Raport 3

Paweł Matławski
album 249732

13 marca 2021

Spis treści

1 Klasyfikacja na bazie modelu regresji liniowej

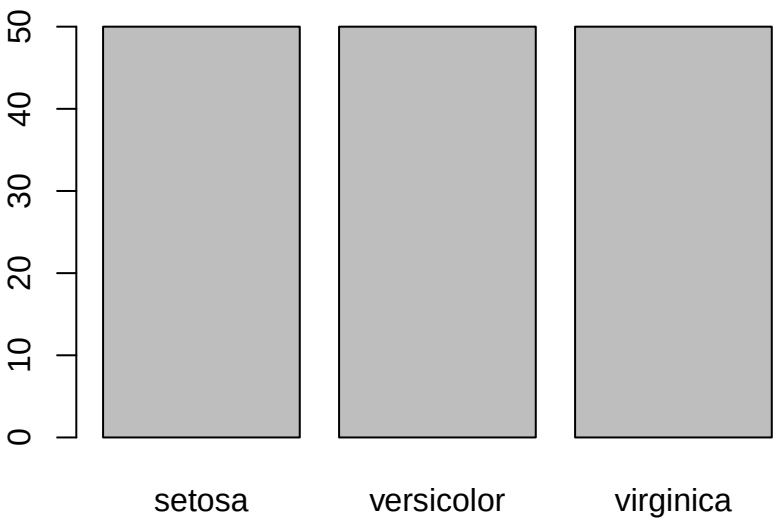
1.1 Wybór danych

Zadanie wykonamy dla ramki danych **iris** z pakietu **datasets**. Przyjrzyjmy się jej:

```
library("datasets")
data("iris")
attach(iris)

#liczba kolumn
ncol(iris)

## [1] 5

#liczba wierszy
nrow(iris)

## [1] 150
```

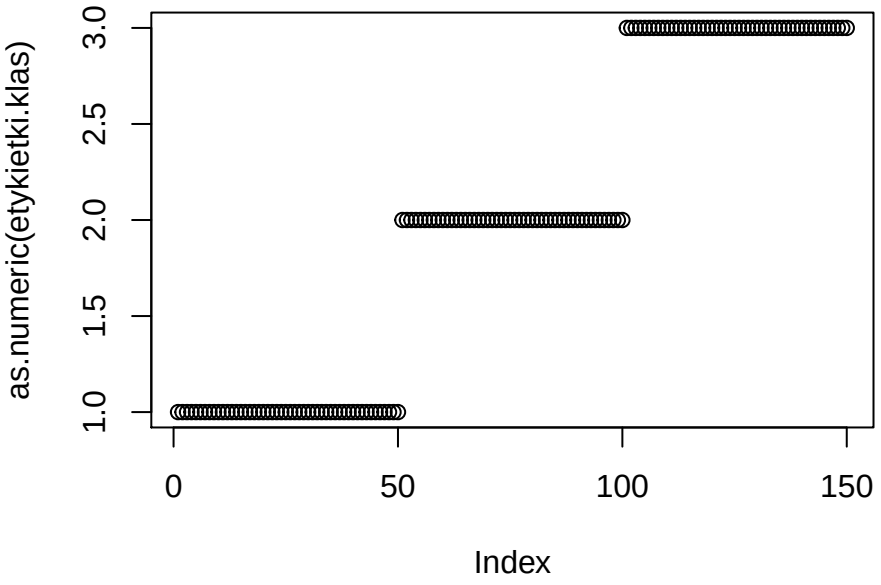
```r
#brakujące dane
sum(is.na(iris))

## [1] 0

#etykietki klas
etykietki.klas <- iris$Species

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

## [1] 150

#liczba klas
(K <- length(levels(etykietki.klas)))

## [1] 3
```

```r
plot(etykietki.klas)
```



Rysunek 1: Przynależność do poszczególnych gatunków

```r
plot(as.numeric(etykietki.klas))
```



Rysunek 2: Przynależność do poszczególnych gatunków

Wnioski:

- Obiekty są uporządkowane ze względu na przynależność do poszczególnych gatunków - po 50 z każdego.
- Mamy 3 klasy i 150 obiektów.

1.2 Podział danych na zbiór uczący i testowy

Podzielimy dane w proporcji:

- $\frac{2}{3}$-zbiór uczący
- $\frac{1}{3}$-zbiór testowy.

```
#losowanie obiektów
learning.indx <- sample(1:n,2/3*n)

#zbiór uczący
learning.set <- iris[learning.indx,]

#zbiór testowy
test.set <- iris[-learning.indx,]
```

1.3 Konstrukcja klasyfikatora i wyznaczenie prognoz

Wyznaczamy macierze eksperymentu *X.ucz* oraz *X.test*, które zawierają wartości poszczególnych zmiennych. W pierwszych kolumnach umieszczamy jedynki, by uwzględnić wyrazy wolne:

	rep(1, 100)	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
100	1.00	5.70	2.80	4.10	1.30
136	1.00	7.70	3.00	6.10	2.30
88	1.00	6.30	2.30	4.40	1.30
126	1.00	7.20	3.20	6.00	1.80
73	1.00	6.30	2.50	4.90	1.50
94	1.00	5.00	2.30	3.30	1.00

Tabela 1: Początkowy fragment macierzy X.ucz

	rep(1, 50)	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
8	1.00	5.00	3.40	1.50	0.20
12	1.00	4.80	3.40	1.60	0.20
13	1.00	4.80	3.00	1.40	0.10
18	1.00	5.10	3.50	1.40	0.30
21	1.00	5.40	3.40	1.70	0.20
22	1.00	5.10	3.70	1.50	0.40

Tabela 2: Początkowy fragment macierzy X.test

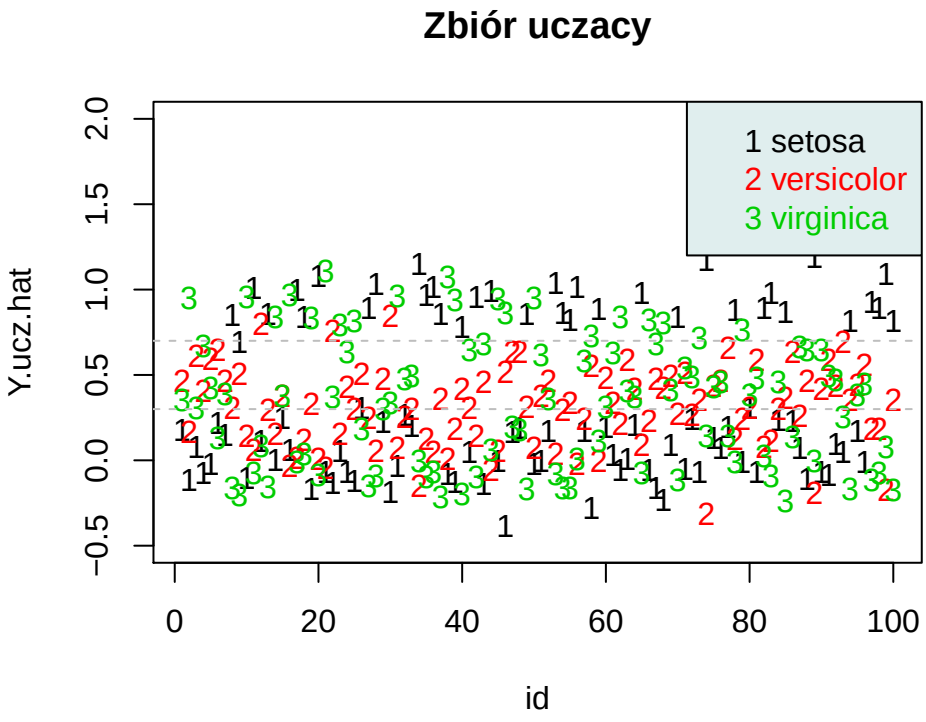
Tworzymy macierze *Y.ucz* oraz *Y.test*, które zawierają zmienne binarne kodujące poszczególne klasy.

Następnie wyznaczymy prognozowane prawdopodobieństwa przynależności do poszczególnych klas.

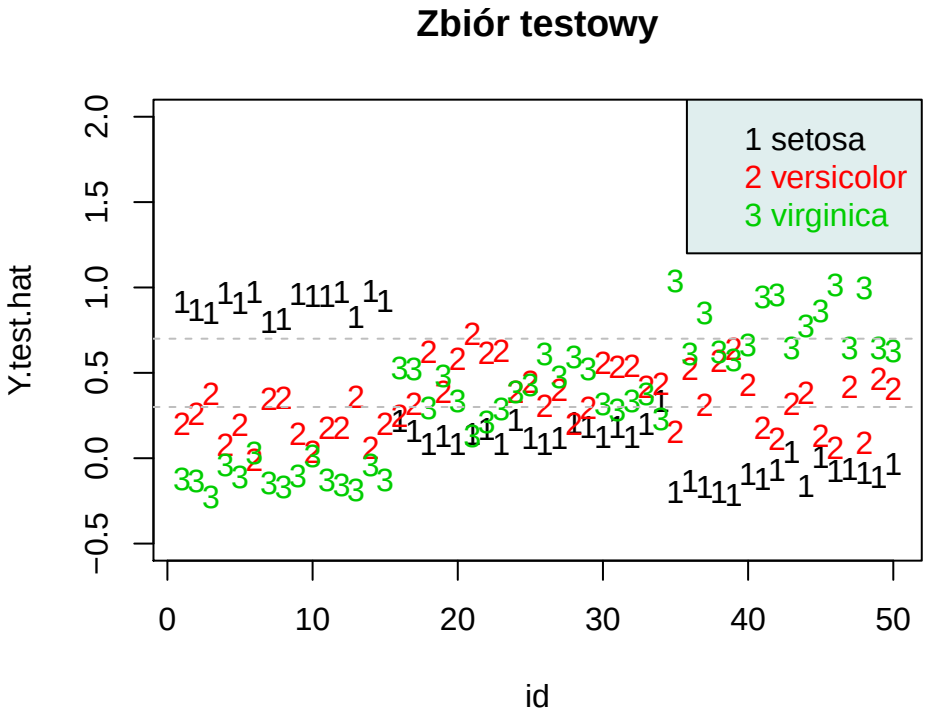
	1	2	3
100	0.18	0.47	0.35
136	-0.11	0.17	0.95
88	0.08	0.61	0.31
126	-0.07	0.41	0.67
73	-0.02	0.60	0.42
94	0.22	0.65	0.13

Tabela 3: Wartości prognozowane

Graficzna prezentacja wyników dla obu zbiorów:



Rysunek 3: Prognozy dla zbioru uczącego



Rysunek 4: Prognozy dla zbioru testowego

1.4 Ocena jakości modelu

	setosa	versicolor	virginica
setosa	35	0	0
versicolor	0	23	8
virginica	0	4	30

Tabela 4: Macierz pomyłek - zbiór uczący

```
# dokładność klasyfikacji - zbiór uczący
sum(diag(macierz.pomylek.ucz))/n.ucz

## [1] 0.88
```

	setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	11	8
virginica	0	1	15

Tabela 5: Macierz pomyłek - zbiór testowy

```
# dokładność klasyfikacji - zbiór testowy
sum(diag(macierz.pomylek.test))/n.test

## [1] 0.82
```

Wnioski:

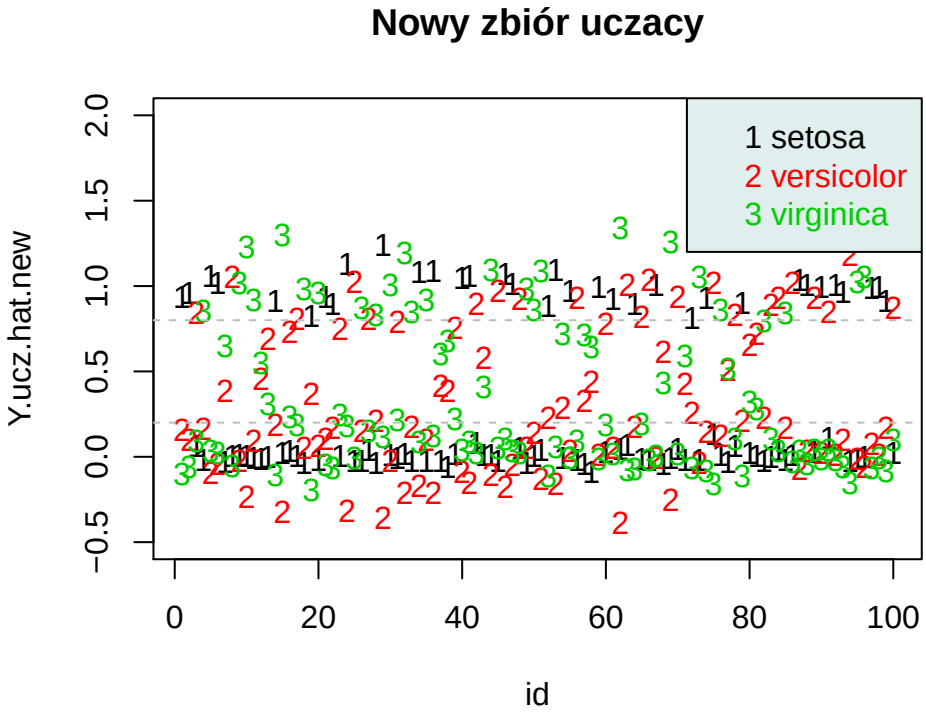
- Dokładność klasyfikacji stoi na przyzwoitym poziomie, choć z pewnością wciąż jest miejsce na poprawę.
- Gatunek setosa jest dobrze wydzielony od reszty, co widać zarówno na wykresie, jak i w macierzy pomyłek.
- W tym przypadku występuje problem maskowania klas, o którym była mowa na wykładzie.

1.5 Budowa modelu liniowego dla rozszerzonej przestrzeni cech

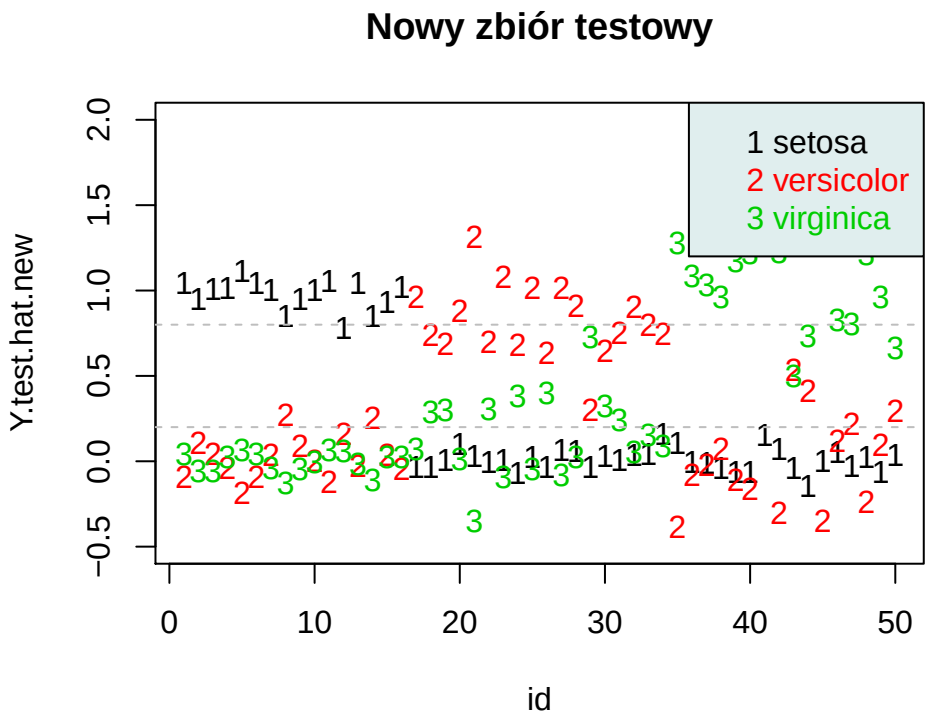
Tworzymy nowy model regresji uzupełniony o składniki wielomianowe stopnia 2 (tzn. $PL^2, PW^2, SL^2, SW^2, PL * PW, PL * SW, PL * SL, PW * SL, PW * SW, SL * SW$). Powtarzamy wszystkie kroki tak, jak w poprzednim przypadku.

	1	2	3
16	0.94	0.16	-0.10
12	0.96	0.10	-0.06
96	0.06	0.84	0.09
122	-0.02	0.16	0.85
5	1.06	-0.10	0.03
49	1.02	-0.04	0.02

Tabela 6: Wartości prognozowane - nowy zbiór



Rysunek 5: Prognozy dla nowego zbioru uczącego



Rysunek 6: Prognozy dla nowego zbioru testowego

	setosa	versicolor	virginica
setosa	34	0	0
versicolor	0	32	0
virginica	0	0	34

Tabela 7: Macierz pomyłek - nowy zbiór uczący

```
# dokładność klasyfikacji - nowy zbiór uczący
sum(diag(macierz.pomylek.ucz.new))/n.ucz.new

## [1] 1
```

	setosa	versicolor	virginica
setosa	16	0	0
versicolor	0	17	1
virginica	0	1	15

Tabela 8: Macierz pomyłek - nowy zbiór testowy

```
# dokładność klasyfikacji - nowy zbiór testowy
sum(diag(macierz.pomylek.test.new))/n.test.new

## [1] 0.96
```

Wnioski:

- Dokładność klasyfikacji w nowym, uzupełnionym modelu jest wyraźnie wyższa niż w pierwotnym.
- W tym przypadku nie występuje już problem maskowania klas.

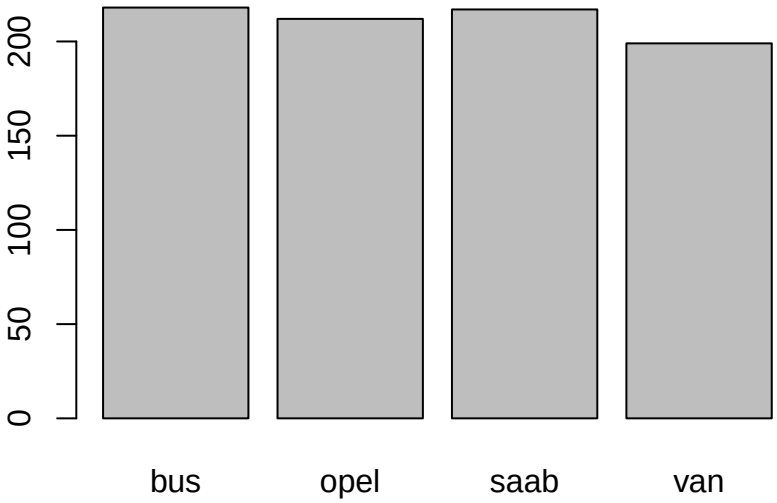
2 Porównanie metod klasyfikacji

2.1 Wybór danych

Zadanie wykonamy z użyciem ramki danych **Vehicle** z biblioteki **mlbench**. Dane zawierają informacje na temat czterech typów pojazdów: dwupiętrowego autobusu, vana marki Chevrolet, Saaba 9000 oraz Opla Manty 400. Celem stworzenia tej ramki była klasyfikacja sylwetek aut za pomocą różnych obserwacji. W teorii rozróżnienie autobusu oraz vana powinno być łatwe, w przeciwieństwie do odróżnienia od siebie dwóch różnych modeli aut.

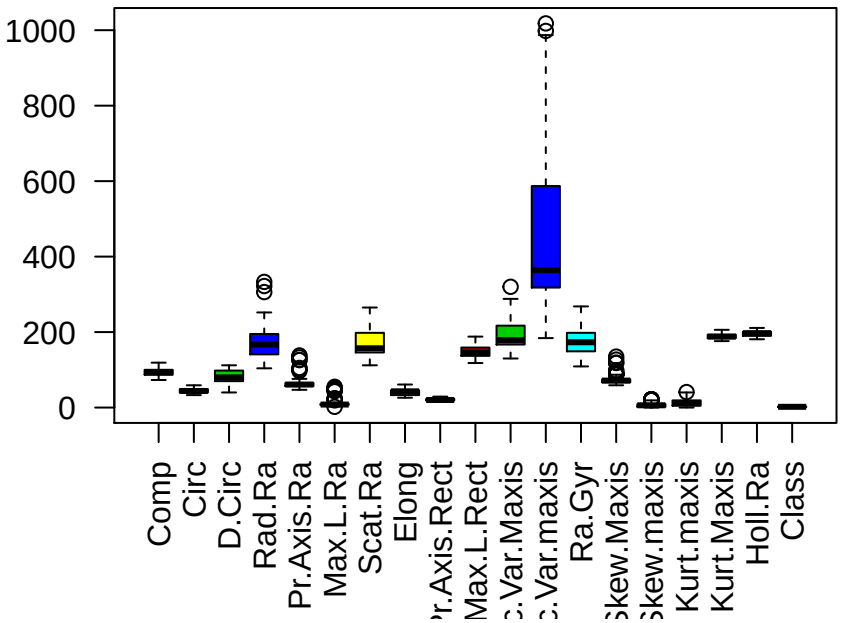
```
library("mlbench")
data("Vehicle")
attach(Vehicle)

#liczba kolumn
ncol(Vehicle)

## [1] 19

#liczba wierszy
nrow(Vehicle)

## [1] 846

#brakujące dane
sum(is.na(Vehicle))

## [1] 0
```

Jak widać w ramce nie brakuje żadnych wartości (nie są one również kodowane w niestandardowy sposób).



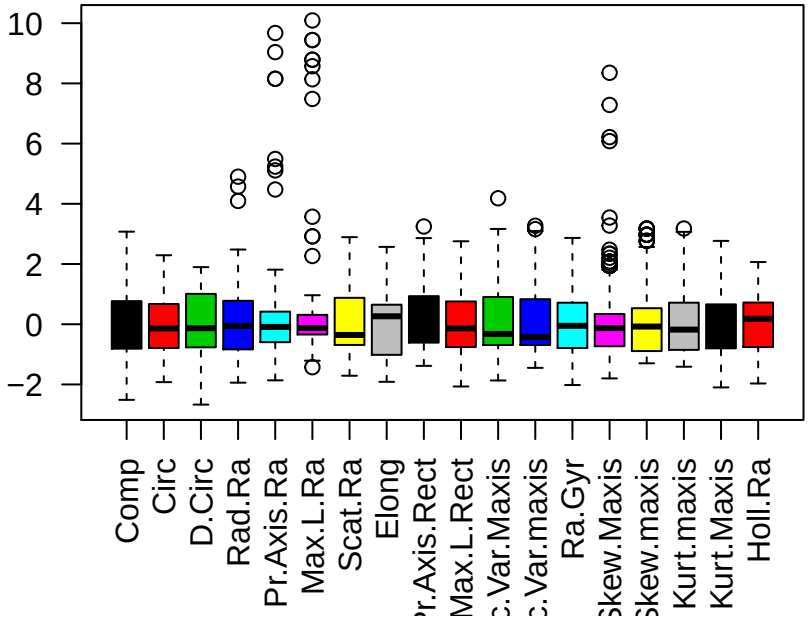
Rysunek 7: Przynależność do poszczególnych klas

Zbadajmy zmienność poszczególnych cech:



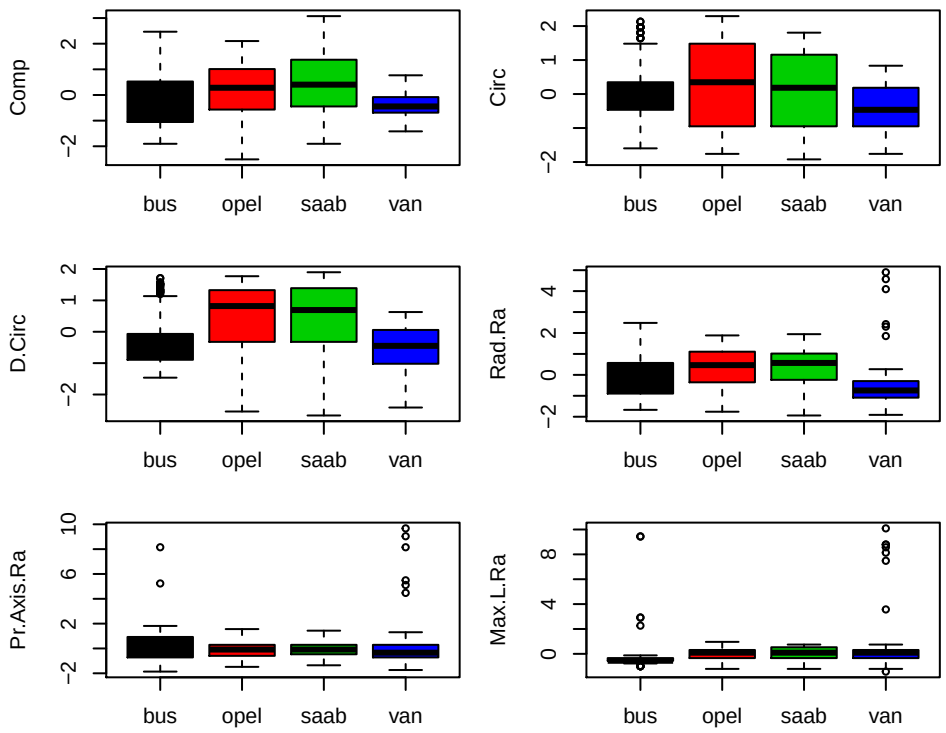
Rysunek 8: Boxploty - badanie zmienności cech

Wariancje poszczególnych cech bardzo wyraźnie się od siebie różnią. Niezbędne będzie zatem zastosowanie standaryzacji.

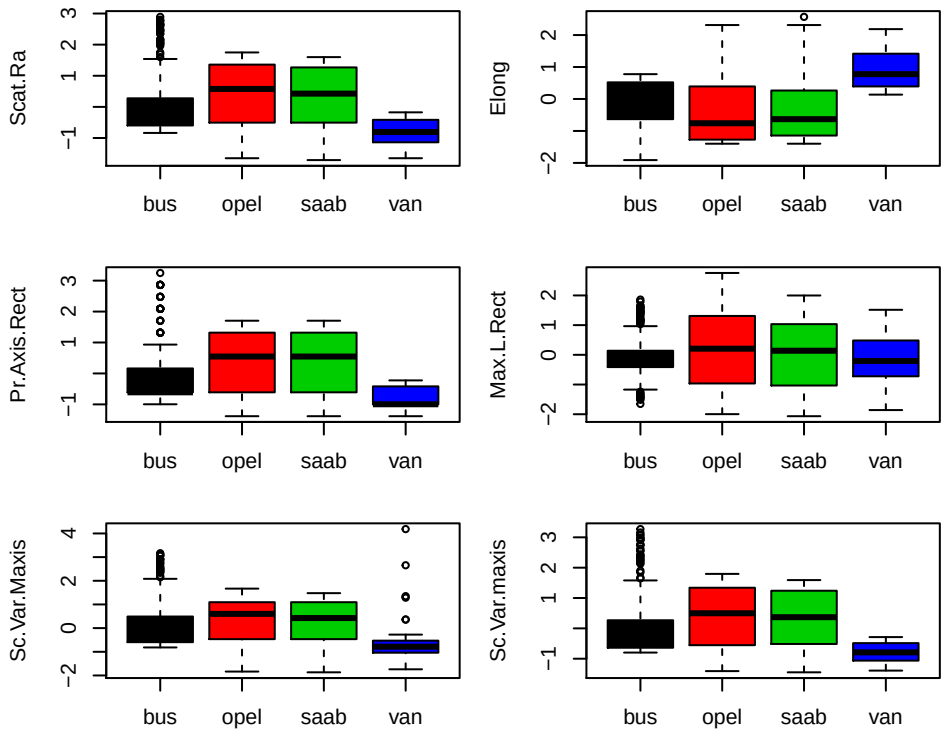


Rysunek 9: Boxploty po standaryzacji

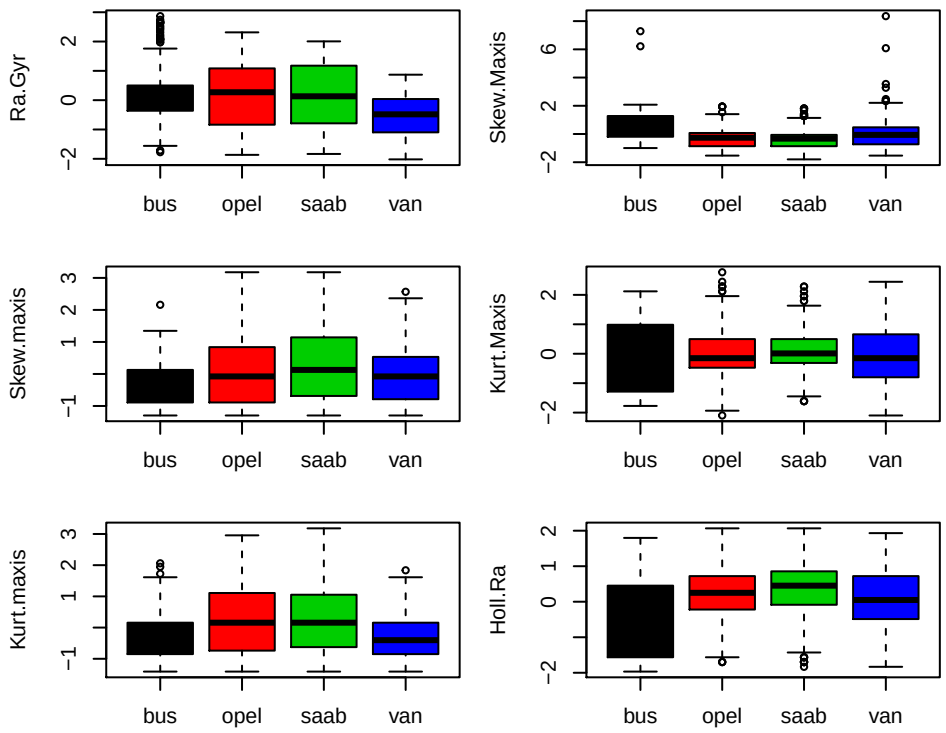
Postaramy się teraz wybrać cechy o najlepszej zdolności dyskryminacyjnej.



Rysunek 10: Zdolności dyskryminacyjne



Rysunek 11: Zdolności dyskryminacyjne



Rysunek 12: Zdolności dyskryminacyjne

Wybór zmiennej o największej zdolności dyskryminacyjnej nie jest w tym przypadku jednoznaczny. Zdecydowaliśmy, że "najbardziej obiecująca" będzie kombinacja zmiennych Elong oraz Holl.Ra (w pierwszej wyróżnia się van, a w drugiej bus). Już na tym etapie zauważamy, że różnice między Oplem a Saabem rzeczywiście nie są wyraźne.

2.2 Podział danych na zbiór uczący i testowy

Podobnie jak w zadaniu pierwszym tworzymy dwa zbiory. Ponadto robimy to samo dla wcześniej wyselekcjonowanych przez nas zmiennych.

2.3 Metoda k-najbliższych sąsiadów

```
#zbiór uczący - wszystkie cechy
blad.klasyf(etykietki.prog.learning.veh, etykietki.learning.veh)

## $macierz.pomylek
##                etykietki
## etykietki.prog bus opel saab van
##           bus  150    3    4   3
##           opel   0   93   28   3
##           saab   0   41  107   1
##           van    2    8    5 116
##
## $blad.klasyf
## [1] 0.1737589

#zbiór testowy - wszystkie cechy
blad.klasyf(etykietki.prog.test.veh, etykietki.test.veh)

## $macierz.pomylek
##                etykietki
## etykietki.prog bus opel saab van
##           bus   64    1    3   9
##           opel   0   26   23   1
##           saab   1   32   42   3
##           van    1    8    5  63
##
## $blad.klasyf
## [1] 0.3085106
```

Jak widać metoda k-najbliższych sąsiadów daj niezłe rezultaty. Zobaczmy jakie efekty da metoda cross-validation:

```
##
## Call:
## errorest.data.frame(formula = etykietki.veh ~ ., data = veh2,
##     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:  0.2742
##
## Call:
## errorest.data.frame(formula = etykietki.veh ~ ., data = veh2,
##     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:  0.3113
## Standard deviation: 0.0034
##
## Call:
## errorest.data.frame(formula = etykietki.veh ~ ., data = veh2,
##     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:  0.2675
```

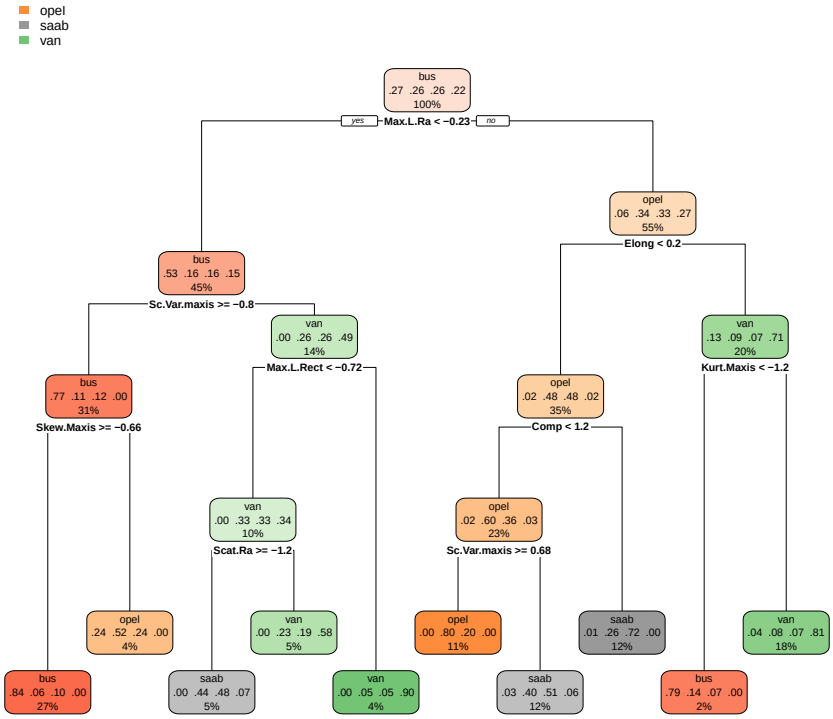
Przetestujmy teraz tę metodę dla różnej liczby sąsiadów:

```
##
## Call:
## errorest.data.frame(formula = etykietki.veh ~ ., data = veh2,
##     model = my.ipredknn, predict = my.predict, estimator = "632plus",
##     est.para = control.errorest(nboot = 50), ile.sasiadow = 1)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2307
##
## Call:
## errorest.data.frame(formula = etykietki.veh ~ ., data = veh2,
##     model = my.ipredknn, predict = my.predict, estimator = "632plus",
##     est.para = control.errorest(nboot = 50), ile.sasiadow = 10)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2774
##
## Call:
## errorest.data.frame(formula = etykietki.veh ~ ., data = veh2,
##     model = my.ipredknn, predict = my.predict, estimator = "632plus",
##     est.para = control.errorest(nboot = 50), ile.sasiadow = 15)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2923
```

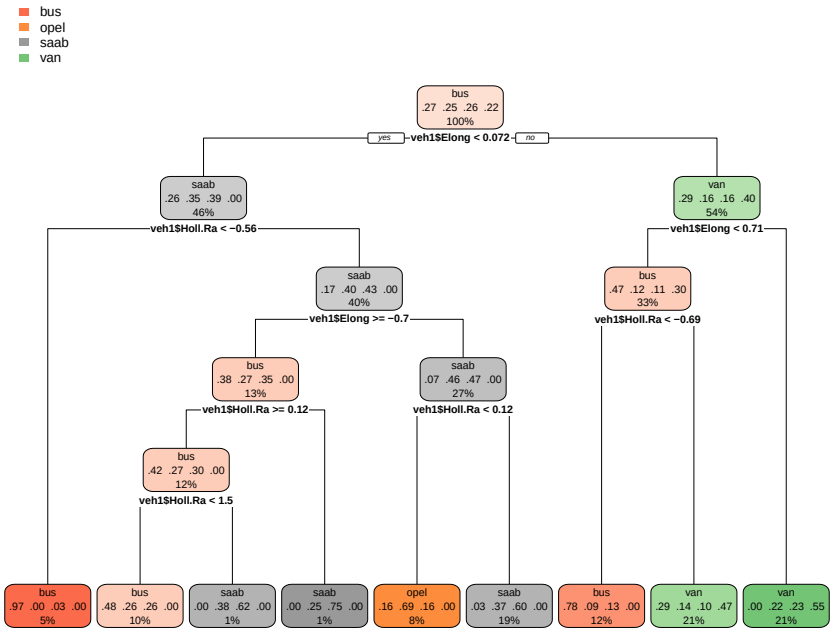
```
##
## Call:
## errorest.data.frame(formula = etykietki.veh ~ ., data = veh2,
##     model = my.ipredknn, predict = my.predict, estimator = "632plus",
##     est.para = control.errorest(nboot = 50), ile.sasiadow = 20)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2994
```

Jak widać w naszym przypadku wraz ze wzrostem liczby sąsiadów rośnie też błąd klasyfikacyjny.

2.4 Drzewa klasyfikacyjne



Rysunek 13: Drzewo klasyfikacyjne - wszystkie cechy



Rysunek 14: Drzewo klasyfikacyjne - wybrane cechy

```r
# zbiór uczący - wszystkie
conf.mat.learning <- table(pred.labels.learning, learning.set.veh$etykietki.veh)
conf.mat.learning
```

```
##
## pred.labels.learning bus opel saab van
##                 bus  140   11   17   0
##                 opel   5   62   18   0
##                 saab   3   56   95   6
##                 van    4   16   14 117
```

```r
(error.rate.learning <- (nrow(learning.set.veh) - sum(diag(conf.mat.learning))) / nrow(l
```

```
## [1] 0.2659574
```

```r
#zbiór testowy - wszystkie
conf.mat.test <- table(pred.labels.test, test.set.veh$etykietki.veh)
conf.mat.test
```

```
##
## pred.labels.test bus opel saab van
##             bus   58   10   10   0
##             opel   6   19   13   0
##             saab   0   33   42   3
##             van    2    5    8  73
```

```r
(error.rate.test <- (nrow(test.set.veh) - sum(diag(conf.mat.test))) / nrow(test.set.veh)
```

```
## [1] 0.3191489
```

```r
# zbiór uczący - wybrane
conf.mat.learning.wybrane <- table(pred.labels.learning.wybrane, learning.set.wybrane$et
conf.mat.learning.wybrane
```

```
##
## pred.labels.learning.wybrane bus opel saab van
##                         bus  111   21   25   0
##                         opel   7   31    7   0
##                         saab   3   45   76   0
##                         van   34   42   40 122
```

```r
(error.rate.learning.wybrane <- (nrow(learning.set.wybrane) - sum(diag(conf.mat.learning
```

```
## [1] 0.3971631
```

```r
# zbiór testowy - wybrane
conf.mat.test.wybrane <- table(pred.labels.test.wybrane, test.set.wybrane$etykietki.veh)
conf.mat.test.wybrane
```

```
##
## pred.labels.test.wybrane bus opel saab van
##                     bus   41   13   14   0
##                     opel   2   15    3   0
##                     saab   4   30   33   0
##                     van   16   15   19  77
```

```
(error.rate.test.wybrane <- (nrow(test.set.wybrane) - sum(diag(conf.mat.test.wybrane)))
```

```
## [1] 0.4113475
```

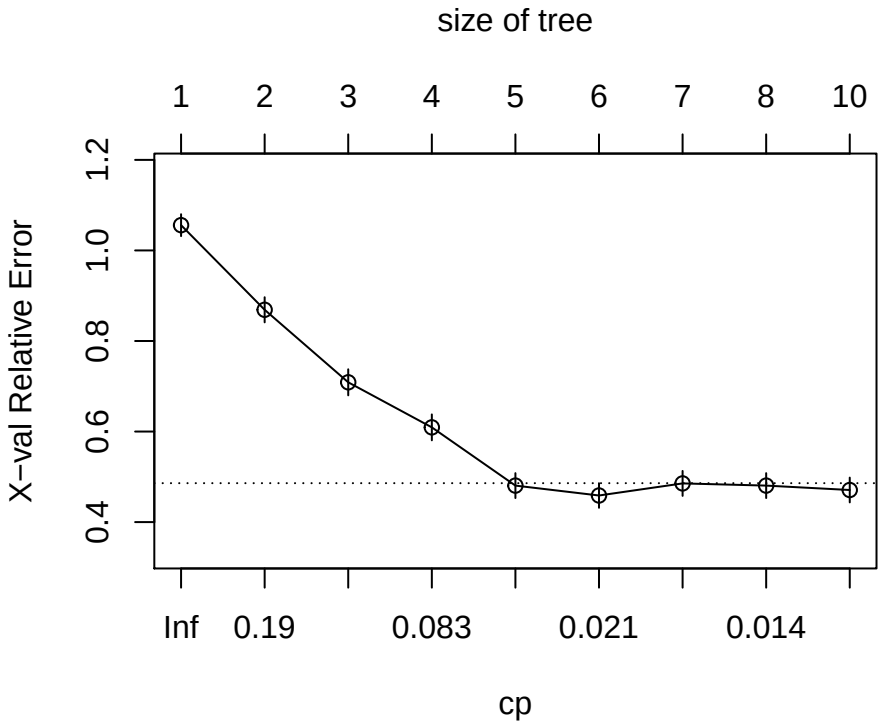
W przypadku drzew klasyfikacynych zbiory złożone ze wszystkich cech sprawdzają się lepiej od tych zawierające jedynie wybrane wcześniej cechy.

```
# Zmiana parametrów -- konstruujemy złożone drzewo
veh.tree.complex  <- rpart(model, data=learning.set.veh, control=rpart.control(cp=.01, m

# Wybór parametru złożoności (cp)
printcp(veh.tree.complex)
```

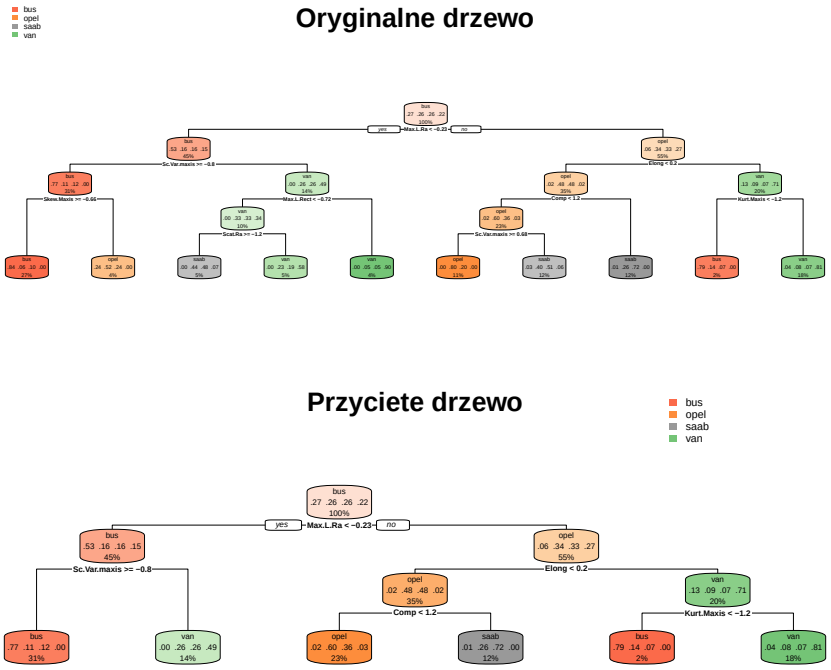
```
##
## Classification tree:
## rpart(formula = model, data = learning.set.veh, control = rpart.control(cp = 0.01,
##     minsplit = 5, maxdepth = 20))
##
## Variables actually used in tree construction:
## [1] Comp         Elong        Kurt.Maxis   Max.L.Ra     Max.L.Rect
## [6] Sc.Var.maxis Scat.Ra      Skew.Maxis
##
## Root node error: 412/564 = 0.7305
##
## n= 564
##
##         CP nsplit rel error  xerror     xstd
## 1 0.211165      0   1.00000 1.05583 0.024210
## 2 0.172330      1   0.78883 0.86893 0.027755
## 3 0.092233      2   0.61650 0.70874 0.028803
## 4 0.075243      3   0.52427 0.60922 0.028647
## 5 0.026699      4   0.44903 0.48058 0.027513
## 6 0.016990      5   0.42233 0.45874 0.027209
## 7 0.014563      6   0.40534 0.48544 0.027576
## 8 0.013350      7   0.39078 0.48058 0.027513
## 9 0.010000      9   0.36408 0.47087 0.027382
```

```
bestcp <- veh.tree.complex$cptable[which.min(veh.tree.complex$cptable[,"xerror"]),"CP"]
bestcp # najmniejszy błąd
```

```
## [1] 0.01699029
```



Rysunek 15: Wybór parametru złożoności



Rysunek 16: Drzewo oryginalne i przycięte

2.5 Naiwny klasyfikator Bayesowski

```
#zbiór uczący - wszystkie cechy
blad.klasyf(etykietki.prog.learning.veh, etykietki.learning.veh)

## $macierz.pomylek
##                etykietki
## etykietki.prog bus opel saab van
##           bus   34    1    3   6
##           opel  22   73   38   2
##           saab   9   28   58   4
##           van   87   43   45 111
##
## $blad.klasyf
## [1] 0.5106383

#zbiór testowy - wszystkie cechy
blad.klasyf(etykietki.prog.test.veh, etykietki.test.veh)

## $macierz.pomylek
##                etykietki
## etykietki.prog bus opel saab van
##           bus   14    1    2   6
##           opel   8   33   23   3
##           saab   7   13   27   2
##           van   37   20   21  65
##
## $blad.klasyf
## [1] 0.5070922

#zbiór uczący - wybrane cechy
blad.klasyf(etykietki.prog.learning.wybrane, etykietki.learning.wybrane)

## $macierz.pomylek
##                etykietki
## etykietki.prog bus opel saab van
##           bus   90    9   15  19
##           opel  17   59   48   8
##           saab  33   35   54   8
##           van   15   36   31  87
##
## $blad.klasyf
## [1] 0.4858156

#zbiór testowy - wybrane cechy
blad.klasyf(etykietki.prog.test.wybrane, etykietki.test.wybrane)
```

```
## $macierz.pomylek
##                etykietki
## etykietki.prog bus opel saab van
##           bus   38    8    5  15
##           opel   7   36   31   3
##           saab   8   18   21   5
##           van   10   11   12  54
##
## $blad.klasyf
## [1] 0.4716312
```

Naiwny klasyfikator Bayesowski w naszym przypadku sprawdza się słabo - błąd klasyfikacyjny oscylujący wokół 50% ciężko uznać choćby za zadowalający. Zbiory z wybranymi cechami dają zbliżone rezultaty do tych ze wszystkimi.

2.6 Wnioski

- W naszym przypadku najlepsza okazała się metoda k-najbliższych sąsiadów.
- Zadowalające wyniki udało się uzyskać również dzięki metodzie drzew klasyfikacyjnych.
- Zdecydowanie najsłabiej wypadł naiwny klasyfikator Bayesa.