Ćwiczenie nr 7
Wprowadzenie do sztucznej inteligencji

Autor
Maciej Prostak

1 Wprowadzenie

Tematem zadania jest klasyfikacja za pomocą modelu Bayesowskiego. Zostanie zaimplementowany naiwny klasyfikator Bayesa, który następnie zostanie użyty do treningu i walidacji stworzonego modelu.

1.1 Technologia

Rozwiązanie zostało zaimplementowane w języku Python (3.9.0) z wykorzystaniem biblioteki pandas (1.3.4) oraz numpy (1.12.4). Dodatkowo dla poprawy czytelności kodu zostały dołączone biblioteki black (21.9b0), flake8 (4.0.1) oraz isort (5.9.3).

1.2 Zbiór danych

Dane składają się z 4 atrybutów (”sepal length”, ”sepal width”, ”petal length”, ”petal width”) i ostatecznej przydzielonej klasy ”class”. Wynikowe klasy to ”Iris-setosa”, ”Iris-versicolor”, ”Iris-virginica”. Wszystkie atrybuty (oprócz ”class”) są danymi ciągłymi.

1.3 Analiza danych

Na wykresie poniżej zostały wyrysowane wartości dwóch atrybutów z wynikową klasą. Jest to analiza całego zbioru danych. Widzimy, że dane są prawie liniowo separowalne. Możliwe, że używając tylko tych dwóch atrybutów można by było uzyskać już dobry model.

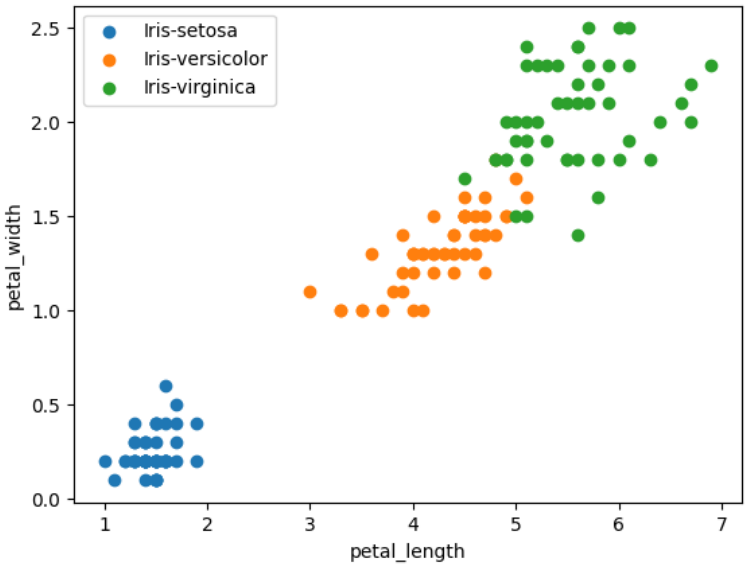

Fig. 1 Analiza danych dla atrybutów ”petal width” oraz ”petal length”.

2 Wyniki

2.1 Wyniki w przypadku losowego mieszania danych

W przypadku pomieszania danych okazało się, że niepotrzebna jest duża liczba danych treningowych. Nawet w przypadku wylosowania danych treningowych w proporcji 20/80 z całego zbioru otrzymaliśmy bardzo obiecujące wyniki. Jednak tym więcej było danych, na których model mógł się wzorować, tym lepsze wyniki. W przypadku jednej próby otrzymaliśmy stu-procentową skuteczność na danych walidacyjnych (mógł być to oczywiście przypadek). Zostały również umieszczone próby przy podziale 60/40.

2.1.1 Podział danych na treningowe / walidacyjne (80/20)

	accuracy	f1-score	fallout	precision	recall
value	0.95550	0.93330	0.03330	0.93330	0.93330

Fig. 2 Pomiary dla pomieszanych danych przy podziale na treningowe / walidacyjne (80/20).

	accuracy	f1-score	fallout	precision	recall
value	1.00000	1.00000	0.00000	1.00000	1.00000

Fig. 3 Pomiary dla pomieszanych danych przy podziale na treningowe / walidacyjne (80/20).

	accuracy	f1-score	fallout	precision	recall
value	0.97770	0.96660	0.01660	0.96660	0.96660

Fig. 4 Pomiary dla pomieszanych danych przy podziale na treningowe / walidacyjne (80/20).

2.1.2 Podział danych na treningowe / walidacyjne (60/40)

	accuracy	f1-score	fallout	precision	recall
value	0.96660	0.95000	0.02500	0.95000	0.95000

Fig. 5 Pomiary dla pomieszanych danych przy podziale na treningowe / walidacyjne (60/40).

	accuracy	f1-score	fallout	precision	recall
value	0.95550	0.93330	0.03330	0.93330	0.93330

Fig. 6 Pomiary dla pomieszanych danych przy podziale na treningowe / walidacyjne (60/40).

2.1.3 Podział danych na treningowe / walidacyjne (20/80)

	accuracy	f1-score	fallout	precision	recall
value	0.93880	0.90830	0.04580	0.90830	0.90830

Fig. 7 Pomiary dla pomieszanych danych przy podziale na treningowe / walidacyjne (20/80).

	accuracy	f1-score	fallout	precision	recall
value	0.94440	0.91660	0.04160	0.91660	0.91660

Fig. 8 Pomiary dla pomieszanych danych przy podziale na treningowe / walidacyjne (20/80).

2.2 Wyniki bez losowego mieszania danych

Podany zbiór jest od razu posortowany względem atrybutu “class”. Widzimy, że w przypadku podziału 80/20 sortowanie nie ma dużego znaczenia. Oznacza to, że w danych treningowych pojawiły się wszystkie wynikowe klasy i wpływają na ostateczne wyniki w testach.

Jednak w przypadku podziałów 50/50 oraz 30/70 widać duży spadek jakości modelu. Jest to dość oczywiste jako, że w danych treningowych w ogóle nie pojawiały się przykłady niektórych klas.

2.2.1 Podział danych na treningowe / walidacyjne (80/20)

	accuracy	f1-score	fallout	precision	recall
value	0.93330	0.90000	0.05000	0.90000	0.90000

Fig. 9 Pomiary dla niepomieszanych danych przy podziale na treningowe / walidacyjne (80/20).

2.2.2 Podział danych na treningowe / walidacyjne (50/50)

	accuracy	f1-score	fallout	precision	recall
value	0.54660	0.32000	0.34000	0.32000	0.32000

Fig. 10 Pomiary dla niepomieszanych danych przy podziale na treningowe / walidacyjne (50/50).

2.2.3 Podział danych na treningowe / walidacyjne (30/70)

	accuracy	f1-score	fallout	precision	recall
value	0.36500	0.04760	0.47610	0.04760	0.04760

Fig. 11 Pomiary dla niepomieszanych danych przy podziale na treningowe / walidacyjne (30/70).

2.3 Confusion Matrix

W momencie robienia predykcji na danych testowych, będziemy odpowiednio aktualizować macierz pomyłek. Przykład wyglądu macierzy został zwizualizowany na rysunku poniżej. Można zauważyć, że najwięcej wartości znajduję się w polach znajdujących się na diagonalnej macierzy. Oznacza to, że model w dużej ilości poprawnie sklasyfikował ostateczną klasę.

	0	1	2
0	11	0	0
1	0	6	0
2	0	1	12

Fig. 12 Przykładowa macierz pomyłek.

3 Wnioski

Gaussian Naive Bayes służący do klasyfikowania danych na podstawie danych ciągłych okazał się dość prosty w implementacji. Za pomocą nawet małej ilości danych treningowych byliśmy w stanie zbudować przydatny model. Jednak przy innych problemach ten algorytm mógłby okazać się niewystarczający. Przeglądając pierwotną tabelę danych można było gołym okiem zobaczyć różnice poszczególnych atrybutów ze względu na klasę.