# Моделі статистичного навчання: моделі на основі дерев

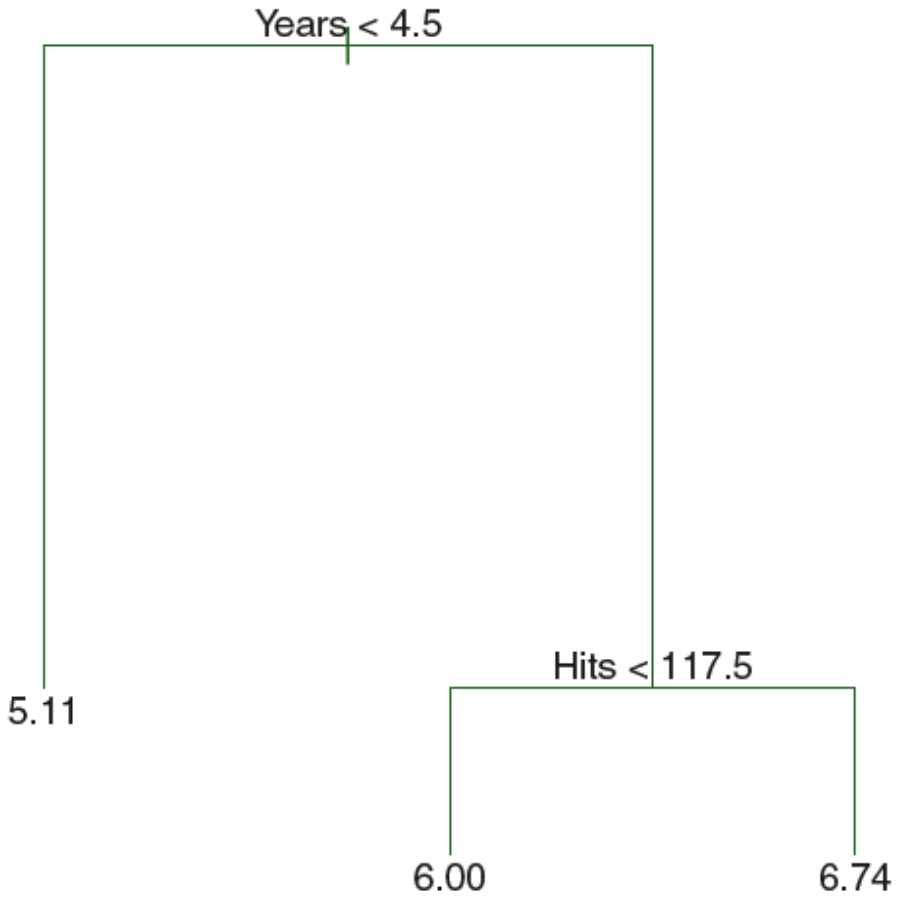
# Дерева рішень (для проблем регресії)

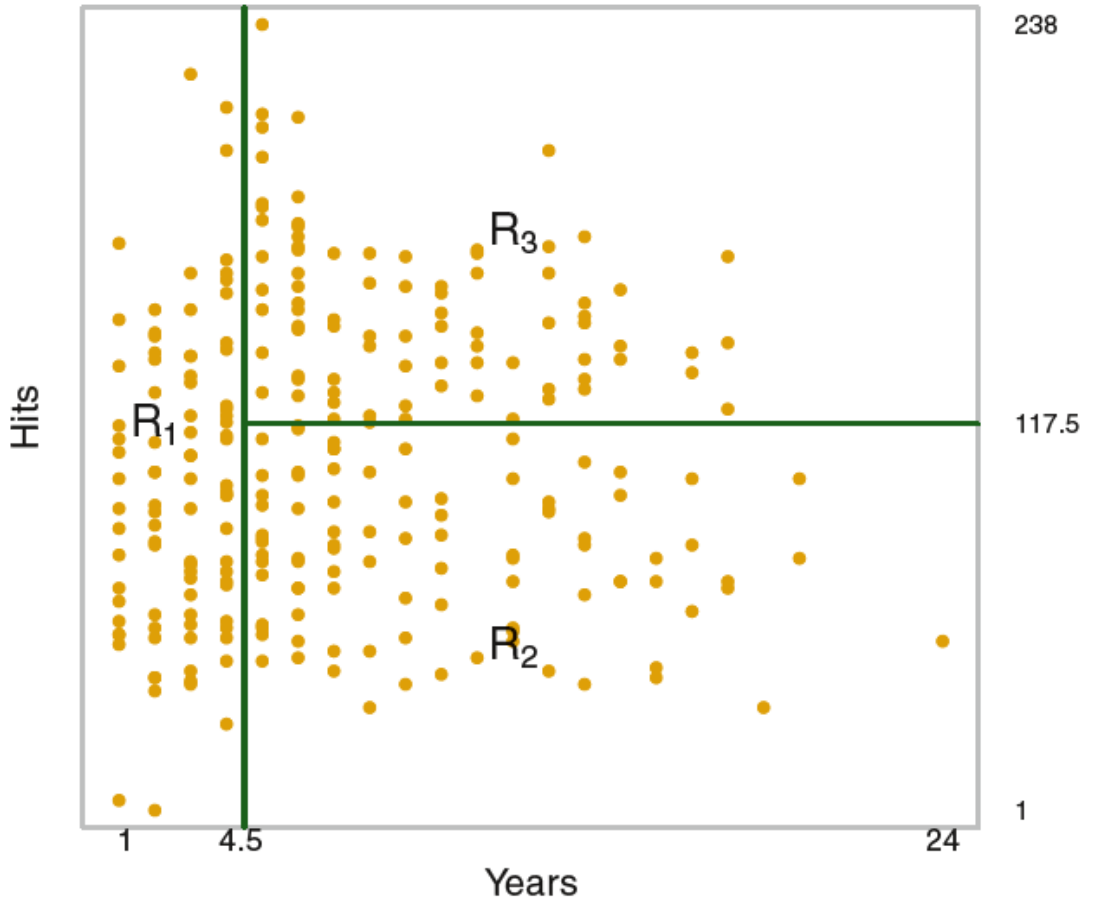
Дерева рішень (для проблем регресії)

Приклад. Передбачення зарплати бейсболістів (Salary) за стажем гравця (Years) та кількістю влучань (Hits), використовуючи набір даних Hitters.

# Дерева рішень (для проблем регресії)

Приклад. Передбачення зарплати бейсболістів (Salary) за стажем гравця (Years) та кількістю влучань (Hits), використовуючи набір даних Hitters.



Інтерпретація:

Інтерпретація: Years є найважливішим фактором при визначенні зарплати: гравці з меншим досвідом отримують істотно нижчі зарплати, ніж досвідченіші гравці.

Інтерпретація: Years є найважливішим фактором при визначенні зарплати: гравці з меншим досвідом отримують істотно нижчі зарплати, ніж досвідченіші гравці. Для менш досвідченого гравця, Hits, які він зробив попереднього року відіграє незначну роль у його зарплаті.

Інтерпретація: Years є найважливішим фактором при визначенні зарплати: гравці з меншим досвідом отримують істотно нижчі зарплати, ніж досвідченіші гравці. Для менш досвідченого гравця, Hits, які він зробив попереднього року відіграє незначну роль у його зарплаті. Але серед гравців, які були у вищій лізі протягом п’яти і більше років, Hits зроблені у попередньому році впливають на зарплату: гравці, які зробили більше Hits минулого року мають істотно вищу зарплату.

Інтерпретація: Years є найважливішим фактором при визначенні зарплати: гравці з меншим досвідом отримують істотно нижчі зарплати, ніж досвідченіші гравці. Для менш досвідченого гравця, Hits, які він зробив попереднього року відіграє незначну роль у його зарплаті. Але серед гравців, які були у вищій лізі протягом п’яти і більше років, Hits зроблені у попередньому році впливають на зарплату: гравці, які зробили більше Hits минулого року мають істотно вищу зарплату.

Побудова дерева (два кроки):

Інтерпретація: Years є найважливішим фактором при визначенні зарплати: гравці з меншим досвідом отримують істотно нижчі зарплати, ніж досвідченіші гравці. Для менш досвідченого гравця, Hits, які він зробив попереднього року відіграє незначну роль у його зарплаті. Але серед гравців, які були у вищій лізі протягом п’яти і більше років, Hits зроблені у попередньому році впливають на зарплату: гравці, які зробили більше Hits минулого року мають істотно вищу зарплату.

Побудова дерева (два кроки):

1. Поділити весь простір предикторів $X_1$, $X_2$, …, $X_p$ на $J$ областей, що не перетинаються $R_1$, $R_2$,…, $R_J$.

Інтерпретація: Years є найважливішим фактором при визначенні зарплати: гравці з меншим досвідом отримують істотно нижчі зарплати, ніж досвідченіші гравці. Для менш досвідченого гравця, Hits, які він зробив попереднього року відіграє незначну роль у його зарплаті. Але серед гравців, які були у вищій лізі протягом п’яти і більше років, Hits зроблені у попередньому році впливають на зарплату: гравці, які зробили більше Hits минулого року мають істотно вищу зарплату.

Побудова дерева (два кроки):

1. Поділити весь простір предикторів $X_1$, $X_2$, …, $X_p$ на $J$ областей, що не перетинаються $R_1$, $R_2$,…, $R_J$.

2. Для кожного спостереження, що попадає в $R_i$ ми будуємо однакові передбачення, що дорівнюють, наприклад, середньому значенню залежної змінної в цій області.

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Рекурсивний бінарний поділ: вибираємо предиктор $X_j$ та точку поділу $s$ так, щоб розбиття на області $R_1(j, s) = \{X \mid X_j < s\}$ та $R_2(j, s) = \{X \mid X_j \geq s\}$ давало найменший RSS, тобто вибираємо $j$ та $s$ так, щоб мінімізувати

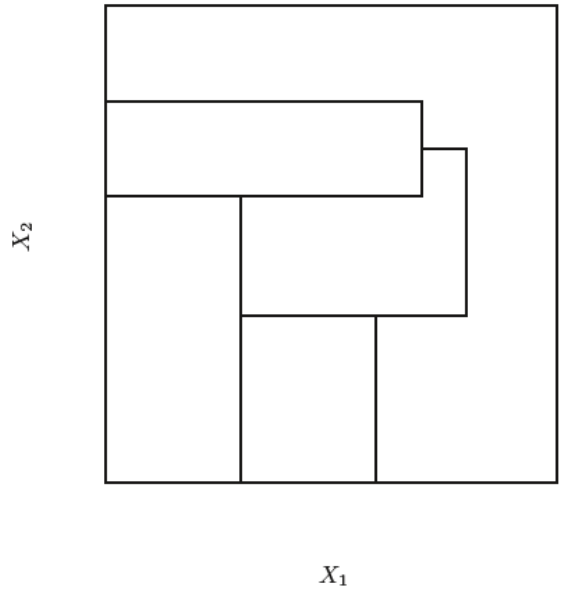
$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

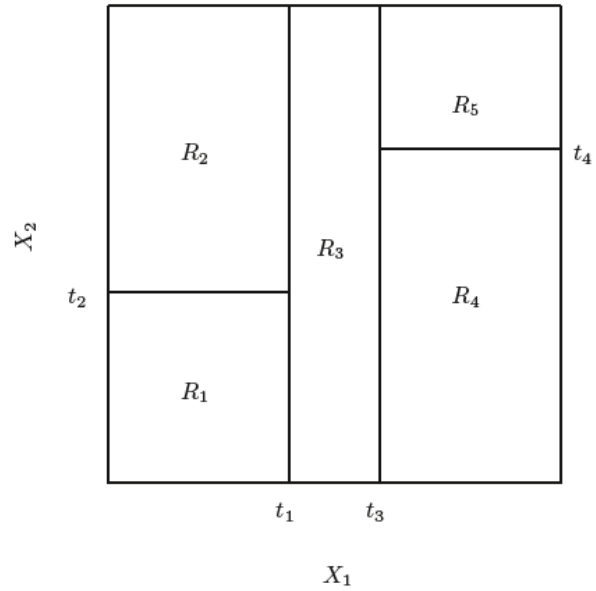
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

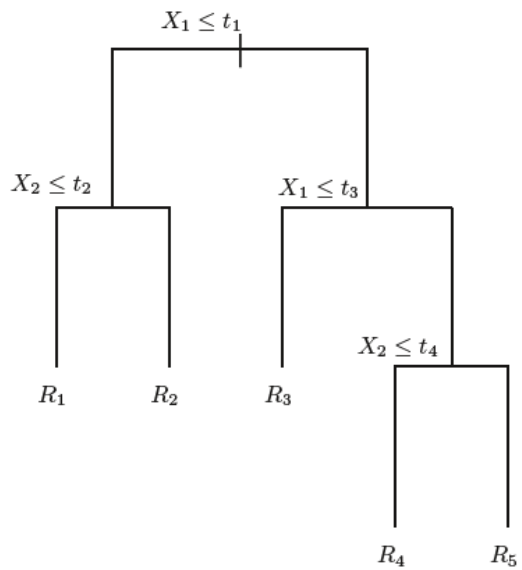
Рекурсивний бінарний поділ: вибираємо предиктор $X_j$ та точку поділу $s$ так, щоб розбиття на області $R_1(j, s) = \{X \mid X_j < s\}$ та $R_2(j, s) = \{X \mid X_j \geq s\}$ давало найменший RSS, тобто вибираємо $j$ та $s$ так, щоб мінімізувати

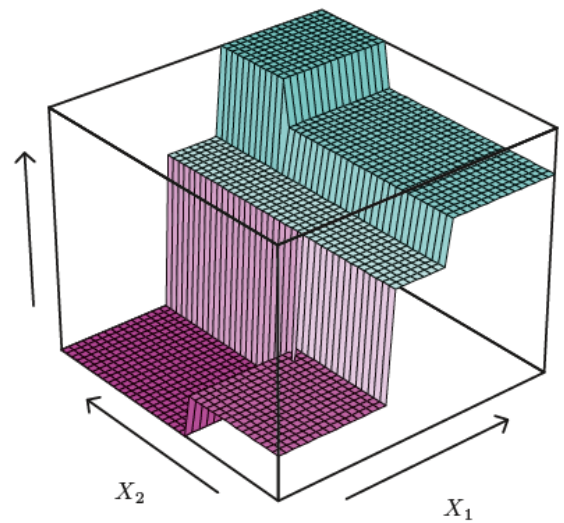
$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Далі продовжуємо так само на одній з областей поки не досягнемо потрібного правила зупинки.


$X_2$
$X_1$


$R_1$
$R_2$
$R_3$
$R_4$
$R_5$
$t_1$
$t_2$
$t_3$
$t_4$
$X_2$
$X_1$


$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
$R_1$
$R_2$
$R_3$
$R_4$
$R_5$


$X_2$
$X_1$

Обрізка дерева.

Обрізка дерева.

Cost complexity pruning:

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Обрізка дерева.

Cost complexity pruning:

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Алгоритм:

Обрізка дерева.

Cost complexity pruning:

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Алгоритм:

1. Використовуючи рекурсивний бінарний поділ, виростити велике дерево на тренувальних даних, зупинившись лише тоді, коли кожна область містить менше спостережень, ніж деяка мінімальна кількість.

Обрізка дерева.

Cost complexity pruning:

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Алгоритм:

1. Використовуючи рекурсивний бінарний поділ, виростити велике дерево на тренувальних даних, зупинившись лише тоді, коли кожна область містить менше спостережень, ніж деяка мінімальна кількість.

2. Побудувати послідовність «найкращих» піддерев як функцію від $\alpha$.

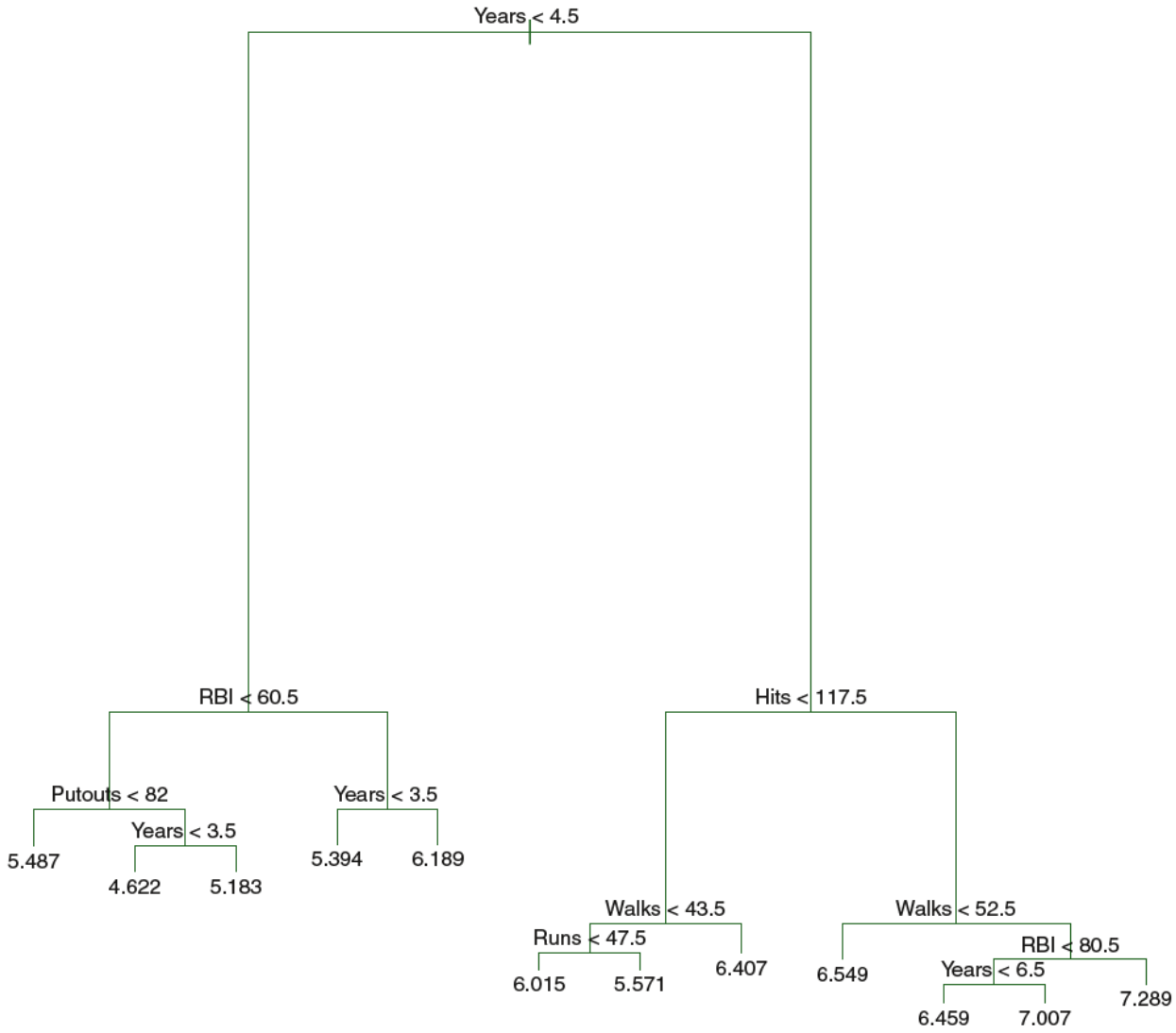
3. Використовуючи перехресну перевірку вибрати $\alpha$:

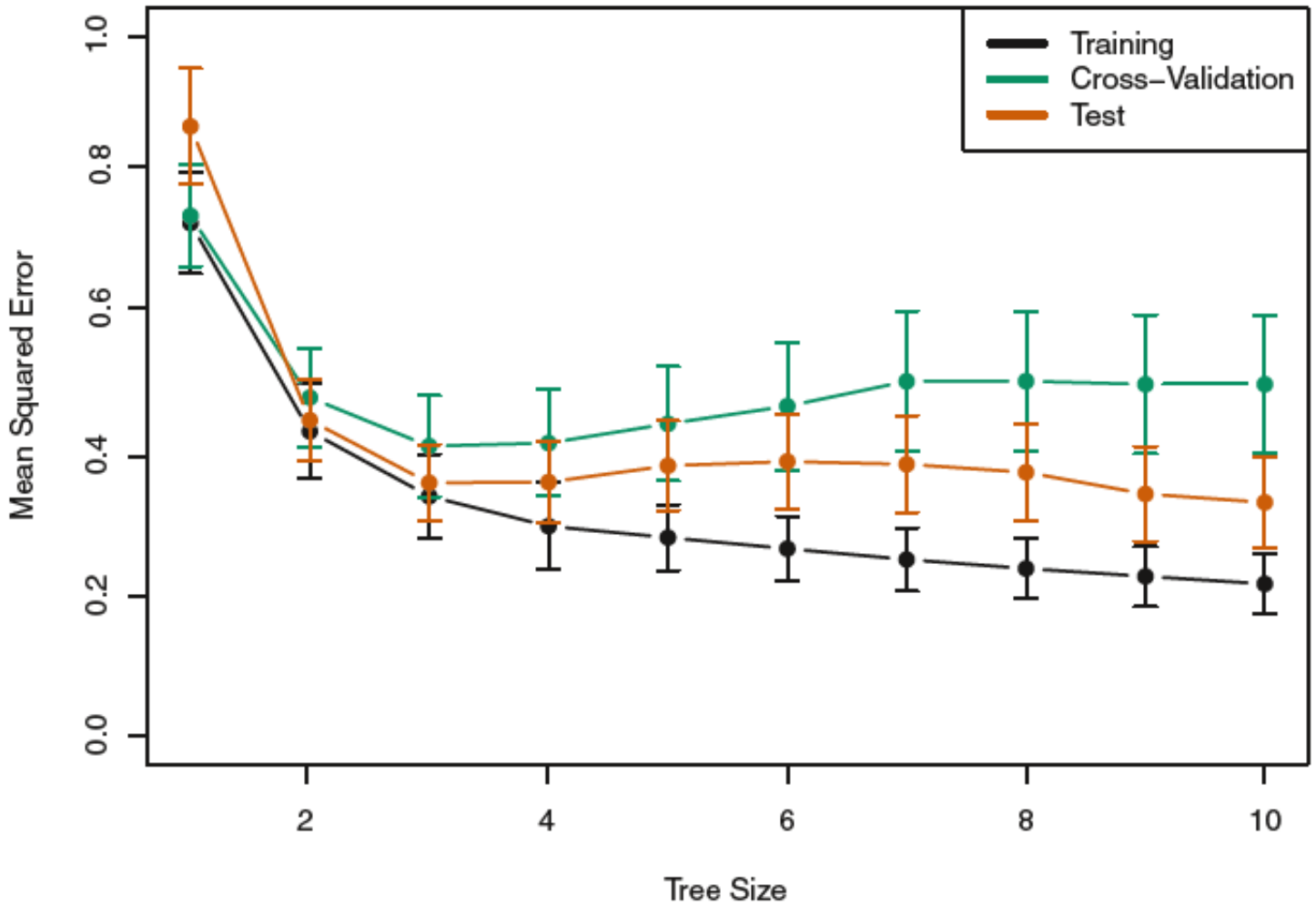
3. Використовуючи перехресну перевірку вибрати $\alpha$:

4. Повернути піддерево з кроку 2, яке відповідає вибраному значенню $\alpha$.

3. Використовуючи перехресну перевірку вибрати $\alpha$:

4. Повернути піддерево з кроку 2, яке відповідає вибраному значенню $\alpha$.




Training
Cross–Validation
Test
Mean Squared Error
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Tree Size

# Дерева рішень (для проблем класифікації)

Дерева рішень (для проблем класифікації)

Критерій для бінарного поділу:

Дерева рішень (для проблем класифікації)

Критерій для бінарного поділу:

Частота помилок класифікації (Classification error rate)

$$E = 1 - \max_k(\hat{p}_{mk})$$

Дерева рішень (для проблем класифікації)

Критерій для бінарного поділу:

Частота помилок класифікації (Classification error rate)

$$E = 1 - \max_k(\hat{p}_{mk})$$

Індекс Gini (Gini index)

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Дерева рішень (для проблем класифікації)

Критерій для бінарного поділу:

Частота помилок класифікації (Classification error rate)

$$E = 1 - \max_k(\hat{p}_{mk})$$

Індекс Gini (Gini index)

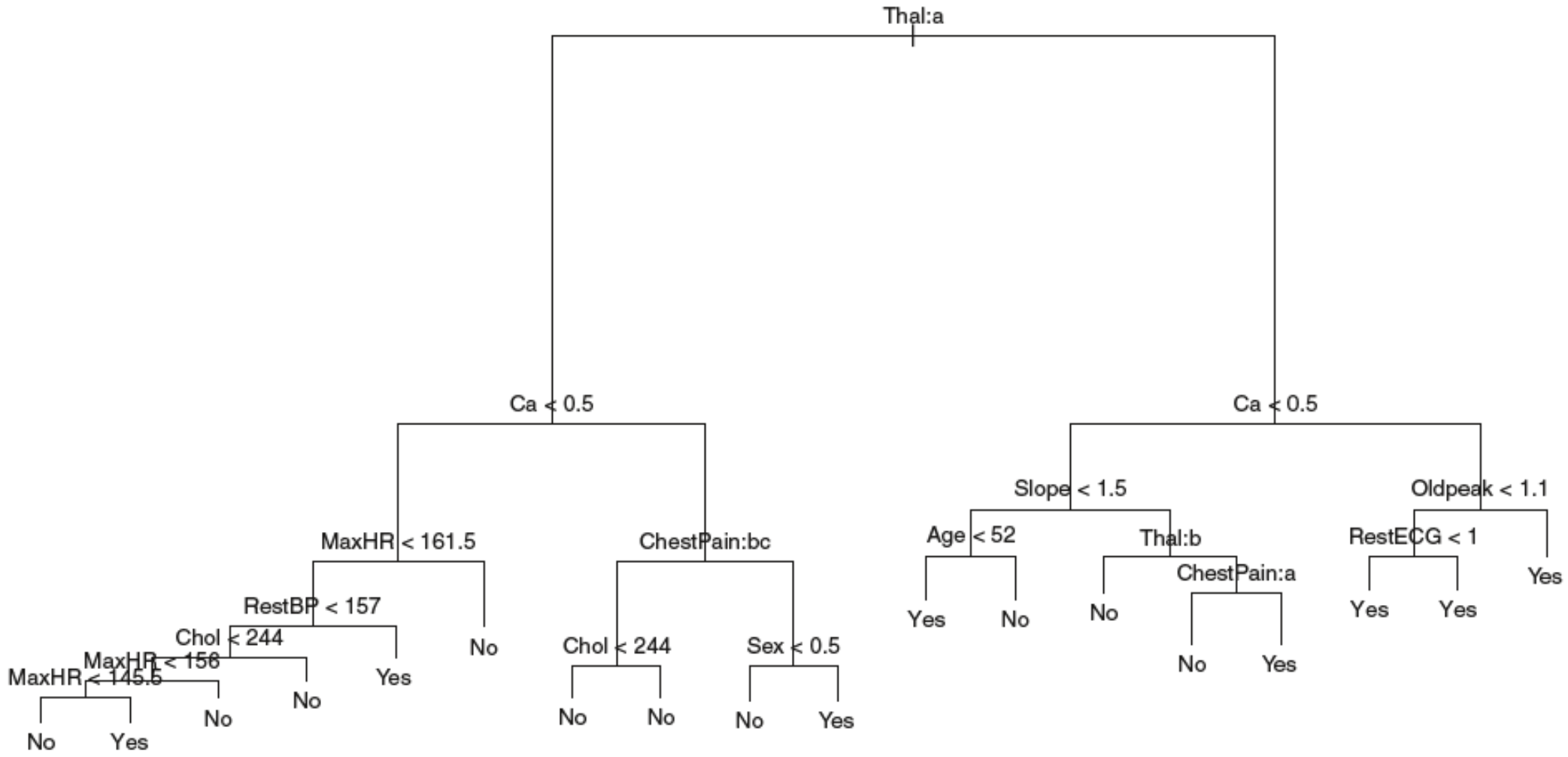
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

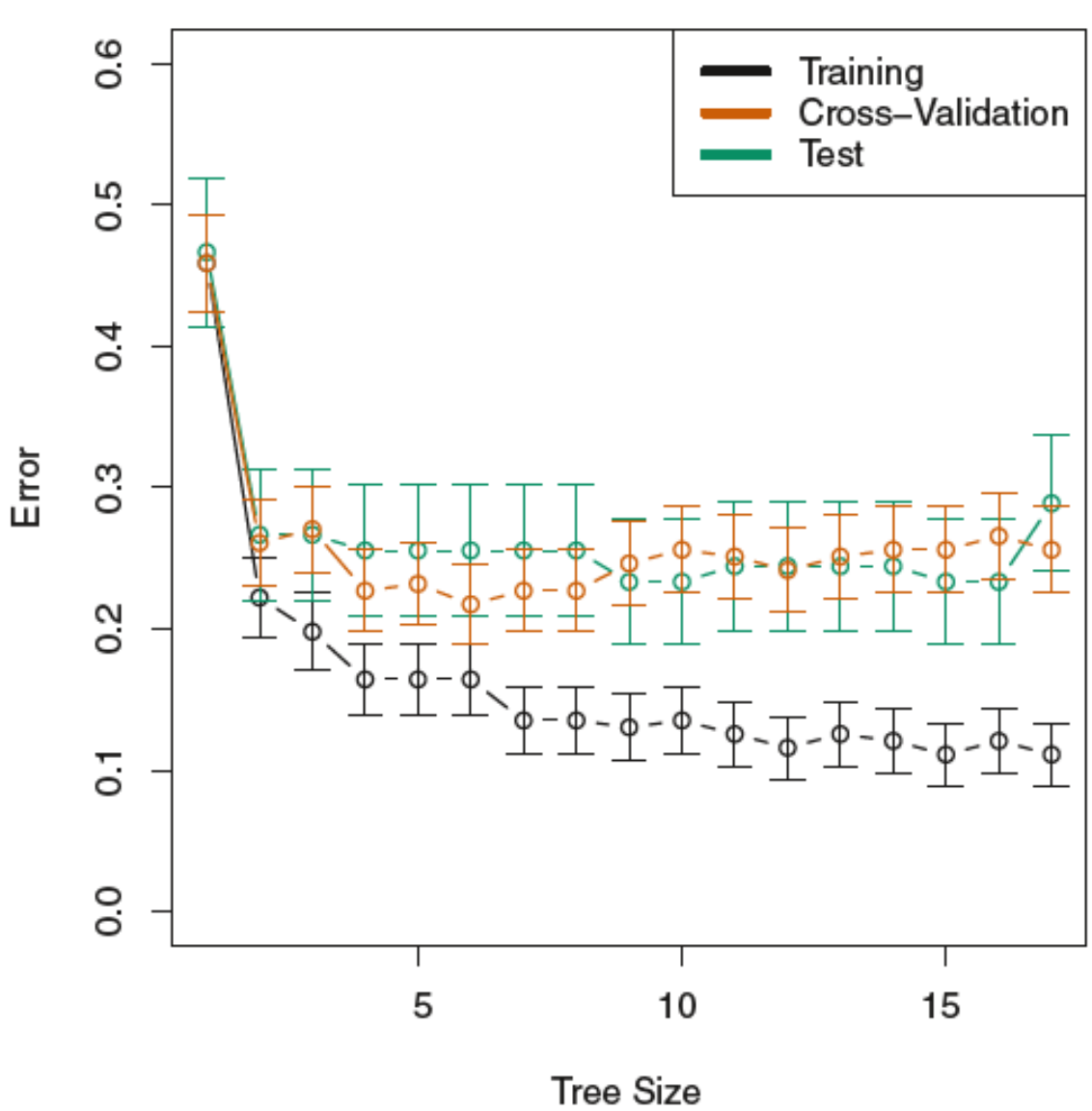
Перехресна ентропія (cross-entropy index)

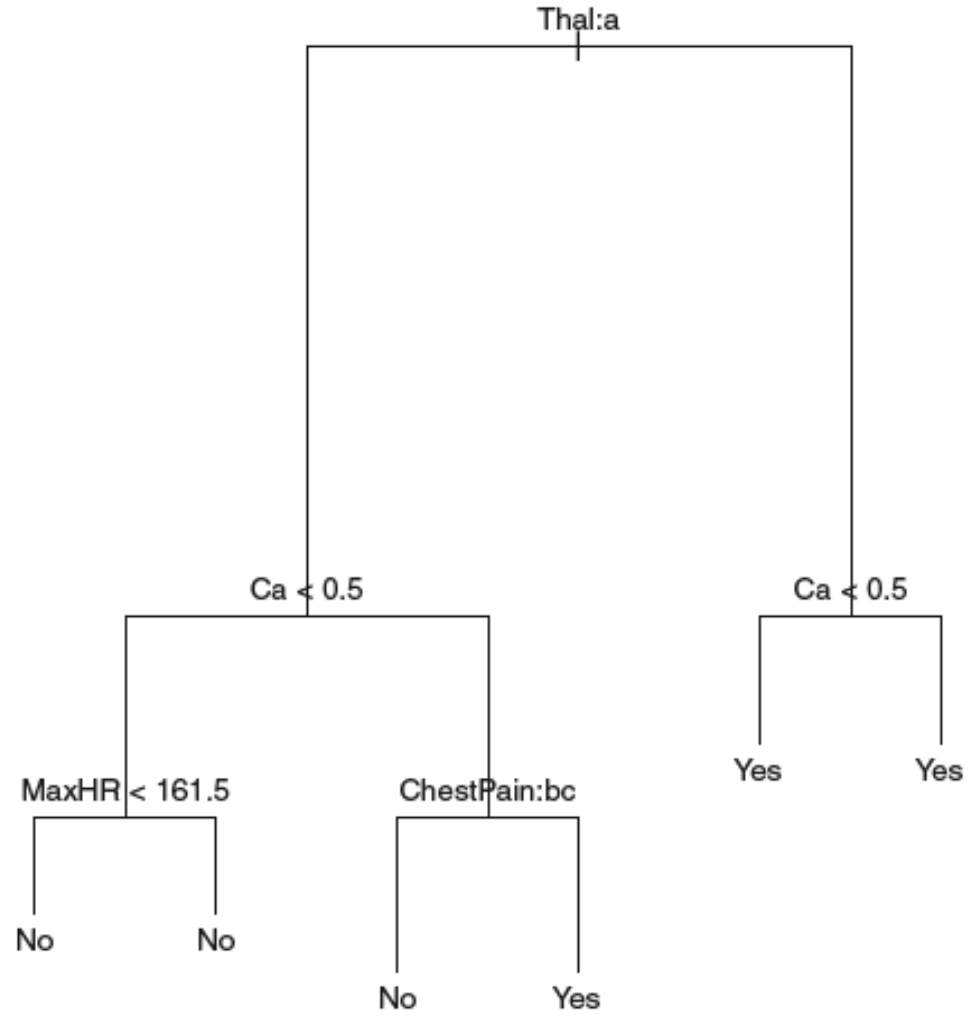
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Приклад. Набір даних Heart з 13 предикторами.

Приклад. Набір даних Heart з 13 предикторами.




Training
Cross–Validation
Test
Error
0.0
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
Tree Size


Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Yes
Yes
No
No
No
Yes

Порівняння лінійної регресії та дерева рішень.

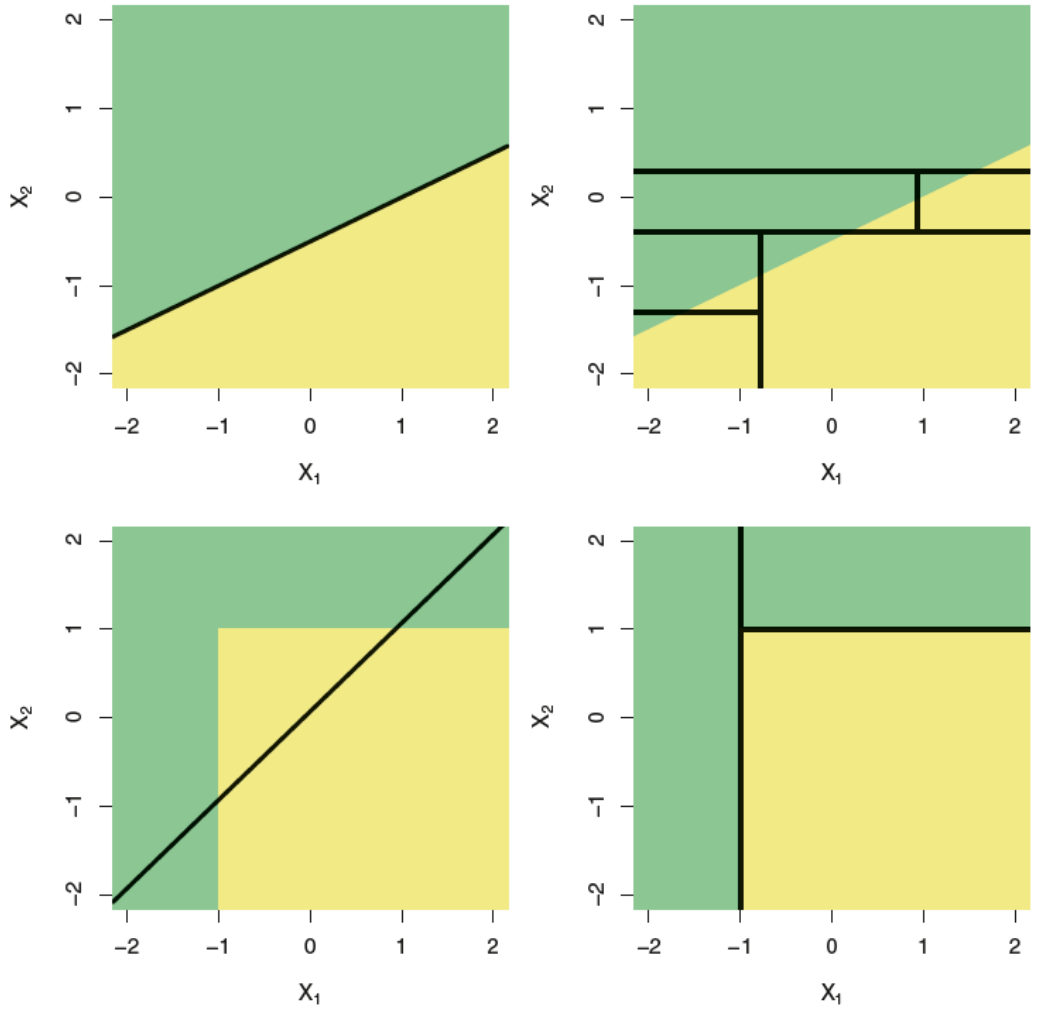
$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

$$f(X) = \sum_{m=1}^{M} c_m \cdot 1_{(X \in R_m)}$$

# Порівняння лінійної регресії та дерева рішень.

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

$$f(X) = \sum_{m=1}^{M} c_m \cdot 1_{(X \in R_m)}$$



Бутстрап агрегація.

Бутстрап агрегація.

Обчислюємо $\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)$ використовуючи $B$ різних навчальних наборів даних і усереднюємо результати

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x)$$

Бутстрап агрегація.

Обчислюємо $\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)$ використовуючи $B$ різних навчальних наборів даних і усереднюємо результати

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

OOB оцінка помилки: в середньому при бутстрапі ми використовуємо 2/3 обсягу усіх даних, отже решту можна розглянути як тестовий набір даних і для всіх дерев, для яких певне спостереження попадає в цей набір можемо оцінити помилку, усереднивши яку за всіма розглянутими деревами, отримаємо оцінку помилки.

Бутстрап агрегація.

Обчислюємо $\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)$ використовуючи $B$ різних навчальних наборів даних і усереднюємо результати

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

OOB оцінка помилки: в середньому при бутстрапі ми використовуємо 2/3 обсягу усіх даних, отже решту можна розглянути як тестовий набір даних і для всіх дерев, для яких певне спостереження попадає в цей набір можемо оцінити помилку, усереднивши яку за всіма розглянутими деревами, отримаємо оцінку помилки.

Також розглянутий метод дозволяє оцінити важливість кожного з предикторів.

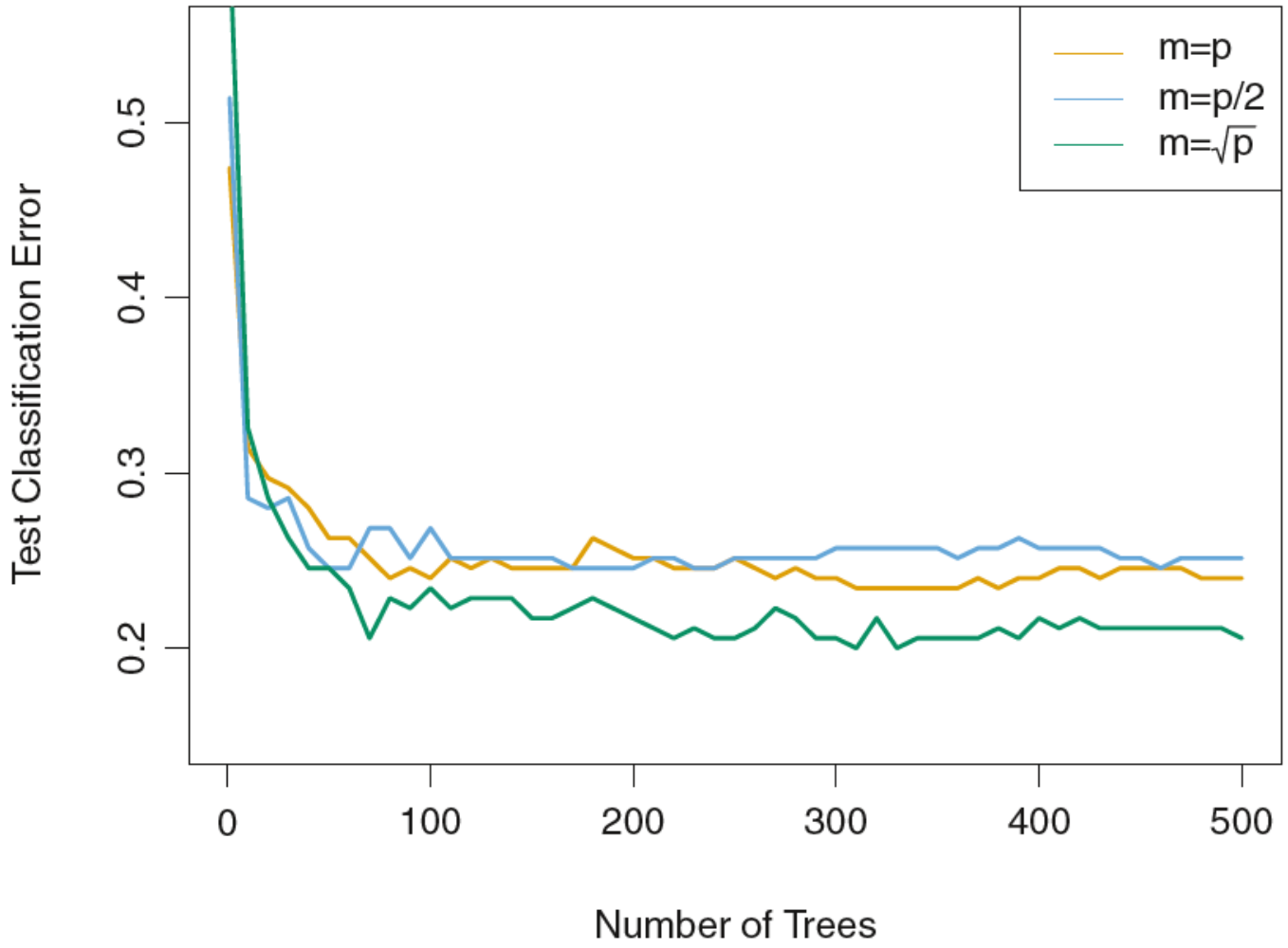
Випадкові ліси.

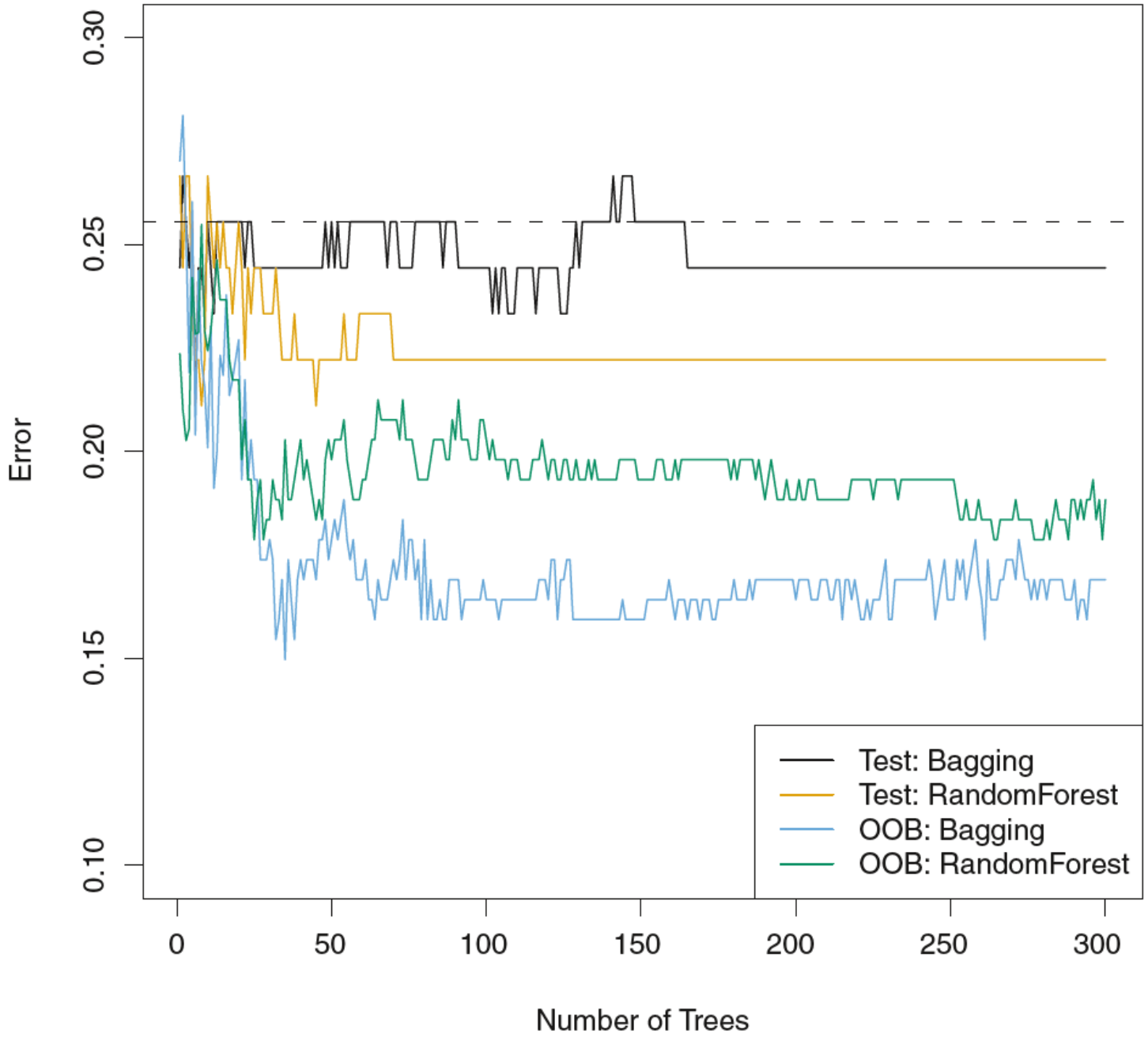
Випадкові ліси.

Ідея така ж, як і при бутстрап агрегації за винятком того, що при кожному поділі ми випадковим чином вибираємо $m$ предикторів як кандидатів для поділу.

Випадкові ліси.

Ідея така ж, як і при бутстрап агрегації за винятком того, що при кожному поділі ми випадковим чином вибираємо *m* предикторів як кандидатів для поділу.




Error
0.10
0.15
0.20
0.25
0.30
0
50
100
150
200
250
300
Number of Trees
Test: Bagging
Test: RandomForest
OOB: Bagging
OOB: RandomForest

Посилення (Boosting).

Посилення (Boosting).

Цей метод не використовує бутстрап, кожне наступне дерево будується на модифікованій множині початкових даних.

Посилення (Boosting).

Цей метод не використовує бутстрап, кожне наступне дерево будується на модифікованій множині початкових даних.

Алгоритм.

1. Покласти $\hat{f}(x) = 0$ та $r_i = y_i$ для всіх $i$ з навчального набору.

Посилення (Boosting).

Цей метод не використовує бутстрап, кожне наступне дерево будується на модифікованій множині початкових даних.

Алгоритм.

1. Покласти $\hat{f}(x) = 0$ та $r_i = y_i$ для всіх $i$ з навчального набору.

2. Для $b = 1, 2, \ldots, B$ повторити

2.1. Побудувати дерево $\hat{f}^b$ використовуючи $d$ поділів на даних $(X, r)$

Посилення (Boosting).

Цей метод не використовує бутстрап, кожне наступне дерево будується на модифікованій множині початкових даних.

Алгоритм.

1. Покласти $\hat{f}(x) = 0$ та $r_i = y_i$ для всіх $i$ з навчального набору.

2. Для $b = 1, 2, \ldots, B$ повторити

2.1. Побудувати дерево $\hat{f}^b$ використовуючи $d$ поділів на даних $(X, r)$

2.2. Оновити $\hat{f}$ за правилом

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

2.3. Оновити залишки

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

2.3. Оновити залишки

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

3. Вивести підсилену модель

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Метод підсилення грунтується на трьох параметрах:

1. Кількість дерев $B$. На відміну від бутстрап агрегації та випадкових лісів, посилення може призвести до переоцінки, якщо $B$ занадто велике. Ми використовуємо перехресну перевірку для вибору $B$.

Метод підсилення грунтується на трьох параметрах:

1. Кількість дерев $B$. На відміну від бутстрап агрегації та випадкових лісів, посилення може призвести до переоцінки, якщо $B$ занадто велике. Ми використовуємо перехресну перевірку для вибору $B$.

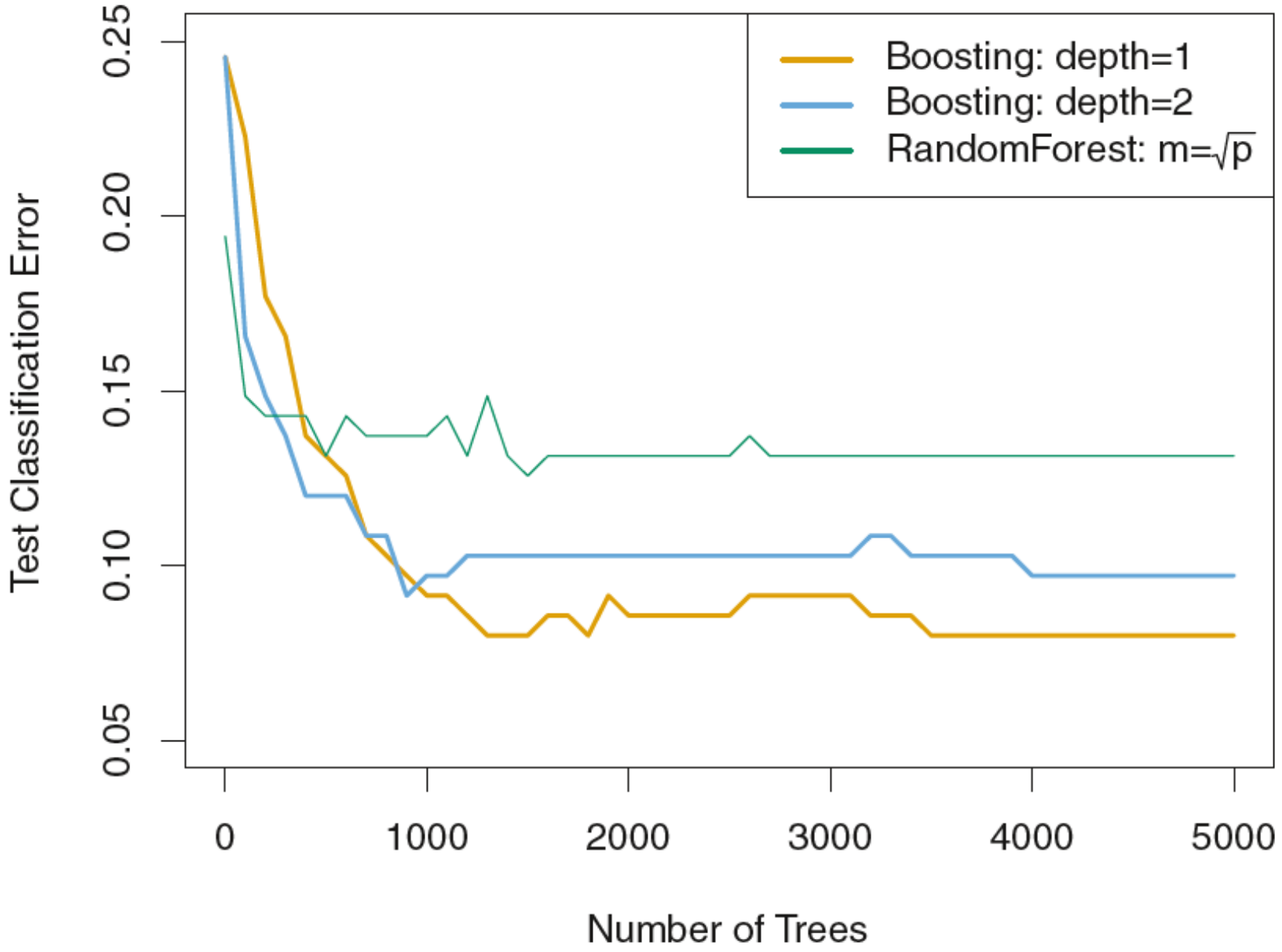
2. Параметр стиснення $\lambda$, невелике додатне число. Цей параметр контролює швидкість, з якою навчається підсилення. Типовими значеннями є 0,01 або 0,001, і правильний вибір може залежати від завдання. Дуже мале значення $\lambda$ може вимагати використання дуже великого B для досягнення хороших показників.

Метод підсилення грунтується на трьох параметрах:

1. Кількість дерев $B$. На відміну від бутстрап агрегації та випадкових лісів, посилення може призвести до переоцінки, якщо $B$ занадто велике. Ми використовуємо перехресну перевірку для вибору $B$.

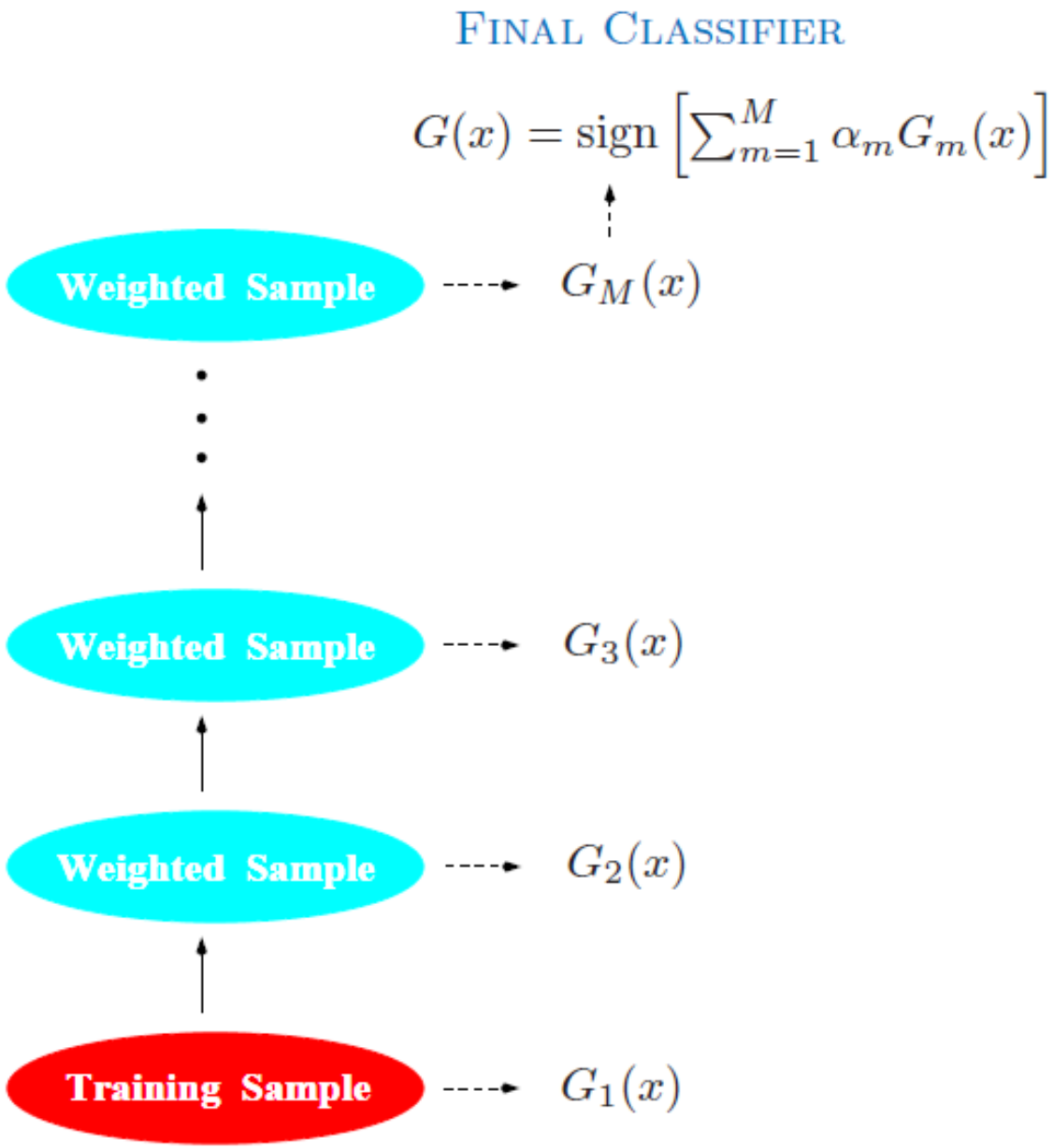
2. Параметр стиснення $\lambda$, невелике додатне число. Цей параметр контролює швидкість, з якою навчається підсилення. Типовими значеннями є 0,01 або 0,001, і правильний вибір може залежати від завдання. Дуже мале значення $\lambda$ може вимагати використання дуже великого B для досягнення хороших показників.

3. Кількість розділів кожного дерева d контролює складність методу підсилення. Часто $d = 1$ працює добре, і в цьому випадку кожне дерево складається з єдиного поділу. Більш загально d - глибина взаємодії. Цей параметр контролює порядок взаємодії підсиленої моделі, оскільки в цьому випадку розбиття може включати не більше d змінних.


Boosting: depth=1
Boosting: depth=2
RandomForest: m=√p
Test Classification Error
0.05
0.10
0.15
0.20
0.25
0
1000
2000
3000
4000
5000
Number of Trees

# AdaBoost.M1. Алгоритм.

# AdaBoost.M1. Алгоритм.



# AdaBoost.M1. Алгоритм.

1. Ініціалізуємо ваги для спостережень $w_i = 1/N$, $i = 1, 2, \ldots, N$.

AdaBoost.M1. Алгоритм.

1. Ініціалізуємо ваги для спостережень $w_i = 1/N$, $i = 1, 2, \ldots, N$.

2. Для $m = 1 \ldots M$:

2.1. Будуємо слабкий класифікатор $G_m(x)$ на тренувальних даних з врахуванням їх ваг $w_i$.

AdaBoost.M1. Алгоритм.

1. Ініціалізуємо ваги для спостережень $w_i = 1/N$, $i = 1, 2, \ldots, N$.

2. Для $m = 1 \ldots M$:

2.1. Будуємо слабкий класифікатор $G_m(x)$ на тренувальних даних з врахуванням їх ваг $w_i$.

2.2. Обчислюємо $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

AdaBoost.M1. Алгоритм.

1. Ініціалізуємо ваги для спостережень $w_i = 1/N$, $i = 1, 2, \ldots, N$.

2. Для $m = 1 \ldots M$:

2.1. Будуємо слабкий класифікатор $G_m(x)$ на тренувальних даних з врахуванням їх ваг $w_i$.

2.2. Обчислюємо $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

2.3. Обчислюємо $$\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m).$$

AdaBoost.M1. Алгоритм.

1. Ініціалізуємо ваги для спостережень $w_i = 1/N$, $i = 1, 2, \ldots, N$.

2. Для $m = 1 \ldots M$:

2.1. Будуємо слабкий класифікатор $G_m(x)$ на тренувальних даних з врахуванням їх ваг $w_i$.

2.2. Обчислюємо $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

2.3. Обчислюємо $$\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m).$$

2.4. Модифікуємо ваги $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.

AdaBoost.M1. Алгоритм.

1. Ініціалізуємо ваги для спостережень $w_i = 1/N$, $i = 1, 2, \ldots, N$.

2. Для $m = 1 \ldots M$:

2.1. Будуємо слабкий класифікатор $G_m(x)$ на тренувальних даних з врахуванням їх ваг $w_i$.

2.2. Обчислюємо $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

2.3. Обчислюємо $$\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m).$$

2.4. Модифікуємо ваги $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.

3. Отримуємо $$G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right].$$

Приклад. $X_1$, $X_2$, …, $X_{10}$ – незалежні стандартно нормально розподілені випадкові величини.

$$Y = \begin{cases} 1, \ якщо \ \sum_{i=1}^{10} X_i^2 > \chi_{10}^2(0.5) \\ -1, \ інакше \end{cases}$$

Приклад. $X_1$, $X_2$, …, $X_{10}$ – незалежні стандартно нормально розподілені випадкові величини.

$$Y = \begin{cases} 1, \ якщо \ \sum_{i=1}^{10} X_i^2 > \chi_{10}^2(0.5) \\ -1, \ інакше \end{cases}$$

Тренувальна вибірка складається з 2000 спостережень (близько 1000 спостережень з кожного класу) і 10000 тестових спостережень.

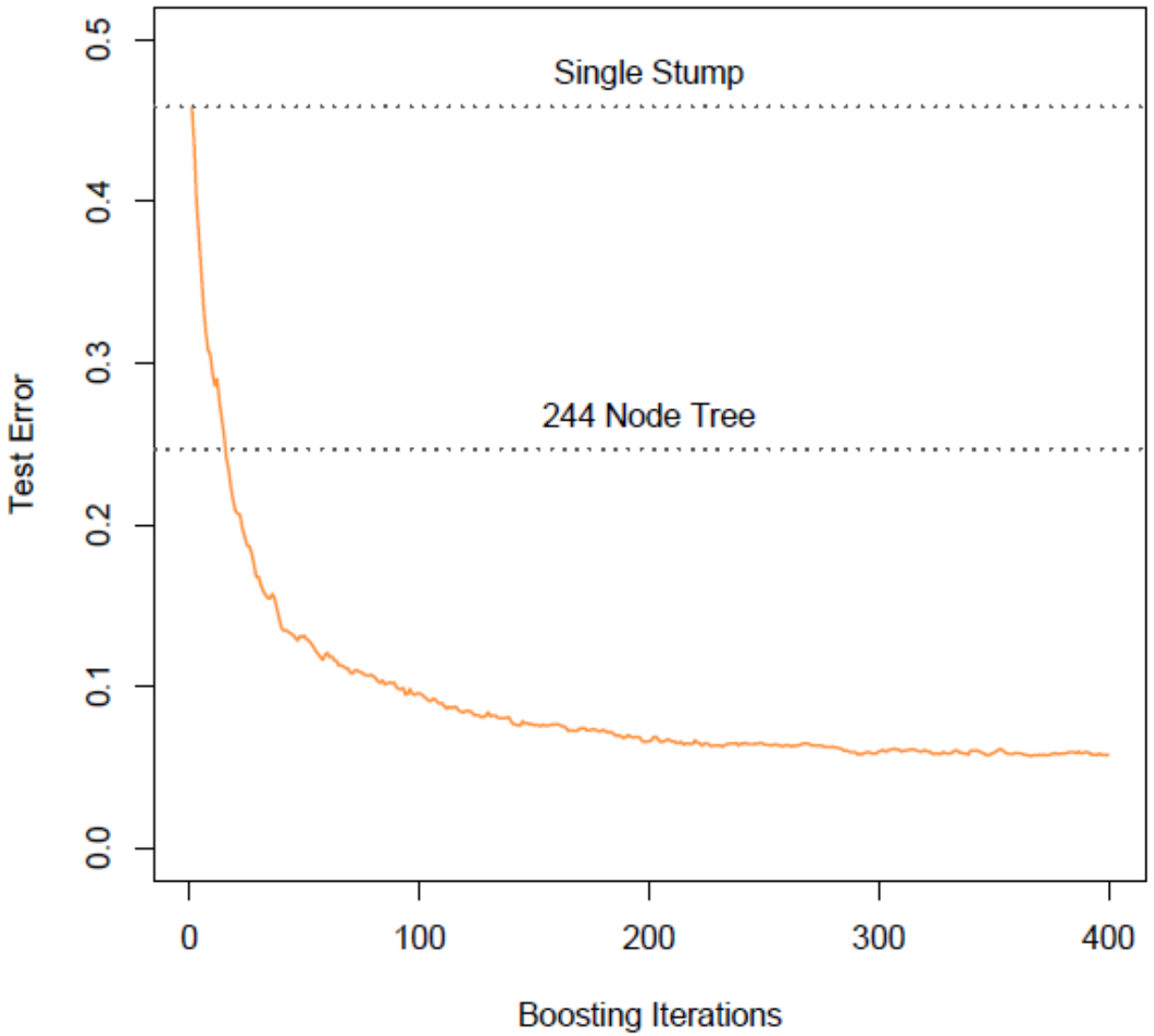
Приклад. $X_1$, $X_2$, …, $X_{10}$ – незалежні стандартно нормально розподілені випадкові величини.

$$Y = \begin{cases} 1, \ якщо \ \sum_{i=1}^{10} X_i^2 > \chi_{10}^2(0.5) \\ -1, \ інакше \end{cases}$$

Тренувальна вибірка складається з 2000 спостережень (близько 1000 спостережень з кожного класу) і 10000 тестових спостережень.

Використання дерева з $d = 2$ вузлами призводить до тестової помилки 45,8%.

Приклад.


Single Stump
244 Node Tree
Test Error
0.0
0.1
0.2
0.3
0.4
0.5
0
100
200
300
400
Boosting Iterations