{n}$;

Outputs a constant (i.e., the average output of all training tuples).

(2) **for** $t = 1$ to k **do** // construct a new weak learner $M_t(x)$ for each round:

(3) **for** $i = 1$ to n //each training tuple:

(4) calculate $\hat{y}_i = F(x_i)$; //predicted value by the current model $F(x)$

(5) calculate the negative gradient $r_i = -\frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}$;

(6) **endfor**

(7) fit a regression tree model $M_t(x)$ for the training set $\{(x_1, r_1), ..., (x_n, r_n)\}$;

(8) update the composite regression model $F(x) \leftarrow F(x) + M_t(x)$.

(9) **endfor**

the negative gradient ri is treated as the targeted output value of the ith training tuple.

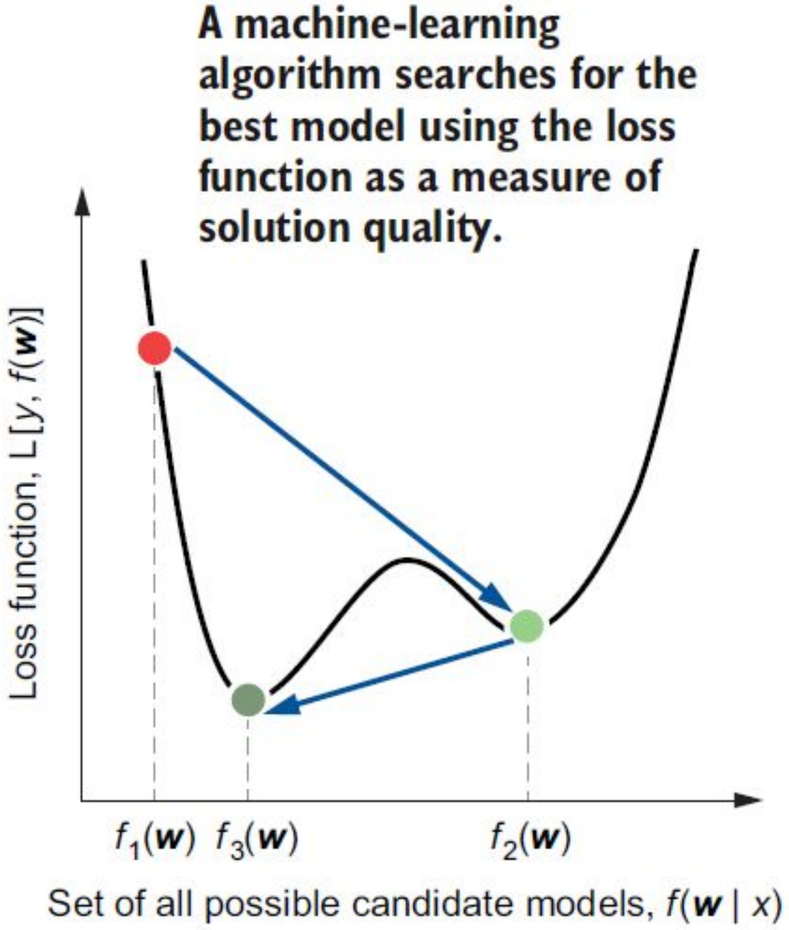
$$L(y_i, F(x_i)) = \frac{1}{2}(y_i - \hat{y}_i)^2$$

$$r_i = y_i - \hat{y}_i$$

Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques*. Morgan kaufmann.

Loss Function: Additional Slides

- Loss functions explicitly measure the fit of a model on a data set. Most often, we measure loss with respect to the true labels, by quantifying the error between the predicted and true labels.
- Given a loss function, training is the search for the optimal model that minimizes the loss, as illustrated in this figure.



An optimization procedure for finding the best model. Machine-learning algorithms search for the best model among all possible candidate models. The optimization procedure sequentially identifies increasingly better models f1, f2, and the final model, f3.

Gradient-based Optimization: Additional Slides

Many methods attempt to use the gradient of the landscape to find an optimum value. The gradient of the objective function is a vector ∇f that gives the magnitude and direction of the steepest slope.

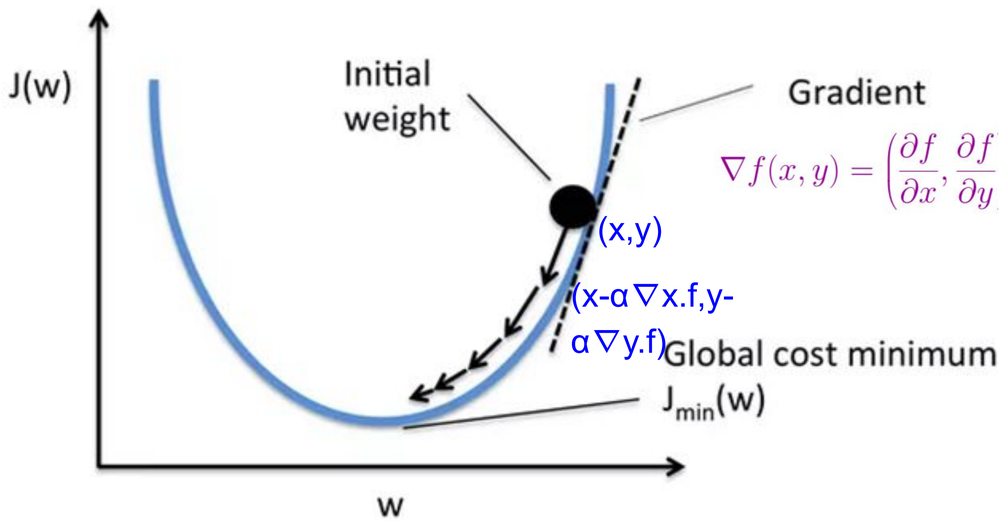
Given objective function f(x,y) to be minimized, and current state is (x,y), gradient is a vector:

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

Use gradient to set successor state:

$x \leftarrow x - \alpha \nabla x.f$

$y \leftarrow y - \alpha \nabla y.f$



https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c

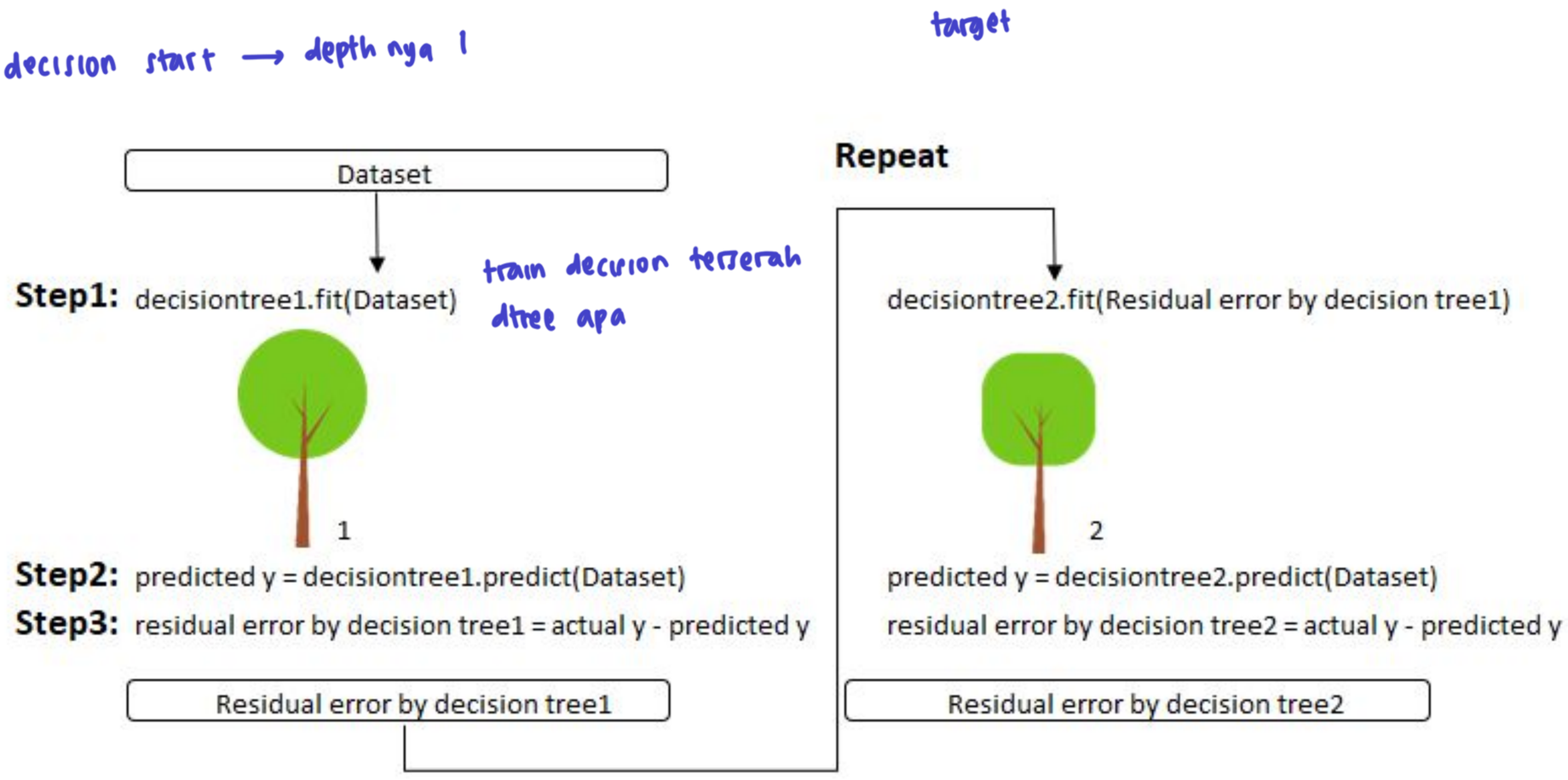
The basic problem is that if α is too small, too many steps are needed; if α is too large, the search could overshoot the maximum.

Eksplorasi Mandiri

kerjain !!!
sp tau ditanya di ujian

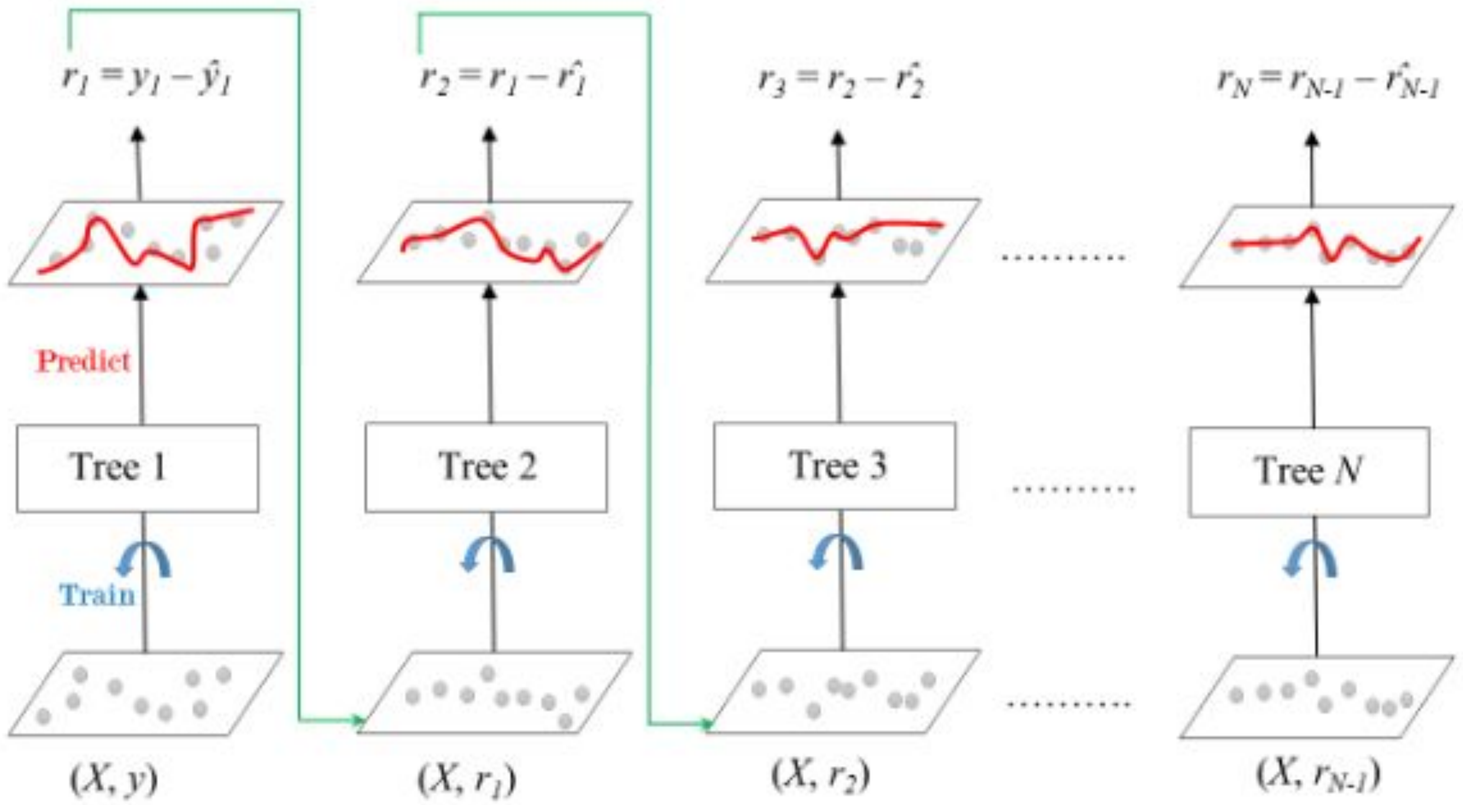
- Eksplorasi library `RandomForestClassifier`, `BaggingClassifier`, `AdaBoostClassifier`, `LightGBM` dari library `sklearn.ensemble`
- Ikuti hands-on pada subbab **2.5 Case study: Breast cancer diagnosis** buku teks Kunapuli, G. (2023). *Ensemble methods for machine learning*. Simon and Schuster

Gradient Boosting: Use Residual Error



https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725

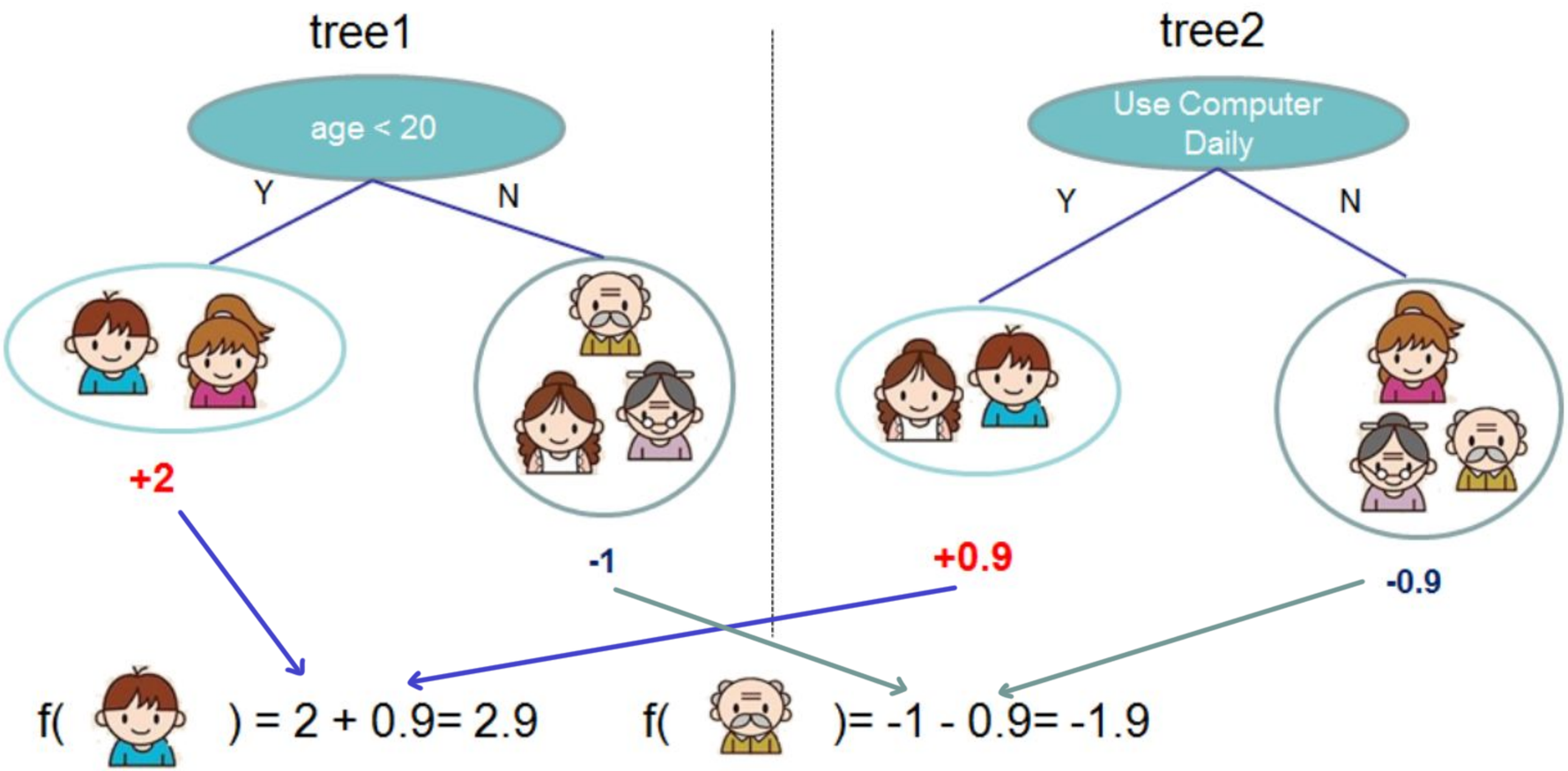
Gradient Boosting: Use Residual Error




```
y(pred) = y1 + (η*r1) + (η*r2) + ..... + (η*rN)
```

https://www.geeksforgeeks.org/ml-gradient-boosting/

Gradient Boosting Prediction: Example



https://xgboost.readthedocs.io/en/latest/tutorials/model.html

buat contoh $\alpha_1 h_1 + \alpha_2 h_2$

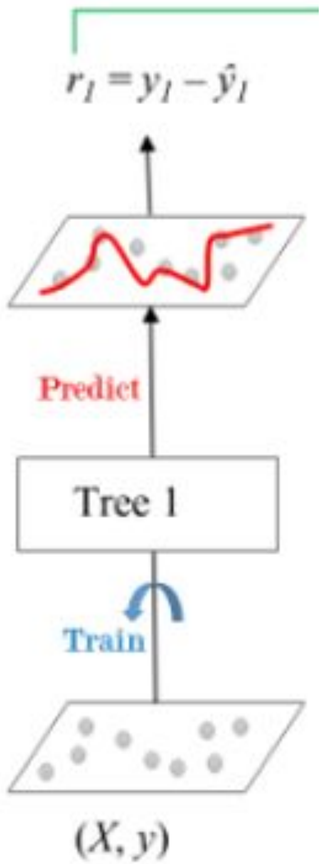
XGBoost Example: ***Predict Person's Age***

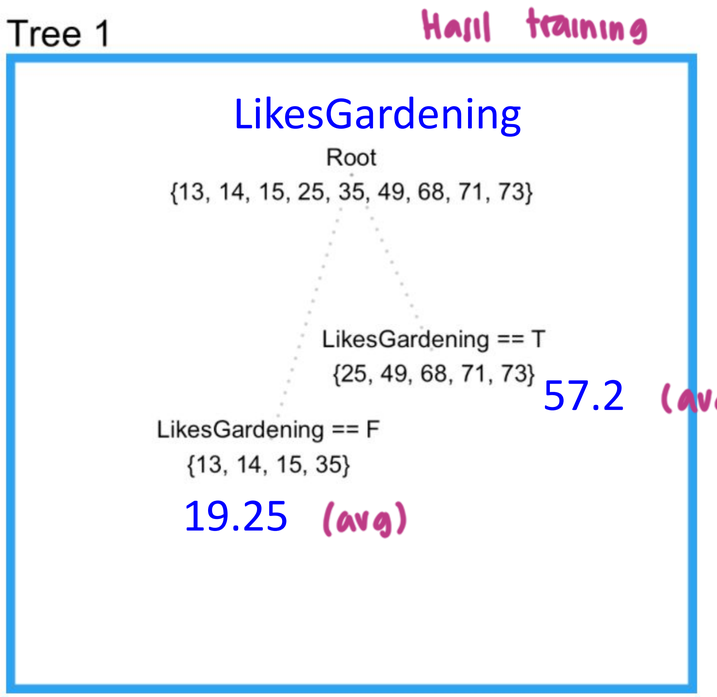
target (y)

PersonID	Age	LikesGardening	PlaysVideoGames	LikesHats
1	13	FALSE	TRUE	TRUE
2	14	FALSE	TRUE	FALSE
3	15	FALSE	TRUE	FALSE
4	25	TRUE	TRUE	TRUE
5	35	FALSE	TRUE	TRUE
6	49	TRUE	FALSE	FALSE
7	68	TRUE	TRUE	TRUE
8	71	TRUE	FALSE	FALSE
9	73	TRUE	FALSE	TRUE

https://www.gormanalysis.com/blog/gradient-boosting-explained/#gradient-descent

XGBoost: Initialization $F_0(x)$=Tree1, Predict, & r1





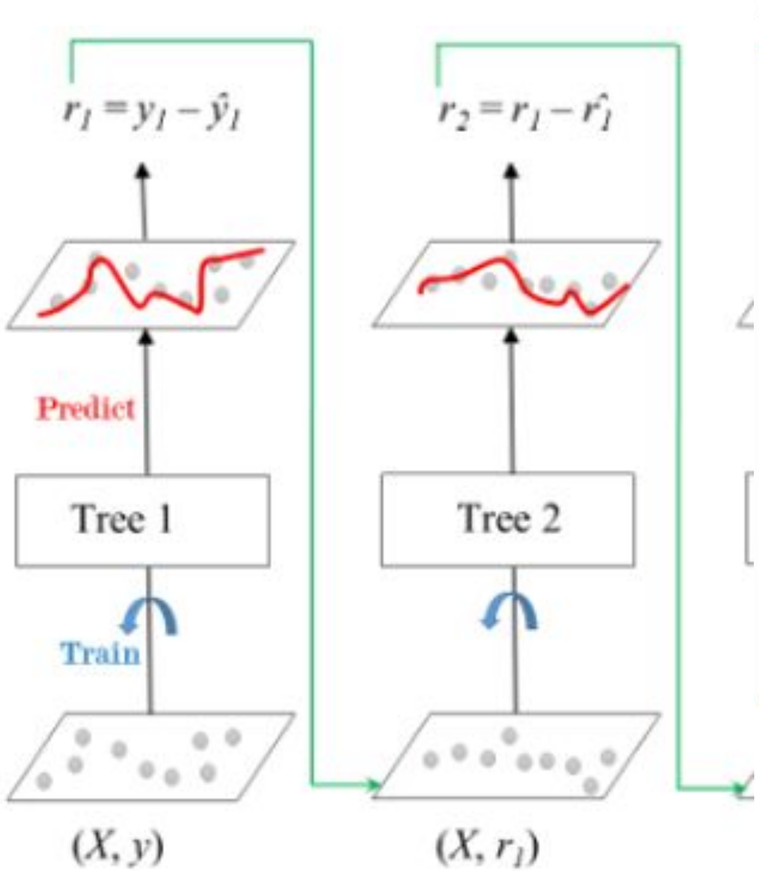
age - Tree1 res

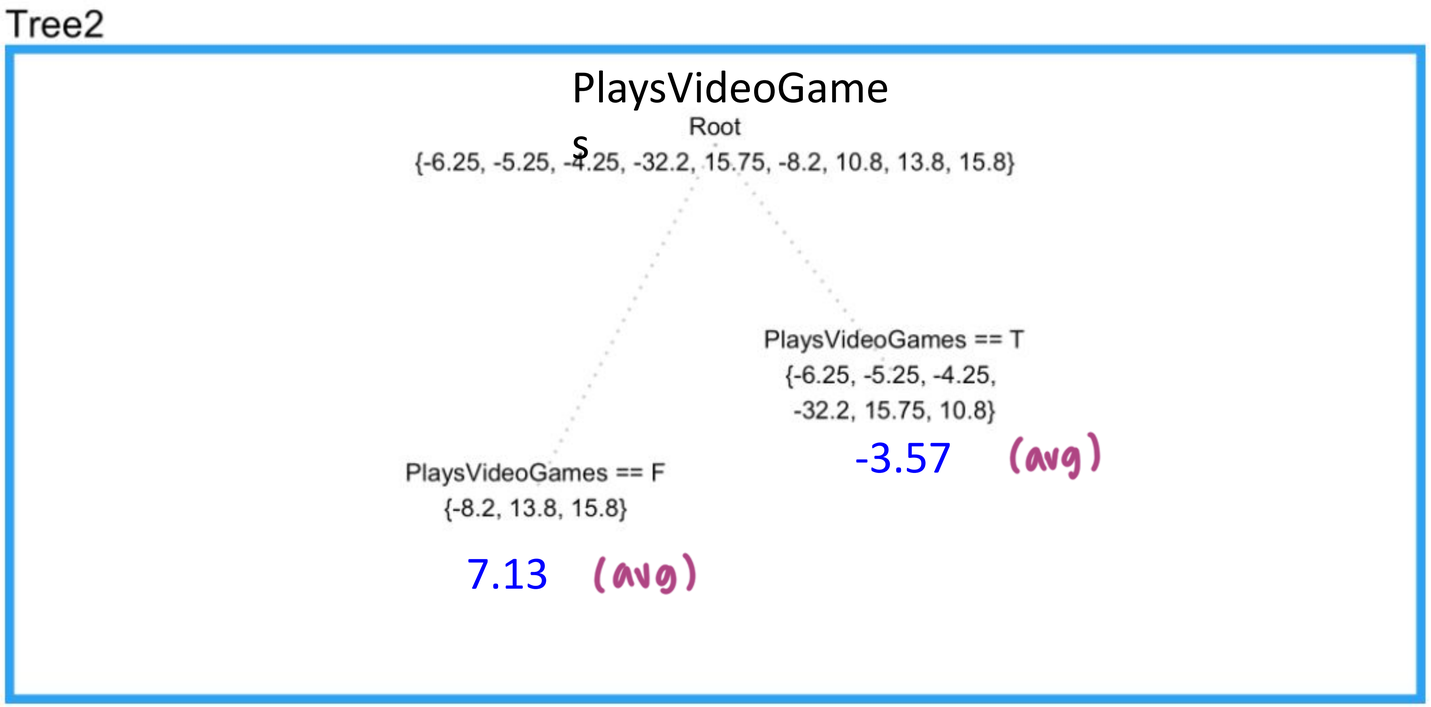
next target pake ini, gapake age lg

PersonID	Age	Tree1 Prediction	Tree1 Residual
1	13	19.25	-6.25
2	14	19.25	-5.25
3	15	19.25	-4.25
4	25	57.20	-32.20
5	35	19.25	15.75
6	49	57.20	-8.20
7	68	57.20	10.80
8	71	57.20	13.80
9	73	57.20	15.80

https://www.gormanalysis.com/blog/gradient-boosting-explained/#gradient-descent

XGBoost: Tree2 □ Combined Prediction

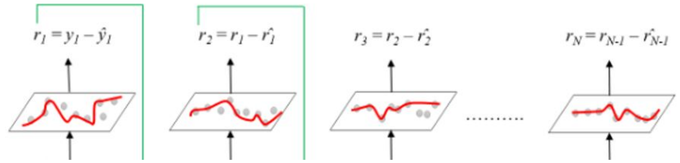




https://www.gormanalysis.com/blog/gradient-boosting-explained/#gradient-descent

XGBoost:
$F_0(x)$=Tree1.predict+Tree2.predict

default nya CART → Classification & Regression Tree



PersonID	Age	Tree1 Prediction	Tree1 Residual	Tree2 Prediction	Combined Prediction	Final Residual
1	13	19.25	-6.25	-3.57	15.68	2.68
2	14	19.25	-5.25	-3.57	15.68	1.68
3	15	19.25	-4.25	-3.57	15.68	0.68
4	25	57.20	-32.20	-3.57	53.63	28.63
5	35	19.25	15.75	-3.57	15.68	-19.32
6	49	57.20	-8.20	7.13	64.33	15.33
7	68	57.20	10.80	-3.57	53.63	-14.37
8	71	57.20	13.80	7.13	64.33	-6.67
9	73	57.20	15.80	7.13	64.33	-8.67

https://www.gormanalysis.com/blog/gradient-boosting-explained/#gradient-descent

Questions ?

Contoh soal

1. (Bobot 15) Diberikan prediksi 5 base-classifier (Decision Tree) dari model ensemble sebagai berikut. Jelaskanlah penentuan hasil prediksi model ensemble jika menggunakan skema:
(a) bagging; (b) Random Forest; (c) AdaBoost.
Jika diperlukan bobot model dalam perhitungan, gunakanlah bobot berikut ini terurut model 1..5 yaitu 0.15, 0.25, 0.3, 0.25, 0.05. Jawaban prediksi tanpa penjelasan caranya tidak akan dinilai.

Model 1	Model 2	Model 3	Model 4	Model 5
Positif	Positif	Negatif	Negatif	Positif

Jawab:
Bagging: argmax{positif: 3, negatif: 2} = positif
RandomForest: argmax{positif: 3, negatif: 2} = positif
AdaBoost: positif 0.15 => positif 0.40 => positif 0.40, negatif 0.3 => positif 0.4, negatif 0.55 => positif 0.45, negatif 0.55. Hasil: negatif

CPMK 1: Mampu melakukan training dalam pembangunan model pembelajaran mesin dengan berbagai teknik supervised learning.

2. (Bobot 10) Diberikan dataset dengan 10 instances 2 atribut, hasil pemodelan SVM

18:42

6. (Bobot 10) Terdefinisi model Random Forest 2 base-classifier (cls1 dan cls2) untuk klasifikasi 3 kelas c1-c2-c3. Pada proses klasifikasi suatu input data, cls 1 berhenti pada suatu daun dengan rasio instances pada daun tersebut adalah c1:c2:c3=3:0:2, sedangkan cls2 berhenti pada daun dengan rasio instances c1:c2:c3=4:3:3. Jelaskanlah hasil hard-classification dan soft-classification sebagai hasil inferensi Random Forest tersebut.

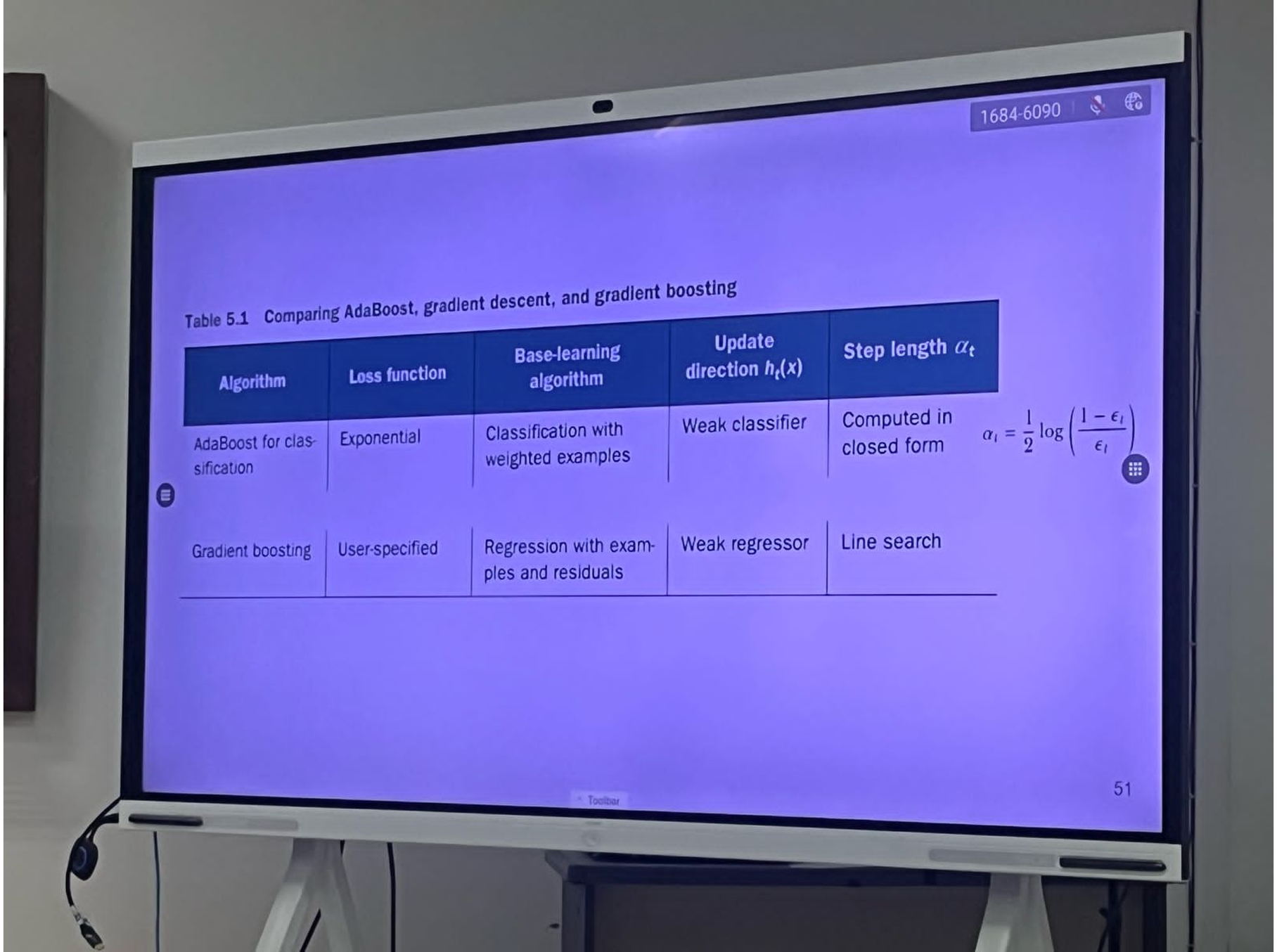
Jawab:

Hard classification:

- Cls1: c1 (c1:c2:c3=3:0:2)
- Cls2: c1 (c1:c2:c3=4:3:3)
- Hard classification (majority voting): c1

Soft classification:

- Cls1: [0.6,0,0.4]
- Cls2: [0.4,0.3,0.3]
- Soft classification: [0.5,0.15,0.35]


1684-6090
Table 5.1 Comparing AdaBoost, gradient descent, and gradient boosting
Algorithm
Loss function
Base-learning algorithm
Update direction $h_t(x)$
Step length α_t
AdaBoost for classification
Exponential
Classification with weighted examples
Weak classifier
Computed in closed form
$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
Gradient boosting
User-specified
Regression with examples and residuals
Weak regressor
Line search
51

Gradient Boosting: alpha value

```
from scipy.optimize import minimize_scalar
from sklearn.tree import DecisionTreeRegressor

def fit_gradient_boosting(X, y, n_estimators=10):
    n_samples, n_features = X.shape
    n_estimators = 10
    estimators = []
    F = np.full((n_samples, ), 0.0)

    for t in range(n_estimators):
        residuals = y - F
        h = DecisionTreeRegressor(max_depth=1)
        h.fit(X, residuals)

        hreg = h.predict(X)
        loss = lambda a: np.linalg.norm(
                              y - (F + a * hreg))**2
        step = minimize_scalar(loss, method='golden')
        a = step.x

        F += a * hreg
        estimators.append((a, h))

    return estimators
```

Gets dimensions of the data set

Initializes an empty ensemble

Predicts the ensemble on the training set

Computes residuals as negative gradients of the squared loss

Fits weak regression tree (h_t) to the examples and residuals

Gets predictions of the weak learner, h_t

Sets up the line search problem

Finds the best step length using the golden section search

Updates the ensemble predictions

Updates the ensemble

```
In [31]:
1 from scipy.optimize import minimize_scalar
2 def f(x):
3     return 2*(x**2)+4*x-6
4 result = minimize_scalar(f)
5 x_min = result.x
6 x_min #x=-b/2a=-4/4=-1
```

```
Out[31]: -0.9999999851900002
```