

Komputerowe Wspomaganie Decyzji

Projekt

Temat:

“Odkrywanie konceptu jadalnego grzyba z użyciem algorytmu ID3.”

Wykonali:

Rafał Paprocki

Piotr Ryzak

1.Opis algorytmu ID3.

Drzewa decyzyjne są graficzną metodą wspomagania procesu decyzyjnego. Jest to jedna z najczęściej wykorzystywanych technik analizy danych. Drzewo składają się z korzenia oraz gałęzi prowadzących z korzenia do kolejnych wierzchołków.

Wierzchołki, z których wychodzi co najmniej jedna krawędź, są nazywane węzłami, a pozostałe wierzchołki – liśćmi. W każdym węźle sprawdzany jest pewien warunek dotyczący danej obserwacji, i na jego podstawie wybierana jest jedna z gałęzi prowadząca do kolejnego wierzchołka.

Algorytm ID3 jest jednym z algorytmów operujących na drzewach decyzyjnych. Jego twórcą jest Ross Quinlan, a algorytm pochodzi z 1986r. Cechą charakterystyczną algorytmu jest wybór atrybutów dla których kolejno przeprowadzane są testy takie, aby końcowe drzewo było jak najprostsze i jak najefektywniejsze. Wybór atrybutów opiera się na liczeniu entropii, co pozwala obliczyć wybór, który z atrybutów da największy przyrost informacji. Takim atrybutem jest ten, który podzieli zbiór przykładów na jak najbardziej równe podzbiory.

Entropia w ramach teorii informacji jest definiowana jako średnia ilość informacji (liczba bitów), przypadająca na znak symbolizujący zajście zdarzenia z pewnego zbioru. Zdarzenia w tym zbiorze mają przypisane prawdopodobieństwa wystąpienia.

Wzór na entropię gdzie $p(i)$ to prawdopodobieństwo zajścia zdarzenia i :

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

Własności entropii:

- jest nieujemna
- jest maksymalna, gdy prawdopodobieństwa zajść zdarzeń są takie same
- jest równa 0, gdy stany systemu przyjmują wartości 0 albo 1
- własność superpozycji - gdy dwa systemy są niezależne to entropia sumy systemów równa się sumie entropii.

W bibliotece scikit-learn drzewa decyzyjne implementowane są przez klasę **DecisionTreeClassifier**. Jako wejście przyjmuje ona dwie matryce: matrycę X zawierającą treningowy zestaw próbek oraz matrycę Y zawierającą klasy etykiet dla treningowego zestawu próbek. Po wytrenowaniu klasyfikatora takim zestawem danych możemy użyć go do przewidywania klas etykiet dla danych, które podamy klasyfikatorowi.

2.Opis danych uczących.

Dane o grzybach pobieraliśmy ze strony

https://archive.ics.uci.edu/ml/datasets/mushroom?fbclid=IwAR2OrnzVLH1mFCnD8NRBZu_i_WMfnudmLR1LounuTPu9iq5kn_MnvJDkUXI

Dane te zawierają opis hipotetycznych próbek odpowiadającym dwudziestu trzem gatunkom blaszkowcy z rodziny Agaricus i Lepiota. Każdemu gatunkowi jest przypisana jedna z dwóch klas etykiet : trujący (poisonous) , jadalny (edible).

Oryginalne dane zawierał jeszcze dwie etykiety nieklasyfikowany i nie zalecany pod kątem jadalności, ale zostały one przypisane do etykiety trujący.

Informacje o atrybutach:

1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
6. gill-attachment: attached=a,descending=d,free=f,notched=n
7. gill-spacing: close=c,crowded=w,distant=d
8. gill-size: broad=b,narrow=n
9. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
10. stalk-shape: enlarging=e,tapering=t
11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
16. veil-type: partial=p,universal=u
17. veil-color: brown=n,orange=o,white=w,yellow=y
18. ring-number: none=n,one=o,two=t
19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z
20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
21. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

	target	1.cap-shape	2.cap-surface	3.cap-color	4.bruises	5.odor	6.gill-attachment	7.gill-spacing	8.gill-size	9.gill-color	10.stalk-shape	11.stalk-root	12.stalk-surface-above-ring
0	p	x	s	n	t	p	f	c	n	k	e	e	s
1	e	x	s	y	t	a	f	c	b	k	e	c	s
2	e	b	s	w	t	l	f	c	b	n	e	c	s
3	p	x	y	w	t	p	f	c	n	n	e	e	s
4	e	x	s	g	f	n	f	w	b	k	t	e	s

13.stalk-surface-below-ring	14.stalk-color-above-ring	15.stalk-color-below-ring	16.veil-type	17.veil-color	18.ring-number	19.ring-type	20.spore-print-color	21.population	22.habitat
s	w	w	p	w	o	p	k	s	u
s	w	w	p	w	o	p	n	n	g
s	w	w	p	w	o	p	n	n	m
s	w	w	p	w	o	p	k	s	u
s	w	w	p	w	o	e	n	a	g

Graficzne przedstawienie kilku pierwszych wierszy danych o grzybach

Dane te zawierają 8124 wiersze i 23 kolumny. Pierwsza kolumna stanowi opis czy grzyb jest jadalny(e) czy trujący(p). Pozostałe 22 kolumny to zestaw atrybutów opisujących grzyb. W danych tych występuje 2480 brakujących wartości atrybutów oznaczonych jako '?', wszystkie te dane występują dla atrybutu #11.

Zastosowaliśmy dwa podejścia by je wyeliminować:

1. Wszystkie brakujące dane uzupełniliśmy najczęściej występującą wartością w atrybucie #11.
2. Usunęliśmy wszystkie wiersze zawierające brakujące dane.

3. Kod programu.

Najważniejsze funkcje programu:

a) pobieranie danych uczących

```
mushroom_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.data',
                             names=colNames)
```

b) data preprocessing

#W naszych danych jedynie kolumna 11 zawiera brakujące dane(występuje jako '?'), możemy:

```

#1.Zastapic je najczesciej wystepujaca wartoscia w kolumnie 11
#2.Wyrzucic wszystkie wiersze z brakujacymi danymi
mushroom_data1 = mushroom_data.copy()
mushroom_data2 = mushroom_data.copy()
#najczesciej wystepujaca wartosc w kolumnie 11
mode_value = mushroom_data.mode().iloc[:,11]
#zastepujemy brakujace wartosci wartoscia najczesciej wystepujaca
mushroom_data1.replace("?", np.nan, inplace=True)
mushroom_data1.replace(np.nan, mode_value[0], inplace=True)
#drugie podejscie usuwamy wiersze zawierajace brakujace dane
mushroom_data2.replace("?", np.nan, inplace=True)
mushroom_data2.dropna(inplace=True)
#zamieniamy wartosci tekstowe na liczbowe
mushroom_data1 = mushroom_data1.apply(lambda x: pd.factorize(x,
sort=True)[0])
mushroom_data2 = mushroom_data2.apply(lambda x: pd.factorize(x,
sort=True)[0])
#wydzielamy dane o jadalnosci od pozostalych
mushroom_x1 = mushroom_data1.iloc[:, 1:23]
mushroom_y1 = mushroom_data1.iloc[:, 0]
mushroom_x2 = mushroom_data2.iloc[:, 1:23]
mushroom_y2 = mushroom_data2.iloc[:, 0]

```

c) podział na dane trenujące i testowe.

```

from sklearn.model_selection import train_test_split

x_train1, x_test1, y_train1, y_test1 = \
    train_test_split(mushroom_x1, mushroom_y1, train_size=0.8,
random_state=67)
x_train2, x_test2, y_train2, y_test2 = \
    train_test_split(mushroom_x2, mushroom_y2, train_size=0.8,
random_state=67)

```

d) trenowanie modelu.

```

from sklearn import tree

filled_data_model = tree.DecisionTreeClassifier(criterion='entropy',
max_depth=12, random_state=37)
dropped_data_model = tree.DecisionTreeClassifier(criterion='entropy',
max_depth=12, random_state=37)
filled_data_model.fit(x_train1, y_train1)
dropped_data_model.fit(x_train2, y_train2)

```

e) graficzna wizualizacja drzewa.

```

#wizualizacje drzew
import graphviz

tree_filled = tree.export_graphviz(filled_data_model, out_file=None)

```

```

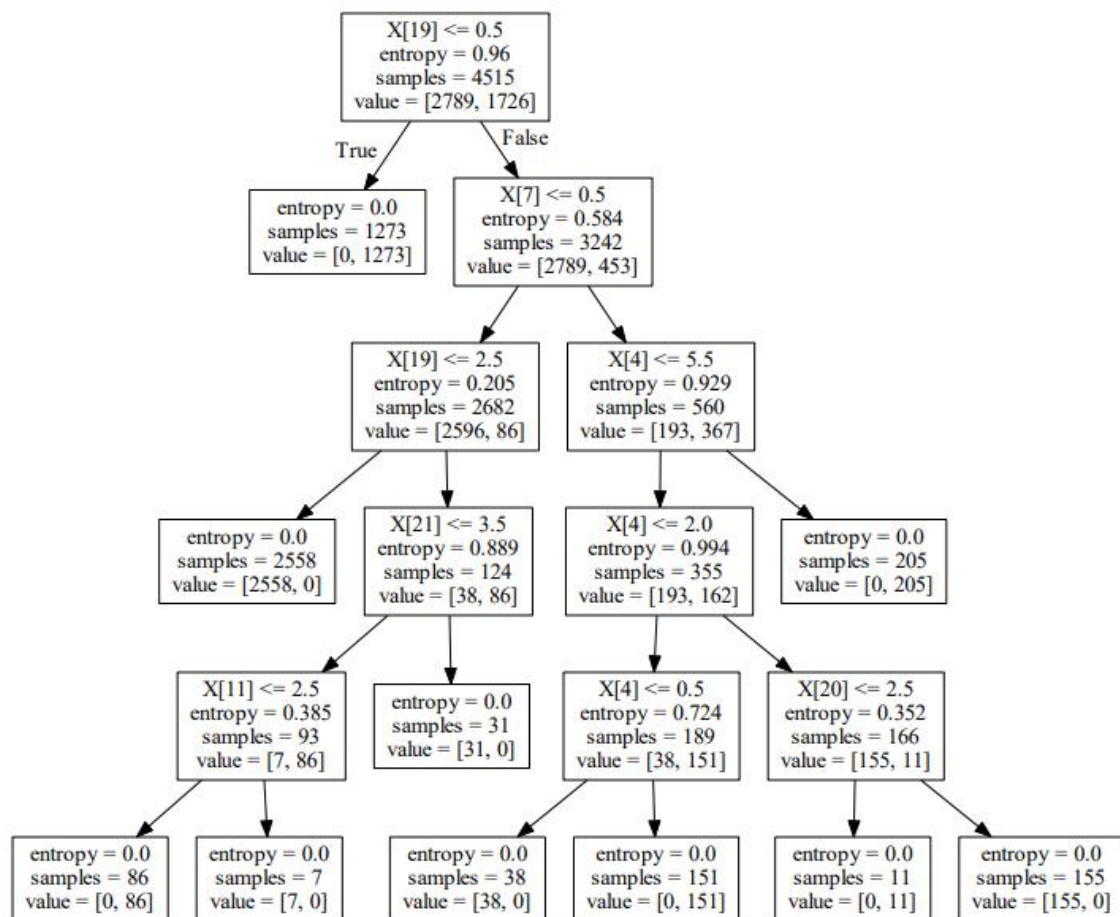
filled_graph = graphviz.Source(tree_filled)
filled_graph.render("Filled data model")

tree_dropped = tree.export_graphviz(dropped_data_model, out_file=None)
dropped_graph = graphviz.Source(tree_dropped)
dropped_graph.render("Dropped data model")

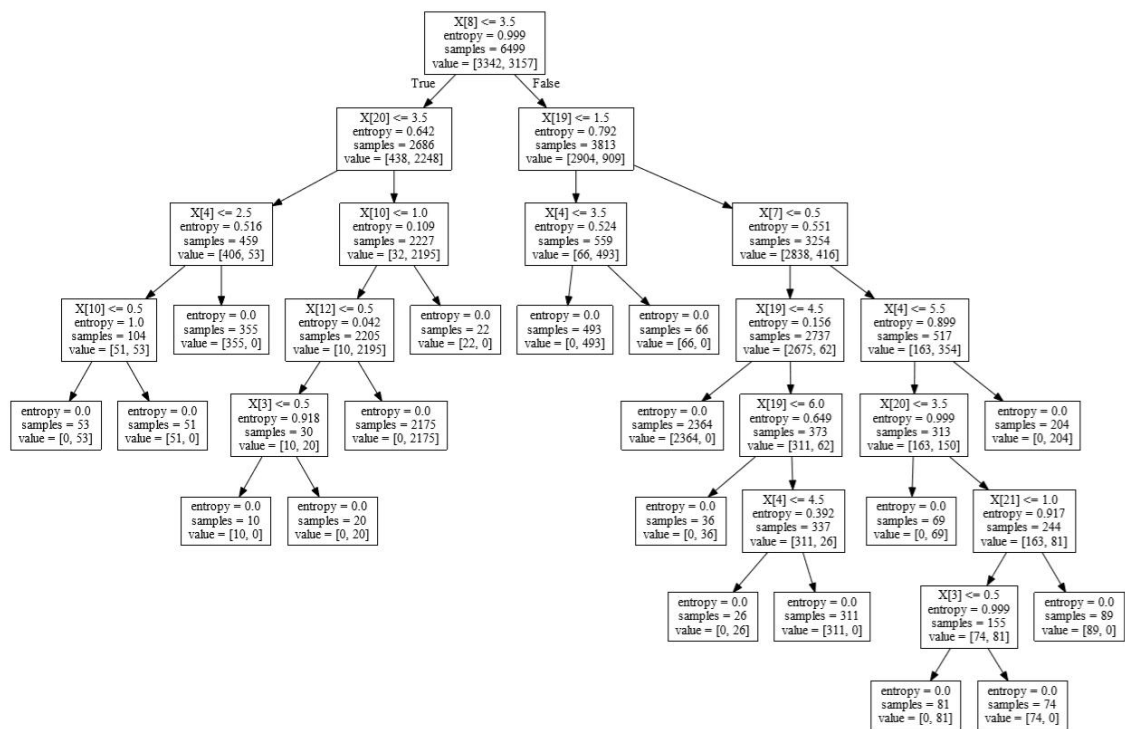
```

Kompletny kod źródłowy można znaleźć pod linkiem:

https://github.com/pmryzak/MushroomClassifier?fbclid=IwAR0VxRkYhfE5HXWF-gZ4CTE3GelAm64epnuGsdtn_Mq-YFxXXGrClePzPDs.



Graficzna wizualizacja powstałego drzewa (podejście z usuwaniem wierszy z brakującymi danymi).



4. Oceniania jakości uzyskanego drzewa.

Dla modelu wytrenowanego na danych testowych, w których brakujące wartości zastąpiono najczęstszymi.

```
In [38]: #wynik modelu danych uzupełnionych na swoich danych testowych
filled_data_model.score(x_test1, y_test1)
```

```
Out[38]: 1.0
```

```
In [39]: #wynik modelu danych uzupełnionych na swoich danych treningowych
filled_data_model.score(x_train1, y_train1)
```

```
Out[39]: 1.0
```

```
In [40]: #wynik modelu danych uzupełnionych na danych testowych drugiego modelu
filled_data_model.score(x_test2, y_test2)
```

```
Out[40]: 0.8423383525243578
```

Dla modelu wytrenowanego na danych testowych, usunięto wiersze z brakującymi wartościami.

```
In [42]: #wynik modelu danych usuniętych na swoich danych testowych
dropped_data_model.score(x_test2, y_test2)
```

```
Out[42]: 1.0
```

```
In [43]: #wynik modelu danych usuniętych na swoich danych treningowych
dropped_data_model.score(x_train2, y_train2)
```

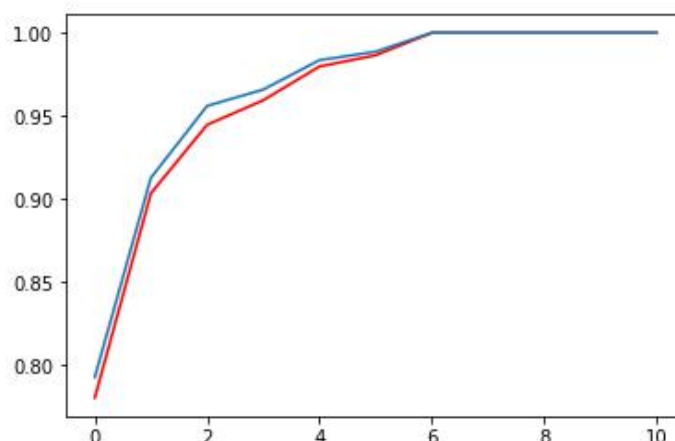
```
Out[43]: 1.0
```

```
In [44]: #wynik modelu danych usuniętych na danych testowych drugiego modelu
dropped_data_model.score(x_test1, y_test1)
```

```
Out[44]: 0.5655384615384615
```

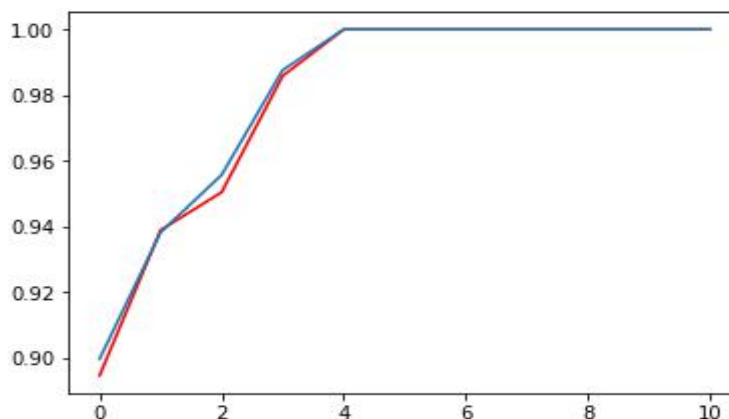
5. Analiza zależności wyników modelu od głębokości drzewa.

a) Zależność skuteczności predykcji modelu od głębokości drzewa dla modelu wytrenowanego na danych w których brakujące wartości zastąpiono najczęściej występującymi przedstawia poniższy wykres.



Możemy zaobserwować że od głębokości drzewa równej 6 model posiada 100% skuteczność przewidywania na swoich danych testowych.

b) Zależność modelu od głębokości drzewa dla modelu wytrenowanego na danych w których usunięto wiersze z brakującymi wartościami.



Obserwujemy, że model już przy głębokości drzewa równej 4 osiąga skuteczność przewidywania blisko 100%. W porównaniu do pierwszego podejścia jest to zdecydowanie mniejsza głębokość drzewa, spowodowane jest to prawdopodobnie mniejszą ilością danych trenujących przez co drzewo decyzyjne nie potrzebuje podejmować, aż tylu decyzji co w pierwszym podejściu.

6. Źródła

1. Link do Githuba:
<https://github.com/pmryzak/MushroomClassifier/blob/master/%5BKWD%20Projekt%5D%201.1%20Rafa%C5%82%20Paprocki%3B%201.2%20Piotr%20Ryzak.ipynb>
2. Dane trenujące: https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.names?fbclid=IwAR1QDnL7ambup7_ubYPuYIF0p6pDV7le8z_P1bCPTDHbpa5retXD-vHSJBU
3. Informacje o drzewach decyzyjnych i algorytmie ID3:
<http://www.is.umk.pl/~duch/Wyklady/CIS/Prace%20zalicz/08-Bujak.pdf>
<https://www.ii.pwr.edu.pl/~kwasnicka/tekstystudenckie/apw/id3.htm>
<http://home.agh.edu.pl/~pmarynow/pliki/iwmet/drzewa.pdf>