Sprawozdanie z listy 3

Eksploracja danych

Daria Grzelak, 277533

2025-05-26

Spis treści

1 Krótki opis zagadnienia

Głównym tematem niniejszego raportu są rozmaite metody klasyfikacji. W pierwszej części przedstawiona zostanie klasyfikacja na bazie modelu regresji liniowej, natomiast w drugiej zostaną porównane trzy metody klasyfikacji: metoda k-najbliższych sąsiadów, drzewa klasyfikacyjne oraz naiwny klasyfikator Bayesowski.

Zanim przejdę do szerszego omówienia zagadnień, załaduję potrzebne pakiety.

```r
# Załadowanie potrzebnych pakietów
library(mlbench) # dane Glass
library(ipred) # k-nn, predykcje
library(rpart) # drzewa klasyfikacyjne
library(rpart.plot) # wykresy dla drzew klasyfikacyjnych
library(e1071) # naive Bayes
library(klaR) # Naive Bayes (z jądrową estymacją gęstości)
```

2 Klasyfikacja na bazie modelu regresji liniowej

W pierwszej części niniejszego raportu skupię się na jednej z metod klasyfikacji – na bazie modelu regresji liniowej.

2.1 Wykorzystane metody

Do wykonania klasyfikacji wykorzystam następujące metody i narzędzia:

- podział danych w sposób losowy,
- utworzenie modeli regresji liniowej dla każdej klasy osobno,
- prognozowanie etykietek na bazie utworzonych modeli,
- ocena jakości modelu poprzez macierz pomyłek,
- utworzenie modelu na podstawie rozszerzonej przestrzeni cech.

2.2 Opis danych

Dane, które będę analizować, to `iris` z R-pakietu datasets, zawierający obserwacje na temat trzech gatunków irysów.

```r
# Wczytanie danych
data(iris)
```

Dane te zawierają 150 obserwacji i 5 zmiennych.

Nazwa	Typ	Opis
Sepal.Length	Liczbowa	Długość działki kielicha
Sepal.Width	Liczbowa	Szerokość działki kielicha
Petal.Length	Liczbowa	Długość płatka
Petal.Width	Liczbowa	Szerokość płatka
Species	Jakościowa	Gatunek

Tabela 1: Opis cech zawartych w danych iris

W tabeli 1 umieszczam opis cech zawartych w tych danych.

2.3 Podział danych na zbiory uczący i testowy

Aby utworzyć model regresji liniowej i na jego podstawie dopasowywać etykietki klas do danych, podzielę je na dwa podzbiory: uczący (2/3 obserwacji) i testowy (lub treningowy; 1/3 obserwacji). Przed losowaniem natomiast ustawię ziarno generatora, aby otrzymać powtarzalność wyników.

```
## Ziarno generatora
set.seed(123)

## Podział na podzbiory uczący i testowy
# Wylosowanie 100 numerów
x <- sample(1:150, 100)
# Posortowanie wylosowanych numerów
x <- sort(x)

# Podział na podzbiory według numerów
iris.learn <- iris[x,]
iris.train <- iris[-x,]
```

Następnie, aby utworzyć modele regresji liniowej dla poszczególnych klas, należy utworzyć dodatkowe zmienne wskazujące, czy dana obserwacja należy do konkretnego gatunku (1), czy nie (0). Potrzebne są trzy zmienne dla każdego gatunku.

```
## Odpowiednie zmienne wskaźnikowe dla klas ze zbioru uczącego
# Liczby wystąpień poszczególnych klas
numbers <- tabulate(as.numeric(iris.learn$Species))

# Wskaźniki
Y1 <- c(rep(1,numbers[1]),rep(0,100-numbers[1]))
Y2 <- c(rep(0,numbers[1]),rep(1,numbers[2]),rep(0,numbers[3]))
Y3 <- c(rep(0,100-numbers[3]),rep(1,numbers[3]))

# Zastąpienie nazw klas etykietkami (należy do klasy/nie należy do klasy)
iris1 <- cbind(iris.learn[,1:4],Y1) # setosa
iris2 <- cbind(iris.learn[,1:4],Y2) # versicolor
iris3 <- cbind(iris.learn[,1:4],Y3) # virginica
```

W wyniku takiego podziału utworzone zostały trzy zbiory stanowiące modyfikacje zbioru `iris.learn` – zamiast etykietek klas (gatunków) dodane zostały zmienne określające przynależność do wybranego gatunku.

2.4 Konstrukcja klasyfikatora i wyznaczenie prognoz

W następnym kroku przechodzę do konstrukcji klasyfikatora. Posłużę się trzema niezależnymi klasyfikatorami, z czego każdy z nich określa prawdopodobieństwo tego, że dana obserwacja należy do wybranego gatunku.

```r
# Dopasowanie 3 niezależnych modeli regresji liniowej
iris1.lm <- lm(Y1~., data=iris1)
iris2.lm <- lm(Y2~., data=iris2)
iris3.lm <- lm(Y3~., data=iris3)
```

Po utworzeniu tych trzech modeli dokonam prognozy na jego podstawie dla zbiorów uczącego i treningowego. Przewidywania wyznaczają prawdopodobieństwo tego, że dana obserwacja należy do wybranej klasy. Następnie porównuję prawdopodobieństwa i dla każdej obserwacji przypisuję tę etykietkę klasy, dla której prawdopodobieństwo jest największe. Taką klasyfikację powtarzam dla zbioru uczącego i treningowego.

```r
## Prognozowanie dla zbioru uczącego
# prognozowanie zmiennej zależnej (prognozowane p-stwo a posteriori)
pred1.lm <- predict(iris1.lm, iris1)
pred2.lm <- predict(iris2.lm, iris2)
pred3.lm <- predict(iris3.lm, iris3)

# zamiana p-stw a posteriori na etykietki klas
# (kodowanie klas: 1-setosa, 2-versicolor, 3-virginica)
pred.lm <- rep(1, length(pred1.lm))
pred.lm[pred2.lm > pmax(pred1.lm,pred3.lm)] = 2
pred.lm[pred3.lm > pmax(pred1.lm,pred2.lm)] = 3

# Prognozowanie dla zbioru treningowego
# prognoza
pred.train1.lm <- predict(iris1.lm, iris.train)
pred.train2.lm <- predict(iris2.lm, iris.train)
pred.train3.lm <- predict(iris3.lm, iris.train)

# zamiana prawdopodobieństw na etykietki klas
pred.train.lm <- rep(1, length(pred.train1.lm))
pred.train.lm[pred.train2.lm > pmax(pred.train1.lm,pred.train3.lm)] = 2
pred.train.lm[pred.train3.lm > pmax(pred.train1.lm,pred.train2.lm)] = 3
```

2.5 Ocena jakości modelu

Aby ocenić jakość modelu, wykorzystam macierz pomyłek oraz błąd klasyfikacji dla zbiorów uczącego oraz testowego.

```r
## Zbiór uczący
# Macierz pomyłek
conf.matrix.iris.learn <- table(pred.lm, iris.learn$Species)
# Dokładność
accuracy.iris.learn <-
  sum(diag(conf.matrix.iris.learn))/sum(conf.matrix.iris.learn)
```

```
# Błąd klasyfikacji
error.iris.learn <- 1 - accuracy.iris.learn

## Zbiór testowy
# Macierz pomyłek
conf.matrix.iris.train <- table(pred.train.lm, iris.train$Species)
# Dokładność
accuracy.iris.train <-
  sum(diag(conf.matrix.iris.train))/sum(conf.matrix.iris.train)
# Błąd klasyfikacji
error.iris.train <- 1 - accuracy.iris.train
```

Macierz pomyłek dla zbioru uczącego prezentuje się następująco:

```
##
## pred.lm setosa versicolor virginica
##       1     30          0         0
##       2      0         32         6
##       3      0          7        25
```

Z wyliczeń wynika natomiast, że dokładność modelu dla zbioru uczącego wynosi 0.78, zatem błąd klasyfikacji wynosi 0.22.

Analogicznie, macierz pomyłek dla zbioru testowego prezentuje się następująco:

```
##
## pred.train.lm setosa versicolor virginica
##             1     20          0         0
##             2      0          9         4
##             3      0          2        15
```

Z wyliczeń wynika, że dokładność modelu dla zbioru testowego wynosi 0.8, zatem błąd klasyfikacji wynosi 0.2.

2.6 Model liniowy dla rozszerzonej przestrzeni cech

W następnej kolejności zbadam, jak rozszerzenie przestrzeni cech o iloczyny poszczególnych zmiennych objaśniających wpłynie na jakość modelu.

```
## Utworzenie rozszerzonego zbioru uczącego
# Kopia obecnego zbioru
iris.learn.new <- iris.learn
# Skrócenie nazw
names(iris.learn.new) <- c("SL","SW","PL","PW","Spec")
# Dodanie wielomianów
iris.learn.new <- transform(iris.learn.new, SL.SW=SL*SW,  PL.PW=PL*PW,
                            PL.SL=PL*SL, PL.SW=PL*SW, PW.SW=PW*SW,
```

```
                              PW.SL=PW*SL, PL2=PL*PL, PW2=PW*PW, SL2=SL*SL,
                              SW2=SW*SW)

# Zastąpienie nazw klas etykietkami (należy do klasy/nie należy do klasy)
iris1.new <- cbind(iris.learn.new[,-5],Y1)
iris2.new <- cbind(iris.learn.new[,-5],Y2)
iris3.new <- cbind(iris.learn.new[,-5],Y3)

## Utworzenie rozszerzonego zbioru treningowego
# Kopia obecnego zbioru
iris.train.new <- iris.train
# Skrócenie nazw
names(iris.train.new) <- c("SL","SW","PL","PW","Spec")
# Dodanie wielomianów
iris.train.new <- transform(iris.train.new, SL.SW=SL*SW,  PL.PW=PL*PW,
                              PL.SL=PL*SL, PL.SW=PL*SW, PW.SW=PW*SW,
                              PW.SL=PW*SL, PL2=PL*PL, PW2=PW*PW, SL2=SL*SL,
                              SW2=SW*SW)
```

Następnie dopasuję modele regresji liniowej i na ich podstawie będę prognozować etykietki klas, dokładnie tak samo jak wcześniej.

```
## Dopasowanie K=3 niezależnych modeli regresji liniowej
iris1.lm.new <- lm(Y1~., data=iris1.new)
iris2.lm.new <- lm(Y2~., data=iris2.new)
iris3.lm.new <- lm(Y3~., data=iris3.new)

## Prognozowanie dla zbioru uczącego
# prognozowanie zmiennej zależnej (prognozowane p-stwo a posteriori)
pred1.lm.new  <- predict(iris1.lm.new, iris1.new)
pred2.lm.new  <- predict(iris2.lm.new, iris2.new)
pred3.lm.new  <- predict(iris3.lm.new, iris3.new)

# zamieniamy p-stwa a posteriori na etykietki klas
# (kodowanie klas: 1-setosa, 2-versicolor, 3-virginica)
pred.learn.new <- rep(1, length(pred1.lm.new) )
pred.learn.new[pred2.lm.new > pmax(pred1.lm.new,pred3.lm.new)] = 2
pred.learn.new[pred3.lm.new > pmax(pred1.lm.new,pred2.lm.new)] = 3

# macierz pomyłek
conf.matrix.iris.learn.new <- table(pred.learn.new, iris.learn.new$Spec)
accuracy.iris.learn.new <-
  sum(diag(conf.matrix.iris.learn.new))/sum(conf.matrix.iris.learn.new)
error.iris.learn.new <- 1 - accuracy.iris.learn.new
```

```
## Prognoza dla zbioru treningowego
# Prognozowanie dla zbioru treningowego
pred.train1.lm.new <- predict(iris1.lm.new, iris.train.new)
pred.train2.lm.new <- predict(iris2.lm.new, iris.train.new)
pred.train3.lm.new <- predict(iris3.lm.new, iris.train.new)

# zamieniamy p-stwa a posteriori na etykietki klas
# (kodowanie klas: 1-setosa, 2-versicolor, 3-virginica)
pred.train.new <- rep(1, length(pred.train1.lm.new) )
pred.train.new[pred.train2.lm.new >
                 pmax(pred.train1.lm.new,pred.train3.lm.new)] = 2
pred.train.new[pred.train3.lm.new >
                 pmax(pred.train1.lm.new,pred.train2.lm.new)] = 3

# macierz pomyłek
conf.matrix.iris.train.new <- table(pred.train.new, iris.train.new$Spec)
accuracy.iris.train.new <-
  sum(diag(conf.matrix.iris.train.new))/sum(conf.matrix.iris.train.new)
error.iris.train.new <- 1 - accuracy.iris.train.new
```

Tak jak wcześniej, przyjrzę się macierzom pomyłek oraz błędom klasyfikacji dla zbiorów uczącego i testowego.

Macierz pomyłek dla zbioru uczącego prezentuje się następująco:

```
##
## pred.learn.new setosa versicolor virginica
##              1     34          0         0
##              2      0         29         5
##              3      0          0        32
```

Z wyliczeń wynika natomiast, że dokładność modelu dla zbioru uczącego wynosi 0.95, zatem błąd klasyfikacji wynosi 0.05.

Analogicznie, macierz pomyłek dla zbioru testowego prezentuje się następująco:

```
##
## pred.train.new setosa versicolor virginica
##              1     16          0         0
##              2      0         21         1
##              3      0          0        12
```

Z wyliczeń wynika, że dokładność modelu dla zbioru testowego wynosi 0.98, zatem błąd klasyfikacji wynosi 0.02.

2.7 Wnioski

W obu modelach dokładność jest dość wysoka i występują niewielkie różnice pomiędzy zbiorem uczącym a testowym (do 0,1), na korzyść tego drugiego. Jednak model rozszerzony daje dużo wyższą dokładność – dla zbioru testowego źle przypisana została jedynie jedna obserwacja – co znaczy, że jest dużo skuteczniejszy od modelu zwykłego.

W zwykłym modelu natomiast częściowo zdaje się zachodzić zjawisko maskowania klas – dla zbioru uczącego około połowa obserwacji z klasy środkowej zostaje błędnie przypisana do klasy trzeciej, a dla zbioru testowego – około 1/3.

Co ciekawe, dla obu modeli wszystkie obserwacje z gatunku setosa zostają poprawnie do niego przypisanego. Natomiast w modelu rozszerzonym wszystkie błędne klasyfikacje to obserwacje z gatunku virginica, które model przypisuje niepoprawnie do klasy versicolor. Oznacza to najprawdopobniej, że gatunek setosa ogólnie łatwiej odróżnić od pozostałych dwóch gatunków.

3 Porównanie metod klasyfikacji

W drugiej części niniejszego sprawozdania porównam ze sobą trzy inne metody klasyfikacji. Będą to metoda k-najbliższych sąsiadów, drzewa klasyfikacyjne oraz naiwny klasyfikator Bayesowski.

3.1 Wykorzystane metody

W tej części sprawozdania wykorzystam następujące metody:

- metody analizy opisowej do wstępnego zbadania danych,
- metody klasyfikacyjne (k-najbliższych sąsiadów, drzewa klasyfikacyjne, naiwny klasyfikator Bayesowski),
- metody graficzne do prezentacji wyników (wykresy),
- macierze pomyłek i błędy klasyfikacji,
- metody oceny dokładności klasyfikacji (cross-validation, bootstrap, 632plus).

3.2 Opis danych

Analizowanymi danymi będą dane `Glass` z pakietu `mlbench`.

```
# Wczytanie danych
data(Glass)
```

Analizowane dane mają 214 obserwacji oraz 10 zmiennych.

Nazwa	Typ	Opis
RI	Liczbowa	Współczynnik załamania
Na	Liczbowa	Zawartość procentowa sodu
Mg	Liczbowa	Zawartość procentowa magnezu
Al	Liczbowa	Zawartość procentowa glinu
Si	Liczbowa	Zawartość procentowa krzemu
K	Liczbowa	Zawartość procentowa potasu
Ca	Liczbowa	Zawartość procentowa wapnia
Ba	Liczbowa	Zawartość procentowa baru
Fe	Liczbowa	Zawartość procentowa żelaza
Type	Jakościowa	Typ szkła

Tabela 2: Opis cech zawartych w danych Glass

Tabela 2 zawiera opis zmiennych.

Zmienną określającą klasy jest zmienna `Type`.

```
# Tabela dla zmiennej Type
table(Glass$Type)
```

```
##
##  1  2  3  5  6  7
## 70 76 17 13  9 29
```

Zawiera ona 6 klas określających typy szkła.

Nie występują obserwacje nietypowe ani brakujące, a wszystkie zmienne od razu miały przypisane poprawnie typy (wszystkie poza Type są liczbowe, Type jest jakościowa).

3.3 Wstępna analiza danych

Przed porównaniem metod klasyfikacji dokonam wstępnej analizy danych.

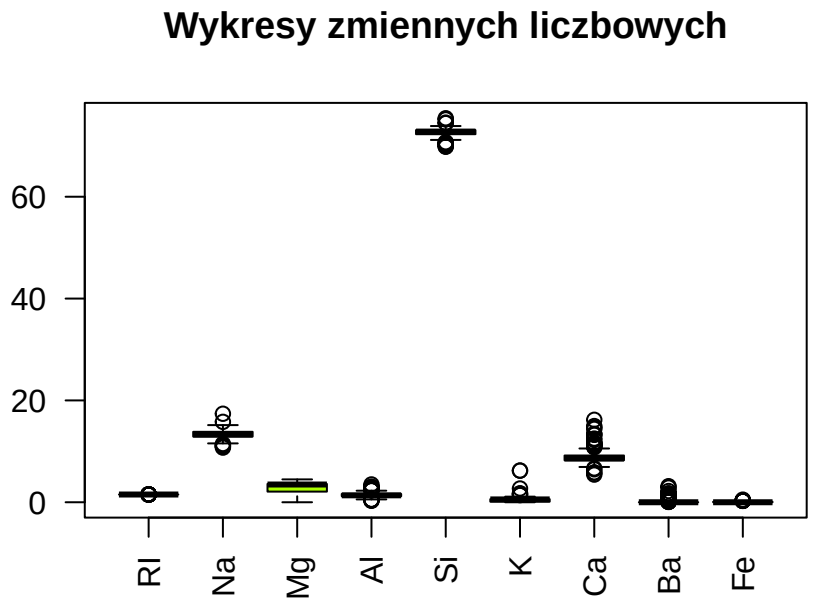
Przywołam jeszcze raz tabelę przedstawiającą przynależność poszczególnych obserwacji do klas.

```
# Tabela dla zmiennej Type
table(Glass$Type)
```

```
##
##  1  2  3  5  6  7
## 70 76 17 13  9 29
```

Zaobserwować można dość istotne dysproporcje pomiędzy przynależnością elementów do poszczególnych klas (najliczniejsza, druga, zawiera 76 obserwacji, a najmniej liczna, szósta, tylko 9). Gdyby wszystkie elementy przypisać do klasy drugiej, błąd klasyfikacji wynosiłby 0.6448598.

Następnie porównam wartości poszczególnych zmiennych, aby ocenić, czy konieczna może być standaryzacja.

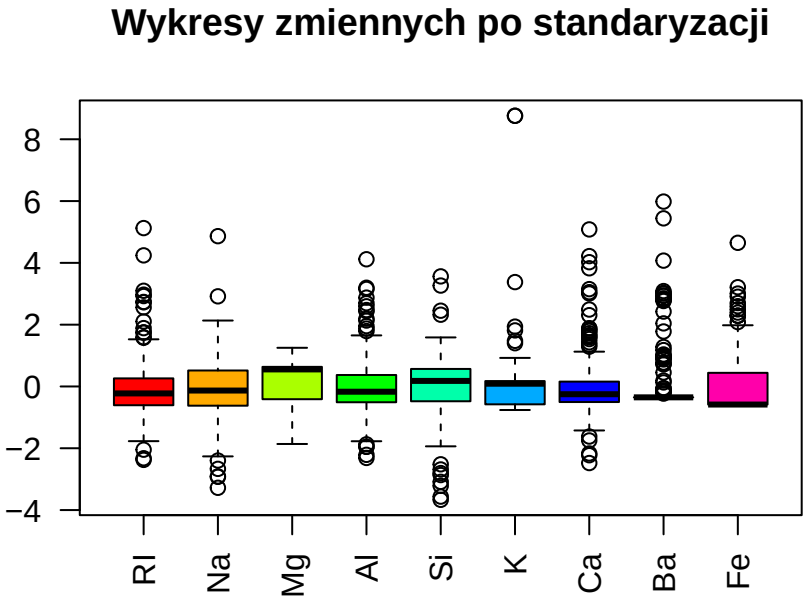


Rysunek 1: Wykresy pudełowe wartości poszczególnych zmiennych liczbowych z danych Glass

Na wykresie 1 łatwo można zauważyć, że poszczególne zmienne dość znacząco różnią się pomiędzy sobą, więc na potrzeby metody k-najbliższych sąsiadów zastosuję standaryzację.

```
# Standaryzacja
Glass.scaled <- cbind(scale(Glass[,1:9]), Glass$Type)
# Dostosowanie nowego obiektu
Glass.scaled <- as.data.frame(Glass.scaled)
names(Glass.scaled)[10] <- "Type"
Glass.scaled$Type <- as.factor(Glass.scaled$Type)
```

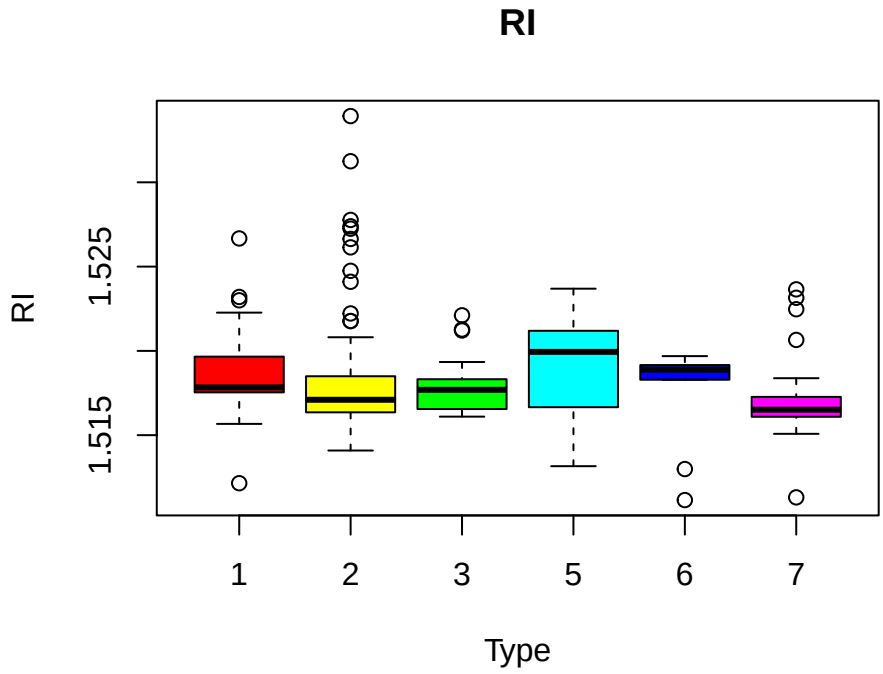
Zobaczę, jak po standaryzacji prezentują się poszczególne zmienne.



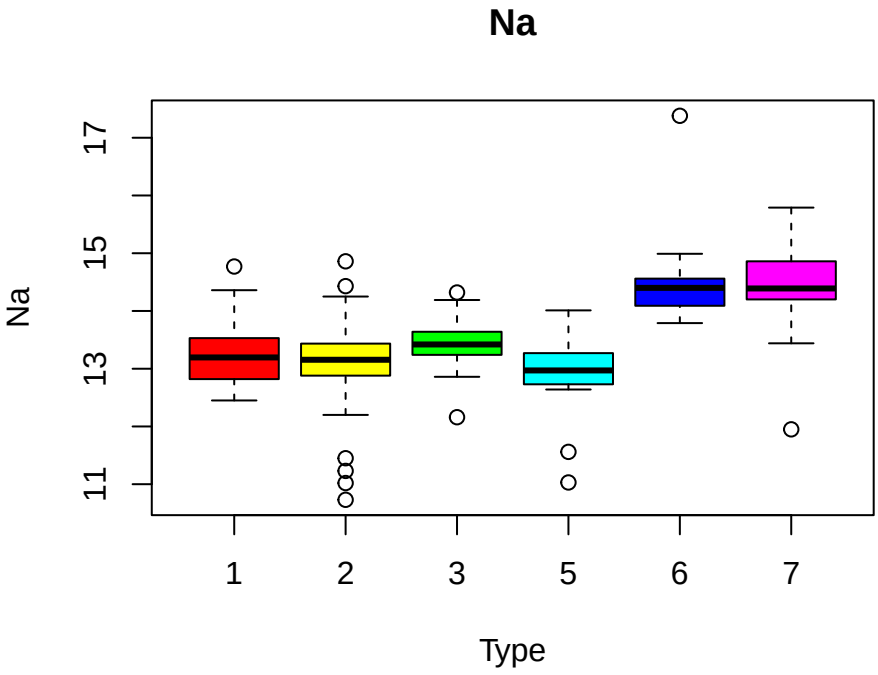
Rysunek 2: Wykresy pudełowe wartości poszczególnych zmiennych liczbowych z danych Glass.scaled

Wykres 2 pokazuje, że dane ustandarysowane rzeczywiście są bardziej jednolite.

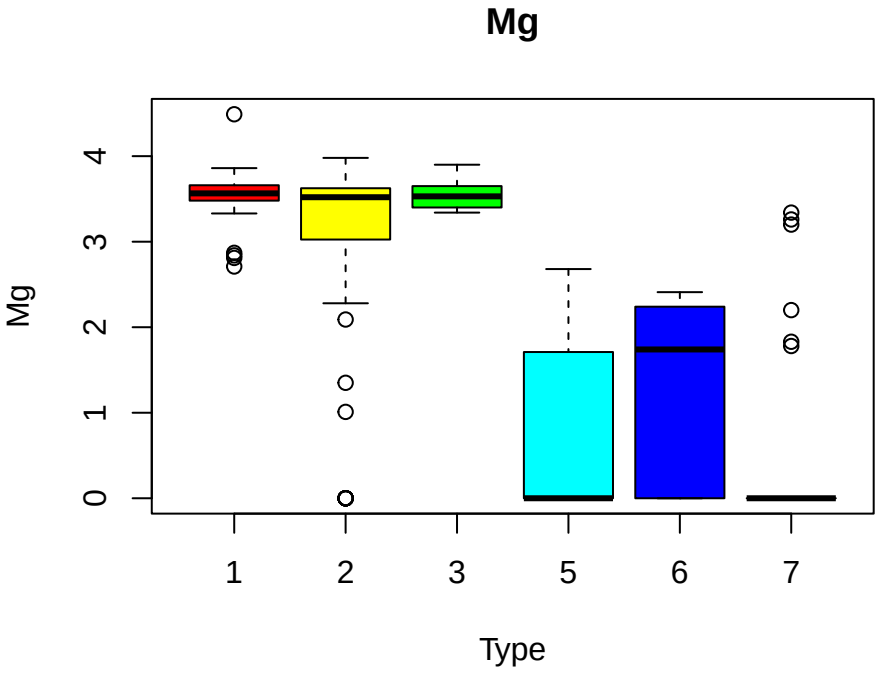
Ostatnim elementem będzie ocena zdolności dyskryminacyjnych dla poszczególnych zmiennych.



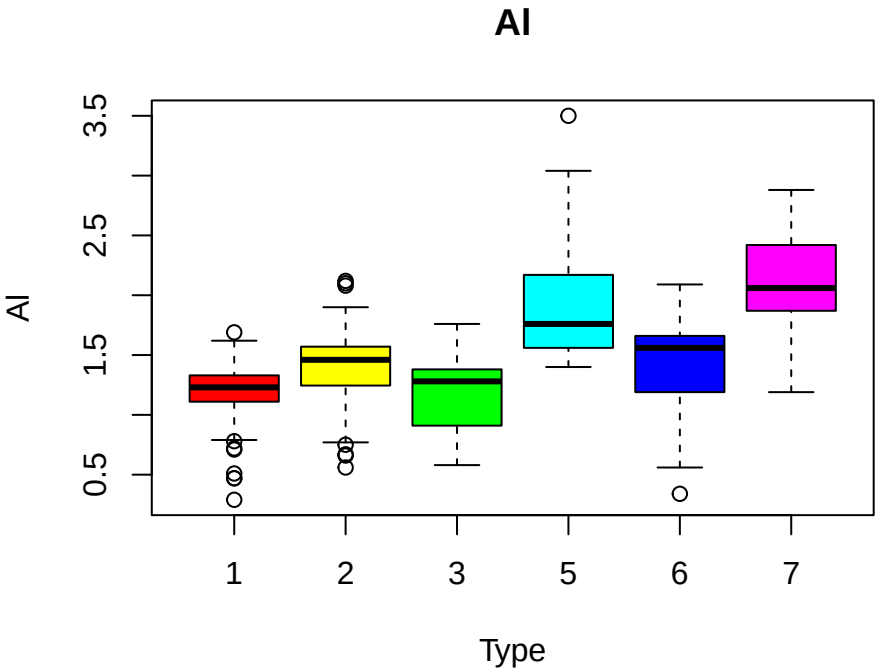
Rysunek 3: Ocena zdolności dyskryminacyjnych zmiennej RI



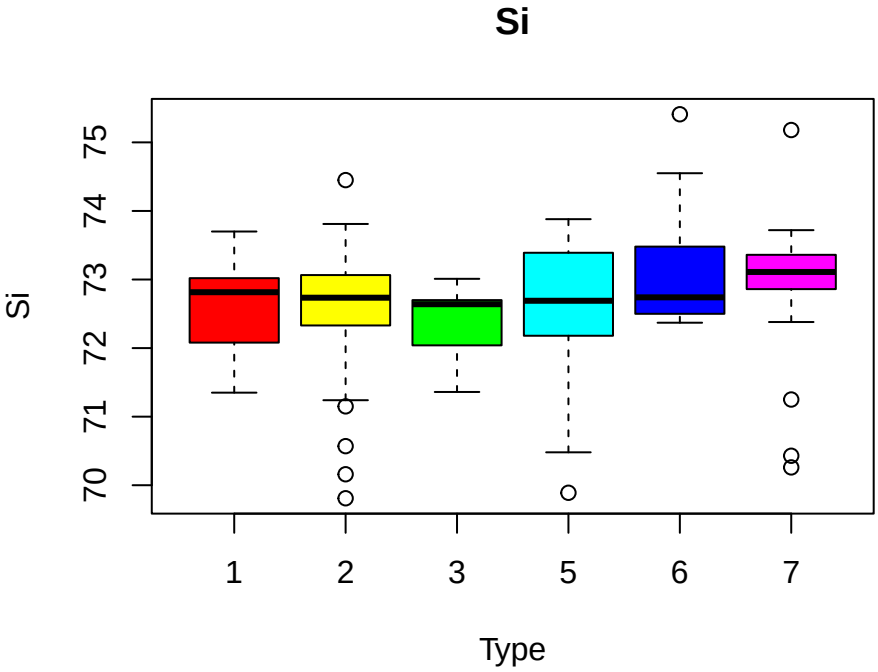
Rysunek 4: Ocena zdolności dyskryminacyjnych zmiennej Na



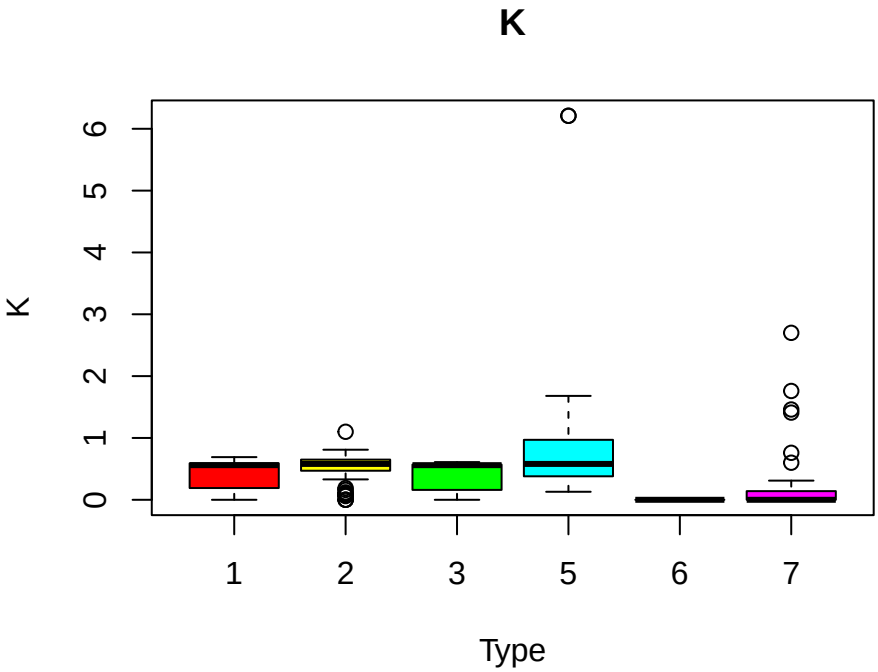
Rysunek 5: Ocena zdolności dyskryminacyjnych zmiennej Mg



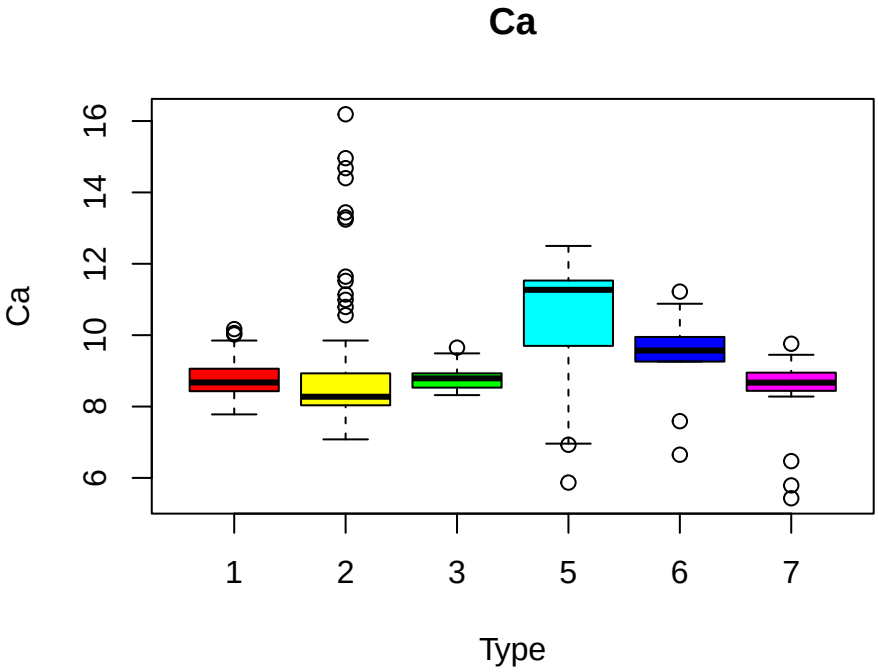
Rysunek 6: Ocena zdolności dyskryminacyjnych zmiennej Al



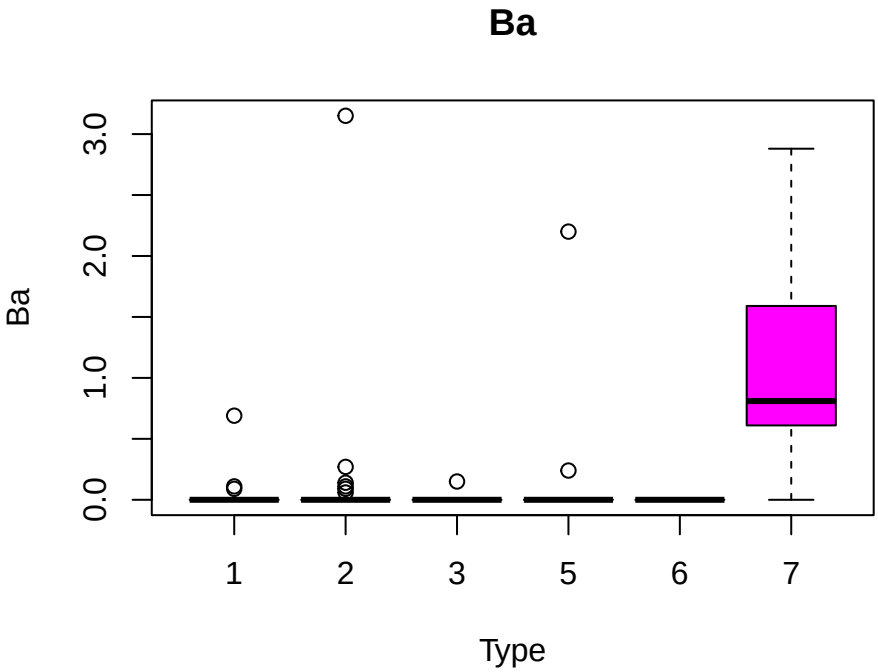
Rysunek 7: Ocena zdolności dyskryminacyjnych zmiennej Si



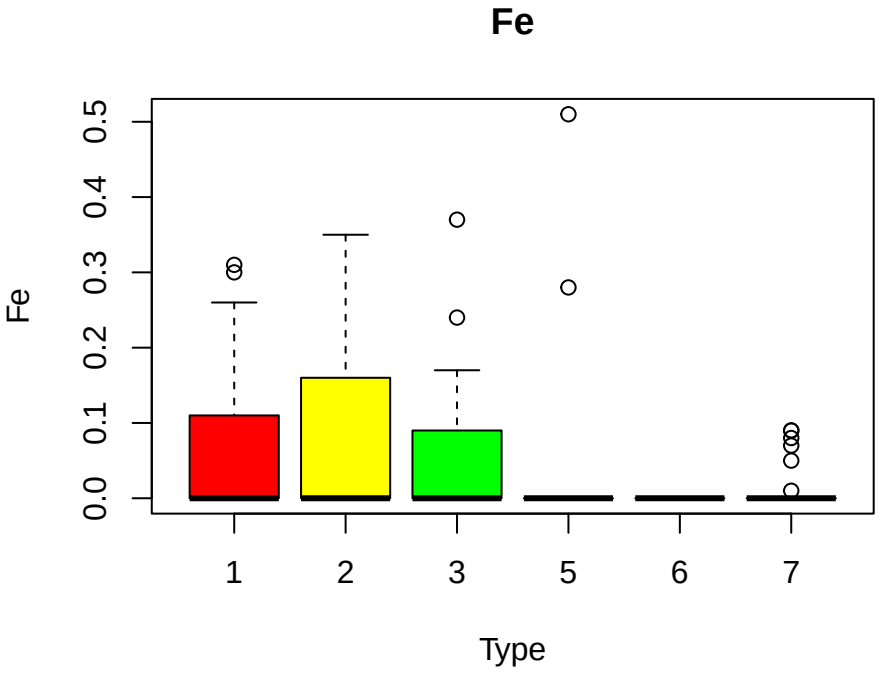
Rysunek 8: Ocena zdolności dyskryminacyjnych zmiennej K



Rysunek 9: Ocena zdolności dyskryminacyjnych zmiennej Ca



Rysunek 10: Ocena zdolności dyskryminacyjnych zmiennej Ba



Rysunek 11: Ocena zdolności dyskryminacyjnych zmiennej Fe

Na podstawie wykresów pudełkowych dla poszczególnych zmiennych można stwierdzić, że szczególnie obiecującymi zmiennymi są `RI`, `Na` i `Al`. Zmienne `Mg`, `Fe` i `Ba` mają natomiast potencjał do rozróżnienia grup klas.

3.4 Podział danych na pozbiory uczący i testowy

W późniejszym toku testów będę dzielić dane wielokrotnie na zbiory uczący i testowy, jednak na samym początku będę testować metody dla z góry określonego podziału.

```
## Losowy podział na pozdbiory uczący i testowy
glass.obs <- dim(Glass)[1]
glass.learn.index <- sample(1:glass.obs,2/3*glass.obs)

# utworzenie zbiorów uczącego i testowego
glass.learn <- Glass[glass.learn.index,]
glass.train <- Glass[-glass.learn.index,]

# rzeczywiste etykietki
etykietki.rzecz.learn <- glass.learn$Type
etykietki.rzecz.train <- glass.train$Type

# wymiary zbiorów
glass.obs.learn <- dim(glass.learn)[1]
glass.obs.train <- dim(glass.train)[1]

## Podział dla danych ustandaryzowanych
glass.scaled.learn <- Glass.scaled[glass.learn.index,]
glass.scaled.train <- Glass.scaled[-glass.learn.index,]
```

3.5 Metoda k-najbliższych sąsiadów

Metoda ta polega na znalezieniu k sąsiadów o najmniejszej odległości (euklidesowej) i na podstawie tego, do której klasy należy najwięcej z tych sąsiadów, testowany obiekt otrzymuje etykietkę.

3.5.1 Utworzenie modelu i wyznaczenie prognozowanych etykietek

Na sam początek przetestuję prosty model na podstawie wyznaczonych wcześniej zbiorów uczącego i testowego, a jako liczbę sąsiadów wybiorę 5. Test jest przeprowadzony dla danych nieustandaryzowanych.

```
# Utworzenie podstawowego modelu
model.knn.1 <- ipredknn(Type ~ ., data=glass.learn, k=5)

# Predykcje dla zbioru uczącego
glass.knn.etykietki.prog.learn <-
  predict(model.knn.1, glass.learn, type="class")

# Predykcje dla zbioru testowego
glass.knn.etykietki.prog.train <-
```

```
  predict(model.knn.1, glass.train, type="class")
```

3.5.2 Macierze pomyłek

Kolejnym krokiem jest ocena dokładności modelu poprzez przeanalizowanie macierzy pomyłek i błędów klasyfikacji dla zbiorów uczącego i testowego.

```
## Zbiór uczący
# Macierz pomyłek
conf.matrix.glass.knn1.learn <-
  table(glass.knn.etykietki.prog.learn, etykietki.rzecz.learn)

# Dokładność
glass.knn1.learn.accuracy <-
  sum(diag(conf.matrix.glass.knn1.learn))/glass.obs.learn

# Błąd klasyfikacji
glass.knn1.learn.error <- 1 - glass.knn1.learn.accuracy

## Zbiór testowy
# Macierz pomyłek
conf.matrix.glass.knn1.train <-
  table(glass.knn.etykietki.prog.train, etykietki.rzecz.train)

# Dokładność
glass.knn1.train.accuracy <-
  sum(diag(conf.matrix.glass.knn1.train))/glass.obs.train

# Błąd klasyfikacji
glass.knn1.train.error <- 1 - glass.knn1.train.accuracy
```

Macierz pomyłek dla zbioru uczącego prezentuje się następująco:

```
##                                etykietki.rzecz.learn
## glass.knn.etykietki.prog.learn  1  2  3  5  6  7
##                               1 34  6  5  0  0  1
##                               2  6 43  2  2  2  1
##                               3  2  0  4  0  0  0
##                               5  0  1  0  8  0  0
##                               6  0  1  0  0  3  1
##                               7  0  0  0  2  1 17
```

Dokładność tej klasyfikacji wynosi 0.7676056, zatem błąd klasyfikacji wynosi 0.2323944.

Analogicznie, dla zbioru testowego, macierz pomyłek prezentuje się następująco:

```
##                                etykietki.rzecz.train
```

```
## glass.knn.etykietki.prog.train  1  2  3  5  6  7
##                               1 20  8  6  0  0  1
##                               2  7 16  0  0  1  2
##                               3  1  1  0  0  0  0
##                               5  0  0  0  0  0  0
##                               6  0  0  0  0  1  0
##                               7  0  0  0  1  1  6
```

Dokładność tej klasyfikacji wynosi 0.5972222, zatem błąd klasyfikacji wynosi 0.4027778.

Dla zbioru uczącego dokładność jest znacznie większa niż dla zbioru testowego, jednak i tak błąd jest bardzo wysoki – bliski ćwierci obserwacji. Dla zbioru testowego jest jeszcze większy, około 40%.

3.5.3 Model, predykcje i ocena dla danych ustandaryzowanych

Następnym krokiem będzie utworzenie analogicznego modelu dla danych ustandaryzowanych.

```
# Utworzenie podstawowego modelu
model.knn.scaled.1 <- ipredknn(Type ~ ., data=glass.scaled.learn, k=5)

# Predykcje dla zbioru uczącego
glass.knn.etykietki.prog.scaled.learn <-
  predict(model.knn.scaled.1, glass.scaled.learn, type="class")

# Predykcje dla zbioru testowego
glass.knn.etykietki.prog.scaled.train <-
  predict(model.knn.scaled.1, glass.scaled.train, type="class")

## Zbiór uczący
# Macierz pomyłek
conf.matrix.glass.knn1.scaled.learn <-
  table(glass.knn.etykietki.prog.scaled.learn, etykietki.rzecz.learn)

# Dokładność
glass.knn1.scaled.learn.accuracy <-
  sum(diag(conf.matrix.glass.knn1.scaled.learn))/glass.obs.learn

# Błąd klasyfikacji
glass.knn1.scaled.learn.error <- 1 - glass.knn1.scaled.learn.accuracy

## Zbiór testowy
# Macierz pomyłek
conf.matrix.glass.knn1.scaled.train <-
  table(glass.knn.etykietki.prog.scaled.train, etykietki.rzecz.train)
```

```
# Dokładność
glass.knn1.scaled.train.accuracy <-
  sum(diag(conf.matrix.glass.knn1.scaled.train))/glass.obs.train

# Błąd klasyfikacji
glass.knn1.scaled.train.error <- 1 - glass.knn1.scaled.train.accuracy
```

Macierz pomyłek dla zbioru uczącego w tym wypadku prezentuje się następująco:

```
conf.matrix.glass.knn1.scaled.learn
```

```
##                                       etykietki.rzecz.learn
## glass.knn.etykietki.prog.scaled.learn  1  2  3  5  6  7
##                                     1 35  6  6  0  0  0
##                                     2  6 41  3  3  1  1
##                                     3  1  1  2  0  0  0
##                                     4  0  3  0  8  1  0
##                                     5  0  0  0  0  4  0
##                                     6  0  0  0  1  0 19
```

Dokładność klasyfikacji wynosi 0.7676056, a jej błąd – 0.2323944.

Macierz pomyłek dla zbioru testowego w tym wypadku prezentuje się następująco:

```
conf.matrix.glass.knn1.scaled.train
```

```
##                                       etykietki.rzecz.train
## glass.knn.etykietki.prog.scaled.train  1  2  3  5  6  7
##                                     1 24  9  4  0  0  1
##                                     2  4 16  2  0  0  2
##                                     3  0  0  0  0  0  0
##                                     4  0  0  0  0  0  0
##                                     5  0  0  0  0  2  0
##                                     6  0  0  0  1  1  6
```

Dokładność klasyfikacji wynosi 0.6666667, a jej błąd – 0.3333333.

Względem danych nieprzeskalowanych nie zmieniła się dokładność dla zbioru uczącego, lecz nieco poprawiła się dla zbioru testowego. Nadal nie daje to jednak zadowalających wyników.

3.5.4 Zaawansowane schematy oceny dokładności

Kolejnym krokiem będzie sprawdzenie dokładności przy pomocy zaawansowanych schematów. Ocenę wykonam na podstawie danych przeskalowanych, gdyż w poprzednim teście dały nieco niższy poziom błędu.

```
# Utworzenie własnej funkcji
my.predict  <-
  function(model, newdata) predict(model, newdata=newdata, type="class")
```

```
my.ipredknn <-
  function(formula1, data1, ile.sasiadow)
    ipredknn(formula=formula1,data=data1,k=ile.sasiadow)

## Porównanie błędów klasyfikacji dla ustalonej liczby sąsiadów
## (metody: 10-fold cross-validation, bootstrap, .632plus)
#CV
glass.knn.cv <- errorest(Type ~., Glass.scaled, model=my.ipredknn,
                         predict=my.predict, estimator="cv",
                         est.para=control.errorest(k = 10), ile.sasiadow=5)

#Boot
glass.knn.boot <- errorest(Type ~., Glass.scaled, model=my.ipredknn,
                           predict=my.predict, estimator="boot",
                           est.para=control.errorest(nboot = 50),
                           ile.sasiadow=5)

#.632plus
glass.knn.632plus <- errorest(Type ~., Glass.scaled, model=my.ipredknn,
                              predict=my.predict, estimator="632plus",
                              est.para=control.errorest(nboot = 50),
                              ile.sasiadow=5)

## Wyświetlenie wyników
glass.knn.cv
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass.scaled,
##     model = my.ipredknn, predict = my.predict, estimator = "cv",
##     est.para = control.errorest(k = 10), ile.sasiadow = 5)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.3318
```

```
glass.knn.boot
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass.scaled,
##     model = my.ipredknn, predict = my.predict, estimator = "boot",
##     est.para = control.errorest(nboot = 50), ile.sasiadow = 5)
##
##   Bootstrap estimator of misclassification error
```

```
##   with 50 bootstrap replications
##
## Misclassification error:  0.365
## Standard deviation: 0.0062
```

```
glass.knn.632plus
```

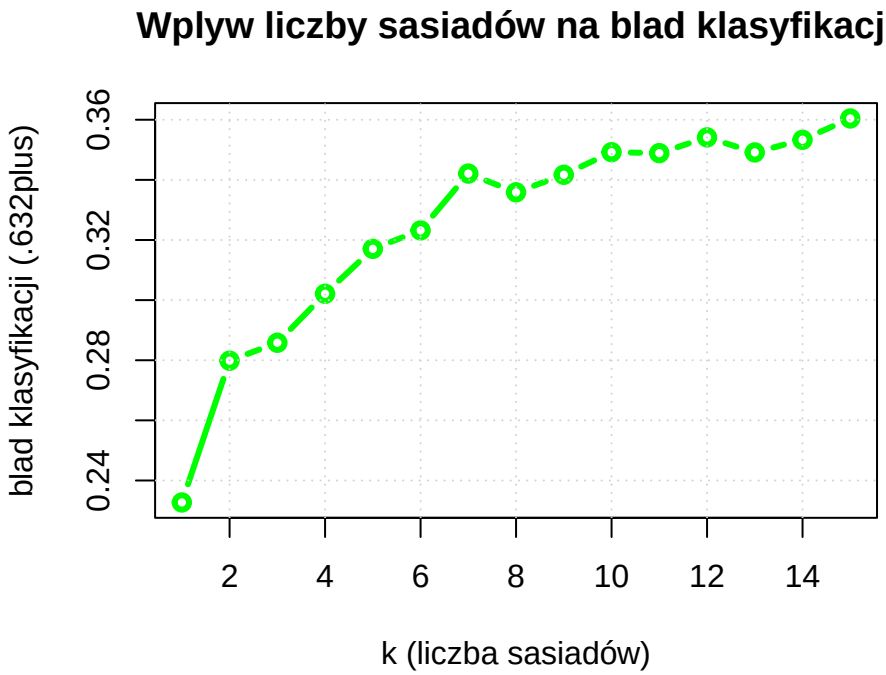
```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass.scaled,
##     model = my.ipredknn, predict = my.predict, estimator = "632plus",
##     est.para = control.errorest(nboot = 50), ile.sasiadow = 5)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.3162
```

Błędy oscylują na podobnym, dość wysokim poziomie, jednak metoda .632plus daje najmniejszy ze wszystkich błędów. Największy błąd daje natomiast metoda bootstrap.

3.5.5 Wpływ liczby sąsiadów na błąd

Kolejnym elementem, który może wpłynąć na wyniki, jest dobór liczby sąsiadów w modelu. Przetestuję od 1 do 15 sąsiadów, a stosowaną metodą walidacji będzie .632plus, gdyż w poprzednim teście dała najniższy poziom błędu. Skorzystam z danych ustandaryzowanych.

```
liczba.sasiadow.zakres <- 1:15
wyniki1 <-  sapply(liczba.sasiadow.zakres, function(k)
  errorest(Type ~., Glass.scaled, model=my.ipredknn, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.sasiadow=k)$error)
plot(liczba.sasiadow.zakres, wyniki1, type="b", col="green", lwd=3,
     main="Wpływ liczby sąsiadów na błąd klasyfikacji",
     xlab="k (liczba sąsiadów)", ylab="błąd klasyfikacji (.632plus)")
grid()
```



Rysunek 12: Wykres wpływu liczby sąsiadów na błąd klasyfikacji

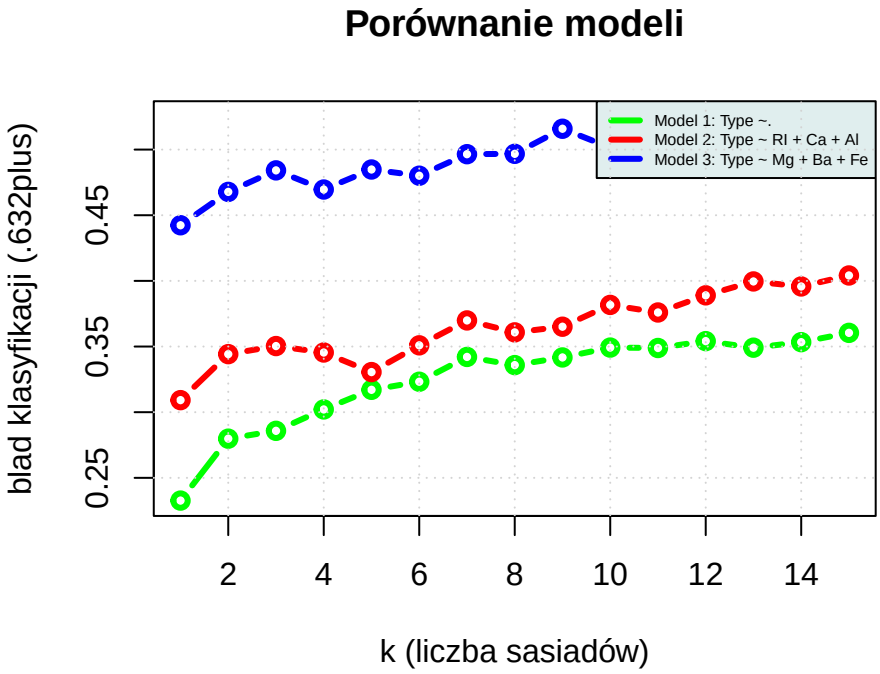
Wykres 12 pokazuje, że im więcej sąsiadów, tym błąd klasyfikacji wzrasta (z drobnymi wyjątkami, ale tendencja jest zdecydowanie rosnąca), jednak już dla jednego sąsiada błąd jest całkiem wysoki i wynosi 0.2326867.

3.5.6 Różne kombinacje zmiennych

Ostatnim testem będzie przetestowanie różnych kombinacji zmiennych. Pod lupę wezmę kombinacje RI + Na + Al oraz Mg + Ba + Fe.

```
wyniki2 <-  sapply(liczba.sasiadow.zakres, function(k)
  errorest(Type ~ RI + Na + Al, Glass.scaled, model=my.ipredknn,
           predict=my.predict, estimator="632plus",
           est.para=control.errorest(nboot = 50), ile.sasiadow=k)$error)
wyniki3 <-  sapply(liczba.sasiadow.zakres, function(k)
  errorest(Type ~ Mg + Ba + Fe, Glass.scaled, model=my.ipredknn,
           predict=my.predict, estimator="632plus",
           est.para=control.errorest(nboot = 50), ile.sasiadow=k)$error)

blad.zakres <- range(c(wyniki1, wyniki2, wyniki3))

plot(liczba.sasiadow.zakres, wyniki1, type="b", col="green", lwd=3,
     main="Porównanie modeli", xlab="k (liczba sąsiadów)",
     ylab="błąd klasyfikacji (.632plus)", ylim=blad.zakres)
lines(liczba.sasiadow.zakres, wyniki2, type="b", col="red", lwd=3)
lines(liczba.sasiadow.zakres, wyniki3, type="b", col="blue", lwd=3)
```

```
legend("topright",legend=c("Model 1: Type ~.","Model 2: Type ~ RI + Ca + Al",
                           "Model 3: Type ~ Mg + Ba + Fe"),
       col=c("green","red", "blue"), lwd=3, bg="azure2", cex=.5)
grid()
```



Rysunek 13: Wykres wpływu doboru zmiennych na błąd klasyfikacji

Na wykresie 13 można łatwo zauważyć, że kombinacja zmiennych RI + Na + Al daje wynik całkiem dobry (minimum 0.3091938), zbliżony do wszystkich zmiennych, za to kombinacja Mg + Ba + Fe, która zdawała się mieć potencjał, jednak nie jest tak skuteczna, a jej błąd sięga niemal połowy przypadków (minimalny błąd wynosi 0.4423705. I tak najlepszy wynik z tych wszystkich opcji daje uwzględnienie wszystkich zmiennych.

3.6 Drzewa klasyfikacyjne

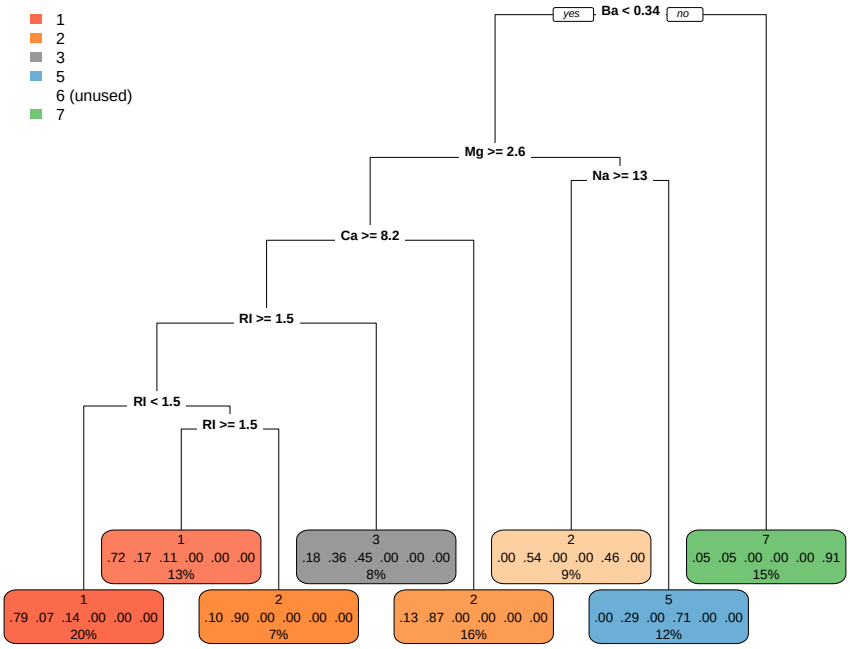
Następną rozpatrywaną metodą są drzewa klasyfikacyjne – drzewa binarne, dla których w każdym węźle dokonywany jest podział względem zmiennej w tym momencie najlepiej separującej dane.

3.6.1 Utworzenie modelu i wyznaczenie prognozowanych etykietek

Zacznę, tak jak wcześniej, od utworzenia modelu dla wyznaczonych zbiorów uczącego i testowego, a także wstępnej oceny jego dokładności. Na początku testować będę najprostszy, najbardziej podstawowy model.

```
# Utworzenie modelu
glass.tree <- rpart(Type ~ ., data=glass.learn)
```

Wyświetlę także utworzone drzewo.



Rysunek 14: Wizualizacja najprostszego drzewa klasyfikacyjnego

Na wykresie 14 można zauważyć, że drzewo to nie przypisuje wartości do klasy 6, więc z pewnością nie jest w pełni dokładne.

Wyznaczę jeszcze prognozowane etykietki klas.

```
# Etykietki dla zbioru uczącego
glass.tree.etykietki.prog.learn <- predict(glass.tree, newdata=glass.learn,
                                           type="class")

# Etykietki dla zbioru testowego
glass.tree.etykietki.prog.train <- predict(glass.tree, newdata=glass.train,
                                           type="class")
```

3.6.2 Macierz pomyłek

Jak we wszystkich poprzednich przypadkach, dokładność klasyfikacji ocenię poprzez macierze pomyłek i zmierzenie błędów klasyfikacji.

```
## Zbiór uczący
# Macierz pomyłek
conf.matrix.glass.tree.learn <- table(glass.tree.etykietki.prog.learn,
```

```
                                    etykietki.rzecz.learn)

# Dokładność
glass.tree.learn.accuracy <-
  sum(diag(conf.matrix.glass.tree.learn))/glass.obs.learn

# Błąd klasyfikacji
glass.tree.learn.error <- 1 - glass.tree.learn.accuracy

## Zbiór testowy
# Macierz pomyłek
conf.matrix.glass.tree.train <- table(glass.tree.etykietki.prog.train,
                                      etykietki.rzecz.train)

# Dokładność
glass.tree.train.accuracy <-
  sum(diag(conf.matrix.glass.tree.train))/glass.obs.train

# Błąd klasyfikacji
glass.tree.train.error <- 1 - glass.tree.train.accuracy
```

Dla zbioru uczącego macierz pomyłek wygląda następująco:

```
conf.matrix.glass.tree.learn
```

```
##                                etykietki.rzecz.learn
## glass.tree.etykietki.prog.learn  1  2  3  5  6  7
##                               1 35  5  6  0  0  0
##                               2  4 36  0  0  6  0
##                               3  2  4  5  0  0  0
##                               5  0  5  0 12  0  0
##                               6  0  0  0  0  0  0
##                               7  1  1  0  0  0 20
```

Dokładność tej klasyfikacji wynosi 0.7605634, zatem błąd jest równy 0.2394366.

Analogicznie, dla zbioru testowego macierz pomyłek wygląda następująco:

```
conf.matrix.glass.tree.train
```

```
##                                etykietki.rzecz.train
## glass.tree.etykietki.prog.train  1  2  3  5  6  7
##                               1 21  3  2  0  0  1
##                               2  7 17  1  0  3  1
##                               3  0  5  3  0  0  0
##                               5  0  0  0  0  0  1
##                               6  0  0  0  0  0  0
```

```
##                                 7   0   0   0   1   0   6
```

Dokładność tej klasyfikacji wynosi 0.6527778, zatem błąd jest równy 0.3472222.

Błąd dla zbioru treningowego jest wyższy niż dla zbioru uczącego, jednak oba są dość wysokie – ponad 1/5 wszystkich obserwacji.

3.6.3 Zaawansowane schematy oceny dokładności

Po utworzeniu pierwotnego modelu przetestuję rozmaite schematy oceny dokładności, aby sprawdzić, czy dla bazowego modelu da się poprawić wyniki.

```
# Utworzenie własnej funkcji
my.rpart <- function(formula1, data1, ile.split, zlozonosc, glebokosc)
  rpart(formula=formula1,data=data1,minsplit=ile.split,cp=zlozonosc,
        maxdepth=glebokosc)

## Porównanie błędów klasyfikacji dla ustalonych z góry parametrów domyślnych
## (metody: 10-fold cross-validation, bootstrap, .632plus)
#CV
glass.tree.cv <- errorest(Type ~., Glass, model=my.rpart, predict=my.predict, estimator=
         est.para=control.errorest(k = 10), ile.split=20,
         zlozonosc=0.01,glebokosc=30)

#Boot
glass.tree.boot <- errorest(Type ~., Glass, model=my.rpart, predict=my.predict, estimat
         est.para=control.errorest(nboot = 50), ile.split=20,
         zlozonosc=0.01,glebokosc=30)

#.632plus
glass.tree.632plus <- errorest(Type ~., Glass, model=my.rpart, predict=my.predict,
         estimator="632plus", est.para=control.errorest(nboot = 50),
         ile.split=20, zlozonosc=0.01,glebokosc=30)

# Wyświetlenie wyników
glass.tree.cv
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass, model = my.rpart,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.split = 20, zlozonosc = 0.01, glebokosc = 30)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.3131
```

```
glass.tree.boot
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass, model = my.rpart,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##     ile.split = 20, zlozonosc = 0.01, glebokosc = 30)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.3536
## Standard deviation: 0.0081
```

```
glass.tree.632plus
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass, model = my.rpart,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot =
##     ile.split = 20, zlozonosc = 0.01, glebokosc = 30)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.3029
```

Błędy są do siebie dość zbliżone, choć najlepszy wynik daje, jak wcześniej, .632plus. Tak więc tej metody walidacji będę używać w dalszej części badań.

3.6.4 Dobór różnych parametrów i zmiennych

W następnej części przetestuję, jak dobór parametrów i zmiennych wpływa na wyniki. Testowanymi parametrami będą:

- minsplit (od 1 do 20),
- cp (od 0,01 do 0,2, co 0,01),
- maxdepth (od 1 do 30).

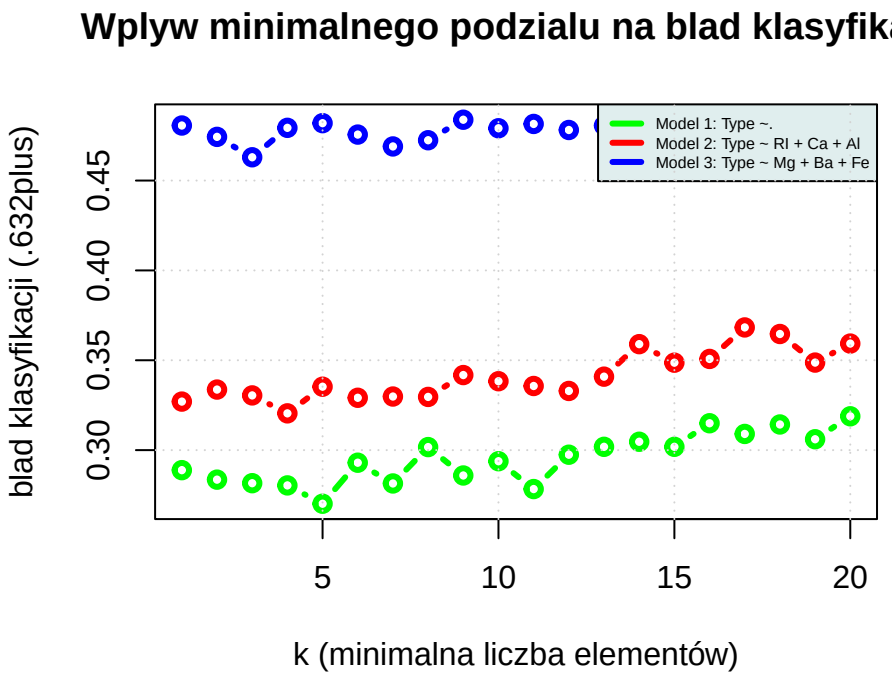
Dla każdego parametru dokonam też sprawdzenia dla poszczególnych zmiennych. Zmiennymi, które będę porównywać, będą:

- wszystkie zmienne,
- RI + Na + Al,
- Mg + Fe + Ba.

```
minsplit.param <- 1:20
cp.param <- seq(0.01, 0.2, by=0.01)
```

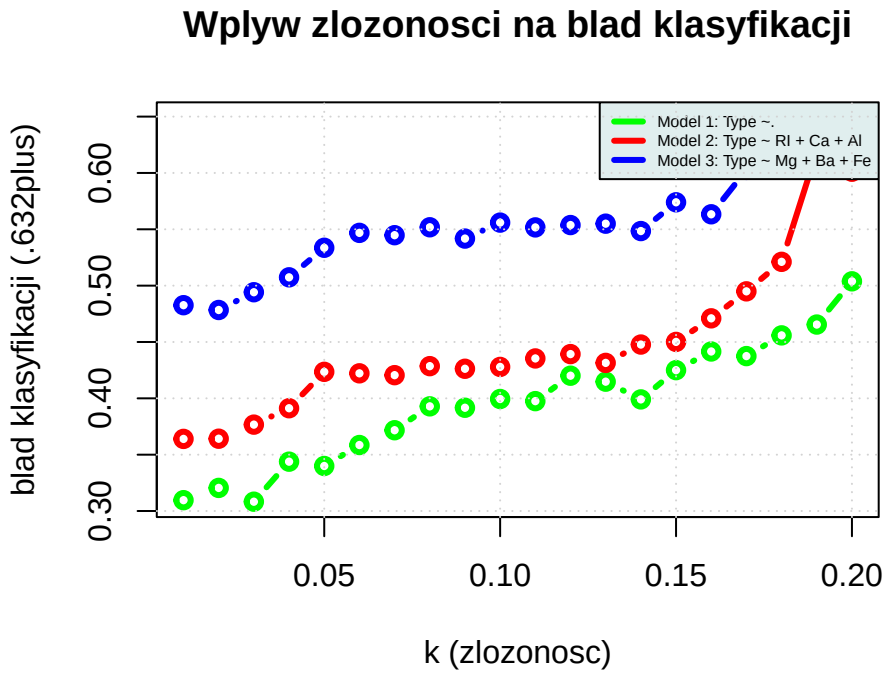
```r
maxdepth.param <- 1:30

# minsplit
wyniki.minsplit1 <- sapply(minsplit.param, function(k)
  errorest(Type ~., Glass, model=my.rpart, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.split=k, zlozonosc=0.01,glebokosc=30)$error)
wyniki.minsplit2 <- sapply(minsplit.param, function(k)
  errorest(Type ~ RI + Na + Al, Glass, model=my.rpart, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.split=k, zlozonosc=0.01,glebokosc=30)$error)
wyniki.minsplit3 <- sapply(minsplit.param, function(k)
  errorest(Type ~ Mg + Ba + Fe, Glass, model=my.rpart, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.split=k, zlozonosc=0.01,glebokosc=30)$error)

# Cp
wyniki.cp1 <- sapply(cp.param, function(k)
  errorest(Type ~., Glass, model=my.rpart, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.split=20, zlozonosc=k,glebokosc=30)$error)
wyniki.cp2 <- sapply(cp.param, function(k)
  errorest(Type ~ RI + Na + Al, Glass, model=my.rpart, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.split=20, zlozonosc=k,glebokosc=30)$error)
wyniki.cp3 <- sapply(cp.param, function(k)
  errorest(Type ~ Mg + Ba + Fe, Glass, model=my.rpart, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.split=20, zlozonosc=k,glebokosc=30)$error)

# Maxdepth
wyniki.maxdepth1 <- sapply(maxdepth.param, function(k)
  errorest(Type ~., Glass, model=my.rpart, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.split=20, zlozonosc=0.01,glebokosc=k)$error)
wyniki.maxdepth2 <- sapply(maxdepth.param, function(k)
  errorest(Type ~ RI + Na + Al, Glass, model=my.rpart, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.split=20, zlozonosc=0.01,glebokosc=k)$error)
wyniki.maxdepth3 <- sapply(maxdepth.param, function(k)
  errorest(Type ~ Mg + Ba + Fe, Glass, model=my.rpart, predict=my.predict,
           estimator="632plus", est.para=control.errorest(nboot = 50),
           ile.split=20, zlozonosc=0.01,glebokosc=k)$error)
```

Sprawdzę wyniki zmiany parametrów na wykresach.



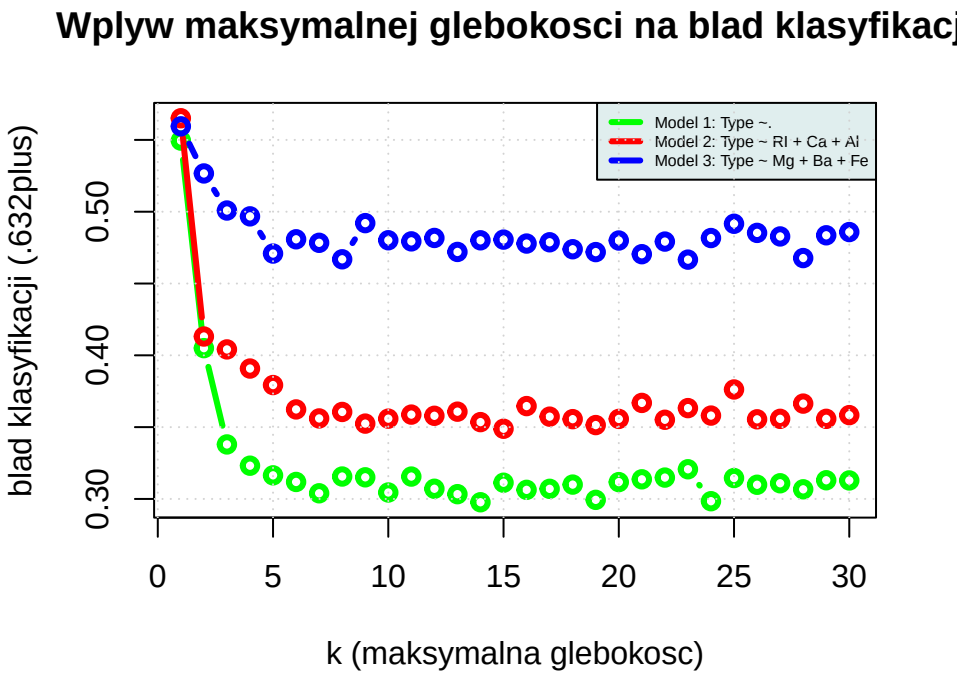
Rysunek 15: Wpływ minimalnej wielkości, dla której dokonywany jest podział, na błąd

Na wykresie 15 można zauważyć, że najlepsze wyniki otrzymujemy dla 5 elementów przy uwzględnieniu wszystkich zmiennych: 0.270182.



Rysunek 16: Wpływ złożoności na błąd

Na wykresie 16 można zauważyć, że najlepsze wyniki otrzymujemy dla złożoności 0,03 przy uwzględnieniu wszystkich zmiennych – błąd wynosi 0.308269.



Rysunek 17: Wpływ maksymalnej głębokości na błąd

Na wykresie 17 można zauważyć (choć nie tak łatwo), że najlepsze wyniki są osiągnięte dla maksymalnej głębokości 14 z uwzględnieniem wszystkich zmiennych. Błąd wynosi wtedy 0.2977011.

Przy wszystkich parametrach najlepsze wyniki daje uwzględnienie wszystkich zmiennych. Nieco gorzej radzi sobie model RI + Na + Al, a najgorzej – Mg + Fe + Ba.

3.7 Naiwny klasyfikator bayesowski

Ostatnim z testowanych klasyfikatorów będzie naiwny klasyfikator Bayesowski. Opiera się on na wyznaczeniu prawdopodobieństwa warunkowego – że dany element należy do wybranej klasy, pod warunkiem, że wektor zmiennych objaśniających ma konkretną wartość. W wariancie naiwnym zakładamy, że wszystkie zmienne objaśniające są niezależne, co upraszcza model.

W analizach wykorzystam dwie funkcje – naiveBayes z pakietu e1071, która zakłada, że zmienne ciągłe mają rozkład normalny, oraz NaiveBayes z pakietu klaR, któa umożliwia jądrową estymację gęstości zmiennych ciągłych.

3.7.1 Utworzenie modeli i wyznaczenie prognozowanych etykietek

Na początek utworzę oba modele w wersji podstawowej i wyznaczę prognozowane etykietki.

```r
## e1071
# model
glass.NB1 <- naiveBayes(Type~., data=glass.learn)
# prognozowane etykietki dla zbioru uczącego
glass.NB1.etykietki.prog.learn <- predict(glass.NB1, glass.learn)
# prognozowane etykietki dla zbioru testowego
glass.NB1.etykietki.prog.train <- predict(glass.NB1, glass.train)

## klaR
# model
glass.NB2 <- NaiveBayes(Type~., data=glass.learn, usekernel=TRUE)
# prognozowane etykietki dla zbioru uczącego
glass.NB2.etykietki.prog.learn <- predict(glass.NB2, glass.learn)$class
# prognozowane etykietki dla zbioru testowego
glass.NB2.etykietki.prog.train <- predict(glass.NB2, glass.train)$class
```

3.7.2 Macierze pomyłek

Po utworzeniu modeli porównam ich dokładność dla zbiorów uczącego i testowego.

```r
### e1071
## Zbiór uczący
# Macierz pomyłek
conf.matrix.glass.NB1.learn <-
  table(glass.NB1.etykietki.prog.learn, etykietki.rzecz.learn)

# Dokładność
glass.NB1.learn.accuracy <-
  sum(diag(conf.matrix.glass.NB1.learn))/glass.obs.learn

# Błąd klasyfikacji
glass.NB1.learn.error <- 1 - glass.NB1.learn.accuracy

## Zbiór testowy
# Macierz pomyłek
conf.matrix.glass.NB1.train <-
  table(glass.NB1.etykietki.prog.train, etykietki.rzecz.train)

# Dokładność
glass.NB1.train.accuracy <-
  sum(diag(conf.matrix.glass.NB1.train))/glass.obs.train

# Błąd klasyfikacji
glass.NB1.train.error <- 1 - glass.NB1.train.accuracy
```

```
### klaR
## Zbiór uczący
# Macierz pomyłek
conf.matrix.glass.NB2.learn <-
  table(glass.NB2.etykietki.prog.learn, etykietki.rzecz.learn)

# Dokładność
glass.NB2.learn.accuracy <-
  sum(diag(conf.matrix.glass.NB2.learn))/glass.obs.learn

# Błąd klasyfikacji
glass.NB2.learn.error <- 1 - glass.NB2.learn.accuracy

## Zbiór testowy
# Macierz pomyłek
conf.matrix.glass.NB2.train <-
  table(glass.NB2.etykietki.prog.train, etykietki.rzecz.train)

# Dokładność
glass.NB2.train.accuracy <-
  sum(diag(conf.matrix.glass.NB2.train))/glass.obs.train

# Błąd klasyfikacji
glass.NB2.train.error <- 1 - glass.NB2.train.accuracy
```

Po wyznaczeniu odpowiednich wielkości przyjrzę się im.

Dla modelu zakładającego rozkład gaussowski, dla zbioru uczącego macierz pomyłek prezentuje się następująco:

```
conf.matrix.glass.NB1.learn
```

```
##                               etykietki.rzecz.learn
## glass.NB1.etykietki.prog.learn  1  2  3  5  6  7
##                               1 31 25  4  0  0  0
##                               2  3 11  0  2  0  0
##                               3  7  8  6  0  0  0
##                               5  0  2  0  1  0  0
##                               6  1  5  1  9  6  1
##                               7  0  0  0  0  0 19
```

Dokładność tej klasyfikacji wynosi 0.5211268, zatem błąd wynosi 0.4788732.

Dla tego samego modelu, dla zbioru testowego macierz pomyłek prezentuje się następująco:

```
conf.matrix.glass.NB1.train
```

```
##                                etykietki.rzecz.train
## glass.NB1.etykietki.prog.train  1  2  3  5  6  7
##                               1 20 20  1  0  0  1
##                               2  2  0  1  0  0  0
##                               3  4  4  3  0  0  0
##                               5  0  0  0  0  0  0
##                               6  2  1  1  0  3  2
##                               7  0  0  0  1  0  6
```

Dokładność tej klasyfikacji wynosi 0.4444444, zatem błąd wynosi 0.5555556.

Dla modelu zakładającego jądrową estymację gęstości, dla zbioru uczącego macierz pomyłek prezentuje się następująco:

```
conf.matrix.glass.NB2.learn
```

```
##                                etykietki.rzecz.learn
## glass.NB2.etykietki.prog.learn  1  2  3  5  6  7
##                               1 42 24  3  0  0  0
##                               2  0 19  0  0  3  0
##                               3  0  6  8  0  0  0
##                               5  0  2  0 12  0  0
##                               6  0  0  0  0  3  0
##                               7  0  0  0  0  0 20
```

Dokładność tej klasyfikacji wynosi 0.7323944, zatem błąd wynosi 0.2676056.

Dla tego samego modelu, dla zbioru testowego macierz pomyłek prezentuje się następująco:

```
conf.matrix.glass.NB2.train
```

```
##                                etykietki.rzecz.train
## glass.NB2.etykietki.prog.train  1  2  3  5  6  7
##                               1 24 16  4  0  0  1
##                               2  0  2  0  0  3  0
##                               3  4  7  2  0  0  1
##                               5  0  0  0  0  0  1
##                               6  0  0  0  0  0  0
##                               7  0  0  0  1  0  6
```

Dokładność tej klasyfikacji wynosi 0.4722222, zatem błąd wynosi 0.5277778.

Dla zbioru uczącego widać znaczną różnicę w modelach – dla jądrowej estymacji gęstości błąd jest niemal dwukrotnie mniejszy niż dla założenia rozkładu gaussowskiego. Natomiast w przypadku zbioru testowego błąd dla drugiego modelu też jest nieco mniejszy, jednak różnica nie jest tak istotna i w obu przypadkach jest większy niż 50%.

3.7.3 Zaawansowane schematy oceny dokładności

W kolejnej części, jak wcześniej, sprawdzę zaawansowane schematy oceny dokładności dla obu modeli.

```
# Utworzenie własnej funkcji
my.nb.e1071 <- function(formula1, data1) naiveBayes(formula=formula1,data=data1)

## Porównanie błędów klasyfikacji
## (metody: 10-fold cross-validation, bootstrap, .632plus)
#CV
glass.NB1.cv <- errorest(Type ~., Glass, model=my.nb.e1071, predict=my.predict,
         estimator="cv", est.para=control.errorest(k = 10))

#Boot
glass.NB1.boot <- errorest(Type ~., Glass, model=my.nb.e1071, predict=my.predict,
         estimator="boot", est.para=control.errorest(nboot = 50))

#.632plus
glass.NB1.632plus <- errorest(Type ~., Glass, model=my.nb.e1071, predict=my.predict,
         estimator="632plus", est.para=control.errorest(nboot = 50))

# Wyświetlenie wyników
glass.NB1.cv
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass, model = my.nb.e1071,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.6449
```

```
glass.NB1.boot
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass, model = my.nb.e1071,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.6096
## Standard deviation: 0.0128
```

```
glass.NB1.632plus
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass, model = my.nb.e1071,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot =
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.5682
```

Dla modelu zakładającego rozkład gaussowski błędy są bardzo wysokie, ponad 50%. Ponadto znacznie różnią się między sobą – pomiędzy najmniejszym błędem, z metody .632 plus, a największym, z metody cross-validation, występuje różnica prawie 0,08.

Dla drugiego modelu została pominięte metoda cross-validation, ze względu na fakt, że losuje w grupach często wyłącznie 1 obserwację. Przez to niemożliwa jest jądrowa estymacja gęstości i zwrócony zostaje błąd.

```
# Utworzenie własnej funkcji
my.predict2  <- function(model, newdata) predict(model, newdata=newdata,
                                                 type="class")$class
my.nb.klaR <- function(formula1, data1)
  NaiveBayes(formula=formula1,data=data1, usekernel=TRUE)

# Porównanie błędów klasyfikacji (metody: bootstrap, .632plus)
#Boot
glass.NB2.boot <- errorest(Type ~., Glass, model=my.nb.klaR, predict=my.predict2,
         estimator="boot", est.para=control.errorest(nboot = 50))

#.632plus
glass.NB2.632plus <- errorest(Type ~., Glass, model=my.nb.klaR, predict=my.predict2,
         estimator="632plus", est.para=control.errorest(nboot = 50))

# Wyświetlenie wyników
glass.NB2.boot
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass, model = my.nb.klaR,
##     predict = my.predict2, estimator = "boot", est.para = control.errorest(nboot = 50
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
```

```
## Misclassification error:  0.4176
## Standard deviation: 0.0087
```

```
glass.NB2.632plus
```

```
##
## Call:
## errorest.data.frame(formula = Type ~ ., data = Glass, model = my.nb.klaR,
##     predict = my.predict2, estimator = "632plus", est.para = control.errorest(nboot =
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.3518
```

Dla modelu zakładającego jądrową estymację gęstości metoda .632plus jest nieco lepsza, ale i tak błąd jest wysoki, wynoszący przynajmniej 35%.

3.7.4 Różne kombinacje zmiennych

Na koniec, dla obu modeli i metody oceny .632plus dokonam oceny dla poszczególnych kombinacji zmiennych. Jak wcześniej, uwzględnię wszystkie zmienne, zmienne RI + Na + Al oraz zmienne Mg + Fe + Ba.

```
## e1071
# Wszystkie zmienne
wyniki.e1071.1 <- errorest(Type ~., Glass, model=my.nb.e1071,
                           predict=my.predict, estimator="632plus",
                           est.para=control.errorest(nboot = 50))
# RI + Na + Al
wyniki.e1071.2 <- errorest(Type ~ RI + Na + Al, Glass, model=my.nb.e1071,
                           predict=my.predict, estimator="632plus",
                           est.para=control.errorest(nboot = 50))
# Mg + Fe + Ba
wyniki.e1071.3 <- errorest(Type ~ Mg + Fe + Ba, Glass, model=my.nb.e1071,
                           predict=my.predict, estimator="632plus",
                           est.para=control.errorest(nboot = 50))

## klaR
# Wszystkie zmienne
wyniki.klaR.1 <- errorest(Type ~., Glass, model=my.nb.klaR,
                          predict=my.predict2, estimator="632plus",
                          est.para=control.errorest(nboot = 50))
# RI + Na + Al
wyniki.klaR.2 <- errorest(Type ~ RI + Na + Al, Glass, model=my.nb.klaR,
                          predict=my.predict2, estimator="632plus",
                          est.para=control.errorest(nboot = 50))
```

```r
# Mg + Fe + Ba
wyniki.klaR.3 <- errorest(Type ~ Mg + Fe + Ba, Glass, model=my.nb.klaR,
                          predict=my.predict2, estimator="632plus",
                          est.para=control.errorest(nboot = 50))
```

Wyniki prezentują się następująco:

Model	Wszystkie zmienne	RI + Na + Al	Mg + Fe + Ba
e1071	0.587133	0.4282563	0.7234631
klaR	0.3745607	0.365217	0.4874417

Tabela 3: Błędy klasyfikacji dla różnych zmiennych w dwóch modelach naiwnego klasyfikatora Bayesowskiego

Na podstawie tabeli 3 zauważyć można, że, w przeciwieństwie do poprzednich metod, użycie kombinacji zmiennych RI + Na + Al zmniejsza błąd klasyfikacji (lepsze wyniki otrzymamy dla klaR). Natomiast kombinacja Mg + Fe + Ba zwiększa ten błąd (w przypadku e1071 nawet do 70%).

3.8 Wnioski

W niniejszym sprawozdaniu porównałam trzy metody klasyfikacji dla danych Glass. Ze względu na dużą liczbę klas we wszystkich błędy klasyfikacji są dość wysokie – żaden nie jest niższy niż 0,2.

3.8.1 Kombinacje zmiennych

Analizowałam trzy kombinacje zmiennych: wszystkie zmienne, zmienne RI + Na + Al oraz zmienne Mg + Fe + Ba. Dla zdecydowanej większości metod najlepszy wynik dawało uwzględnienie wszystkich zmiennych, w drugiej kolejności kombinacja RI + Na + Al, a w trzeciej kombinacja Mg + Fe + Ba. Zapewne wynika to z faktu, że połączenie wszystkich zmiennych daje największe zróżnicowanie poszczególnych klas.

Jedynym wyjątkiem okazał się naiwny klasyfikator Bayesowski – w jego przypadku najlepszy wynik daje uwzględnienie kombinacji zmiennych RI + Na + Al, a uwzględnienie wszystkich zmiennych jest na drugim miejscu. Może to wynikać z faktu, że w tym klasyfikatorze czynione jest założenie dotyczące niezależności zmiennych, niekoniecznie prawdziwe. Im więcej zmiennych branych jest pod uwagę, tym błąd wynikający z tego założenia będzie większy, zatem dzięki ograniczeniu liczby zmiennych objaśniających model w rzeczywistości się poprawia.

3.8.2 Parametry

Dobór różnych parametrów również miał znaczenie.

- Dla metody knn im mniej sąsiadów, tym lepsze wyniki – najlepszy wynik został osiągnięty dla uwzględnienia tylko 1 sąsiada.
- Dla drzew klasyfikacyjnych najniższy błąd udało się osiągnąć dla parametrów: minsplit=5, cp=0,01, maxdepth=30. Może to wynikać z faktu, że klasa 6 jest bardzo mało liczna i dla wyższych wartości minsplit przydział elementów do tej klasy w ogóle nie następuje (patrz rysunek 14 dla modelu ogólnego, który nie przypisuje żadnych elementów do klasy 6). Mimo wszystko jednak zmiany parametru minsplit aż tak nie zmieniają wyników. Dla parametru cp generalnie im wyższa złożoność, tym wyższy błąd. Dla parametru maxdepth dla bardzo niskich wartości jest bardzo wysoki błąd (co wynika z tego, że nie ma wystarczająco dużo podziałów), ale dla większych możliwych głębokości błąd ten utrzymuje się na w miarę podobnym poziomie.
- Dla naiwnego klasyfikatora Bayesowskiego zastosowanie jądrowej estymacji gęstości znacząco poprawia wyniki. Pokazuje to, że założenie, iż zmienne objaśniające mają rozkład gaussowski, w tym przypadku mija się z prawdą.

3.8.3 Metody klasyfikacji

Przedstawię najniższe błędy, jakie udało się uzyskać poszczególnymi metodami, w odpowiednich momentach eksperymentu.

Metoda knn:

- dla zbioru uczącego: 0.2323944,
- dla zbioru testowego: 0.3333333,
- dla metod oceny dokładności: 0.3162458,
- dla różnej liczby sąsiadów i różnych zbiorów zmiennych (razem): 0.2326867.

Metoda tree:

- dla zbioru uczącego: 0.2394366,
- dla zbioru testowego: 0.3472222,
- dla metod oceny dokładności: 0.3029322,
- dla parametru minsplit (wszystkie kombinacje zmiennych): 0.270182,
- dla parametru cp (wszystkie kombinacje zmiennych): 0.308269,
- dla parametru maxdepth (wszystkie kombinacje zmiennych): 0.2977011.

Metoda naive Bayes (oba modele):

- dla zbioru uczącego: 0.2676056,
- dla zbioru testowego: 0.5277778,
- dla metod oceny dokładności: 0.3518245,
- dla poszczególnych kombinacji zmiennych: 0.365217.

Zatem:

- dla zbioru uczącego najlepsze wyniki daje knn,
- dla zbioru testowego najlepsze wyniki daje również knn,
- dla różnych metod walidacji najlepsze wyniki ostatecznie zwraca tree,

- dla różnych parametrów i podzbiorów zmiennych objaśniających najlepsze wyniki zwraca knn.

Oznacza to, że po odpowiednim doborze parametrów ostatecznie najlepiej sprawdziła się metoda k-najbliższych sąsiadów. Choć wyniki drzew klasyfikacyjnych były nieco gorsze (przez algorytm, który może pomijać mało liczne klasy), to ostatecznie okazały się dosyć zbliżone. Najgorzej poradził sobie naiwny klasyfikator Bayesowski, nawet po zastosowaniu jądrowej estymacji gęstości rozkładów zmiennych objaśniających.

3.8.4 Metody oceny dokładności

Właściwie dla wszystkich metod klasyfikacji ocena dokładności przy użyciu metody .632plus zwraca najmniejszy błąd, zatem wybór metody ma wpływ na wyniki.