Ensemble Classification Tree

Bagus Sartono
Departemen Statistika – IPB University

IPB University
Bogor Indonesia

Bagus Sartono

Pengalaman Kerja

- 2000 – sekarang, Dosen di IPB (Departemen Statistika)
- 2002 – 2008, Statistician – PT Ganesha Cipta Informatika
- 2013, Technical Advisor – MarkPlus Insight
- 2013, Consultant – CIFOR
- 2014, Project Leader – PT Trans Intra Asia
- 2015 – 2017, Data Science Advisor – Starcore Analytics
- 2018, Consultant – IACCB

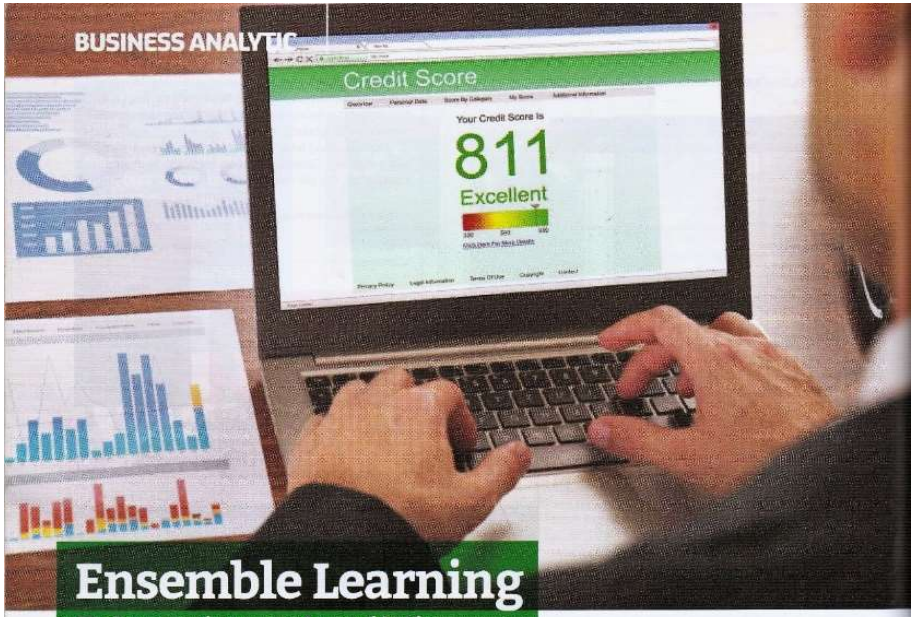
Pengalaman Lainnya

- Trainer bidang Statistika, Data Mining, Analitika di berbagai instansi, antara lain Bank Mandiri, Bank Syariah Mandiri, Bank Sinar Mas, Bank Danamon, Bank Indonesia, OJK, Telkomsel, Bank Permata, CIMB Niaga, LIPI, Kementerian Pertanian, Kementerian Keuangan
- Tenaga ahli di beberapa kegiatan kajian, antara lain Kementerian Pendidikan dan Kebudayaan, BPJS, Kementerian Keuangan, OJK, Bank Indonesia, LIPI, Astra Digital.
- Penulis rubrik Business Analytics di majalah InfoKomputer

Agenda

- Pengantar Pemodelan Prediktif Ensemble
- Bagging
  - Algoritma dan Implementasinya
  - Breiman, Leo (1996). "Bagging predictors". Machine Learning. 24 (2): 123–140
- Random Forest
  - Algoritma dan Implementasinya
  - Breiman L (2001). "Random Forests". Machine Learning. 45 (1): 5–32.
- Boosting
  - Algoritma dan Implementasinya
  - Freund, Yoav; Schapire, Robert E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting". Journal of Computer and System Sciences. 55: 119–139.
- Pendekatan ensemble untuk penanganan data kelas tidak seimbang
  - EasyEnsemble, RUS-Boost

Pengantar Pemodelan Ensemble



Ensemble Learning

Primadona Analitik di Masa Depan

Salah satu aktivitas penting dalam proses analitik adalah menghasilkan model prediktif, yaitu suatu model yang diharapkan dapat memberikan prediksi yang sangat baik terhadap kejadian di masa mendatang.

Naskah:
Bagus Sartono
Dosen di Departemen Statistika FMIPA IPB
bagusco@gmail.com, bagusco@ipb.ac.id
Ilustrasi Foto:
thenmerill.com

Model Prediktif

Model prediktif tersebut antara lain diperlukan oleh bank dan perusahaan pembiayaan dalam bentuk *credit scoring model*. Tujuannya, memperkirakan apakah seseorang yang mengajukan aplikasi pinjaman akan macet kreditnya atau tidak. Tentu saja, prediksi kreditnya tersebut perlu dilakukan jauh hari sebelum diberikan keputusan apakah aplikasinya ditolak atau diterima. Mereka yang diprediksi akan memiliki peluang besar untuk gagal bayar akan memperoleh skor kecil berdasarkan model yang dibangun. Sebaliknya, yang diprediksi akan mampu membayar dengan lancar diberi skor besar oleh model.

Model-model yang serupa juga diperlukan oleh banyak perusahaan berbasis *telemarketing* yang memerlukan *short-list* calon pelanggan untuk dihubungi dan ditawari produk. *Short-list* tersebut umumnya diperoleh dari *list* yang sangat panjang dan memuat banyak nama individu. Perusahaan memerlukan model prediktif untuk memisahkan individu yang potensial dan yang tidak. Individu yang potensial adalah mereka yang diprediksi akan menerima tawaran produk yang diajukan oleh petugas *telemarketer*. Aktivitas ini sangat identik dengan yang dikerjakan dalam *campaign* via SMS (*short message service*) oleh berbagai perusahaan retail.

Tersedia banyak pemodelan prediktif untuk melakukan prediksi terjadinya (atau tidak terjadinya) suatu kejadian masa mendatang. Beberapa yang disebut berikut adalah teknik dan algoritma pemodelan yang sering digunakan oleh analis baik yang berbasis pemikiran statistika maupun *machine learning*, yaitu: regresi logistik, analisis diskriminan, *k-nearest-neighbor*, *Bayesian classifier*, *classification tree*, *neural network*, dan *support vector machine*. Ada beberapa algoritma lain yang dapat ditemukan dengan mudah di banyak literatur ilmiah maupun praktis.

Berbagai macam algoritma yang disebutkan di atas dapat digunakan untuk menjawab tujuan sama, dan banyak orang berpendapat bahwa satu sama lain dapat dipandang memiliki sifat *complementary*. Karena itu, kemudian muncul pertanyaan besar: algoritma atau teknik mana yang sebaiknya digunakan? Tidak hanya itu, dengan menerapkan salah satu teknik yang sama, dua orang analis dapat menghasilkan model yang berbeda karena dalam proses pemodelannya dapat saja mereka menggunakan prediktor yang berbeda, menggunakan sampel data yang berbeda, serta menerapkan *pre-processing* yang berbeda sesuai dengan kreativitas masing-masing. Dengan demikian, sekali lagi kemudian muncul pertanyaan: model mana yang sebaiknya digunakan?

Model Selection

Pertanyaan tersebut kemudian berujung pada penggunaan berbagai kriteria untuk menentukan model terbaik. Diskusi kemudian berkembang dalam ranah *model selection* (pemilihan model) yang menggunakan berbagai macam kriteria.

Secara umum, penulis memahami bahwa ada dua kriteria besar dalam penentuan model mana yang digunakan. Kriteria pertama terkait dengan kinerja prediksinya. Dalam bahasa lain, orang menggunakan istilah akurasi atau ketepatan prediksi. Model dengan akurasi yang lebih tinggi disebut sebagai model yang sebaiknya digunakan. Kriteria ini dikenal sebagai *goodness of fit*. Ukuran yang termasuk dalam kategori ini antara lain *likelihood function*, *correct classification rate*, *sensitivity*, dan *specificity*.

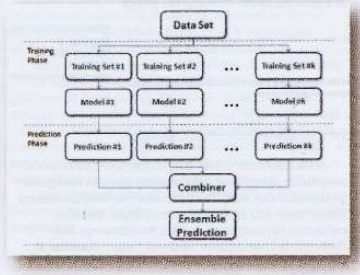
Kriteria yang kedua adalah terkait dengan kesederhanaan model. Secara naluriah, model yang disukai adalah model yang lebih ringkas, menggunakan *predictor* yang lebih sedikit, atau bentuk-bentuk fungsi yang lebih sederhana. Kriteria kedua ini dikenal sebagai *complexity cost*. Ukuran yang tergolong dalam kriteria ini meliputi banyaknya parameter dalam model, banyaknya simpul pada *tree* dan *neural network*, serta derajat *polinomial* dari variabel *predictor*. *Complexity cost* ini penting diperhatikan agar model prediksi tidak mengalami masalah *overfit*.

Kriteria-kriteria di atas selanjutnya digunakan oleh para analis untuk menentukan model mana yang digunakan. Dua jenis kriteria tersebut banyak digabungkan menjadi satu kriteria gabungan seperti yang dilakukan pada AIC (*Akaike's Information Criterion*) dan yang sejenisnya. Model dengan *goodness-of-fit* besar dan *complexity cost* kecil merupakan model yang terpilih dalam proses *model selection* ini.

Pergeseran Paradigma

Kemajuan teknologi komputasi mendorong berbagai perubahan dan perkembangan dalam analitik. Perkembangan tidak hanya terjadi dengan munculnya algoritma dan teknik baru, yang awalnya tidak mudah dan tidak murah dari sisi komputasi. Perkembangan juga terjadi pada paradigma penggunaan model akhir dalam melakukan prediksi.

Pada saat komputasi masih menjadi kendala besar dalam pemodelan, ada pemikiran bahwa algoritma yang diterapkan tinggal menggunakan salah satu saja dari yang tersedia. Pasalnya, untuk memperoleh model dari satu algoritma bisa jadi memerlukan waktu yang tidak sedikit. Dengan teknologi terkini, satu buah algoritma dapat menghasilkan sebuah model prediktif dalam waktu yang singkat apabila data-data yang diperlukan telah tersedia.



Gambar 1. Mekanisme dasar pendekatan *ensemble learning* (Tsiakis, 2015).

Kondisi ini kemudian memunculkan ide untuk melakukan prediksi tidak hanya didasarkan pada satu buah model (yang dianggap paling baik), namun melakukan prediksi dengan cara menggabungkan hasil prediksi dari banyak model. Paradigma ini yang dikenal sebagai *ensemble learning*. Theodosios Tsiakis (2015) dalam buku yang diieditnya berjudul *Trends and Innovations in Marketing Information Systems* memuat bagaimana *ensemble learning* ini bekerja. Gambar 1 menyajikan secara ringkas sistem *ensemble* ini dipergunakan untuk melakukan prediksi.

Dari satu buah *dataset* dapat diperoleh banyak model prediksi baik menggunakan berbagai teknik yang berbeda maupun menggunakan algoritma yang sejenis. Setiap model selanjutnya menghasilkan prediksi yang dapat berbeda satu dengan yang lainnya. Pendekatan *ensemble learning* menggabungkan berbagai macam prediksi tersebut menjadi satu buah prediksi akhir. Teknik penggabungan yang banyak digunakan adalah *averaging* dan *majority vote*. Pada penerapan *majority vote* untuk *credit scoring* misalnya, keputusan apakah individu yang mengajukan aplikasi pinjaman akan ditolak atau diterima aplikasinya didasarkan pada suara terbanyak dari hasil prediksi macet-lancar dari banyak model.

Secara umum *ensemble learning* terbagi menjadi dua kelompok yaitu *hybrid ensemble* dan non-*hybrid ensemble*. Yang disebut *hybrid ensemble* adalah jika model-model yang nanti digabungkan prediksinya merupakan model-model yang dihasilkan dari berbagai jenis algoritma berbeda. Sementara *non-hybrid ensemble* menggabungkan model-model yang diperoleh dari algoritma sejenis.

Ensemble Learning, Pilihan yang Tepat

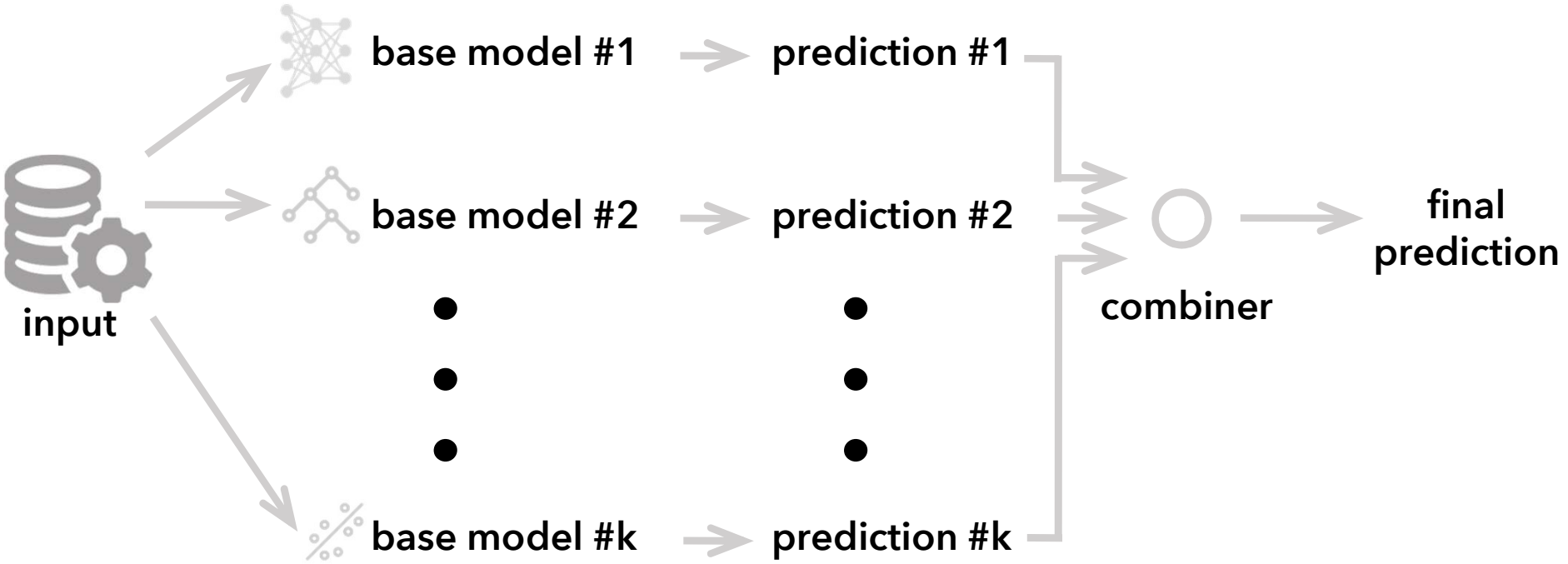
Kenyataan bahwa pendekatan *ensemble learning* mampu memberikan solusi prediksi yang lebih akurat daripada model-model tunggal dapat ditemui dari berbagai paper di jurnal ilmiah. Teknik-teknik *ensemble* yang mengandalkan variasi dari pendekatan *random forest* dan *boosting* mampu memberikan prediksi dengan akurasi yang sangat baik. *Random forest* bekerja dengan membuat model-model penyusun *ensemble* sedemikian rupa sehingga berbagai kemungkinan dapat terakomodir secara maksimal, sedangkan *boosting* bekerja secara *iterative* sehingga kasus-kasus yang tidak mudah diprediksi menjadi bukan masalah lagi.

Kemampuan pendekatan *ensemble* ini tidak hanya tertuang pada berbagai paper ilmiah, namun juga dapat dilihat pada penyelesaian kasus-kasus aplikatif seperti yang dapat dilihat pada kompetisi *data science* Kaggle (https://www.kaggle.com/). Kompetisi ini terbuka bagi pegiat *data science* dan *data mining* untuk memberikan solusi prediktif dari kasus-kasus yang disampaikan oleh banyak perusahaan besar berskala internasional.

Setiap tim atau individu dipersilakan mengembangkan solusi dan menyajikan prediksinya untuk kemudian dinilai. Mereka yang memberikan prediksi dengan akurasi yang paling tinggi yang dinyatakan sebagai pemenang. Peringkat tiga besar dalam lima tahun terakhir dari kompetisi ini didominasi oleh mereka yang menggunakan pendekatan *ensemble* yang digabungkan dengan berbagai macam algoritma dasar.

Berdasarkan apa yang berkembang saat ini, pendekatan *ensemble* dalam pemodelan prediktif menjadi pilihan tepat bagi mereka yang berupaya memperoleh prediksi yang memuaskan dengan cara yang sangat mudah untuk dikerjakan. Hal senada juga telah dikemukakan oleh Mu Zhu (University of Waterloo) pada jurnal The American Statistician pada tahun 2008.

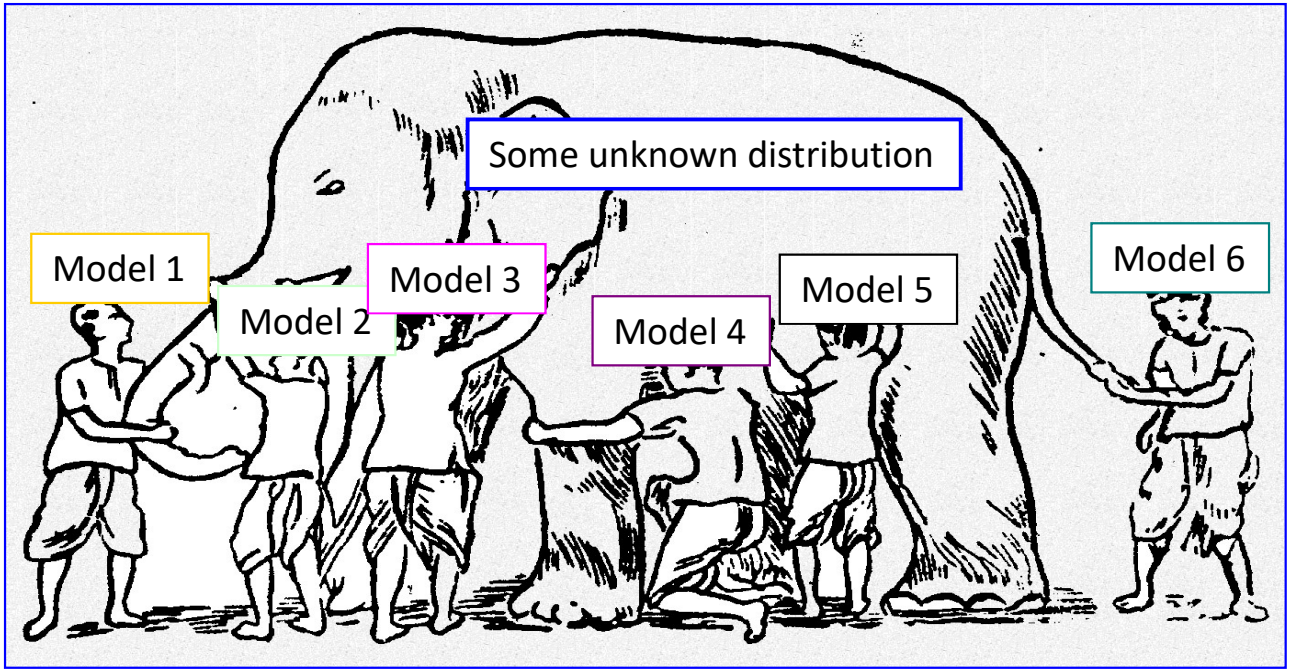
Apa itu pemodelan ensemble?



Rationale

- There is no algorithm that is always the most accurate
- Generate a group of base-learners which when combined has higher accuracy
- Each algorithm makes assumptions which might be or not be valid for the problem at hand.
- Different learners use different
  - Algorithms
  - Hyperparameters
  - Representations (Modalities)
  - Training sets
  - Subproblems

Why Ensemble Works?

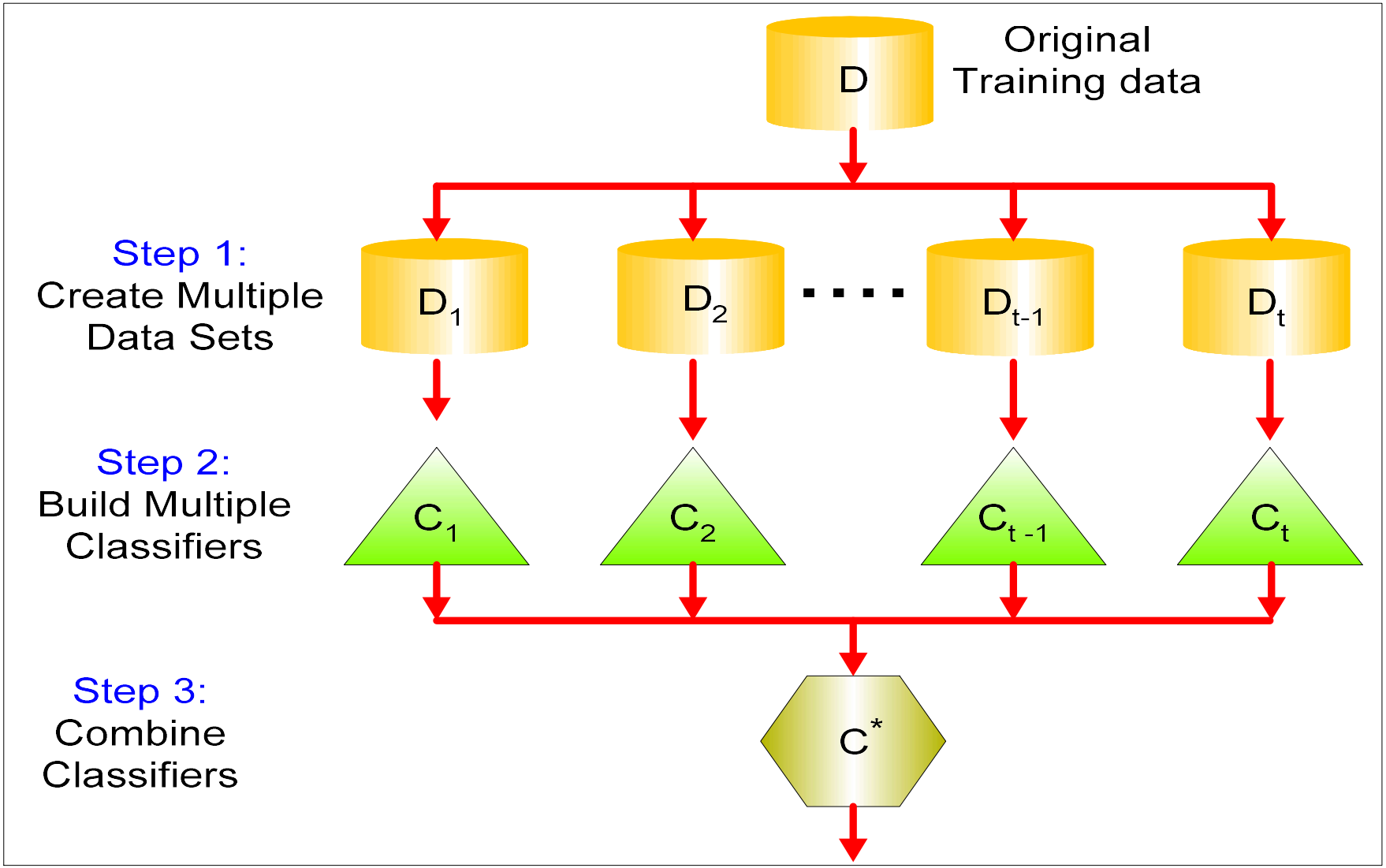


Ensemble gives the global picture!

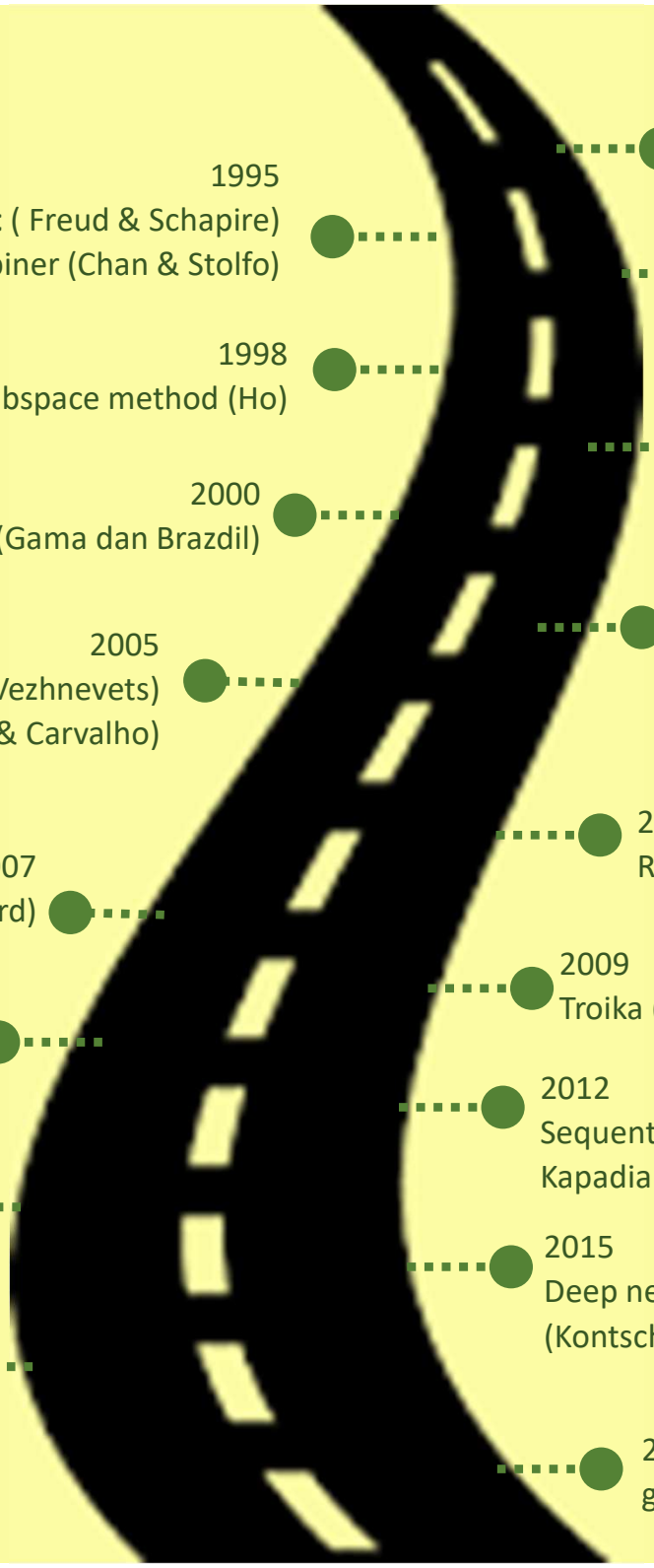
What is the Main Challenge for Developing Ensemble Models?

- The main challenge is not to obtain highly accurate base models, but rather to obtain base models which make different kinds of errors.
- High accuracies can be accomplished if different base models misclassify different training examples, even if the base classifier accuracy is low.

General Idea



milestones



1992 Stacking (Wolpert)

1995
Adaboost (Freud & Schapire)
Combiner (Chan & Stolfo)

1996
Bagging (Breiman)

1998
random subspace method (Ho)

1999
Wagging/Weight
aggregating(Bauer & Kohav)

2000
Cascade Generalization(Gama dan Brazdil)

2001
Random Forest (Breiman)
Gradient Boosting Machine
(Friedman)

2005
Modest AdaBoost (Vezhnevets & Vezhnevets)
Stacked sequential learning (Cohen & Carvalho)

2006
Rotation Forest (Rodriguez *et al*.)

2007
Super Learner (van der Laan, Polley, Hubbard)

2009
Troika (Menahem *et al*.)

2010
GA-stacking (Ledezma *et al*.)

2012
Sequential Stacking (Trivedi and
Kapadia)

2014
Random projection random discretization
ensembles (Ahmad & Brown)

2015
Deep neural decision forests
(Kontschieder *et al*.)

2016
XGBoost - Extreme Gradient Boosting
(Chen & Guestrin)

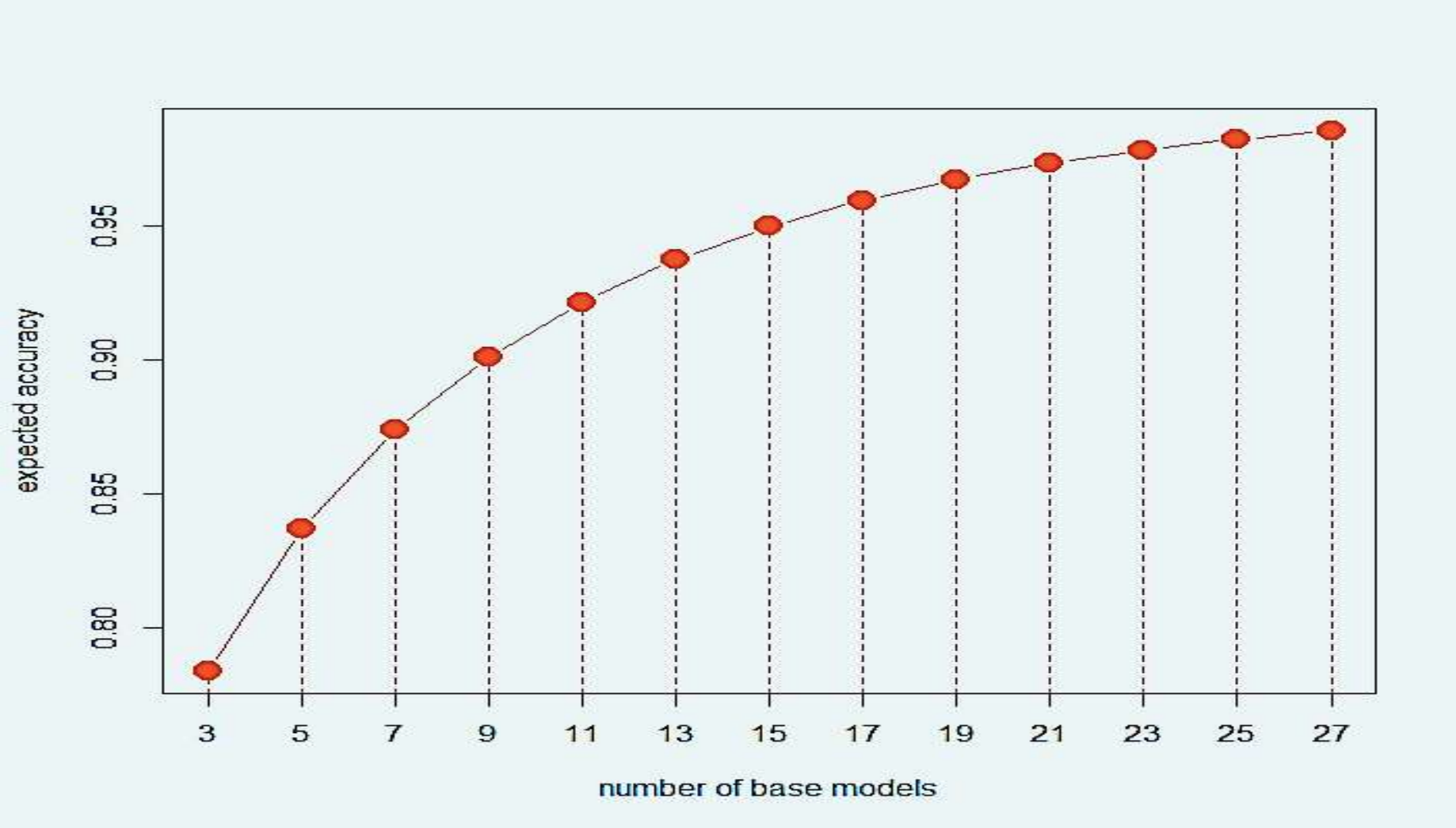
2017
gcForest (Zhou & Feng)

Akurasi model ensemble

- 5 base models, independent
- accuracy level @70%
- ensemble prediction → majority vote
- correctly predict if at least 3 base models do so
- accuracy of the ensemble:

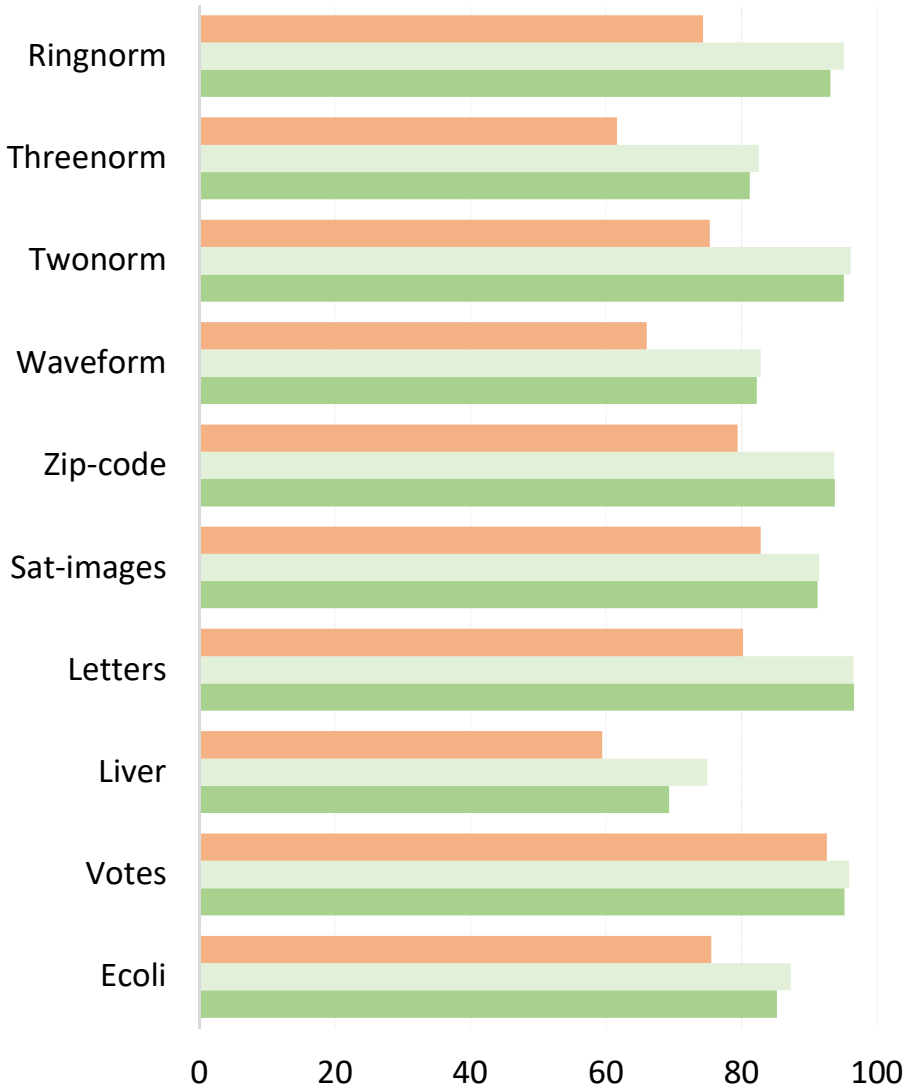
$$\sum_{k=3}^{5} \binom{5}{k} \times 0.7^{k} \times 0.3^{5-k} = 83.69\%$$

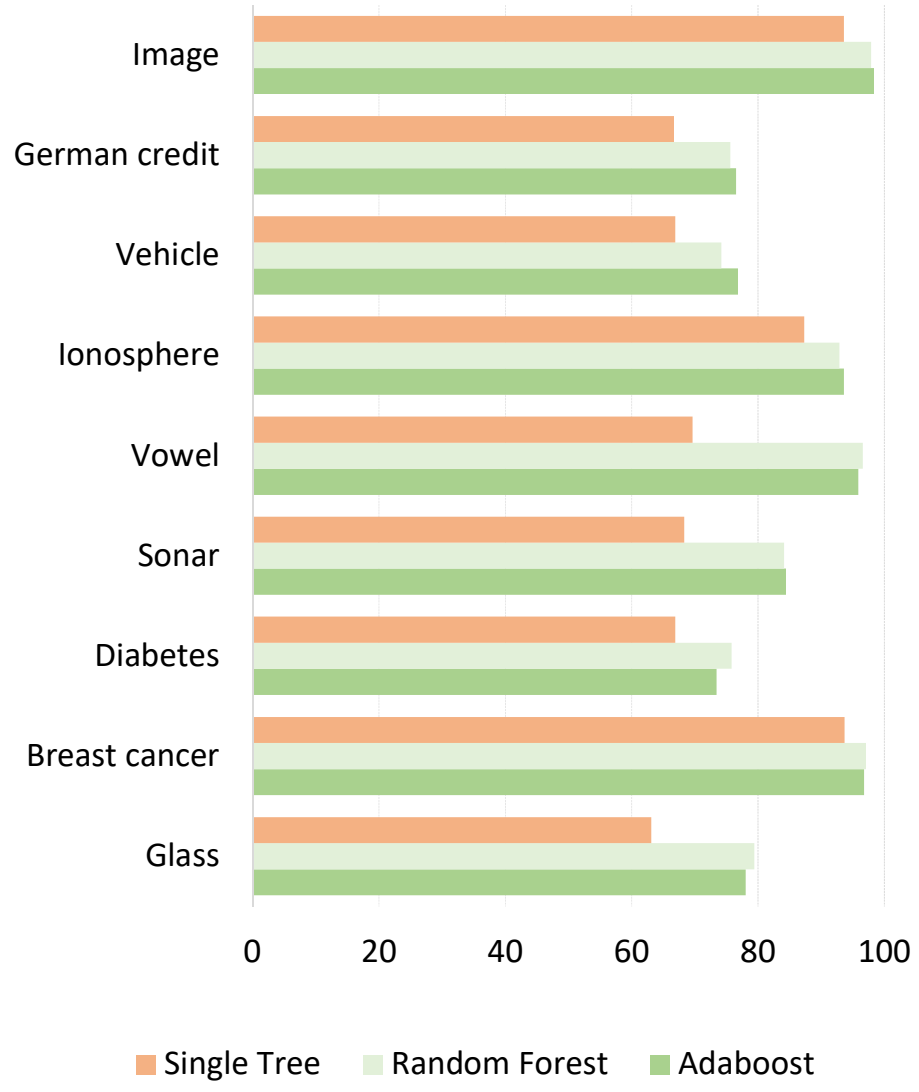
Akurasi model ensemble



Success Story

*Breiman L (2001). "Random Forests". Machine Learning. **45** (1): 5–32.*





Bagging

Ilustrasi untuk Motivasi

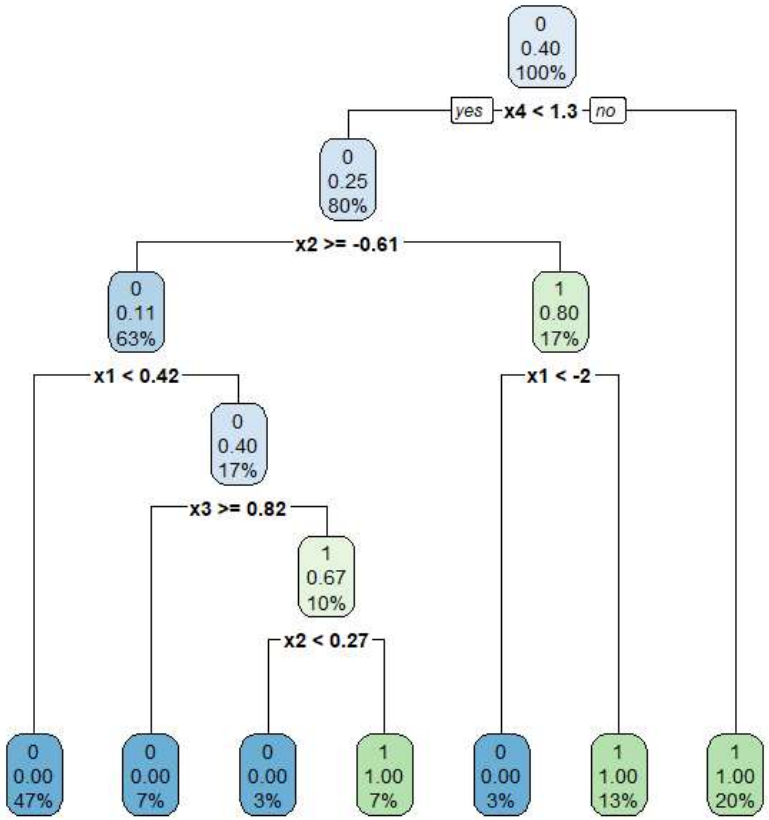
- Menggunakan sampel dari dataset yang sama, pohon klasifikasi yang dihasilkan bisa sangat berebda

- Buka file program: bagging 01.R

```
#membangkitkan data
#n=30
#5 prediktor
set.seed(1000)
sigma = matrix(0.95, 5, 5)
diag(sigma) = c(1,1,1,1,1)
mean = c(0,0,0,0,0)
library(MASS)
data = data.frame(mvrnorm(30, mean, sigma))
colnames(data)=c("x1", "x2", "x3", "x4", "x5")
peluang = ifelse(data$x1<0.5, 0.2, 0.8)
data$class <- rbinom(30, 1, peluang)
```

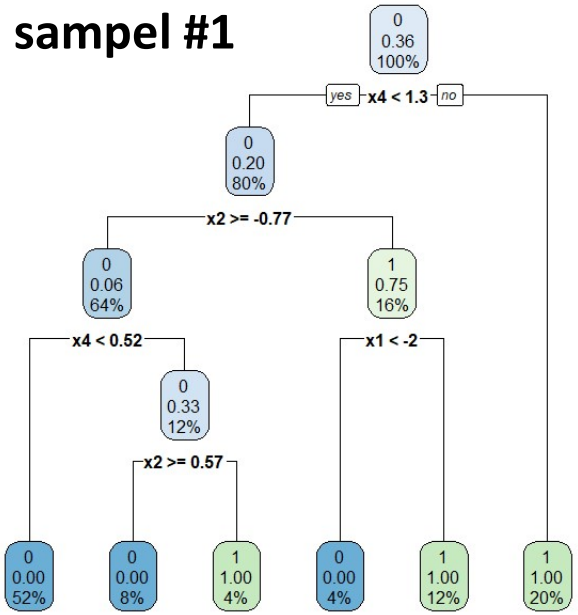
```
#bentuk pohon data asli
library(rpart)
pohon = rpart(class ~ . , data=data, method="class",
        control=rpart.control(cp=0, minsplit=3))
library(rpart.plot)
rpart.plot(pohon)

#bentuk pohon data sampel pertama
acak = sample(1:30, 25)
sampel = data[acak,]
pohon.a = rpart(class ~ . , data=sampel,method="class",
          control=rpart.control(cp=0, minsplit=3))
rpart.plot(pohon.a)
```

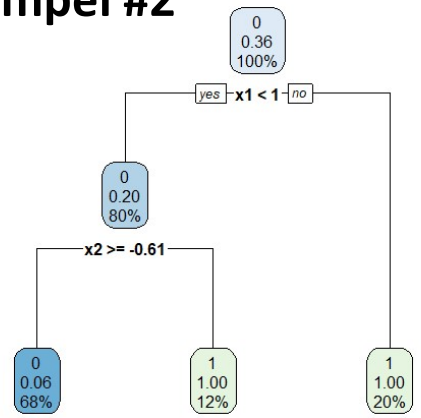


pohon asli

sampel #1



sampel #2



Algoritma Bagging

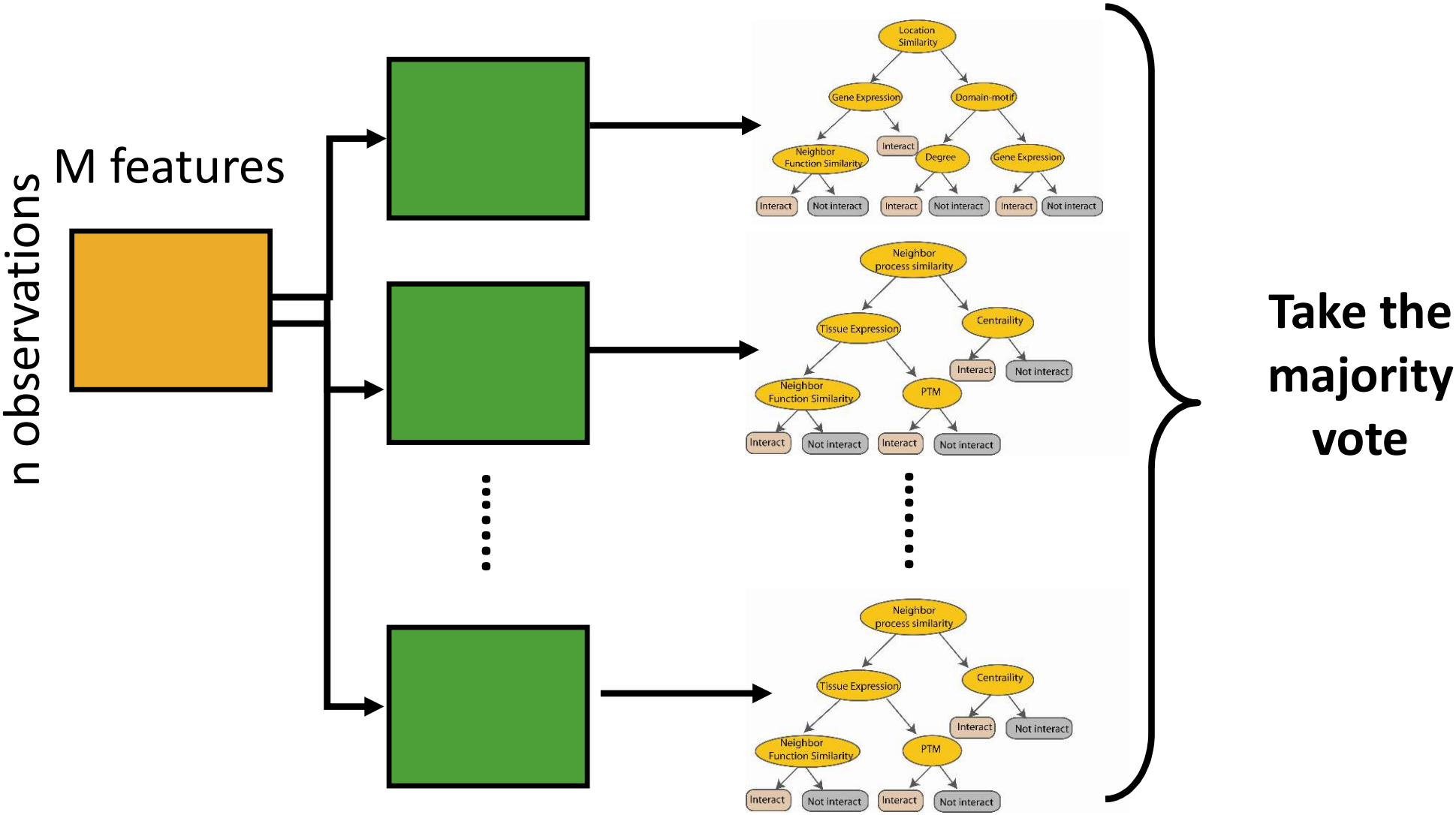
Algorithm 1 Standard bagging algorithm

Input: D: original training set of examples of size N, k: number of bootstrap samples, LA: learning algorithm;
Output: C^* bagging ensemble with k component classifiers
Learning phase:

1: **for** $i = 1 \rightarrow k$ **do**
2: $S_i \leftarrow$ bootstrap sample from D;
3: generate classifier $C_i \leftarrow LA(S_i)$
4: **end for**
Predicting class label for new instance x:
5:

$$C^*(x) = \arg\max_y \sum_{i=1}^{k} [C_i(x) = y] \quad (1)$$

Bagging


n observations
M features
Location Similarity
Gene Expression
Domain-motif
Neighbor Function Similarity
Interact
Degree
Gene Expression
Interact
Not interact
Interact
Not interact
Interact
Not interact
Neighbor process similarity
Tissue Expression
Centrality
Interact
Not interact
Neighbor Function Similarity
PTM
Interact
Not interact
Interact
Not interact
Neighbor process similarity
Tissue Expression
Centrality
Interact
Not interact
Neighbor Function Similarity
PTM
Interact
Not interact
Interact
Not interact
Take the majority vote

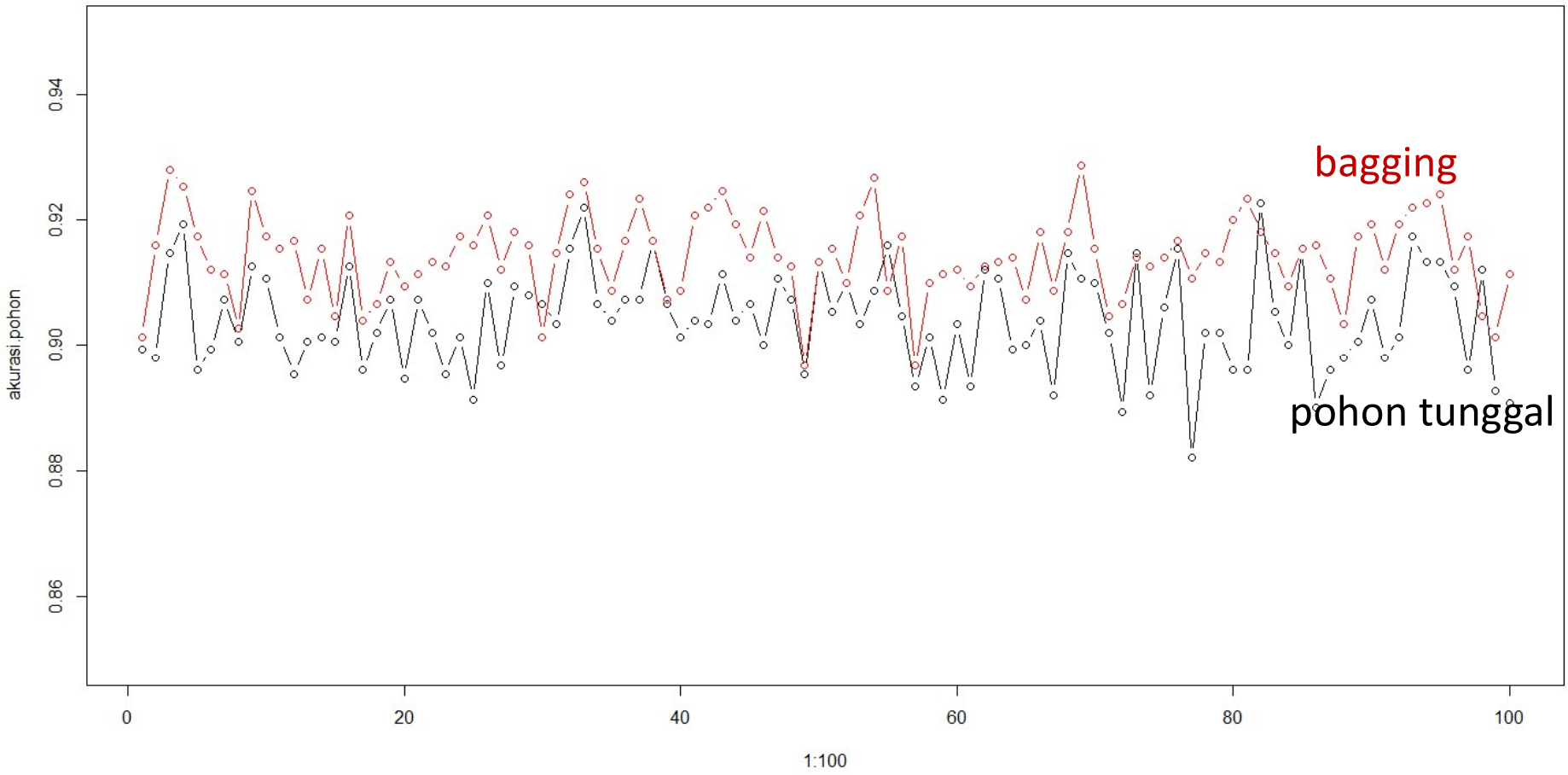
Implementasi

- Buka file “bagging 02.R” dan “bagging 03.R”

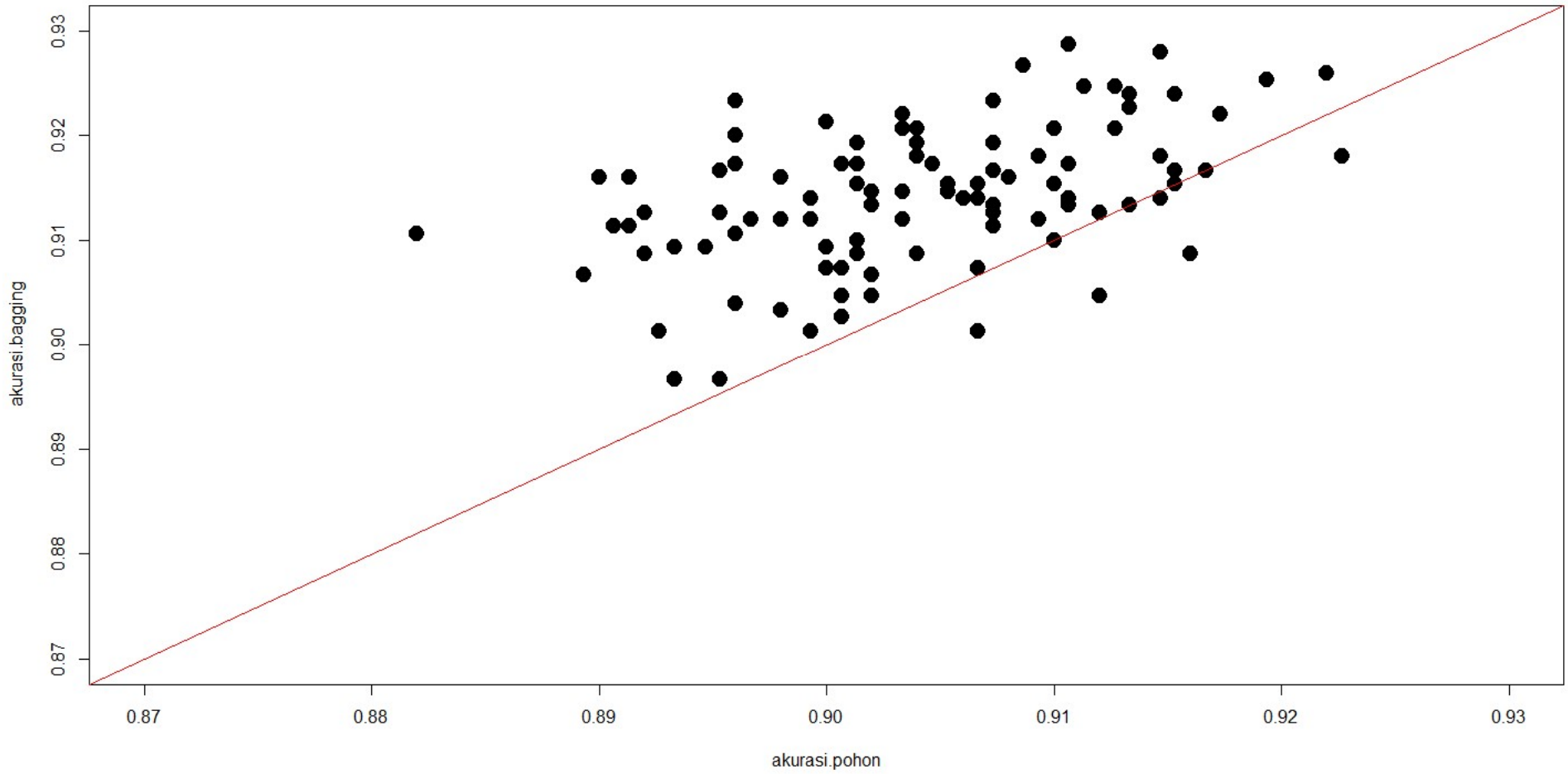
- Data yang digunakan: latihanskoring.csv
  - Variabel Target
    - Status
  - Variabel Prediktor
    - DBR
    - usia
    - jeniskelamin
    - marital_status
    - duration_job
    - Pekerjaan
    - pendidikan

package → ipred
fungsi → ipredbagg

Perbandingan akurasi



Perbandingan akurasi



Random Forest

Dasar Random Forest

- Korelasi antar prediksi pohon pada bagging umumnya masih memiliki korelasi yang tinggi → mengurangi efektifitas model ensemble

- Perlu upaya untuk membuat korelasi antar pohon menjadi lebih rendah → antar pohon dibuat se-random mungkin

- Bagging → Random Forest

Dasar Random Forest

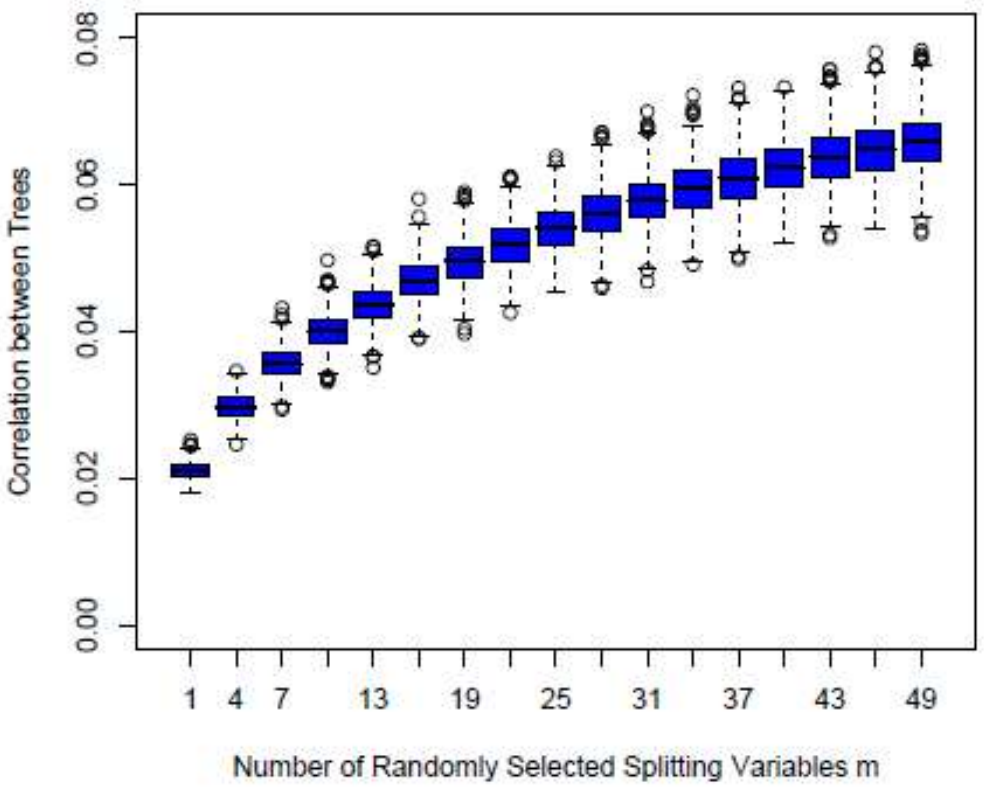


FIGURE 15.9. *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of m. The boxplots represent the correlations at 600 randomly chosen prediction points x.*

Algoritma Random Forest

For b = 1 to B:

(a) Draw a bootstrap sample Z∗ of size *N* from the training data.

(b) Grow a random-forest tree to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $\boldsymbol{n_{min}}$ is reached.

i. Select ***m*** variables at random from the *p* variables.

ii. Pick the best variable/split-point among the *m*.

iii. Split the node into two daughter nodes.

Output the ensemble of trees.

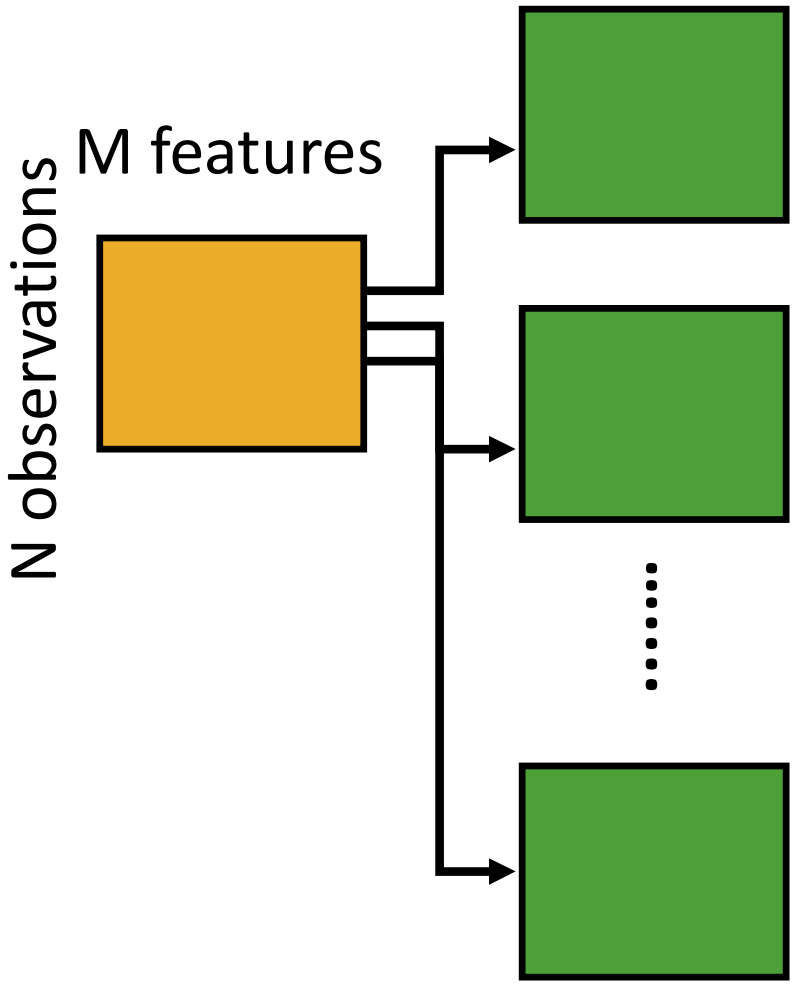
To make a prediction at a new point *x* we do:

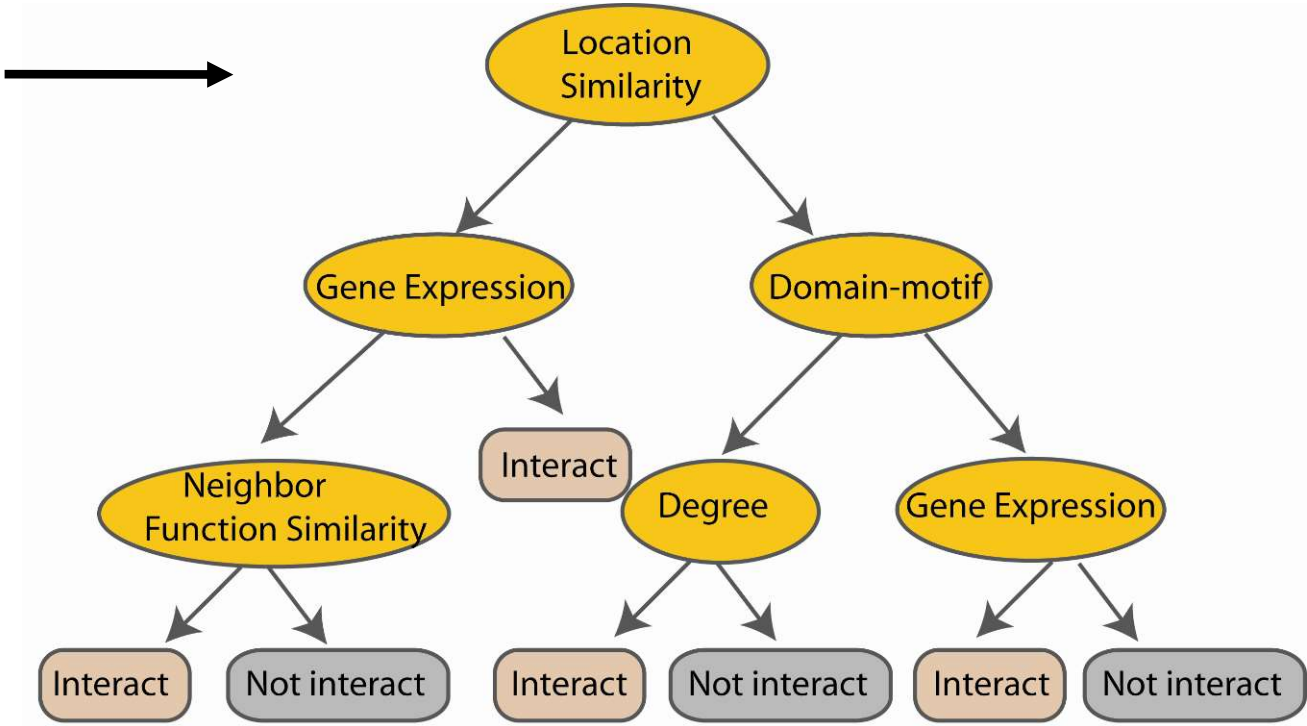
For regression: average the results

For classification: majority vote

Algoritma Random Forest

At each node in choosing the split feature
choose only among $m<M$ features





- Implementasi algoritma random forest di R
  - Gunakan file “random forest 01.R”
  - Ubah hyperparameter algoritma dan bandingkan performanya

- Perbandingan akurasi dari algoritma random forest dengan pohon klasifikasi tunggal
  - Gunakan file “random forest 02.R”

Estimating the test error:

- While growing forest, estimate test error from training samples
- For each tree grown, 33-36% of samples are not selected in bootstrap, called out of bootstrap (OOB) samples
- Using OOB samples as input to the corresponding tree, predictions are made as if they were novel test samples
- Through book-keeping, majority vote (classification), average (regression) is computed for all OOB samples from all trees.
- Such estimated test error is very accurate in practice, with reasonable N

- Penghitungan error rate (1 – akurasi) menggunakan pendekatan OOB
  - Gunakan file “random forest 03.R”
  - Ganti hyperparameter algoritma dan bandingkan hasilnya

Estimating the importance of each predictor

- Denote by ê the OOB estimate of the loss when using original training set, D.
- For each predictor x_p where $p \in \{1, .., k\}$
  - Randomly permute p_{th} predictor to generate a new set of samples $D' = \{(y_1, x'_1), \ldots, (y_N, x'_N)\}$
  - Compute OOB estimate $\hat{e}_p$ of prediction error with the new samples
- A measure of importance of predictor x_p is $\hat{e}_p - \hat{e}$, the increase in error due to random perturbation of *p*-th predictor

- Implementasi penentuan tingkat kepentingan variabel prediktor
  - Gunakan file “random forest 04.R”
  - Bandingkan urutan tingkat kepentingan jika kriterianya diganti

Boosting

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round
- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Boosting Algorithm

Initialization step: for each example x, set

$D(x)=\frac{1}{N}$, where N is the number of examples

Iteration step (for t=1…T):

1.Find best weak classifier $h_t(x)$ using weights $D_t(x)$

2. Compute the error rate ε_t as

$$\varepsilon_t = \sum_{i=1}^{N} D(x_i).I[y_i \neq h_t(x_i)]$$

3. assign weight α_t to classifier $h_t(x)$ in the final hypothesis

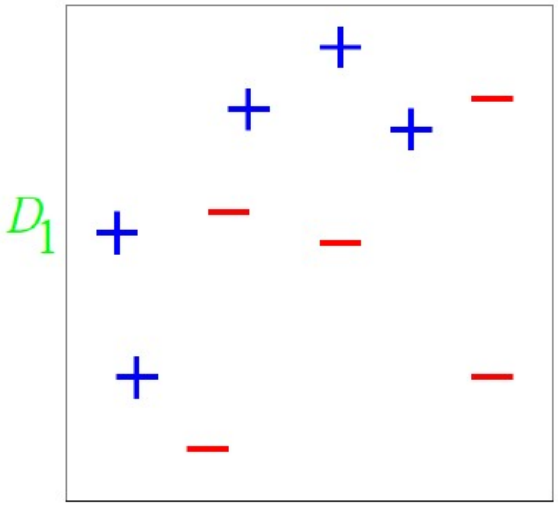
$$\alpha_t = \log((1-\varepsilon_t)/\varepsilon_t)$$

4. For each x_i , $D(x_i) = D(x_i).\exp(\alpha_t.I[y_i \neq h_t(x_i)])$

5.Normalize $D(x_i)$ so that $\sum_{i=1}^{N} D(x_i) = 1$

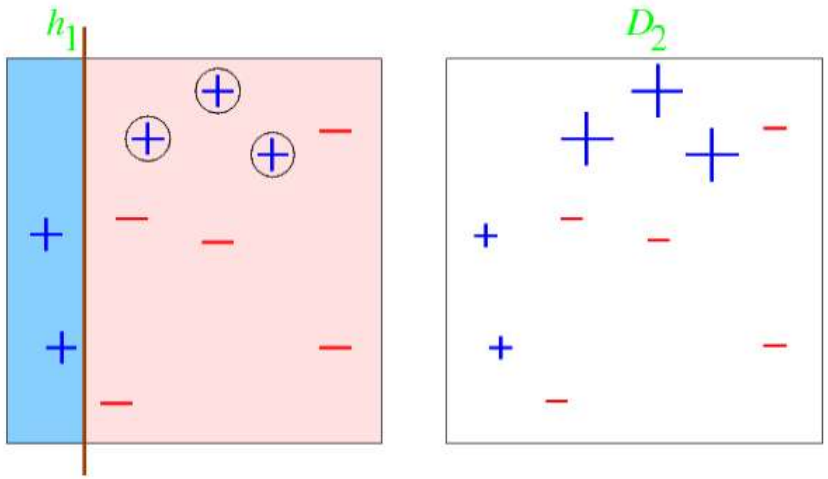
$$f_{final}(x) = sign\ [\sum \alpha_t\ h_t(x)]$$

Boosting: Illustration



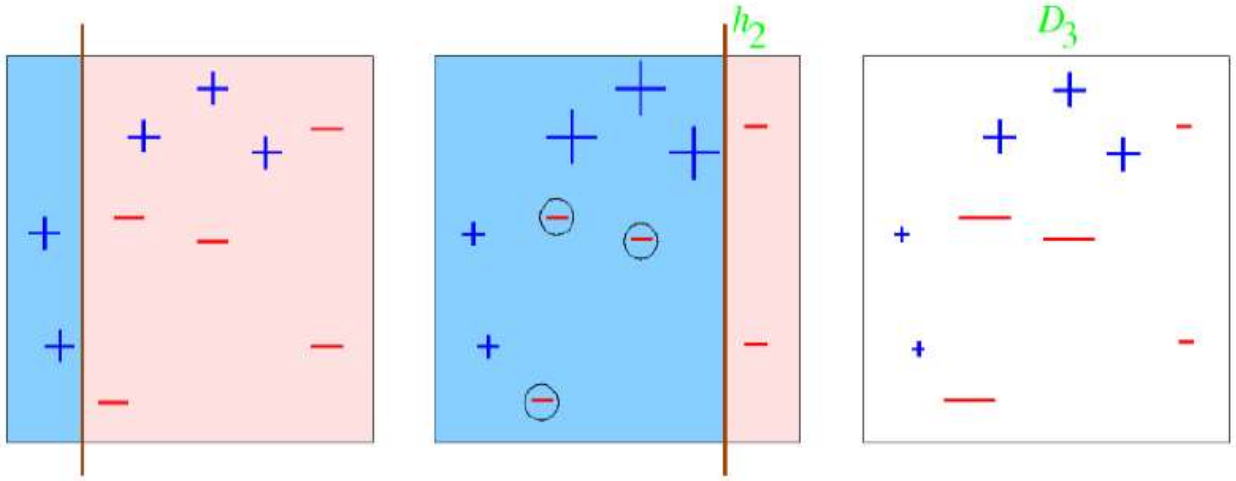
- Consider binary classification with 10 training examples
- Initial weight distribution D1 is uniform (each point has equal weight = 1/10)
- Each of our weak classifiers will be an axis-parallel linear classier

After round 1



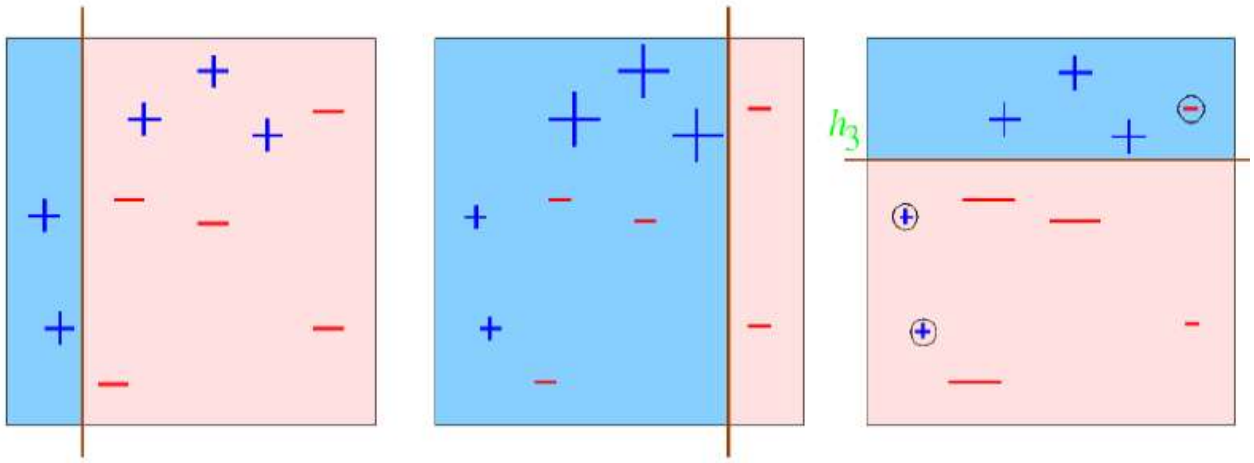
- Error rate of h1: e1 = 0.3; weight of h1: $\alpha 1 = 0.42$
- Each misclassified point upweighted (weight multiplied by exp($\alpha 1$))
- Each correctly classified point downweighted (weight multiplied by exp(- $\alpha 1$))

After round 2

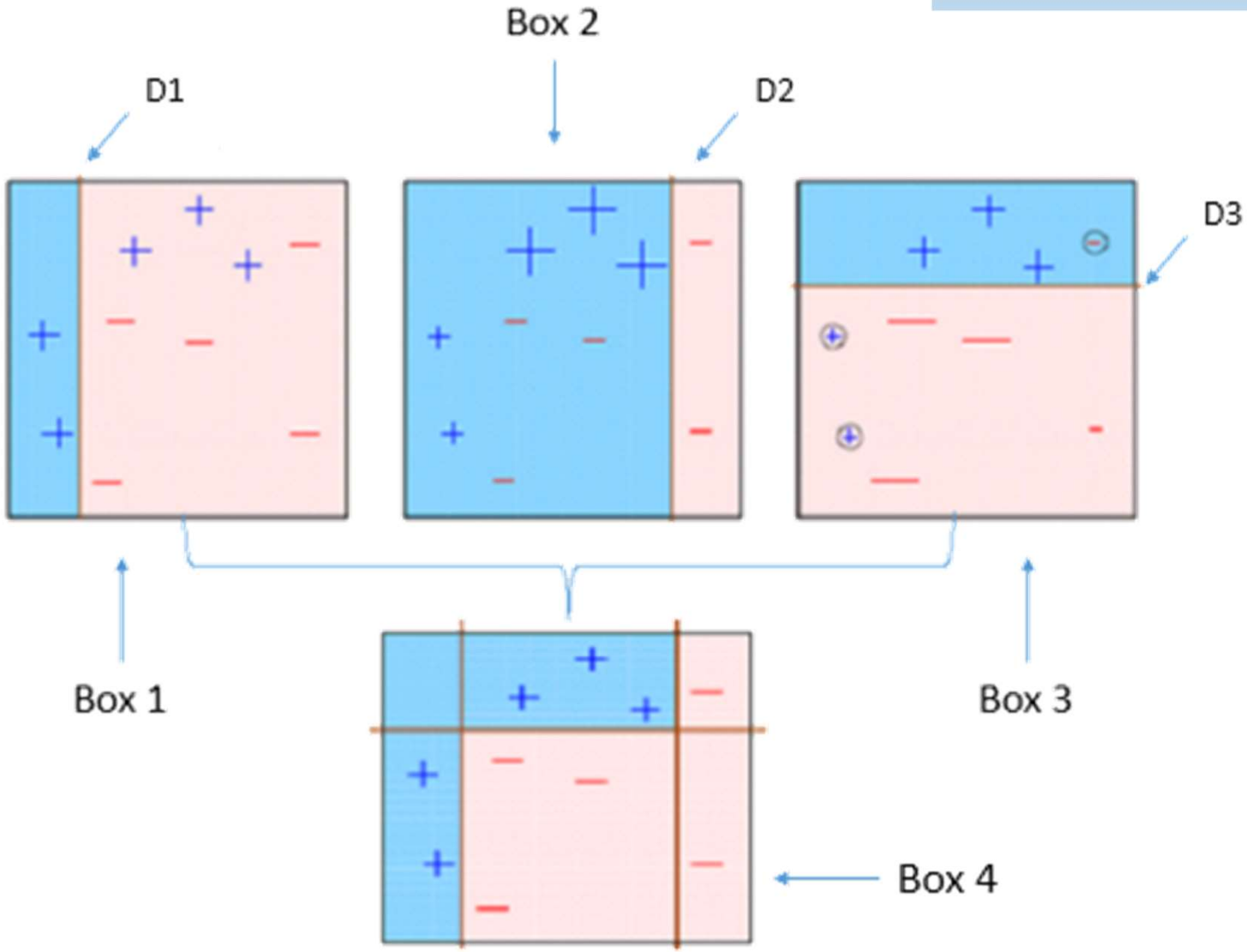


- Error rate of h2: e1 = 0.21; weight of h1: $\alpha 1 = 0.65$
- Each misclassified point upweighted (weight multiplied by exp($\alpha 2$))
- Each correctly classified point downweighted (weight multiplied by exp(- $\alpha 2$))

After round 3

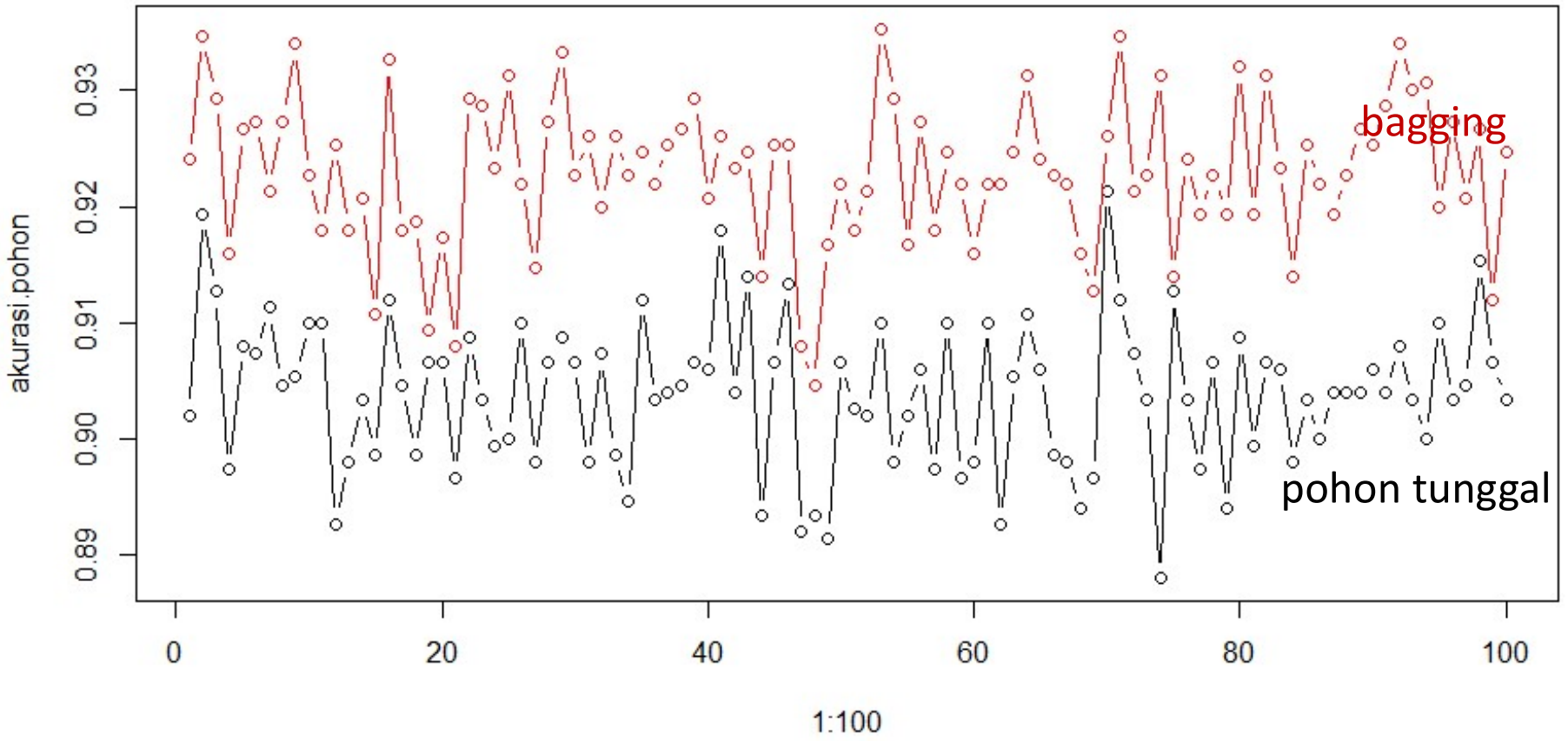


- Error rate of h3: e3 = 0.14; weight of h3: α3 = 0.92
- Suppose we decide to stop after round 3
- Our ensemble now consists of 3 classifiers: h1; h2; h3


Box 2
D1
D2
D3
Box 1
Box 3
Box 4

- Implementasi algoritma adaboost
  - Gunakan file “boosting 01.R”
  - Ganti nilai hyperparameter algoritma dan bandingkan performanya

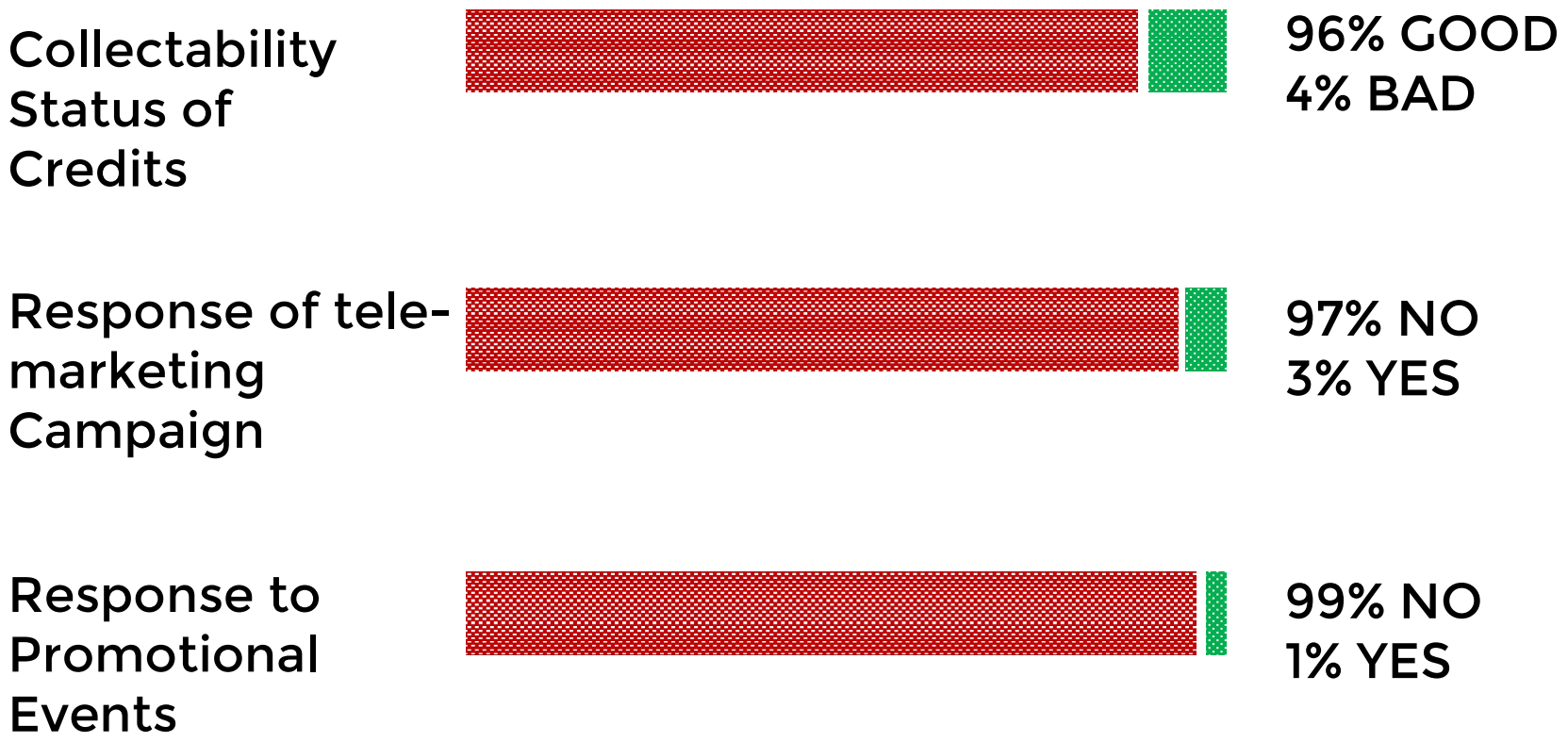
- Perbandingan performa boosting dengan pohon klasifikasi tunggal
  - Gunakan file “boosting 02.R”


akurasi.pohon
0.89
0.90
0.91
0.92
0.93
0
20
40
60
80
100
1:100
bagging
pohon tunggal

Pendekatan Ensemble untuk Penanganan Data Tidak Seimbang

Class-Imbalance Problem

the proportion of a class is far less than the proportion of the other class



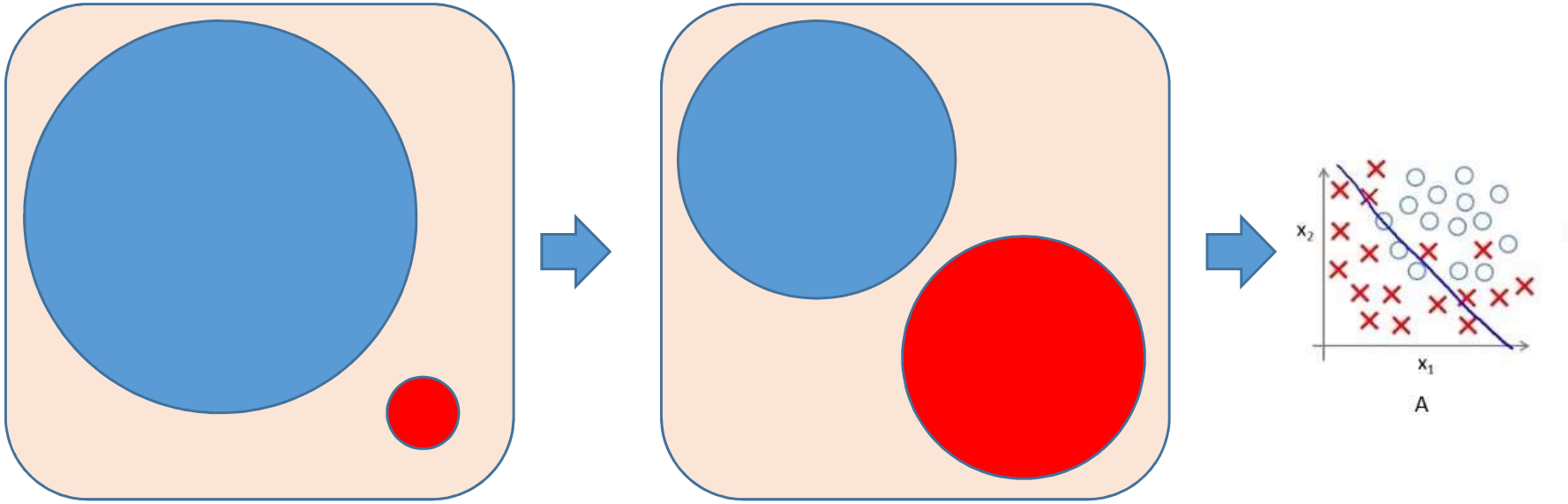
Paradox of Accuracy

Accuracy = 97%

Good Model? NO

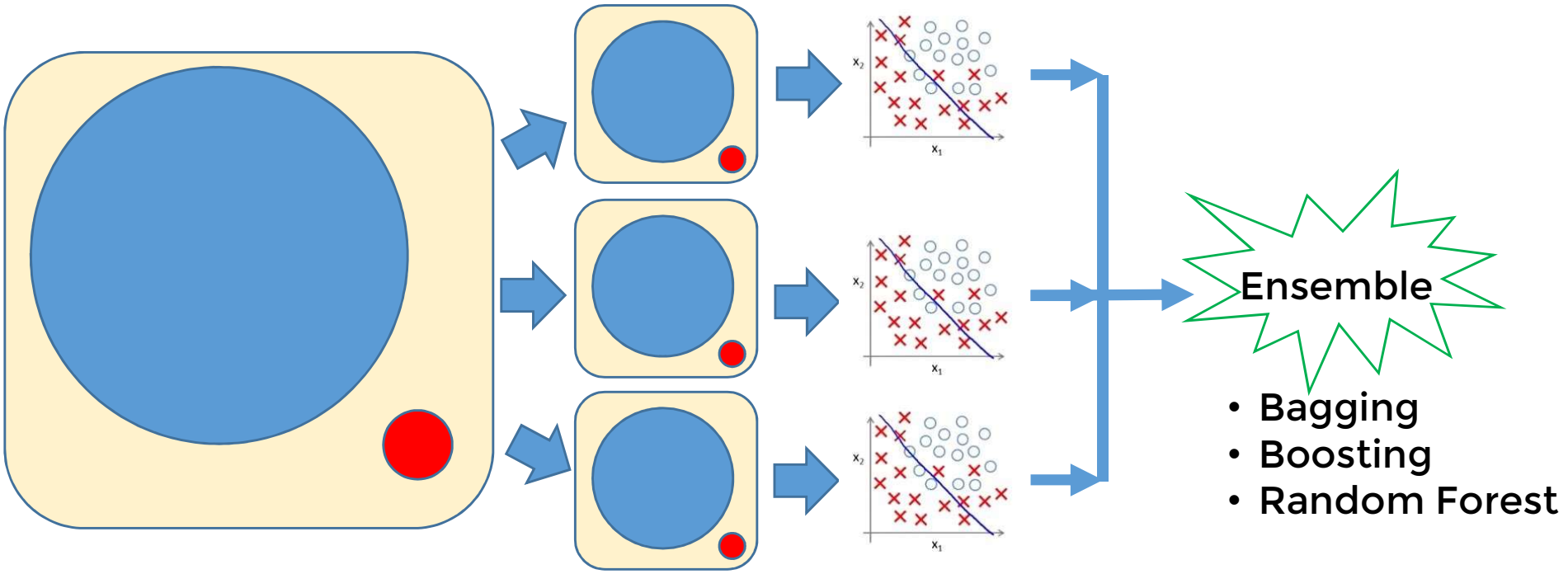
Strategy #1

Oversampling/Undersampling Pre-Processing

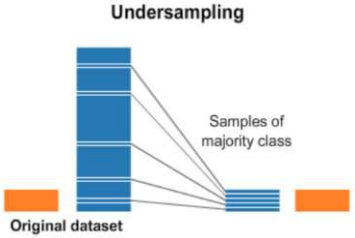


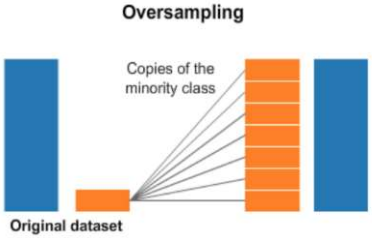
Strategy #2

Ensemble Learning Approach



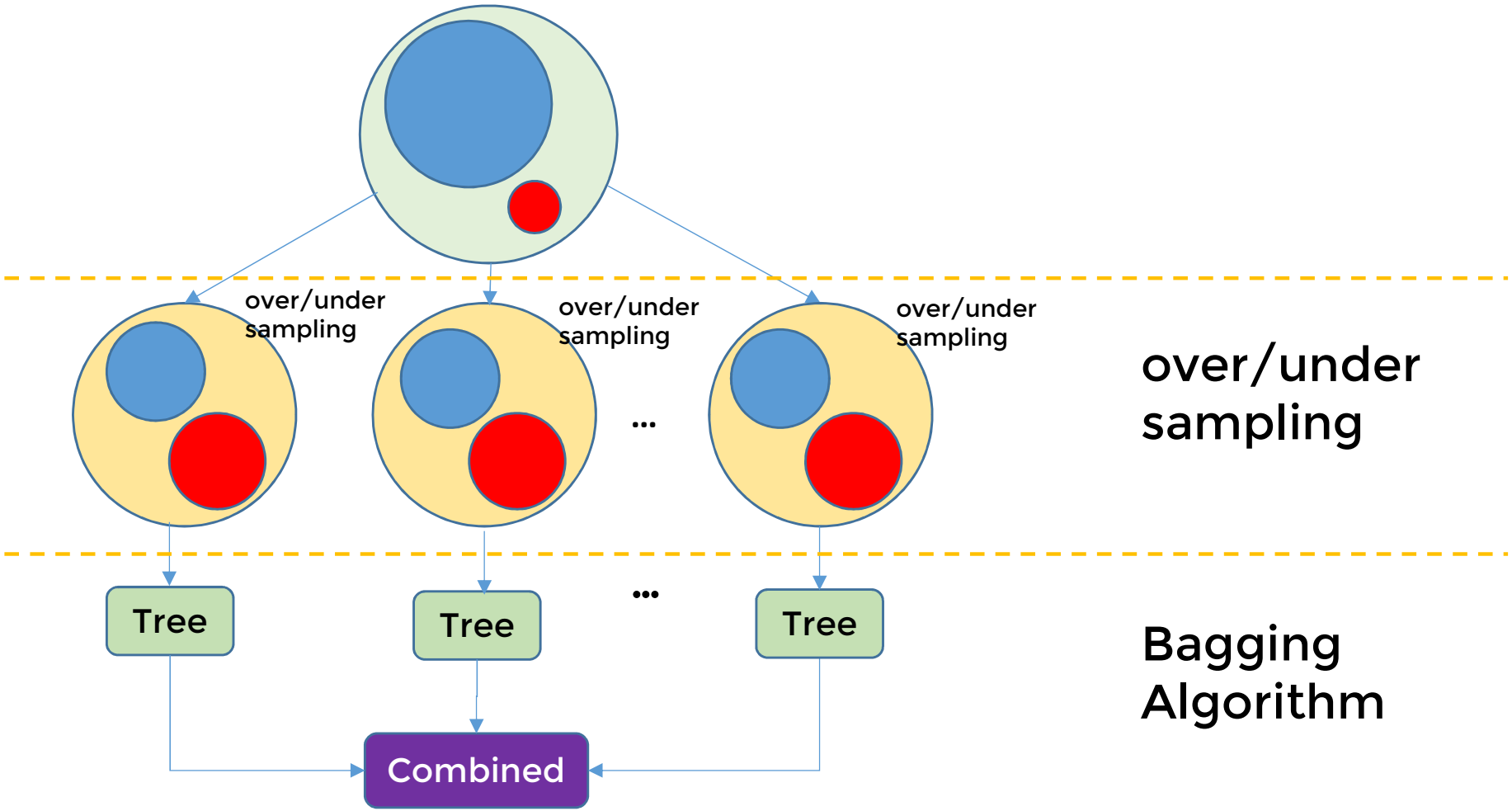
Hybrid Approach





- Over/Under Sampling + Ensemble Learning
- Methods:
  - Over/Under Bagging
  - RUS-Boost (random under sampling + boosted tree)
  - EasyEnsemble
  - BalanceCascade

Over/Under Bagging



EasyEnsemble

Algorithm 1 The `EasyEnsemble` algorithm.

1: {Input: A set of minority class examples $\mathcal{P}$, a set of majority class examples $\mathcal{N}$, $|\mathcal{P}| < |\mathcal{N}|$, the number of subsets T to sample from $\mathcal{N}$, and s_i, the number of iterations to train an AdaBoost ensemble H_i}

2: $i \Leftarrow 0$

3: **repeat**

4: $\quad i \Leftarrow i + 1$

5: $\quad$ Randomly sample a subset $\mathcal{N}_i$ from $\mathcal{N}$, $|\mathcal{N}_i| = |\mathcal{P}|$.

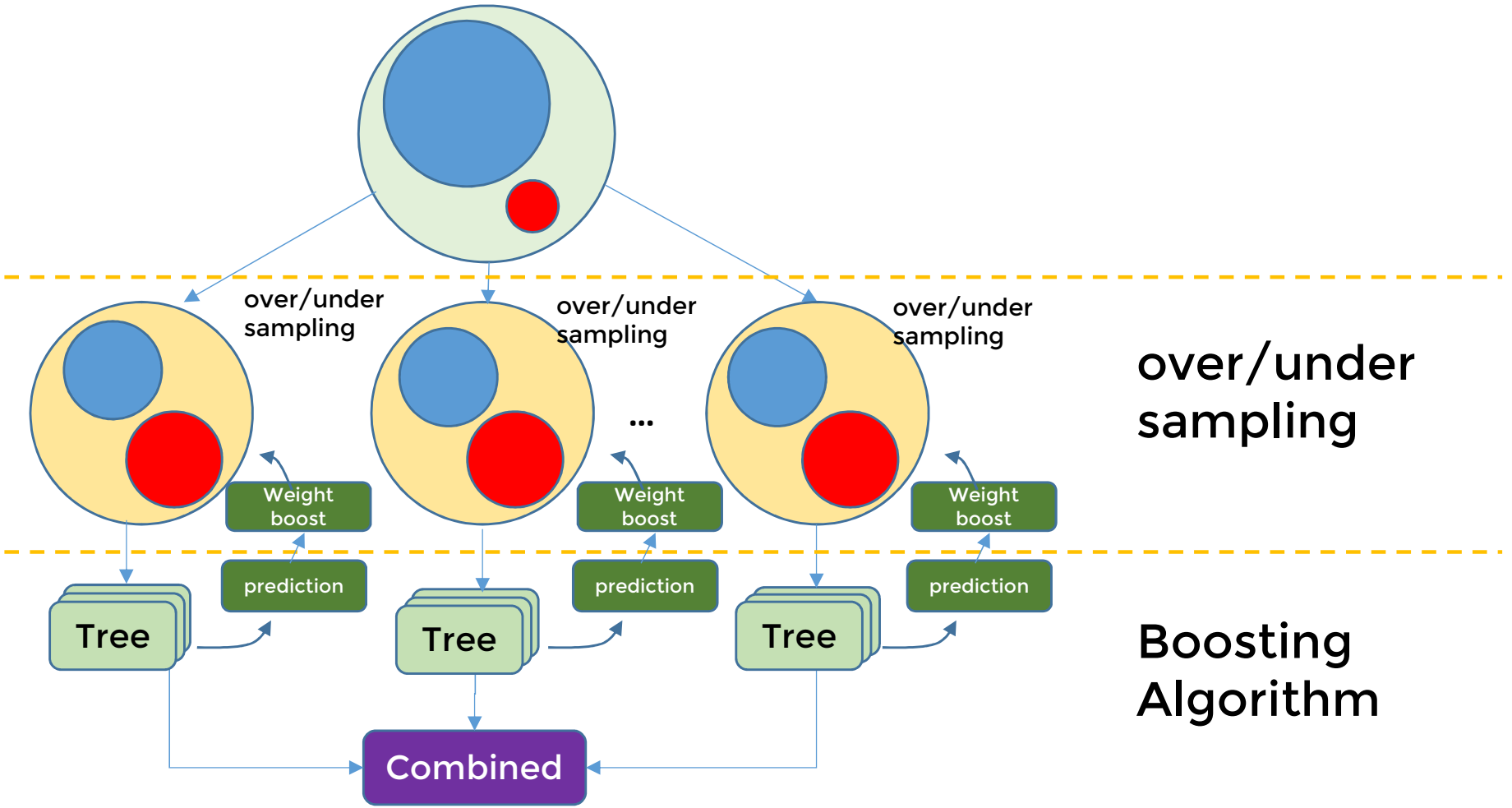
6: $\quad$ Learn H_i using $\mathcal{P}$ and $\mathcal{N}_i$. H_i is an AdaBoost ensemble with s_i weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i,j}$. The ensemble's threshold is θ_i, i.e. $H_i(x) = \text{sgn}\left(\sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \theta_i\right)$.

7: **until** $i = T$

8: Output: An ensemble: $H(x) = \text{sgn}\left(\sum_{i=1}^{T} \sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \sum_{i=1}^{T} \theta_i\right)$.

Liu, X. Y., Wu, J., & Zhou, Z. H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539-550.

EasyEnsemble



RUS-Boost

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *40*(1), 185-197.

Algorithm RUSBoost

Given: Set S of examples $(x_1, y_1), ..., (x_m, y_m)$ with minority class $y^r \in Y$, $|Y| = 2$

Weak learner, $WeakLearn$

Number of iterations, T

Desired percentage of total instances to be represented by the minority class, N

1 Initialize $D_1(i) = \frac{1}{m}$ for all i.

2 Do for $t = 1, 2, ..., T$

a Create temporary training dataset S'_t with distribution D'_t using random undersampling

b Call $WeakLearn$, providing it with examples S'_t and their weights D'_t.

c Get back a hypothesis $h_t : X \times Y \rightarrow [0, 1]$.

d Calculate the pseudo-loss (for S and D_t):

$$\epsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y)).$$

e Calculate the weight update parameter:

$$\alpha_t = \frac{\epsilon_t}{1 - \epsilon_t}.$$

f Update D_t:

$$D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y:y\neq y_i))}.$$

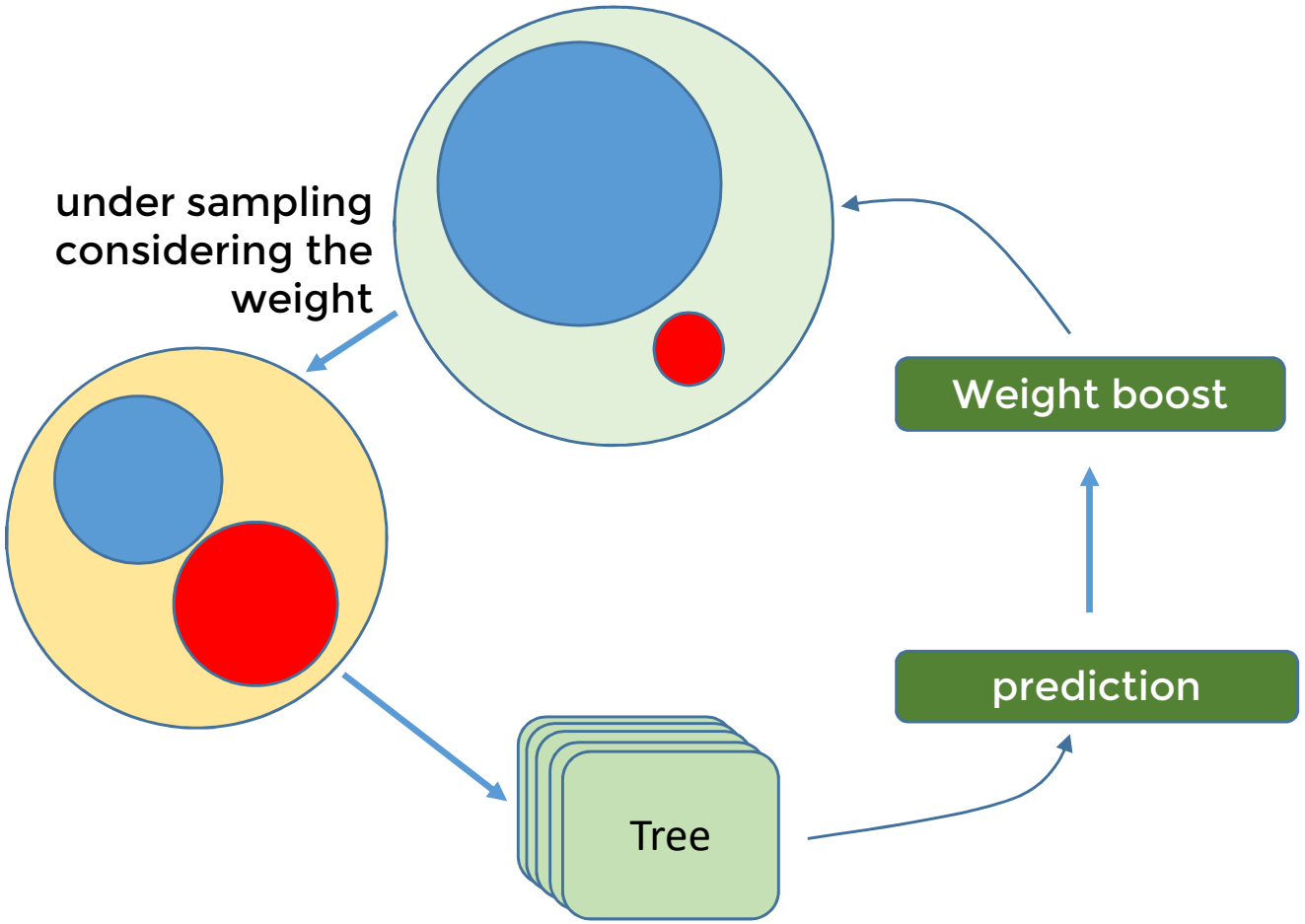
g Normalize D_{t+1}: Let $Z_t = \sum_i D_{t+1}(i)$.

$$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}.$$

3 Output the final hypothesis:

$$H(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}.$$

RUS-Boost


under sampling
considering the
weight
Weight boost
prediction
Tree

Comparative Study

A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches

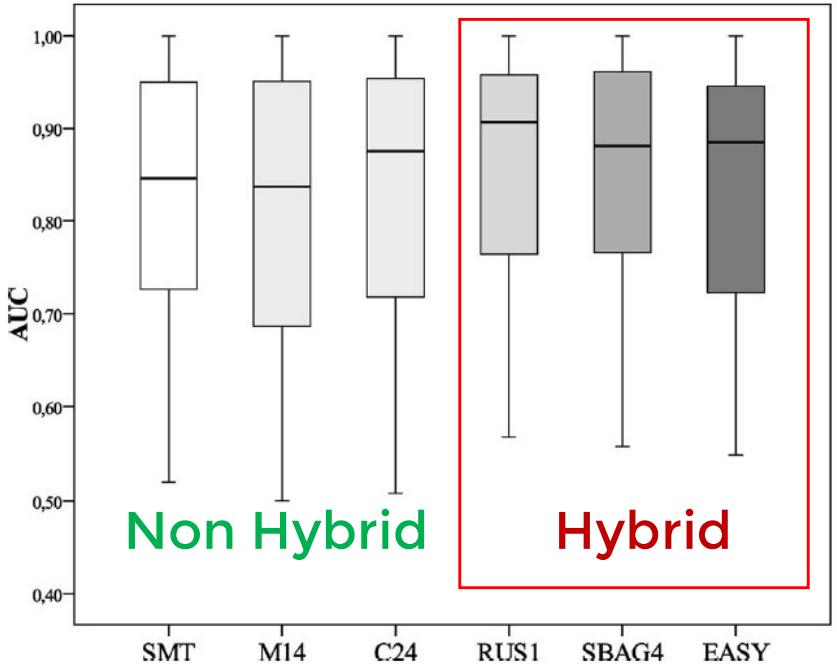
Galar et al. (2011)

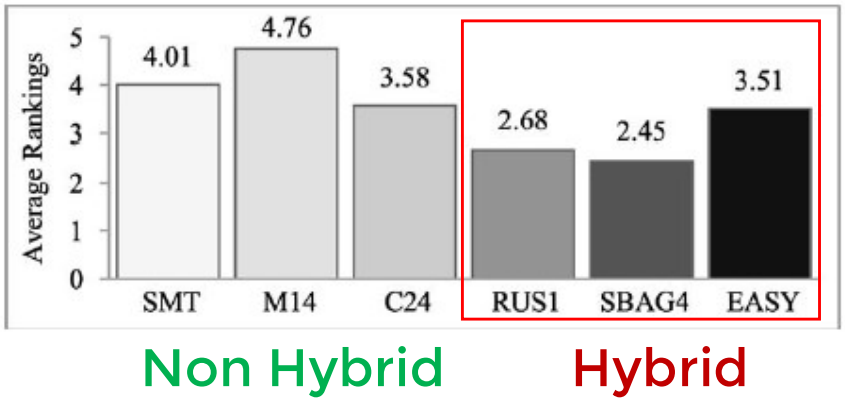
Abbr.	Method
SMT	SMOTE
M14	AdaBoost.M2 ($T = 40$)
C24	AdaC2 ($T = 40$)
RUS1	RUSBoost ($T = 10$)
SBAG4	SMOTEBagging ($T = 40$)
EASY	EasyEnsemble

Non Hybrid

Hybrid Approach

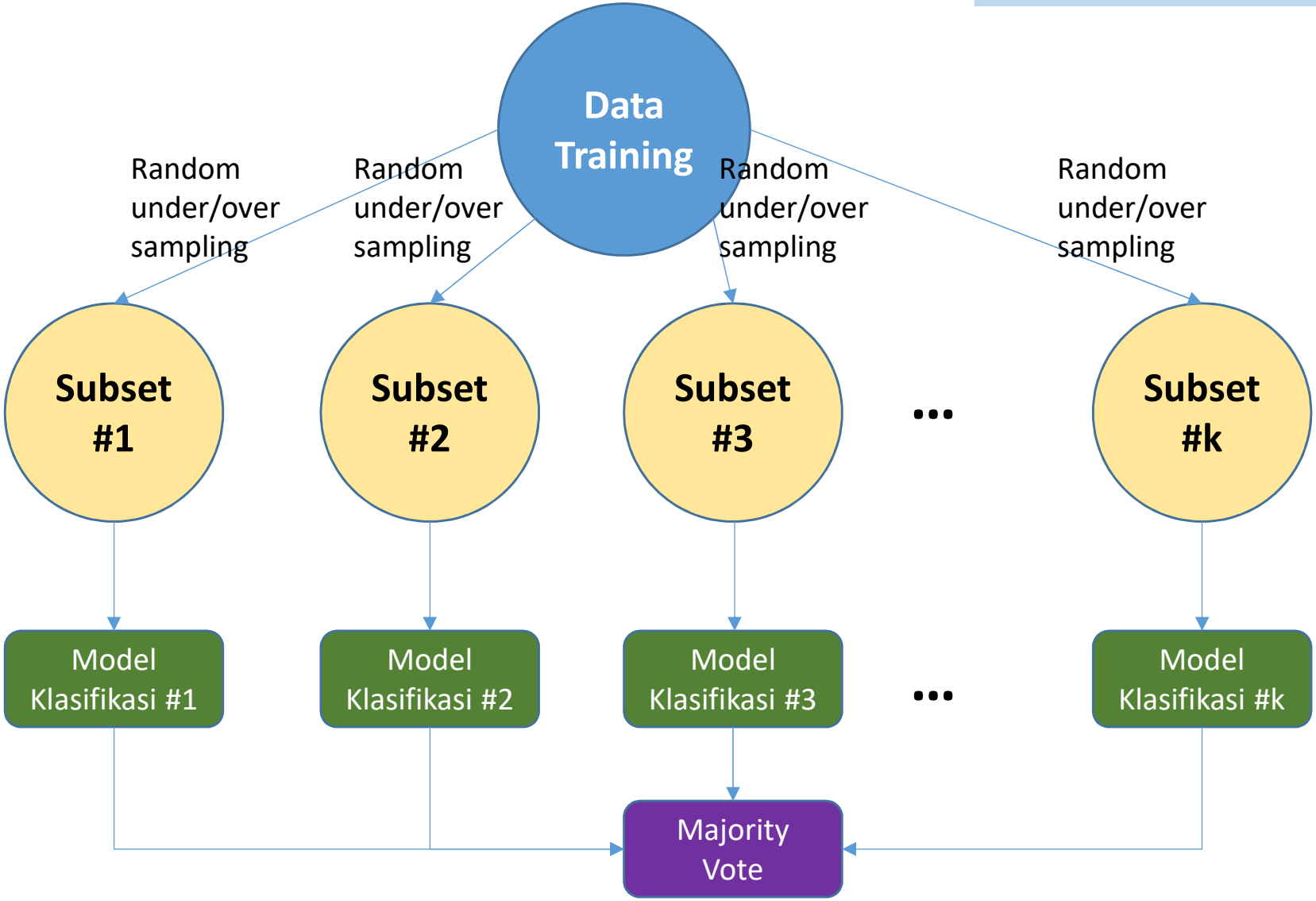
Comparative Study





- The study was involving 44 binary data-sets from KEEL data-set repository
- AUC was used as the performance measurement
- Hybrid approach outperform the conventional ones

Under/Over-Bagging



EasyEnsemble

Algorithm 1 The `EasyEnsemble` algorithm.

1: {Input: A set of minority class examples $\mathcal{P}$, a set of majority class examples $\mathcal{N}$, $|\mathcal{P}| < |\mathcal{N}|$, the number of subsets T to sample from $\mathcal{N}$, and s_i, the number of iterations to train an AdaBoost ensemble H_i}

2: $i \Leftarrow 0$

3: **repeat**

4: $\quad i \Leftarrow i + 1$

5: $\quad$ Randomly sample a subset $\mathcal{N}_i$ from $\mathcal{N}$, $|\mathcal{N}_i| = |\mathcal{P}|$.

6: $\quad$ Learn H_i using $\mathcal{P}$ and $\mathcal{N}_i$. H_i is an AdaBoost ensemble with s_i weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i,j}$. The ensemble's threshold is θ_i, i.e. $H_i(x) = \text{sgn}\left(\sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \theta_i\right)$.

7: **until** $i = T$

8: Output: An ensemble: $H(x) = \text{sgn}\left(\sum_{i=1}^{T} \sum_{j=1}^{s_i} \alpha_{i,j} h_{i,j}(x) - \sum_{i=1}^{T} \theta_i\right)$.

Liu, X. Y., Wu, J., & Zhou, Z. H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539-550.

RUS-Boost

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *40*(1), 185-197.

Algorithm RUSBoost

Given: Set S of examples $(x_1, y_1), ..., (x_m, y_m)$ with minority class $y^r \in Y$, $|Y| = 2$

Weak learner, $WeakLearn$

Number of iterations, T

Desired percentage of total instances to be represented by the minority class, N

1 Initialize $D_1(i) = \frac{1}{m}$ for all i.

2 Do for $t = 1, 2, ..., T$

a Create temporary training dataset S'_t with distribution D'_t using random undersampling

b Call $WeakLearn$, providing it with examples S'_t and their weights D'_t.

c Get back a hypothesis $h_t : X \times Y \rightarrow [0, 1]$.

d Calculate the pseudo-loss (for S and D_t):

$$\epsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y)).$$

e Calculate the weight update parameter:

$$\alpha_t = \frac{\epsilon_t}{1 - \epsilon_t}.$$

f Update D_t:

$$D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y:y\neq y_i))}.$$

g Normalize D_{t+1}: Let $Z_t = \sum_i D_{t+1}(i)$.

$$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}.$$

3 Output the final hypothesis:

$$H(x) = \underset{y \in Y}{\text{argmax}} \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}.$$

- Implementasi RUS-Boost dan perbandingan ketepatan prediksinya dengan pohon klasifikasi tunggal
  - Gunakan file “RUS-Boost.R”

Pohon Tunggal

```
Confusion Matrix and Statistics

          Reference
Prediction    no   yes
       no  10551   675
       yes   413   717

               Accuracy : 0.9119
                 95% CI : (0.9068, 0.9169)
    No Information Rate : 0.8873
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5202
 Mcnemar's Test P-Value : 2.518e-15

            Sensitivity : 0.51509
            Specificity : 0.96233
         Pos Pred Value : 0.63451
         Neg Pred Value : 0.93987
             Prevalence : 0.11266
         Detection Rate : 0.05803
   Detection Prevalence : 0.09145
      Balanced Accuracy : 0.73871

       'Positive' Class : yes
```

RUS-Boost

```
Confusion Matrix and Statistics

          Reference
Prediction    no  yes
       no  9384  132
       yes 1580 1260

               Accuracy : 0.8614
                 95% CI : (0.8552, 0.8675)
    No Information Rate : 0.8873
    P-Value [Acc > NIR] : 1

                  Kappa : 0.5234
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9052
            Specificity : 0.8559
         Pos Pred Value : 0.4437
         Neg Pred Value : 0.9861
             Prevalence : 0.1127
         Detection Rate : 0.1020
   Detection Prevalence : 0.2298
      Balanced Accuracy : 0.8805

       'Positive' Class : yes
```

terima kasih

bagusco@apps.ipb.ac.id

IPB University
Bogor Indonesia