Laboratorium 5 – 6 — Metody Klasyfikacji

Algorytm k najbliższych sąsiadów (algorytm k-nn z ang. *k nearest neighbours*) jest algorytmem regresji nieparametrycznej używanym do prognozowania wartości pewnej zmiennej losowej. Algorytm może zostać użyty w zadaniu klasyfikacji oraz regresji. W zadaniu klasyfikacji, próbka otrzymuje etykietę którą najczęściej występuje wśród k najbliższych sąsiadów. W zadaniu regresji metoda w trakcie obliczania odpowiedzi modelu bierze pod uwagę k najbliższych punktów i oblicza średnią arytmetyczną wartości ich zmiennej zależnej y. Obliczanie odpowiedzi modelu w zapisie matematycznym wygląda następująco:

$$y = \frac{1}{k} \sum_{i \in N_k(x,X)} y_i,$$

gdzie $N_k(x, X)$ to indeksy k najbliższych punktów do punktu x w całym zbiorze uczącym X. W swojej klasycznej formie algorytm k-NN używa odległości euklidesowej do wybrania najbliższych sąsiadów.

W klasycznej wersji algorytmu k-NN w celu odnalezienia najbliższych sąsiadów obliczana jest odległość każdego punktu uczącego od punktu dla którego chcemy policzyć odpowiedź modelu. Wymaga to zachłannego sprawdzenia wszystkich punktów uczących, co staje się problematyczne w przypadku dużych zbiorów danych. W celu usprawnienia tej procedury zbiór danych można przedstawić w postaci drzewa binarnego. Struktura taka nazywa się **kD-drzewem** i dzieli ona przestrzeń wejść przy pomocy hiperpłaszczyzny na dwie podprzestrzenie. Następnie każda z podprzestrzeni dzielona jest rekursywnie na kolejne podprzestrzenie. Struktura danych nazywana jest kD-drzewem ponieważ przechowuje ona zbiór punktów w k-wymiarowej przestrzeni.

Celem laboratorium jest zapoznanie się z algorytmem najbliższych sąsiadów. Zaimplementowanie metody k-nn i użycie jej w zadaniu klasyfikacji oraz regresji. Przystosowanie algorytmu do korzystania z kD-drzew.

Zadanie 1 - implementacja

Zaimplementuj algorytm k-nn jako klasę w języku Python posiadającą następujące metody:

- Konstruktor: `def __init__(self, n_neighbors = 1, use_KDTree = False)}`
  - `n_neighbors` - liczba sąsiadów,
  - `use_KDTree` - definiuje czy należy korzystać z kD-drzew.
- Metoda do uczenia modelu: `def fit(self, X, y)`
  - `X` - dane wejściowe,
  - `y` - atrybuty decyzyjne/zmienna zależna
- Metoda do dokonywania predykcji: `def predict(self, X)`
- Metoda zwracająca wskaźnik jakości dopasowania: `def score(self, X, y)`, metoda ta powinna zwracać błąd średniokwadratowy dla zadania regresji lub procentową dokładność w zadaniu klasyfikacji.

Algorytm powinien zostać zaimplementowany w dwóch wersjach do zadania klasyfikacji oraz regresji (można zastosować dodatkową flagę w konstruktorze, lub sprawdzać czy zmienna y posiada dane typu INT, czy DOUBLE).

Zadanie 2 - kD-drzewa

Rozszerz implementację aby korzystała z kD-drzew. W celu realizacji zadania należy skorzystać z gotowej struktury dostępnej w Pythonie (`sklearn.neighbors.KDTree`).

Zadanie 3 - klasyfikacja

Wygeneruj dane uczące przy pomocy metody `sklearn.datasets.make_classification`,

```
X, y = datasets.make_classification(
        n_samples=100,
        n_features=2,
        n_informative=2,
        n_redundant=0,
        n_repeated=0,
        random_state=3)
```

a następnie:

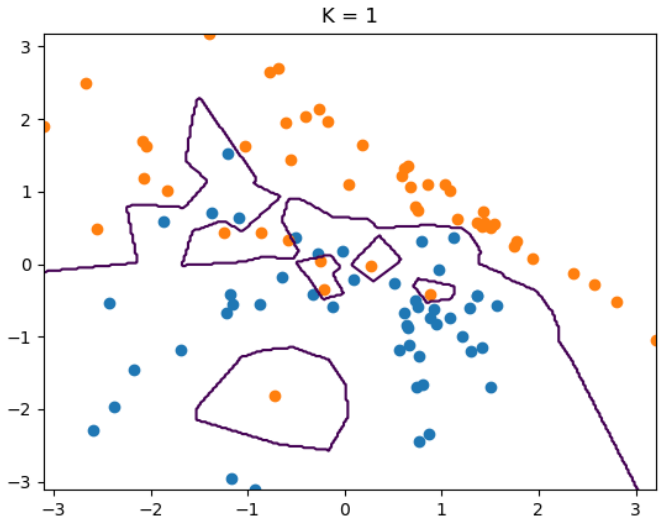
1. Dokonaj klasyfikacji przy pomocy metody k-nn;
2. Zwizualizuj dane oraz granicę separacji w przestrzeni 2D, dla róznej liczbya sąsiadów $k \in [1, 5]$. Granica separacji może zostać zobrazowana przy pomocy metody `contour`. W celu jej narysowania należy wygenerować regularną siatkę punktów (metoda `meshgrid`), a następnie dla każdego węzła siatki obliczyć odpowiedź algorytmu k-nn. Przykładowa wizualizacja została zaprezentowana na Rysunku 1
3. Wczytaj dane `iris` oraz rozdziel je na cześć wejściową (X) oraz klasy (Y), a następnie dokonaj ich klasyfikacji.
4. Zwizualizuj dane przy pomocy metody PCA (Rysunek 2) rzutując dane na dwie pierwsze składowe główne. Umieść na wykresie granicę separacji. W celu wizualizaji granicy separacji należy:
   (a) Wygenerować regularną siatkę punków w przestrzeni 2D.
   (b) Punkty opisujące siatkę należy przekonwertować do „oryginalnej" przestrzeni (4D) przy pomocy metody `pca.inverse_transform`.
   (c) Rezultat punktu poprzedniego należy podać do funkcji `predict`, a uzyskany wynik wraz z regularną siatką (z pkt. a) narysować przy pomocy metody `contour`.
5. Przy pomocy kroswalidacji leave-one-out przetestuj jak algorytm zachowuje się przy różnych wartościach parametru k. Wyniki wypisz na ekran w postaci tabeli.
6. Porównaj czas działania algorytmu w wersji podstawowej i wersji korzystającej z kD-Drzew.

Zadanie 4 - regresja

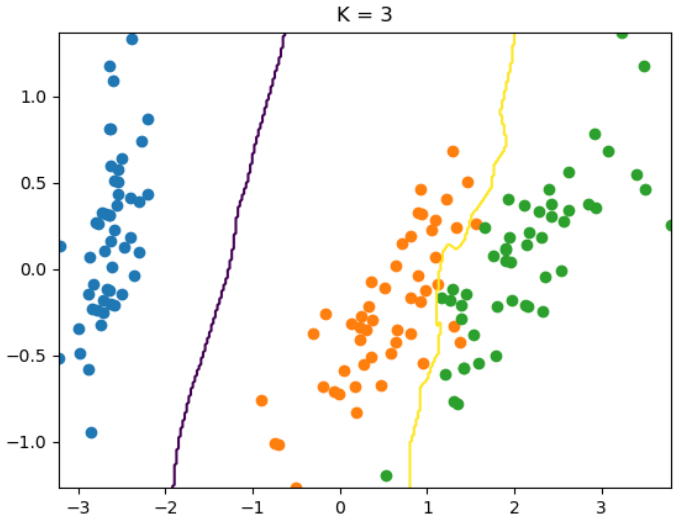
Wygeneruj dane uczące (2D) przy pomocy metody `sklearn.datasets.make_regression`.

1. Dokonaj regresji przy pomocy metody k-nn.
2. Zwizualizuj dane oraz odpowiedź modelu (linia trendu). Wypisz na ekran błąd popełniany przez model.

Rysunek 1: Przykładowa wizualizacja – random data.



Rysunek 2: Przykładowa wizualizacja – iris.



3. Wczytaj dane `boston` (`sklearn.datasets.load_boston`).

4. Przy pomocy 10-krotnej krzyżowej walidacji przetestuj model dla różnych wartości parametru k. Wyniki wypisz na ekran w postaci tabeli.