Lista 3

Eksploracja danych

Igor Misterowicz, 282245

2025-05-28

Contents

1 Zadanie 1

1.1 a.

Klasyfikacja danych iris

```r
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
```

1.2 b.

Dzielę dane na zbiór uczący i testowy w proporcji 2:1.

```r
choices <- sample(1:nrow(iris),nrow(iris)*2/3)
training <- iris[choices,]
training <- training[order(training$Species),]
test <- iris[-choices,]
test <- test[order(test$Species),]
nrow(training)
```

```
## [1] 100
```

```r
nrow(test)
```

```
## [1] 50
```

1.3 c.

Konstruuję klasyfikator.

```r
X <- cbind(rep(1,nrow(training)), training[,1:4])
X <- as.matrix(X)

Y <- matrix(0, nrow=nrow(training), ncol=3)

num_labels <- as.numeric(training$Species)

for (i in 1:3)
  Y[num_labels == i, i] <- 1
```

```
B.hat <- solve(t(X)%*%X) %*% t(X) %*% Y
Y.hat <- X%*%B.hat

est_1 <- levels(training$Species)[apply(Y.hat, 1, FUN=function(x) which.max(x))]

head(est_1)
```

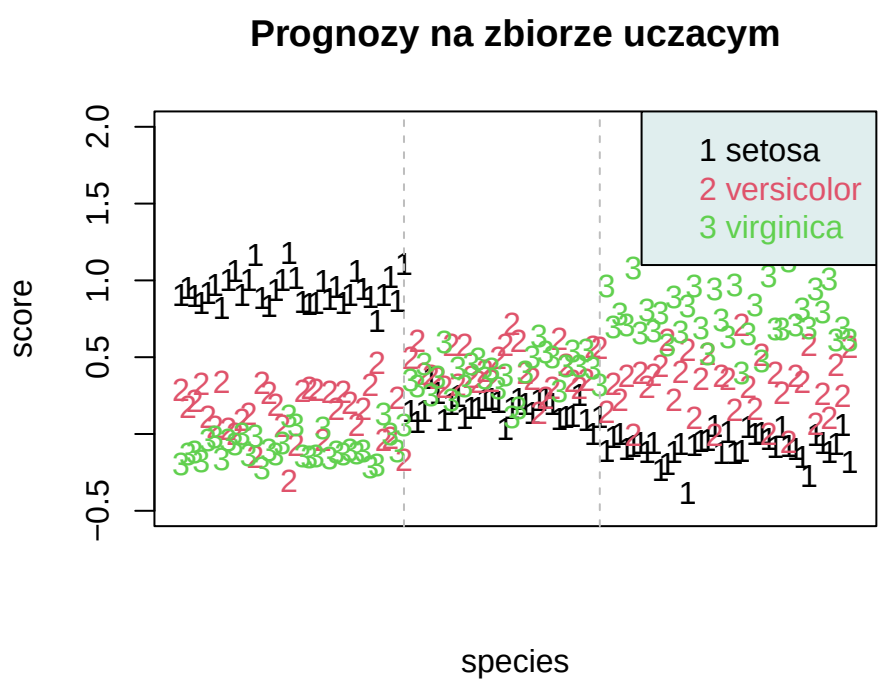
```
## [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
```

```
Z <- cbind(rep(1,nrow(test)), test[,1:4])
Z <- as.matrix(Z)

W <- matrix(0, nrow=nrow(test), ncol=3)

num_labels <- as.numeric(test$Species)

for (i in 1:3)
  W[num_labels == i, i] <- 1
W.hat <- Z%*%B.hat

est_2 <- levels(test$Species)[apply(W.hat, 1, FUN=function(x) which.max(x))]

head(est_2)
```

```
## [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
```

Wizualizacja wyników na zbiorze uczącym. Pionowe linie przerywane tworzą przedziały dla prawdziwych klas. Po lewej mamy setosę, na środku versicolor, a na końcu virginicę.

```
line_1 <- sum(training$Species == "setosa")
line_2 <- line_1 + sum(training$Species == "versicolor")
matplot(Y.hat, main="Prognozy na zbiorze uczącym",xlab="species", ylim=c(-.5,2),
        ylab = "score", xaxt = "n")
abline(v=c(line_1, line_2), lty=2, col="gray")
legend(x="topright", legend=paste(1:3,levels(training$Species)), col=1:3,
       text.col=1:3, bg="azure2")
```



Macierz pomyłek oraz skuteczność na zbiorze uczącym

```
confusion_1 <- table(training$Species,est_1)

qual_1 <- sum(diag(confusion_1))/nrow(training)

confusion_1
```

```
##              est_1
##               setosa versicolor virginica
##   setosa          34          0         0
##   versicolor       0         14        15
##   virginica        0          4        33
```

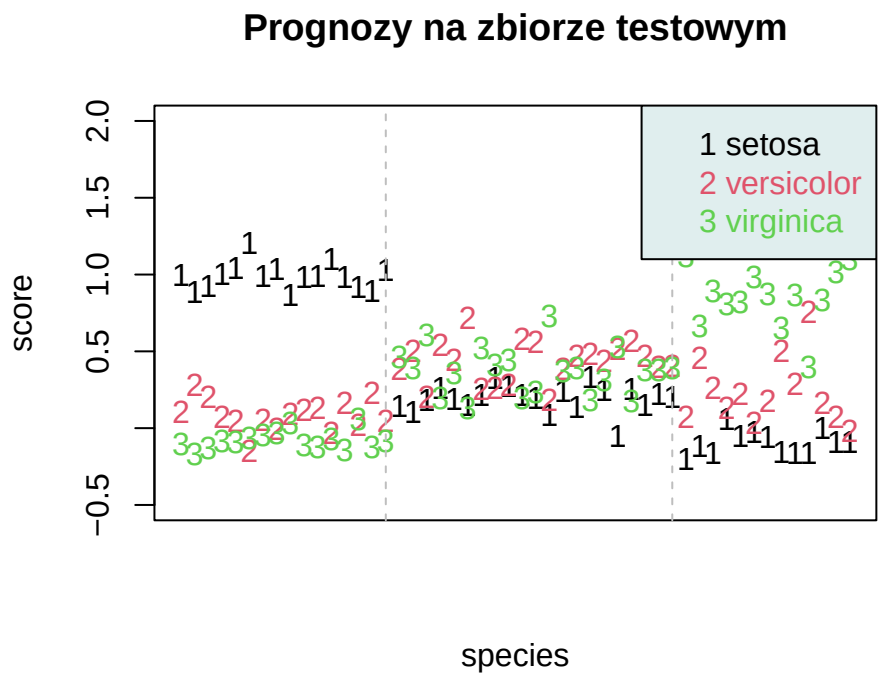
```
qual_1
```

```
## [1] 0.81
```

Widzimy, że często nieprawidłowo klasyfikowana jest odmiana versicolor.

Wizualizacja wyników na zbiorze testowym

```
line_1 <- sum(test$Species == "setosa")
line_2 <- line_1 + sum(test$Species == "versicolor")
matplot(W.hat, main="Prognozy na zbiorze testowym",xlab="species",
        ylim=c(-.5,2), ylab = "score", xaxt = "n")
abline(v=c(line_1, line_2), lty=2, col="gray")
legend(x="topright", legend=paste(1:3,levels(test$Species)), col=1:3,
       text.col=1:3, bg="azure2")
```



1.4 d.

Macierz pomyłek oraz skuteczność na zbiorze testowym

```
confusion_2 <- table(test$Species,est_2)

qual_2 <- sum(diag(confusion_2))/nrow(test)

confusion_2
```

```
##              est_2
##               setosa versicolor virginica
##   setosa          16          0         0
```

```
##   versicolor      0         14         7
##   virginica       0          1        12
```

```
qual_2
```

```
## [1] 0.84
```

Tutaj również widzimy problem z klasyfikacją odmiany versicolor, co sugeruje, że może zachodzić zjawisko maskowania tej klasy.

1.5 e.

Tworzę nowy zbiór danych z dodatkowymi kolumnami.

```
sqr_iris <- iris
sqr_iris$sqr_PL <- iris$Petal.Length^2
sqr_iris$sqr_PW <- iris$Petal.Width^2
sqr_iris$sqr_SL <- iris$Sepal.Length^2
sqr_iris$sqr_SW <- iris$Sepal.Width^2
sqr_iris$PL_PW <- iris$Petal.Length*iris$Petal.Width
sqr_iris$PL_SW <- iris$Petal.Length*iris$Sepal.Width
sqr_iris$PL_SL <- iris$Petal.Length*iris$Sepal.Length
sqr_iris$PW_SL <- iris$Petal.Width*iris$Sepal.Length
sqr_iris$PW_SW <- iris$Petal.Width*iris$Sepal.Width
sqr_iris$SL_SW <- iris$Sepal.Length*iris$Sepal.Width
```

Dzielę na podzbiór uczący i testowy.

```
choices <- sample(1:nrow(sqr_iris),nrow(sqr_iris)*2/3)
training <- sqr_iris[choices,]
training <- training[order(training$Species),]
test <- sqr_iris[-choices,]
test <- test[order(test$Species),]
nrow(training)
```

```
## [1] 100
```

```
nrow(test)
```

```
## [1] 50
```

Konstruuję model.

```
X <- cbind(rep(1,nrow(training)), training[,-5])
X <- as.matrix(X)

Y <- matrix(0, nrow=nrow(training), ncol=3)

num_labels <- as.numeric(training$Species)

for (i in 1:3)
  Y[num_labels == i, i] <- 1

B.hat <- solve(t(X)%*%X) %*% t(X) %*% Y
Y.hat <- X%*%B.hat

est_1 <- levels(training$Species)[apply(Y.hat, 1, FUN=function(x) which.max(x))]

head(est_1)
```

```
## [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
```

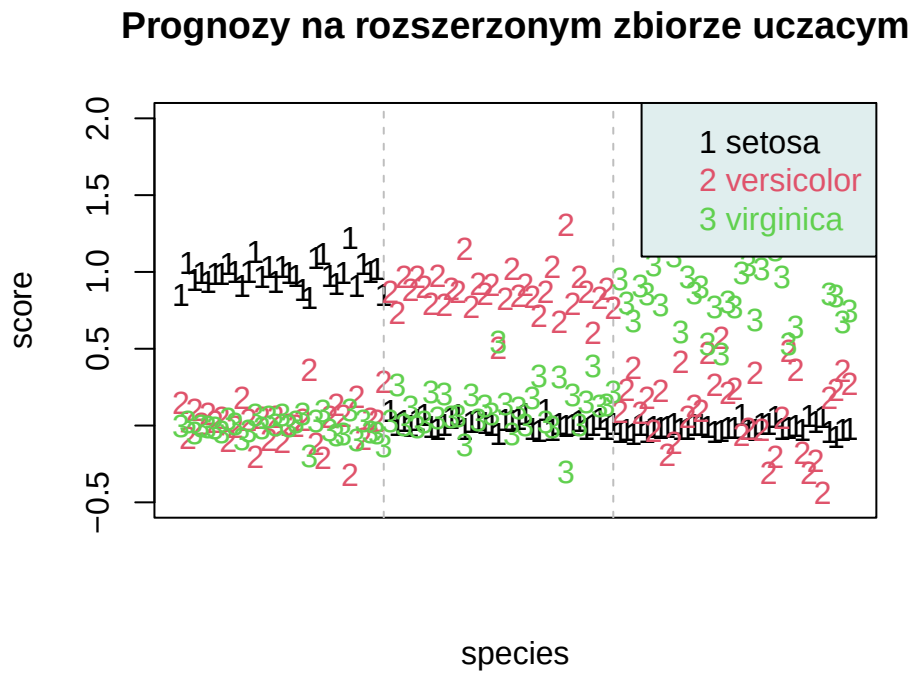
```
Z <- cbind(rep(1,nrow(test)), test[,-5])
Z <- as.matrix(Z)

W <- matrix(0, nrow=nrow(test), ncol=3)

num_labels <- as.numeric(test$Species)

for (i in 1:3)
  W[num_labels == i, i] <- 1
W.hat <- Z%*%B.hat

est_2 <- levels(test$Species)[apply(W.hat, 1, FUN=function(x) which.max(x))]

head(est_2)
```

```
## [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
```

Wizualizacja wyników na zbiorze uczącym

```
line_1 <- sum(training$Species == "setosa")
line_2 <- line_1 + sum(training$Species == "versicolor")
matplot(Y.hat, main="Prognozy na rozszerzonym zbiorze uczącym",
        xlab="species", ylim=c(-.5,2), ylab = "score", xaxt = "n")
abline(v=c(line_1, line_2), lty=2, col="gray")
```

```
legend(x="topright", legend=paste(1:3,levels(training$Species)), col=1:3,
       text.col=1:3, bg="azure2")
```



Macierz pomyłek oraz skuteczność na zbiorze uczącym

```
confusion_1 <- table(training$Species,est_1)

qual_1 <- sum(diag(confusion_1))/nrow(training)

confusion_1
```

```
##              est_1
##               setosa versicolor virginica
##    setosa         31          0         0
##    versicolor      0         33         1
##    virginica       0          1        34
```

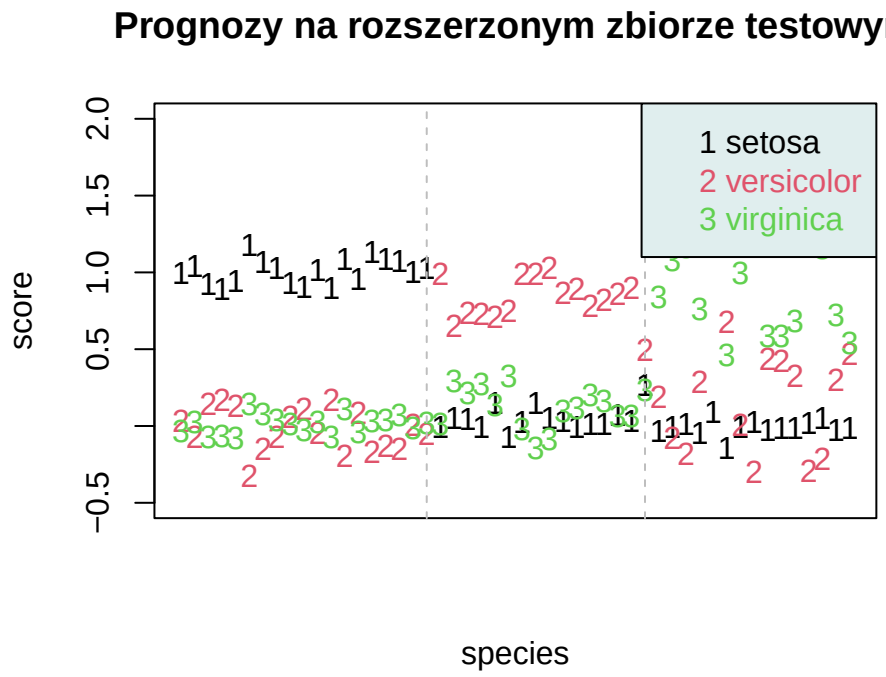
```
qual_1
```

```
## [1] 0.98
```

Wyniki są niemal idealne.

Wizualizacja wyników na zbiorze testowym.

```
line_1 <- sum(test$Species == "setosa")
line_2 <- line_1 + sum(test$Species == "versicolor")
matplot(W.hat, main="Prognozy na rozszerzonym zbiorze testowym",
        xlab="species", ylim=c(-.5,2), ylab = "score", xaxt = "n")
abline(v=c(line_1, line_2), lty=2, col="gray")
legend(x="topright", legend=paste(1:3,levels(test$Species)), col=1:3,
       text.col=1:3, bg="azure2")
```



Macierz pomyłek oraz skuteczność na zbiorze testowym

```
confusion_2 <- table(test$Species,est_2)

qual_2 <- sum(diag(confusion_2))/nrow(test)

confusion_2
```

```
##             est_2
##              setosa versicolor virginica
##   setosa         19          0         0
##   versicolor      0         16         0
##   virginica       0          1        14
```

```
qual_2
```

```
## [1] 0.98
```

Na zbiorze testowym klasyfikator zadziałał prawie bezbłędnie, udało się wyeliminować zjawisko przysłaniania klas. Tak znaczące zwiększenie skuteczności modelu sugeruje, że między zmiennymi danych iris mamy nieliniowe zależności.

2 Zadanie 2

2.1 a.

Wczytuję dane PimaIndiansDiabetes2.

```
data("PimaIndiansDiabetes2")
data <- PimaIndiansDiabetes2
head(data)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1        6     148       72      35      NA 33.6    0.627  50      pos
## 2        1      85       66      29      NA 26.6    0.351  31      neg
## 3        8     183       64      NA      NA 23.3    0.672  32      pos
## 4        1      89       66      23      94 28.1    0.167  21      neg
## 5        0     137       40      35     168 43.1    2.288  33      pos
## 6        5     116       74      NA      NA 25.6    0.201  30      neg
```

Liczba cech

```
ncol(data)
```

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

Liczba przypadków

```
nrow(data)
```

```
## [1] 768
```

Mamy dwie klasy tj. osoby z cukrzycą oraz bez. Ta informacja kodowana jest w zmiennej diabetes.

Sprawdzam brakujące dane

```
colSums(is.na(data))
```

```
## pregnant  glucose pressure  triceps  insulin     mass pedigree      age
##        0        5       35      227      374       11        0        0
## diabetes
##        0
```

Występują liczne brakujące dane kodowane za pomocą NA. Większość z nich znajduję się w zmiennych triceps i insulin.

Sprawdzam typy zmiennych.

```
str(data)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ pregnant: num  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose : num  148 85 183 89 137 116 78 115 197 125 ...
##  $ pressure: num  72 66 64 66 40 74 50 NA 70 96 ...
##  $ triceps : num  35 29 NA 23 35 NA 32 NA 45 NA ...
##  $ insulin : num  NA NA NA 94 168 NA 88 NA 543 NA ...
##  $ mass    : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 NA ...
##  $ pedigree: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age     : num  50 31 32 21 33 30 26 29 53 54 ...
##  $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...
```

Typy zostały rozpoznane poprawnie.

W celu przgotowania danych do analizy zastępuję brakujące wartości średnimi z odpowiednich kolumn.

```
fill <- function(column){
  mn <- sum(column[!is.na(column)])/ sum(!is.na(column))
  mn <- round(mn,2)
  sapply(column, FUN = function(x) ifelse(is.na(x),x <- mn,x <- x))
}
data[,1:8] <- lapply(data[,1:8], FUN = fill)

head(data)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1        6     148       72   35.00  155.55 33.6    0.627  50      pos
## 2        1      85       66   29.00  155.55 26.6    0.351  31      neg
## 3        8     183       64   29.15  155.55 23.3    0.672  32      pos
## 4        1      89       66   23.00   94.00 28.1    0.167  21      neg
## 5        0     137       40   35.00  168.00 43.1    2.288  33      pos
## 6        5     116       74   29.15  155.55 25.6    0.201  30      neg
```

2.2 b.

W dalszej części porównam ze sobą algorytmy klasyfikacyjne:

- k najbliższych sąsiadów
- drzewo klasyfikacyjne
- naiwny klasyfikatow bayesowski

2.3 c.

Badam rozkład klas

```
table(data$diabetes)
```

```
##
## neg pos
## 500 268
```

Około 1/3 danych stanowią wyniki pozytywne, zatem przypisując wszystkim obserwacjom wynik negatywny będziemy mieli rację w 2/3 przypadków.

Sprawdzam wariancję poszczególnych zmiennych.

```
lapply(data[,1:8], FUN = function(x) sd(x)^2)
```

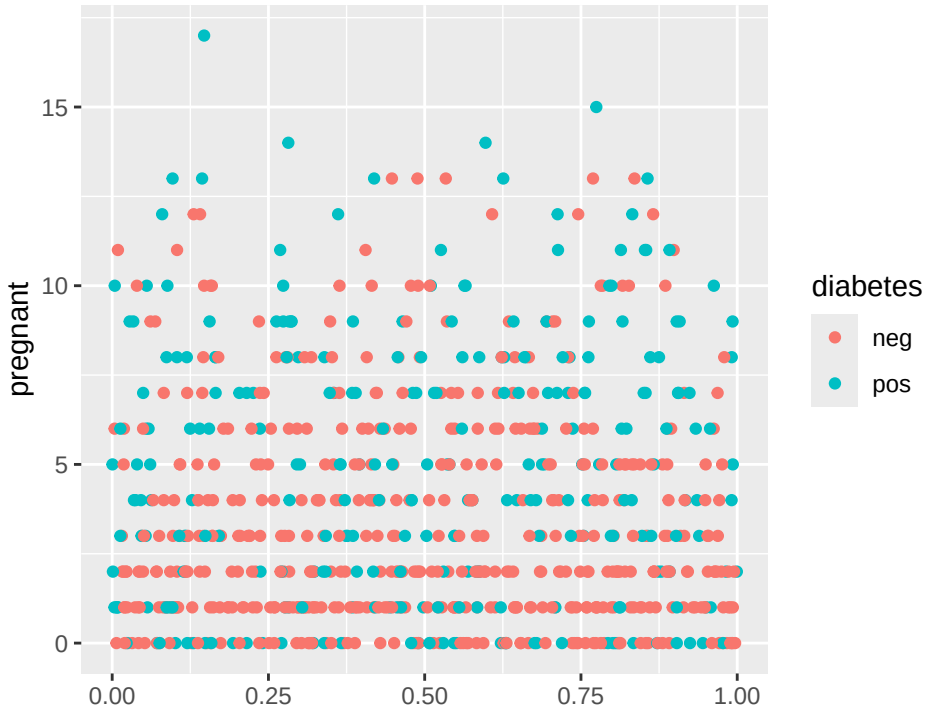
```
## $pregnant
## [1] 11.35406
##
## $glucose
## [1] 926.347
##
## $pressure
## [1] 146.3216
##
## $triceps
## [1] 77.28066
##
## $insulin
## [1] 7228.589
##
## $mass
## [1] 47.26771
##
```

```
## $pedigree
## [1] 0.1097786
##
## $age
## [1] 138.303
```

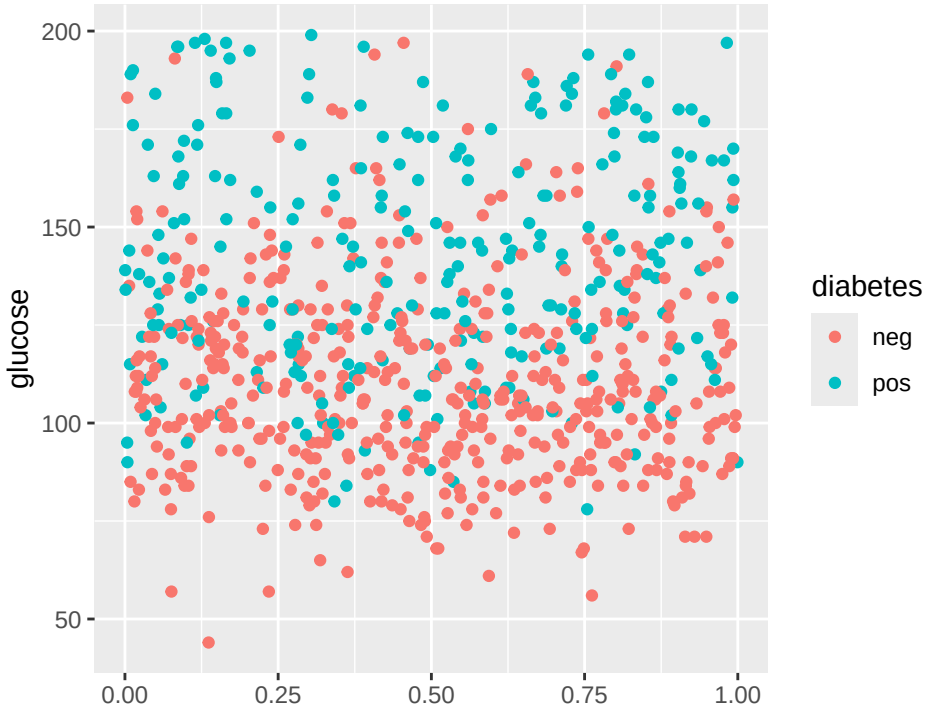
Widzimy duże różnice w wariancjach, co sugeruje konieczność standaryzacji. Największą zmienność obserwujemy w insulin.

Przedstawiam graficznie zdolności grupujące poszczególnych zmiennych.

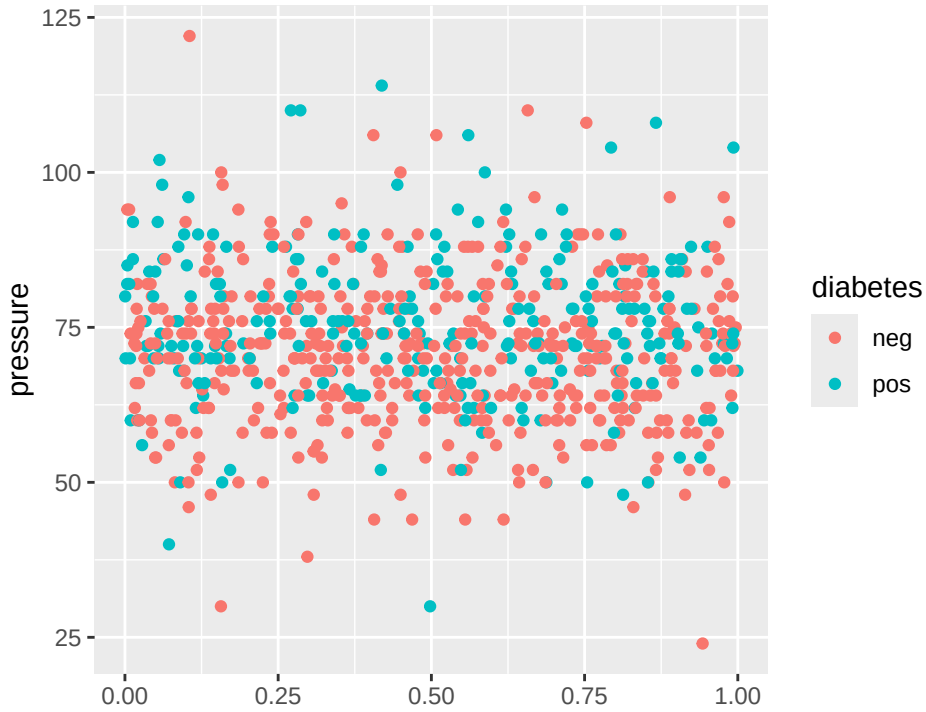
```
x_ax = runif(nrow(data))

ggplot(data,aes(x = x_ax, y = pregnant, colour = diabetes)) +
  geom_point() +xlab("")
```



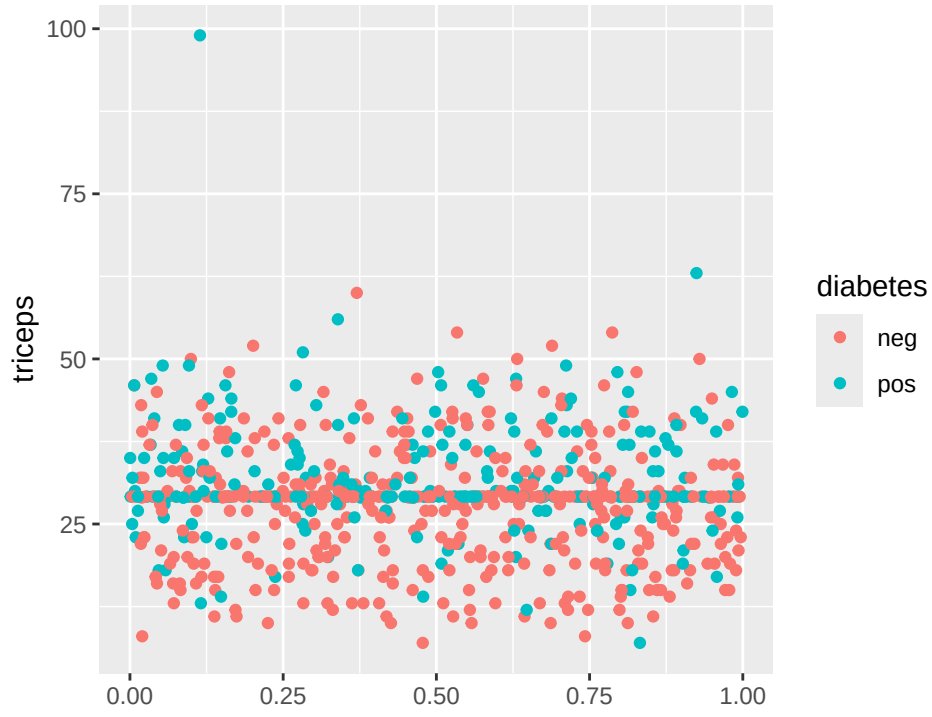

```
ggplot(data,aes(x = x_ax, y = glucose, colour = diabetes)) +
  geom_point() + xlab("")
```



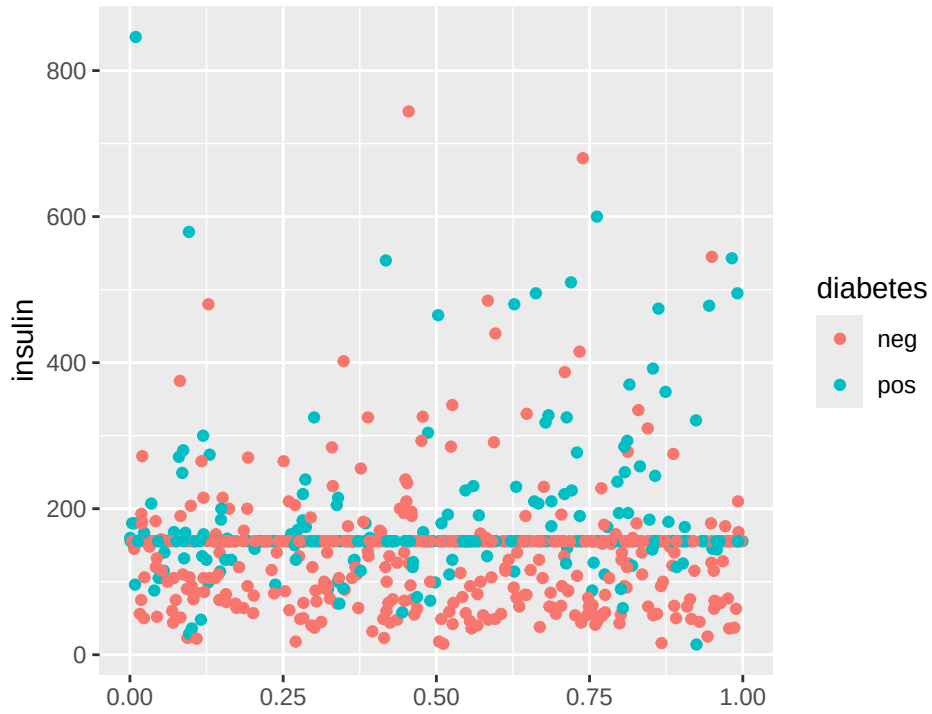

```
ggplot(data,aes(x = x_ax, y = pressure, colour = diabetes)) +
  geom_point() + xlab("")
```



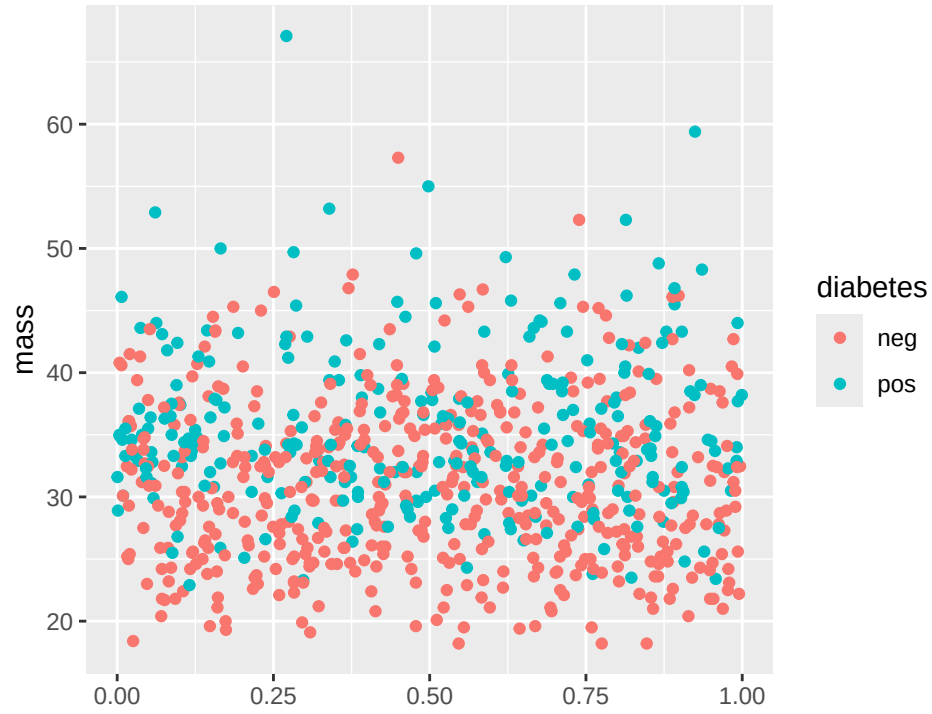
```
ggplot(data,aes(x = x_ax, y = triceps, colour = diabetes)) +
  geom_point() + xlab("")
```



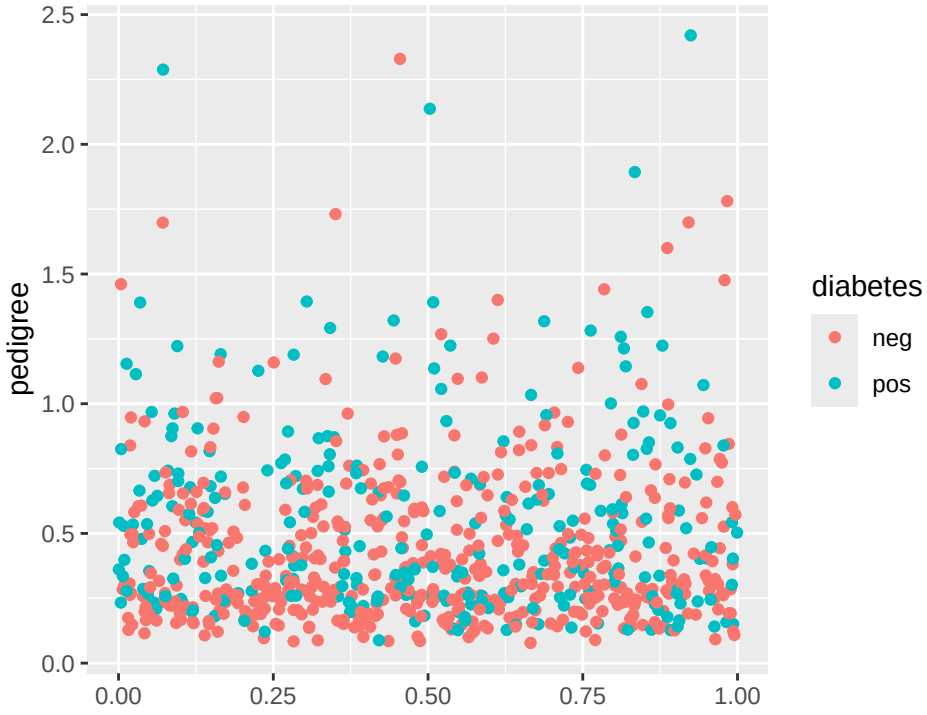

```
ggplot(data,aes(x = x_ax, y = insulin, colour = diabetes)) +
  geom_point() + xlab("")
```



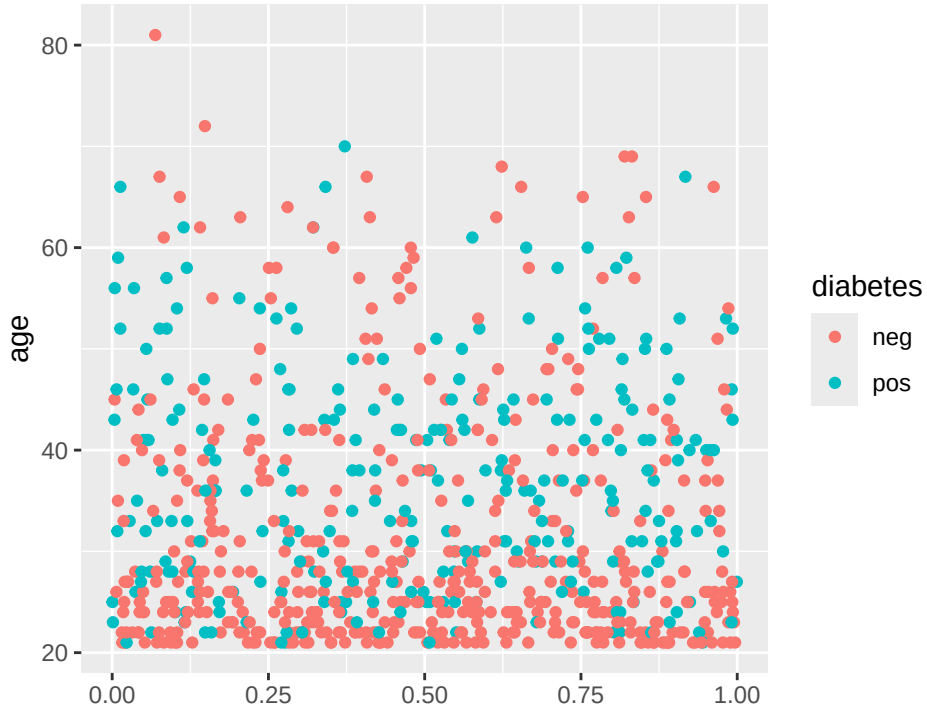

```
ggplot(data,aes(x = x_ax, y = mass, colour = diabetes)) +
  geom_point() + xlab("")
```



```
ggplot(data,aes(x = x_ax, y = pedigree, colour = diabetes)) +
  geom_point() + xlab("")
```




```
ggplot(data,aes(x = x_ax, y = age, colour = diabetes)) +
  geom_point() + xlab("")
```



Wykresy sugerują że zmiennymi o najlepszych zdolnościach klasyfikujących będą insulin oraz glucose, a zmienną o najgorszych będzie pressure.

2.4 d.

Dzielę dane na podzbiór uczący i testowy

```
set.seed(123)
choices <- sample(1:nrow(data),nrow(data)*2/3)
training <- data[choices,]
test <- data[-choices,]
nrow(training)
```

```
## [1] 512
```

```
nrow(test)
```

```
## [1] 256
```

Zaczynam od algrorytmu KNN, w tym celu konieczna jest standaryzajcja. Używam wszystkich dostępnych cech oraz K = 5.

```
training_scaled <- training
training_scaled[,1:8] <- as.data.frame(scale(training_scaled[,1:8]))

test_scaled <- test
test_scaled[,1:8] <- as.data.frame(scale(test_scaled[,1:8]))

test_real <- test_scaled$diabetes
training_real <- training_scaled$diabetes
```

Błąd na zbiorze testowym

```
est <- knn(training_scaled[,1:8], test_scaled[,1:8],
           training_scaled$diabetes, k=5)

(result <- table(est,test_real))
```

```
##      test_real
## est   neg pos
##   neg 136  30
##   pos  31  59
```

```
1 - (sum(diag(result))/ nrow(test_scaled))
```

```
## [1] 0.2382812
```

Błąd na zbiorze uczącym

```
est <- knn(training_scaled[,1:8], training_scaled[,1:8],
           training_scaled$diabetes, k=5)

(result <- table(est,training_real))
```

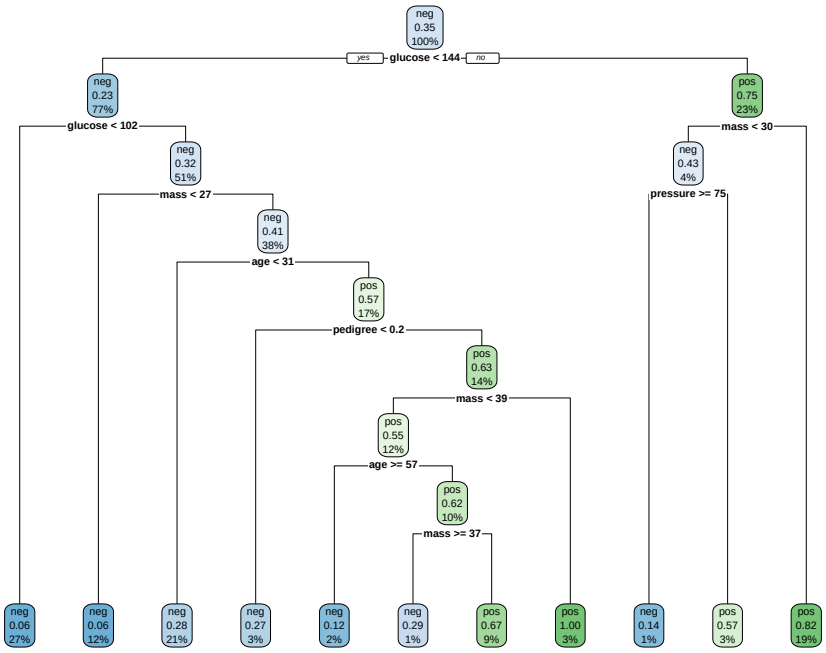
```
##      training_real
## est   neg pos
##   neg 295  49
##   pos  38 130
```

```
1 - (sum(diag(result))/ nrow(training_scaled))
```

```
## [1] 0.1699219
```

Następne jest drzewo decyzyjne. Tutaj również używam wszystkich dostępnych cech.

```
model <- diabetes ~ .
tree <- rpart(model, data = training)

rpart.plot(tree)
```




```
pred.labels.learning <- predict(tree, newdata=training, type = "class")
pred.labels.test <- predict(tree, newdata=test, type = "class")

n.learning <- nrow(training)
n.test <- nrow(test)

conf.mat.test <- table(pred.labels.test, test$diabetes)
conf.mat.learning <- table(pred.labels.learning, training$diabetes)
```

Skuteczność na zbiorze testowym

```
table(pred.labels.test, test$diabetes)
```

```
##
## pred.labels.test neg pos
##              neg 132  32
##              pos  35  57
```

```
1 - (sum(diag(conf.mat.test))/n.test)
```

```
## [1] 0.2617188
```

Skuteczność na zbiorze uczącym

```
table(pred.labels.learning, training$diabetes)
```

```
##
## pred.labels.learning neg pos
##                  neg 295  50
##                  pos  38 129
```

```
1 - (sum(diag(conf.mat.learning))/n.learning)
```

```
## [1] 0.171875
```

Konstruuję naiwny klasyfikator bayesowski.

```
model.NB <- naiveBayes(diabetes~., data = training)

etykietki.prog.learn <- predict(model.NB, training)
etykietki.prog.test <- predict(model.NB, test)

etykietki.rzecz.learn <- training$diabetes
etykietki.rzecz.test <- test$diabetes

blad.klasyf <- function(etykietki.prog, etykietki.rzecz)
{
  n <- length(etykietki.prog)
  macierz.pomylek <- table(etykietki.prog,etykietki.rzecz)
  list(macierz.pomylek=macierz.pomylek,
       jakosc.klasyf=1-(sum(diag(macierz.pomylek)) / n))
}
```

Sprawdzam skuteczność dla zbioru testowego

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

```
## $macierz.pomylek
##                 etykietki.rzecz
## etykietki.prog neg pos
##            neg 138  37
##            pos  29  52
##
## $jakosc.klasyf
## [1] 0.2578125
```

Sprawdzam skuteczność dla zbioru uczącego

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

```
## $macierz.pomylek
##                 etykietki.rzecz
## etykietki.prog neg pos
##            neg 281  72
##            pos  52 107
##
## $jakosc.klasyf
## [1] 0.2421875
```

Testuję teraz skuteczność tych algorytmów, ale stosując bardziej zaawansowane metody walidacji.

Cross validation oraz bootstrap dla KNN.

```
set.seed(123)
my.predict  <- function(model, newdata){
  predict(model, newdata=newdata, type="class")
}
my.ipredknn <- function(formula1, data1, ile.sasiadow){
  ipredknn(formula=formula1,data=data1,k=ile.sasiadow)
}

scaled_data <- data
scaled_data[,1:8] <- as.data.frame(scale(data[,1:8]))

errorest(diabetes ~., scaled_data, model=my.ipredknn, predict=my.predict,
         estimator="cv",
         est.para=control.errorest(k = 10), ile.sasiadow=5)
```

```
##
## Call:
```

```
## errorest.data.frame(formula = diabetes ~ ., data = scaled_data,
##     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:  0.2695
```

```
errorest(diabetes ~., scaled_data, model=my.ipredknn, predict=my.predict,
         estimator="boot",
         est.para=control.errorest(nboot = 50), ile.sasiadow=5)
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = scaled_data,
##     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:  0.2883
## Standard deviation: 0.0031
```

Cross validation oraz bootstrap dla drzewa decyzyjnego.

```
set.seed(123)
my.predict  <- function(model, newdata){
  predict(model, newdata=newdata, type="class")
}

my.rpart <- function(formula1, data1) {
  rpart(formula = formula1, data = data1)
}

errorest(diabetes ~ ., data = data, model = my.rpart, predict = my.predict,
         estimator = "cv", est.para = control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.rpart,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
```

```
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2487
```

```
errorest(diabetes ~ ., data = data, model = my.rpart, predict = my.predict,
         estimator = "boot", est.para = control.errorest(nboot = 50))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.rpart,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2663
## Standard deviation: 0.0033
```

Cross validation oraz bootstrap dla naiwnego bayesa.

```
set.seed(123)
my.predict  <- function(model, newdata){
  predict(model, newdata=newdata, type="class")
}

my.NB <- function(formula1, data1) {
  model.NB <- naiveBayes(formula1, data = data1)
}


errorest(diabetes ~ ., data = data, model = my.NB, predict = my.predict,
         estimator = "cv", est.para = control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.NB,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2526
```

```
errorest(diabetes ~ ., data = data, model = my.NB, predict = my.predict,
         estimator = "boot", est.para = control.errorest(nboot = 50))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.NB,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2501
## Standard deviation: 0.0015
```

- Metoda bootstrap surowiej oceniła algorytmy KNN oraz drzewa decyzycjnego, za to zwróciła niewiele korzystniejszy wynik niż cross validation przy metodzie naiwnego bayesa.
- KNN na zbiorze testowym został oceniony lepiej niż przy metodach bootstrap oraz cross validation.
- Drzewo decyzyjne na zbiorze uczącym zostało ocenione podobnie jak przy metodzie bootstrap, jednak cross validation zwróciło nieznacznie lepszy wynik.
- W przypadku naiwnego bayesa wyniki wszystkich trzech metod są podobne.

2.5 e.

Dla uproszczenia będę używał jedynie schematu testowania cross validation. Sprawdzam teraz skuteczność modeli na różnych podzbiorach dostępnych cech.

Model KNN

Najpierw wyrzucam triceps oraz pegigree.

```
set.seed(123)

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age ,
         scaled_data, model=my.ipredknn, predict=my.predict, estimator="cv",
         est.para=control.errorest(k = 10), ile.sasiadow=5)
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = scaled_data, 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:  0.2305
```

Teraz zostawiam jedynie glucose, insulin oraz age.

```
set.seed(123)

errorest(diabetes ~ glucose + insulin + age, scaled_data, model=my.ipredknn, predict=my.
         est.para=control.errorest(k = 10), ile.sasiadow=5)
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ glucose + insulin +
##     age, data = scaled_data, 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:  0.263
```

Model drzewa decyzyjnego

Wyrzucam ze zbioru cech triceps i pegigree.

```
set.seed(123)

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age,
         data = data, model = my.rpart, predict = my.predict,
         estimator = "cv", est.para = control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = data, model = my.rpart,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2344
```

Zostawiam jedynie glucose, insulin oraz age.

```
set.seed(123)

errorest(diabetes ~ glucose + insulin + age, scaled_data, model=my.rpart,
         predict=my.predict, estimator="cv",
         est.para=control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ glucose + insulin +
##     age, data = scaled_data, model = my.rpart, predict = my.predict,
##     estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2617
```

Model naiwny bayes

Wyrzucam ze zbioru cech triceps oraz pegigree.

```
set.seed(123)

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age,
         data = data, model = my.NB, predict = my.predict,
         estimator = "cv", est.para = control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = data, model = my.NB,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2539
```

Zostawiam jedynie glucose insulin oraz age.

```
set.seed(123)

errorest(diabetes ~ glucose + insulin + age, scaled_data, model=my.NB,
         predict=my.predict, estimator="cv",
         est.para=control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ glucose + insulin +
##     age, data = scaled_data, model = my.NB, predict = my.predict,
##     estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.263
```

We wszystkich przypadkach otrzymaliśmy lepsze wyniki wyrzucając zmienne triceps oraz pedigree. Są one nieznacznie gorsze jeśli ograniczyć się tylko do insulin, glucose oraz age. Dla modelu naiwny bayes zmiany były nieznaczne.

Testuję teraz różne parametry modeli na zbiorze cech, który dawał najlepsze wyniki tj. wszystkie oprócz triceps i pedigree.

Model KNN

3 sąsiadów

```
set.seed(123)

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age,
         scaled_data, model=my.ipredknn, predict=my.predict, estimator="cv",
         est.para=control.errorest(k = 10), ile.sasiadow=3)
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = scaled_data, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 3)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2721
```

15 sąsiadów

```
set.seed(123)

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age,
         scaled_data, model=my.ipredknn, predict=my.predict, estimator="cv",
         est.para=control.errorest(k = 10), ile.sasiadow=15)
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = scaled_data, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 15)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2357
```

50 sąsiadów

```
set.seed(123)

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age,
         scaled_data, model=my.ipredknn, predict=my.predict, estimator="cv",
         est.para=control.errorest(k = 10), ile.sasiadow=50)
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = scaled_data, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 50)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2422
```

Model drzewa decyzyjnego

Wariant z parametrami “gini”

```
set.seed(123)

my.rpart <- function(formula1, data1) {
  rpart(formula = formula1, data = data1, parms = list(split = "gini"))
}

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age,
         data = data, model = my.rpart, predict = my.predict,
         estimator = "cv", est.para = control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = data, model = my.rpart,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2344
```

Wariant z parametrem "information"

```
set.seed(123)

my.rpart <- function(formula1, data1) {
  rpart(formula = formula1, data = data1, parms = list(split = "information"))
}

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age,
         data = data, model = my.rpart, predict = my.predict,
         estimator = "cv", est.para = control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = data, model = my.rpart,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.237
```

Sprawdzam różne arguenty dla zmiennej "control"

```
set.seed(123)

my.rpart <- function(formula1, data1) {
  rpart(formula = formula1, data = data1, control = rpart.control(minsplit = 3, maxdepth
}

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age,
         data = data, model = my.rpart, predict = my.predict,
         estimator = "cv", est.para = control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = data, model = my.rpart,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2318
```

Model naiwnego bayesa

Sprawdzam parametry "laplace", "usekernel" oraz "adjust"

```
set.seed(123)

my.NB <- function(formula1, data1) {
  model.NB <- naiveBayes(formula1, data = data1, laplace = 1, usekernel = TRUE,
                         adjust = 0.5)
}

errorest(diabetes ~ pregnant + glucose + pressure + insulin + mass + age,
         data = data, model = my.NB, predict = my.predict,
         estimator = "cv", est.para = control.errorest(k = 10))
```

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ pregnant + glucose +
##     pressure + insulin + mass + age, data = data, model = my.NB,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2539
```

2.6 f.

- Sterując dostępnymi parametrami otrzymywałem wyniki gorsze lub jedynie nieznacznie lepsze np. w przypadku drzewa decyzyjnego.
- Pośród testowanych przeze mnie podzbiorów cech najlepszy jest taki, który składa się z wszystkich poza triceps oraz pedigree. Może to być spowodowane dużą liczbą brakujących danych w zmiennej triceps.

- Największą skuteczność miał algorytm KNN na powyższym podzbiorze cech. Rozsądną alternatywą wyadaje się również naiwny klasyfikator bayesowski, ponieważ dobrze sobie radził nawet z bardzo małą liczbą zmiennych. Dzięki pójściu na taki kompriomis mielibyśmy bardziej przejrzysty model.

- Przy metodach testowania skuteczności, największą rozbieżność udało się uzyskać przy algorytmie KNN, który wg. metody bootstrap miał błąd ok. 28.8%, a przy podziale na podzbiór uczący i testowy otrzymaliśmy błąd 23.8%. Ogólnie trudno zauważyć jakąś tendencję jeśli chodzi o metody testowania skuteczności.