Дерева прийняття рішень

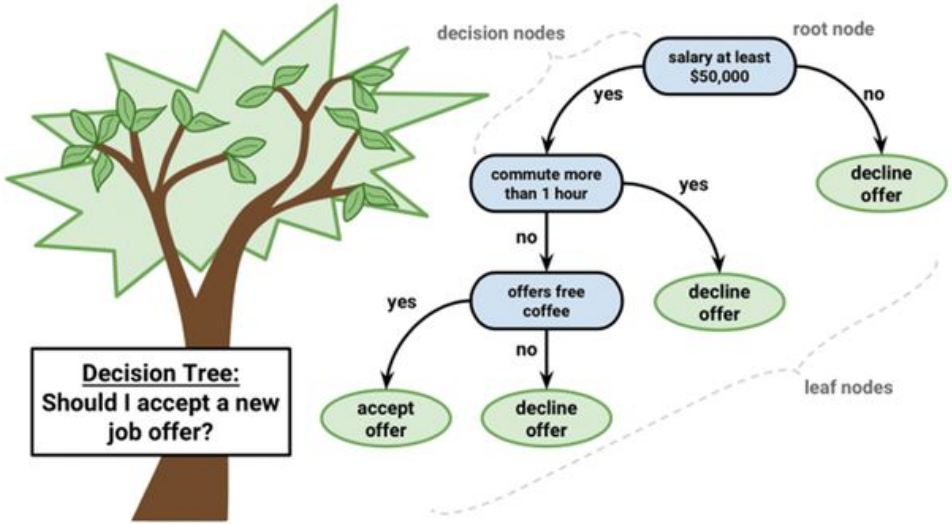
План теми

+ Суть моделі, переваги та недоліки
+ Принцип роботи моделі
+ Дерева рішень у sklearn для задач класифікації та регресії
+ Параметри дерева прийняття рішень
+ Візуалізація дерева рішень
+ Виявлення важливості ознак (Feature importance)

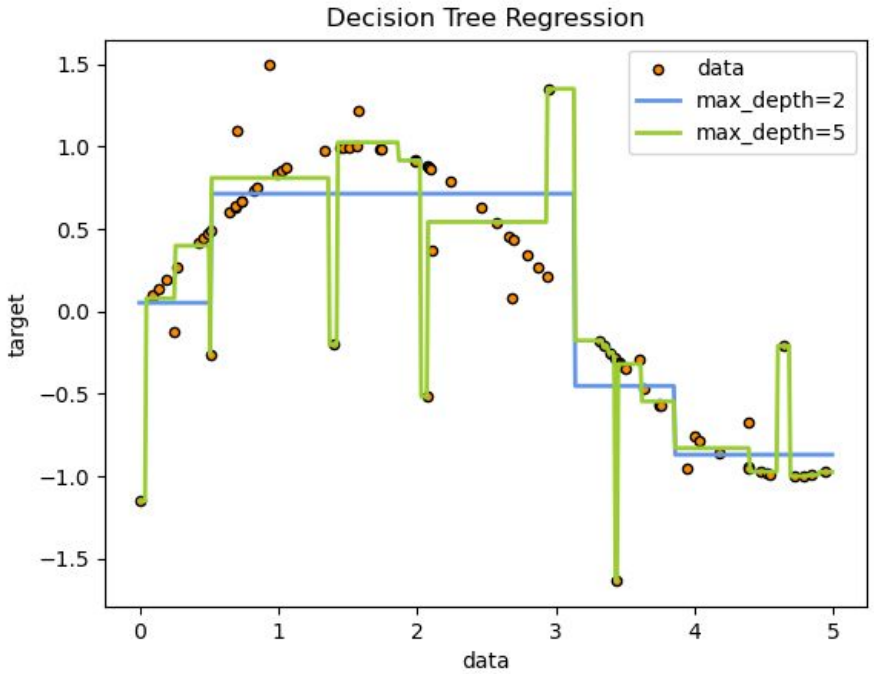
Дерева прийняття рішень

Дерево рішень - це *непараметричний* алгоритм навчання з учителем на основі дерева, що використовується для вирішення завдань регресії та класифікації.

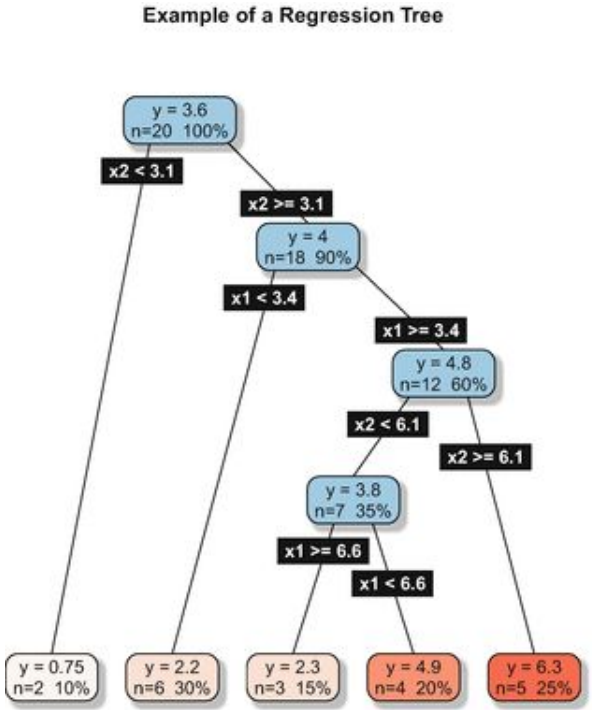
Мета полягає в створенні моделі, яка передбачає значення цільової змінної, вивчаючи прості правила прийняття рішень, видобуті з характеристик даних. Дерево можна розглядати як кусково-константну апроксимацію.

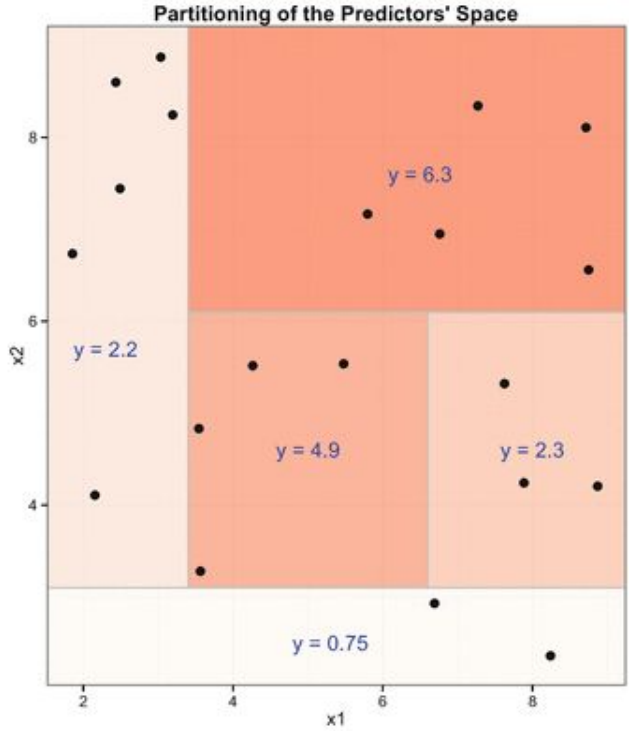


Приклад моделі дерева вибору рішень



Приклад регресійного дерева





Переваги дерев прийняття рішень 1/2

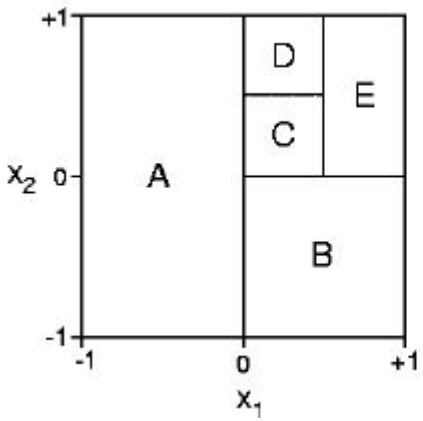
+ Прості у розумінні, візуалізації, інтерпретації прогнозу
+ Використовують модель “білого ящика”. У такому випадку рішення моделі легко пояснити за допомогою булевої логіки. Навпаки, в моделі “чорного ящика” (наприклад, у штучній нейронній мережі) інтерпретувати результати може бути складніше.
+ Можуть застосовуватися для аналізу даних (EDA фаза)
+ Здатні розв'язувати задачі з кількома виходами (multi-output problem)
+ Висока швидкість навчання і передбачення (у порівнянні з глибокими нейронними мережами)

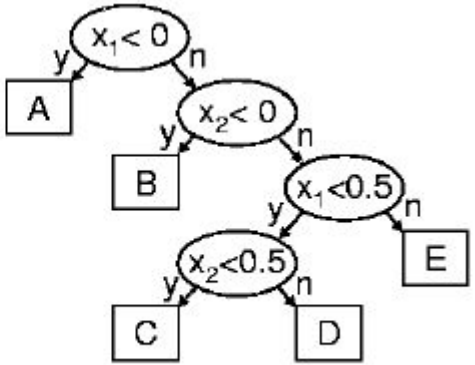
Переваги дерев прийняття рішень 2/2

+ Не потребують попередньої обробки даних (нормалізації, створення фіктивних змінних, як при роботі з регресіями; може обробляти NaNs з коробки, але реалізація sklearn цього не вміє)
+ Непараметричний метод: не має жодних припущень про розподіл простору і структуру (формулу) класифікатора
+ Ефективно справляється з колінеарністю незалежних змінних.
+ Стійкі до викидів

Стійкість до викидів

Оскільки дерева рішень розділяють елементи лініями, не має значення, наскільки далеко точка від ліній.





Недоліки

- Ймовірність **перенавчання** моделі: можуть створюватися занадто складні моделі, які погано узагальнюють дані
- Дерева рішень можуть бути **нестабільними**, оскільки невеликі зміни в даних можуть призвести до створення зовсім іншого дерева. Ця проблема розв'язується використанням дерев рішень у **ансамблі**.
- Прогнози дерев рішень не є ані **гладкими**, ані **неперервними**, а є ланцюжково-сталими наближеннями. Тому вони не вміють **екстраполювати**.
- Дерева вибору рішень створюють **упереджені** (biased) дерева, якщо деякі класи домінують. Тому рекомендується збалансувати набір незбалансованих щодо класів даних перед підгонкою під дерево рішень.

Лінійна регресія проти дерев прийняття рішень

Дерева прийняття рішень підтримують побудову нелінійних функцій гіпотез, тоді як лінійна регресія підтримує лише лінійні.

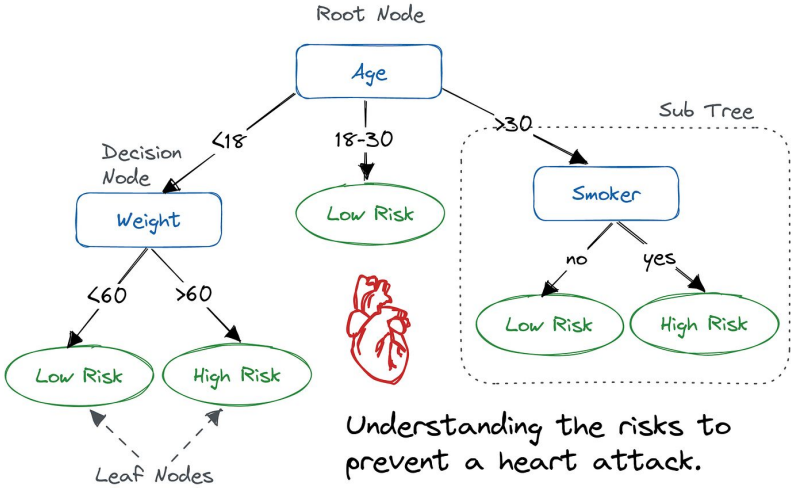
Термінологія

Назва	Опис
Вузол (node)	Внутрішній вузол дерева, вузол перевірки
Кореневий вузол (root)	Початковий вузол дерева рішень
Лист (leaf)	Кінцевий вузол дерева, вузол рішення, термінальний вузол
Правило прийняття рішення (decision rule)	Умова в вузлі, перевірка типу “якщо, то”

Структура дерева рішень 1/3

Саме дерево рішень — це метод подання правил прийняття рішень в ієрархічній структурі, що складається з елементів двох типів — **вузлів** (node) та **листків** (leaf). У вузлах знаходяться правила прийняття рішень, та проводиться перевірка відповідності прикладів цьому правилу за якоюсь ознакою навчального набору.

У найпростіших випадках, в результаті перевірки, множина прикладів, що потрапили в вузол, розбивається на дві підмножини, в одну з яких потрапляють приклади, що задовольняють умову правила, а в іншу — ті, що не задовольняють.

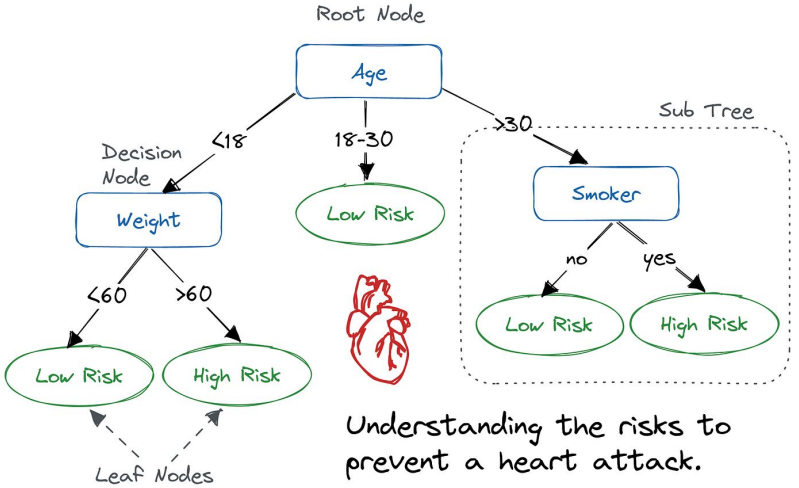


Структура дерева рішень 2/3

Потім до кожного підмножини знову застосовується правило процедура **рекурсивно** повторюється, поки не буде досягнуто якась **умова зупинки** алгоритму.

В результаті в останньому вузлі перевірка і розбиття не здійснюється і він оголошується **листом**. Лист визначає рішення (клас або число) для кожного прикладу, що в нього потрапив.

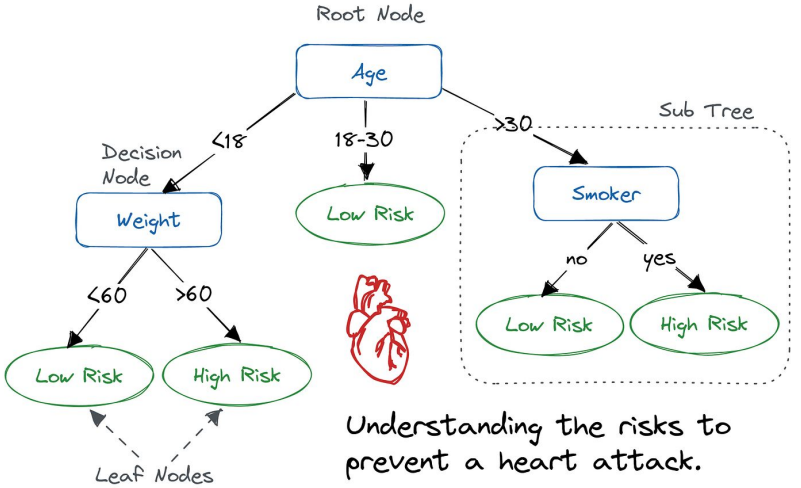
Для дерева класифікації — це клас, асоційований з вузлом, а для дерева регресії — відповідний листу інтервал цільової змінної (число — це середнє інтервалу).



Структура дерева рішень 3/3

Отже, на відміну від вузла, в листі міститься не правило, а підмножина об'єктів, що задовольняють усі правила гілки, яка завершується цим листом.

Очевидно, що щоб потрапити в лист, приклад повинен відповідати всім правилам, що лежать на шляху до цього листа. Оскільки шлях у дереві до кожного листа єдиний, то й кожний приклад може потрапити тільки в один лист, що забезпечує однозначність рішення.



Як ~~виростити~~ натренувати дерево?

Жадібний алгоритм (Greedy algorithm)

Алгоритми побудови дерев рішень належать до категорії так званих **жадібних**.

Жадібними називаються алгоритми, які допускають, що локально-оптимальні рішення на кожному кроці (розділення в вузлах), призводять до оптимального кінцевого рішення (ми не глибоко не дивимось).

У випадку дерев рішень це означає: якщо ознака була вибрана, і за нею було проведено розділення на підмножини, то алгоритм не може повернутися назад і вибрати іншу ознаку, яка дала б краще кінцеве розділення.

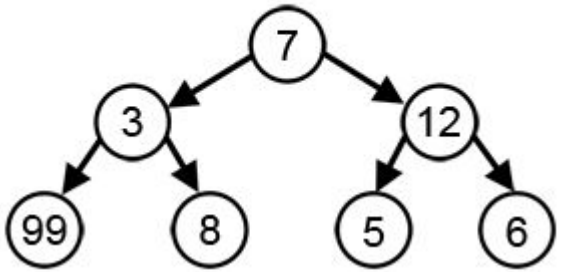
Тому на етапі побудови не можна сказати, чи забезпечить вибраний атрибут в кінцевому результаті **оптимальне** розділення.

Втім це **значно** прискорює весь процес.

Відео-пояснення на прикладі.

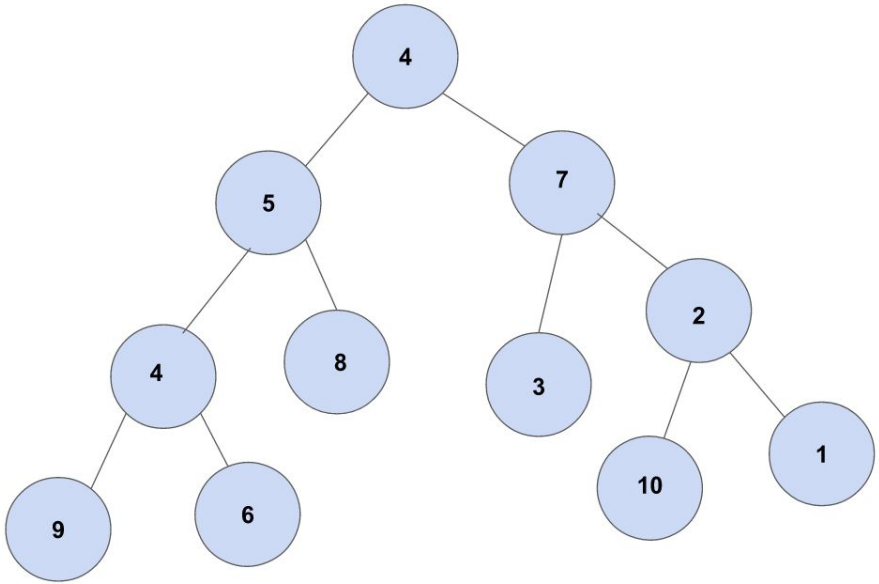
Приклад жадібного алгоритму

Мета — досягнути найбільшої суми. На кожному кроці жадібний алгоритм вибиратиме те, що здається оптимальним безпосереднім вибором, тому він вибере 12 замість 3 на другому кроці та не досягне найкращого рішення, яке містить 99.



Приклад жадібного алгоритму

Спробуйте самостійно знайти найбільшу суму жадібним способом і проходячи всі можливі шляхи та обираючи справді найбільшу.

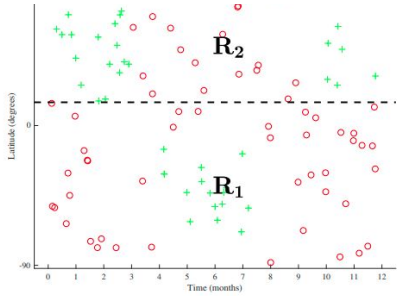


Приклад побудови дерева

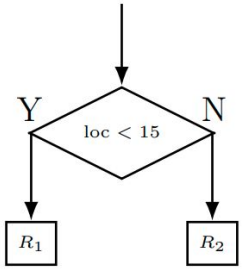
Подивимося на приклад побудови дерева, а потім докладніше розглянемо, що відбувається на кожному етапі.

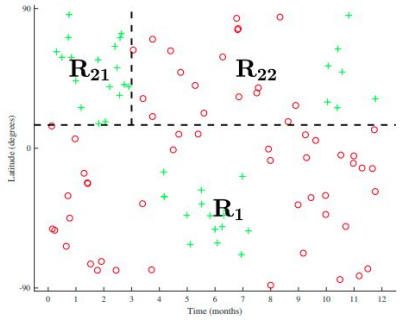
На кожному етапі ми знаходимо критерій, за яким нам найкраще розділити дані. Критерій складається з ознаки та її значень.

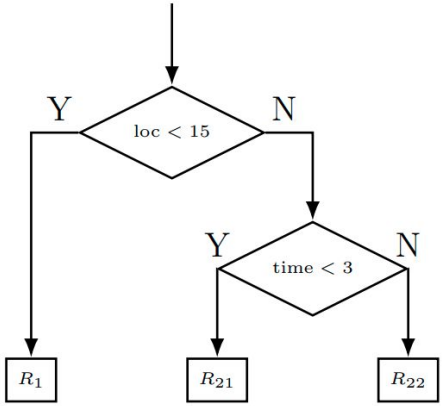
Етапи рекурсивно повторюються.



(a)

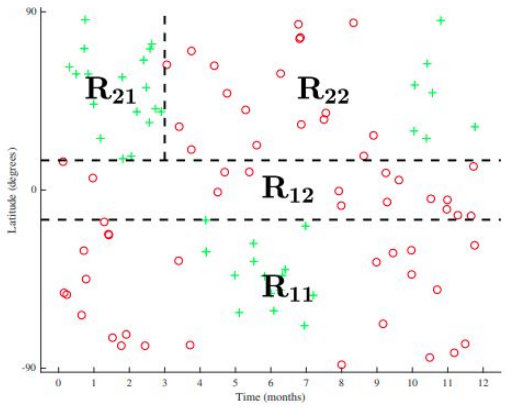




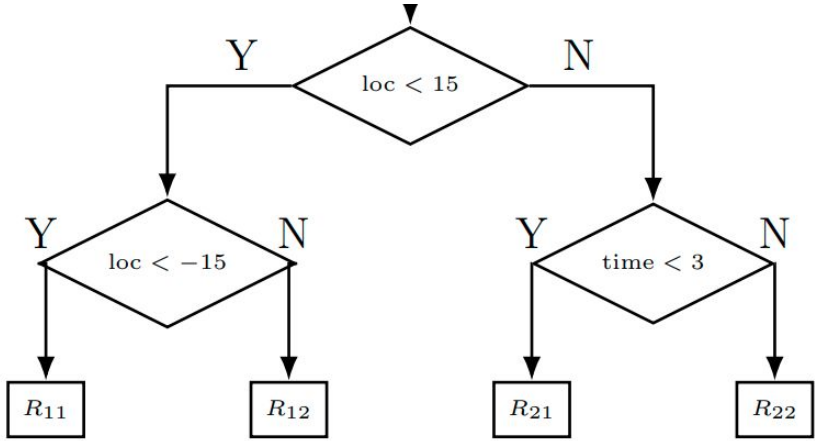


Приклад побудови дерева

Ми продовжуємо розділяти дані, поки не досягнемо умови зупинки.



(c)



Процес побудови дерева

Загалом, щоб побудувати дерево, нам потрібно вирішити наступні задачі:

1. Вибір атрибута, за яким буде проводитися розбиття в кожному наступному вузлі (атрибута розбиття).
2. Вибір критерію зупинки навчання.
3. Вибір методу відсічення (prunning) гілок (спрощення моделі).
4. Оцінка точності побудованого дерева.

Далі дізнаємось детально як вирішувати кожну з цих задач.

Вирішення задач на шляху до побудови дерева

Вибір атрибута розбиття

Загальне правило: обраний атрибут повинен розділити множину спостережень у вузлі так, щоб результуючі підмножини містили приклади з **однаковими мітками класу**, або були **максимально наближені до цього**, тобто кількість об'єктів з інших класів («домішок») в кожному з цих множин було якомога менше.

Для цього були обрані різні критерії, найбільш популярними з яких стали **теоретико-інформаційний** та **статистичний**.

Теоретико-інформаційний критерій

Як випливає з назви, критерій ґрунтується на поняттях **теорії інформації**, а саме — інформаційної **ентропії** (ентропія грубо кажучи — інформаційна невизначеність (або хаос) в системі):

$$H = -\sum_{i=1}^{n} \frac{N_i}{N} \log\left(\frac{N_i}{N}\right)$$

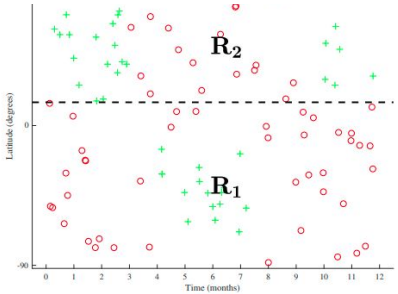
де n — кількість класів у вихідній підмножині, N_i — кількість прикладів i-го класу, N — загальна кількість прикладів у підмножині.

Ентропія — міра **неоднорідності** підмножини за представленими в ній класами. Коли класи представлені у рівних частках в листі і невизначеність класифікації найбільша, ентропія також максимальна. Якщо всі приклади в вузлі відносяться до одного класу, тобто N=N_i, логарифм від одиниці перетворює ентропію в 0.

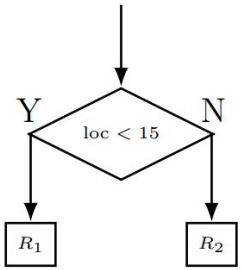
Найкращим атрибутом ознаки для розбиття буде той, що забезпечить максимальне зниження ентропії результатуючої підмножини в порівнянні з батьківською.

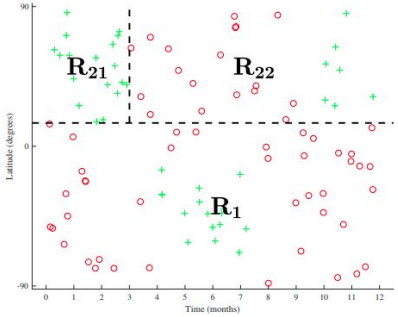
На прикладі

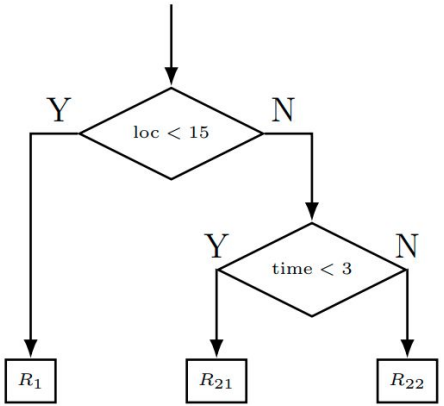
Тобто тут ми шукали б за теоретико-інформаційним критерієм таке правило, яке б нам зменшило ентропію. При цьому ми перебираємо всі можливі правила і межі атрибутів.



(a)







Статистичний підхід

Основою статистичного підходу є використання **індексу Джині** (названий на честь італійського статистика і економіста Коррадо Джині). Індекс Джині *мінімізує ймовірність того, що екземпляр буде класифікований невірно*.

Статистичний зміст показника: наскільки часто випадково обраний приклад навчальної множини буде розпізнаний неправильно, за умови, що цільові значення в цій множині були взяті з певного визначеного статистичного розподілу.

Індекс Джині фактично показує відстань між двома розподілами — розподілом цільових значень, і розподілом передбачень моделі. Чим менше дана відстань, тим краще працює модель.

Буде зрозуміліше після розгляду формули.

Формула індексу Джині

$$\text{Gini}(D) = 1 - \sum_{i=1}^{m} P_i^2$$

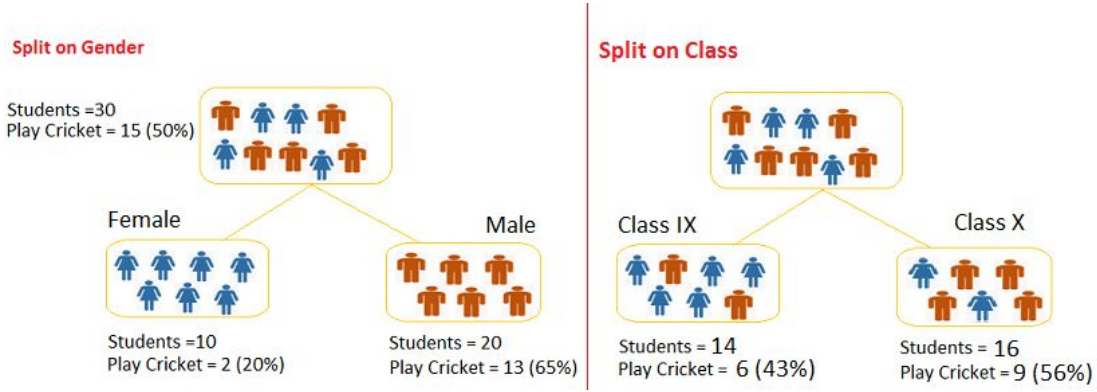
де D — результуюча множина, m — кількість класів у ній, p_i — ймовірність і-го класу виражена як відносна частота прикладів відповідного класу.

Розуміємо, що цей показник змінюється від 0 до 1. При цьому він дорівнює 0, якщо всі приклади D належать до одного класу, і прямує до 1, коли класи представлені в однакових пропорціях і рівноймовірні. Тоді найкращим буде те розбиття, для якого значення індексу Джині буде мінімальним.

Для певного конкретного розділення на дві підмножини формула наступна

$$\text{Gini}_A(D) = \frac{|D1|}{|D|}\text{Gini}(D_1) + \frac{|D2|}{|D|}\text{Gini}(D_2)$$

Приклад застосування критерію Джині: розділення студентів на основі цільової змінної (грають у крикет чи ні).



Розбиття за Статтю:

Джині для підвузла Жінки = 1- ((0.2)*(0.2)+(0.8)*(0.8))=0.32

Джині для підвузла Чоловіки = 1- ((0.65)*(0.65)+(0.35)*(0.35))=0.45

Зважений Джині для спліту за Статтю = (10/30)*0.32+(20/30)*0.45 = 0.407

Аналогічно розбиття за Класом (роком навчання):

Джині для підвузла Клас IX = 1 - ((0.43)*(0.43)+(0.57)*(0.57))=0.49

Джині для підвузла Клас X = 1 - ((0.56)*(0.56)+(0.44)*(0.44))=0.49

Зважений Джині для спліту за Класом = (14/30)*0.49+(16/30)*0.49 = 0.49

Джині при розбитті за Статтю нижче, ніж при розбитті за Класом, отже наступний вузол буде розділяти за Статтю.

Вибір критерію розбиття в scikit-learn

У моделях sklearn.tree параметр criterion присвячено вибору критерію розбиття:

- criterion: {"gini", "entropy"}, default="gini"

Умови зупинки

Рекурсивний процес розбиття даних триває до тих пір, поки всі **вузли** на кінці всіх гілок не будуть оголошені **листками**.

Оголошення вузла листком відбувається природним чином, *коли він буде містити один об'єкт або об'єкти лише одного класу*, але перше може викликати перенавчання і погану узагальненість (точність на цільовій вибірці), а друге малоймовірно.

Рішення: зупинка за **користувацькими** умовами.

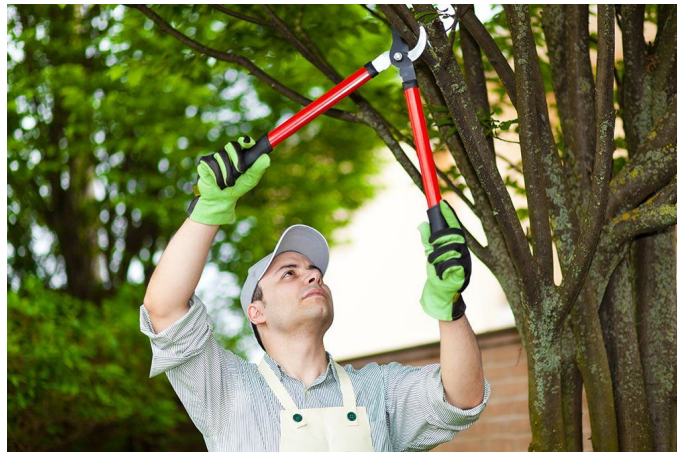
Користувацькі умови зупинки

- **Рання зупинка**: алгоритм буде зупинено, як тільки буде досягнуто вказаного значення певного критерію, наприклад відсоткової частки правильно розпізнаних прикладів.
    - Перевага: зменшення часу навчання.
    - Недолік: гірша точність дерева, тому багато авторів рекомендують віддавати перевагу відкусуванню гілок.
- **Обмеження глибини дерева**: задання максимальної кількості розбиттів у гілках, при досягненні якого навчання припиняється. Цей метод також призводить до зменшення точності дерева.
- **Задання мінімально допустимої кількості прикладів у вузлі:** заборонити алгоритму створювати вузли з числом прикладів менше заданого (наприклад, 5). Це дозволить уникнути створення тривіальних розбиттів і, відповідно, малозначущих правил

Відсічення гілок (prunning)

Підхід, альтернативний ранній зупинці — побудувати всі можливі дерева й вибрати те, яке при розумній глибині забезпечує прийнятний рівень помилок в передбачення, тобто **знайти найвигідніший баланс між складністю та точністю дерева**.

- Погані новини: ця задача відноситься до класу задач NP-повних (для неї не існує ефективних алгоритмів розв'язання), що було показано Л. Хайфілем (L. Hyafill) та Р. Рівестом (R. Rivest).
- Добрі: дослідники вже придумали спрощений обхід 👌



Відсічення гілок

Містить наступні кроки:

- Побудувати повне дерево (щоб **всі** листки містили приклади **одного** класу).
- Визначити два показники:
  - відносну точність моделі — відношення кількості правильно впізнаних прикладів до загальної кількості прикладів,
  - і абсолютну помилку — кількість неправильно класифікованих прикладів.
- Видалити з дерева листки і вузли, відсічення яких не призведе до значущого зменшення точності моделі або збільшення помилки.

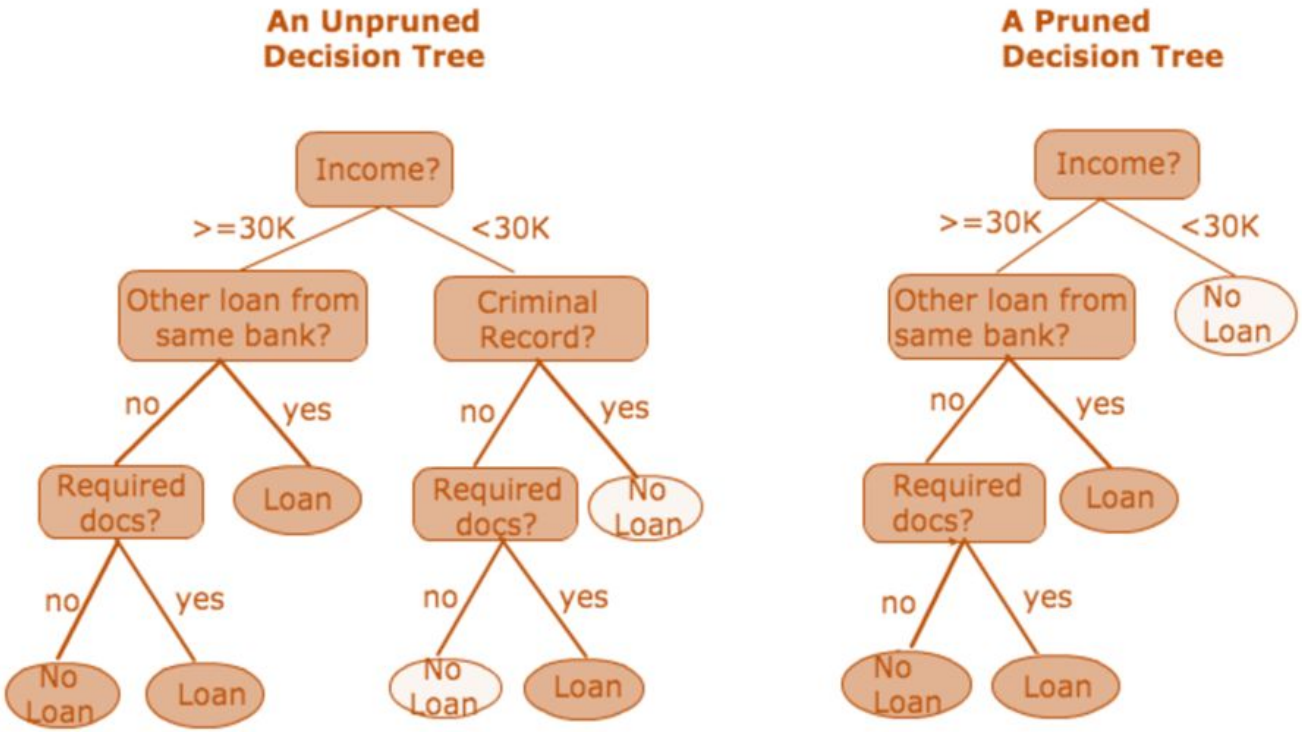
Відсічення гілок

Відсікання гілок, як ми розуміємо, виконується у напрямку, протилежному напряму росту дерева, тобто знизу вгору, шляхом послідовного перетворення вузлів у листя.

Перевага відсікання гілок в порівнянні з раннім припиненням: можливість пошуку оптимального співвідношення між точністю та зрозумілістю дерева.

Недолік: більш тривалий час навчання через необхідність спочатку побудувати повне дерево.

Приклад


An Unpruned Decision Tree
Income?
>=30K
<30K
Other loan from same bank?
Criminal Record?
no
yes
no
yes
Required docs?
Loan
Required docs?
No Loan
no
yes
no
yes
No Loan
Loan
No Loan
Loan
A Pruned Decision Tree
Income?
>=30K
<30K
Other loan from same bank?
No Loan
no
yes
Required docs?
Loan
no
yes
No Loan
Loan

Отримання правил з дерев

Іноді навіть спрощене дерево прийняття рішень все ще є надто складним для візуального сприйняття та інтерпретації. У цьому випадку може бути корисним **видобути з дерева рішень правила** та організувати їх у набори, які описуватимуть класи.

Для видобутку правил потрібно відстежити всі шляхи від кореневого вузла до листя дерева. Кожен такий шлях надасть правило, що складатиметься з множини умов, що є перевіркою в кожному вузлі шляху.

```
|--- petal width (cm) <= 0.80
|   |--- class: 0
|--- petal width (cm) >  0.80
|   |--- petal width (cm) <= 1.75
|   |   |--- class: 1
|   |--- petal width (cm) >  1.75
|   |   |--- class: 2
```

Приклад правил, отриманих з дерева

Алгоритми навчання дерев прийняття рішень

Існує безліч алгоритмів навчання дерев рішень: ID3, CART, C4.5, C5.0, NewId, ITrule, CHAID, CN2 та інші. Однак найбільш поширені та популярні наступні:

- ID3 (Iterative Dichotomizer 3) — алгоритм дозволяє працювати лише з дискретною цільовою змінною, тому дерева рішень, побудовані за допомогою цього алгоритму, є **класифікуючими**. Кількість нащадків у вузлі дерева не обмежена. **Не може працювати з пропущеними** (NULL) значеннями.
- C4.5 — вдосконалена версія алгоритму ID3, до якої додана можливість роботи з **пропущеними** значеннями атрибутів (за версією видання Springer Science 2008 року алгоритм посів перше місце в топ-10 найпопулярніших алгоритмів Data Mining).
- CART (Classification and Regression Tree) — алгоритм навчання дерев рішень, що дозволяє використовувати як дискретну, так і неперервну цільову змінну, тобто вирішувати як завдання **класифікації**, так і **регресії**. Алгоритм будує дерева, які в кожному вузлі мають тільки двох нащадків. В sklearn використовується саме цей алгоритм.

Додатково для ознайомлення

- https://scikit-learn.org/stable/modules/tree.html