Raport 3

Dominik Ochej
268838

14 czerwca 2023

Spis treści

1 Wstęp **1**

2 Klasyfikacja na bazie modelu regresji liniowej **1**

3 Model regresji dla składników wielomianowych stopnia 2. **4**

4 Zadanie2 **6**
4.1 Algorytm knn 9
4.2 Algorytm drzewa klasyfikacyjnego 10
4.3 Naiwny klasyfikator bayesowski. 13

5 Testowanie metod dla różnych parametrów oraz podzbioru cech **15**

1 Wstęp

Raport będzie dotyczył poznania metod klasyfikacji takich jak:

- Regresja liniowa
- Algorytm k najbliższych sąsiadów
- Drzewa klasyfikacyjne
- Naiwny klasyfikator bayesowski.

Następnie metody te będą porównane na przykladowych danych.

2 Klasyfikacja na bazie modelu regresji liniowej

```
#W pierwszym zdaniu pracujemy na danych dotyczących irysów.
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
#Zmienne objaśniające
names(which(sapply(iris, is.numeric) == TRUE))
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
```

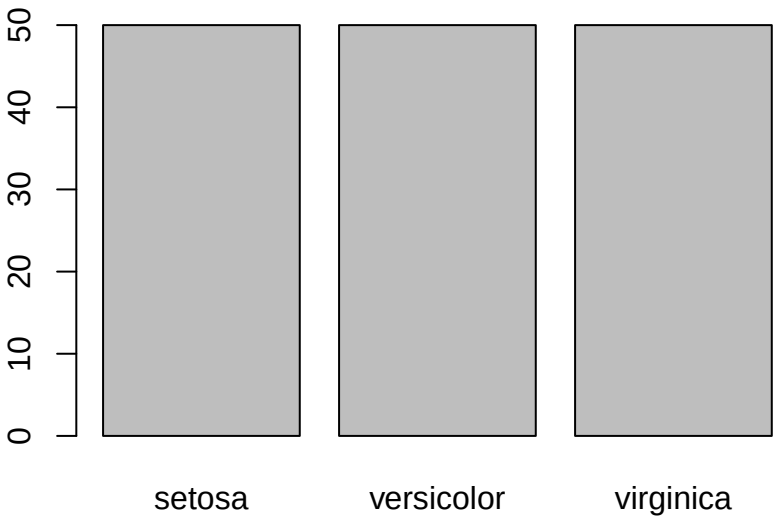
```
p<-length(names(which(sapply(iris, is.numeric) == TRUE)))
etykietki.klas <- iris$Species

#liczba obiektów-150
(n<-length(etykietki.klas))
```

```
## [1] 150
```

```
#Liczba klas-3
(K<-length(levels(etykietki.klas)))
```

```
## [1] 3
```

```
#Obserwacje są po równo podzielone między trzy klasy
plot(etykietki.klas)
```



```r
set.seed(1)
#Dzielimy dane na zbiór uczący oraz testowy
learning.set.index <- sample(1:n,2/3*n)
learning.set <- iris[learning.set.index,]
test.set <- iris[-learning.set.index,]

X_learning<- cbind(rep(1,nrow(learning.set)),learning.set[,1:4])

X_test<- cbind(rep(1,nrow(test.set)),test.set[,1:4])

X_learning <- as.matrix(X_learning)
X_test <- as.matrix(X_test)


Y_learning<- matrix(0,nrow=nrow(learning.set),ncol=K)
Y_test<-matrix(0,nrow = nrow(test.set),ncol=K)


etykietki.learning<-as.numeric(learning.set$Species)
for(k in 1:K){
  Y_learning[etykietki.learning==k,k]<-1
}
#Tworzymy macierz estymowanych współczynników
B<-solve(t(X_learning)%*%X_learning) %*% t(X_learning) %*% Y_learning

#
Y.hat_learning <- X_learning%*%B
Y.hat_test <- X_test%*%B


Y.hat<-rbind(Y.hat_learning,Y.hat_test)

klasy <- levels(iris$Species)

maks.ind <-apply(Y.hat,1,FUN=function(x) which.max(x))
prognozowane.etykietki<-klasy[maks.ind][1:100]
rzeczywiste.etykietki <-learning.set$Species

macierz.pomylek <- table(rzeczywiste.etykietki,prognozowane.etykietki)
#Macierz pomyłek oraz błąd klasyfikacji na zbiorze uczącym.
macierz.pomylek
```

```
##                      prognozowane.etykietki
## rzeczywiste.etykietki setosa versicolor virginica
##            setosa         34          0         0
##            versicolor      0         20        11
##            virginica       0          4        31
```

```r
1-sum(diag(macierz.pomylek))/100
```

```
## [1] 0.15

prognozowane.etykietki<-klasy[maks.ind][101:150]
rzeczywiste.etykietki<-test.set$Species
macierz.pomylek<-table(rzeczywiste.etykietki,prognozowane.etykietki)
#Macierz pomyłek oraz błąd klasyfikacji na zbiorze testowym.
macierz.pomylek

##                      prognozowane.etykietki
## rzeczywiste.etykietki setosa versicolor virginica
##            setosa         16          0         0
##            versicolor      0         12         7
##            virginica       0          2        13

1-sum(diag(macierz.pomylek))/50

## [1] 0.18
```

Obserwujemy przysłanianie klasy Versinicolor przez klase virginica. Jak widzimy w macierzy około 1/3 przypadków z klasy versicolor jest prognozowana jako virginica.

3 Model regresji dla składników wielomianowych stopnia 2.

```
X <- iris[, 1:4]

# Utworzenie składników wielomianowych
X_poly <- cbind( X^2, X[,1]*X[,2], X[,1]*X[,3], X[,1]*X[,4], X[,2]*X[,3], X[,2]*X[,4], X
Y <- matrix(0, nrow = nrow(X), ncol = ncol(X_poly))

colnames(X_poly)<-c("SL^2","SW^2","PL^2","PW^2","SL*SW","SL*PL","SL*PW","SW*PL","SW*PW",
head(X_poly)

##    SL^2  SW^2 PL^2 PW^2 SL*SW SL*PL SL*PW SW*PL SW*PW PL*PW iris$Species
## 1 26.01 12.25 1.96 0.04 17.85  7.14  1.02  4.90  0.70  0.28       setosa
## 2 24.01  9.00 1.96 0.04 14.70  6.86  0.98  4.20  0.60  0.28       setosa
## 3 22.09 10.24 1.69 0.04 15.04  6.11  0.94  4.16  0.64  0.26       setosa
## 4 21.16  9.61 2.25 0.04 14.26  6.90  0.92  4.65  0.62  0.30       setosa
## 5 25.00 12.96 1.96 0.04 18.00  7.00  1.00  5.04  0.72  0.28       setosa
## 6 29.16 15.21 2.89 0.16 21.06  9.18  2.16  6.63  1.56  0.68       setosa

set.seed(1)
n<-nrow(X_poly)
#Dzielimy na zbiór uczący oraz testowy
learning.set.index <- sample(1:n,2/3*n)
learning.set <- X_poly[learning.set.index,]
test.set <- X_poly[-learning.set.index,]
```

```r
X_learning<- cbind(rep(1,nrow(learning.set[1:4])),learning.set[1:4])

X_test<- cbind(rep(1,nrow(test.set[1:4])),test.set[1:4])

X_learning <- as.matrix(X_learning)
X_test <- as.matrix(X_test)


Y_learning<- matrix(0,nrow=nrow(learning.set),ncol=K)
Y_test<-matrix(0,nrow = nrow(test.set),ncol=K)


etykietki.learning<-as.numeric(learning.set$`iris$Species`)
for(k in 1:K){
  Y_learning[etykietki.learning==k,k]<-1
}

B<-solve(t(X_learning)%*%X_learning) %*% t(X_learning) %*% Y_learning

Y.hat_learning <- X_learning%*%B

Y.hat_test <- X_test%*%B

Y.hat<-rbind(Y.hat_learning,Y.hat_test)

klasy <- levels(iris$Species)

maks.ind <-apply(Y.hat,1,FUN=function(x) which.max(x))
prognozowane.etykietki<-klasy[maks.ind][1:100]
rzeczywiste.etykietki <-learning.set$`iris$Species`
#Macierz pomyłek oraz błąd klasyfikacji na zbiorze uczącym.
macierz.pomylek <- table(rzeczywiste.etykietki,prognozowane.etykietki)
macierz.pomylek
```

```
##                      prognozowane.etykietki
## rzeczywiste.etykietki setosa versicolor virginica
##            setosa         33          1         0
##            versicolor      0         26         5
##            virginica       0          4        31
```

```r
1-sum(diag(macierz.pomylek))/100
```

```
## [1] 0.1
```

```r
#Macierz pomyłek oraz błąd klasyfikacji na zbiorze testowym.
prognozowane.etykietki<-klasy[maks.ind][101:150]
rzeczywiste.etykietki<-test.set$`iris$Species`
macierz.pomylek<-table(rzeczywiste.etykietki,prognozowane.etykietki)
macierz.pomylek
```

```
##                       prognozowane.etykietki
## rzeczywiste.etykietki setosa versicolor virginica
##             setosa         16          0         0
##             versicolor      0         14         5
##             virginica       0          0        15
```

```
1-sum(diag(macierz.pomylek))/50
```

```
## [1] 0.1
```

Model liniowy dla rozszerzonej przestrzeni cech jest dokładniejszy od podstawowego ponieważ błąd klasyfikacji na zbiorze testowym jest mniejszy o ∼10%.

4 Zadanie2

```
data(Glass)

#Liczba przypadków- 214
n<-length(Glass$Type)
#Liczba cech- 9
length(names(Glass[1:9]))
```

```
## [1] 9
```

```
class(Glass$Type)
```

```
## [1] "factor"
```

```
#Liczba klas- 6. Klasa numer 4 nie jest zawarta w tej bazie danych.
length(levels(Glass$Type))
```

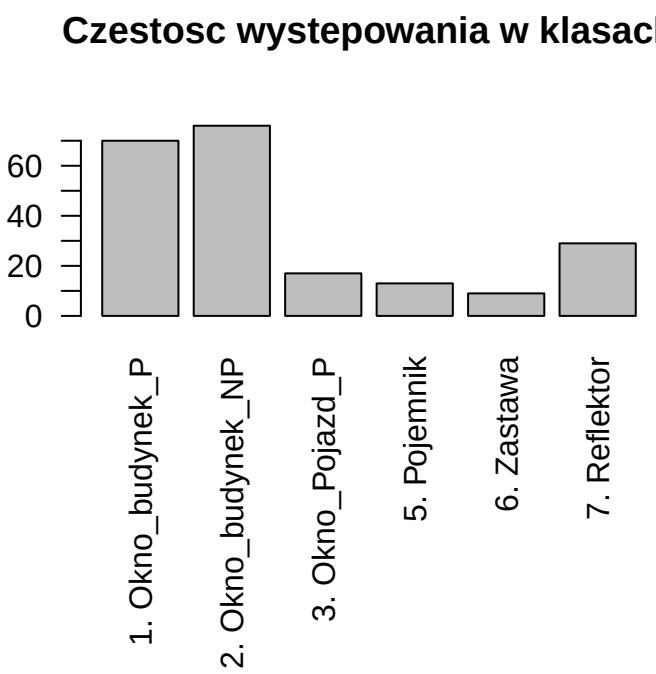
```
## [1] 6
```

```
#1- oznacza przetworzone szkło z okna budnku
#2- oznacza nieprzetworzone szkło z okna budnku
#3- oznacza przetworzone szkło z okna pojazdu
#4- oznacza nieprzetworzone szkło z okna pojazdu (Brak w tej bazie)
#5- oznacza szklany pojemnik
#6- oznacza szklaną zastawę stołową
#7- oznacza reflektor
levels(Glass$Type)
```

```
## [1] "1" "2" "3" "5" "6" "7"
```

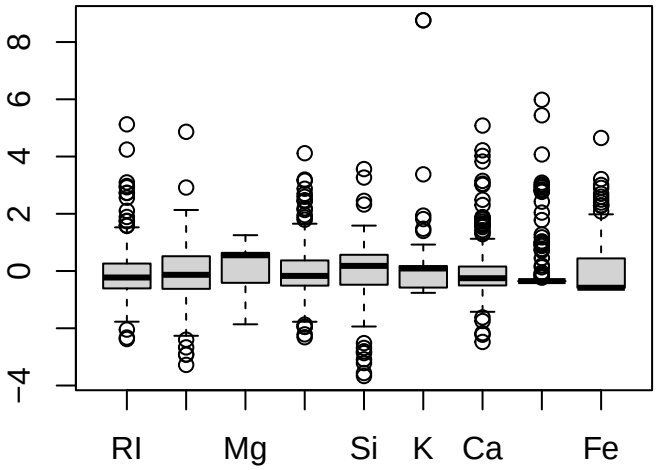
```
str(Glass)
```

```
## 'data.frame': 214 obs. of  10 variables:
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 72.7 73 72.6 73.1 ...
##  $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
##  $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ Type: Factor w/ 6 levels "1","2","3","5",..: 1 1 1 1 1 1 1 1 1 1 ...

#Wszystkie zmienne są typu num, z wyjątkiem Type, która jest typu factor.
Glass.czestosc<-Glass$Type
levels(Glass.czestosc)<-c("1. Okno_budynek_P","2. Okno_budynek_NP","3. Okno_Pojazd_P","5
par(mai = c(2, 1, 1, 1))
plot(Glass.czestosc,las=2,main="Częstość występowania w klasach")
```

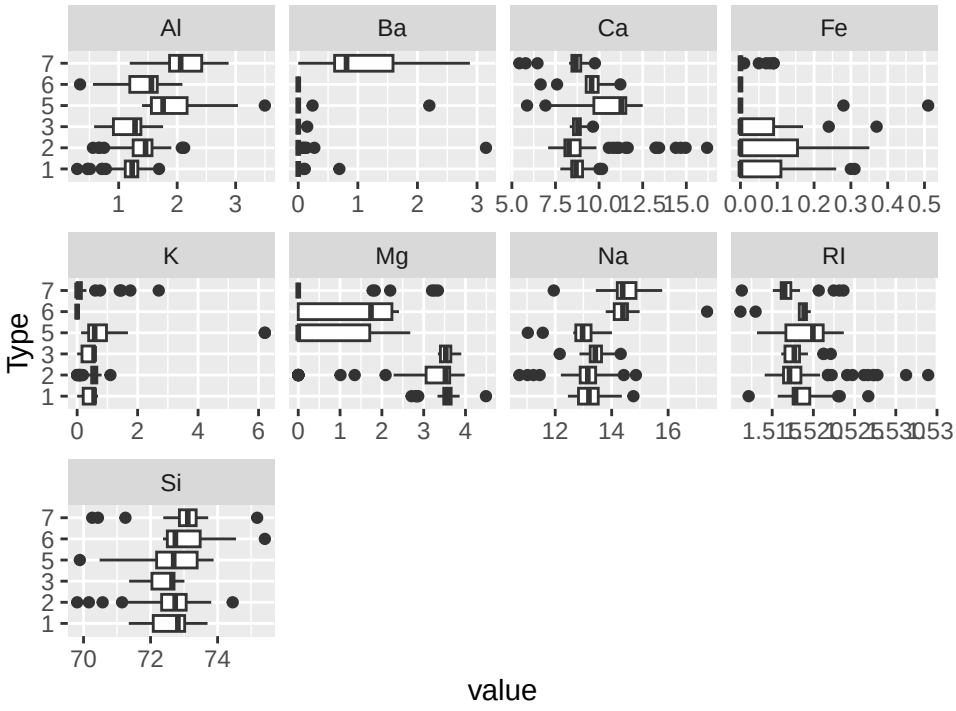



```
par(mai = c(1, 1, 1, 1))
#Najczęściej występującymi klasami są: Klasa 2- 76 przypadków
#oraz klasa 1 - 70 przypadków
#Obie te klasy dotyczą szkła z okna budynków.
#Wstępnie można uważać, że klasy te będzie ciężej rozróżnić.

#Przypisując obserwacje do  najczęstszej grupy błąd klasyfikacyjny wynosi ~64.5%
1-length(Glass[which(Glass$Type==2),][,1])/214

## [1] 0.6448598
```

```r
dane<-Glass
dane.numeryczne<-scale(dane[,1:9])
#Obserwujemy istotne różnice w wariancji poszczególnych cech.
#Musimy więc standaryzować dane.

dane<-cbind(as.data.frame(dane.numeryczne),Type=dane[,10])
boxplot(dane.numeryczne)
```




```r
plot_boxplot(Glass,by="Type")
```




```
#Zmienne charakteryzujące które najgorzej separują obiekty to Rl oraz Si.
```

4.1 Algorytm knn

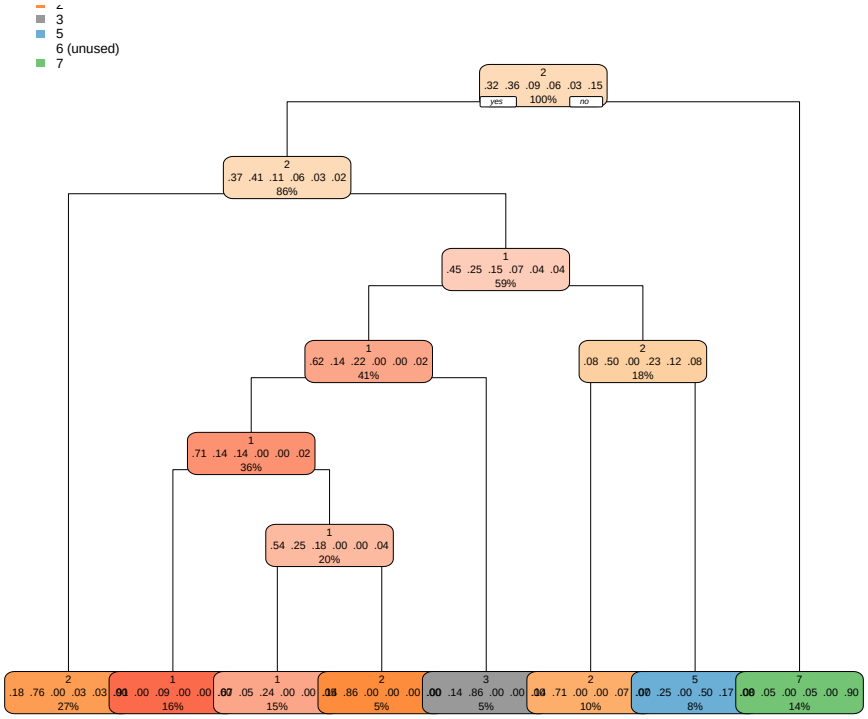
```
# Tworzymy zbiór uczący i testowy
# dla k powtorzeń
k=1000
średnia=0
for (i in 1:k){
# losujemy obiekty do zbioru uczącego i testowego

learning.set.index <- sample(1:n,2/3*n)
n.learning <- nrow(learning.set)
n.test <- nrow(test.set)

learning.set <- dane[learning.set.index,]
test.set     <- dane[-learning.set.index,]
#Algorytm knn
etykietki.rzecz <- test.set$Type
etykietki.prog<-knn(learning.set[,-10],test.set[,-10],learning.set$Type,k=5)
#Macierz pomyłek
(wynik.tablica <- table(etykietki.prog,etykietki.rzecz))

#Średni błąd klasyfikacyjny wynosi ~30%
n.test <- dim(test.set)[1]
```

```
a<-(n.test - sum(diag(wynik.tablica))) / n.test
średnia=średnia +a

}
#Pojedynczy błąd klasyfikacyjny
(n.test - sum(diag(wynik.tablica)))/n.test

## [1] 0.375

#Macierz pomyłek
print(wynik.tablica)

##                etykietki.rzecz
## etykietki.prog  1  2  3  5  6  7
##              1 18 11  2  1  1  0
##              2  2 16  2  3  1  0
##              3  0  0  0  0  0  0
##              5  0  2  0  0  1  0
##              6  0  0  0  0  2  0
##              7  0  0  0  0  1  9

#Średni błąd klasyfikacyjny
średnia/k->sr_knn
sr_knn

## [1] 0.3521667
```

4.2 Algorytm drzewa klasyfikacyjnego

```
# Konstrukcja drzewa klasyfikacyjnego
# Określamy formułę  modelu
model <- Type ~ . # wszystkie cechy objaśniające
k=100 # ile razy tworzymy zbiory uczące i testowe
sr.test=0
sr.learining=0

for (i in 1:k){
# losujemy obiekty do zbioru uczącego i testowego
learning.set.index <- sample(1:n,2/3*n)
n.learning <- nrow(learning.set)
n.test <- nrow(test.set)
learning.set <- dane[learning.set.index,]
test.set     <- dane[-learning.set.index,]

# budujemy drzewo (parametry domyślne)
Glass.tree.simple <- rpart(model, data=learning.set)
```

```
Glass.tree <- rpart(model, data=learning.set, control=rpart.control(cp=.03, minsplit=10,

# prognozy dla zbioru uczącego
pred.labels.learning <- predict(Glass.tree.simple, newdata=learning.set, type = "class")

# prognozy dla zbioru testowego
pred.labels.test <- predict(Glass.tree.simple, newdata=test.set, type = "class")

# wyznaczenie prognozowanych prawdopodobieństw a posteriori
pred.probs.test <- predict(Glass.tree.simple, newdata=test.set, type = "prob")
conf.mat.learning <- table(pred.labels.learning, learning.set$Type)
conf.mat.test <- table(pred.labels.test, test.set$Type)



(error.rate.learning <- (n.learning - sum(diag(conf.mat.learning))) / n.learning)
(error.rate.test <- (n.test - sum(diag(conf.mat.test))) / n.test)

sr.learining = sr.learining + error.rate.learning
sr.test = sr.test + error.rate.test
}

# Wizualizacja

rpart.plot(Glass.tree.simple,xpd=TRUE, cex=.35)
```



```r
# Ocena dokładności klasyfikacji

# macierz pomyłek (confusion matrix)
#Zbiór testowy

conf.mat.test
```

```
##
## pred.labels.test  1  2  3  5  6  7
##                1 14  2  2  0  0  0
##                2  8 17  1  1  2  0
##                3  2  1  1  0  0  0
##                5  0  5  0  4  3  0
##                6  0  0  0  0  0  0
##                7  1  0  0  0  0  8
```

```r
#Zbiór uczący

conf.mat.learning
```

```
##
## pred.labels.learning  1  2  3  5  6  7
##                    1 35  1  7  0  0  1
##                    2 10 45  0  1  2  1
##                    3  0  1  6  0  0  0
##                    5  0  3  0  6  2  1
##                    6  0  0  0  0  0  0
##                    7  0  1  0  1  0 18
```

```r
# błąd klasyfikacji (na zbiorze uczącym i testowy) dla pojedynczego podziału na zbiór
(error.rate.learning <- (n.learning - sum(diag(conf.mat.learning))) / n.learning)
```

```
## [1] 0.2253521
```

```r
(error.rate.test <- (n.test - sum(diag(conf.mat.test))) / n.test)
```

```
## [1] 0.3888889
```

```r
#Błąd klasyfikacji dla k podziałów na zbiory uczące/testowe

sr.learining/k
```

```
## [1] 0.2357746
```

```r
sr.test/k->sr_drzewo
sr_drzewo
```

```
## [1] 0.3418056
```

4.3 Naiwny klasyfikator bayesowski.

```
k=100 # ile razy tworzymy zbiory uczące i testowe
sr.test=0
sr.learining=0

for (i in 1:k){

# losujemy obiekty do zbioru uczącego i testowego
learning.set.index <- sample(1:n,2/3*n)
n.learning <- nrow(learning.set)
n.test <- nrow(test.set)
learning.set <- dane[learning.set.index,]
test.set     <- dane[-learning.set.index,]
model.nb <- naiveBayes(Type~., data = learning.set)

# prognozowane etykietki zbiór uczący
pred.labels.nb.learn <- predict(model.nb, newdata = learning.set)

# macierz pomyłek zbiór uczący
conf.mat.nb.learning <- table(learning.set$Type, pred.labels.nb.learn)

# błąd klasyfikacji na zbior uczący
(blad.nb <- (n.learning-sum(diag(conf.mat.nb.learning)))/n.learning)
sr.learining= sr.learining+blad.nb

# prognozowane etykietki zbiór testowy
pred.labels.nb.test <- predict(model.nb, newdata = test.set)

# macierz pomyłek zbiór testowy
conf.mat.nb.test <- table(test.set$Type, pred.labels.nb.test)

# błąd klasyfikacji na zbior testowy
(blad.nb <- (n.test-sum(diag(conf.mat.nb.test)))/n.test)
sr.test = sr.test + blad.nb
}

# macierz pomyłek na zbiorze uczacym
conf.mat.nb.learning
```

```
##    pred.labels.nb.learn
##      1  2  3  5  6  7
##   1  3  2 37  1  1  0
##   2  1  5 36  5  4  0
##   3  0  0 13  0  0  0
##   5  0  0  0  7  0  0
##   6  0  0  0  0  6  0
##   7  0  0  1  4 12  4
```

```r
# błąd klasyfikacji na zbiorze uczącym
(blad.nb <- (n.learning-sum(diag(conf.mat.nb.learning)))/n.learning)

## [1] 0.7323944

# średni błąd klasyfikacji na zbiorze uczącym dla k podziałów
sr.learining/k

## [1] 0.5844366

# macierz pomyłek na zbiorze testowym
conf.mat.nb <- table(test.set$Type, pred.labels.nb.test)
conf.mat.nb

##    pred.labels.nb.test
##      1  2  3  5  6  7
##   1  3  0 20  0  3  0
##   2  1  3 17  2  2  0
##   3  1  0  2  0  1  0
##   5  0  1  0  2  3  0
##   6  0  0  0  0  3  0
##   7  0  0  0  2  2  4

# błąd klasyfikacji na zbiorze testowym
(blad.nb <- (n.test-sum(diag(conf.mat.nb)))/n.test)

## [1] 0.7638889

# średni błąd klasyfikacji na zbiorze testowym dla k podziałów
sr.test/k->sr_bayes
sr_bayes

## [1] 0.6541667
```

```r
metody <- c("sr_knn", "sr_drzewo", "sr_bayes")
values <- c(sr_knn, sr_drzewo,sr_bayes)
data <- data.frame(metody, values)
xtable_data <- xtable(data, caption = "Średnie wartości błędu algorytmów.")
print(xtable_data)
```

	metody	values
1	sr_knn	0.35
2	sr_drzewo	0.34
3	sr_bayes	0.65

Tabela 1: Średnie wartości błędu algorytmów.

Jak widzimy w tym przypadku najgorzej radzi sobie naiwna metoda bayesa z błędem na

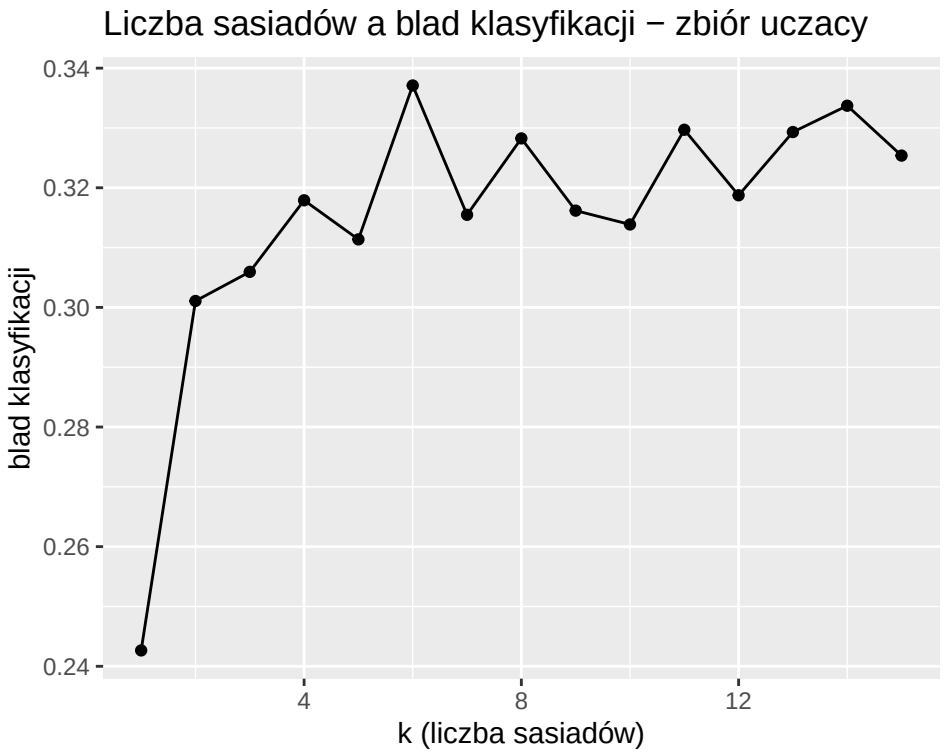
poziomie przypisania wszystkich obiektów do jednej klasy. Najlepiej radzą sobie metody drzewa klasyfikacyjnego oraz knn.

5 Testowanie metod dla różnych parametrów oraz podzbioru cech

```
set.seed(123)
learning.set.index <- sample(1:214,2/3*214)
n.learning <- nrow(learning.set)
n.test <- nrow(test.set)
#Z poprzednich wniosków wiemy, że RI oraz Si najgorzej seprują dlatego nie bierzmy ich

learning.set <- dane[learning.set.index,c(2:4,6:10)]
test.set     <- dane[-learning.set.index,c(2:4,6:10)]

#Przetestujemy metodę k-NN dla róznych parametrów tj. ilość sąsiadów oraz porównamy bł


my.predict  <- function(model, newdata) predict(model, newdata=newdata, type="class")
my.ipredknn <- function(formula1, data1, ile.sasiadow) ipredknn(formula=formula1,data=da

# porownanie błędów klasyfikacji: cv, boot, .632plus
errorest(Type ~., learning.set, model=my.ipredknn, predict=my.predict, estimator="cv",

##
## Call:
## errorest.data.frame(formula = Type ~ ., data = learning.set,
##     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:  0.2958

errorest(Type ~., learning.set, model=my.ipredknn, predict=my.predict, estimator="boot",

##
## Call:
## errorest.data.frame(formula = Type ~ ., data = learning.set,
##     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:  0.3406
## Standard deviation: 0.008
```

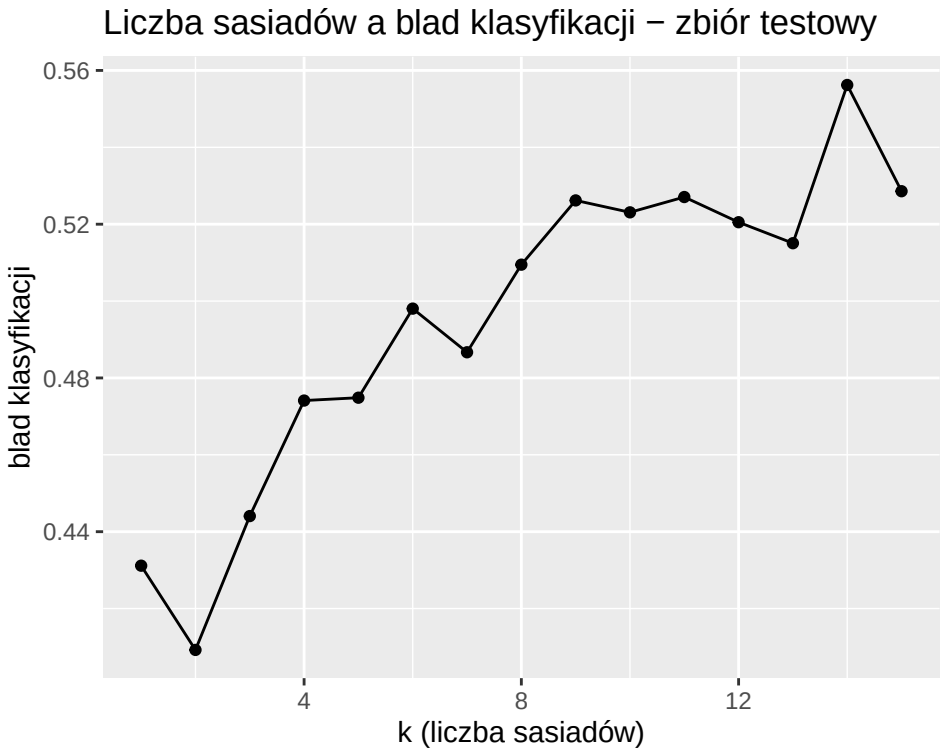
```
errorest(Type ~., learning.set, model=my.ipredknn, predict=my.predict, estimator="632plu

##
## Call:
## errorest.data.frame(formula = Type ~ ., data = learning.set,
##     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:  0.3067

errorest(Type ~., test.set, model=my.ipredknn, predict=my.predict, estimator="cv",     e

##
## Call:
## errorest.data.frame(formula = Type ~ ., data = test.set, 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:  0.4722

errorest(Type ~., test.set, model=my.ipredknn, predict=my.predict, estimator="boot",   e

##
## Call:
## errorest.data.frame(formula = Type ~ ., data = test.set, 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:  0.4857
## Standard deviation: 0.0117

errorest(Type ~., test.set, model=my.ipredknn, predict=my.predict, estimator="632plus",e

##
## Call:
## errorest.data.frame(formula = Type ~ ., data = test.set, model = my.ipredknn,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot =
##     ile.sasiadow = 5)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.4586
```

```r
# Liczba sąsiadów a błąd klasyfikacji
liczba.sasiadow.zakres <- 1:15
library(ggplot2)

# Wykres dla zbioru uczącego
wyniki_uczace <- sapply(liczba.sasiadow.zakres, function(k)
  errorest(Type ~., learning.set, model=my.ipredknn, predict=my.predict, estimator="632p

df_uczace <- data.frame(k = liczba.sasiadow.zakres, błąd_klasyfikacji = wyniki_uczace)

ggplot(df_uczace, aes(x = k, y = błąd_klasyfikacji)) +
  geom_line() +
  geom_point() +
  labs(title = "Liczba sąsiadów a błąd klasyfikacji - zbiór uczący",
       x = "k (liczba sąsiadów)",
       y = "błąd klasyfikacji")
```




```r
# Wykres dla zbioru testowego
wyniki_testowe <- sapply(liczba.sasiadow.zakres, function(k)
  errorest(Type ~., test.set, model=my.ipredknn, predict=my.predict, estimator="632plus"

df_testowe <- data.frame(k = liczba.sasiadow.zakres, błąd_klasyfikacji = wyniki_testowe)

ggplot(df_testowe, aes(x = k, y = błąd_klasyfikacji)) +
  geom_line() +
  geom_point() +
  labs(title = "Liczba sąsiadów a błąd klasyfikacji - zbiór testowy",
```

```
       x = "k (liczba sąsiadów)",
       y = "błąd klasyfikacji")
```



Błąd klasyfikacyjny pogorszył się po usunięciu cech. Dla parametru 632plus błąd na zbiorze testowym jest najmniejszy. Błąd rośnie również z liczbą sąsiadów. Dlatego aby otrzymać najlepsze wyniki należy wziąć po uwagę wszystkie cechy, wybrać metodę knn lub drzewa klasyfikacyjnego. Dla knn należy wybrać parametr 632plus oraz wybrać jak najmniejszą liczbę sąsiadów.

Literatura

[1] Peter Dalgaard, *Introductory Statistics with R*, Springer-Verlag New York, 2008.

[2] Ryszard Paweł Kostecki, *W miarę krótki i praktyczny kurs LaTeX-a w π^e minut*, http://www.fuw.edu.pl/~kostecki/kurs_latexa.pdf, 2008.