hillel IT school

Lesson 07
Tree Based Models

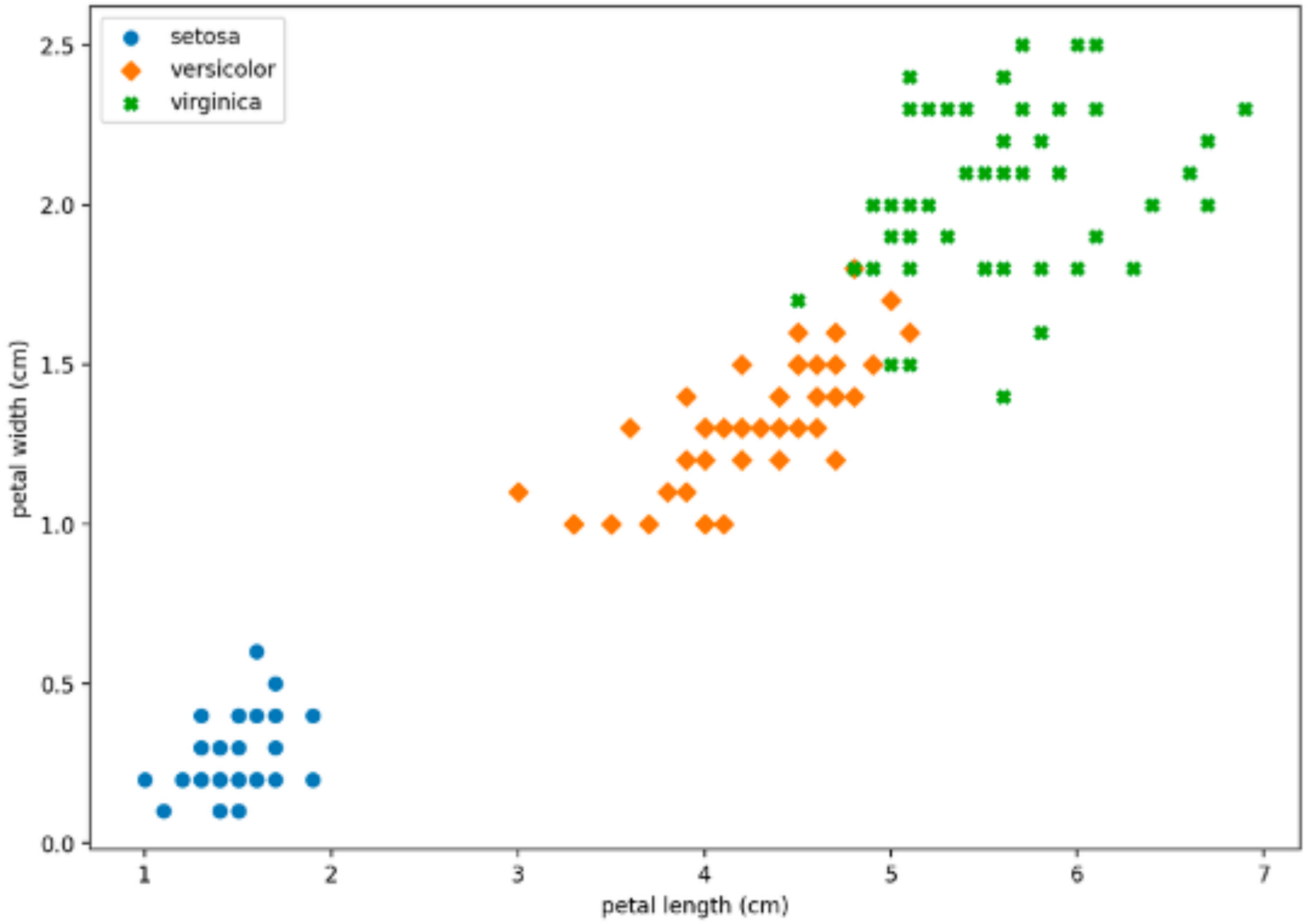
In the Previous Lesson

- Binary Classification
- Logistic Regression
- Multiclass Classification
- Logistic Regression for Multiclass Classification
- Metrics
- Regularization
- L1
- L2
- ElasticNet

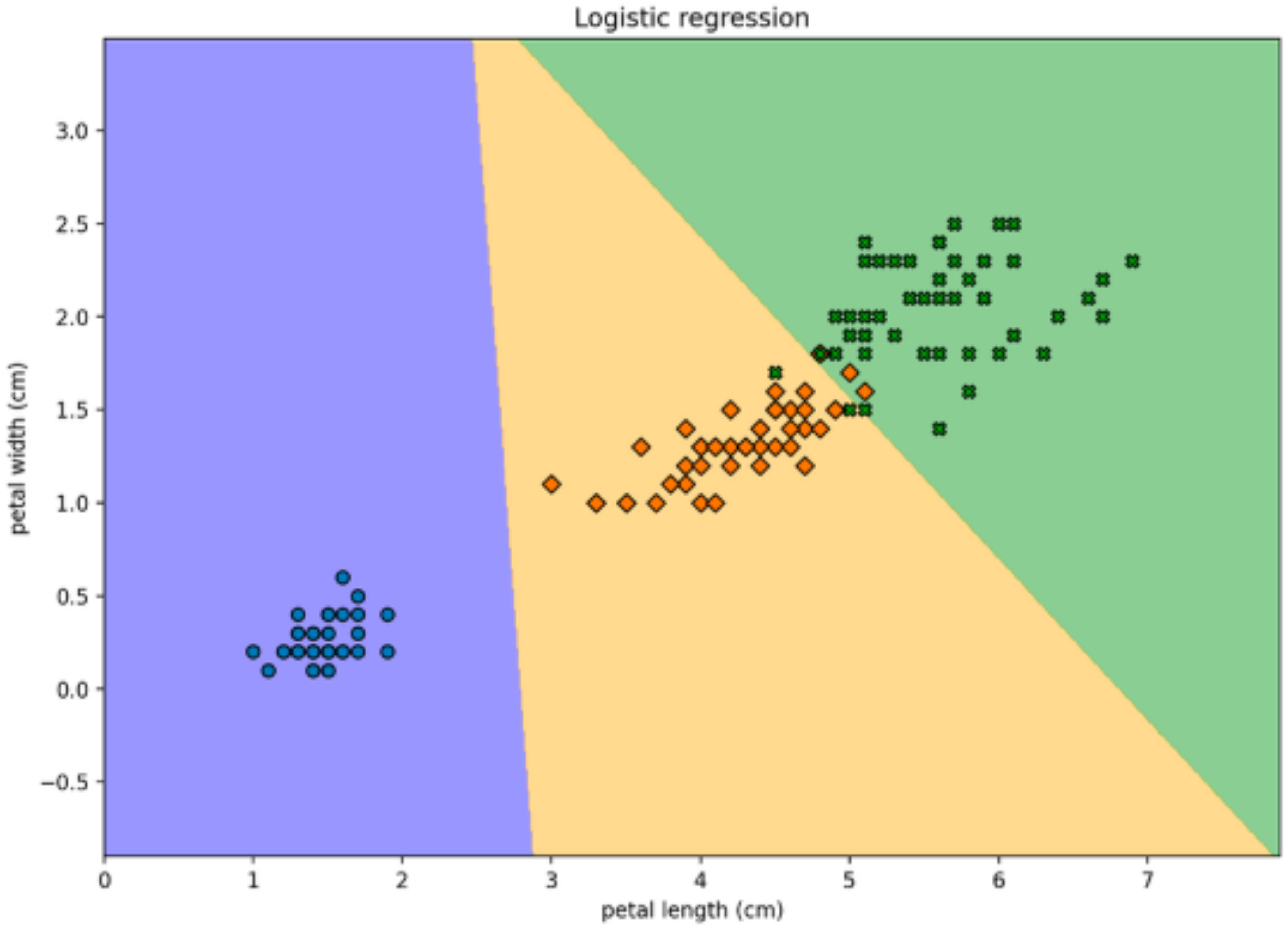
Introduction

- Regression Trees
- Decision Trees
- Bagging
- Random Forest
- Boostings (AdaBoost, XGBoost, LightGBM)

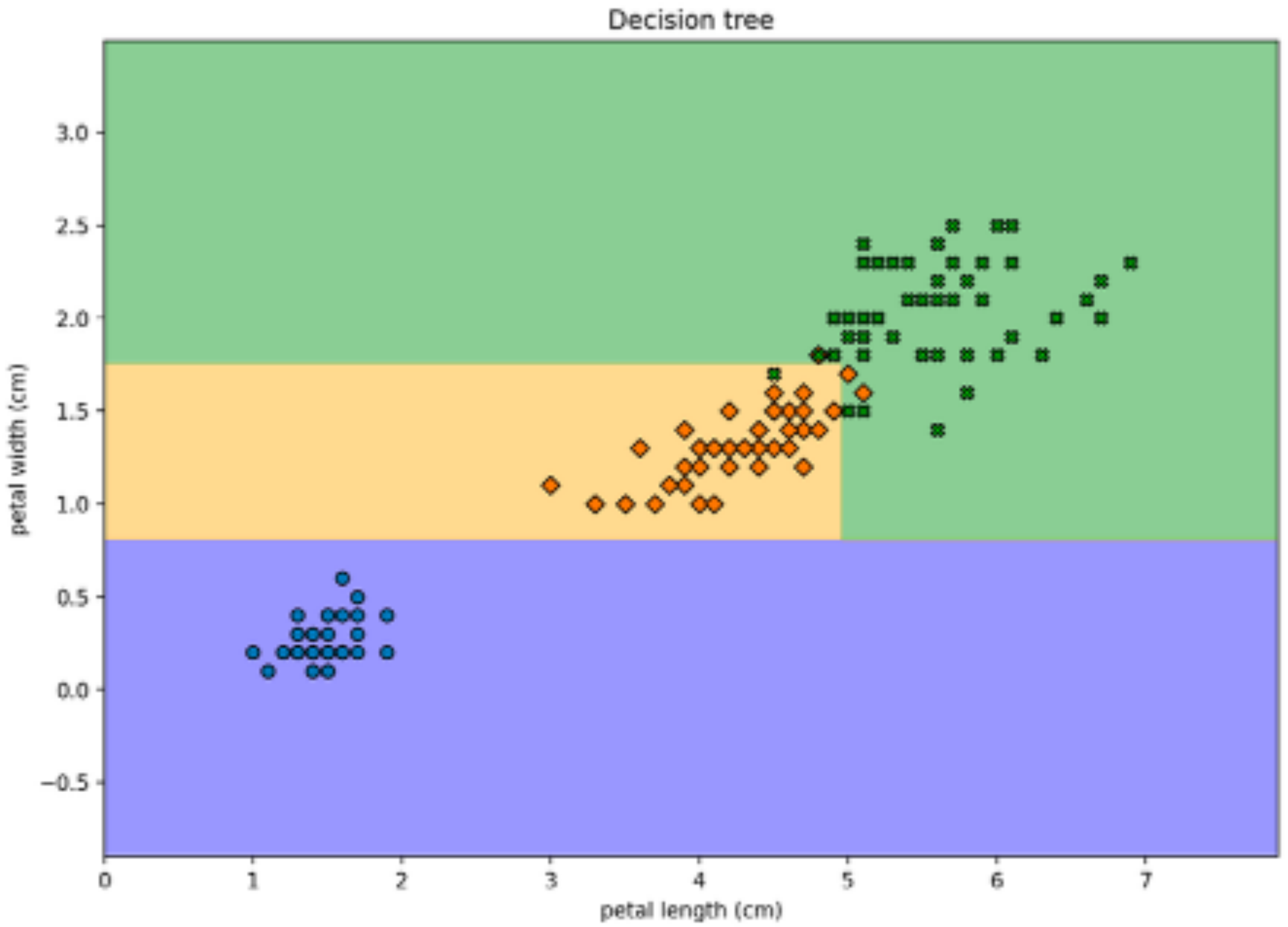
Iris classification

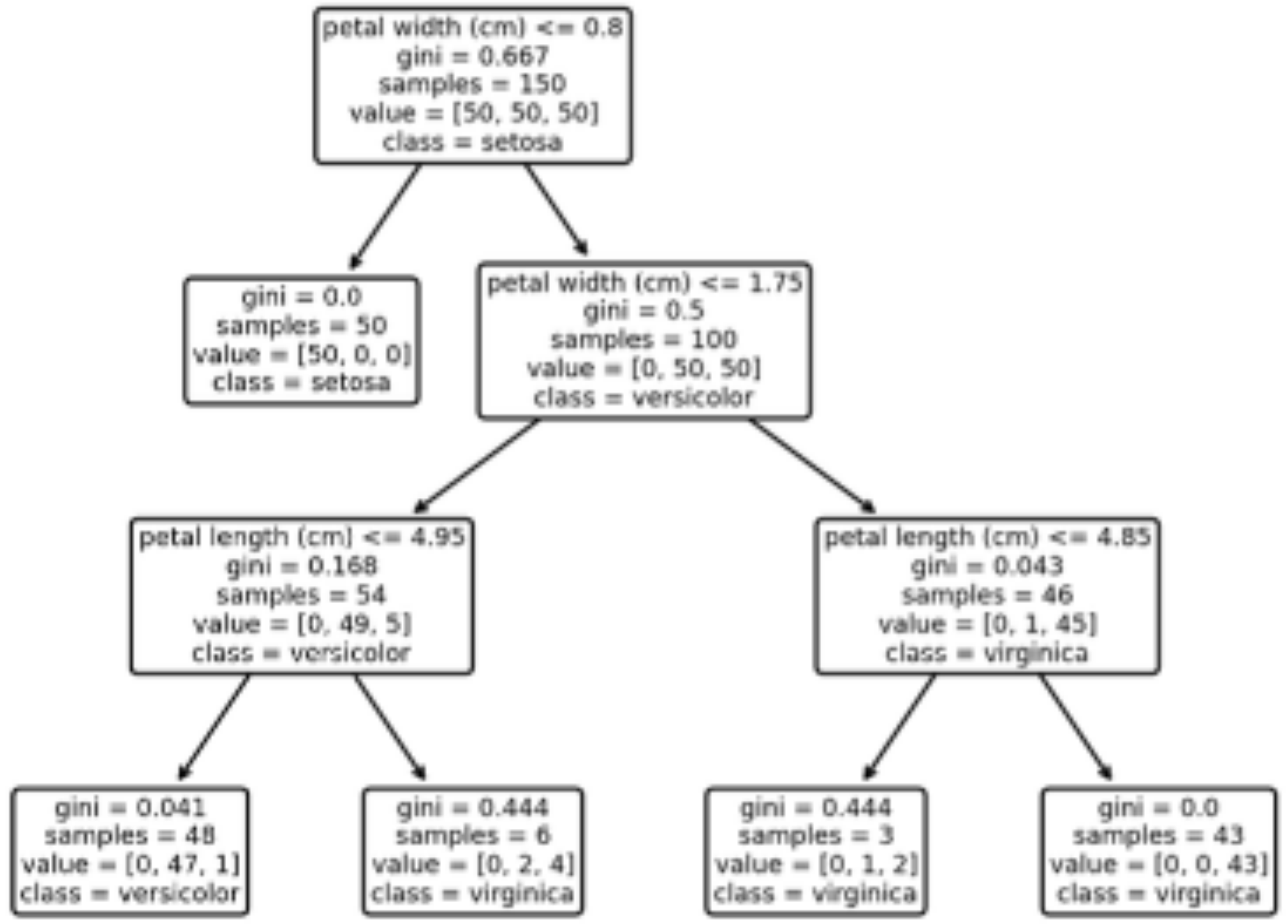


Iris classification

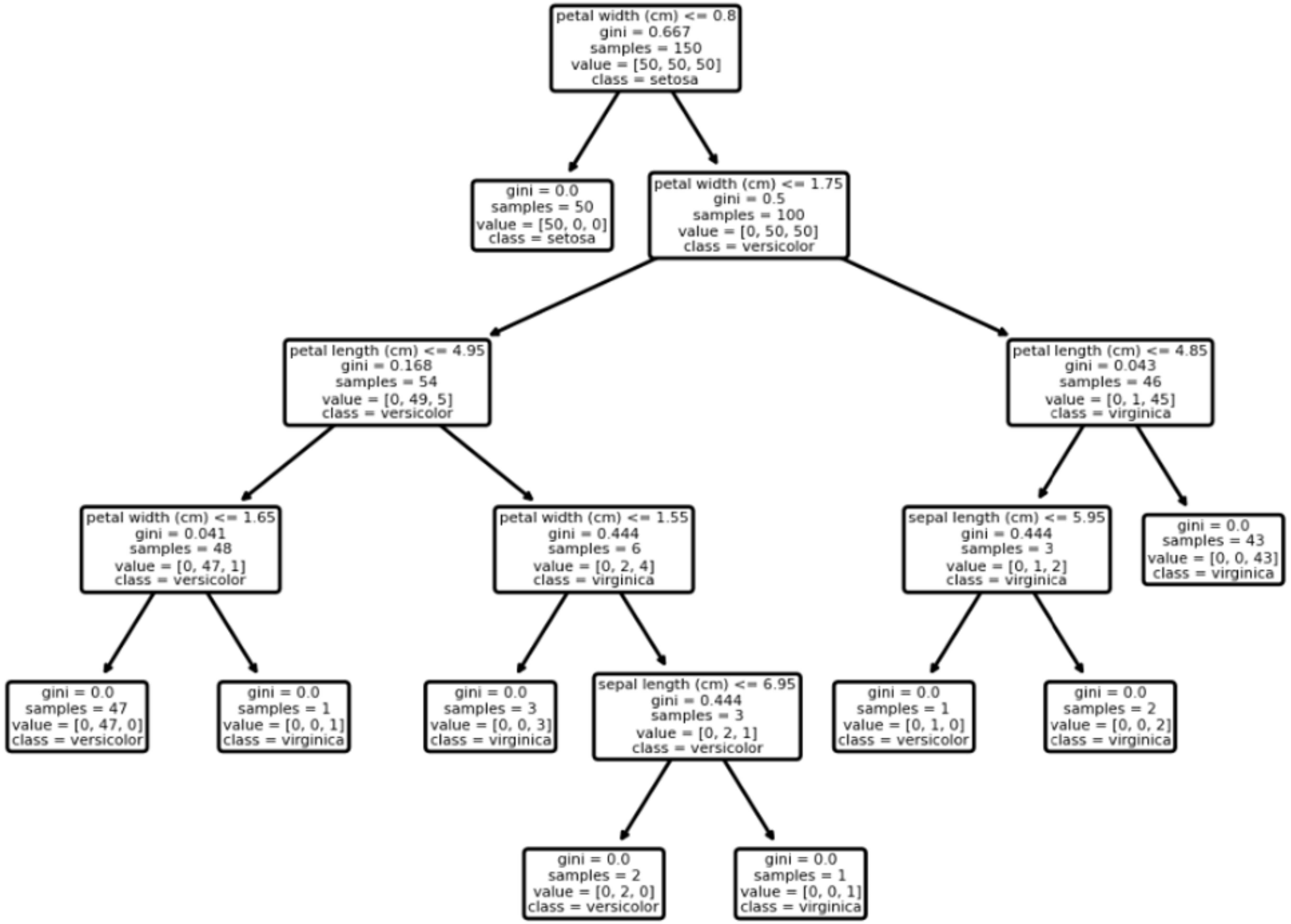


Iris classification





Tree Based Models

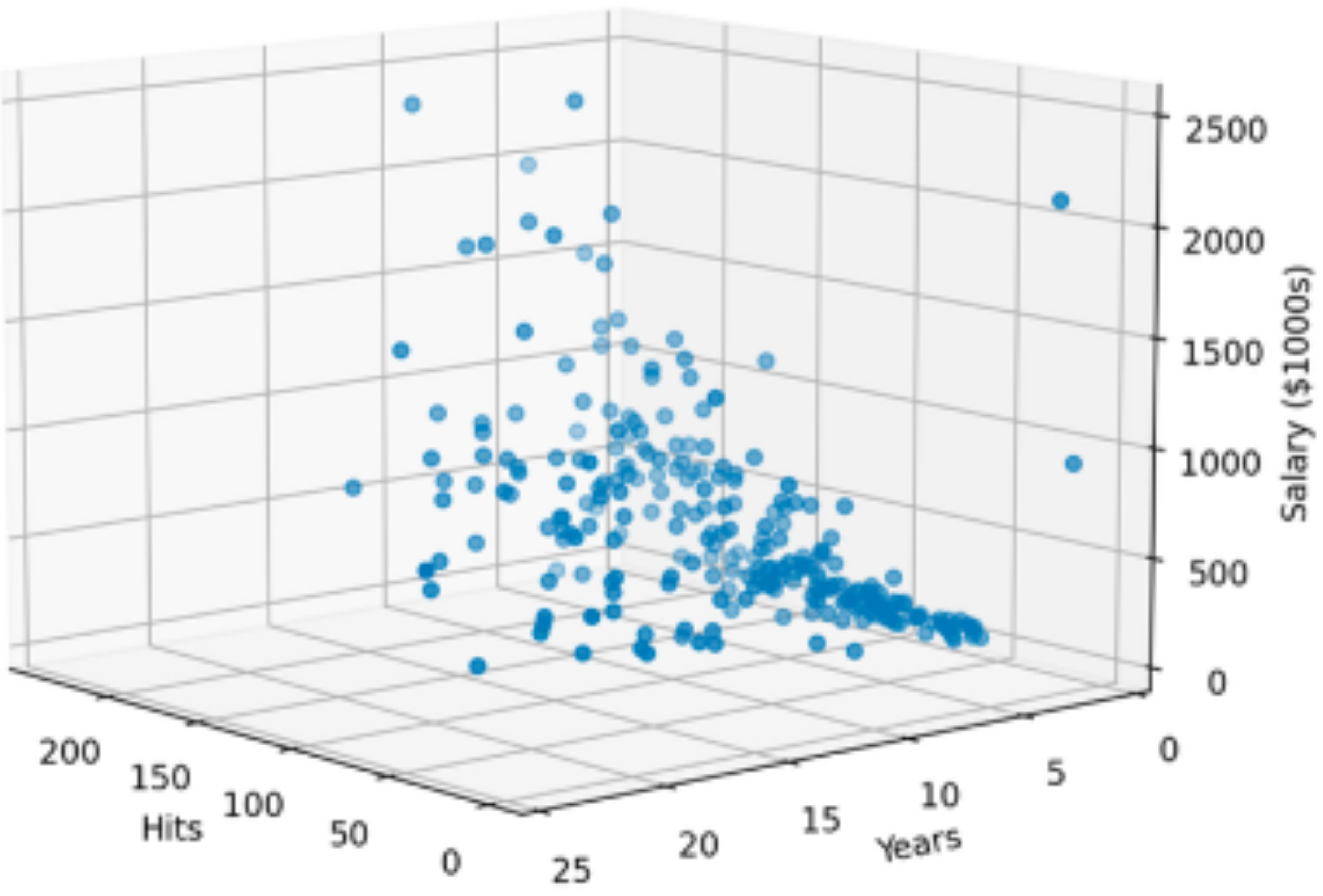


Regression Trees

- Давайте використаємо набір даних «Hitters Dataset» як приклад і побудуємо модель для прогнозування зарплати бейсболістів.

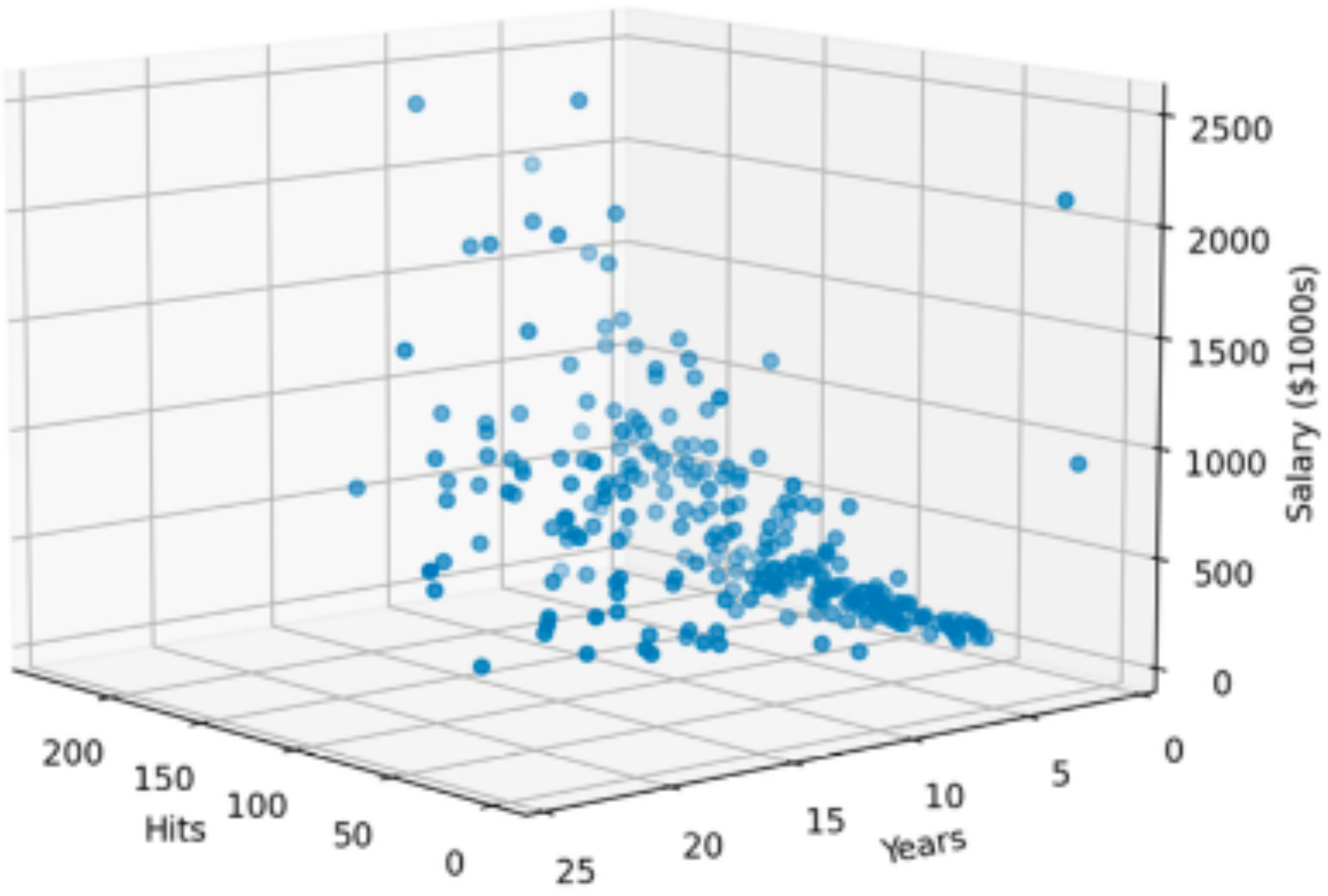
- Для спрощення ми будемо використовувати ознаки «Years» та «Hits», щоб передбачити цільову функцію «Salary».

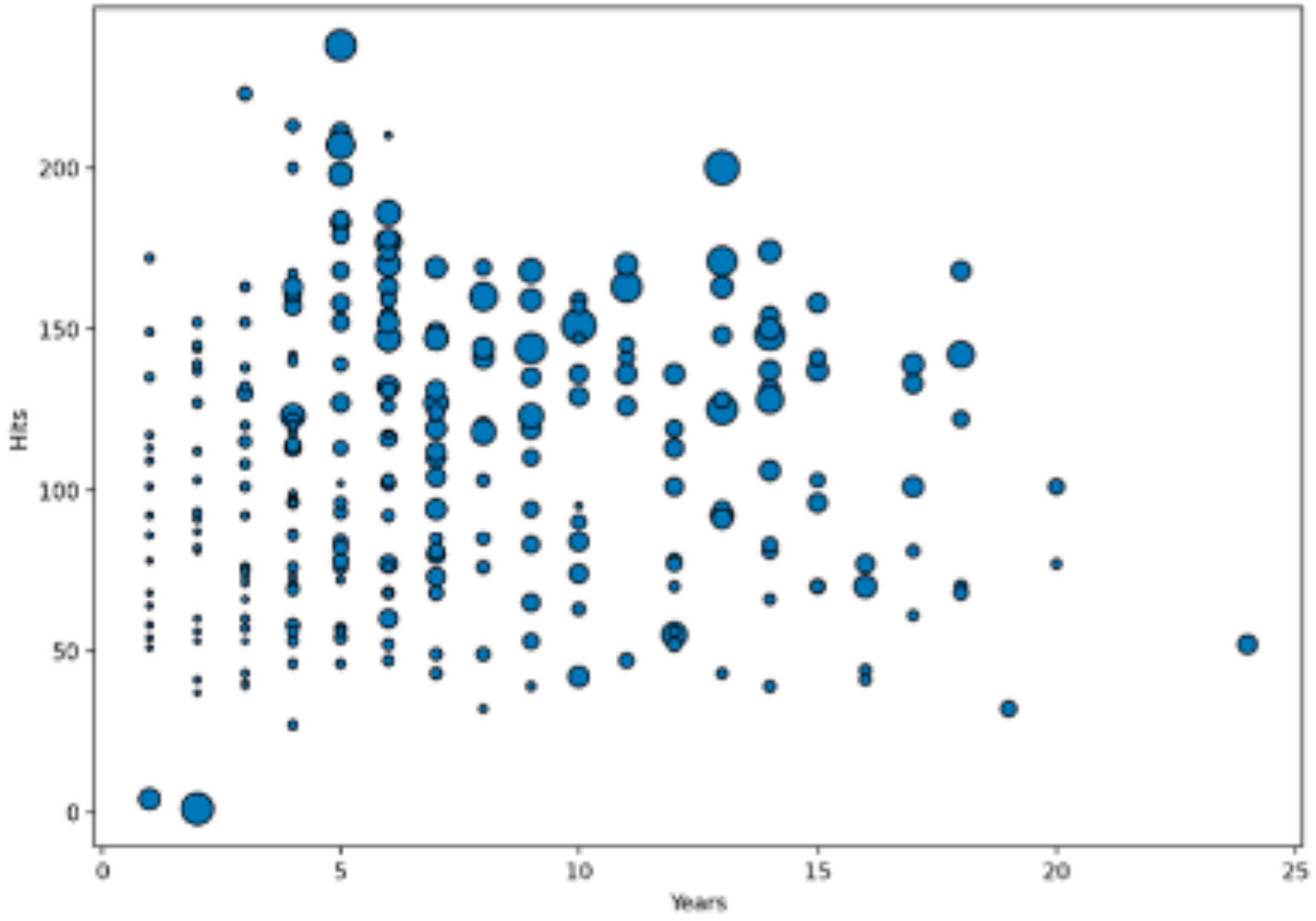
Regression Trees



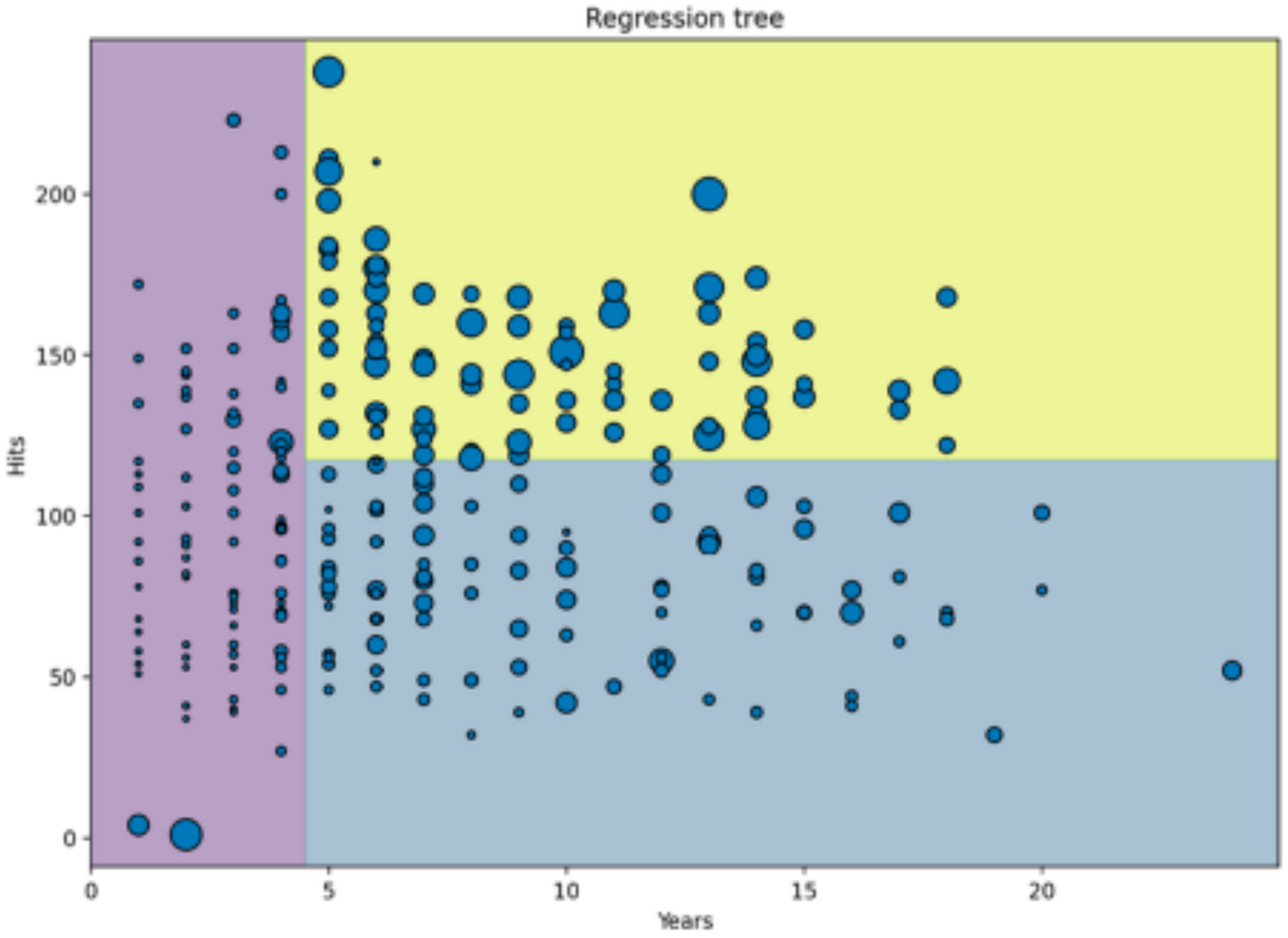
	Years	Hits	Salary
0	14	81	475.0
1	3	130	480.0
2	11	141	500.0
3	2	87	91.5
4	11	169	750.0
...	...	...	...
258	5	127	700.0
259	12	136	875.0
260	6	126	385.0
261	8	144	960.0
262	11	170	1000.0

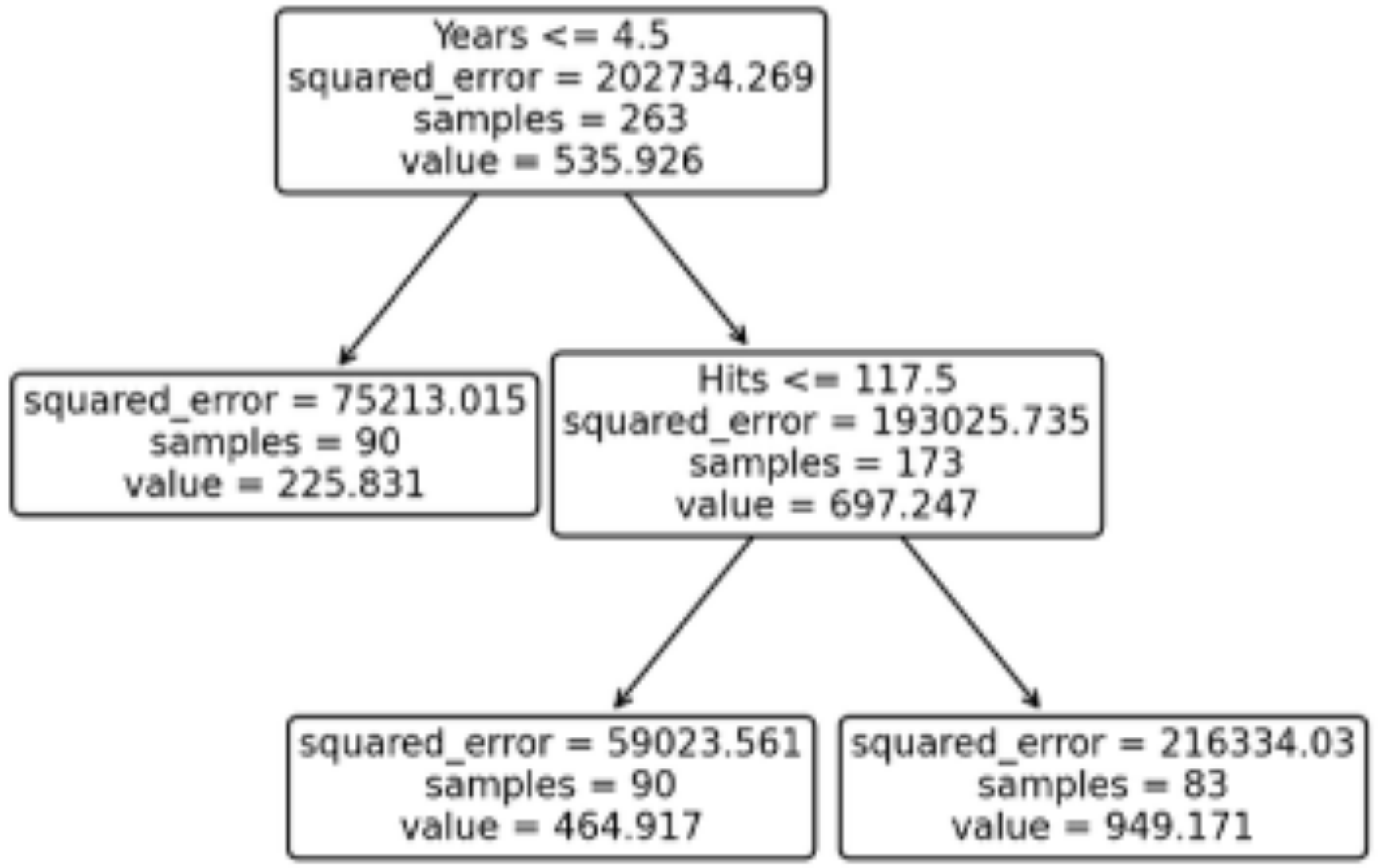
Regression Trees





Regression Trees





Model:

$$f(x;\Theta) = \sum_{m=1}^{M} c_m \mathbb{I}\{x \in R_m\}$$

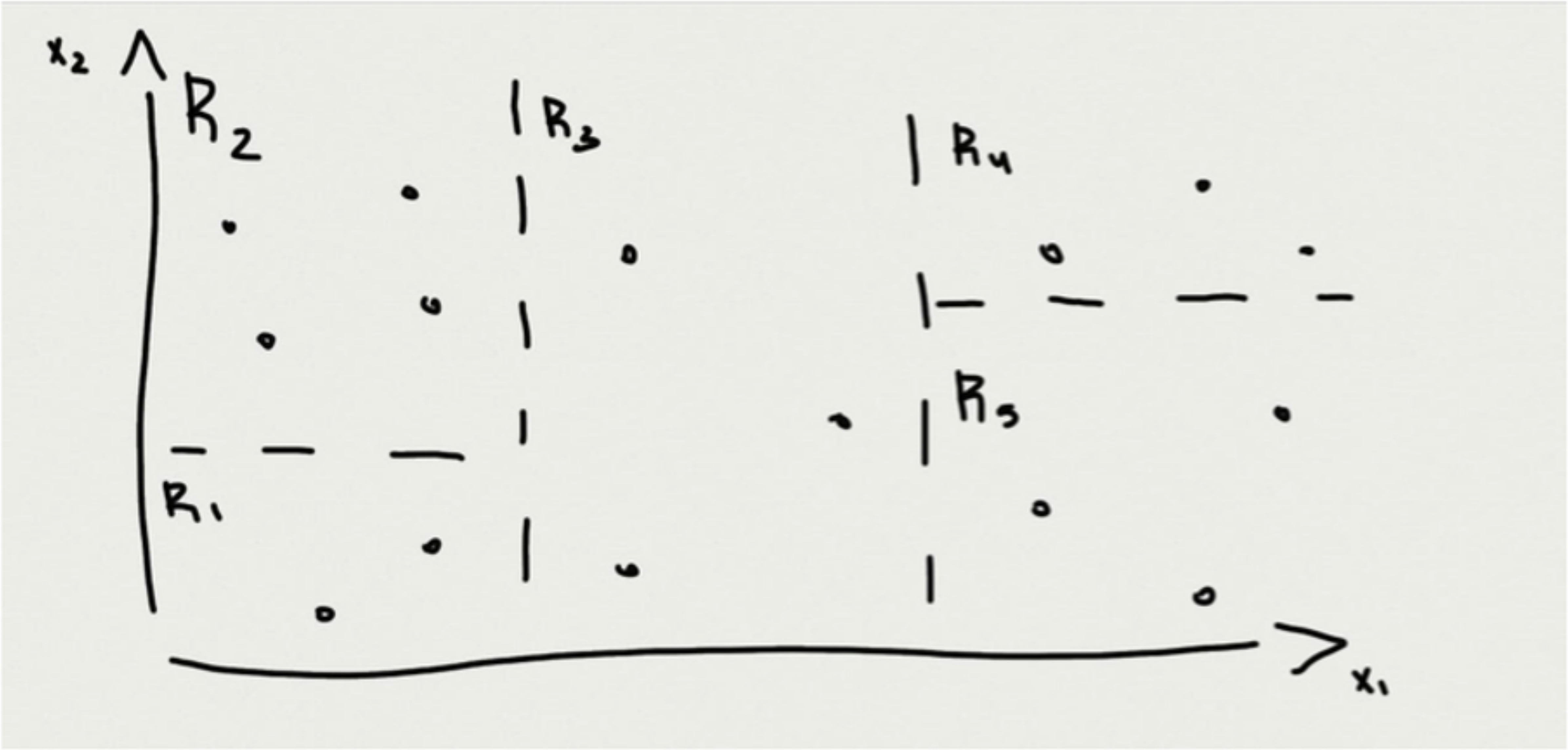
Loss:

$$Loss(\Theta) = \sum_{n}^{N} (y^{(n)} - f(x^{(n)}))^2 = \sum_{n=1}^{N} (y^{(n)} - \sum_{m=1}^{M} c_m \mathbb{I}\{x \in R_m\})^2$$

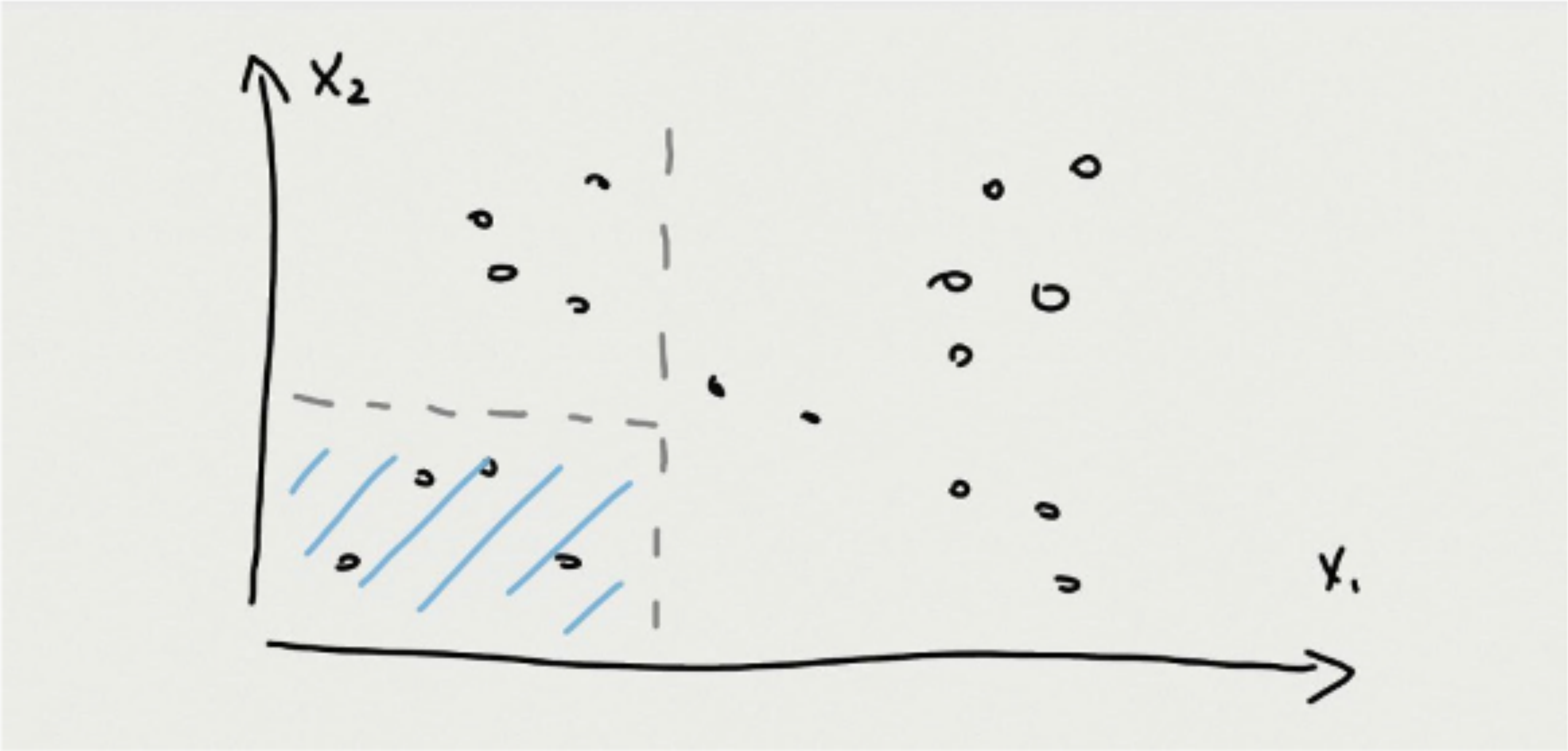
Parameters:

$$\Theta = \begin{cases} R_1, \ldots, R_M \\ c_1, \ldots, c_m \end{cases}$$

How do we learn Θ?



Regression Tree Model



Якщо ми знаємо регіони, тоді $c_m = \frac{1}{N_m} \sum_{i:x^{(i)} \in R_m} y^{(i)}$, де N_m містить кілька навчальних ознак в R_m.

Regression Tree Algorithm

Алгоритм створення дерева (зверху-вниз):

1. Почніть з верхньої частини дерева (все в одній області)

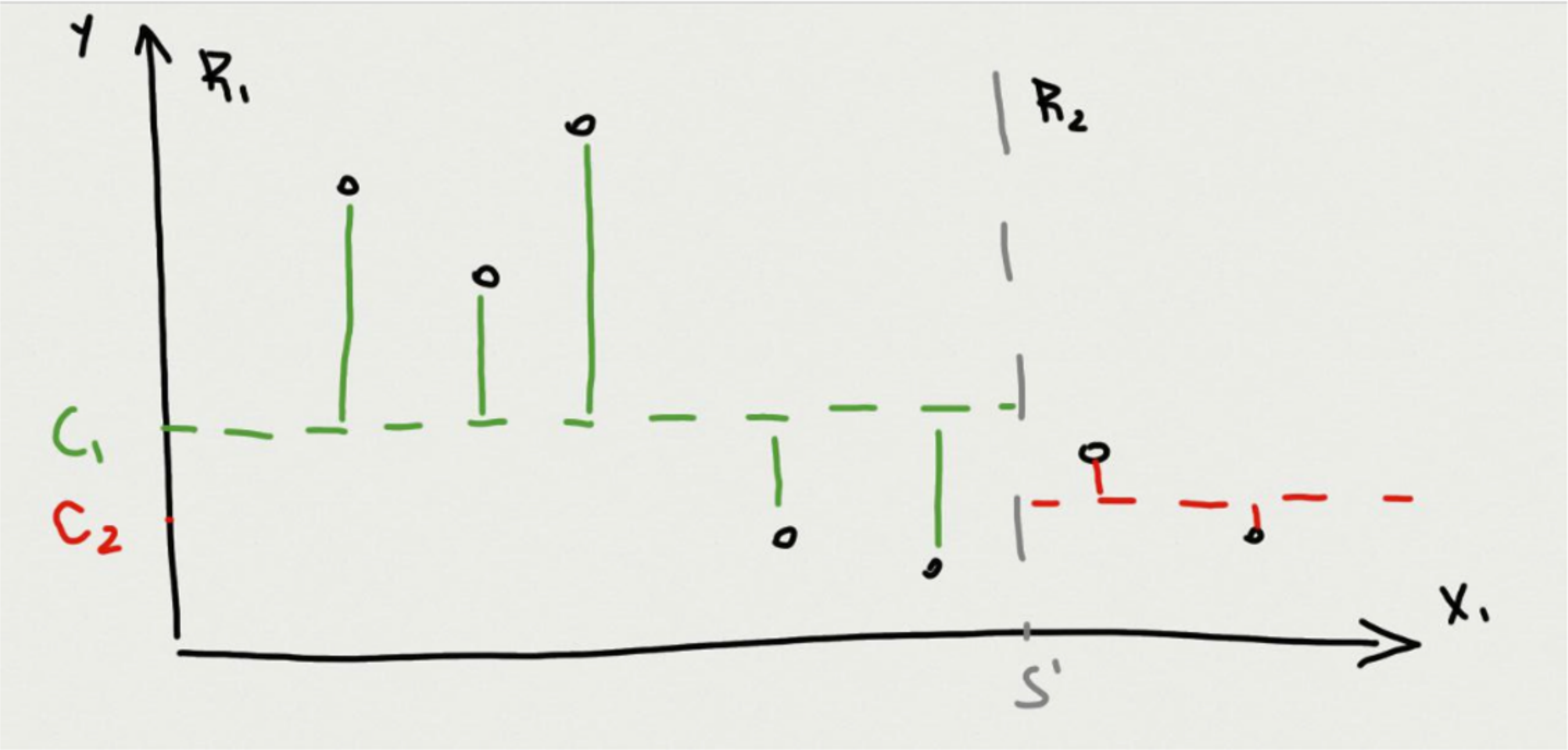
2. Для кожного листкового вузла (області):
   Для кожної функції x_j і точки розділення s:
      Обчисліть зменшення втрат, якщо ми розділимо їх.

3. Виберіть найкращу (j, s) комбінацію для розділення (жадібний підхід - greedy); Створіть нові дочірні вузли (області).

4. Повторюйте, починаючи з (2), доки не буде виконано умову зупинки.

Regression Tree Algorithm

По суті, на кроці 3 ми хочемо знайти $\min\limits_{j,s} \left\{ \sum_{i:x^{(i)} \in R_1} (y^{(i)} - c_1)^2 + \sum_{i:x^{(i)} \in R_2} (y^{(i)} - c_2)^2 \right\}$.



Regression Tree in Practice

- Може легко перенавчитися (overfitting)

- Як регулювати складність (regularization)?
  - Зупинитися, якщо покращення невелике
  - Обрізка дерев

Частка точок в області R_m від класу k:

$$\hat{p}_{m,k} = \frac{pointsinR_m withlabelk}{pointsinR_m} = \frac{1}{N_m} \sum_{i:x^{(i)} \in R_m} \{y^{(n)} = k\}$$

$$\hat{y}_m = \arg\max_k \hat{p}_{m,k}, y \in \{1,2,3\}$$

Decision Tree Algorithm

Алгоритм створення дерева (зверху-вниз):

1. Почніть з верхівки дерева (усе в одній області).

2. Для кожного листкового вузла (області):
   - Для кожної функції x_j і точок розділення S:
     - Розрахувати зменшення leaf impurity

3. Виберіть найкращу (j, S) комбінацію для розділення (greedy); Створення нових дочірніх вузлів (регіонів)

4. Повторюйте, починаючи з (2), доки не буде виконано умову зупинки.

Для вимірювання leaf impurity використовується індекс Gini: $$G_m = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}).$$

Bagging

Припустимо, що у нас є набір даних $X = \begin{bmatrix} - & (x^{(1)})^T & - \\ & \cdots & \\ - & (x^{(N)})^T & - \end{bmatrix}; \; y = \begin{bmatrix} y^{(1)} \\ \cdots \\ y^{(N)} \end{bmatrix}.$

Вибірка (із заміною) N вибірок із цього набору даних, щоб сформувати «новий» набір даних X_1, y_1 і натренувати для нього модель $f_1(x; \theta_1)$. Ці вибіркові набори даних називаються **bootstrap** зразками.

Повторіть вибірку набору даних і тренування моделі B разів, щоб отримати B різних прогнозів.

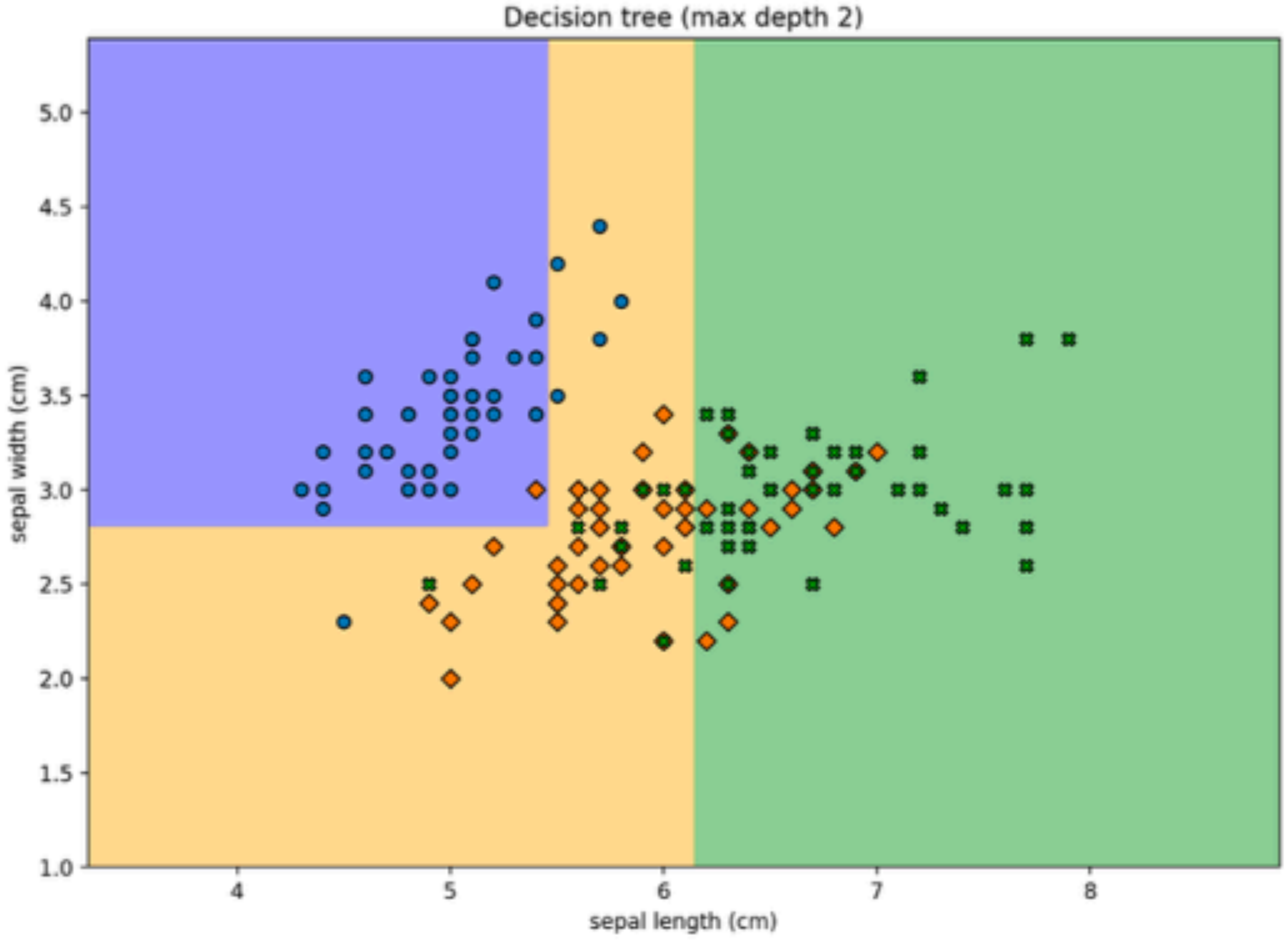
Щоб обчислити оцінку для задачі регресії: $f(x; \Theta) = \frac{1}{B} \sum_{b=1}^{B} f_b(x; \Theta_b).$

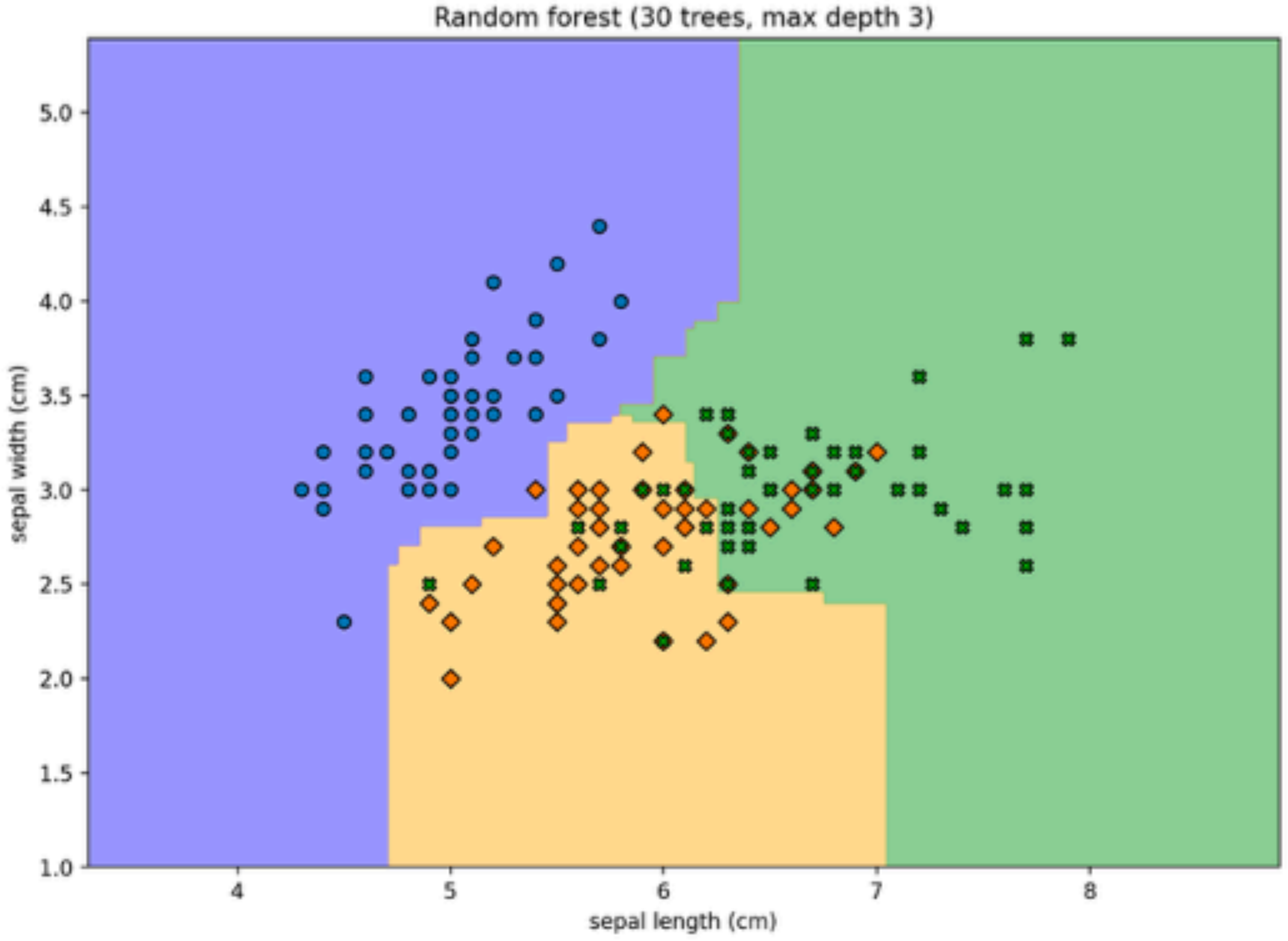
Для задач класифікації вам потрібно використовувати більшість голосів (majority vote) або середньозважену ймовірність класу.

Random Forest

- Спеціально використовується з деревами рішень і регресії.
- Використовує bagging - тренує кожне дерево на різних bootstrap зразках.
- Кожен раз, коли ми розглядаємо розбиття (split), розглядаємо лише підмножину вхідних ознак $M < D$.
- Зазвичай $M = \sqrt{D}$.
- Щоб отримати прогноз, ми обчислюємо середнє значення, як у bagging
  $(f(x;\Theta) = \frac{1}{B}\sum_{b=1}^{B} f_b(x;\Theta_b)$ для регресії або majority voting для класифікацію.

Decision Tree vs Random Forest





Boosting

1. Ініціалізуйте $r^{(n)} = y^{(n)}$ для всіх N тренувальних елементів та встановіть $f(x, \theta) = 0$.

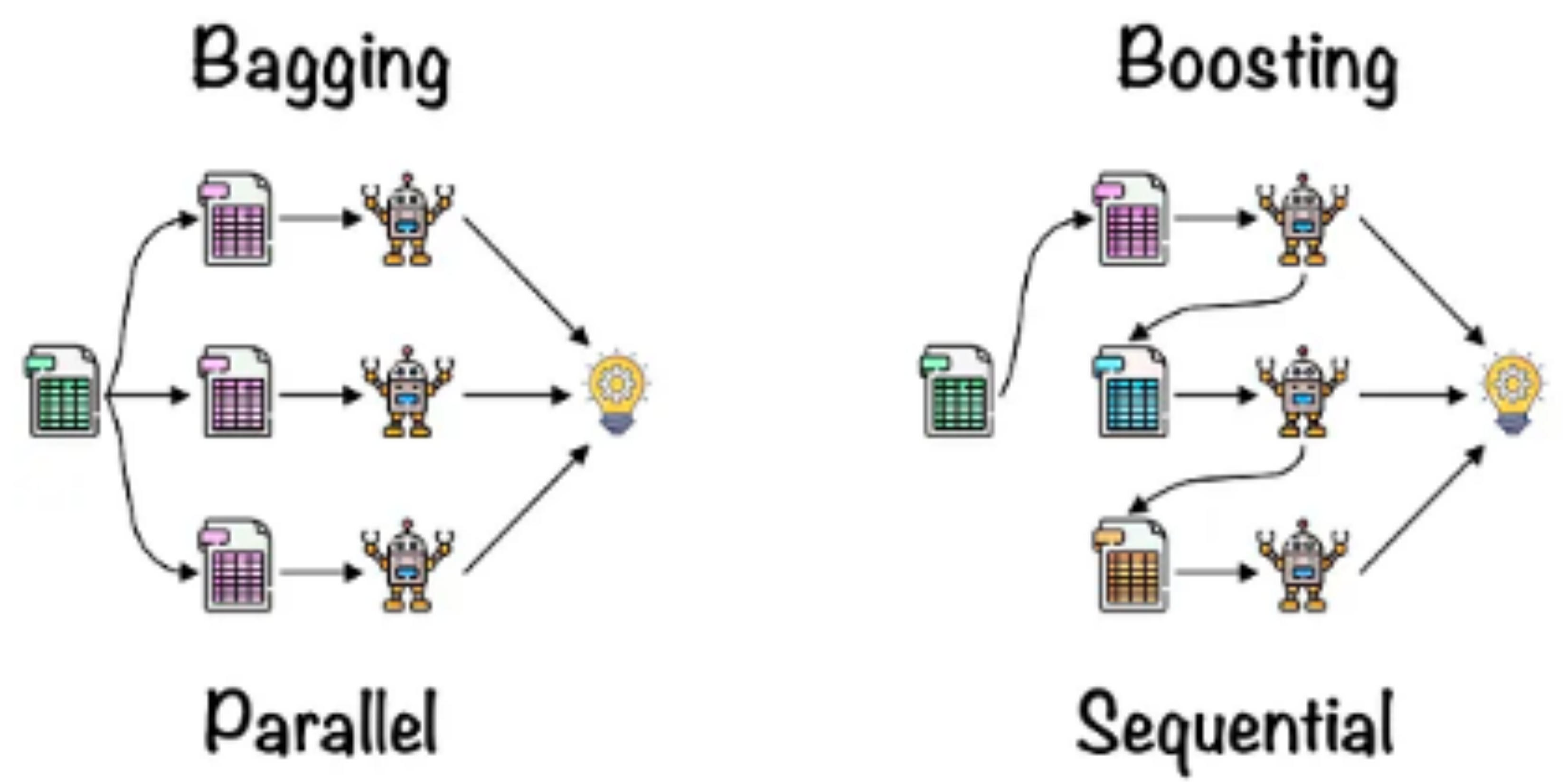
2. Для ітерації $b = 1$ до B:

   a) Підібрати модель $f_b(x; \theta_b)$ до входів X, виходів r.
   b) Оновіть модель, додавши скорочену версію (shrunken version):

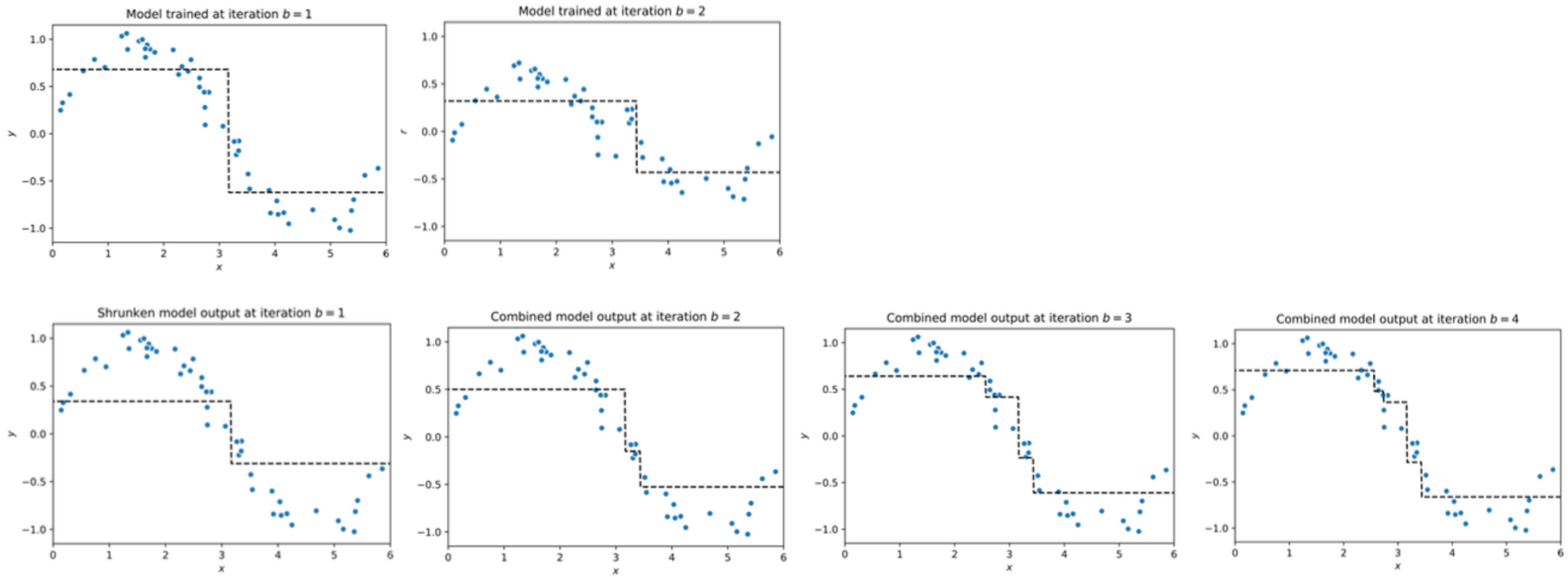
$$f(x; \theta) \leftarrow f(x; \theta) + \lambda f_b(x; \theta_b)$$

   e) Оновіть залишки (residuals): $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(x^{(n)}; \theta_b)$

3. Фінальна модель: $f(x; \theta) = \sum_{b=1}^{B} \lambda f_b(x; \theta_b)$.

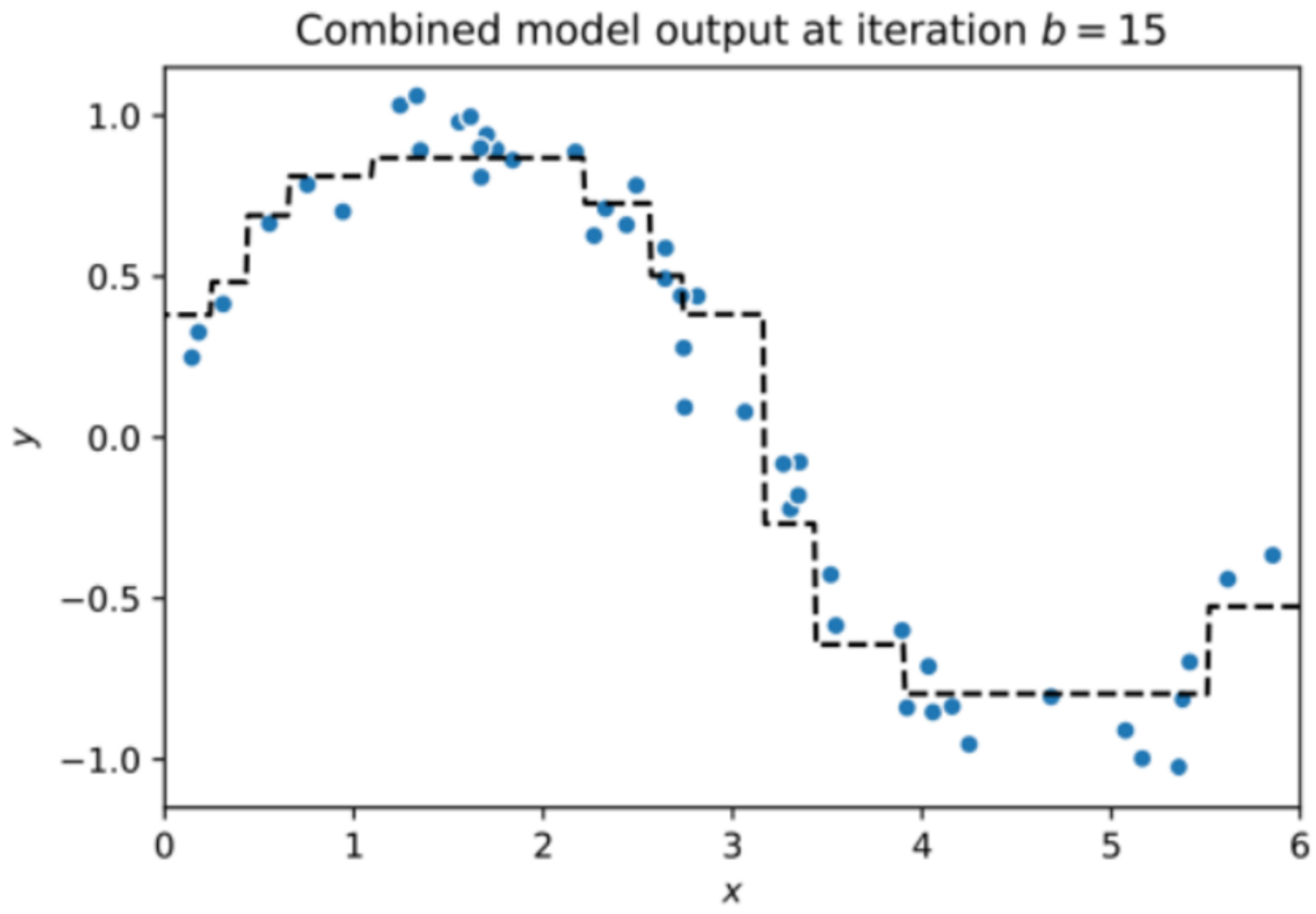


Boosting



Model trained at iteration $b = 1$
Model trained at iteration $b = 2$
Shrunken model output at iteration $b = 1$
Combined model output at iteration $b = 2$
Combined model output at iteration $b = 3$
Combined model output at iteration $b = 4$

Boosting



Combined model output at iteration $b = 15$
1.0
0.5
0.0
−0.5
−1.0
y
0
1
2
3
4
5
6
x

Boosting for Classification: AdaBoost

AdaBoost

1. Initialise training item weights $w^{(n)} = 1$ for all N training items.

2. For iteration $b = 1$ to B:

   (a) Fit model $f_b(x; \theta_b)$ so that it minimises classification error weighted by $w^{(n)}$.

   $$e = \frac{\sum_{n=1}^{N} w^{(n)} \mathbb{1}\{y^{(n)} \neq f_b(x^{(n)}; \theta_b)\}}{\sum_{n=1}^{N} w^{(n)}}$$

   (b) Set model weight using error e:

   $$\lambda_b = \frac{1}{2} \log\left(\frac{1-e}{e}\right)$$

   (c) Update training item weights:

   $$w^{(n)} \leftarrow \begin{cases} w^{(n)} e^{-\lambda_b} = w^{(n)} \sqrt{\frac{e}{1-e}} & \text{if } f_b(x^{(n)}; \theta_b) \text{ correct} \\ w^{(n)} e^{\lambda_b} = w^{(n)} \sqrt{\frac{1-e}{e}} & \text{if } f_b(x^{(n)}; \theta_b) \text{ incorrect} \end{cases}$$

3. Final model: $f(x; \theta) = \text{sign}\left[\sum_{b=1}^{B} \lambda_b f_b(x; \theta_b)\right]$.

Boosting for Classification: XGBoost

XGBoost

Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$, a differentiable loss function $L(y, F(x))$, a number of weak learners M and a learning rate α.

Algorithm:

1. Initialize model with a constant value: $\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta)$.

2. For $m = 1$ to M:

   (a) Compute the "gradients" and "hessians":

   $$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{m-1}(x)}, \quad \hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}_{m-1}(x)}.$$

   (b) Fit a base learner (or weak learner, e.g. tree) using the training set $\left\{x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right\}_{i=1}^{N}$ by solving the optimisation problem below:

   $$\hat{\phi}_m = \arg\min_{\phi \in \Phi} \sum_{i=1}^{N} \frac{1}{2}\hat{h}_m(x_i)\left[\phi(x_i) - \frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right]^2 \text{g}$$

   $$\hat{f}_m(x) = \alpha\hat{\phi}_m(x)$$

   (c) Update the model:

   $$\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x).$$

3. Output $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$.

dmlc XGBoost

Boosting for Classification: LightGBM

LightGBM

Input: training set $\left\{(x_i, y_i)\right\}_{i=1}^{N}$, loss function $L(y, F(x))$, iterations M, sampling ratio of large gradient data a, sampling ratio of small gradient data b.

1. Combine features that are mutually exclusive (i.e. features never take nonzero values simultaneously) of x_i, $i = \{1,..,N\}$ by the exclusive feature building (EFB) method.

2. Set $\theta_0(x) = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$

3. For $m = 1$ to M:

    (a) Calculate gradient absolute values $r_i = \left| \frac{\partial L(y_i, \theta(x_i))}{\partial \theta(x_i)} \right|_{\theta(x)=\theta_{m-1}(x)}, i = \{1,..,N\}$.

    (b) Resample dataset using gradient-based one-side sampling (GOSS) process:

$$
\begin{aligned}
&\text{topN} = a \times \text{len}(D)\\
&\text{randN} = b \times \text{len}(D)\\
&\text{sorted} = \text{GetSortedIndices}(\text{abs}(r))\\
&A = \text{sorted}[1 : \text{topN}]\\
&B = \text{RandomPick}(\text{sorted}[\text{topN} : \text{len}(D)], \text{randN})\\
&D' = A + B
\end{aligned}
$$

    (c) Calculate information gains $V_j(d) = \frac{1}{n}\left(\frac{(\sum_{x_i \in A_l} r_i + \frac{1-a}{b}\sum_{x_i \in B_l} r_i)^2}{n_r^j(d)} + \frac{(\sum_{x_i \in A_r} r_i + \frac{1-a}{b}\sum_{x_i \in B_r} r_i)^2}{n_r^j(d)} \right)$.

    (d) Create a new decision tree $\theta_m(X)'$ on set D'.

    (e) Update $\theta_m(X) = \theta_{m-1}(X) + \theta_m(X)'$

4. Return $\theta'_M(X) = \theta_M(X)$.

LightGBM

Documentation

- sklearn.tree.DecisionTreeRegressor
- sklearn.tree.DecisionTreeClassifier
- sklearn.tree.ExtraTreeRegressor
- sklearn.tree.ExtraTreeClassifier
- sklearn.ensemble.BaggingRegressor
- sklearn.ensemble.BaggingClassifier
- sklearn.ensemble.RandomForestRegressor
- sklearn.ensemble.RandomForestClassifier
- sklearn.ensemble.AdaBoostRegressor
- sklearn.ensemble.AdaBoostClassifier
- xgboost.XGBRegressor
- xgboost.XGBClassifier
- lightgbm.LGBMRegressor
- lightgbm.LGBMClassifier

Thanks for your attention