3

KLASYFIKACJA

DRZEWA DECYZYJNE, KNN, NAIVE BAYES

KLASYFIKACJA I PODZIAŁ DATASETU

Na laboratoriach przetestujemy różne algorytmy klasyfikujące. Sprawdzimy czy uda się nam sklasyfikować irysy po gatunku, biorąc pod uwagę ich pomiary. Zobaczymy też czy lepiej klasyfikujemy ręcznie (my ludzie, zad 1) czy jednak AI (zadanie 2 i 3).

Zanim jednak przejdziemy do klasyfikacji, musimy wiedzieć jak dzielić zbiór na zbiór treningowy i testowy. Przejrzyj i uruchom poniższe wstawki pythonowe.

a) Użyj paczki pandas do wczytania tego pliku do Pythona i wyświetl bazę danych.

```
import pandas as pd

df = pd.read_csv("iris.csv")

print(df)
```

b) Pora na podział zbioru na zbiór testowy i treningowy. Wykorzystamy do tego funkcję train_test_split z paczki sklearn. Dodaj na górze programu odpowiedni import:

```
from sklearn.model_selection import train_test_split
```

Następnie dopisz na dole programu funckję dzielącą zbiór df.values na dwa kawałki w proporcjach 70% i 30%, w losowy sposób.

```
#podzial na zbior testowy (30%) i treningowy (70%), ziarno losowosci = 13
(train_set, test_set) = train_test_split(df.values, train_size=0.7,
random_state=13)
```

c) Sprawdź jakie rekordy znalazły się w zbiorze testowym i ile ma on rekordów.

```
print(test_set)
print(test_set.shape[0])
```

d) Bywa, że zbiór testowy i treningowy trzeba podzielić dodatkowo na inputy (kolumny numeryczne) i klasy (odmiany irysów). Wówczas z 2 zbiorów danych zrobią się nam cztery.

```
train_inputs = train_set[:, 0:4]
train_classes = train_set[:, 4]
test_inputs = test_set[:, 0:4]
test_classes = test_set[:, 4]
```

Sprawdź zawartość czterech zbiorów.

Grzegorz Madejski
grzegorz.madejski@ug.edu.pl
Zakład Sztucznej Inteligencji,
Instytut Informatyki, Uniwersytet Gdański

💻 ZADANIE 1: **KLASYFIKACJA PRZEZ CZŁOWIEKA**

Naszym zadaniem jest napisanie prostego klasyfikatora dla bazy irysów i zewaluowanie go. Klasyfikator skonstruujemy i zewaluujemy ręcznie, bez użycia AI. W zadaniu 2 skorzstamy z kolei z metod AI i porównamy czy lepszy wynik mają ludzie (Ty) czy maszyny.

Rozpocznij od przygotowania zbioru treningowego i testowego, tak jak w poprzednim zadaniu.

```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv("iris.csv")

(train_set, test_set) = train_test_split(df.values, train_size=0.7,
random_state=13)
```

Random state ustaw na numer swojego indeksu (zamień 13 na inny numer).

a) Napiszemy w Pythonie funkcję, która na podstawie czterech numerycznych parametrów irysa odgadnie jego gatunek wykorzystując do tego jedynie dwie instrukcje if, elif oraz else (jednokrotnie).

```
def classify_iris(sl, sw, pl, pw):
    if sl > 4:
        return("Setosa")
    elif pl <= 5:
        return("Virginica")
    else:
        return("Versicolor")
```

Widać, że funkcja wykorzystuje sepal.length i petal.length, sprawdzając ich wartości. Na tej podstawie funkcja decyduje, jaką odmianę ma irys. Funkcja ta działa dla jednego irysa.

b) Chcemy teraz tę funkcję uruchomić dla wszystkich irysów ze zbioru testowego i zliczyć ile razy dobrze zgadnie odmianę irysa. Zrobimy to w dość prostu sposób – wykorzystując pętlę for. Schemat działania:

Dla każdego irysa ze zbioru testowego: sprawdź czy odpowiedź, którą zwraca funkcja classify_iris, jest taka sama jak prawdziwa odmiana irysa z ostatniej kolumny. Jeśli tak, to dodaj ten rekord do licznika dobrych odpowiedzi. Na końcu podaj ile rekordów funkcja classify_iris zgadła (liczbowo i procentowo).

Przełóżmy to na kod Pythonowy. Uwaga! Poniższy kod ma dwa miejsca ze znakami zapytania, które trzeba uzupełnić odpowiednimi fragmentami kodu!

```
good_predictions = 0
len = test_set.shape[0]

for i in range(len):
    if classify_iris(????) == test_set[????]:
        good_predictions = good_predictions + 1

print(good_predictions)
print(good_predictions/len*100, "%")
```

Jaka odpowiedź wyszła? U mnie niestety kiepska 🙁 14 zgadniętych irysów (na 45 możliwych), co dało około 31%.

Grzegorz Madejski
grzegorz.madejski@ug.edu.pl
Zakład Sztucznej Inteligencji,
Instytut Informatyki, Uniwersytet Gdański

c) Pora na poprawienie funkcji classify_iris, tak aby działała lepiej (Kto kupi nasz program, jeśli działa beznadziejnie?). Wyświetl zbiór treningowy np. poprzez...

```
print(train_set)
```

(możesz dla ułatwienia też go posortować wg gatunku – jak to zrobić?)
...i przyglądając się mu, postaraj się znaleźć jakieś fajne zależności między pomiarami a odmianami irysa (użyj jedynie oczu i mózgu! 😉)
Nie zaglądaj do zbioru testowego!

Twoim zadaniem jest poprawić dwa warunki w ifach w funkcji classify iris, tak aby ta funkcja działała dobrze. Reszty funkcji nie modyfikuj.

```
def classify_iris(sl, sw, pl, pw):
    if ?????:
        return("Setosa")
    elif ?????:
        return("Virginica")
    else:
        return("Versicolor")
```

Zadanie możesz uznać za zakończone, gdy procent poprawnie odgadniętych odpowiedzi, który nazywamy dokładnością klasyfikatora (ang. accuracy), będzie wynosił więcej niż 80%.

Czy uda Ci się przekroczyć 90%?

💻 ZADANIE 2: **DRZEWA DECYZYJNE**

W poprzednim zadaniu, w funkcji classify_iris, stworzyliśmy małe binarne drzewo decyzyjne postaci:

Musieliśmy jednak stworzyć to drzewo sami. Są jednak algorytmy, takie jak ID3 czy C4.5, które tworzą takie drzewa automatycznie i to z o wiele większą precyzją niż człowiek. W Pythonie można skorzystać z paczki sklearn (tree). Należy skorzystać ze źródeł internetowych, aby rozwiązać to zadanie, np.

- https://scikit-learn.org/stable/modules/tree.html, lub
- https://medium.com/@haydar_ai/learning-data-science-day-21-decision-tree-on-iris-dataset-267f3219a7fa

Wykorzystując wiedzę z samouczków i poprzednich zadań wykonaj następujące polecenia.

a) Podziel w losowy sposób bazę danych irysów na zbiór treningowy i zbiór testowy w proporcjach 70%/30%. Wyświetl oba zbiory. Podziel te zbiory na cztery części (inputy i class), jeśli jest taka potrzeba.
b) Zainicjuj drzewo decyzyjne metodą DecisionTreeClassifier.
c) Wytrenuj drzewo decyzyjne na zbiorze treningowym, wykorzystując funkcję fit.

Grzegorz Madejski
grzegorz.madejski@ug.edu.pl
Zakład Sztucznej Inteligencji,
Instytut Informatyki, Uniwersytet Gdański

d) Wyświetl drzewo w formie tekstowej i/lub w formie graficznej. Jeśli masz problemy z tym podpunktem – pomiń go.
e) Dokonaj ewaluacji klasyfikatora: sprawdź jak drzewo poradzi sobie z rekordami ze zbioru testowego. Wyświetl dokładność klasyfikatora, czyli procent poprawnych odpowiedzi. Wykorzystaj do tego funkcję score lub predict.
f) Wyświetl macierz błędów (ang. confusion matrix), która zestawia liczby błędnych i poprawnych odpowiedzi, dla wszystkich klas.

Czy drzewo nauczone automatycznie jest lepsze niż Twoje drzewko stworzone ręcznie w zadaniu 2? 😉 Kto wygrał Ty czy AI?

💻 ZADANIE 3: **INNE KLASYFIKATORY**

Dla zbioru danych z irysami przeprowadź klasyfikację metodą k-najbliższych sąsiadów dla kilku przypadków:

- k-NN, k=3
- k-NN, k=5
- k-NN, k=11

oraz klasyfikatorem Naive Bayes.

W rozwiązaniu zadania uwzględnij następujące punkty:

a) Podziel w losowy sposób bazę danych na zbiór treningowy (70%) i testowy (30%).
b) Uruchom każdy z klasyfikatorów wykorzystując paczki i dokonaj ewaluacji ma zbiorze testowym wyświetlając procentową dokładność i macierz błędu. Przydatne linki:
KNN:
   - https://towardsdatascience.com/k-nearest-neighbor-python-2fccc47d2a55
   - https://scikit-learn.org/stable/modules/neighbors.html
   - https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/
   - https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75

   NaiveBayes:
   - https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn
   - https://scikit-learn.org/stable/modules/naive_bayes.html

c) Porównaj wszystkie 5 klasyfikatorów (DD, 3NN, 5NN, 11NN, NB) pod względem dokładności. Który z nich jest najlepszy?

Grzegorz Madejski
grzegorz.madejski@ug.edu.pl
Zakład Sztucznej Inteligencji,
Instytut Informatyki, Uniwersytet Gdański