

Wprowadzenie teoretyczne do KNN i podziału danych

April 22, 2025

Ćwiczenie 4 (1pkt)

KNN i Podział danych

Zadanie do wykonania

1. Czytamy teorię.
2. Wczytujemy bazę próbek iris. Ostatni atrybut (kolumna) określa klasę próbek
3. Dzielimy zbiór danych na treningowy i testowy w proporcjach 70% do 30%.
4. Implementujemy algorytm w Python:
 - Podstawowy klasyfikator KNN, wykorzystujący 3 metryki : Euklidesową, Cosinusową i Manhattan.
5. Obliczamy accuracy dla naszego modelu

Uczenie maszynowe oraz analiza danych opierają się na umiejętności wyciągania wniosków z danych. Kluczowym elementem tej analizy jest ocena skuteczności modelu, która wymaga odpowiedniego podziału danych

Czym są zbiory danych w uczeniu maszynowym?

Zbiory danych to zbiór przykładów (rekordów), które zawierają informacje na temat konkretnego zjawiska. W praktyce, dane te są zwykle zapisane w plikach, takich jak CSV, JSON czy Excel. Przykładem danych może być tabela, w której kolumny zawierają różne cechy (np. wiek, oceny), a wiersze reprezentują różne obserwacje (np. studenci).

Name	Age	Grade
Alice	20	85
Bob	22	90
Charlie	23	78

Table 1: Tabela przedstawiająca wiek i oceny studentów.

Dlaczego dzielimy dane na zbiory treningowe i testowe?

Podział danych na zestawy treningowe i testowe jest kluczowy, ponieważ pozwala ocenić rzeczywistą skuteczność modelu.

Zbiór treningowy:

- Używany do nauki modelu, czyli do dostosowania parametrów modelu na podstawie wzorców w danych.
- Model "widzi" te dane podczas procesu uczenia.

Zbiór testowy:

- Używany do oceny modelu, czyli sprawdzenia, jak dobrze model generalizuje na danych, których wcześniej nie widział.
- Model "nie widzi" tych danych podczas procesu uczenia

Problem bez podziału danych

Jeżeli model oceniamy na danych, na których został wytrenowany, jego wyniki będą bardzo optymistyczne (często bliskie 100%). Nie odzwierciedlają one jednak rzeczywistej skuteczności, ponieważ model może "zapamiętać" dane, zamiast nauczyć się ogólnych wzorców.

Typowe proporcje podziału danych

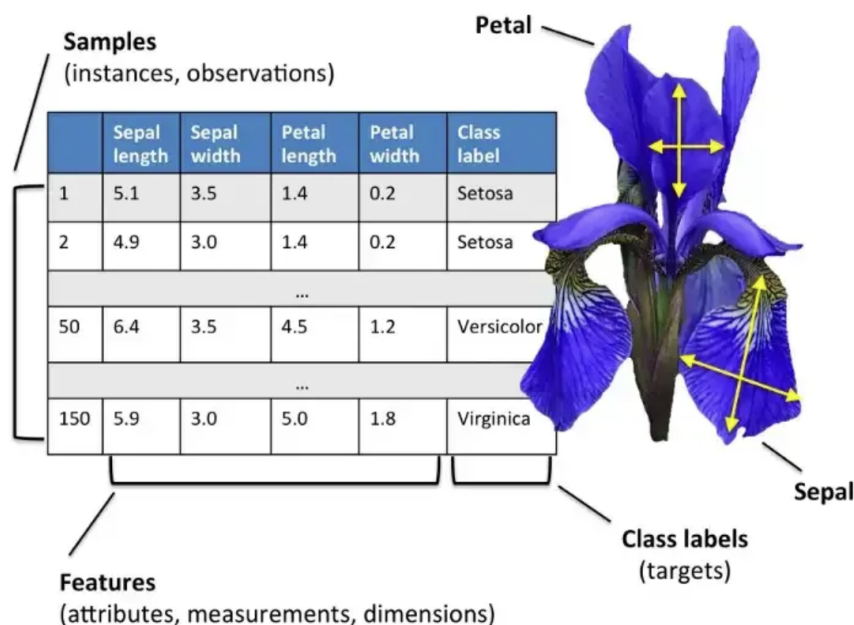
Najczęściej używane proporcje podziału:

- 70% treningowe / 30% testowe (najczęściej stosowane w praktyce),
- 80% treningowe / 20% testowe (przy dużych zbiorach danych),
- 60% treningowe / 20% walidacyjne / 20% testowe (dla bardziej złożonych modeli).

Dlaczego nie używamy całego zbioru na testy?

Jeśli zostawimy zbyt mało danych do treningu, model nie nauczy się wystarczająco dobrze wzorców. Optymalny podział zależy od wielkości i różnorodności danych.

Zbiór danych irys



Zestaw danych Iris jest bezpośrednio dostępny jako część scikit-learn. Można go załadować za pomocą funkcji `load_iris`. Przy domyślnych parametrach zwracany jest obiekt Bunch zawierający dane, wartości docelowe, nazwy funkcji i nazwy docelowe.

```
from sklearn.datasets import load_iris  
  
iris = load_iris(as_frame=True)  
print(iris.keys())
```

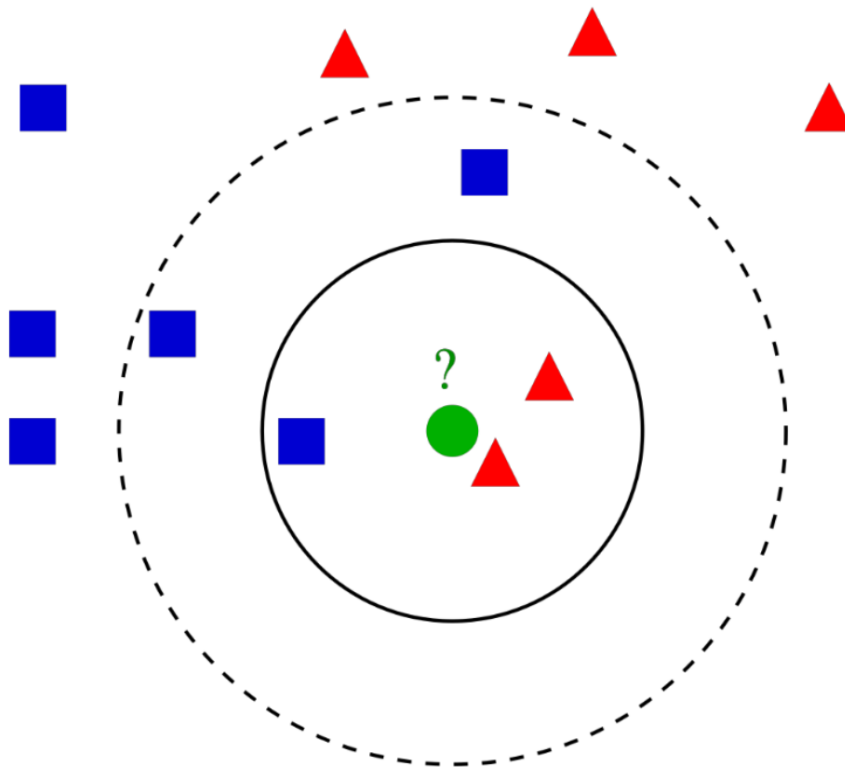
KNN (K-Nearest Neighbors)

Teoria

W statystyce algorytm k -najbliższych sąsiadów (k -NN) jest nieparametryczną metodą. Służy do klasyfikacji i regresji. W obu przypadkach dane wejściowe składają się z k najbliższych przykładów uczących w zbiorze danych. Wynik zależy od tego, czy k -NN jest używane do klasyfikacji czy regresji:

- W klasyfikacji k -NN wynikiem jest przynależność do klasy. Obiekt jest klasyfikowany na podstawie wielu głosów jego sąsiadów, przy czym obiekt jest przypisywany do klasy najczęściej występującej wśród jego k najbliższych sąsiadów (k jest dodatnią liczbą całkowitą, zwykle małą). Jeśli $k = 1$, to obiekt jest po prostu przypisywany do klasy tego najbliższego sąsiada.
- W regresji k -NN wynikiem jest wartość właściwości obiektu. Ta wartość jest średnią z wartości k najbliższych sąsiadów. Jeśli $k = 1$, to wyjście jest po prostu przypisywane do wartości tego pojedynczego najbliższego sąsiada.

k -NN to typ klasyfikacji, w którym funkcja jest aproksymowana tylko lokalnie, a wszystkie obliczenia są odraczane do czasu oceny funkcji. Ponieważ ten algorytm opiera się na klasyfikacji na odległości, jeśli cechy reprezentują różne jednostki fizyczne lub występują w bardzo różnych skalach, normalizacja danych treningowych może radykalnie poprawić ich dokładność.



Przykład klasyfikacji k -NN. Próbkę testową (zielona kropka) należy podzielić na niebieskie kwadraty lub czerwone trójkąty. Jeśli $k = 3$ (ciągła linia koła) jest przypisane do czerwonych trójkątów, ponieważ wewnątrz wewnętrznego koła są 2 trójkąty i tylko 1 kwadrat. Jeśli $k = 5$ (przerywana linia okręgu), jest przypisany do niebieskich kwadratów (3 kwadraty vs. 2 trójkąty wewnątrz zewnętrznego koła).

Popularne metryki

Metryka Euklidesowa

W matematyce odległość euklidesowa między dwoma punktami w przestrzeni euklidesowej to długość odcinka między tymi dwoma punktami. Można ją obliczyć ze współrzędnych kartezjańskich punktów za pomocą twierdzenia Pitagorasa, dlatego czasami nazywa się ją odległością Pitagorasa. Dla punktów podanych przez współrzędne kartezjańskie w n -wymiarowej przestrzeni euklidesowej, odległość wynosi:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

Metryka Manhattan

Geometria Manhattanu to geometria, której zwykła funkcja odległości lub metryka geometrii euklidesowej została zastąpiona nową metryką, w której odległość między dwoma punktami jest sumą bezwzględnych różnic ich współrzędnych kartezjańskich.

$$d_T(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_T = \sum_{i=1}^n |p_i - q_i| \quad (2)$$

Na przykład w \mathbb{R}^2 , odległość taksówki między $\mathbf{p} = (p_1, p_2)$ i $\mathbf{q} = (q_1, q_2)$ jest $|p_1 - q_1| + |p_2 - q_2|$.

Wskazówki do wykonania zadania

Pomocne narzędzia do implementacji: scikit-learn, numpy, pandas

1. Pobieranie zbioru danych from sklearn import load_iris – zbiór 3-klasowy, można wyeliminować jedną z klas, można też działać na wszystkich
2. Podział danych na zbiór treningowy i testowy Jednym ze sposobów jest np. użycie funkcji train_test_split
 - Należy losowo wyselekcjonować próbkę treningową i testową
 - optymalny podział w tym przypadku to 70% dla treningu, i 30% dla testu Od zbioru testowego należy oddzielić atrybuty decyzyjne – czyli klasy – i przechowywać je w odrębnej liście
 - Ze zbioru treningowego też można te klasy oddzielić, ale nie trzeba – wystarczy pamiętać, żeby ostatniej kolumny nie uwzględniać w obliczeniach.
3. Klasyfikator KNN jest to funkcja, która zwraca tablicę (listę), zawierającą klasy przewidywane dla danego zbioru danych. W naszej sytuacji przekazywać będziemy liczbę całkowitą reprezentującą liczbę k sąsiadów, z których zaczerpnijemy naszą decyzję, nasz zbiór testowy (bez klasy decyzyjnej) do ponownej klasyfikacji, oraz zbiór treningowy. Zbiór treningowy służy naszemu klasyfikatorowi jako jego baza wiedzy, i do jego elementów porównywane będą elementy ze zbioru testowego.

Na początku wykonywania funkcji należy zainicjować listę o długości zbioru testowego – w niej będziemy przechowywać przewidziane klasy, i ją będziemy zwracać na końcu funkcji. Następnie tworzymy pętlę iterującą po wszystkich wierszach zbioru testowego. W tej pętli tworzymy nową listę o długości zbioru treningowego – w niej będziemy przechowywać odległości naszego elementu testowego do elementów zbioru treningowego. Następnie będziemy wyliczać te dystanse (w pętli iterującej po obiektach treningowych) i zapisywać je do tej listy. Metody obliczania dystansu nazywamy metrykami. Przykładami najpopularniejszych metryk są:

- Dystans euklidesowy
- Metryka Manhattan
- Metryka cosinusowa

Przy obliczaniu dystansu traktujemy każdy z atrybutów jako wymiar. W wypadku datasetu iris operujemy w czterech wymiarach. Po obliczeniu odległości danego obiektu testowego do wszystkich obiektów zbioru treningowego należy odnaleźć k obiektów treningowych o najkrótszych dystansach, i sprawdzić ich klasy. Najczęstsza z tych klas zostanie naszą klasą przewidzianą dla danego obiektu testowego, i zapisana zostanie na liście przewidzianych klas (tej, którą utworzyliśmy na samym początku funkcji). Jeżeli przekazaliśmy klasy decyzyjne zbioru treningowego jako odrębny argument, pomocna może się okazać funkcja `argsort` z biblioteki NumPy. Zwraca ona listę indeksów danej tablicy posortowaną względem jej wartości. Po ustaleniu klasy dla każdego obiektu testowego i zapisaniu każdej z nich do listy, kończymy działanie funkcji, zwracając tę listę. Po zakończeniu działania funkcji sprawdzamy dokładność naszego klasyfikatora. W tym celu porównujemy elementy listy zwróconej przez funkcję z decyzjami, które oddzieliliśmy od naszego zbioru testowego. Dla zbioru iris przy $k=3$ wyniki dla każdej z metryk wymienionej powyżej powinny wynosić około 90%, natomiast zdarza się, że metryka cosinusowa osiągnie nawet 100%.