Laboratorium 3

Klasyfikacja

Baza danych o irysach

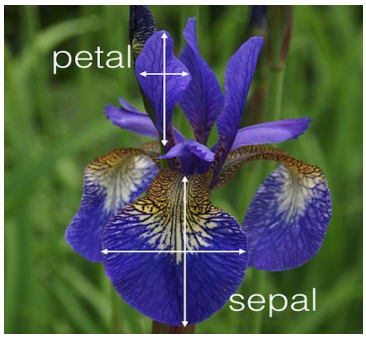
Na tym laboratorium będziemy działać na zbiorach danych, które są powszechnie dostępne w Internecie i dobrze się na nich ćwiczy uczenie maszynowe.

Pierwsza baza danych iris.csv, która zawiera dane o trzech gatunkach irysa.

Iris setosa

Iris versicolor

Iris virginica



Każdemu z trzech gatunków mierzono długość i szerokość płatka „petal" i działki kielicha kwiatu „sepal" (to te większe płatki skierowane w dół).

Mamy więc 4 atrybuty o wartościach rzeczywistych dla każdego kwiatu. Dodatkowo w piątej kolumnie podana jest jego odmiana (gatunek) w forma łańcucha znaków.

	Sepal length	Sepal width	Petal length	Petal width	Class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
⋮	⋮	⋮	⋮	⋮	⋮
150	5.9	3.0	5.1	1.8	virginica

Dziś zajmiemy się tematem klasyfikacji. Chcemy wyręczyć botaników w rozpoznawaniu kwiatów, więc piszemy program ekspercki, który na podstawie 4 parametrów numerycznych kwiatu odgadnie jego gatunek.

Zadanie 1

Naszym pierwszym zadaniem jest napisanie prostego klasyfikatora dla bazy irysów i zewaluowanie go. Bazę irysów można ściągnąć ze strony zajęć. W Pythonie można pobrać ją również z paczki sklearn.

a) Napisz w Pythonie funkcję, która na podstawie czterech numerycznych parametrów irysa odgadnie jego gatunek wykorzystując do tego jedynie dwie instrukcje if-else. Schemat pseudokodowy:

```
classify_iris(sl,sw,pl,pw) {
  if (...) {
    return(...)
  } else {
     if (...) {
          return(...)
     } else {
          return(...)
     }
  }
}
```

W miejsce kropek należy wpisać warunki na sl,sw,pl,pw np. (sl>3.5 lub podobne), a `return` musi dawać jedną z wartości: versicolor, virginica, setosa. Jakie warunki wpisać? Po prostu spróbuj zgadnąć patrząc na bazę danych. Może masz wprawne oko i uda ci się dostrzec jakieś zależności w danych.
Zauważ, że jak wyciągasz wnioski na podstawie wszystkich 150 rekordów, to zbiór treningowy to cała baza danych.

b) Teraz dokonamy ewaluacji. Napisz funkcję lub fragment kodu, który przejdzie po wszystkich 150 rekordach (zbiór testowy to też cała baza danych!), dla każdego uruchomi classify_iris i sprawdzi czy klasa przewidziana zgadza się z rzeczywistą. Na koniec zwróci procent poprawnie odgadniętych odpowiedzi z wszystkich rekordów np. „140/150 rekordów, 93%”. Pomysł na kod (schemat):

```
zgodnosc = 0
For i=1 to 150:
     If (myPredictRow == iris_klasa) then zgodnosc++
Print(zgodnosc/150)
```

c) Czy Twój klasyfikator działa bardzo dobrze (>90%), dobrze (>80%) czy tylko średnio (>60%)? A może działa równie efektywnie jak ślepe wybieranie (ok. 33%)? :-P
Porównaj swoje wyniki z wynikami kolegów i koleżanek.

Zadanie 2

W poprzednim zadaniu 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): 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 wykonaj następujące polecenia.

a) Podziel w losowy sposób bazę danych irysów na zbiór treningowy i zbiór testowy w proporcjach 70%/30%. Wyświetl oba zbiory.
b) Wytrenuj drzewo decyzyjne na zbiorze treningowym.
c) Wyświetl drzewo w formie tekstowej i w formie graficznej.
d) Dokonaj ewaluacji klasyfikatora: sprawdź jak drzewo poradzi sobie z rekordami ze zbioru testowego. Wyświetl procent poprawnych odpowiedzi.
e) Wyświetl macierz błędu (confusion matrix) dla tej ewaluacji. Wyjaśnij jakie błędy popełniał klasyfikator wskazując na liczby w macierzy błędu.

Zadanie 3

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

W rozwiązaniu zadania uwzględnij następujące punkty:

a) Podziel w losowy sposób bazę danych na zbiór treningowy (67%) i testowy (33%).

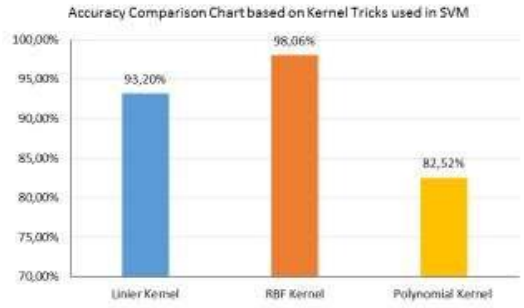
b) Uruchom każdy z klasyfikatorów wykorzystując paczki i dokonaj ewaluacji ma zbiorze testowym wyświetlając procentową dokładność i macierz błędu.
Przydatne linki:
https://towardsdatascience.com/k-nearest-neighbor-python-2fccc47d2a55
https://scikit-learn.org/stable/modules/neighbors.html
https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/
https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75

c) Jakie *k* daje najlepszy wynik?

Zadanie 4

Powtórz eksperyment z zadania 2 i 3 (klasyfikatory: drzewo, 1NN, 3NN, 5NN, 7NN, 11 NN) dla innego zbioru danych: diabetes.csv (załączony plik). Tutaj klasyfikacja polega na diagnozowaniu cukrzycy (u kobiet pochodzących z rdzennych plemion w Ameryce). Zgadujemy czy osoba jest chora, czy zdrowia, patrząc na parametry jej organizmu (wagę, parametry krwi, liczbę ciąż, itp.).

Dokładności wszystkich klasyfikatorów zestaw na wykresie słupkowym, coś typu:



Pytanie dodatkowe: chcemy zminimalizować błędy, gdy klasyfikator chore osoby klasyfikuje jako zdrowe (i odsyła do domu bez leków). Który z klasyfikatorów najbardziej się do tego nadaje?

Po zakończeniu rozwiązywania zadań

Proszę pamiętać, by rozwiązania zadań i raport z wykonania dodać do repozytorium. Najlepiej w osobnym folderze.