**Klasyfikacja i grupowanie**

**Algorytm kNN w R**

Dane dotyczące raka piersi obejmują 569 obserwacji, zawierających dane z biopsji raka. Każda obserwacja zawiera 32 zmienne. Jedną z cech jest numer identyfikacyjny, kolejna to diagnoza raka, natomiast 30pozostałych, to liczbowe wyniki laboratoryjne. Diagnoza jest kodowana jako "M" - malignant - oznacza złośliwy lub "B" - benign - oznacza łagodny. Pozostałe 30 pomiarów numerycznych zawiera średnią(mean), błąd standardowy(se) i worst - średnia z 3 "najgorszych" pomiarów. Mamy

1. Promień
2. Tekstura
3. Obwód
4. Powierzchnia
5. Gładkość
6. Zwartość
7. Wklęsłość
8. Punkty wklęsłe
9. Symetria
10. Wymiar

Ładujemy plik:

wbcd <- read.csv("wisc\_bc\_data.csv", stringsAsFactors = FALSE)

Proste podsumowanie:

str(wbcd)

Usuwamy pierwszą kolumnę (ID)

wbcd <- wbcd[-1]

Zbadamy rozkład zmiennej diagnosis w zbiorze.

table(wbcd$diagnosis)

Zmieńmy typ danych w zmiennej celu (diagnosis), dodatkowo zmienimy nazwy etykiet w zmiennej diagnosis

wbcd$diagnosis<- factor(wbcd$diagnosis, levels = c("B", "M"),labels = c("łagodny", "złośliwy"))

Możemy zobaczyć efekt zmiany tym razem badając rozkład procentowy

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

Zobaczmy jak wyglądają wartości dla pierwszych trzech zmiennych - czy wymagana jest normalizacja?

summary(wbcd[c("radius\_mean", "area\_mean", "smoothness\_mean")])

I sposób skalowania - normalizacja.  
Napiszemy i przetestujemy funkcję normalizującą dane.

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

normalize(c(1, 2, 3, 4, 5, 6))

Następnie iteracyjnie stosujemy funkcję normalize dla kazdej zmiennej w zbiorze danych.

wbcd\_n <- as.data.frame(lapply(wbcd[2:31], normalize))

Aby sprawdzić efekt działania możemy zbadać rozkład jednej ze zmiennych:

> summary(wbcd\_n$area\_mean)

Budujemy zbiór treningowy i zbiór testowy

wbcd\_train <- wbcd\_n[1:469, ]

wbcd\_test <- wbcd\_n[470:569, ]

wbcd\_train\_diag <- wbcd[1:469, 1]

wbcd\_test\_diag <- wbcd[470:569, 1]

Odczytanie wartości zmniennej diagnosis, pozwoli ocenić model.

Funkcja knn() z pakietu class odpowiada za działanie algorytmu kNN.

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

gdzie

* train - data.frame z danymi treningowymi
* test - data.frame z danymi testowymi
* class - wektor(factor) z klasą decyzyjną
* k - liczba sąsiadów

library(class)

wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test,cl = wbcd\_train\_diag, k = 21)

Rezultatem działania funkcji kNN jest wektor, w którym zostały zwrócone wyniki klasyfikacji.

wbcd\_test\_pred

Ewaluacja modelu

library(gmodels)

CrossTable(x = wbcd\_test\_diag, y = wbcd\_test\_pred,prop.chisq=FALSE)

Otrzymalismy TP=21 , TN=77 ,FN=2 , FP=0.  
II sposób skalowania - standaryzacja.

wbcd\_z <- as.data.frame(scale(wbcd[-1]))

summary(wbcd\_z)

Budujemy zbiór treningowy i testowy

wbcd\_train\_z <- wbcd\_z[1:469, ]

wbcd\_test\_z <- wbcd\_z[470:569, ]

oraz model

wbcd\_test\_pred\_z <- knn(train = wbcd\_train\_z, test = wbcd\_test\_z,cl = wbcd\_train\_diag, k = 3)

CrossTable(x = wbcd\_test\_diag, y = wbcd\_test\_pred\_z,prop.chisq=FALSE)

Przetestuj różne wartości k.

**Algorytm k-średnich**

Zbiór snsdata.csv zawiera losowe wyniki badań około 30,000 ucznów szkół średnich(high school), którzy posiadają profile w sieciach społecznościowych (Social Networking Service). Zbiór zawiera 40 zmiennych: gender, age, friends, basketball, football, soccer, softball, volleyball,swimming, cute, sports, rock, god, church, hair, mall, clothes, hollister, drugs itp.

ds<- read.table(header = TRUE,"snsdata.csv",sep = ",")

str(ds)

Poznajmy zbiór danych. (Rattle lub R). Wykonaj histogram dla zmiennej gender. Uzupełnij brak danych w gender jako nieznany.

ds$gender<-ifelse(is.na(ds$gender), 3, ds$gender)

Kolejny problem to zmienna age.

summary(ds$age)

Mamy z tą zmienną dwa problemy - punkty oddalone(dane niepoprawne) oraz dane brakujące.  
Przede wszystkim odrzućmy dane niepoprawne

ds$age <- ifelse(ds$age >= 13 & ds$age < 20, ds$age, NA)

summary(ds$age)

Następnie usuwamy braki w danych. Pogrupujemy dane według roku ukończenia(gradyear) a następnie w grupach brakujące dane uzupełnimy średnią.

aggregate(data = ds, age ~ gradyear, mean, na.rm = TRUE)

ave\_age <- ave(ds$age, ds$gradyear, FUN =function(x) mean(x, na.rm = TRUE))

tail(ave\_age, n=200)

Funkcja ave() pogrupowała wyniki w zależności od gradyear, tworząc wektor, w którym dla każdego factora gradeyare została przypisana średnia w grupie.  
i NA zastępujemy obliczoną średnią.

ds$age <- ifelse(is.na(ds$age), ave\_age, ds$age)

summary(ds$age)

Algorytm k-średnich realizuje funkcja kmeans() z pakietu stats Najpierw znormalizujemy dane

interests <- ds[5:40]

interests\_z <- as.data.frame(lapply(interests, scale))

Teraz możemy już zastosować algorytm (dlaczego 5???)

teen\_clusters <- kmeans(interests\_z, 5)

Zanalizujmy wyniki

teen\_clusters$size

teen\_clusters$centers