Raport 3
Klasyfikacja na bazie modelu regresji
Porównanie metod klasyfikacji

Romana Żmuda
249706
Adrian Kit
249746

18 maja 2020

Spis treści

1 Wstęp - wprowadzenie do podanych zagadnień

W tym sprawozdaniu będziemy badali dwa zbiory danych. Pierwszy z nich, podobnie jak w poprzednim raporcie, dotyczy kwiatków irysów. W drugim zadaniu możemy wybrać spośród pięciu zbiorów. Zdecydowaliśmy się na zbiór o winach.

Celem raportu jest:

- Klasyfikacja na bazie modelu regresji liniowej
- Porównanie wybranych metod klasyfikacji

Poniżej zamieściliśmy krótkie charakteryzacje metod wykorzystanych w raporcie.

1.1 Opis wybranych pojęć

- *Klasyfikacja*
  Klasyfikacja danych to metoda, która umożliwia tworzenie klasyfikatorów. Klasyfikator to pewna funkcja (zazwyczaj w postaci algorytmu lub programu komputerowego), która przypisuje analizowane obiekty do jednej z rozpatrywanych klas. Zazwyczaj klasyfikator konstruuje się na podstawie pewnego zbioru danych uczących, zawierającego opisy i klasy pewnej liczby obiektów.
- *Klasyfikator K najbliższych sąsiadów (KNN)*
  Przykładem najprostszego klasyfikatora jest klasyfikator K najbliższych sąsiadów (ang. K Nearest Neighbors, KNN). Jest to funkcja, która dla rozpatrywanego obiektu, wyszukuje w zbiorze danych uczących K obiektów najbardziej do niego podobnych i zwraca etykietę tej klasy, do której należy większość z tych K obiektów. Często spotykanym przypadkiem klasyfikacji danych jest klasyfikacja danych wektorowych. Obiekty są opisywane przez wektory liczb rzeczywistych. Wówczas prosty klasyfikator K najbliższych sąsiadów (KNN) polega na policzeniu odległości opisu rozpatrywanego obiektu od opisów każdego obiektu ze zbioru danych uczących (często stosowana jest odległość Euklidesowa), a następnie na wybraniu K obiektów o najmniejszych odległościach od x.
- *Naiwny klasyfikator bayesowski*
  Naiwny klasyfikator bayesowski to prosty klasyfikator oparty na wnioskowaniu statystycznym. Naiwność klasyfikatora przejawia się założeniem niezależności zmiennych losowych reprezentujących cechy klasyfikowanych obiektów, które nie zawsze, a raczej prawie nigdy, nie jest prawdziwe w zagadnieniach rzeczywistych. Popularność naiwnego klasyfikatora bayesowskiego wynika z jego prostoty, zarówno konceptualnej jak i obliczeniowej, która mimo swoich naiwnych założeń często prowadzi do dosyć dobrych wyników.
- *Drzewa klasyfikacyjne*
  Inną klasą klasyfikatorów są cieszące się dużą popularnością metody bazujące na drzewach decyzyjnych (klasyfikacyjnych). W tym przypadku klasyfikator jest reprezentowany przez drzewo binarne, w którego węzłach znajdują się pytania o wartości określonej cechy, a w liściach znajdują się oceny klas.
- *Metoda regresji liniowej*
  Głównym celem regresji jest zbudowanie modelu, który podobnie jak model klasyfikacji posłuży do predykcji jednej zmiennej na podstawie znanych wartości innych zmiennych. Podstawową różnicą pomiędzy regresją i klasyfikacją jest to, że w klasyfikacji przewidywana zmienna przyjmuje wartość kategoryczną, natomiast w regresji celem jest przewidzenie zmiennej przyjmującej wartość ciągłą (numeryczną).

2 Zadanie 1 - Klasyfikacja na bazie modelu regresji liniowej

2.1 Wybór i przygotowanie danych

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

ncol(iris) # ilość kolumn

## [1] 5

nrow(iris) #ilość przypadków

## [1] 150

sapply(iris, class) # identyfikacja cech

## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##    "numeric"    "numeric"    "numeric"    "numeric"     "factor"

ilebrakujacych<-sum(is.na(iris))
ilebrakujacych # liczba brakujących danych

## [1] 0
```

2.2 Podział danych na zbiór uczący i testowy

```
n <- dim(iris)[1]
learn.indx <- sample(1:n,2/3*n)
learn.set <- iris[learn.indx,] # zbiór uczący
test.set <- iris[-learn.indx,] # zbiór testowy
head(learn.set)

##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 117          6.5         3.0          5.5         1.8  virginica
## 61           5.0         2.0          3.5         1.0 versicolor
## 130          7.2         3.0          5.8         1.6  virginica
## 28           5.2         3.5          1.5         0.2     setosa
## 116          6.4         3.2          5.3         2.3  virginica
## 49           5.3         3.7          1.5         0.2     setosa
```

2.3 Konstrukcja klasyfikatora i wyznaczenie prognoz

Definiujemy podstawowe zmienne potrzebne do dalszej pracy.

Potrzebujemy reprezentacji macierzowej naszych danych. Pierwszą kolumnę wypełniamy jedynkami (uwzględnienie stałej w modelu regresji).

```
Xu <- cbind(rep(1,100), learn.set[,1:4])
Xu <- as.matrix(Xu)
Xt <- cbind(rep(1,50), test.set[,1:4])
Xt <- as.matrix(Xt)
```

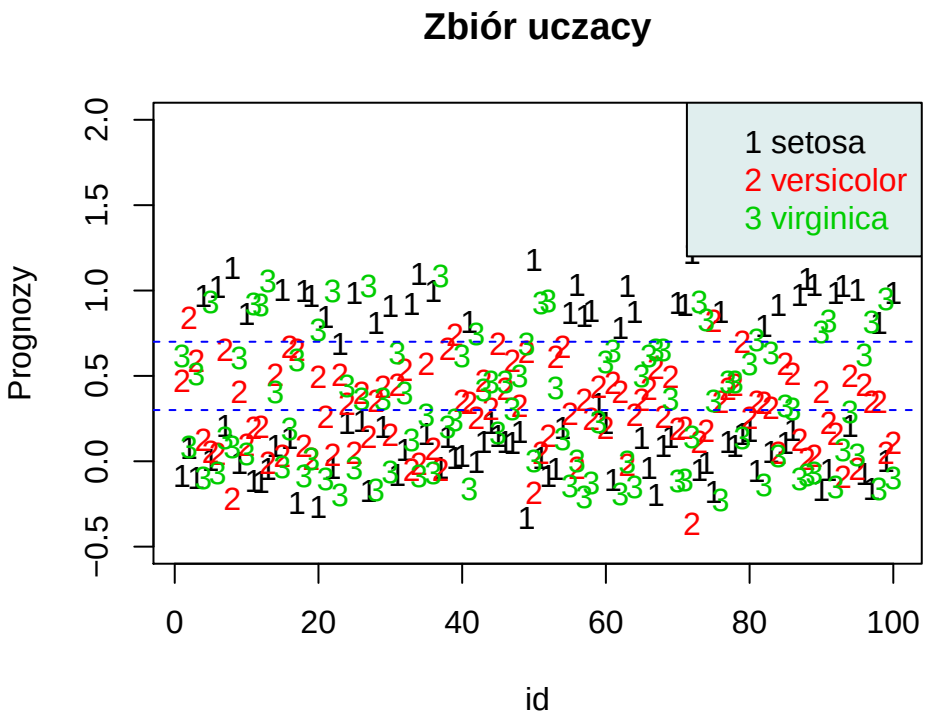
Następnie kodujemy poszczególne klasy. Rezultat umieszczamy w macierzy wskaźnikowej Y. Zaczniemy od wygenerowania macierzy zerowej, zmieniając zera na jedynki, jeżeli i-ta obserwacja należy do j-tej klasy.

```
##      [,1] [,2] [,3]
## [1,]    0    0    1
## [2,]    0    1    0
## [3,]    0    0    1
## [4,]    1    0    0
## [5,]    0    0    1
## [6,]    1    0    0
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 117          6.5         3.0          5.5         1.8  virginica
## 61           5.0         2.0          3.5         1.0 versicolor
## 130          7.2         3.0          5.8         1.6  virginica
## 28           5.2         3.5          1.5         0.2     setosa
## 116          6.4         3.2          5.3         2.3  virginica
## 49           5.3         3.7          1.5         0.2     setosa
```

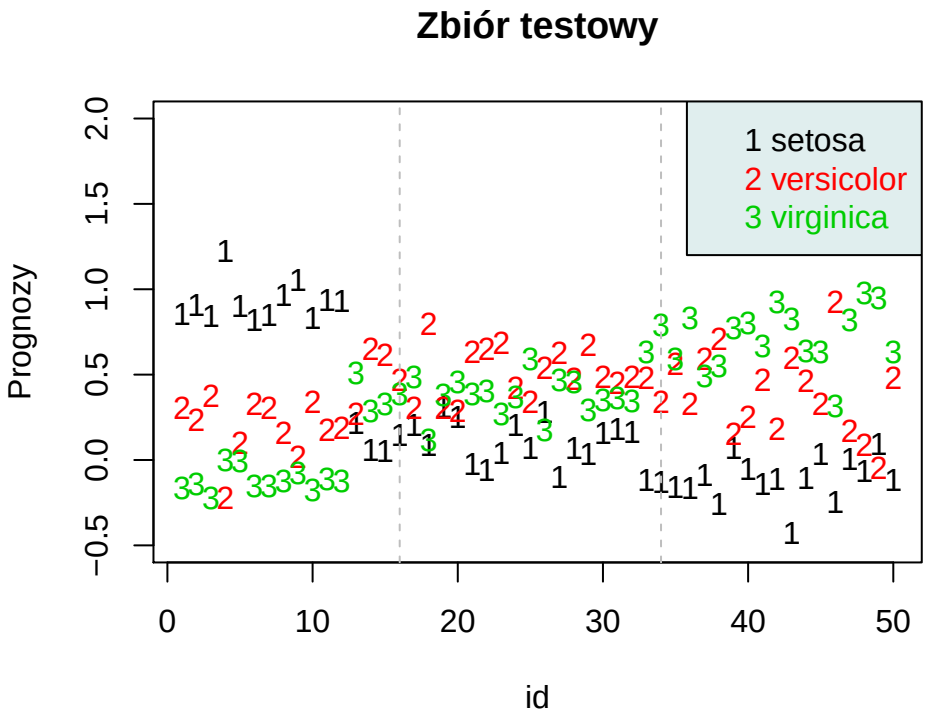
Istotnie, na przykładzie pierwszych sześciu rekordów, widać, że kodowanie jest poprawne (setosa - "1", versicolor - "2", virginica - "3").
Przedstawimy prognozowane prawdopodobieństwa przynależności do danych klas obiektów ze zbioru

```
B <- solve(t(Xu)%*%Xu) %*% t(Xu) %*% Yu #klasyfikator
Yu.hat <- Xu%*%B
Yt.hat <- Xt%*%B
head(Yu.hat)

##              [,1]       [,2]        [,3]
## 117 -0.084919250 0.46949848  0.61542077
## 61   0.078296166 0.83923972  0.08246412
## 130 -0.095894728 0.58787670  0.50801803
## 28   0.969927065 0.12716622 -0.09709328
## 116  0.008216176 0.05637633  0.93540750
## 49   1.024442100 0.04557983 -0.07002193
```

Poniżej wykresy prognoz dla zbioru uczącego i testowego



Rysunek 1: Prognozy dla zbioru uczącego



Rysunek 2: Prognozy dla zbioru testowego

Wnioski:

- Na obu wykresach widać wyraźne oddzielenie setosy od pozostałych klas.
- Wiele obserwacji może zostać błędnie sklasyfikowanych (głównie versicolor i virginica), ponieważ ich prognozy wynoszą około 0,5. Na tej podstawie (wspomagając się obserwacjami) możemy wnioskować, że wystąpuje zjawisko maskowania klas.
- Z wykresu dla zbioru uczącego można odczytać, że obserwacje należące do klas setosa (1) i virginica (3) będą dobrze sklasyfikowane. Dla versicolor (2) mogą wystąpić błędy.

2.4 Ocena jakości modelu

Macierz pomyłek i poprawność dla zbioru uczącego:

```
##                       prognozowane.etykietki
## rzeczywiste.etykietki setosa versicolor virginica
##             setosa         38          0         0
##             versicolor      0         20        10
##             virginica       0          4        28
## [1] 0.86
```

Macierz pomyłek i poprawność dla zbioru testowego:

```
##                       prognozowane.etykietki
## rzeczywiste.etykietki setosa versicolor virginica
##             setosa         12          0         0
##             versicolor      0         15         5
##             virginica       0          3        15
## [1] 0.84
```

Wnioski:

- Nasze przypuszczenia okazały się słuszne, około 40 % gatunku versicolor zostało błędnie sklasyfikowanych.
- Poprawność kształtuje się na poziomie 80%, jednak czy da się uzyskać większą?

2.5 Budowa modelu liniowego dla rozszerzonej przestrzeni cech

Zbudujemy nowy model liniowy z dodatkowymi cechami:
$PL^2, PW^2, SL^2, SW^2, PL * PW, PL * SW, PL * SL, PW * SL, PW * SW, SL * SW$

Tak wygląda nasz nowy zbiór:

```
##    SL  SW  SL^2  SW^2 PL^2 PW^2 SL.SW SL.PL SL.PW SW.PL SW.PW PL.PW Species
## 1 5.1 3.5 26.01 12.25 1.96 0.04 17.85  7.14  1.02  4.90  0.70  0.28  setosa
## 2 4.9 3.0 24.01  9.00 1.96 0.04 14.70  6.86  0.98  4.20  0.60  0.28  setosa
## 3 4.7 3.2 22.09 10.24 1.69 0.04 15.04  6.11  0.94  4.16  0.64  0.26  setosa
## 4 4.6 3.1 21.16  9.61 2.25 0.04 14.26  6.90  0.92  4.65  0.62  0.30  setosa
## 5 5.0 3.6 25.00 12.96 1.96 0.04 18.00  7.00  1.00  5.04  0.72  0.28  setosa
## 6 5.4 3.9 29.16 15.21 2.89 0.16 21.06  9.18  2.16  6.63  1.56  0.68  setosa
```

Krótki opis danych:

```
ncol(iris.new) # ilość kolumn

## [1] 15

nrow(iris.new) #ilość przypadków

## [1] 150

sapply(iris.new, class) # identyfikacja cech

##        SL        SW        PL        PW      SL^2      SW^2      PL^2      PW^2
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##     SL.SW     SL.PL     SL.PW     SW.PL     SW.PW     PL.PW   Species
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"  "factor"

ilebrakujacych<-sum(is.na(iris.new))
ilebrakujacych # liczba brakujących danych

## [1] 0
```

2.5.1 Klasyfikator, Prognozy i Wnioski

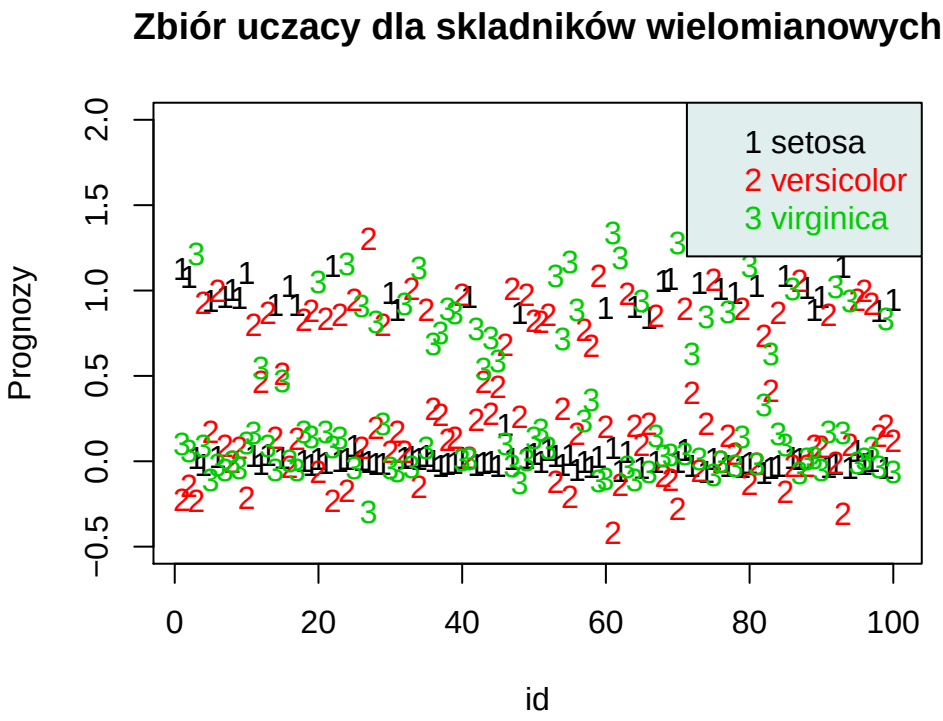
Tworzymy zbiór uczący, na którym skonstruujemy klasyfikator regresji oraz zbiór testowy, na którym zweryfikujemy czy stworzony model jest poprawny.

```
n <- dim(iris.new)[1]
learn.indx <- sample(1:n,2/3*n)

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

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

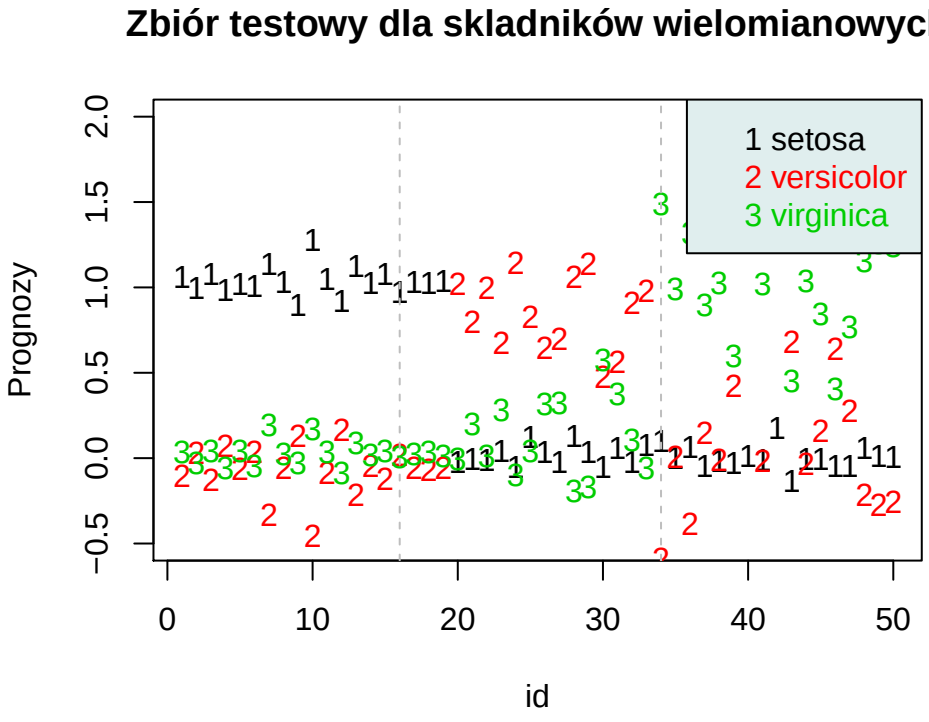
Wyliczamy klasyfikator ze zbioru uczącego, a następnie wyznaczamy prognozy dla obydwu zbiorów:

```
B <- solve(t(Xu)%*%Xu) %*% t(Xu) %*% Yu #klasyfikator
Yu.hat <- Xu%*%B
Yt.hat <- Xt%*%B
```

Poniżej wykresy prognoz dla zbioru uczącego i testowego po uzupełnieniu wyjściowych cech o składniki wielomianowe stopnia 2:



Rysunek 3: Prognozy dla zbioru uczącego



Rysunek 4: Prognozy dla zbioru testowego

Wyznaczenie prognoz dla poszczególnych ID zbioru testowego (badanie maksymalnej wartości etykietki).

```
##   1   2   5  10  11  13  15  18  19  23  29  31  38  39  43  46  47  49  50  51
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2
##  56  59  60  61  65  67  69  80  82  84  85  87  94 101 103 106 109 110 111 115
##   2   2   2   2   2   2   2   2   2   3   2   2   2   3   3   3   3   3   3   3
## 116 119 120 121 125 134 135 136 141 145
##   3   3   2   3   3   2   3   3   3   3
```

Wnioski:

- Wyraźny podział na 3 podzbiory, zgodnie z naszym podziałem.
- W tym przypadku nie ma maskowania środkowej klasy, jednak w 3 zbiorze, dla virginici, widać że prognoza dla klasy numer 3 nie zawsze jest najwyżej, tym samym klasa wybrana przez model regresji może być niewłaściwa.

Sprawdzimy ile z nich jest rozpoznana błędnie, używając macierzy pomyłek. Zaczniemy od zbioru uczącego:

```
##                   prognozowane.etykietki.u
## etykietki.klas.ucz setosa versicolor virginica
##         setosa         31          0         0
##         versicolor      0         36         0
##         virginica       0          0        33
## [1] "Dokładność klasyfikacji dla zbioru uczącego: "
## [1] 1
```

Błąd przydzielenia elementu do złej klasy jest bliski $1 - 3\%$. Jest to o wiele dokładniejszy pomiar, niż w przypadku poprzednich podpunktów.

Sprawdźmy zbiór testowy:

```
##                    prognozowane.etykietki.t
## etykietki.klas.test setosa versicolor virginica
##          setosa         19          0         0
##          versicolor      0         13         1
##          virginica       0          2        15
## [1] "Dokładność klasyfikacji dla zbioru testowego: "
## [1] 0.94
```

Prawie wszystkie elementy ze zbioru testowego zostały przydzielone do właściwych etykiet.

Podsumowanie i wnioski:

- Uzupełniając dane o składniki wielomianowe stopnia 2 uzyskujemy lepsze rezultaty niż dla danych wyjściowych z błędem wynoszącym maks kilka punktów procentowych. Jest to łatwy sposób na poprawienie dokładności klasyfikatora.
- Pozbywamy się także efektu maskowania klas.

3 Zadanie 2 - Porównanie metod klasyfikacji

3.1 Wybór i przygotowanie danych

Wybraliśmy dane z pakietu *HDclassif* o nazwie "WINE", poniżej krótka charakteryzacja pliku:

```
library(HDclassif)
data(wine)
colnames(wine) <- c('Type', 'Alcohol', 'Malic', 'Ash',
                    'Alcalinity', 'Magnesium', 'Phenols',
                    'Flavanoids', 'Nonflavanoids',
                    'Proanthocyanins', 'Color', 'Hue',
                    'Dilution', 'Proline')
wine$Type <- as.factor(wine$Type)


head(wine)
```

```
##   Type Alcohol Malic  Ash Alcalinity Magnesium Phenols Flavanoids Nonflavanoids
## 1    1   14.23  1.71 2.43       15.6       127    2.80       3.06          0.28
## 2    1   13.20  1.78 2.14       11.2       100    2.65       2.76          0.26
## 3    1   13.16  2.36 2.67       18.6       101    2.80       3.24          0.30
## 4    1   14.37  1.95 2.50       16.8       113    3.85       3.49          0.24
## 5    1   13.24  2.59 2.87       21.0       118    2.80       2.69          0.39
## 6    1   14.20  1.76 2.45       15.2       112    3.27       3.39          0.34
##   Proanthocyanins Color  Hue Dilution Proline
## 1            2.29  5.64 1.04     3.92    1065
## 2            1.28  4.38 1.05     3.40    1050
## 3            2.81  5.68 1.03     3.17    1185
## 4            2.18  7.80 0.86     3.45    1480
## 5            1.82  4.32 1.04     2.93     735
## 6            1.97  6.75 1.05     2.85    1450
```

```
ncol(wine) # ilośc kolumn
```

```
## [1] 14
```

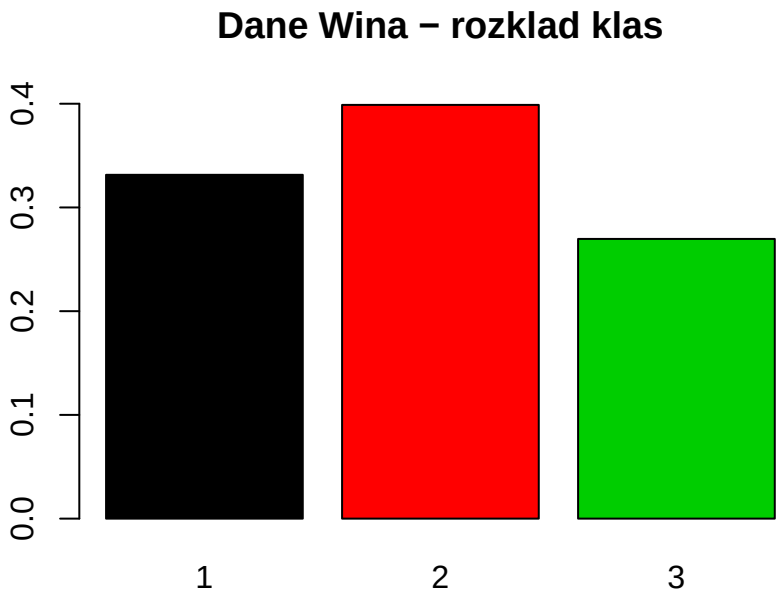
```
nrow(wine) #ilość przypadków
```

```
## [1] 178
```

```
sapply(wine, class) # identyfikacja cech
```

```
##            Type         Alcohol           Malic             Ash      Alcalinity
##        "factor"       "numeric"       "numeric"       "numeric"       "numeric"
##       Magnesium         Phenols      Flavanoids   Nonflavanoids Proanthocyanins
##       "integer"       "numeric"       "numeric"       "numeric"       "numeric"
##           Color             Hue        Dilution         Proline
##       "numeric"       "numeric"       "numeric"       "integer"
```

```r
ilebrakujacych<-sum(is.na(wine))
ilebrakujacych # liczba brakujących danych

## [1] 0

#procentowu udział 3 odmian wina
barplot(prop.table(table(wine$Type)), col=1:3,
        main="Dane Wina - rozkład klas", ylim=c(0,0.4))
```



Plik zawiera 14 cech i 178 przypadków, w których kolumny odpowiednio nazywają się:

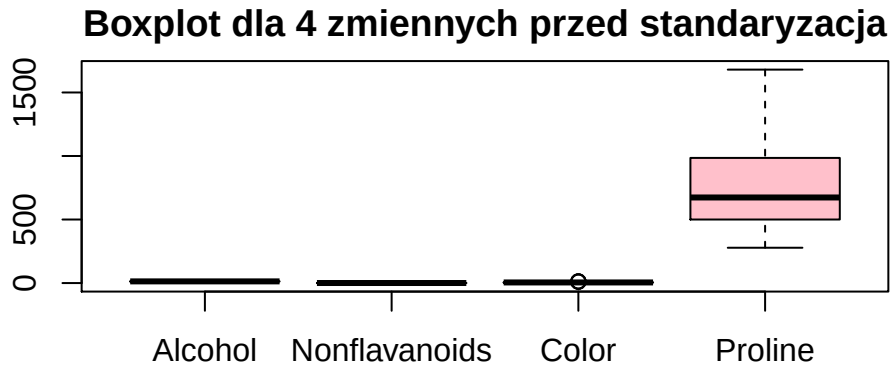
- Type - Wektor klasowy, trzy różne odmiany wina reprezentowane przez trzy liczby całkowite od 1 do 3.
- Alcohol - procent alkoholu
- Malic - wskaźnik określający jabłkowość
- Ash - wskaźnik winny mętności
- Alcalinity - Alkaliczność
- Magnesium - Magnez
- Phenols - Fenole
- Flavanoids - Flawanoidy
- Nonflavanoids - Nieflawanoidy
- Proanthocyanins - Proantocyjaniny

- Color - Kolor
- Hue - Odcień
- Dilution - Roztwór
- Proline - Prolina

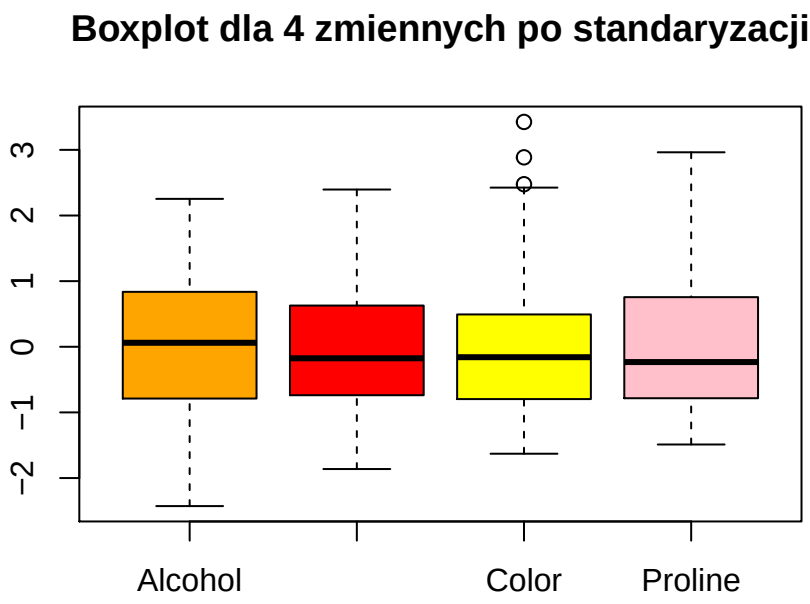
Zaczniemy od sprawdzenia zmienności cech. Jeśli wariancje poszczególnych zmiennych będą zbyt zróżnicowane, wtedy konieczna będzie standaryzacja.

```
##     Alcohol    Malic        Ash Alcalinity Magnesium   Phenols Flavanoids
## 1 0.6590623 1.248015 0.07526464   11.15269  203.9893 0.3916895  0.9977187
##   Nonflavanoids Proanthocyanins    Color        Hue  Dilution  Proline
## 1    0.01548863       0.3275947 5.374449 0.05224496 0.5040864 99166.72
```

Obserwacje: Wariancje wyjściowych zmiennych są bardzo zróżnicowane. Celem uniknięcia dominacji zmiennej o bardzo dużej wariancji (zmienna Proline) przeprowadzimy standaryzację.

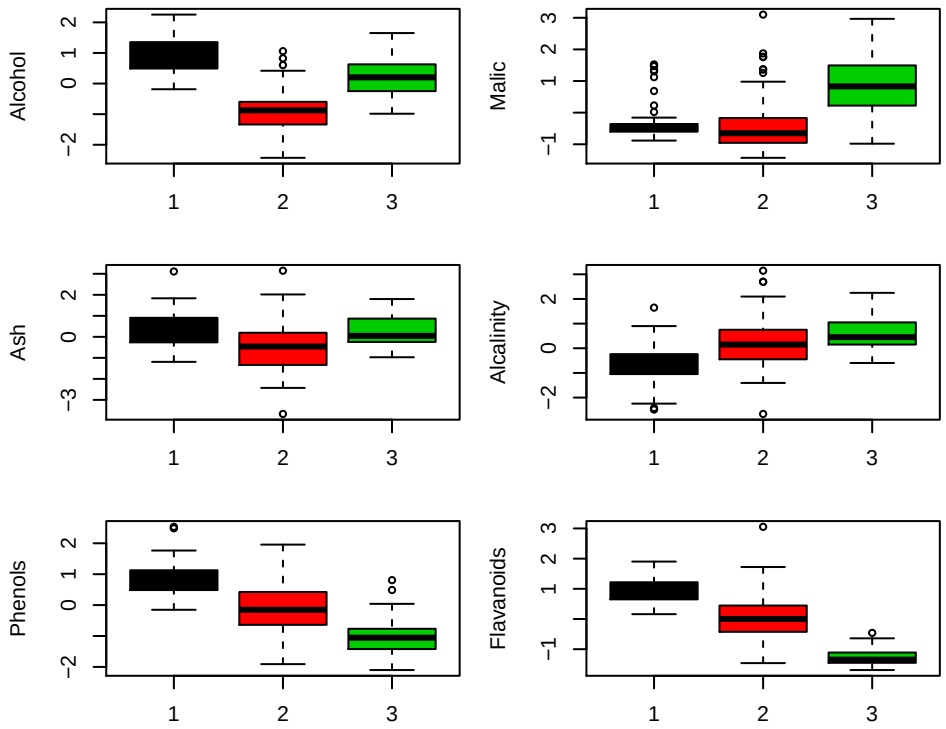


Rysunek 5: Rozrzut zmiennych przed standaryzacją

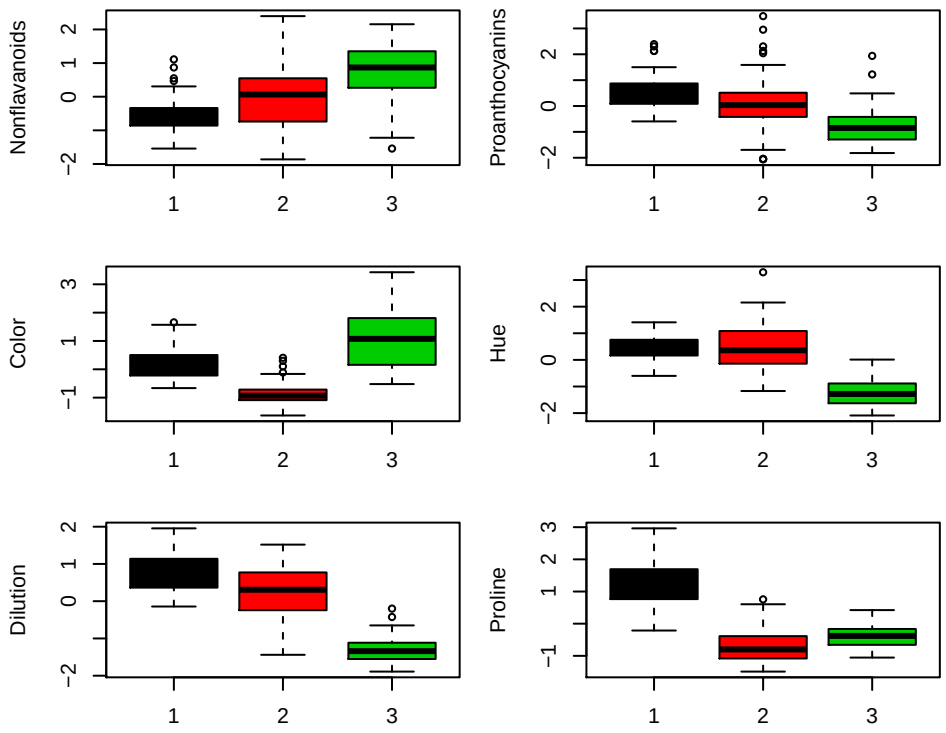


Rysunek 6: Rozrzut zmiennych po standaryzacji

Zdolności dyskryminacyjne zmiennych:



Rysunek 7: Zdolności dyskryminacyjne poszczególnych zmiennych



Rysunek 8: Zdolności dyskryminacyjne poszczególnych zmiennych

Wniosek: Najlepszymi cechami do separacji klas są Phenols i Flavanoids.

3.1.1 Podział danych na zbiór uczący i testowy

Stworzymy 2 zbiory uczącę i 2 zbiory testowe. W jednym znajdą się wszystkie cechy, w drugim zaś te o najlepszej zdolności dyskryminacyjnej (Phenols i Flavanoids).

```
etykietki<-wine$Type
etykietki<-data.frame(etykietki)
wine2<-cbind(etykietki, wine1)

np <- dim(wine2)[1]
learn.indx.all <- sample(1:np,2/3*np)
learn.set.all <- wine2[learn.indx.all,] # zbiór uczący wszystkich zmiennych
test.set.all <- wine2[-learn.indx.all,] # zbiór testowy wszystkich zmiennych

wine3<-wine2[7:8]
wine3<-cbind(etykietki, wine3)

etykietki.learn.all<-learn.set.all$etykietki
etykietki.test.all <- test.set.all$etykietki

n2 <-dim(wine3)[1]
learn.indx.subset <- sample(1:n2,2/3*n2)
learn.set.subset <- wine3[learn.indx.subset,] # zbiór uczący pozdbioru danych
test.set.subset <- wine3[-learn.indx.subset,] # zbiór testowy pozdbioru danych
```

```
etykietki.learn.subset<-learn.set.subset$etykietki
etykietki.test.subset <- test.set.subset$etykietki
```

Przeprowadzimy analizę trzema metodami na wyżej pokazanych podzbiorach. Porównamy uzyskane rezultaty i postaramy się wybrać najlepszą z metod, badając przy tym różne przypadki poboczne.

3.2 Naiwny Bayes

Wyznaczamy macierz pomyłek i błąd klasyfikacyjny dla metody naiwnego Bayesa. Na początek przeanalizujemy wyjściowy zbiór danych z podziałem na zbiór uczący i testowy.
Macierz pomyłek i błąd dla zbioru uczącego wyjściowych danych:

```
## $macierz.pomylek
##                etykietki
## etykietki.prog  1  2  3
##              1 37  0  0
##              2  1 43  0
##              3  0  1 36
##
## $blad.klasyf
## [1] 0.01694915
```

Macierz pomyłek i błąd dla zbioru testowego wyjściowych danych:

```
## $macierz.pomylek
##                etykietki
## etykietki.prog  1  2  3
##              1 21  1  0
##              2  0 25  0
##              3  0  1 12
##
## $blad.klasyf
## [1] 0.03333333
```

Następnie rozważymy podzbiór danych wyjściowych z cechami o najlepszej zdolności dyskryminacyjnej.
Macierz pomyłek i błąd dla zbioru uczącego pozdbioru wyjściowych danych:

```
## $macierz.pomylek
##                etykietki
## etykietki.prog  1  2  3
##              1 28 12  0
##              2  6 35  4
##              3  0  5 28
##
## $blad.klasyf
## [1] 0.2288136
```

Macierz pomyłek i błąd dla zbioru testowego pozdbioru wyjściowych danych:

```
## $macierz.pomylek
##                etykietki
## etykietki.prog  1  2  3
##              1 18  3  0
##              2  7 12  0
##              3  0  4 16
##
## $blad.klasyf
## [1] 0.2333333
```

Wnioski:

- Analiza całego zbioru: Podobne, niemal identyczne wyniki zbiorów uczącego i testowego. Błąd kształtuje się w okolicach 2%, co jest świetnym rezultatem.
- Analiza podzbioru cech o najlepszych zdolnościach dyskryminacyjnych: Wyniki również są podobne dla obu zbiorów, jednak na wiele niższym poziomie, bowiem błąd wynosi około 20%

Otrzymane wyniki porównamy z tymi uzyskami dzięki algorytmowi "bootstrap":

```
library(ipred)
#Wine3-pozdbiór wyjściowych danych z cechami o najlepszej zdolności dyskryminacyjnej,
#bez podziału na zbiór uczący i testowy

#Wine2-wyjściowe dane po standaryzacji

my.predict  <- function(model, newdata)
{ predict(model, newdata=newdata, type="class") }

my.naiveBayes <- function(formula, data)
{ naiveBayes(formula=formula,data=data)}

#Testujemy model dla nboot=50, nboot=3
(blad.bootstrap <- errorest(etykietki~., wine3, model=my.naiveBayes, predict=my.predict,
 estimator="boot", est.para=control.errorest(nboot = 50)))

##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine3, model = my.naiveBayes,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2233
## Standard deviation: 0.0035
```

```r
(blad.bootstrap <- errorest(etykietki~., wine3, model=my.naiveBayes, predict=my.predict,
 estimator="boot", est.para=control.errorest(nboot = 3)))

##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine3, model = my.naiveBayes,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 3))
##
##   Bootstrap estimator of misclassification error
##   with 3 bootstrap replications
##
## Misclassification error:  0.2328
## Standard deviation: 0.0329

(blad.bootstrap <- errorest(etykietki~., wine2, model=my.naiveBayes, predict=my.predict,
 estimator="boot", est.para=control.errorest(nboot = 50)))

##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine2, model = my.naiveBayes,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.0295
## Standard deviation: 0.0024

(blad.bootstrap <- errorest(etykietki~., wine2, model=my.naiveBayes, predict=my.predict,
 estimator="boot", est.para=control.errorest(nboot = 3)))

##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine2, model = my.naiveBayes,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 3))
##
##   Bootstrap estimator of misclassification error
##   with 3 bootstrap replications
##
## Misclassification error:  0.0451
## Standard deviation: 0.0115

(blad.cv <- errorest(etykietki~., wine3, model=my.naiveBayes, predict=my.predict,
            estimator="cv", est.para=control.errorest(k = 5)))

##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine3, model = my.naiveBayes,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 5))
```

```
##
##   5-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2247

(blad.cv <- errorest(etykietki~., wine3, model=my.naiveBayes, predict=my.predict,
            estimator="cv", est.para=control.errorest(k = 100)))

##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine3, model = my.naiveBayes,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 100))
##
##   100-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2303
```

Wnioski:

- Algorytm "bootstrap" uzyskujemy podobne do siebie rezultaty, bez względu na wartość parametru nboot (testowane dla nboot=3 oraz nboot=50) zarówno dla zbioru wyjściowego jak i podzbioru z danymi o najlepszej zdolności dyskryminacyjnej. Maksymalna różnica wynosi kilka punktów procentowych.
- Wyniki otrzymanę metodą "bootstrap" są praktycznie identyczne jak te otrzymane macierzą pomyłek. Maksymalny błąd szacujemy na 1%.
- Algorytm "cross validation" daje niemal identyczne rezultaty jak algorytm "bootstrap" (dla różnych wartości k)

3.3 Metoda k-NN

Zaczniemy od analizy wyjściowego zbioru.

```
## $macierz.pomylek
##                 etykietki.rzecz
## etykietki.prog  1  2  3
##              1 38  2  0
##              2  0 40  0
##              3  0  2 36
##
## $blad.klasyf
## [1] 0.03389831
## $macierz.pomylek
##                 etykietki.rzecz
## etykietki.prog  1  2  3
##              1 21  4  0
##              2  0 21  0
##              3  0  2 12
```

```
##
## $blad.klasyf
## [1] 0.1
```

Model dla zbiorów o najlepszej zdolności dyskryminacyjnej:

```
# budujemy model
model.knn.2 <- ipredknn( etykietki~ ., data=learn.set.subset, k=5) #na zbiorze uczącym


etykietki.prog.learn.subset<- predict(model.knn.2, learn.set.subset,type="class")
etykietki.prog.test.subset<- predict(model.knn.2, test.set.subset,type="class")


(blad.klasyf(etykietki.prog.learn.subset, etykietki.learn.subset))

## $macierz.pomylek
##                etykietki.rzecz
## etykietki.prog  1  2  3
##              1 28 11  0
##              2  6 35  4
##              3  0  6 28
##
## $blad.klasyf
## [1] 0.2288136

(blad.klasyf(etykietki.prog.test.subset, etykietki.test.subset))

## $macierz.pomylek
##                etykietki.rzecz
## etykietki.prog  1  2  3
##              1 22  4  0
##              2  3 13  3
##              3  0  2 13
##
## $blad.klasyf
## [1] 0.2
```

Wnioski:

- Analiza dla wyjściowego zbioru: Występuje istotna różnica na poziomie kilku punktów procentowych między zbiorem uczącym a testowym, która nie występowała w metodzie naiwnego Bayesa.
- Analiza podzbioru cech o najlepszych zdolnościach dyskryminacyjnych: Wciąż występuje różnica w błędach, jednak nie jest już tak istotna, mimo wszystko nie występowała ona w metodzie Bayesa.
- Test był wykonywany dla k=5, może się okazać, że nie jest to optymalna liczba.

Badanie jakości klasyfikacji dla różnych kombinacji danych (bez podziału na zbiory uczące i testowe), ale dla różnej liczby najbliższych sąsiadów (testowane k=1, k=20 i k=100):

```
#Wine3-pozdbiór wyjściowych danych z cechami o najlepszej zdolności dyskryminacyjnej,
#bez podziału na zbiór uczący i testowy

#Wine2-wyjściowe dane po standaryzacji
my.predict  <- function(model, newdata) predict(model, newdata=newdata, type="class")
my.ipredknn <- function(formula1, data1, ile.sasiadow) ipredknn(formula=formula1,data=da
attach(wine2)

errorest(etykietki ~., wine2, model=my.ipredknn, predict=my.predict, estimator="boot",
         est.para=control.errorest(nboot = 50), ile.sasiadow=1)

##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine2, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##     ile.sasiadow = 1)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.0488
## Standard deviation: 0.0019

errorest(etykietki ~., wine2, model=my.ipredknn, predict=my.predict, estimator="boot",
         est.para=control.errorest(nboot = 50), ile.sasiadow=20)

##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine2, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##     ile.sasiadow = 20)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.0486
## Standard deviation: 0.0031

errorest(etykietki ~., wine2, model=my.ipredknn, predict=my.predict, estimator="boot",
         est.para=control.errorest(nboot = 50), ile.sasiadow=100)

##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine2, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##     ile.sasiadow = 100)
```

```
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.1446
## Standard deviation: 0.0142
```

```
errorest(etykietki ~., wine3, model=my.ipredknn, predict=my.predict, estimator="boot",
         est.para=control.errorest(nboot = 50), ile.sasiadow=1)
```

```
##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine3, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 1)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2915
## Standard deviation: 0.0059
```

```
errorest(etykietki ~., wine3, model=my.ipredknn, predict=my.predict, estimator="boot",
         est.para=control.errorest(nboot = 50), ile.sasiadow=20)
```

```
##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine3, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 20)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2273
## Standard deviation: 0.0039
```

```
errorest(etykietki ~., wine3, model=my.ipredknn, predict=my.predict, estimator="boot",
         est.para=control.errorest(nboot = 50), ile.sasiadow=100)
```

```
##
## Call:
## errorest.data.frame(formula = etykietki ~ ., data = wine3, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 100)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.3944
## Standard deviation: 0.0131
```

Wnioski:

- Wyjściowy zbiór: Dla k=1 błąd na poziomie 5%, dla k=20 na poziomie 4%, dla k=100 aż 15%.
- Pozdbiór z cechami o najlepszych zdolnościach dyskryminacyjnych: Błąd dla k=1 bliski 30%, dla k=20 około 22%, k=100 około 40%. Tutaj różnica jest wyraźna.
- Zwiększenie k powoduje zmniejszenie błędu, ale tylko do pewnego momentu. Zatem istnieje takie k, dla którego analiza jest optymalna. W dalszej analizie przyjmujemy k=20 jako optymalne.
- Dla k=1 błąd jest niewielki, jednak tutaj mamy dużą losowość.

Dowolne podzbiory danych dla takiego samego k=20:

```
##
## Call:
## errorest.data.frame(formula = etykietki ~ Ash + Alcalinity +
##     Proline + Hue, data = wine2, model = my.ipredknn, predict = my.predict,
##     estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 20)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.1198
## Standard deviation: 0.0038
##
## Call:
## errorest.data.frame(formula = etykietki ~ Ash + Hue + Flavanoids +
##     Proline, data = wine2, model = my.ipredknn, predict = my.predict,
##     estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 20)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.0611
## Standard deviation: 0.0025
```

Obserwacje:

- Po zamianie jednej z cech zbioru na cechę mającą wysokie zdolności dyskryminacyjne błąd klasyfikacji maleje o połowę.

Sprawdzimy, czy dodanie kolejnej cechy o wysokiej zdolności dyskryminacyjnej znowu zmniejszy błąd.

```
##
## Call:
## errorest.data.frame(formula = etykietki ~ Ash + Hue + Flavanoids +
##     Proline + Phenols, data = wine2, model = my.ipredknn, predict = my.predict,
##     estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 20)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.0731
## Standard deviation: 0.0029
##
## Call:
## errorest.data.frame(formula = etykietki ~ Phenols + Hue + Flavanoids +
##     Ash, data = wine2, model = my.ipredknn, predict = my.predict,
##     estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 20)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.185
## Standard deviation: 0.0043
```

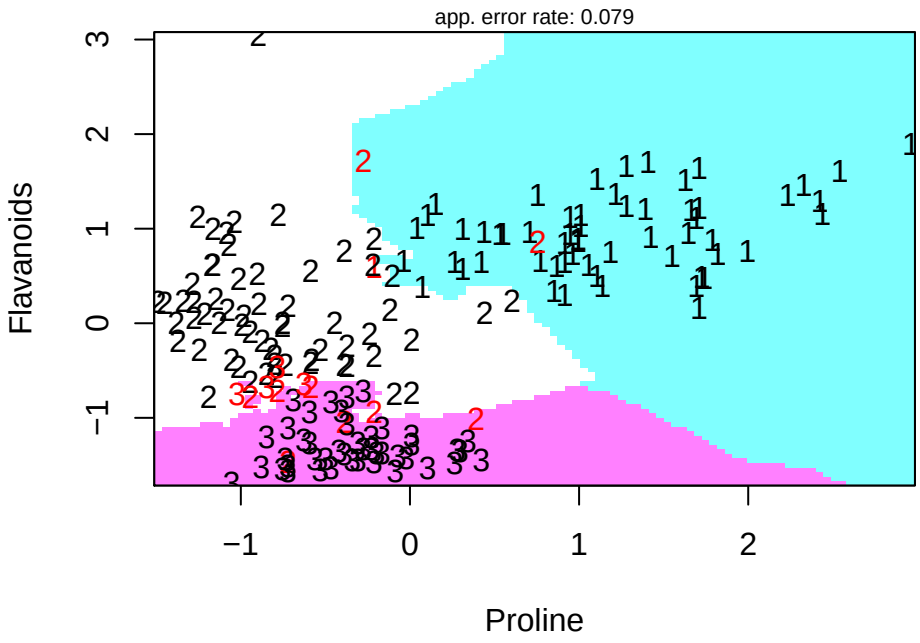
Obserwacje:

- Dodanie kolejnej cechy o wysokiej zdolności dyskryminacyjnej nie zmniejsza błędu.
- Usunięcie zmiennej Proline znacznie zwiększa błąd.

Wnioski:

- Model z cechami o wysokiej zdolności dyskryminacyjnej nie musi minimalizować błędu klasyfikacyjnego.
- Istotny wpływ zmiennej Proline na poziom błędu klasyfikacyjnego (dlaczego?).

Wykres zależności prognozowanych i rzeczywistych etykiet (tu zmienne Flavanoids i Proline)



Rysunek 9: Prognozy

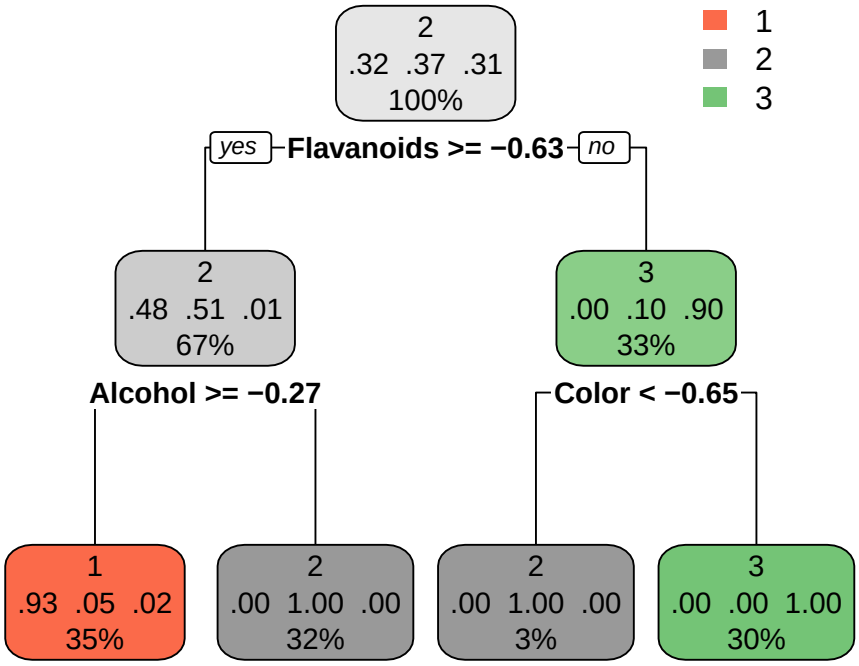
Kolorowe obszary oznaczają prognozowane etykiety. Na niebiesko - (1), fioletowo - (2), biało - (3)

3.4 Metoda drzew klasyfikacyjnych

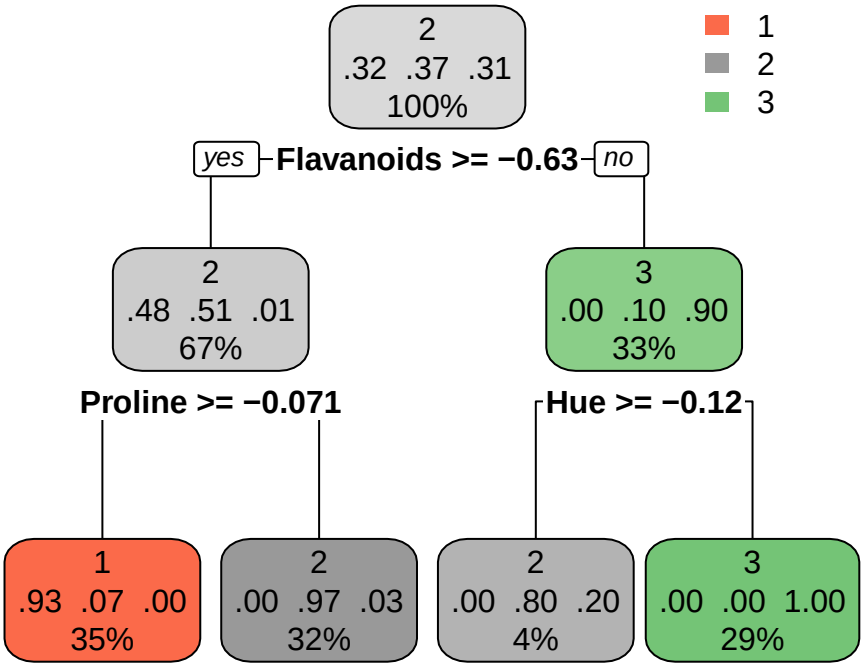
Wyznaczenie drzewa i klasyfikatora:
Stworzymy dwa modele. Jeden ze wszystkimi danymi w zbiorze, a drugi z wybranym podzbiorem. Model pierwszy zastosujemy do analizy danych wyjściowych oraz do podzbioru danych z cechami o najlepszej zdolności dyskryminacyjnej. Natomiast model drugi, zbudowanych na wybranych cechach, przeanalizujemy na danych wyjściowych i porównanym, jakie znaczenie ma ilość zmiennych przy tworzeniu modelu klasyfikatora metodą drzew.

```
library(MASS)
library(rpart)
library(rpart.plot)
library(klaR)
attach(wine2)
set.seed(1) # ustalamy ziarno generatora aby otrzymać powtarzalne wyniki
n.learn.all <- nrow(learn.set.all)
n.learn.subset<-nrow(learn.set.subset)

n.test.all <- nrow(test.set.all)
n.test.subset <- nrow(test.set.subset)
model<- etykietki~ .   #model klasyczny
#model 2
```

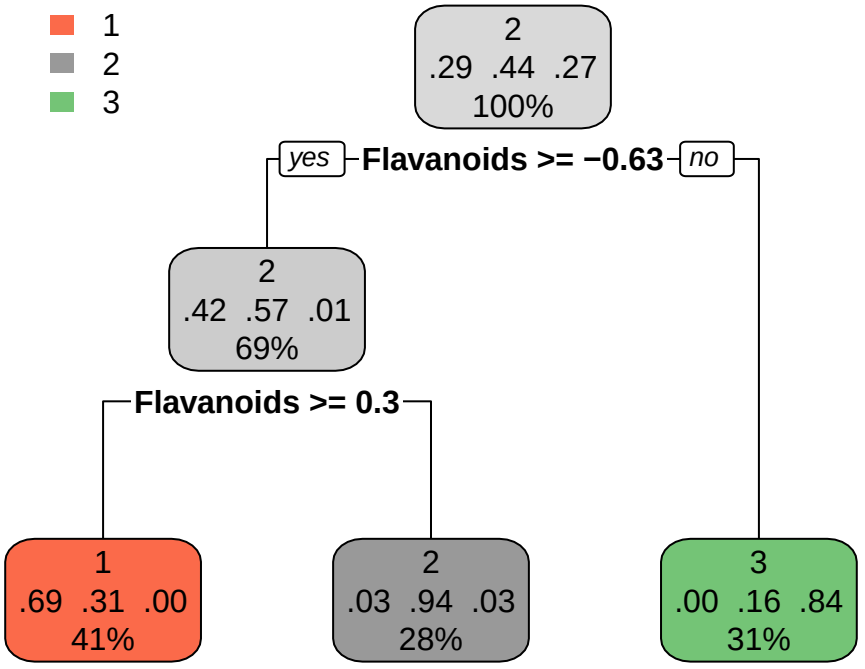
```
model.all.1 <- etykietki ~ Proline + Phenols + Hue + Flavanoids


# budujemy drzewo
tree.all <- rpart(model, data=learn.set.all, control=rpart.control(cp=.03, minsplit=10,
tree.all.1 <- rpart(model.all.1, data=learn.set.all, control=rpart.control(cp=.03, minsp
tree.subset <- rpart(model, data=learn.set.subset, control=rpart.control(cp=.03, minspli

rpart.plot(tree.all)
```




```
rpart.plot(tree.all.1)
```




```
rpart.plot(tree.subset)
```



Wnioski:

- Największe znaczenie w klasyfikowaniu ma zmienna Proline, kolejnymi zmiennymi są Duration i Phenols, jeśli rozważamy zbiór danych wyjściowych, podobne zjawisko występuje w modelu drugim.

- Proline i Phenols mają zależność decydującą na poziomie zera, pamiętay że dane uległy standaryzacji, wię możemy uznać, że zmienna Proline dzieli drzewko na dwie ilościwo podobne zbiory
- Analizując drzewo z danymi o najlepszej zdolności dyskryminacyjnej, widzimy że klasa 3 jest mocno oddzielona od klas 1 oraz 2

Macierze pomyłek:

Analiza dla danych wyjściowych:

```
# macierz pomyłek dla zbioru uczącego
conf.mat.learn.all <- table(prog.learn.all, etykietki.learn.all)
conf.mat.learn.all

##               etykietki.learn.all
## prog.learn.all  1  2  3
##              1 38  2  1
##              2  0 42  0
##              3  0  0 35

(error.learn.all <- (n.learn.all - sum(diag(conf.mat.learn.all))) / n.learn.all)

## [1] 0.02542373

# macierz pomyłek dla zbioru testowego
conf.mat.test.all <- table(prog.test.all, etykietki.test.all)
conf.mat.test.all

##              etykietki.test.all
## prog.test.all  1  2  3
##             1 21  6  0
##             2  0 20  0
##             3  0  1 12

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

## [1] 0.1166667
```

Analiza dla danych o największych zdolnościach dyskryminacyjnych:

```
# macierz pomyłek dla zbioru uczącego
conf.mat.learn.subset<-table(prog.learn.subset, etykietki.learn.subset)
conf.mat.learn.subset

##                  etykietki.learn.subset
## prog.learn.subset  1  2  3
##                 1 33 15  0
##                 2  1 31  1
##                 3  0  6 31
```

```
(error.learn.subset <- (n.learn.subset - sum(diag(conf.mat.learn.subset))) / n.learn.sub

## [1] 0.1949153

# macierz pomyłek dla zbioru testowego
conf.mat.test.subset<-table(prog.test.subset, etykietki.test.subset)
conf.mat.test.subset

##                  etykietki.test.subset
## prog.test.subset  1  2  3
##                1 25  4  0
##                2  0 11  0
##                3  0  4 16

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

## [1] 0.1333333
```

Analiza danych wyjściowych w drugim modelu:

```
#macierz pomyłek dla modelu 2
#Zbiót uczący
conf.mat.all.1.learn<-table(prog.all.1.learn, etykietki.learn.all)
conf.mat.all.1.learn

##                  etykietki.learn.all
## prog.all.1.learn  1  2  3
##                1 38  3  0
##                2  0 41  2
##                3  0  0 34

(error.learn.1.all <- (n.learn.all - sum(diag(conf.mat.all.1.learn))) / n.learn.all)

## [1] 0.04237288

#Zbiór testowy
conf.mat.all.1.test<-table(prog.all.1.test, etykietki.test.all)
conf.mat.all.1.test

##                 etykietki.test.all
## prog.all.1.test  1  2  3
##               1 20  1  0
##               2  1 24  0
##               3  0  2 12

(error.rate.1.test <- (n.test.all - sum(diag(conf.mat.all.1.test))) / n.test.all)

## [1] 0.06666667
```

Wnioski:

- Stosunkowa wysoka różnica pomiędzy zbiorem testowym, a uczącym w danych wyjściowych w obu modelach.
- Model 2 zastosowany na mniejszej ilości danych niewiele różni się w stosunku do modelu na wszystkich zmiennych, wnioskujemy że głównymi zmiennymi, które w znacznym stopniu klasyfikują drzewo to Proline oraz Phenols
- Tak jak w innych klasyfikatorach istnieje spora niedokładność dla zbioru o najlpeszych cechach dyskryminacyjnych, dla obu zbiorów (uczący i testowy) wynosi ona około 18%

3.5 Podsumowanie i wnioski

Podsumowanie:

- Naiwny Bayes:
  -Analiza całego zbioru: Błąd na poziomie 2%
  -Analiza podzbioru cech o najlepszych zdolnościach dyskryminacyjnych: Błąd na poziomie 20%
- k-NN:
  -Analiza całego zbioru: Błąd poniżej 2%
  -Analiza podzbioru cech o najlepszych zdolnościach dyskryminacyjnych: Błąd na poziomie 18-23%
- Drzewa klasyfikacyjne:
  -Analiza całego zbioru: Błąd na poziomie 7-10%
  -Analiza podzbioru cech o najlepszych zdolnościach dyskryminacyjnych: Błąd na poziomie 14-22%

Wnioski:

- Metoda naiwnego Bayesa oraz k-NN daje bardzo zbliżone, niemal identyczne rezultaty.
- Metoda drzew klasyfikacyjnych gorzej radzi sobie gdy analizujemy cały zbiór, jednak daje szansę na zmniejszenie błędu, gdy ograniczymy się do podzbioru cech o najlepszych zdolnościach dyskryminacyjnych.