Raport
Wprowadzenie do Sztucznej Inteligencji – Ćwiczenie 7.

Modele bayesowskie

Autor: Aleksandra Jamróz, nr albumu: 310 708

Treść zadania

Należy zaimplementować naiwny klasyfikator bayesowski bez użycia dodatkowych bibliotek i zastosować go do zbadania załączonego zbioru danych. W moim przypadku były to dane dotyczące chemicznych parametrów trzech odmian wina rosnącego w tym samym rejonie Włoch. Link: https://archive.ics.uci.edu/ml/datasets/Wine.

Przygotowanie danych

Do pracy nad danymi wykorzystałam bibliotekę pandas. Do tabeli z danymi wczytuję zawartość pliku *wine.data*. Dodaję do niej nagłówki, które mają w zasadzie jedynie wartość estetyczną. Kolumna określająca klasę wina jest pierwszą kolumną w tabeli. Jeżeli nie określiłabym nagłówków na stałe, do programu możnaby było załadować dowolny plik, o ile miałby taki sam format, a klasy były w pierwszej kolumnie.

Obliczenie statystyk

W celu obliczenia wartości potrzebnych do późniejszego klasyfikowania wina, musiałam zaimplementować funkcje odpowiadające za kolejne kroki:

1. *split_by_class* – podzielenie całego zbioru danych na podzbiory względem klasy. W ten sposób otrzymujemy tyle podzbiorów, ile występuje różnych klas. Funkcja automatycznie usuwa kolumnę zawierającą klasy z każdego z podzbiorów. Zwracana jest lista z podzbiorami oraz lista klas. Kolejność pozbiorów odpowiada kolejności listy klas, dzięki czemu możemy dopasować zbiór do klasy poprzez podanie jednakowego indeksu.
2. *stats_for_column* – przyjmuje jako parametr listę wartości. Powstała do liczenia statystyk dla danej kolumny w zbiorze. Na podstawie otrzymanych danych liczy średnią, odchylenie standardowe oraz ich liczbę, co zwraca w postaci listy.
3. *stats_for_dataset* – dla każdej kolumny występującej w podanym zbiorze liczy statystyki za pomocą funkcji *stats_for_column*. Zwraca listę statystyk o długości równej liczbie kolumn. Zakładamy, że w tabeli podawanej jako parametr tej funkcji nie ma już kolumny z klasami, ponieważ nie chcemy liczyć statystyk z wartości klas.
4. *Stats_for_classes* – łączy funkcje *split_by_class* oraz *stats_for_dataset* licząc statystyki każdego podzbioru powstałego ze zbioru głównego.

Obliczanie prawdopodobieństwa

Klasyfikacja za pomocą modelu bayesowskiego polega na obliczeniu prawdopodobieństwa, z jakim podane dane można przypisać do danej klasy. Liczy się to wykorzystując prawdopodobieństwo warunkowe. Wartość prawdopodobieństwa dla danej klasy inicjujemy prawdopodobieństwem jej wystąpienia w zbiorze danych w ogóle (dzielimy liczbę jej wystąpień przez liczbę wszystkich rzędów w tabeli). Następnie mnożymy ją po kolei przez wartości otrzymane ze wzoru:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

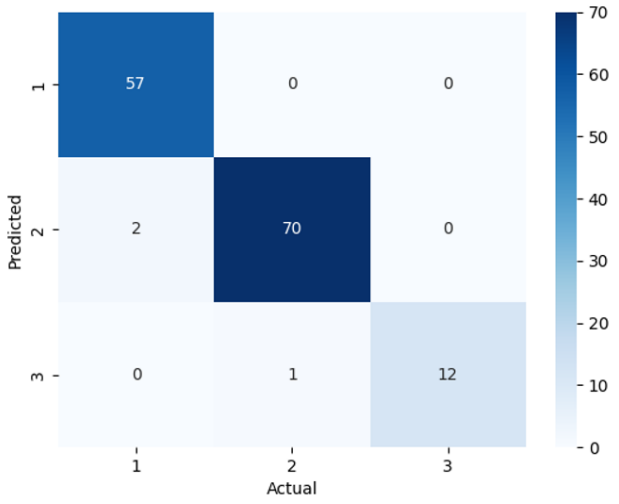
dla statystyk wszystkich pozostałych kolumn. Wykorzystujemy tu statystyki z funkcji *stats_for_classes.*
Funkcja *predict_probabilities* oblicza listę prawdopodobieństw dla podanej listy danych. Funkcja *predict_class* wybiera największe otrzymane prawdopodobieństwo i łączy je z nazwą klasy.

Klasyfikacja dla zbiorów danych

Wszystkie wyżej wymienione funkcje zostają połączone w funkcji *predict_dataset.* Jako argumenty przyjmuje 2 zbiory danych: treningowy, na podstawie którego liczy statystyki dla klas, oraz testowy. Kolejne wiersze zbioru testowego dzielone są na wartości poddane klasyfikacji oraz oczekiwane klasy. Lista przewidywanych klas oraz oczekiwanych klas jest wynikiem działania funkcji.

Macierz pomyłek

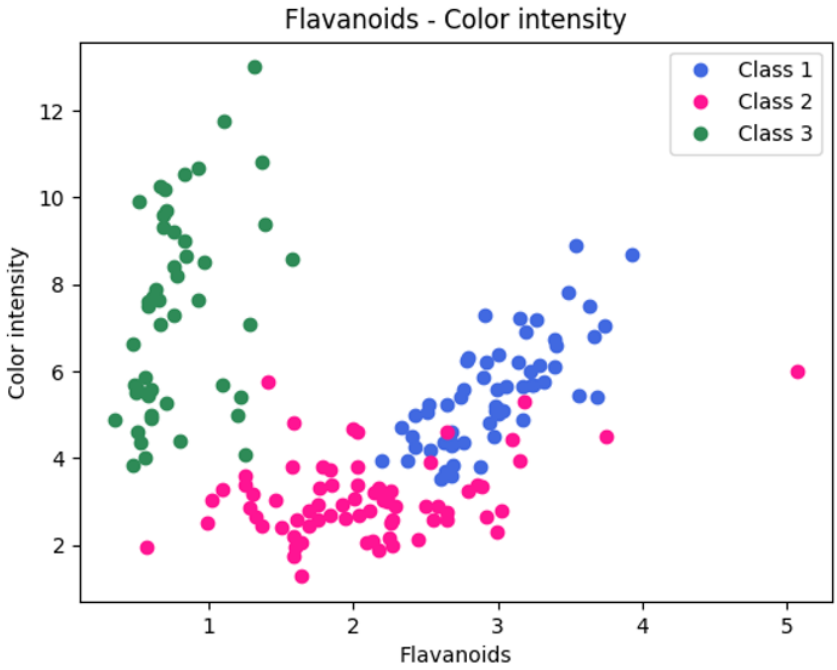
Na podstawie listy przewidywanych klas i listy oczekiwanych klas generuję macierz pomyłek. Wykorzystała do tego bibliotekę pandas, seaborn oraz matplotlib. Tworzona jest macierz wyświetlana jako wykres.



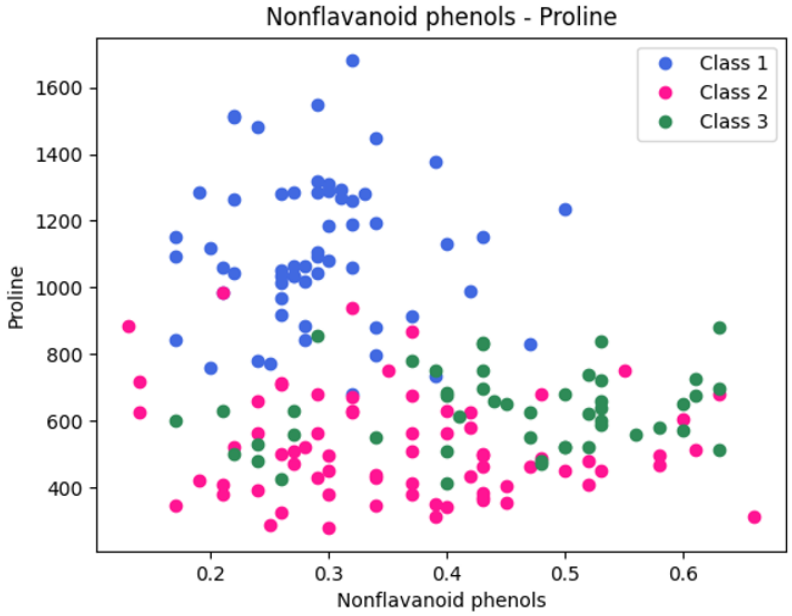
Rysunek 1. Przykładowa macierz pomyłek

Sprawdzenie danych

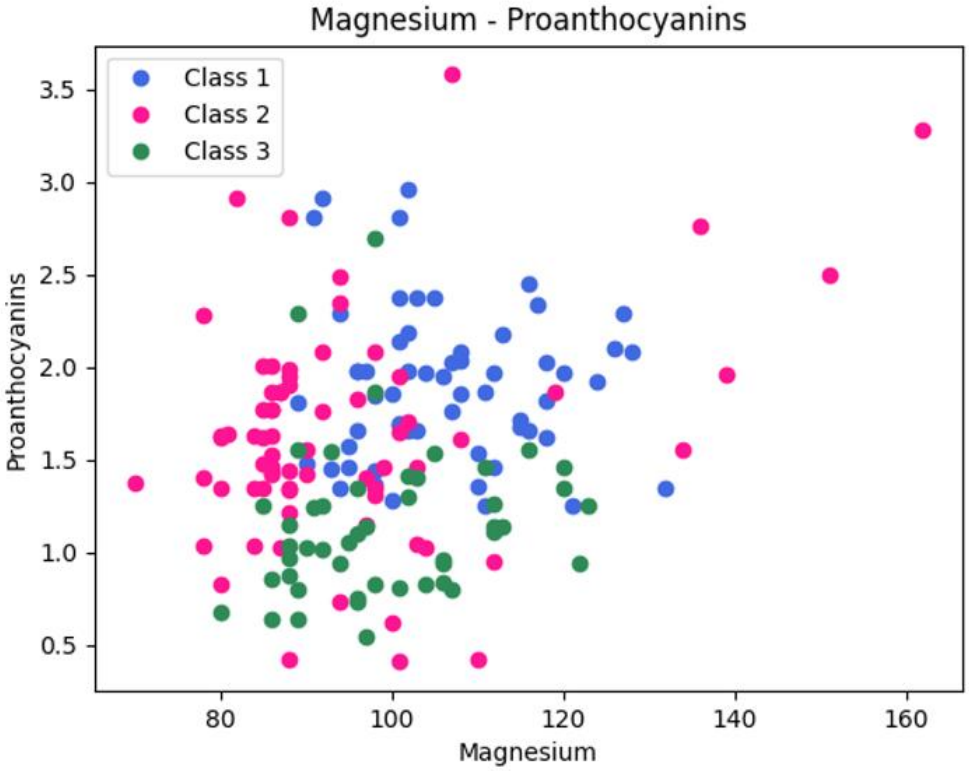
Przeanalizowałam dane na temat win poprzez narysowanie wykresów zależności pomiędzy wartościami różnych kategorii. Dobierałam je w pary na zasadzie każdy z każdym. Zauważyłam, że niektóre z nich umożliwiają podział zbioru na niemal odrębne podgrupy. Inne natomiast zlewały się w jedną plamę, nie pozwalając na wyodrębnienie żadnego z nich. Klasyfikator Bayesa łączy wszystkie kategorie, więc może wykorzystać małe zależności i poprawnie zakwalifikować klasy.



Rysunek 2. Wykres przedstawiający niemalże wyodrębniającą zależność



Rysunek 3. Na tym wykresie klasa 2 i 3 zlewają się, natomiast 1 jest wyodrębniona



Rysunek 4. W tym przypadku wszystkie klasy się zlewają.

Testowanie algorytmu

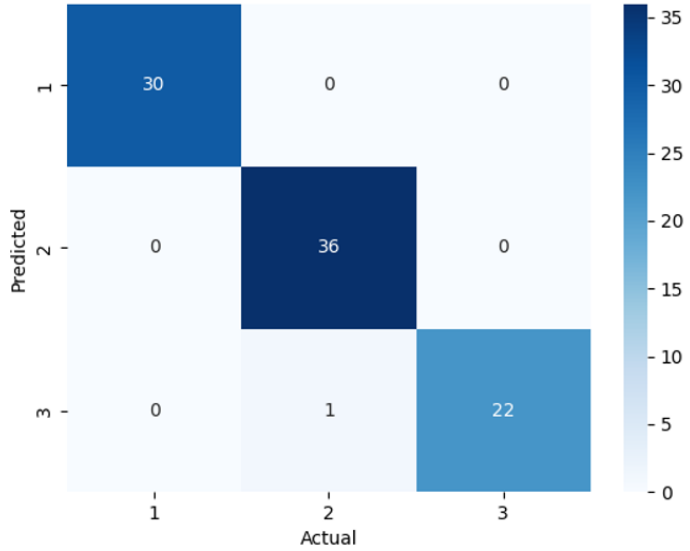
Zdecydowałam się przeprowadzić testy w zależności od dwóch parametrów: proporcji wielkości zbioru testowego i treningowego oraz tego, czy zbiór danych jest początkowo pomieszany. Zmienna shuffled równa 1 oznacza, że zbiór jest pomieszany.

shuffled	test data proportion	recall	fall_out	precision	accuracy	f1_score
1	0,1	1	1	0,991	0,981	0,995
1	0,2	1	1	0,993	0,986	0,996
1	0,3	1	1	0,988	0,976	0,994
1	0,4	1	1	0,986	0,972	0,993
1	0,5	1	1	0,994	0,989	0,997
1	0,6	1	1	0,986	0,972	0,993
1	0,7	1	1	0,990	0,981	0,995
1	0,8	1	1	0,986	0,971	0,993
1	0,9	1	1	0,970	0,941	0,985

Rysunek 5. Wyniki testów dla danych pomieszanych

Możemy zauważyć że klasyfikator osiąga fenomenalne wyniki, jeżeli pracuje na pomieszanych danych. Nawet jeżeli zbiór treningowy ma wielkość 10% całego zbioru, osiągana precyzja

oscyluje na poziomie 97%. Oczywiście im większy zbiór treningowy, tym lepiej wyuczony klasyfikator i tym lepsze osiągi.

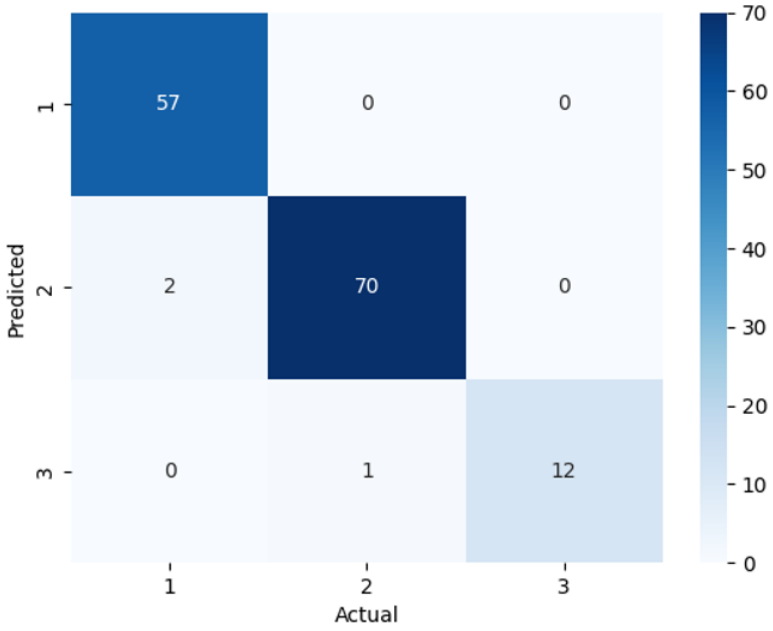


Rysunek 6. Macierz pomyłek dla klasyfikacji danych pomieszanych przy proporcji zbioru testowego do treningowego 1:1

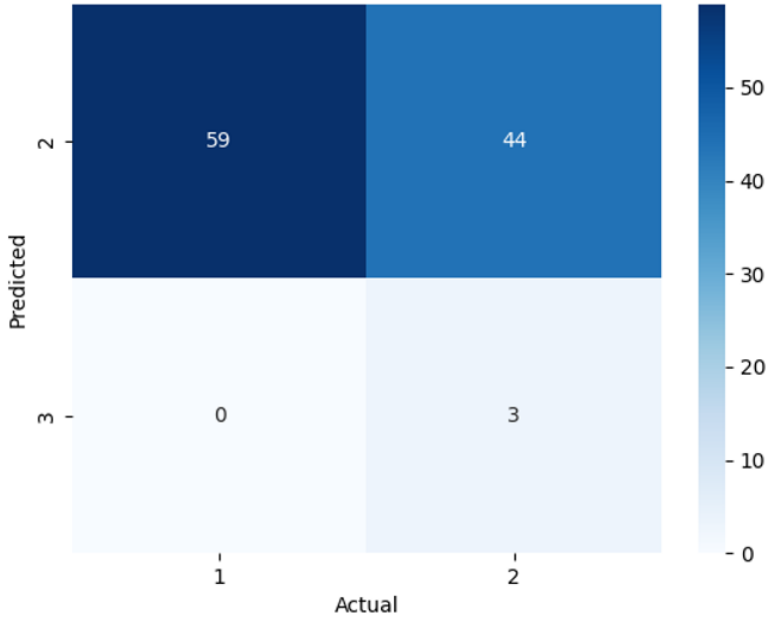
shuffled	test data proportion	recall	fall_out	precision	accuracy	f1_score
0	0,1	1	1	0,994	0,988	0,997
0	0,2	1	1	0,989	0,979	0,995
0	0,3	0,955	0,955	0,907	0,827	0,930
0	0,4	0,952	0,952	0,599	0,429	0,735
0	0,5	0,937	0,937	0,468	0,315	0,625
0	0,6	0,952	0,952	0,234	0,148	0,375
0	0,7	0	0	0	0,5	0
0	0,8	0	0	0	0,5	0
0	0,9	0	0	0	0,5	0

Rysunek 7. Wyniki testów dla danych niepomieszanych

Wyniki są dużo gorsze, jeżeli decydujemy się podać klasyfikatorowi zbiór niepomieszany. Bierzemy pod uwagę, że dane są posortowane względem klas, co znaczy że najpierw mamy wiersze klasy pierwszej, następnie drugiej itd. Do pewnej wielkości zbioru testowego klasyfikator radzi sobie w miarę nieźle. Przy najmniejszym zbiorze testowym ma szansę policzyć statystyki dla wszystkich klas, więc nie ma problemu z klasyfikacją ostatniej klasy. Przy równym podziale zbioru głównego klasyfikator rozpoznaje tylko jedną klasę, ponieważ podczas treningu miał do czynienia z klasą pierwszą oraz drugą, a w zbiorze testowym z drugą i trzecią. Na koniec niemożliwe jest rozpoznanie żadnej z klasy poprawnie, ponieważ klasyfikator nie spotkał się dotychczas z żadną z klas, które ma rozpoznać.



Rysunek 8. Macierz pomyłek klasyfikacji przy pomieszanych danych i proporcji zbioru treningowego do testowego 4:1



Rysunek 9. Macierz pomyłek klasyfikacji przy pomieszanych danych i proporcji zbioru treningowego do testowego 3:2

Na powyższym wykresie mamy przykład klasyfikacji, kiedy w zbiorze treningowym znajdowały się wiersze tylko i wyłącznie klas 2 i 3, a w zbiorze testowym: 2 i 1. Klasyfikator nie ma problemu z rozpoznaniem drugiej klasy. Pierwszej klasy natomiast nigdy nie spotkał, ale wyniki nie pasują mu do klasy numer trzy, więc wiersze pierwszej klasy rozpoznaje jako wiersze drugiej klasy.