

---

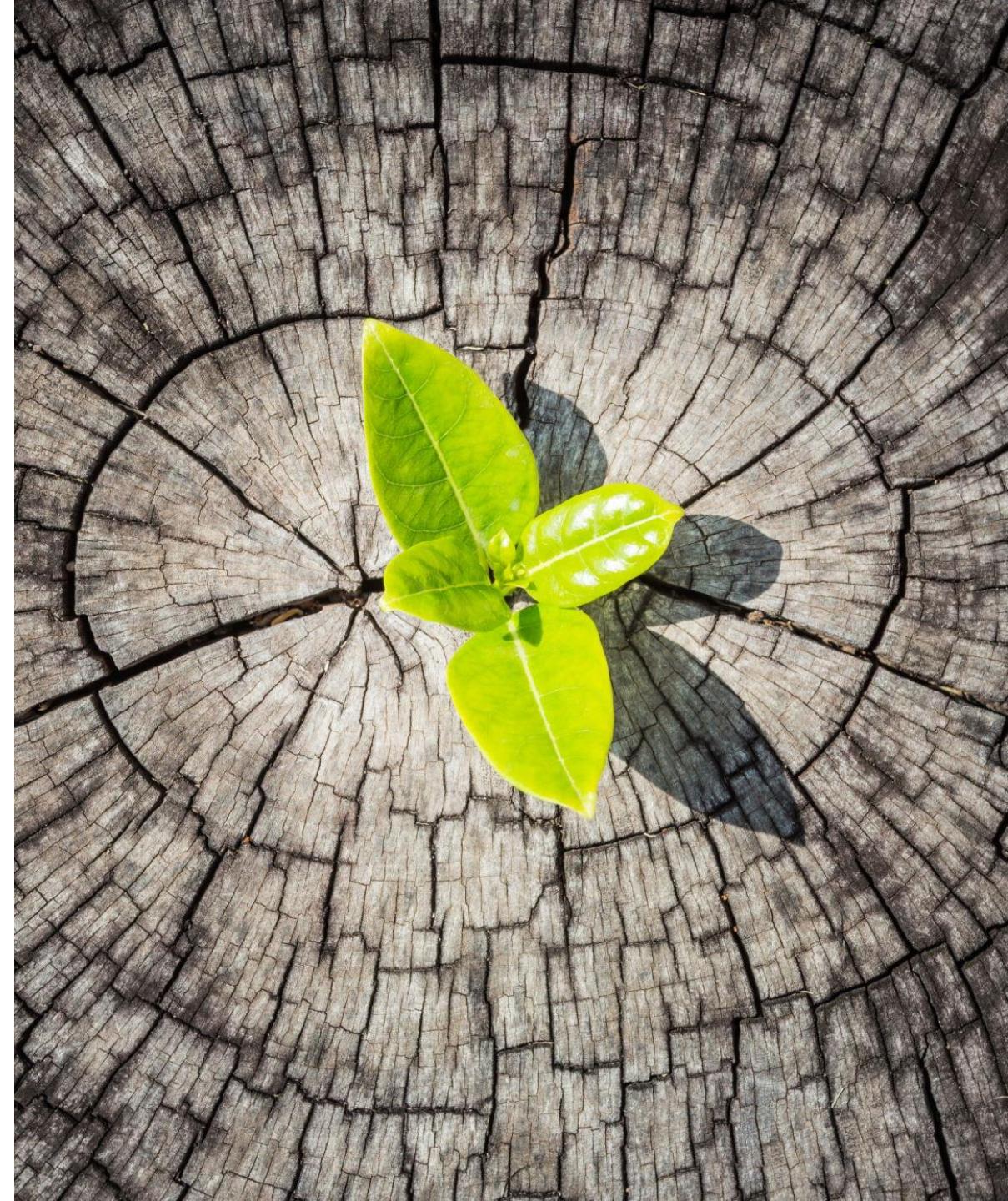
# DECISION TREE

Autorzy: Wojciech Kozub

Jakub Kozdrój

[Repo](#)

---



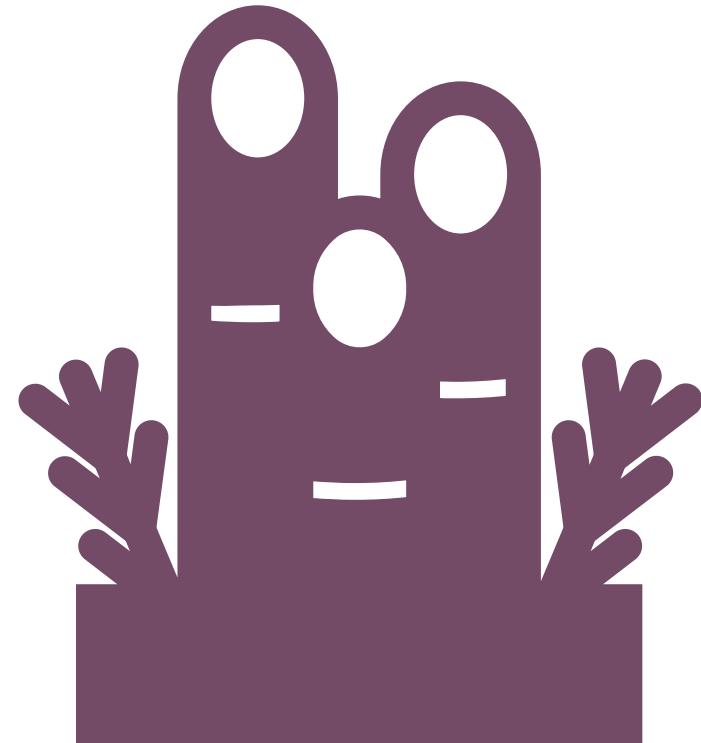
---

# CEL PROJEKTU

Celem projektu było zaimplementowanie własnego drzewa decyzyjnego od podstaw.

Przetestowanie na Breast cancer oraz Wine dataset.

Porównanie wyników z drzewem decyzyjnym sklearn.



# CZYM JEST DRZEWO DECYZYJNE

**Drzewo decyzyjne** to model uczenia nadzorowanego, który przewiduje etykietę poprzez kolejne podejmowanie decyzji opartych na wartościach cech.

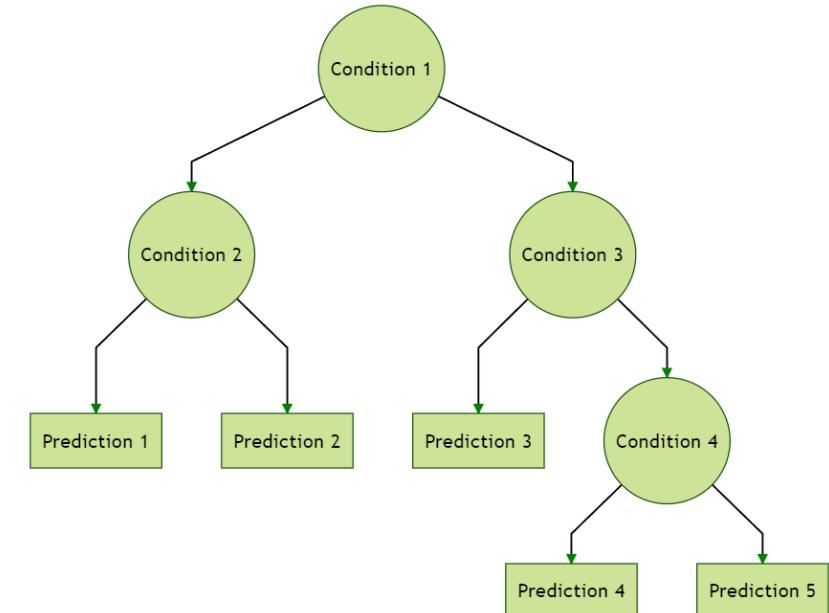
**Najważniejsze elementy:**

- **Root node** – początek drzewa, pierwszy podział danych
- **Internal nodes** – węzły, w których wykonywane są decyzje („czy cecha  $X \leq$  wartość  $Y$ ?“)
- **Leaves** – końcowe węzły zawierające przewidywaną klasę

**Jak działa?**

Algorytm iteracyjnie **dzieli dane na coraz bardziej jednorodne grupy**, wybierając cechę i próg podziału, które najlepiej redukują niejednorodność (impurity).

- Przewidywanie odbywa się poprzez **przejście od korzenia do liścia** zgodnie z warunkami podziałów.



---

# KRYTERIUM PODZIAŁU GRUP

## Cel podziału:

Wybrać takie miejsce, które maksymalnie zwiększa jednorodność danych w powstałych gałęziach.

- **Gini impurity**

Mierzy poziom nieuporządkowania klas w grupie.

## gdzie:

- $n$  – liczba klas
- $p_i$  – proporcja (udział) elementów klasy  $i$  w danym zbiorze
- $p_i^2$  – „czystość” klasy  $i$ , czyli prawdopodobieństwo, że dwa losowe elementy będą tej samej klasy

$$Gini = 1 - \sum_{i=1}^n (p_i)^2$$

## Interpretacja:

- Gini = 0 → zbiór jest **idealnie jednorodny** (wszystko jedna klasa)
- Gini blisko 1 → zbiór jest **maksymalnie wymieszany**
- Im mniejsza wartość Gini, tym lepszy podział



---

# BUDOWANIE DRZEWIA

## 1. Start od całego zbioru danych

Tworzony jest korzeń drzewa, zawierający wszystkie próbki.

## 2. Szukanie najlepszego podziału

Dla każdej cechy i możliwego progu:

- obliczana jest nieczystość (np. Gini) w lewej i prawej części,
- wybierany jest split dający największy zysk.

## 3. Tworzenie gałęzi

Zbiór dzielony jest na dwa podzbiory – lewy i prawy węzeł.

Algorytm wywołuje się rekurencyjnie dla obu części.

## 4. Warunki stopu

Tworzenie nowych węzłów kończy się, gdy:

- osiągnięto maksymalną głębokość,
- liczba próbek jest zbyt mała,
- wszystkie próbki mają tę samą klasę.

## 5. Tworzenie liścia

Węzeł staje się liściem i przechowuje przewidywaną klasę.

---

---

# IMPLEMENTACJA

- Kod podzielony na moduły

src/

node.py # pojedynczy węzeł drzewa  
metrics.py # gini\_impurity, accuracy  
decision\_tree.py # logika drzewa

- Styl scikit-learn

fit(X, y)

predict(X)

score(X, y)

---

# UNIT TESTS

```
pytest
=====
test session starts =====
platform linux -- Python 3.12.3, pytest-9.0.1, pluggy-1.
6.0
rootdir: /home/wojtek/Documents/decision_tree
collected 11 items

tests/test_metrics.py .... [ 36%]
tests/test_node.py .. [ 54%]
tests/test_tree.py ... [ 81%]
tests/test_tree_sklearn.py .. [100%]

===== 11 passed in 3.51s =====
```



pytest

---

# PORÓWNANIE

- Podział: 70% train / 30% test
- Accuracy zblżone do scikit-learn (różnice ~1–2 p.p.)
- Nasz kod jest dużo prostszy i przez to dużo wolniejszy w treningu
- Czasy predykcji podobne - małe zbiory

| Dataset       | Sample | Feature | Metric           | our      | sklearn  | Difference / ratio |
|---------------|--------|---------|------------------|----------|----------|--------------------|
| Breast cancer | 569    | 30      | Accuracy         | 0.9181   | 0.9298   | 0.0117             |
|               |        |         | Train time [s]   | 2.326839 | 0.005548 | 419.37             |
|               |        |         | Predict time [s] | 0.000183 | 0.000091 | 2.02               |
| Wine          | 178    | 13      | Accuracy         | 0.9444   | 0.9630   | 0.0185             |
|               |        |         | Train time [s]   | 0.157467 | 0.001076 | 146.33             |
|               |        |         | Predict time [s] | 0.000047 | 0.000074 | 0.64               |

DZIĘKUJEMY ZA  
UWAGĘ



