Raport 3
Klasyfikacje i regresja liniowa

Jan Solarz 243889
Szymon Suszek 237288

18 maja 2020

Spis treści

1 Zadanie 1. Model regresji liniowej i klasyfikacja **1**
1.1 Model regresji liniowej . . . 2
1.2 Oceny i prognozy . . . 4
1.3 Zbiór testowy 1 . . . 5
1.4 Poszerzenie regresji o wielomianowe składniki . . . 6
1.4.1 Zbiór testowy 2 . . . 7
1.5 Wnioski do regresji liniowej . . . 8

2 Zadanie 2. Porównanie metod klasyfikacji **8**
2.1 Pierwszy kontakt z danymi . . . 9
2.2 Przedstawienie algorytmów klasyfikacji . . . 12
2.2.1 Podział danych . . . 12
2.2.2 Algorytm- drzewa klasyfikacyjne . . . 12
2.2.3 Algorytm- metoda k-najbliższych sąsiadów . . . 16
2.2.4 Klasyfikacja z wykorzystaniem naiwnego klasyfikatora bayesowskiego . . . 19
2.3 Porównania i wnioski . . . 22

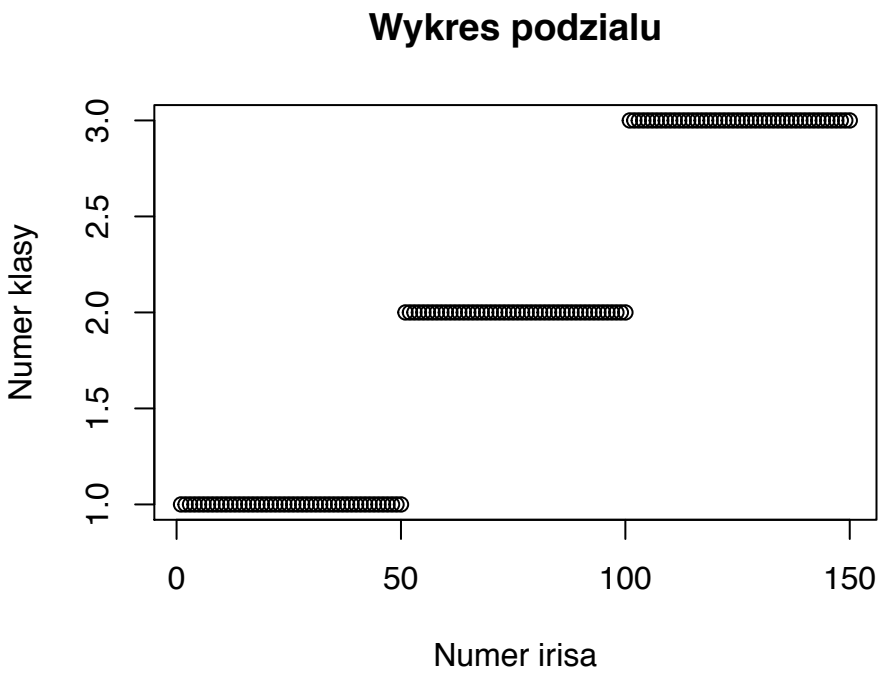
1 Zadanie 1. Model regresji liniowej i klasyfikacja

Klasyfikacje przeprowadzać będziemy na zbiorze *iris* składającym się ze 150 danych podzielonych na 3 klasy (zmienna *Spacies*), charakteryzującym się cechami *Sepal Length*, *Sepal Width*, *Petal Length*, *Petal Width*

Zaczynamy od podzielenia danych na zbiór uczący i testowy zawierjących odpowiednio 60 i 40 procent

```
data(iris)

etykiety_klas <- iris$Species
n <- length(etykiety_klas)
K <- length(levels(etykiety_klas))
n #liczba przypadków
```

```
## [1] 150

K #liczba klas

## [1] 3

set.seed(123)
library(caTools)
#dzielimy dane na dwa zbiory
sample <- sample.split(iris, SplitRatio = 0.75)
#zbiór uczący <treningowy>
train <- subset(iris,sample == TRUE)

#zbiór testowy
test = subset(iris, sample == FALSE)
```



Obiekty są uporządkowane ze względu na przynależność do poszczególnych klas (gatunków).

1.1 Model regresji liniowej

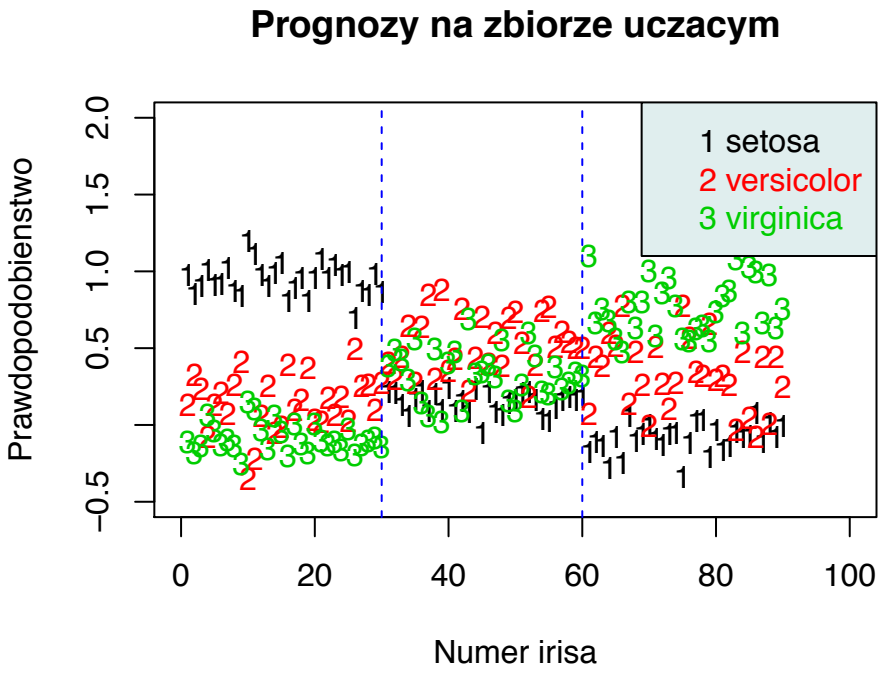
```
# MACIERZ EKSPERYMENTU
# liczba obiektów w zbiorze uczącym
n_train <- nrow(train)
# macierz obiektów z zbioru uczącego z dodana kolumna jedynkowa
X_train <- cbind(rep(1, n_train), train[,1:4])
# konwersja do formatu macierzy
X_train <- as.matrix(X_train)
```

```r
# inicjalizacja macierzy etykiet
Y_train <- matrix(0, nrow= n_train, ncol = K)
etykiety_train_num <- as.numeric(train$Species)

for(k in 1:K)
  Y_train[etykiety_train_num == k, k] <- 1

Y <- Y_train
X <- X_train

#Macierz estymowanych współczynników - wykorzystana metoda najmniejszych kwadratów
B <- solve(t(X)%*%X)%*%t(X)%*%Y

# Wartości prognozowane (prognozowane prawdopodobieństwa przynależności do poszczególn
Y_hat <- X%*%B
abb<-head(Y_hat)
colnames(abb)<-c("Klasa 1. <setosa>","Klasa 2. <versicolor>","Klasa 3. <virginica>")
# sprawdzenie czy prognozowane prawdopodobienstwa sumuja sie do 1
head(rowSums(Y_hat))

## 1 2 3 6 7 8
## 1 1 1 1 1 1

tail(rowSums(Y_hat))

## 141 142 143 146 147 148
##   1   1   1   1   1   1
```

	Klasa 1. <setosa>	Klasa 2. <versicolor>	Klasa 3. <virginica>
1	0.977	0.133	-0.110
2	0.854	0.327	-0.181
3	0.904	0.234	-0.137
6	1.010	-0.074	0.064
7	0.910	0.131	-0.041
8	0.920	0.212	-0.133

Tabela 1: Zestawienie prognozowanych prawdopodobieństw

- Budujemy model na zbiorze uczącym, następnie sprawdzamy jego skuteczność na zbiorze testowym
- Budując macierz estymowanym współczynników korzystamy z metody najmniejszych kwadratów
- Prognozowane prawdopodobieństwa sumują się do "jedynki" w każdym przypadku



Rysunek 1: Prognoza na zbiorze uczacym

- Widzimy bardzo dobrą separacje pierwszej klasy w pierwszym podzbiorze- jej skuecznoś spada w drugiej i trzeciej klasie (odpowiednio w kolejnych podzbiorach)
- Klasy "versicolor" i "virginica" mieszają się ze sobą, "setosa" bardziej sie izolouje

1.2 Oceny i prognozy

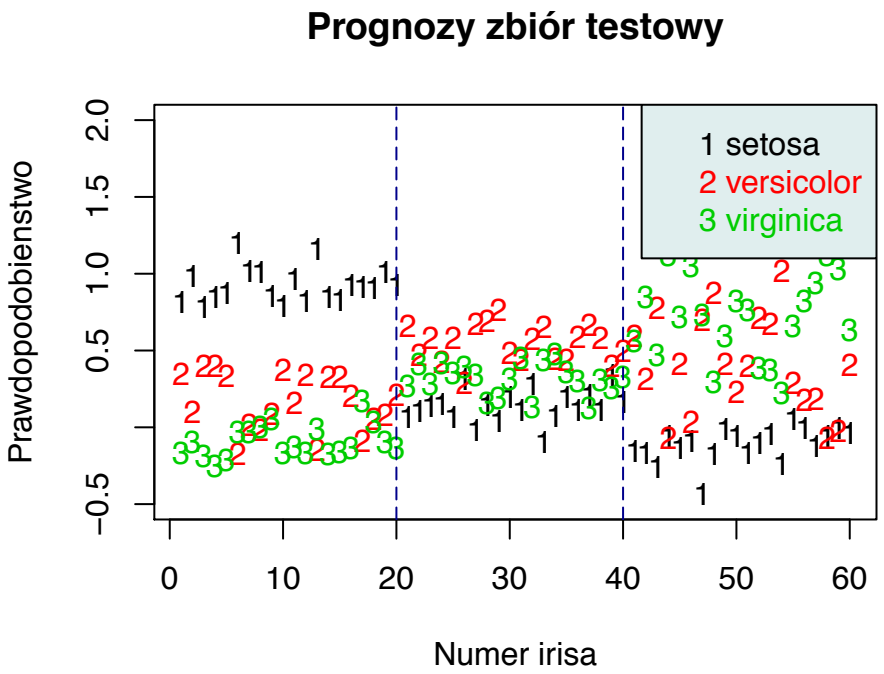
```
# OCENA DOKŁADNOŚCI
klasy <- levels(train$Species)
maks_ind <- apply(Y_hat, 1, FUN=function(x) which.max(x))

prognozowane_etykiety <- klasy[maks_ind]
rzeczywiste_etykiety <- train$Species

macierz_pomylek <- table(rzeczywiste_etykiety, prognozowane_etykiety)



# dokładność klasyfikacji
sum(diag(macierz_pomylek))/nrow(train)

## [1] 0.8555556

1-sum(diag(macierz_pomylek))/nrow(train)

## [1] 0.1444444
```

	setosa	versicolor	virginica
setosa	30	0	0
versicolor	0	21	9
virginica	0	4	26

Tabela 2: Macierz pomyłek dla zbioru uczącego

1.3 Zbiór testowy 1

```
##          [,1]       [,2]        [,3]
## 4   0.8197133  0.3499175 -0.16963081
## 5   0.9888574  0.1040344 -0.09289178
## 9   0.7858681  0.4001417 -0.18600980
## 10  0.8473104  0.4025409 -0.24985130
## 14  0.8773758  0.3369736 -0.21434947
## 15  1.2014242 -0.1720381 -0.02938602
```




```
##                            prognozowane_etykiety_test
## rzeczywiste_etykiety_test setosa versicolor virginica
##                 setosa         20          0         0
##                 versicolor      0         17         3
##                 virginica       0          6        14
## [1] 0.85
## [1] 0.15
```

- "Macierz pomyłek" dokładnie pokazuje ile irysów zostało źle przyporządkowanych. (te które nie leżą na diagonali).

- Klasa "setosa" została w stu procentach dobrze odseparowana/sklasyfikowana. 4 irisy "virginica" zostały rozpoznane jako "versicolor", a aż 10 "versicolor" jako "virginica".
- Dokładnośc klasyfikacji zbioru uczącego (90 irysów) wynosi 85.5 procenta, błąd więc na poziomie 14.5 procenta.
- Dokładnośc klasyfikacji zbioru testowego (60 irysów) wynosi 85 procenta, błąd więc na poziomie 15 procenta.

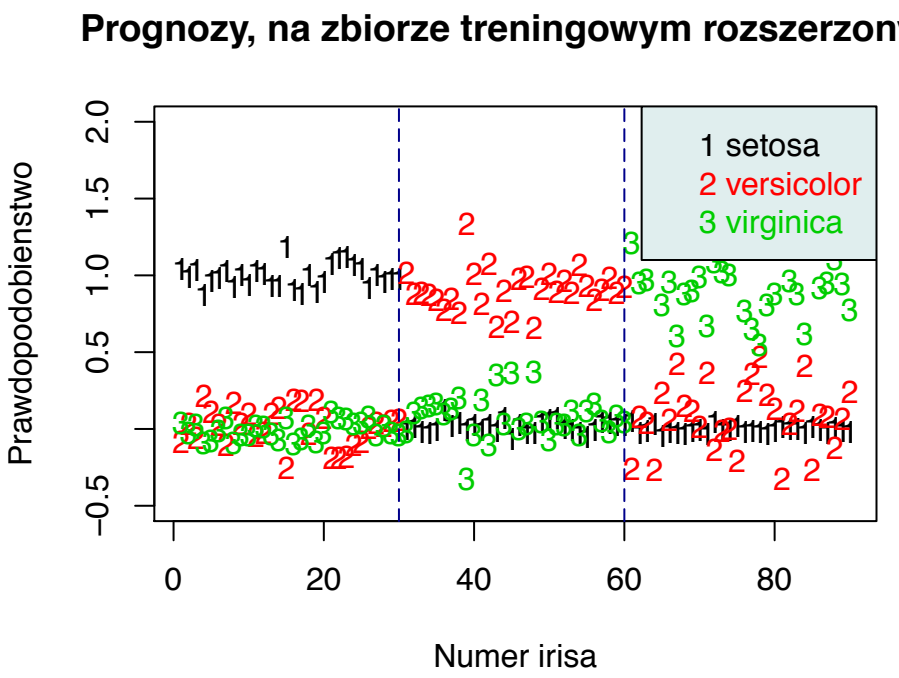
1.4 Poszerzenie regresji o wielomianowe składniki

Wzbogacamy dane o wielomianowe składniki stopnia drugiego. Przeprowadzamy taką samą klasyfikacje jak dla normalnego zbioru.

```
# ROZSZERZENIE MODELU REGRESJI LINIOWEJ
X_extended_train <- X_train
for(i in 1:4)
  X_extended_train <- cbind(X_extended_train,train[,i]*train[,i])
X_extended_train <-cbind(X_extended_train,train[,1]*train[,2], train[,1]*train[,3],train

# konwersja do formatu macierzy
X_extended_train <- as.matrix(X_extended_train)
colnames(X_extended_train)<-c("jedynki","SP", "SW", "PL",  "PW", "SL2", "SW2","PL2","PW2
Y <- Y_train
X <- X_extended_train
#Macierz estymowanych współczynników - wykorzystana metoda najmniejszych kwadratów
B <- solve(t(X)%*%X)%*%t(X)%*%Y

# prognozowanie na zbiorze treningowym
Y_hat <- X%*%B
```

	jedynki	SP	SW	PL	PW	SL2	SW2	PL2	PW2	SL*SW	SL*PL	SL*PW
1	1.000	5.100	3.500	1.400	0.200	26.010	12.250	1.960	0.040	17.850	7.140	1.020
2	1.000	4.900	3.000	1.400	0.200	24.010	9.000	1.960	0.040	14.700	6.860	0.980
3	1.000	4.700	3.200	1.300	0.200	22.090	10.240	1.690	0.040	15.040	6.110	0.940
6	1.000	5.400	3.900	1.700	0.400	29.160	15.210	2.890	0.160	21.060	9.180	2.160
7	1.000	4.600	3.400	1.400	0.300	21.160	11.560	1.960	0.090	15.640	6.440	1.380
8	1.000	5.000	3.400	1.500	0.200	25.000	11.560	2.250	0.040	17.000	7.500	1.000

Tabela 3: Poczatek zbioru danych z poszerzonymi cechami



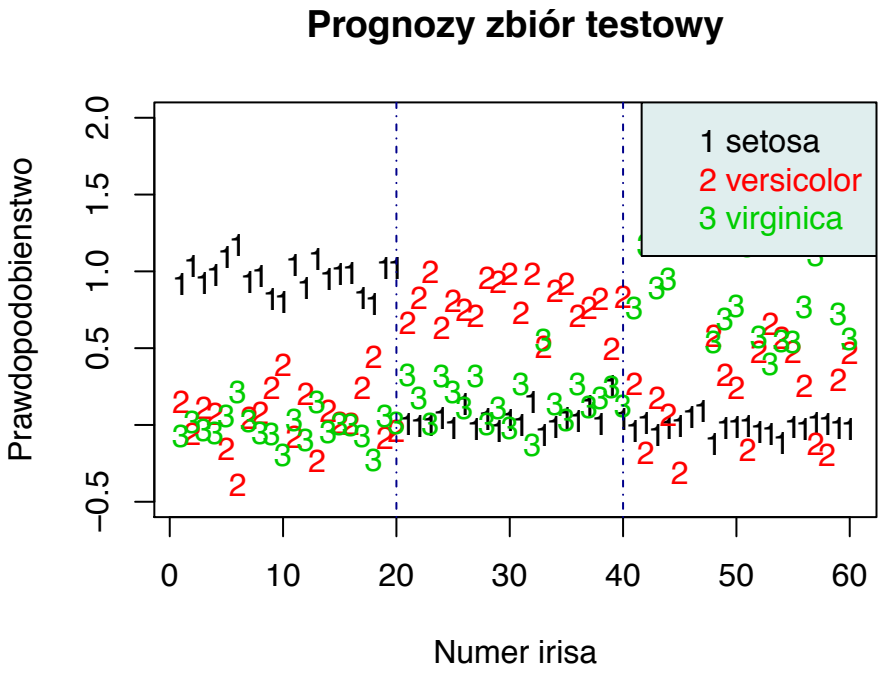
Rysunek 2: Prognoza na zbiorze uczacym

```
##                     prognozowane_etykiety
## rzeczywiste_etykiety setosa versicolor virginica
##           setosa         30          0         0
##           versicolor      0         30         0
##           virginica       0          0        30
## [1] 1
## Dokładność klasyfikacji w tym modelu wynosi 100 procent
```

- Widzimy duzo lepszą separacje klas, nie tylko w przypadku klasy pierwszej (setosa).
- Dokładność klasyfikacji w tym przypadku wynosi aż 100 procent, pokazuje nam to również "Macierz pomyłek", na zbiorze testowym 93.3 procent.
- Oczywiście prognozowane prawdopodieństa sumują się również do jedynek.

1.4.1 Zbiór testowy 2

```
##                          prognozowane_etykiety_test
## rzeczywiste_etykiety_test setosa versicolor virginica
##                 setosa         20          0         0
##                 versicolor      0         19         1
##                 virginica       0          3        17
## [1] 0.9333333
## Dokładność klasyfikacji w tym modelu wynosi 93.33333 procent
```



Rysunek 3: Prognozy zbioru testowego/model rozsz

1.5 Wnioski do regresji liniowej

- Dokładność klasyfikacji jest wyższa w zbiorze uczącym w obu przypadkach, nie jest jednak to duża różnica. Wynika to oczywiście z tego, że zbiór testowy jest zbiorem nowym, niezależnym na którym algorytm nie był budowany
- Zdolność do prawidłowej klasyfikacji zdecydowanie posiada zbiór wzbogacony o składniki wielomianowe, maskowanie klas jest zredukowane wtedy prawie do minimum.
- Wprowadzenie składników wielomianowych opartych na podstaowych cechach jest bardzo przydatne przy budowaniu modelu regresji liniowej- przekłada się to na lepszą dokładność.
- W każdym przypadku prognozowane wartości sumują się do jedynek.
- **Maskowanie** Generalny problem maskowania klas nie zachodzi, w każdym przypadku wyróżnione są wszystkie klasy z podobnymi ilościami kwiatów. Zdecydowanie lepiej odseparowana jest pierwsza klasa od pozostałych. Widoczne jest jednak zjawisko nachodzenia na siebie przypadków "virginica" oraz "versicolor"

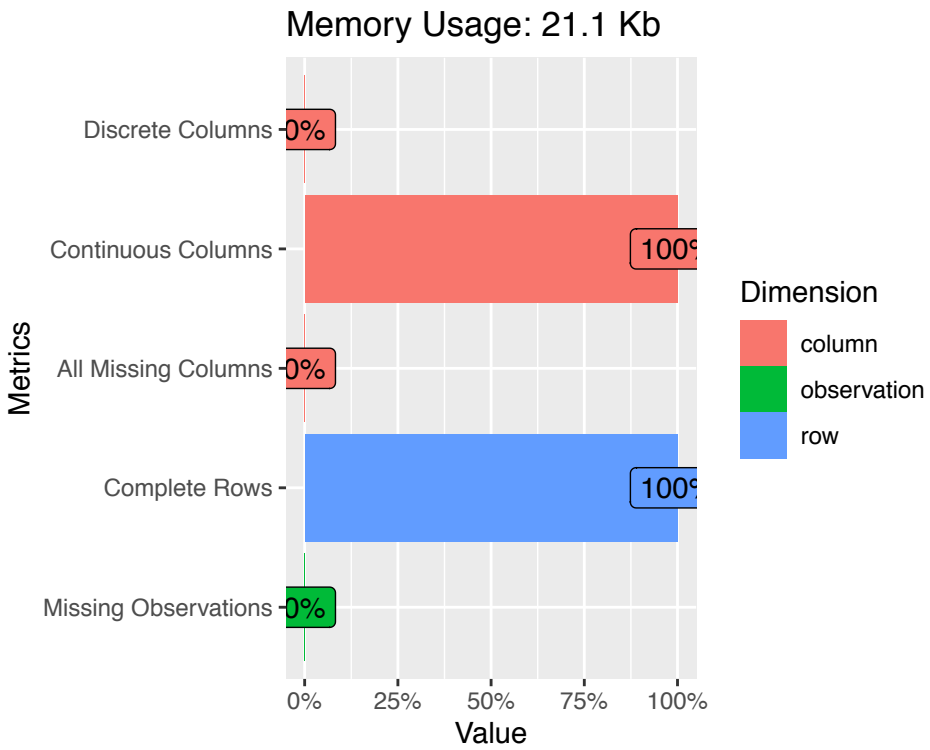
2 Zadanie 2. Porównanie metod klasyfikacji

Celem analizy w tym zadaniu jest zastosowanie poznanych algorytmów klasyfikacji i szczegółowe porównanie ich dokładności. Będziemy opierać się na metodzie k- najbliższych sąsiadów (*k-nearest k-Neighbors*), drzewach klasykacyjnych (*classification Trees*) oraz naiwnym klasyfikatorze Bayesowskim (*naive Bayes classifier*). Będziemy starać się przyjrzeć różnym wskaźnikom oraz pokazać skuteczność danych algorytmów.

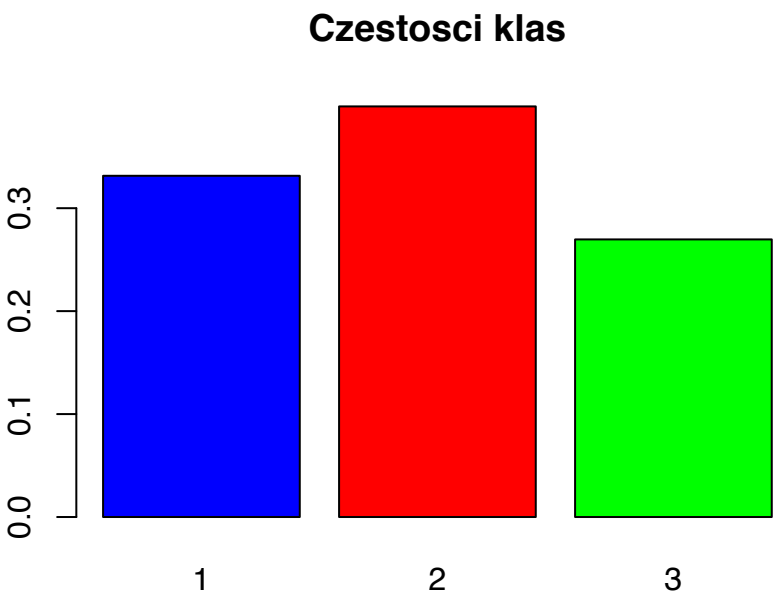
2.1 Pierwszy kontakt z danymi

```
library(HDclassif)

## Loading required package: MASS

library(MASS)
library(DataExplorer)

data(wine)

colnames(wine) <- c("Class","Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium"

","Proline")

attach(wine)
```

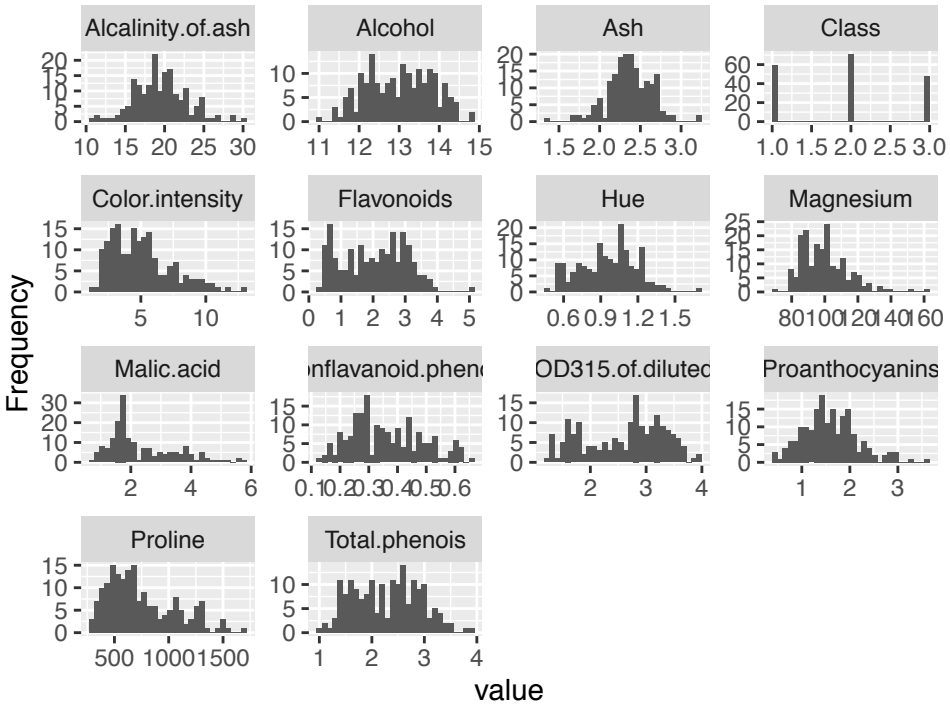
```
## [1] 178  14
```



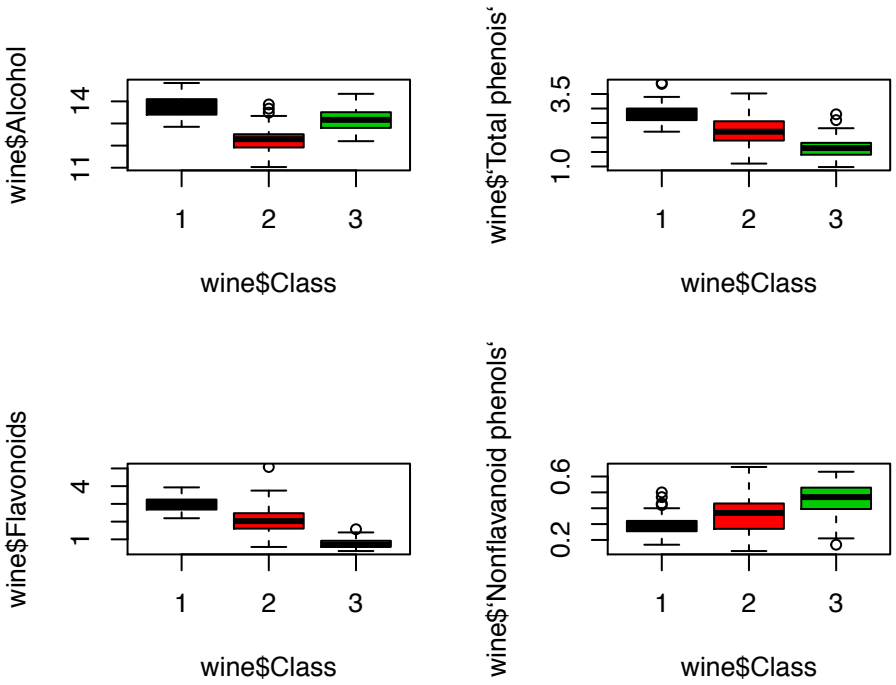
Rysunek 4: Wykresy do wprowadzenia



Rysunek 5: Wykresy do wprowadzenia



Rysunek 6: Wykresy do wprowadzenia



Rysunek 7: Wykresy do wprowadzenia

- Do zadania wykorzystujemy zbiór Wine, składający się ze 178 przypadków win, które opisują 14 zmiennych. Wszystkie zmienne są ilościowe
- Brak nietypowych danych, nie występują braki
- Zmienną klasyfikującą jest zmienna "Class", która określa dana klasę winą. Mamy 3 klasy.
- Klasy ułożone są rosnąco, ich liczby są zbliżone do siebie, jednak najwięcej jest drugiej klasy, najmniej trzeciej.
- Po wstępnej analizie wykresów najlepszą dyskretyzacją charakteryzują się zmienne Alcohol, Flavonoids, Total.phenois oraz Nonflavonoid.phenols.

	Class	Alcohol	Total phenois	Flavonoids	Nonflavanoid phenols
1	1	14.230	2.800	3.060	0.280
2	1	13.200	2.650	2.760	0.260
3	1	13.160	2.800	3.240	0.300
4	1	14.370	3.850	3.490	0.240
5	1	13.240	2.800	2.690	0.390
6	1	14.200	3.270	3.390	0.340

Tabela 4: Zredukowane dane

2.2 Przedstawienie algorytmów klasyfikacji

2.2.1 Podział danych

W analizie posłużymy się znów podziałem danych na zbiór uczący i testowy Zbiory te przyporządkowują sobie losowo dane ze wbioru wine. Modele klasyfikatorów będziemy budować na zbiorze uczącym, testy wykonujemy na niezależnych zbiorach testowych. Dzięki temu spodziewamy się bardziej wiarygodnej dokładości.

```
# losujemy obiekty do zbioru uczącego i testowego
wine$Class<-factor(wine$Class)
set.seed(1)
n<-nrow(wine)
learning.set.index <- sample(1:n,2/3*n)

learning.set <- wine[learning.set.index,]
nl<-nrow(learning.set)
test.set     <- wine[-learning.set.index,]
nt<-nrow(test.set)
```

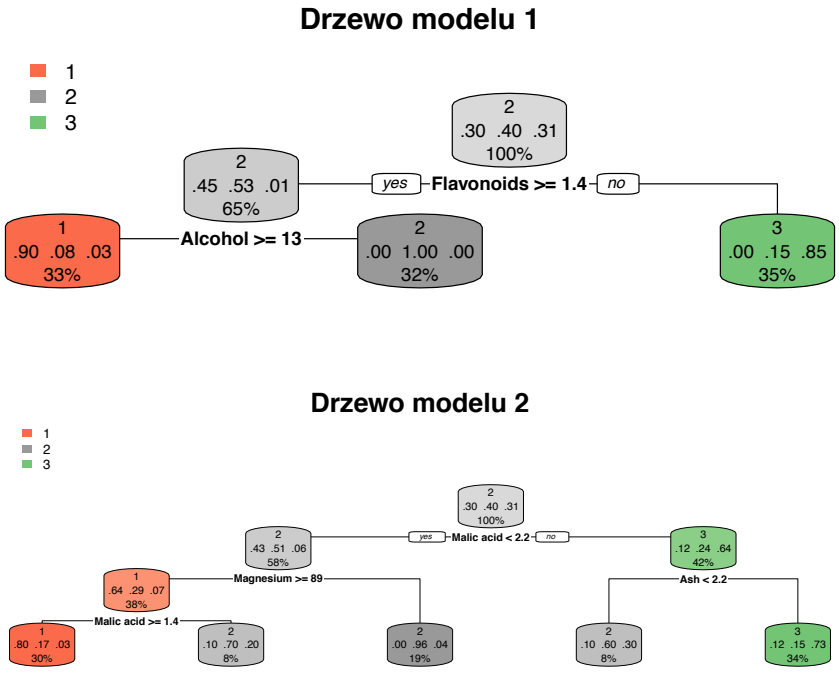
2.2.2 Algorytm- drzewa klasyfikacyjne

Zaczynamy prace z algorytmem od stworzenia dwóch modeli porównawczych. Jeden z nich składa się z cech o lepszej dyskretyzacji, drugi zawiera pozostałe cechy. *Model9* jest modelem porównywanym w kazdym algorytmie

```
library(MASS)
library(rpart)
library(rpart.plot)
library(rpart)


#Tworzymy modele porównawcze

model1 <- Class ~Alcohol+Flavonoids
model2<- Class ~Magnesium + Ash+ `Malic acid`
model9<- Class~Alcohol+Ash+Hue+Flavonoids

#tworzymy drzewo na danych uczących
wine.tree.simple1 <- rpart(model1, data=learning.set)
wine.tree.simple2 <- rpart(model2, data=learning.set)
wine.tree.simple9 <- rpart(model9, data=learning.set)
```

Korzystamy z modelów doboru cech o różnej dyskretyzacji.

```
par(mfrow=c(2,1))
rpart.plot(wine.tree.simple1, main="Drzewo modelu 1")
rpart.plot(wine.tree.simple2, main="Drzewo modelu 2")
```



Rysunek 8: Drzewa klasyfikacyjne dwóch modeli

```
# Ocena dokladnosci klasyfikacji

# macierz pomylek (confusion matrix)
conf.mat.test1 <- table(pred.labels.test1, test.set$Class)
conf.mat.test2 <- table(pred.labels.test2, test.set$Class)
conf.mat.test9 <- table(pred.labels.test9, test.set$Class)

conf.mat.learning1 <- table(pred.labels.learning1, learning.set$Class)
conf.mat.learning2 <- table(pred.labels.learning2, learning.set$Class)
conf.mat.learning9 <- table(pred.labels.learning9, learning.set$Class)

# blad klasyfikacji (na zbiorze uczacym i testowy)
#model 1
(error.rate.learning1 <- (nl - sum(diag(conf.mat.learning1))) / nl)

## [1] 0.08474576

(error.rate.test1 <- (nt - sum(diag(conf.mat.test1))) / nt)

## [1] 0.1666667

#model 2
(error.rate.learning2 <- (nl - sum(diag(conf.mat.learning2))) / nl)

## [1] 0.220339
```

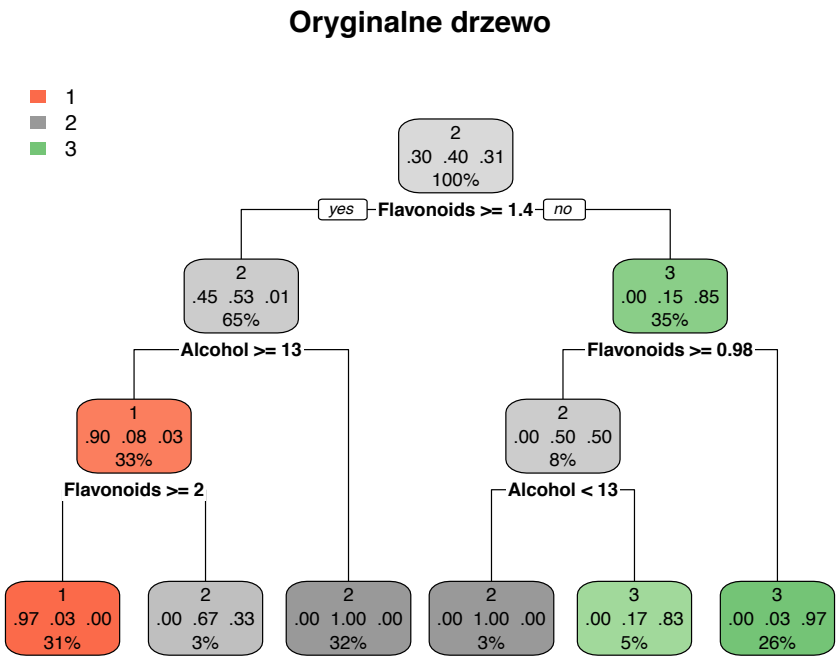
```
(error.rate.test2 <- (nt - sum(diag(conf.mat.test2))) / nt)

## [1] 0.2666667

#model 9
(error.rate.learning9 <- (nl - sum(diag(conf.mat.learning9))) / nl)

## [1] 0.05932203

(error.rate.test9 <- (nt - sum(diag(conf.mat.test9))) / nt)

## [1] 0.08333333
```

- Błąd klasyfikacji modelu 1 wynosi : 8.5 procent dla zbioru uczącego, 17 procent dla niezależneo zbioru testowego.
- Błąd klasyfikacji modelu 1 wynosi : 22 procent dla zbioru uczącego, 27 procent dla niezależneo zbioru testowego.
- Na drzewach widzimy podobne procentowe przyporządkowanie do klas, większa złożoność drewa modelu 2, ze względu o jedną zmienną więcej
- Mimo wybrania o jedną ceche porównawczą mniej, model 1 daje nam lepszą skuteczność
- Model9 daje nam błedy rzędu: 6 i 8.3 procent

Wybór złożoności modelu

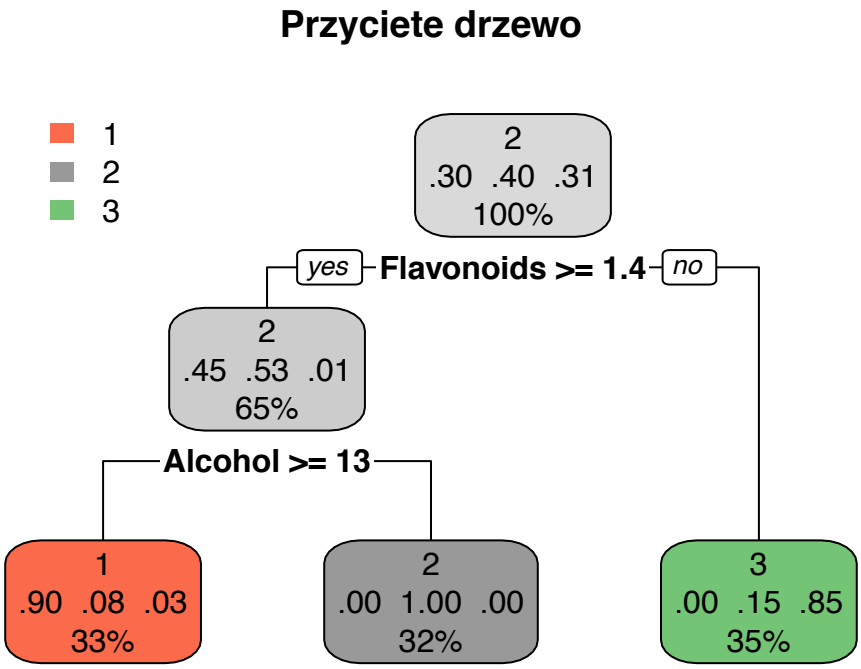
Reguła 1SE Przycinanie drzew na podstawie kryterium kosztu-złożoności Możemy teraz skonstruować prognozy i porównać skuteczność obu modeli (przed i po przycięciu)

```
cp.opt <- 0.1 #uwaga wartość wpisana na potrzeby ilustracji

#
wine.tree.complex.pruned <- prune(wine.tree.complex, cp = cp.opt)

par(mfrow=c(1,1))
rpart.plot(wine.tree.complex, main="Oryginalne drzewo")
```



Rysunek 9: Wykres drzew z przycieciem

```
rpart.plot(wine.tree.complex.pruned, main="Przyciete drzewo")
```



Rysunek 10: Wykres drzew z przycieciem

```
# blad klasyfikacji (na zbiorze uczacym i testowy)
#model oryginalny
(error.rate.learning11 <- (nl - sum(diag(conf.mat.learning11))) / nl)

## [1] 0.03389831

(error.rate.test11 <- (nt - sum(diag(conf.mat.test11))) / nt)

## [1] 0.08333333

#model po przycięciu
(error.rate.learning12 <- (nl - sum(diag(conf.mat.learning12))) / nl)

## [1] 0.08474576

(error.rate.test12 <- (nt - sum(diag(conf.mat.test12))) / nt)

## [1] 0.1666667
```

- Dokonaliśmy znaczącej redukcji rozgałęzień drzewa
- Prognozowany błąd w oryginalnym drzewie wynosi 3.4 i 8.4 procenta (kolejno zbiór uczący i testowy), a w drzewie po redukcji (usuneliśmy kryterium "Alcohol¡13") odpowiednio 8.5 i 16.7 procenta.

2.2.3 Algorytm- metoda k-najbliższych sąsiadów

Klasyfikacja obiektów na przykladzie metody k-nn z pakietu class (wbudowana funkcja knn)

```
library(class)

# rzeczywiste klasy win
etykietki.rzecz <- test.set$Class

# teraz robimy prognozę
etykietki.prog <- knn(learning.set[,-1], test.set[,-1], learning.set$Class, k=1)

# macierz pomyłeek (ang. confusion matrix)
(wynik.tablica <- table(etykietki.prog,etykietki.rzecz))

##                etykietki.rzecz
## etykietki.prog  1  2  3
##              1 23  1  0
##              2  0 18  3
##              3  1  5  9

# błąd klasyfikacji
n.test <- dim(test.set)[1]
(n.test - sum(diag(wynik.tablica))) / n.test
```

```
## [1] 0.1666667
```

Idea budowy modeli klasyfikacyjnych w R na przykladzie metody k-nn z pakietu ipred (funkcja ipredknn(). Tworzymy własne modele porównawcze.

- model9- Alcohol+Ash+Hue+Flavonoids
- model2- Alcohol+Magnesium
- model3- Magnesium+Proanthocyanins+Proline+Flavonoids

```
library(ipred)

# budujemy model
model.knn.9 <- ipredknn(Class ~ Alcohol+Ash+Hue+Flavonoids, data=learning.set, k=2)
model.knn.2 <- ipredknn(Class ~ Alcohol+Magnesium, data=learning.set, k=1)
model.knn.3 <- ipredknn(Class ~ Magnesium+Proanthocyanins+Proline+Flavonoids , data=lear




etykietki.prog9 <- predict(model.knn.9, test.set,type="class")
etykietki.prog2 <- predict(model.knn.2, test.set,type="class")
etykietki.prog3 <- predict(model.knn.3, test.set,type="class")

# macierz pomyłek (ang. confusion matrix)
(wynik.tablica9 <- table(etykietki.prog9,etykietki.rzecz))

##                etykietki.rzecz
## etykietki.prog9  1  2  3
##               1 23  5  0
##               2  1 17  0
##               3  0  2 12

(wynik.tablica2 <- table(etykietki.prog2,etykietki.rzecz))

##                etykietki.rzecz
## etykietki.prog2  1  2  3
##               1 15  3  7
##               2  0  9  0
##               3  9 12  5

(wynik.tablica3 <- table(etykietki.prog3,etykietki.rzecz))

##                etykietki.rzecz
## etykietki.prog3  1  2  3
##               1 23  1  0
##               2  0 12  6
##               3  1 11  6
```

```
# błąd klasyfikacji
(n.test - sum(diag(wynik.tablica9))) / n.test

## [1] 0.1333333

(n.test - sum(diag(wynik.tablica2))) / n.test

## [1] 0.5166667

(n.test - sum(diag(wynik.tablica3))) / n.test

## [1] 0.3166667
```

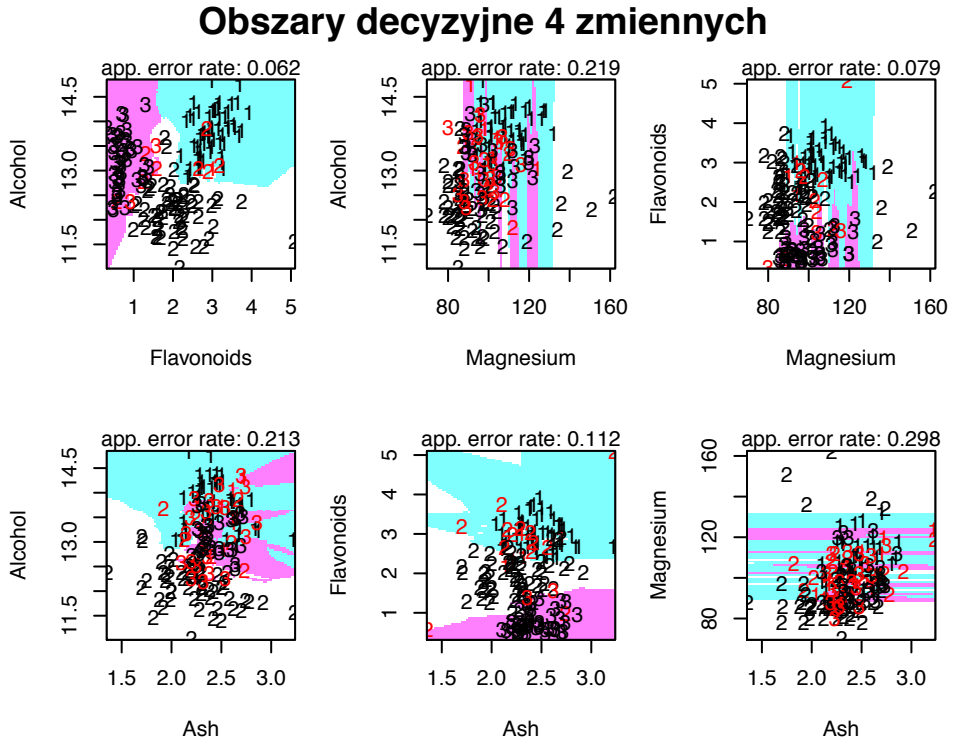
- Błąd klasyfikacji oparty z na modelu gotowej funkcji¡knn¿ wynosi 16.7 procent
- Błędy oparte na modelach 1-3 wynoszą kolejno: 10, 51.5 oraz 30 procent.
- Widzimy, że model 9 (porównywany wszedzie) ma największą skuteczność, ze względu na korzystanie z cech Alcohol i Flavonoids, które cechują sie wysoką dykretyzacją(widać na boxplotach)
- Na macierzach pomyłek widać że nie ma podobnych tendecji miedzy diagonalami macierzy co do wartości na nich

Porównanie błędów estymatorów klasyfikacji: 10 fold cross-validation, bootstrap i Algorytm ".632+"

```
library(ipred)

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

błąd_cv<-errorest(Class ~Alcohol, wine, model=my.ipredknn, predict=my.predict, estimator
błąd_cv$error

## [1] 0.3258427

błąd_boot<-errorest(Class ~Alcohol, wine, model=my.ipredknn, predict=my.predict, estimat
błąd_boot$error

## [1] 0.3531369

błąd_632plus<-errorest(Class ~Alcohol, wine, model=my.ipredknn, predict=my.predict, esti
błąd_632plus$error

## [1] 0.3203864
```

Błędy są na podobnym poziomie 31-35 procent, najbardzej skuteczny estymator *632+*, najmniej *boot*.

```
library(klaR)


partimat(wine$Class ~ Alcohol+Flavonoids+Magnesium+Ash, data = wine, method = "sknn",  p
```



Rysunek 11: Obszary decyzyjne

- Po wykresach obszarów decyzyjnych widzimy, że znowu najlepszą skuteczność posiada zestawienie Alcohol i Flavonoids. Po nim Flavonoids i Magnesium
- W niekótrych przypadkach widać maskowanie się klas jak np w Magnesium i Ash

2.2.4 Klasyfikacja z wykorzystaniem naiwnego klasyfikatora bayesowskiego

Konstruujemy prognozy

```
# prognozy dla zbioru uczącego
etykietki.prog.learn <- predict(model.NB, learning.set)
etykietki.prog.learn9 <- predict(model.NB9, learning.set)
# prognozy dla zbioru testowego
etykietki.prog.test <- predict(model.NB, test.set)
etykietki.prog.test9 <- predict(model.NB9, test.set)
```

Dla porównania tworzymy wektory zawierające rzeczywiste etykietki

```
etykietki.rzecz.learn <- learning.set$Class
etykietki.rzecz.test <- test.set$Class
```

Ocena dokładności prognoz (Macierz pomyłek i błędy na zbiorze uczącym i testowym)

```
ar1<-table(etykietki.prog.learn, etykietki.rzecz.learn)
ar2<-table(etykietki.prog.test, etykietki.rzecz.test)



# Do liczenia błędudu klasyfikacji na podstawie dwóch zestawów etykietek można posłuży

blad.klasyf <- function(etykietki.prog, etykietki.rzecz)
{
  n <- length(etykietki.prog)
  macierz.pomylek <- table(etykietki.prog,etykietki.rzecz)
  list(macierz.pomylek=macierz.pomylek, blad.klasyf=(n - sum(diag(macierz.pomylek))) / n
}

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

## $macierz.pomylek
##                etykietki.rzecz
## etykietki.prog  1  2  3
##              1 33  1  0
##              2  2 41  2
##              3  0  5 34
##
## $blad.klasyf
## [1] 0.08474576

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

## $macierz.pomylek
##                etykietki.rzecz
## etykietki.prog  1  2  3
##              1 24  4  0
##              2  0 19  0
##              3  0  1 12
##
## $blad.klasyf
## [1] 0.08333333

(blad.klasyf(etykietki.prog.learn9, etykietki.rzecz.learn))

## $macierz.pomylek
##                etykietki.rzecz
## etykietki.prog  1  2  3
##              1 34  2  0
```

```
##              2  1 44  0
##              3  0  1 36
##
## $blad.klasyf
## [1] 0.03389831
```

```
(blad.klasyf(etykietki.prog.test9, etykietki.rzecz.test))
```

```
## $macierz.pomylek
##                etykietki.rzecz
## etykietki.prog  1  2  3
##              1 24  2  0
##              2  0 22  0
##              3  0  0 12
##
## $blad.klasyf
## [1] 0.03333333
```

```
tab7 <- xtable( ar1, digits = 3, row.names = TRUE, caption = "Macierz pomyłek na zbiorze
print(tab7, type = "latex", table.placement="b")
```

```
tab8 <- xtable( ar2, digits = 3, row.names = TRUE, caption = "Macierz pomyłek na zbiorze
print(tab8, type = "latex", table.placement="b")
```

- Błąd dokładności modelu1 wynosi 8.5 i 8.3 procenta
- Błąd dokładności modelu9 (porównywanym) wynosi 3.4 i 3.3 procenta
- Widać bardzo zbliżone do siebie wartości między zbiore uczącym i testowym
- Szybkość implementacji i działania algorytmy Bayesa jest na wysokim poziomie

Porównanie błędów schematów klasyfikacji: 10 fold cross-validation, bootstrap i Algorytm ".632+"

```
library(ipred)

edit_predict  <- function(model, newdata)
{ predict(model, newdata=newdata, type="class") }
```

	1	2	3
1	33	1	0
2	2	41	2
3	0	5	34

Tabela 5: Macierz pomyłek na zbiorze uczącym

```
edit_naiveBayes <- function(formula, data)
{ naiveBayes(formula=formula,data=data)}

# 10-krotny sprawdzian krzyżowy (10 fold cross-validation)
#polega na losowym podziale ca?ego zbioru na 10 r?wnolicznych zbior?w i wykonujemy 10
# zmieniaj?c pozbi?r kt?ry pe?ni role zbioru testowego, pozosta?e zbiory tworz? zbi?r
# Schemat cross-validation estymuje efektynie oczekiwany błąd na zbiorze testowym
(blad.cv <- errorest(Class~Alcohol+Flavonoids, wine, model=edit_naiveBayes, predict=edit
                     estimator="cv", est.para=control.errorest(k = 10)))$error

## [1] 0.08426966
```

```
# Algorytm bootstrap


(blad.bootstrap <- errorest(Class~Alcohol+Flavonoids, wine, model=edit_naiveBayes, predi
                            estimator="boot", est.para=control.errorest(nboot = 20)))$er

## [1] 0.08632797
```

```
# Algorytm ".632+"
# modyfikacja algorytmu bootstrap

(blad.632 <- errorest(Class~Alcohol+Flavonoids, wine, model=edit_naiveBayes, predict=edi
                            estimator="632plus", est.para=control.errorest(nboot = 20)))

## [1] 0.07869741
```

Wszystkie na poziomie błędu 8-9 procent. W typ przypadku najskuteczniejszy Algorytm 632+.

2.3 Porównania i wnioski

Najważniejsze wnioski, jakie udało się wysnuć na podstawie przeprowadzonych analiz.

- W metodzie drzew klasyfikacyjnych mamy duże możliwości zmieniania poszczególnych rozgałęzień, dzięki czemu możemy go dosotowac pod własne potrzeby klasyfikacji.
- Metoda Knn w implementacji przypomina naive Bayes, występują tu macierze pomyłek i korzystamy z gotowych funkcji w R. Wykresy obszarów decyzyjnych dają nam wgląd do wszystich zestawień cech.

	1	2	3
1	24	4	0
2	0	19	0
3	0	1	12

Tabela 6: Macierz pomyłek na zbiorze testowym

- Algorytmy cv, 632+ i bootstrap cechuja się bardzo podobna skutecznoscia, zarówno w modelu knn i naive Bayes "632+" generuje najniższy błąd
- Zdecydowanie używanie zmiennych, które wykazywały największe cechy dyskretyzacji dają największą skuteczność- generują najmniejsze błedy przy porównaniu z rzeczywistymi etykietami klas. Zmienne Alcohol, Flavonoid są bardziej optymalne procesów klasyfikacji niż takie jak Ash czy Magnesium
- Aby porównanie metod klasyfikacji było wiarygodne, wszędzie pojawiał się model9, który bazowal na tych samych cechach. Analizy przeprowadzone na zbiorach testowych wskazują na największa skutecznosc metody naive Bayes (błąd 3.3 proc), później drzewa klasyfikacyjne (błąd 8,3 proc), a na końcu metoda knn (błąd 11.5).
- Błedy w naive Bayes są bardzo bliskie siebie- w zbiorze testowym i uczącym

Literatura

[1] dr inż. Adam Zagdański, http://prac.im.pwr.wroc.pl/~zagdan/polish_ver/ED2020/index.html, 2020.