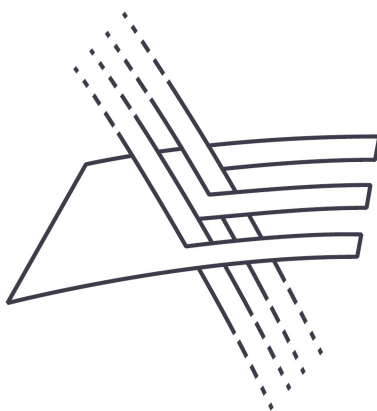


# Ćwiczenie 4

## Klasyfikator ID3

Wprowadzenie do sztucznej inteligencji

Mikołaj Roszczyk



Wydział Elektroniki i Technik Informatycznych  
Politechnika Warszawska  
28 kwietnia 2023

## Spis treści

<b>1</b>	<b>Zadanie i kod</b>	<b>2</b>
1.1	Polecenie do wykonania . . . . .	2
1.2	Napisany kod . . . . .	2
<b>2</b>	<b>Ocena trenowania na zbiorach danych</b>	<b>3</b>
2.1	Dokładność . . . . .	3
2.1.1	Dane <i>breast-cancer.data</i> . . . . .	3
2.1.2	Dane <i>agaricus-lepiota.data</i> . . . . .	4
2.2	Macierz pomyłek . . . . .	4
2.2.1	Dane <i>breast-cancer.data</i> . . . . .	5
2.2.2	Dane <i>agaricus-lepiota.data</i> . . . . .	5
<b>3</b>	<b>Porównanie wyników na dwóch zbiorach</b>	<b>6</b>

# 1 Zadanie i kod

## 1.1 Polecenie do wykonania

Zaimplementować klasyfikator ID3 (drzewo decyzyjne). Atrybuty nominalne, testy tożsamościowe. Podać dokładność i macierz pomyłek na zbiorach: Breast cancer i mushroom. Dlaczego na jednym zbiorze jest znacznie lepszy wynik niż na drugim? Do potwierdzenia lub odrzucenia postawionych hipotez konieczne może być przeprowadzenie dodatkowych eksperymentów ze zmodyfikowanymi zbiorami danych. Sformułować i spisać wnioski.

## 1.2 Napisany kod

Ze względu na obszerność kodu, do sprawozdania tekstowego wstawiam jedynie link do repozytorium:

[https://github.com/Roszczyk/WSI\\_cwiczenia/blob/master/Cw4/ID3\\_universal.py](https://github.com/Roszczyk/WSI_cwiczenia/blob/master/Cw4/ID3_universal.py)

## 2 Ocena trenowania na zbiorach danych

### 2.1 Dokładność

Abu zbadać Accuracy wykorzystano następujący kod:

```
sumAccuracy=0
iterations=25
for i in range(iterations):
    dataArray, testingData = divideData(initFile(fileName))
    columns = len(dataArray[0]) - 1 # liczba kolumn bez klasy
    tree=recurrentID3(dataArray, countEntropy(dataArray), len(dataArray[0])-1)
    countTestingData=len(testingData)
    countTrue=0
    for i in range(countTestingData):
        predicted=predict(testingData[i], tree)
        if predicted==testingData[i][0]:
            countTrue=countTrue+1
    accuracy=countTrue/countTestingData
    print("accuracy: ", accuracy)
    sumAccuracy=accuracy+sumAccuracy
print("averageAccuracy: ", sumAccuracy/iterations)
```

#### 2.1.1 Dane *breast-cancer.data*

Dokładność w 25 wywołaniach:

```
accuracy: 0.6
accuracy: 0.6271186440677966
accuracy: 0.652542372881356
accuracy: 0.5811965811965812
accuracy: 0.6324786324786325
accuracy: 0.6055045871559633
accuracy: 0.7058823529411765
accuracy: 0.6875
accuracy: 0.6186440677966102
accuracy: 0.5816326530612245
accuracy: 0.6052631578947368
accuracy: 0.6456692913385826
accuracy: 0.7064220183486238
accuracy: 0.6875
accuracy: 0.5862068965517241
accuracy: 0.6509433962264151
accuracy: 0.6095238095238096
accuracy: 0.5952380952380952
accuracy: 0.6513761467889908
accuracy: 0.5841584158415841
accuracy: 0.6982758620689655
```

```
accuracy: 0.5963302752293578
accuracy: 0.6181818181818182
accuracy: 0.5048543689320388
accuracy: 0.6347826086956522
averageAccuracy: 0.6266890420975892
```

Średnia dokładność wynosi około **0.63** dla tego zbioru danych.

### 2.1.2 Dane *agaricus-lepiota.data*

Dokładność w 25 wywołaniach:

```
accuracy: 0.9987692307692307
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 0.9987397605545053
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
accuracy: 1.0
averageAccuracy: 0.9999003596529493
```

Średnia dokładność wynosi około **1.0** dla tego zbioru danych.

## 2.2 Macierz pomyłek

Aby zbadać macierz pomyłek wykorzystano następujący kod:

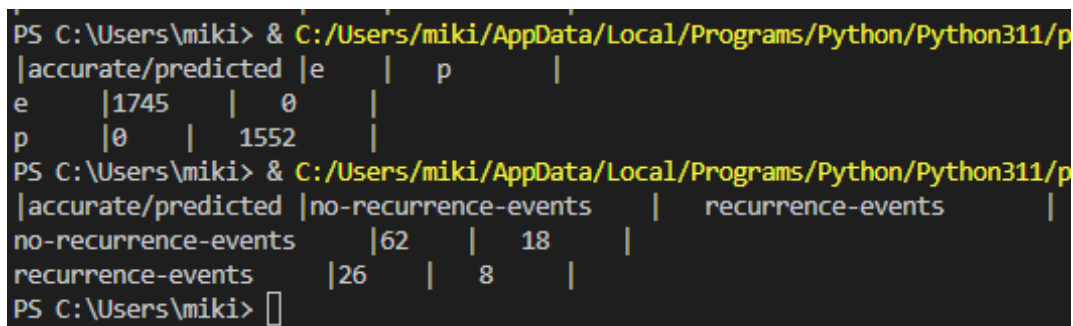
```
dataArray, testingData = divideData(initFile(fileName))
columns = len(dataArray[0]) - 1 # liczba kolumn bez klasy
tree=recurrentID3(dataArray, countEntropy(dataArray), columns)
countTestingData=len(testingData)
```

```

countTrue=0
classSet=defineClassSet(testingData)
matrix=[0,0,0,0]
for i in range(countTestingData):
    predicted=predict(testingData[i], tree)
    if predicted==classSet[0] and testingData[i][0]==classSet[0]:
        matrix[0]=matrix[0]+1
    if predicted==classSet[0] and testingData[i][0]==classSet[1]:
        matrix[1]=matrix[1]+1
    if predicted==classSet[1] and testingData[i][0]==classSet[1]:
        matrix[2]=matrix[2]+1
    if predicted==classSet[1] and testingData[i][0]==classSet[0]:
        matrix[3]=matrix[3]+1
print(f"|accurate/predicted |{classSet[0]}      |      {classSet[1]}      |")
print(f"|{classSet[0]}      |{matrix[0]}      |      {matrix[3]}      |")
print(f"|{classSet[1]}      |{matrix[1]}      |      {matrix[2]}      |")

```

Przykładowy output tego kodu przedstawiono na rysunku 1.



```

PS C:\Users\miki> & C:/Users/miki/AppData/Local/Programs/Python/Python311/p
|accurate/predicted |e      |      p      |
e      |1745   |      0      |
p      |0      |     1552    |
PS C:\Users\miki> & C:/Users/miki/AppData/Local/Programs/Python/Python311/p
|accurate/predicted |no-recurrence-events |      recurrence-events |
no-recurrence-events |62      |      18      |
recurrence-events   |26      |      8       |
PS C:\Users\miki> 

```

Rys. 1: Macierz pomyłek dla grzybów (wyżej) i raka piersi (niżej)

### 2.2.1 Dane *breast-cancer.data*

data/predicted	no-recurrence-events	recurrence-events
no-recurrence-events	62	18
recurrence-events	26	8

### 2.2.2 Dane *agaricus-lepiota.data*

data/predicted	e	p
e	1745	0
p	0	1552

### 3 Porównanie wyników na dwóch zbiorach

Na zbiorze *agaricus-lepiota.data* algorytm uzyskuje dużo lepsze wyniki niż na zbiorze *breast-cancer.data*. Wynika to z następujących powodów:

1. Zbiór *agaricus-lepiota.data* jest dużo liczniejszy. Można wobec tego wytrenować model na dużo większej liczbie danych.
2. Zbiór *agaricus-lepiota.data* posiada więcej atrybutów niż *breast-cancer.data*. Wobec tego można było zrobić bardziej szczegółowy podział przy tworzeniu modelu.
3. Zbiór *breast-cancer.data* zawiera więcej wartości atrybutów, co powoduje zmniejszenie przypadków wystąpienia danej wartości atrybutu przy trenowaniu drzewa.
4. Przy trenowaniu modelu na zbiorze *breast-cancer.data* występowały sytuację, kiedy drzewo nie było wytrenowane dla ścieżki, którą napotkało w danych testowych ze względu na zbyt małą reprezentację danych trenujących. W celu uniknięcia zwracania przez funkcję rekurencyjną *None* został dodany element losowy. W sytuacji, gdy w drzewie nie ma drogi odpowiadającej aktualnemu stanowi, droga jest losowana. Wprowadza to element losowy, który wpływa na dokładność wyniku.