

# Drzewa Decyzyjne

Igor Wojnicki

March 24, 2023

# Plan prezentacji

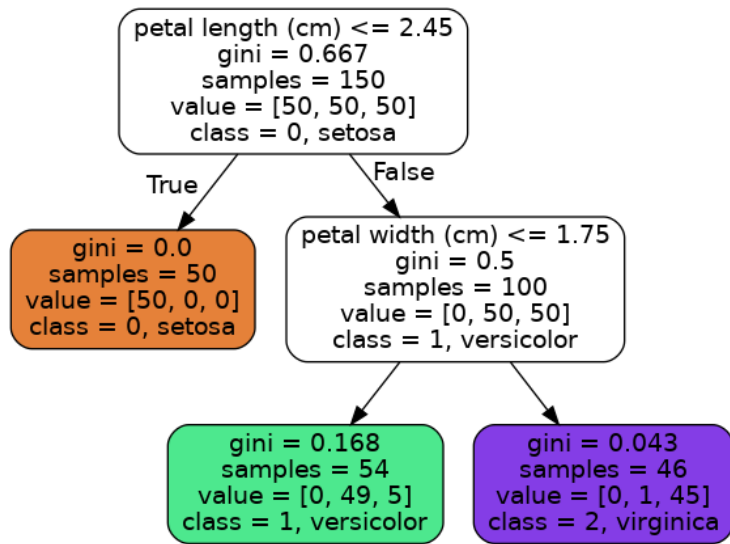
## Drzewa Decyzyjne

## Drzewa Decyzyjne, uczenie

- ▶ klasyfikacja i regresja
- ▶ Nie wymagają skalowania cech!

[illegible]

## Rezultat procesu uczenia



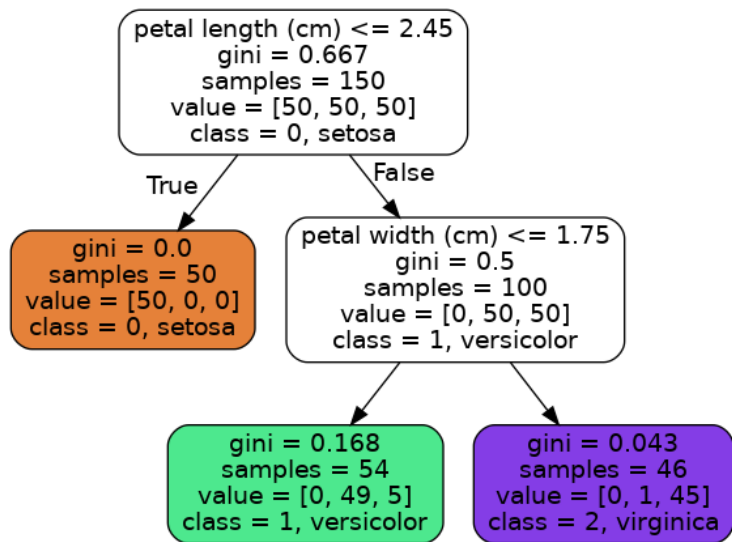
Czy jest to model parametryczny?

## Co to jest drzewo decyzyjne?

```
1  from sklearn.tree import export_graphviz
2  f = "iris_tree.dot"
3  export_graphviz(
4      tree_clf,
5      out_file=f
6      feature_names=iris.feature_names[2:],
7      class_names=[str(num)+"", "+name
8                  for num,name in
9                  zip(set(iris.target),
10                     iris.target_names)],
11      rounded=True,
12      filled=True
13  )
14  print(f)
    iris_tree.dot
```

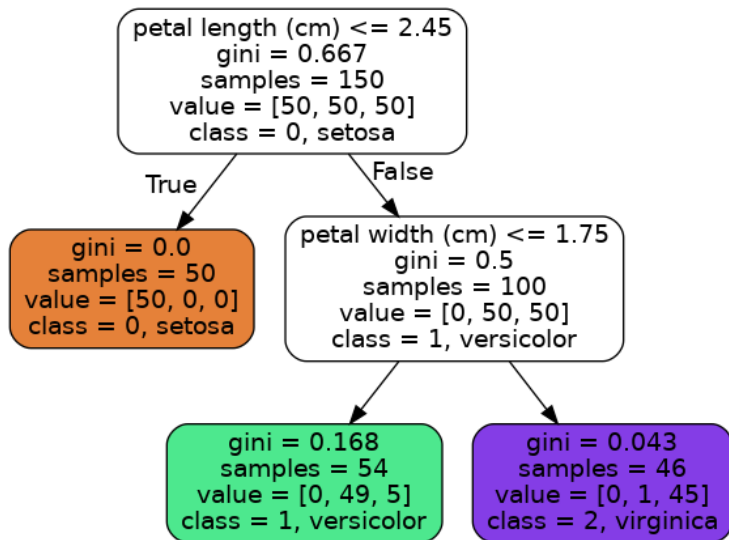
## Pokaż drzewo!

```
dot -Tpng iris_tree.dot -o iris_tree.png
```



# Pokaż drzewo, Python!

```
1 import graphviz
2 print(graphviz.render('dot', 'png', f))
```



## Pokaż drzewo, Jupyter!

```
1  import graphviz
2  graph = graphviz.Source.from_file(f)
3  graph
4
1  str_dot = export_graphviz(
2      tree_clf,
3      out_file=None,
4      feature_names=iris.feature_names[2:],
5      class_names=[str(num)+"", "+name
6                      for num,name in
7                      zip(set(iris.target),
8                          iris.target_names)],
9      rounded=True,
10     filled=True)
11
12 graph = graphviz.Source(str_dot)
13 graph
```



# Drzewa decyzyjne, predykcja

```
1 print(tree_clf.classes_)
2 print(tree_clf.predict([[5.4,1.7]]))
3 print(tree_clf.predict_proba([[5.4,1.7]]))
```

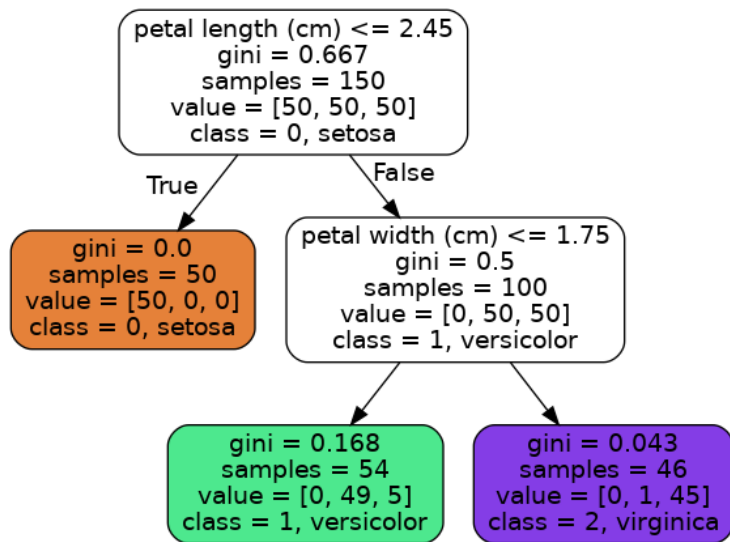
```
[0 1 2]
```

```
[1]
```

```
[[0.          0.90740741  0.09259259]]
```

Prawdopodobieństwo:  $49/54=0.907407407407$

## Reszta informacji



- ▶ *samples* – ile instancji ze zbioru uczącego
- ▶ *value* – ile instancji poszczególnych klas

# Zanieczyszczenie Giniego

- ▶ *gini* – zanieczyszczenie, ile instancji należy do klasy (=0 wszystkie)

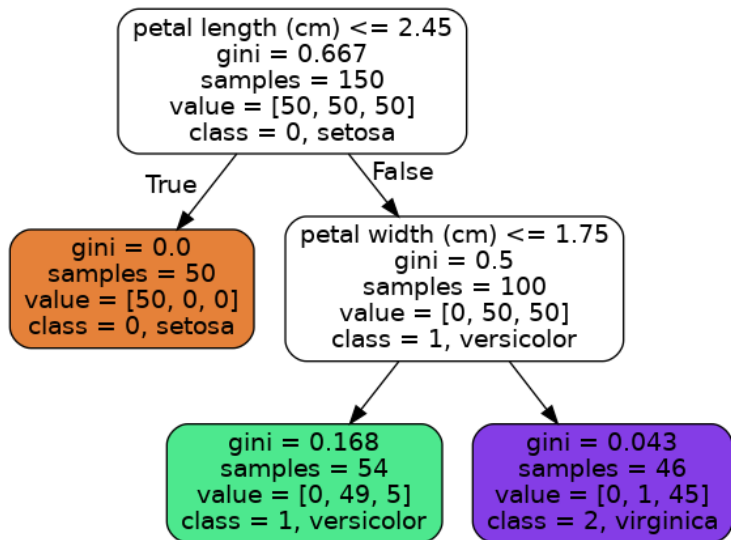
$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2$$

- ▶  $p_{i,k}$  iloraz liczby instancji klasy  $k$  w stosunku do liczby instancji w węźle  $i$

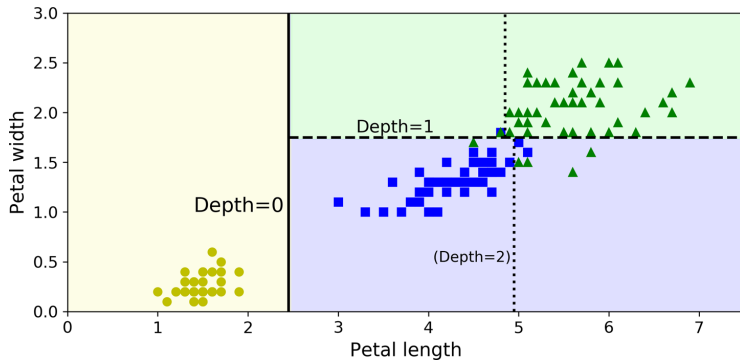
## Zanieczyszczenie

$$1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168$$

$$1 - (0/46)^2 - (1/46)^2 - (45/46)^2 \approx 0.043$$



# Granice podziału drzewa



# Drzewa decyzyjne, regresja, dane

```
1 import numpy as np
2 np.random.seed(42)
3 m = 200
4 X = np.random.rand(m, 1)
5 y = 4 * (X - 0.5) ** 2
6 y = y + np.random.randn(m, 1) / 10
```

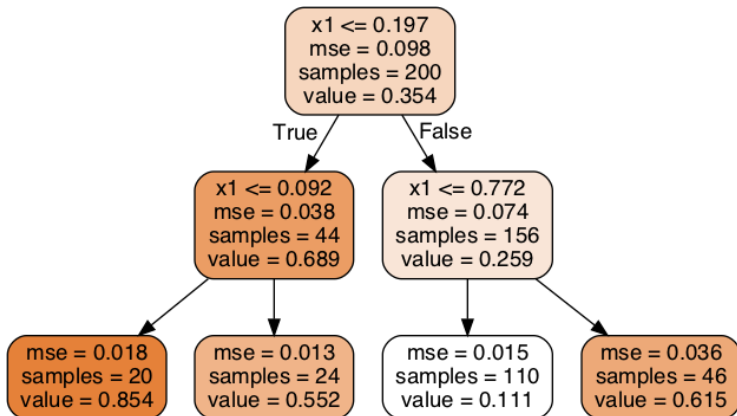
# Drzewa decyzyjne, regresja, uczenie i predykcja

```
1  from sklearn.tree import DecisionTreeRegressor
2
3  tree_reg = DecisionTreeRegressor(max_depth=2,
4                                   random_state=42)
5  tree_reg.fit(X, y)
```

# Drzewa decyzyjne, regresja, predykcja

```
1 print(tree_reg.predict([[0.6]]))
```

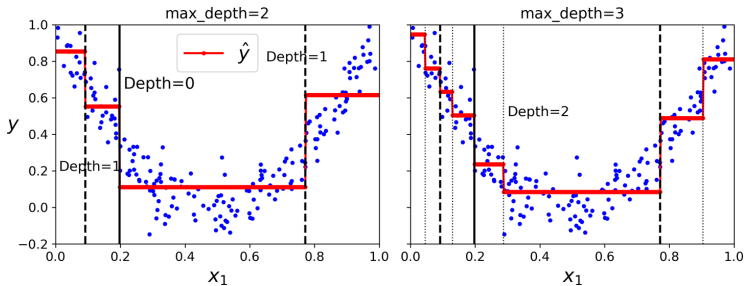
[0.11063973]



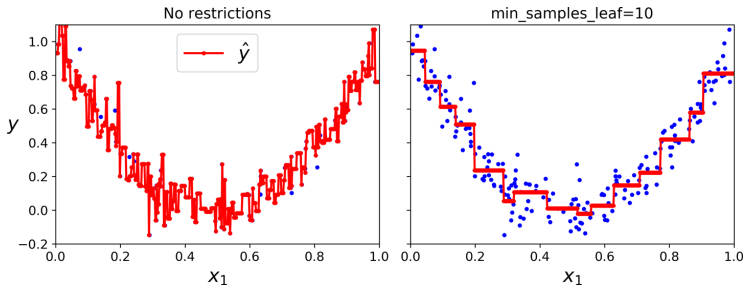
wartość = średnia z wszystkich instancji w węźle



# Drzewa decyzyjne, regresja, drzewo, głębokość?



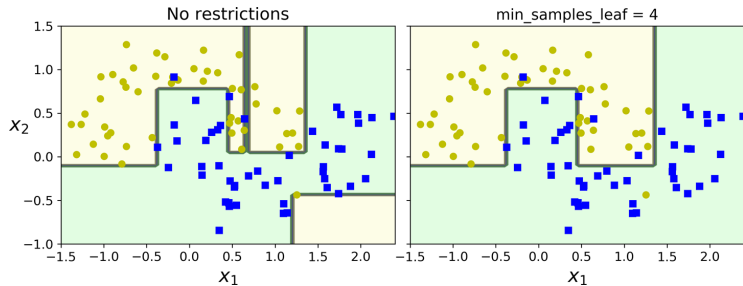
# Drzewa decyzyjne, regresja, hiperparametr



# Drzewa decyzyjne, inne hiperparametry

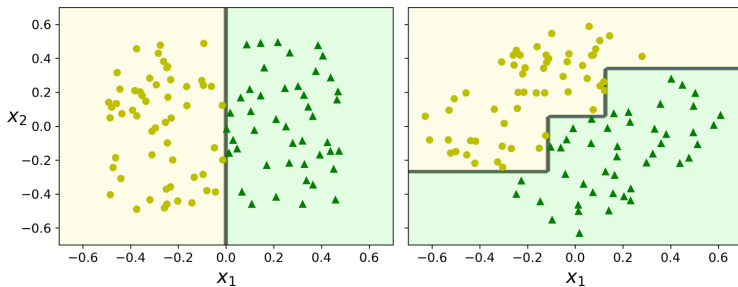
- ▶ Drzewa decyzyjne nie zakładają rodzaju modelu (np. że jest liniowy).
- ▶ Skłonność do overfitting.
- ▶ Jest to model *nieparametryczny* tzn. ilość parametrów nie jest znana przed rozpoczęciem procesu uczenia; w przeciwieństwie do modelu np. liniowego, gdzie ilość parametrów jest znana (współczynniki równania liniowego).
- ▶ *max\_depth* – im mniejsza tym bardziej regularny model, mniejsze ryzyko overfitting.
- ▶ *min\_samples\_split* – minimalna liczba instancji, przy których można podzielić węzeł.
- ▶ *min\_samples\_leaf* – minimalna liczba instancji dla liścia.
- ▶ *min\_weight\_fraction\_leaf*, *max\_leaf\_nodes*, *max\_features*.
- ▶ Zwiększanie *min\_\**, zmniejszanie *max\_\** -> regularyzacja.

# Drzewa decyzyjne, przykładowa regularyzacja



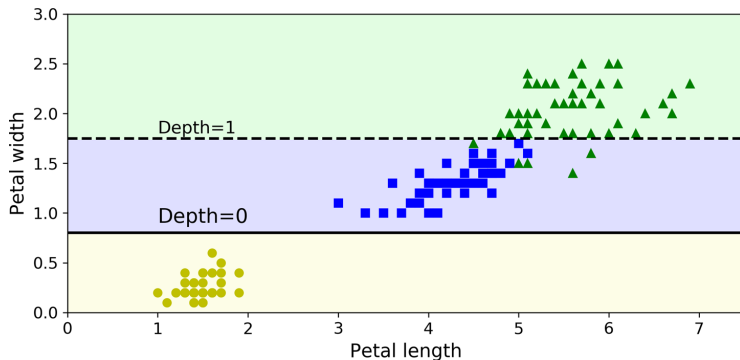
# Drzewa decyzyjne, niestabilność

"Obrót" zbioru uczącego?



# Drzewa decyzyjne, niestabilność

Czułość na zmiany w zbiorze uczącym: np. usunięcie *Iris versicolor* dł. 4.8, szer. 1.8.



# Algorytm CART

- ▶ Classification and Regression Tree.
- ▶ Tworzy drzewo binarne.
- ▶ Algorytm zachłanny.
- ▶ Rozwiązanie nie jest optymalne, ale wystarczająco dobre.

# Algorytm CART, szczegóły

1. Podziel zbiór uczący na dwa, ze względu na pojedynczą cechę  $k$  i próg  $t_k$ ,
  - ▶ tj. znajdź taką parę  $k$  i  $t_k$ , dla której w/w zbiory będą najczystsze (normalizowane liczebnością zbioru).
2. Wykonaj rekurencyjnie dalsze podziały na uzyskanych zbiorach, aż do osiągnięcia maksymalnej głębokości lub jeżeli nie będzie można znaleźć najczystszych zbiorów.  
Minimalizowana funkcja celu (kosztu):

$$J(k, t_k) = \frac{m_{lewa}}{m} G_{lewa} + \frac{m_{prawa}}{m} G_{prawa}$$

gdzie:

- ▶  $G_{lewa/prawa}$  – nieczystość lewego/prawego zbioru,
- ▶  $m_{lewa/prawa}$  – liczba instancji lewego/prawego zbioru,
- ▶  $m$  – liczba wszystkich instancji.



# Algorytm CART, złożoność

- ▶ uczenia:  $O(n * m * \log_2(m))$ ,  $n$  – ilość cech,  $m$  – ilość instancji
- ▶ predykcji:  $O(\log_2(m))$ ,  $m$  – ilość węzłów drzewa.

# Inne algorytmy

- ▶ ID3: Iterative Dichotomizer 3.
  - ▶ Algorytm zachłanny.
  - ▶ Rozwiązanie nie jest optymalne, ale wystarczająco dobre.
  - ▶ Podział bazujący na entropii (niepewności związanej ze zmienną losową).
  - ▶ klasa `DecisionTreeClassifier`, parametr:  
`criterion='entropy'`
- ▶ C4.5
  - ▶ Podział bazujący na zysku informacyjnym (Information Gain).
  - ▶ Ulepszona wersja ID3.