WSTĘP DO ANALIZY DANYCH

Lab. 2: Metoda najbliższych sąsiadów

v. 1.0.0_zodp

PLAN LAB. 2

1. Zasada działania metody najbliższych sąsiadów (algorytmy leniwe, ang. *lazy learning*).
2. Metoda k-NN w praktyce (odległość między obserwacjami, dobór parametrów, przygotowanie danych).
3. Przykład zastosowania k-NN na danych rzeczywistych.

METODA NAJBLIŻSZYCH SĄSIADÓW

(do klasyfikacji)

- Pomysł jest prosty: **podobne obiekty mają podobne właściwości**.

  Przykład: restauracja w ciemności — próba rozpoznania dania na podstawie porównania jego cech do cech znanych nam dań; jeśli pachnie jak kaczka i smakuje jak kaczka, to z dużym prawdopodobieństwem jemy kaczkę.

- W ML używa się tego pomysłu do klasyfikowania danych poprzez przypisywanie im klasy, do której należą „podobne" dane (najbliżsi sąsiedzi).

- Pomimo swojej prostoty metoda najbliższych sąsiadów z powodzeniem wykorzystywana jest do zadań takich jak: komputerowe rozpoznawanie obrazów (w tym np. rozpoznawanie twarzy na zdjęciach i nagraniach wideo), czy systemy rekomendacji (np. filmów lub utworów muzycznych).

METODA k-NN

- Przykładem metody uczenia maszynowego, która opiera się na pomyśle najbliższych sąsiadów, jest **metoda k najbliższych sąsiadów** (ang. *k-nearest neighbors*; metoda k-NN).

- Metoda k-NN przewiduje klasę danego przykładu na podstawie k najbliższych obserwacji ze zbioru uczącego. Zmienna k jest parametrem tej metody i teoretycznie może przyjąć każdą wartość naturalną dodatnią.

- Dla ustalonego k, dla każdego przykładu ze zbioru testowego (przykłady te nie mają etykiet) algorytm k-NN znajduje k przykładów ze zbioru uczącego, które są najbliższe pod względem podobieństwa (odpowiednio zdefiniowanego) do danego przykładu ze zbioru testowego.

  Następnie przykłady ze zbioru testowego przypisywane są do klas, które stanowią większość wśród ich k sąsiadów ze zbioru testowego.

METODA k-NN

- Mimo że jest to chyba najprostszy algorytm uczenia maszynowego, nadal jest szeroko stosowany.

Mocne strony	Słabe strony
Prosty i skuteczny	Nie jest budowany model; utrudnione zrozumienie zależności między cechami a klasami
Brak założeń o rozkładzie danych	Wymaga wyboru odpowiedniego parametru k
Krótki czas uczenia/budowy modelu	Czasochłonny proces klasyfikacji
	Zmienne jakościowe oraz brakujące dane nie są domyślnie obsługiwane

PRZYKŁAD ILUSTRUJĄCY

Powrót do przykładu z jedzeniem w ciemności.

- Załóżmy, że przed zjedzeniem posiłku w ciemności stworzyliśmy zbiór danych, w którym zapisaliśmy nasze wrażenia z jedzenia różnych składników dań.

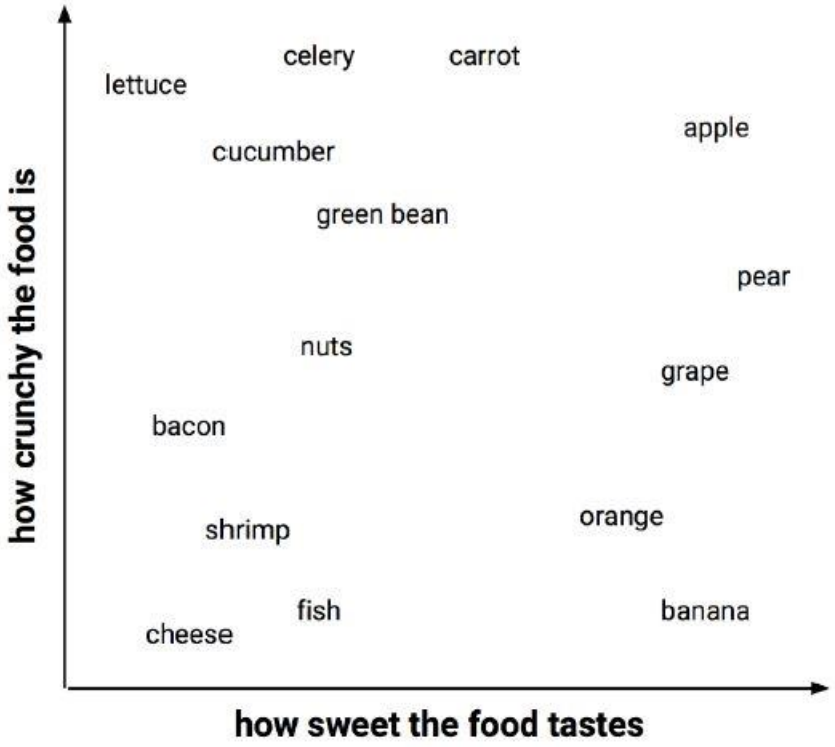
  Dla uproszczenia załóżmy, że ocenialiśmy dwie ich cechy: chrupkość oraz słodkość (obie w skali 1–10). Ponadto załóżmy, że rozważamy tylko trzy kategorie żywności: owoce, warzywa i źródła białka.

- Kilka wierszy takie zbioru danych mogłoby wyglądać w następujący sposób:

Ingredient	Sweetness	Crunchiness	Food type
Apple	10	9	Fruit
Bacon	1	4	Protein
Banana	10	1	Fruit
Carrot	7	10	Vegetable
Celery	3	10	Vegetable
Cheese	1	1	Protein

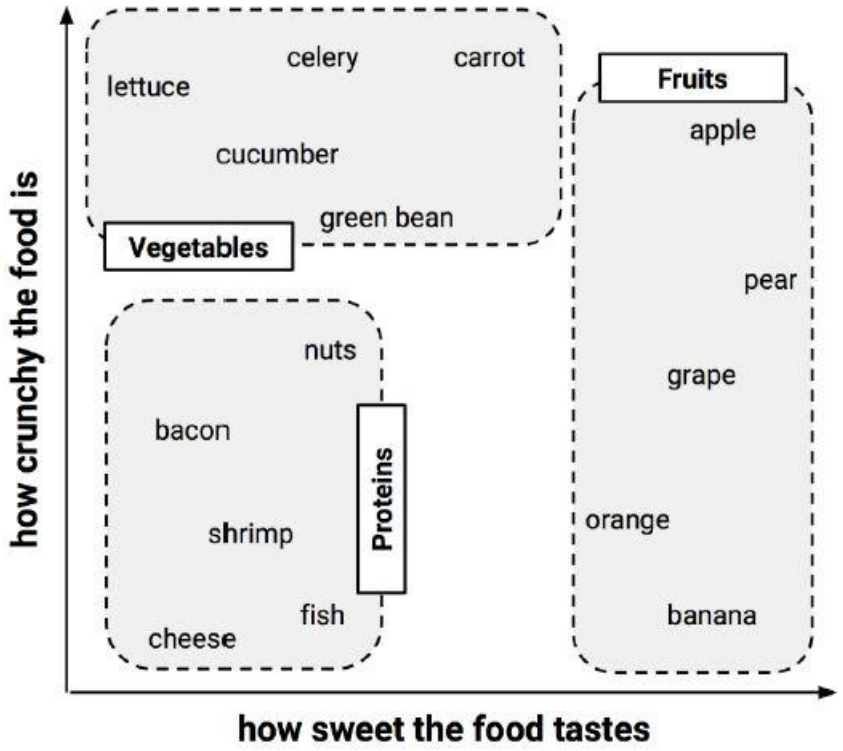
PRZYKŁAD ILUSTRUJĄCY

- Algorytm k-NN traktuje cechy jako współrzędne obserwacji w wielowymiarowej przestrzeni cech. Wykres punktowy dla naszego wymyślonego przykładu mógłby wyglądać na przykład tak:



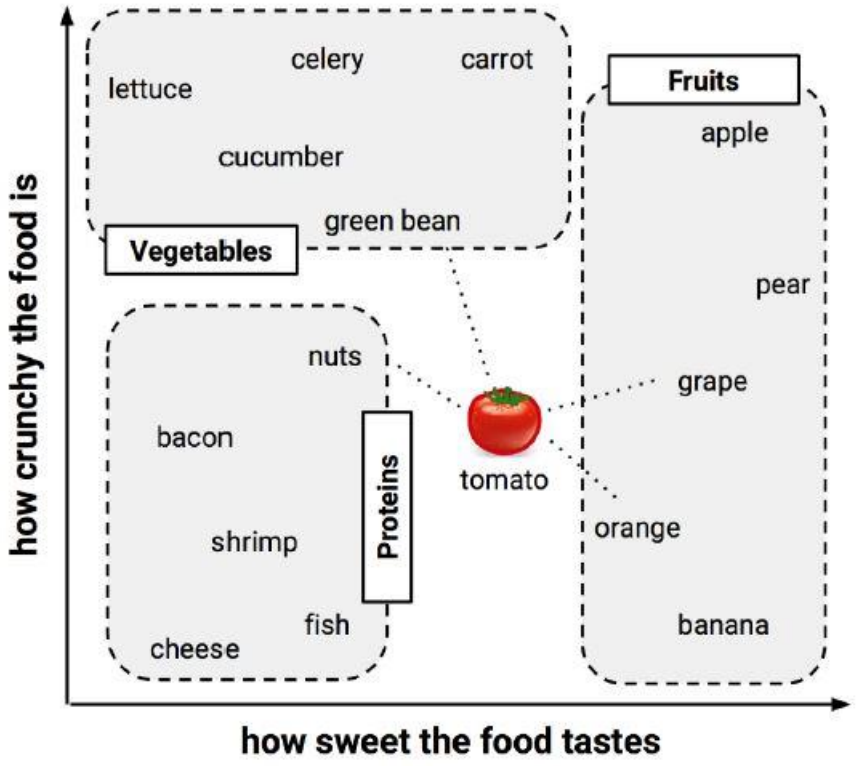
PRZYKŁAD ILUSTRUJĄCY

- Przykłady składników w ramach jednej klasy tworzą widoczne grupy.



PRZYKŁAD ILUSTRUJĄCY

- Dysponując takim zbiorem danych, za pomocą metody k-NN możemy klasyfikować inne składniki do jednej z trzech klas: owoce, warzywa, źródła białka.



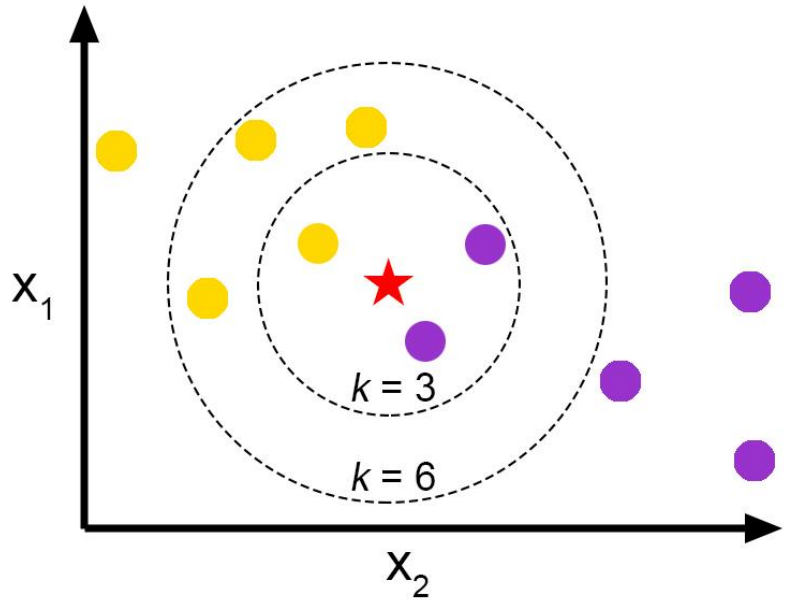
ODLEGŁOŚĆ POMIĘDZY OBSERWACJAMI

- Istnieje wiele sposobów obliczania odległości pomiędzy punktami z przestrzeni wielowymiarowej. Standardowa wersja k-NN używa metryki euklidesowej:

$$\text{dist}(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2}.$$

- Czasami stosuje się kwadrat metryki euklidesowej (który nie jest już metryką), ważoną odległość euklidesową, odległość Mahalanobisa (ta miara uwzględnia korelacje pomiędzy cechami).

- Rzadziej stosuje się też inne miary.



WYBÓR PARAMETRU k

- Decyzja o tym, ilu sąsiadów użyć w metodzie k-NN, ma wpływ na to, jak dobrze model będzie generalizował się na przyszłe dane.
- Trzeba wypośrodkować pomiędzy nadmiernym dopasowaniem (ang. *overfitting*) modelu do danych uczących a jego niedopasowaniem (ang. *underfitting*) — kompromis pomiędzy obciążeniem (poziomem uproszczenia modelu) a wariancją (poziomem wrażliwości modelu).
- Wybór dużej wartości parametru k zmniejsza wpływ zaszumionych danych (wariancję), ale może prowadzić do dużego obciążenia i ignorowania „małych", ale ważnych wzorców w danych.
- Skrajna sytuacja — wybieramy k równe ilości obserwacji w zbiorze uczącym. Wówczas każdej nowej obserwacji algorytm k-NN przypisywał będzie (tę samą) klasę większości obserwacji ze zbioru uczącego.
- Z drugiej strony, używając $k = 1$, pozwolimy na to, aby odstające oraz zaszumione dane (np. dane przypadkowo błędnie oznaczone) miały wpływ na klasyfikację nowych przykładów.

WYBÓR PARAMETRU k W PRAKTYCE

- W praktyce wybór k zależy od złożoności rozważanego problemu oraz od ilości przykładów w zbiorze uczącym i nie ma na to prostej recepty.
- Jedną z praktyk jest rozpoczęcie poszukiwania optymalnego parametru k od wartości równej pierwiastkowi z liczby przykładów w zbiorze uczącym. Taka wartość k jednak nie zawsze działa dobrze.
- Alternatywne podejście (zwykle stosowane w praktyce) polega na testowaniu różnych wartości k na ustalonym zbiorze testowym lub testowaniu różnych wartości k na różnych zbiorach testowych i wyborze tego k, które daje „najlepsze" wyniki.

  Nawet gdy dane są bardzo zaszumione, przy dużym zbiorze danych uczących wybór k jest nieco mniej jednoznaczny. Dzieje się tak, ponieważ nawet subtelne zależności mają wystarczająco dużo przykładów, które będą głosować jako najbliżsi sąsiedzi.
- Mniej powszechnym, ale ciekawym rozwiązaniem jest wybór większych wartości k i zastosowanie **głosowania z wagami**, w którym bliższych sąsiadów uważa się za bardziej wiarygodnych (przypisuje się im wyższe wagi niż sąsiadom bardziej oddalonym).

PRZYGOTOWANIE DANYCH POD k-NN

- Odległość cech zależy od tego, na jakim zakresie wartości są one mierzone. Z tego powodu przed zastosowaniem metody k-NN wszystkie cechy zwykle przekształca się do jednego ustalonego zakresu wartości (aby nie doprowadzić do sytuacji, że jedna cecha ma przeważające znaczenie w rozróżnianiu sąsiadów).

- Powszechnie stosowaną metodą skalowania danych pod k-NN jest **normalizacja min-max** (wszystkie wartości sprowadzamy do przedziału $[0, 1]$):

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}.$$

- Inna popularna metoda to **standaryzacja z-score** (standaryzacja Z):

$$X_{new} = \frac{X - \mu}{\sigma} = \frac{X - \mathrm{Mean}(X)}{\mathrm{StdDev}(X)},$$

  gdzie μ to średnia, natomiast σ — odchylenie standardowe na standaryzowanym zbiorze danych.

  Nowe wartości nie mają założonego z góry minimum lub maksimum.

PRZYGOTOWANIE DANYCH POD k-NN

- **Uwaga**: przykłady ze zbioru testowego muszą zostać również przeskalowane tą samą metodą co przykłady ze zbioru uczącego.
- **Problemy**: min i max nowych przykładów może znajdować się poza zakresem danych ze zbioru uczącego. Jeśli znamy z góry wiarygodne min i max dla analizowanych danych, to można użyć tych wartości zamiast min i max obserwacji.

  Podobny problem występuje w naturalnym sposób przy klasyfikacji nowych pojedynczych obserwacji.

  Jeśli możemy założyć, że przyszłe przypadki będą miały podobną średnią i odchylenie standardowe jak obserwacje ze zbioru uczącego, to dobrą opcją jest zastosowanie standaryzacji z-score.

PRZYGOTOWANIE DANYCH POD k-NN

- Odległość euklidesowa nie jest zdefiniowana dla danych jakościowych.
- Typowe rozwiązanie (*dummy coding*) polega na przypisaniu wartości 1 jednej kategorii, a wartości 0 drugiej kategorii, np.

$$\text{male} = \begin{cases} 1, & \text{gdy } x = \text{male}, \\ 0 & \text{w przeciwnym wypadku.} \end{cases}$$

- Zmienne, które posiadają n kategorii, kodowane są natomiast za pomocą $n-1$ indykatorów dla $n-1$ poziomów danej zmiennej. Na przykład kodowanie zmiennej *temperature*, która przyjmuje trzy wartości: *hot*, *medium*, *cold*, może wyglądać w następujący sposób:

$$\text{hot} = \begin{cases} 1, & \text{gdy } x = \text{hot}, \\ 0 & \text{w przeciwnym wypadku,} \end{cases} \qquad \text{medium} = \begin{cases} 1, & \text{gdy } x = \text{medium}, \\ 0 & \text{w przeciwnym wypadku.} \end{cases}$$

- Nie potrzebujemy trzeciej zmiennej dla kategorii *cold*.

,,LENIWY" ALGORYTM

- Metoda k najbliższych sąsiadów należy do grupy tzw. algorytmów leniwych (ang. *lazy algorithms*), czyli takich, które nie tworzą wewnętrznej reprezentacji danych uczących (nie jest budowany model analityczny, w zasadzie pomijany jest krok generalizacji), lecz szukają rozwiązania dopiero w momencie pojawienia się nowej obserwacji, którą np. trzeba sklasyfikować.
- Leniwe metody w zasadzie niczego się nie uczą. Zamiast tego po prostu przechowują dane uczące.

PRZYKŁAD NA DANYCH RZECZYWISTYCH

Zastosowanie k-NN do automatyzacji procesu badań przesiewowych w kierunku raka piersi.

- Tego typu zastosowania uczenia maszynowego (jak automatyzacja procesu identyfikacji komórek rakowych) mają oczywiście sens:

  poprawa wydajności procesu wykrywania — dzięki temu lekarz może poświęcić więcej czasu na leczenie, zamiast na badanie,

  zautomatyzowane systemy badań przesiewowych mają potencjał zapewnienia większej dokładności, ponieważ eliminują z natury subiektywny czynnik ludzki.

KROK 1. ZBIERANIE DANYCH

- Wykorzystamy zbiór danych *Breast Cancer Wisconsin (Diagnostic)* z UCI Machine Learning Repository (`http://archive.ics.uci.edu/ml`).

- Dane przedstawiają charakterystyki wyznaczone na podstawie zdigitalizowanego obrazu pobranych komórek metodą aspiracji cienkoigłowej piersi.

- Dane zawierają 569 obserwacji i 32 zmienne:

  1: *id* — identyfikację obserwacji,

  2: *diagnosis* (M = malignant, B = benign) — diagnozę zmiany (M = złośliwa, B = łagodna),

  3-32: rzeczywiste charakterystyki opisujące poszczególne komórki:
  radius — promień (średnia odległość od środka do punktów na brzegu), *texture* — odchylenie standardowe wartości skali szarości, *perimeter* — obwód, *area* — pole, *smoothness* — lokalna zmienność długości promienia, *compactness* — $(\text{obwód}^2/\text{pole} - 1)$, *concavity* — proporcja wklęsłych fragmentów brzegu, *concave points* — liczba wklęsłych fragmentów brzegu, *symmetry* — symetria, *fractal dimension* — wymiar fraktalny.

KROK 1. ZBIERANIE DANYCH

- Pierwsze 10 zmiennych opisuje średnie wartości powyższych charakterystyk w próbce, drugie 10 zmiennych opisuje błąd standardowy, ostatnie 10 zmiennych opisuje „najgorsze” lub największe wartości charakterystyk komórek w obrazie.

- Patrząc na nazwy, wydaje się, że wszystkie cechy związane są z kształtem i rozmiarem komórki. Jeśli nie jesteś onkologiem, prawdopodobnie nie wiesz, w jaki sposób każda z tych cech jest związana z rodzajem nowotworu (złośliwy/łagodny). Wykorzystując uczenie maszynowe (metodę k-NN), spróbujemy znaleźć te zależności.

KROK 2. EKSPLORACJA I PRZYGOTOWANIE DANYCH

Zadanie 1.

a) Pobrać dane **wisc_bc_data.csv** i umieścić je w katalogu roboczym R. Aby sprawdzić, który katalog jest aktualnie katalogiem roboczym, korzystamy z funkcji `getwd()`. Aby określić lub zmienić katalog roboczy, korzystamy z funkcji `setwd()`.

b) Wczytać dane do ramki danych.

c) Sprawdzić poprawność wczytanych danych pod kątem ilości przykładów i cech. Wykorzystać funkcję `str()`.

d) Pierwsza zmienna to zmienna całkowita o nazwie *id*. Usunąć tę zmienną ze stworzonej ramki danych, nie zmieniając jej nazwy. Zmienne tego typu powinny być wykluczane z analizy, niezależnie od stosowanej metody uczenia maszynowego. Uzasadnić, dlaczego.

KROK 2. EKSPLORACJA I PRZYGOTOWANIE DANYCH

e) Wartości zmiennej *diagnosis* będziemy chcieli przewidywać, z tego powodu jest ona szczególnie ważną zmienną w analizowanym zbiorze danych. Używając funkcji `table()`, sprawdzić liczebności poszczególnych klas dla tej zmiennej.

f) Wiele bibliotek uczenia maszynowego w R wymaga, aby przewidywana zmienna byłą zakodowana jako czynnik (ang. *factor*). Przekodować zmienną *diagnosis* na czynnik. Przy okazji nadać wartościom *B* oraz *M* etykiety *Benign* oraz *Malignant*. Następnie wykonać i przeanalizować polecenie:

```
> round(prop.table(table(wbcd$diagnosis)) * 100, digits = 1)
```

g) Używając funkcji `summary()`, wyświetlić podstawowe statystyki dla zmiennych *radius_mean*, *area_mean*, *smoothness_mean*. Porównując te trzy przykładowe zmienne między sobą, wskazać problem, z którym będzie trzeba się zmierzyć przed zastosowaniem metody k-NN.

KROK 2. CD.

Normalizacja zmiennych numerycznych.

Zadanie 2.

a) Stworzyć funkcję `normalize()`, która nromalizuje wszystkie współrzędne wejściowego wektora x i zwraca wektor ze znormalizowanymi współrzędnymi:

```
> normalize <- function(x) {
+     return ((x - min(x)) / (max(x) - min(x)))
+ }
```

b) Przetestować utworzoną funkcję na kilku wybranych przykładach. W szczególności wskazać dwa różne wektory, które po znormalizowaniu są sobie równe.

KROK 2. CD.

Normalizacja zmiennych numerycznych.

c) Zastosować funkcję `normalize()` do zmiennych numerycznych z analizowanego zbioru danych. Zautomatyzować ten proces, używając w tym celu funkcji `lapply()`. Na koniec przekonwertować listę zwróconą przez funkcję `lapply()` do ramki danych. W tym celu użyć funkcji `as.data.frame()`.

d) Celem potwierdzenia, że operacja normalizacji została przeprowadzona w prawidłowy sposób, wyświetlić statystyki podsumowujące dla wybranej zmiennej (np. dla *area_mean*).

KROK 2. CD.

Przygotowanie zbioru uczącego i testowego.

- Aby zasymulować działanie budowanego modelu w warunkach rzeczywistych (np. w laboratorium dla nowych próbek danych, niepochodzących z analizowanego przez nas zbioru danych), podzielimy dane na dwie części: **zbiór uczący**, który zostanie wykorzystany do budowy modelu k-NN, **zbiór testowy**, który zostanie wykorzystany do oszacowania zdolności predykcyjnej zbudowanego modelu. Z pierwszych 469 rekordów utworzymy zbiór uczący, natomiast z pozostałych 100 — zbiór testowy.

- Uwaga: podczas konstrukcji zbiorów uczących i testowych ważne jest, aby zadbać o to, by każdy z utworzonych zbiorów danych był reprezentatywnym podzbiorem pełnego zbioru danych.

- Rekordy w analizowanym przez nas zbiorze danych ułożone są w sposób losowy, dlatego możemy po prostu wybrać 100 ostatnich rekordów i z nich utworzyć zbiór testowy. Gdyby dane w wyjściowym zbiorze były ułożone chronologicznie lub gdyby były pogrupowane względem podobieństwa pewnych cech, zastosowane przez nas podejście nie byłoby poprawne.

KROK 2. CD.

Przygotowanie zbioru uczącego i testowego.

Zadanie 3.

a) Utworzyć dwie ramki danych: ze zbiorem uczącym i ze zbiorem testowym.

b) Podczas konstrukcji znormalizowanych zbiorów danych uczących i testowych pominięta została zmienna *diagnosis*, dla której robiona będzie predykcja. Na etapie budowy modelu k-NN potrzebowali będziemy dwóch wektorów, w których, jako czynniki, przechowywane są etykiety dla rekordów, odpowiednio, ze zbioru uczącego i testowego. Utworzyć takie struktury.

KROK 3. BUDOWA MODELU

- W przypadku metody k-NN na etapie uczenia w zasadzie nie jest budowany żaden model analityczny. Algorytm po prostu przechowuje w ustrukturyzowanej formie dane uczące.
- Do klasyfikacji przykładów ze zbioru testowego użyjemy implementacji k-NN z pakietu **class**, który zawiera zestaw podstawowych funkcji do klasyfikacji.
- Funkcja `knn()` zawiera implementację standardowej wersji metody k-NN: odległość euklidesowa, klasyfikacja przez głosowanie k najbliższych sąsiadów (wygrywa większość, a w przypadku remisu klasa przypisywana jest losowo).

Składnia metody k-NN

z użyciem funkcji `knn()` z pakietu **class**

Budowa klasyfikatora i predykcja:

```
p <- knn(train, test, class, k)
```

- `train` jest ramką danych zawierającą numeryczny zbiór uczący
- `test` jest ramką danych zawierającą numeryczny zbiór testowy
- `class` jest wektorem czynników z klasami dla przykładów ze zbioru uczącego
- `k` jest liczbą całkowitą oznaczającą liczbą sąsiadów w metodzie k-NN

Funkcja zwraca wektor przewidzianych klas dla przykładów ze zbioru testowego.

Przykład:

```
wbcd_pred <- knn(train = wbcd_train, test = wbcd_test,
                 cl = wbcd_train_labels, k = 3)
```

KROK 3. BUDOWA MODELU

Zadanie 4. Używając funkcji `knn()`, przeprowadzić klasyfikację przykładów ze zbioru testowego. Utworzyć nową zmienną z wynikami. Ponieważ dane uczące składają się z 469 przykładów, spróbować $k = 21$ (liczba nieparzysta w przybliżeniu równa $\sqrt{469}$). Jaki wpływ na głosowanie w analizowanym przykładzie będzie miało wybranie nieparzystej liczby sąsiadów?

KROK 4. OCENA MODELU

- Aby sprawdzić, w jakim stopniu wyniki zapisane w *wbcd_test_pred* pokrywają się z rzeczywistymi klasami (zapisanymi w *wbcd_test_labels*), użyjemy funkcji `CrossTable()` z pakietu **gmodels**.
- Aby zainstalować na systemie Windows pakiet **gmodels**, może być wymagana instalacja oprogramowania **RTools** (`https://cran.rstudio.com/bin/windows/Rtools/rtools40.html`).
- Po instalacji **RTools** trzeba go jeszcze umieścić na `PATH`. Najłatwiej utworzyć w katalogu roboczym plik tekstowy o nazwie **.Renviron**, który zawiera następującą linię:

```
PATH="${RTOOLS40_HOME}\usr\bin;${PATH}"
```

- Następnie zrestartuj R i sprawdź:

```
> Sys.which("make")
                                  make
"C:\\rtools40\\usr\\bin\\make.exe"
```

KROK 4. OCENA MODELU

Zadanie 5. Używając funkcji `CrossTable()`, stworzyć **tabelę krzyżową** (ang. *cross tabulation*) przedstawiającą, w jakim stopniu przewidziane za pomocą modelu 21-NN klasy pokrywają się z prawdziwymi klasami przykładów ze zbioru testowego. Zinterpretować otrzymane wyniki (TP — wyniki prawdziwie dodatnie, FP — wyniki fałszywie dodatnie, FN — wyniki fałszywie ujemne, TN — wyniki prawdziwie ujemne).

KROK 5. DOPRACOWANIE MODELU

- Dwa ze 100 przykładów ze zbioru testowego zostały sklasyfikowane niepoprawnie, co wydaje się bardzo dobrym wynikiem jak na kilka linijek kodu w R. Pomimo to spróbujemy ten wynik poprawić (zwłaszcza, że błędy były niebezpiecznymi wynikami fałszywie ujemnymi).
- Sprawdzimy dwa proste podejścia:

  a) spróbujemy innych wartości k,

  b) zastosujemy inną metodę skalowania cech.

KROK 5. DOPRACOWANIE MODELU

Testowanie alternatywnych wartości parametru k.

Zadanie 6. Dla znormalizowanych uprzednio danych przetestować działanie algorytmu k-NN z różnymi wartościami parametru k.

a) Jak metoda k-NN zachowuje się w analizowanym przypadku w zależności od wartości parametru k?

b) Czy istnieje k, dla którego liczba wyników fałszywie negatywnych jest mniejsza niż dla $k = 21$? Jeśli tak, to jakim kosztem?

KROK 5. DOPRACOWANIE MODELU

Standaryzacja z-score.

- W przypadku standaryzacji z-score nie ma z góry zadanego ograniczenia na wartości przetransformowanych danych. Skrajne wartości w wyjściowym zbiorze danych nie są „ściągane" do odcinka $[0, 1]$, jak to ma miejsce w przypadku zwykłej normalizacji, dzięki czemu lepiej zachowują swój odróżniający je od innych danych charakter.

Zadanie 7.

a) Używając funkcji `scale()`, przeprowadzić na wyjściowych danych standaryzację z-score.

b) Aby upewnić się, że standaryzacja została przeprowadzona poprawnie, sprawdzić statystyki podsumowujące dla wybranych zmiennych.

c) Sprawdzić, jak działa model k-NN zbudowany na przetransformowanych za pomocą przekształcenia z-score danych (stworzyć zbiór uczący i zbiór testowy, przeprowadzić klasyfikację i przeanalizować otrzymane wyniki; sprawdzić różne k). Jak taki model działa w porównaniu do zbudowanego wcześniej modelu 21-NN na danych znormalizowanych do $[0, 1]$?

KROK 5. DOPRACOWANIE MODELU

Zadanie 8. — dodatkowe

a) Aby upewnić się, że algorytm k-NN z wybranym k dobrze będzie generalizował się na przyszłe dane, wybrać w sposób losowy kilka innych zbiorów testowych (po 100 pacjentów) i dla nowych podziałów przetestować działanie metody.

b) W szczególności tym samym sposobem sprawdzić zachowanie modelu 1-NN na danych znormalizowanych do $[0, 1]$, który dla rozważanego wcześniej zbioru testowego wykazywał się najmniejszą liczbą wyników fałszywie negatywnych.