Sprawozdanie PD3 WdUM

Jakub Niemyjski

20 grudnia 2023

1 Cel pracy

W tym raporcie ukazany zostanie opis implementacji metody k najbliższych sąsiadów oraz jej przetestowanie w kontekście różnych danych (w celu sprawdzenia poprawności metody) i w kontekście jej czasu wykonania w porównaniu z zewnętrznymi bibliotekami.

2 Opis i implementacja metody `knn`

2.1 Opis

Utworzona funkcja przyjmuje pięć argumentów, kolejno:

- **X**- zbiór treningowy o r atrybutach i n obserwacjach reprezentowany jako `numpy.ndarray`,
- **y**- wektor etykiet dla zbioru treningowego reprezentowany jako `numpy.ndarray` o własności `shape` $(n, 1)$,
- **Z**- zbiór testowy o r atrybutach i m obserwacjach reprezentowany jako `numpy.ndarray`,
- $k \in \mathbb{Z}$ taka, że $1 \leqslant k \leqslant n$, która mówi o tym, ilu najbliższych sąsiadów bierze udział w ustaleniu docelowej etykiety dla zbioru testowego,
- $p \in [1, \infty) \cup \{\text{'infty'}\}$, która mówi jaka norma l_p jest używana do mierzenia odległości między punktami.

Zwraca ona m-elementową tablicę `numpy.ndarray`, która jest etykietami dla kolejnych wierszy macierzy **Z**.

2.2 Implementacja

Pierwszymi operacjami funkcji jest sprawdzenie poprawności danych wejściowych. Następnie inicjalizowany jest wektor $m-$elementowy, który na końcu zostanie zwrócony. Dla każdego wiersza z **Z** wyznaczane są kolejne odległości do każdego z punktów z **X** zgodnie z metryką l_p, potem punkty z **X** są sortowane w kolejności od najbliższych punktów i wybierany zostaje wektor k punktów o najbliższych odległościach. Ostatecznie jest wyznaczana moda etykiet dla tych punktów i ustawiana jako etykieta dla badanego wiersza z **Z**. Jeżeli mód jest kilka, losowana jest zgodnie z rozkładem jednostajnym jedna wartość spośród nich.

3 Sprawdzenie poprawności działania

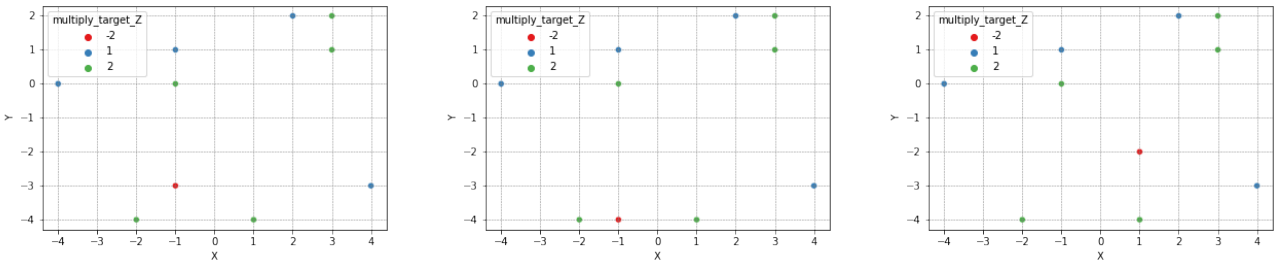
3.1 Tożsamy zbiór treningowy i testowy

Pierwszy test polega na wprowadzeniu identycznego zbioru **X** oraz **Z** i zbadanie dla parametrów $p \in \{1, 2, \text{'infty'}\}$, czy dla $k = 1$ wektor zwrócony będzie identyczny z wektorem **y**. W tym celu utworzona została macierz $\mathbf{X} \in \mathbb{R}^{1000 \times 2}$ o losowych elementach wygenerowanych z rozkładu jednostajnego z $[0, 1)$, oraz wektor etykiet $\mathbf{y} \in \{1, 2, \ldots, 10\}^{1000}$ każdy element z tą samą miarą prawdopodobieństwa. Dla wszystkich tych parametrów p wyniki faktycznie są tożsame wektorowi etykiet **y**.

3.2 Manualne sprawdzenie na dużo mniejszych zbiorach

Niech **X** będzie 10 obserwacjami z dwoma atrybutami, każdy z nich jest losową liczbą całkowitą ze zbioru $\{-4, -3, \ldots, 4\}$. Zbiorem **Z** ustanowiono podobny zbiór, z jedyną różnicą w liczbie obserwacji - 5. Na poniższych wykresach widzimy przykładowe 3 punkty, zaznaczone kolorem czerwonym, co oznacza, że właśnie teraz zostaną sklasyfikowane do jednej z klas 1 lub 2, które są zaznaczone kolorami niebieskim i zielonym. Wszystkie one we wszytkich trzech

przypadkach dla $p \in \{1, 2, \text{'infty'}\}$ zostają sklasyfikowane wartością 2 niezależnie od tego, czy wybierzemy metodą $k-$NN dla $k = 1, 2, 3$.
Istotnie, na rysunkach z lewej strony oraz na środku mamy, że dwoma najbliżej położonymi punktami względem punktów czerwonych są $(-2, -4)$ oraz $(1, -4)$, niezależnie od metryki która zostaje tu zastosowana (spośród tych trzech). Są one sklasyfikowane jako 2, więc zarówno metoda 1-NN, 2-NN oraz 3-NN zaklasyfikuje czerwone punkty jako 2.
Po prawej stronie natomiast widać, że dla metryki l_1 i l_2 najbliżej położonym punktem jest $(1, -4)$, co oznacza, że dla 1-NN dla tych metryk zwróci 2. Natomiast dla metryki l_∞ najbliższymi punktami są jednocześnie $(1, -4)$ oraz $(-1, 0)$, od których odległość wynosi 2. Obydwa one mają etykietę 2, więc 1-NN oraz 2-NN dla tego punktu przypisze w metryce l_∞ wartość 2, a co za tym idzie również dla 3-NN. Pozostało jeszcze zbadać metodę 2-NN oraz 3-NN dla prawego rysunku. odległość od punktu $(-1, 0)$ w normie l_2 wynosi $\sqrt{8}$, a od punktu $(4, -3)$ z kolei $\sqrt{10}$. Okazuje się zatem, że dla 2-NN i 3-NN w normie l_2 klasyfikuje punkt czerwony jako 2. Jeśli chodzi o normę l_1, to trzy najbliższe punkty to $(1, -4), (4, -3), (-1, 0)$, których odległości do czerwonego punktu to kolejno 2,4,4, co oznacza, że 3-NN zwróci tu 2. W 2-NN prawdopodobnie jest wzięty punkt $(-1, 0$ jako drugi od najbliższych, ponieważ metoda po kilkukrotnym wykonaniu zwraca stale 2, co oznacza, że nie mu elementu losowości, na który by wskazywało uwzględnienie punktu $(4, -3)$.



4 Porównanie działania z zewnętrznymi implementacjami

Do porównywania działania funkcji został wykorzystany zbiór, z którego korzystano wielokrotnie na laboratoriach - `pima.csv`. Początkowo, profilaktycznie, aby algorytmy knn wiarygodniej przewidywały, zastosowano `ColumnTransformer` do ustandaryzowania wszystkich zmiennych objaśniających. Podzielono zbiór w losowy sposób na zbiór treningowy i testowy, z ustawionym parametrem `random_state` na 311052.

sklearn - KNeighborsClassifier

4.1 Zgodność wyników

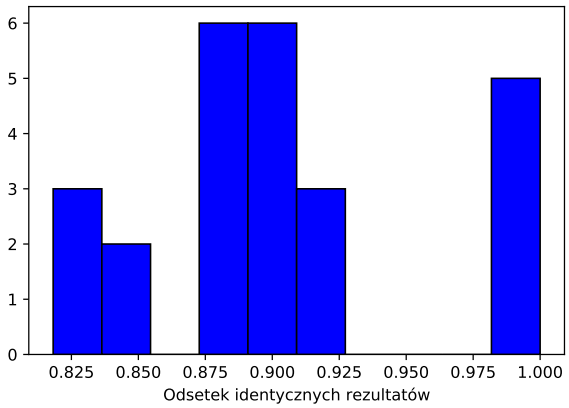
Na rysunku 3 przedstawiono zgodność wyników dwóch podejść, gdzie modele są budowane na zbiorze treningowym, a na zbiorze testowym wyliczane są estymowane etykiety. Tymi dwoma podejściami są funkcja `knn` oraz z pakietu `sklearn KNeighborsClassifier`. Ponadto, z histogramu mamy dane zamieszone w tabeli 2, które wskazują na dużą zgodność obu podejść. Zgodność nie jest jednak we wszystkich przypadkach stuprocentowa. Prawdopodobnie jest to spowodowane tym, że nieco inne działanie ma metoda w Pythonie. Mogło również wydarzyć się tak, że w przypadku danych których odległości były równe, dwie funkcje faworyzowały różnych sąsiadów, pomimo ich jednakowej odległości od klasyfikowanego punktu.

średnia arytmetyczna	mediana	odchylenie standardowe
0.91	0.90	0.05

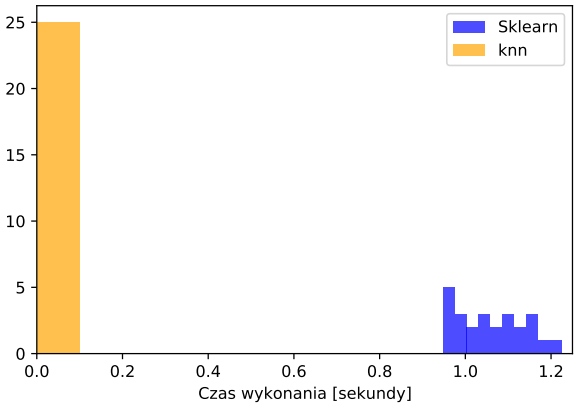
Rysunek 2: Statystyki porównawcze

4.2 Czasy wykonania

Z rysunku 4 w prosty sposób można zauważyć znaczącą różnicę w czasie wykonania obydwu operacji dla różnych parametrów wejściowych. Nieporównywalnie dłużej zachowuje się metoda wbudowana w `sklearn`. Może się to wiązać z tym, że w tle, wraz ze zbudowaniem modelu wykonywane jest wiele innych, bardziej zawiłych i skomplikowanych instrukcji, tak, aby potencjalnie przy większych zbiorach danych zoptymalizować czas wykonania. Zbiór `pima` jak na standardy uczenia maszynowego nie jest jednak duży.



Rysunek 3: Histogram wyniku zgodności dwóch algorytmów



Rysunek 4: Histogram czasów zbudowania modeli

4.3 Ocena testów

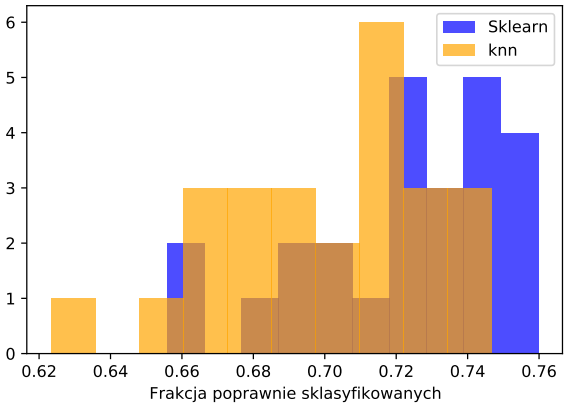
Pomimo dłuższego czasu oczekiwania na rezultaty, funkcja z pakietu `sklearn` lepiej radzi sobie z przewidywaniem dla zbioru testowego. Użyłem tutaj chyba najprostszej oceny, czyli sprawdzenie jaki procent etykiet estymowanych zgadza się z faktycznymi etykietami, co widać na rysunku 6. Na potwierdzenie tej tezy spójrzmy na niektóre statystyki ukazane w tabeli 5

funkcja	średnia arytmetyczna	mediana	odchylenie standardowe
sklearn	0.72	0.73	0.03
knn	0.70	0.71	0.03

Rysunek 5: Statystyki porównawcze

5 Wnioski

Okazało się, że metoda k najbliższych sąsiadów w jednej z jej odsłon nie jest bardzo skomplikowana w implementacji. Okazało się również, że przy zestawieniu jej z klasą `KNeighboursClassifier` faktycznie, średnio ma gorszą wydajność, jednakże czas wykonania jest znacznie szybszy niż metody wbudowanej. Ponadto różnica w jakości modelu nie była wielka, nie to, co różnica w czasie wykonania.



Rysunek 6: Histogram zgodności rozwiązań z rzeczywistością