Algorytm VSL – wsi lab 4

Autor: Patryk Zdziech

Nr.a 311028

Środowisko: Python

Użyte biblioteki: pandas , numpy , matplotlib.pyplot, scipy, sklearn

Implementacja

Funkcje:

- `get_data` – czyta z pliku csv i zwraca X i Y.
  - X to przestrzeń wejść
  - Y to przestrzeń dyskretnych wyjść
- `make_meshgrid, plot_contours, plot_svm` – funkcje przydatne do rysowania wykresów

Klasa VSM zawierającą:

Zmienne:

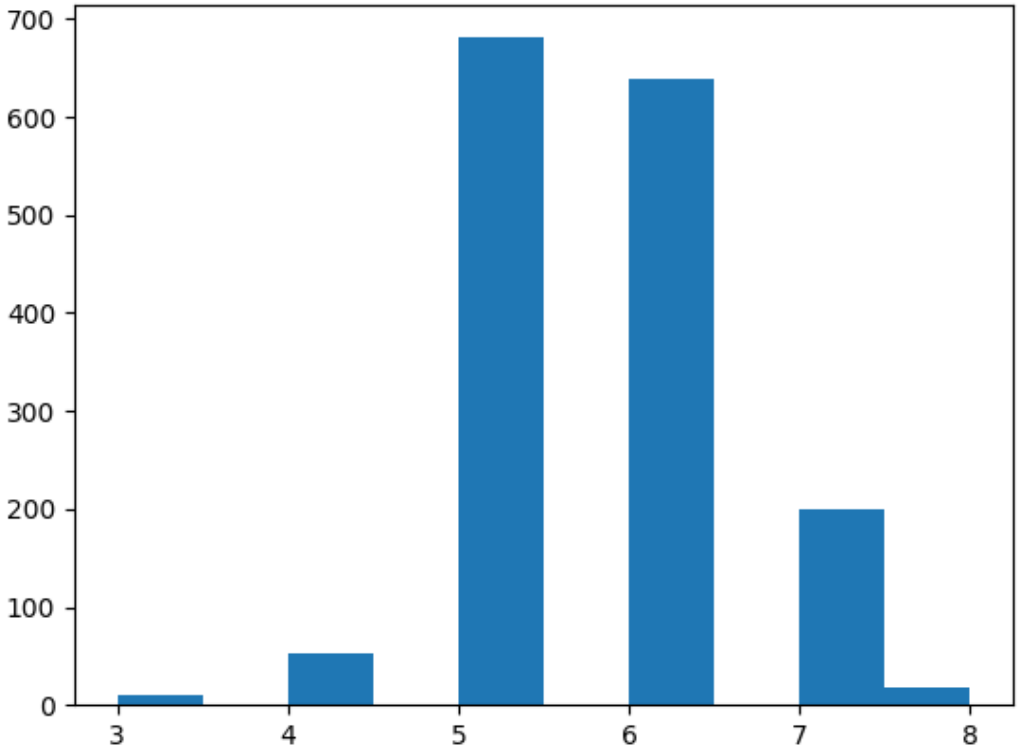
`X_train, Y_train` - zbiór uczący
`X_test, Y_test` - zbiór testujący
`LAMBDA` - parametr λ
`kernel` – obecnie używana funkcja jądrowa
`W` – przestrzeń wag, którą będę optymalizować

Funkcje:

- `__init__` – na wejściu przyjmuje parametr `LAMBDA` oraz zbiory X i Y które są następnie normalizowane przy pomocy biblioteki sklearn i dzielone na zbiór uczący oraz testujący.
- `compute_cost` – funkcja straty którą będę optymalizować
- `linear, poly` – funkcje jądrowe
- `train` – trenowanie modelu przy użyciu algorytmu 'BFGS' z biblioteki scipy
- `test` – testowanie modelu z wypisaniem dokładności względem stanu faktycznego

Przygotowanie danych

Będe korzystać z bazy Wine Quality Data Set. Na podstawie parametrów wina algorytm będzię się starał przewidzieć jego jakość. Ponieważ jednak zmienna quality może przyjmwać wartości od 0 do 10 zdyskretyzuję ją dla lepszego działania algorytmu. Poniższy histogram prezentuje rozkład wartości Y:



Ponieważ zmienna objaśniana powinna mieć zbalansowane proporcje podzielę ją na wartości większę od 5 (wino dobrej jakości) oraz na wartości mniejsze-równe 5(niskiej jakości wino).

Realizacja algorytmu

W implementacji algorytmu będę w szczególności używał jądra liniowego oraz wielomianowego. Dołączyłem również do zbioru przestrzeni wejść X kolumnę o stałej wartości równej jeden. Będzie ona zastępować zmienną ‘b’ w funkcji straty.

Dostosowywanie parametru λ

W celu dobrania odpowiedniego parametru lambda postanowiłem sprawdzić Dokładność algorytmu dla kilku różnych jego wartości:

1 Accuracy: 0.54375

10 Accuracy: 0.54375

0.1 Accuracy: 0.54375

0.025 Accuracy: 0.765625

0.01 Accuracy: 0.740625

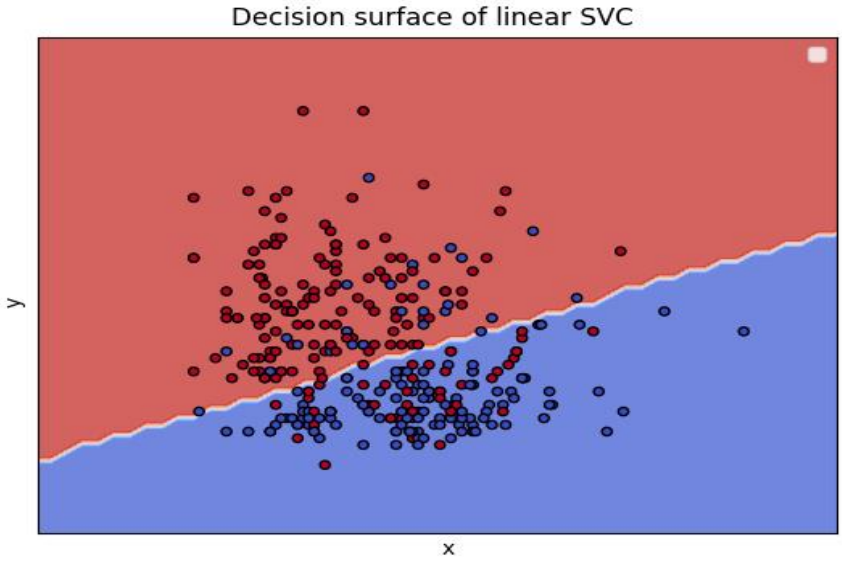
0.001 Accuracy: 0.771875

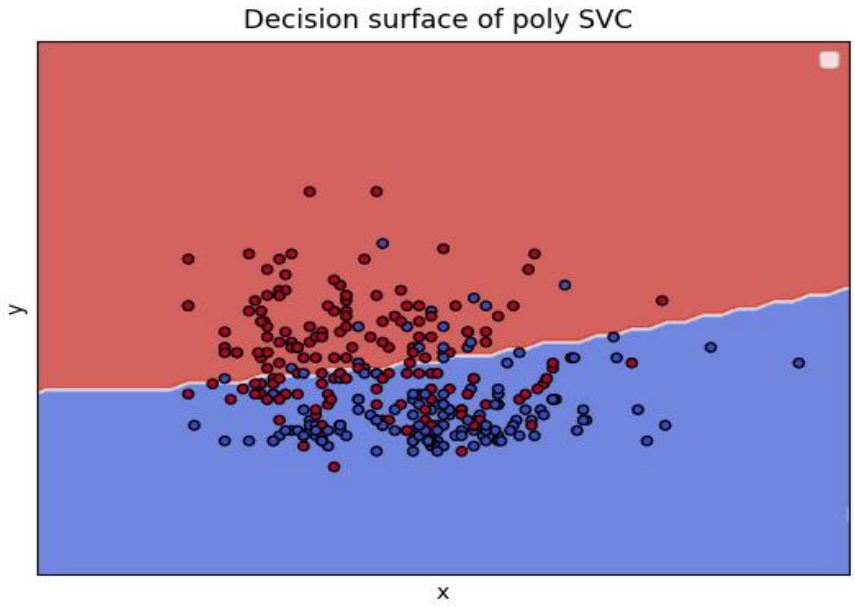
Warto zauważyć że dla wartości większych od 0.025 algorytm w ogóle sobie nie radził z przewidywaniem wartości. Wartość 0.54375 pojawia się trzykrotnie ponieważ jest to proporcja win dobrej jakości w grupie testującej.

Dalsze zmniejszanie parametru lambda nie przyniosło znaczącej poprawy dla tego w dalszych eksperymentach będę przyjmował LAMBDA = 0.01.

Działanie algorytmu dla jądra liniowego i wielomianowego

Przetestowałem jeszcze raz działanie algorytmu dla jądra liniowego i wielomianowego. Wygenerowałem następnie grafy przy użyciu dwóch najsilniej skorelowanych z Y wartości.





Obie funkcje jądrowe przyniosły podobne rezultaty, ich skuteczność przyjmuje wartości bliskie 0.75.

Wnioski

Algorytm svm w rozpatrzonym przypadku osiągnął skuteczność ~75%. Nie jest to zbyt duża dokładność jednak pokazuje że algorytm działa. Możliwe że udało by się polepszyć wyniki algorytmu gdyby:

- przestrzeń wejść posiadałaby więcej parametrów

- zbiór danych trenujących był większy

- użyto bardziej zaawansowanych funkcji jądrowych