# Modele bayesowskie– wsi lab 7

Autor: Patryk Zdziech

Nr.a 311028

Środowisko: Python

Użyte biblioteki: train_test_split from sklearn

## Implementacja

Klasa naive_bayes zawierającą:

Zmienne:

`data` – dane z pliku wejściowego
`train_d, test_d` - zbiór uczący i testujący
`weights` - zbiór wag używanych do aproksymacji

Funkcje:

- `__init__` – na wejściu przyjmuje nazwę pliku z danymi oraz wielkość zbioru testującego w przedziale od 0 do 1. Zbiór uczący i testujący jest wydzielany przez `train_test_split`.
- `train` – trenuje klasyfikator przy użyciu zbioru uczącego
- `predict` – przewiduje klasę obiektu przy użyciu wag klasyfikatora
- `test` – wyświetla zbiór testujący wraz z przewidzianymi przez klasyfikator wynikami i informacją czy są one poprawne
- `accuracy` – trenuje i weryfikuje wyniki na `data`, za każdym razem losując nowy zbiór testowy i uczący. Na koniec wyświetla łączny stosunek sukcesów do liczby wszystkich testów.

Uczenie I aproksymacja

Starałem się podążać za standardowym algorytmem naiwnym Bayesa. W szczególności:
Sumuje cechy poszczególnych klas a następnie normalizuje cechy każdej klasy tak by sumowały się do jedynki w obrębie klasy.
Przy przewidywaniu również noramalizuje wejścia tak by sumowały się do jedynki, by zapobiec dyskryminacji dużych wartości.

Efektywność algorytmu

Eksperymenty przeprowadzono zgodnie z poleceniem na `Iris_data_set`. Po treningu modelu możemy sprawdzić jego działanie funkcjami `test` I `accuracy`. Algorytm osiąga skuteczność powyżej 90% (najczęściej od 92 do 94). Klasyfikator utrzymuje skuteczność nawet jeśli ograniczymy zbiór uczący do 1/5.

Wnioski

Bardzo szybko udało mi się doprowadzić klasyfikator w wersji naiwnej do działania. Wykorzystuje on bardzo uproszczone założenia, mimo to dla prostego problem poradził sobie wyjątkowo dobrze. Parametry bazy danych `Iris_data_set` dobrze separują poszczególne klasy co daje algorytmowi dużą skutezność.