

# Raport 3

## Eksploracja danych

Mikołaj Langner, Marcin Kostrzewa

nr albumów: 255716, 255749

2021-04-19

## Spis treści

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Wstęp</b>  | <b>1</b> |
| <b>2</b> | <b>Zadanie 1</b>  | <b>2</b> |
| 2.1      | Wczytanie danych i podział na zbiór uczący i testowy . . . . .                | 2        |
| 2.2      | Konstrukcja klasyfikatora i wyznaczenie prognoz . . . . .                     | 2        |
| 2.3      | Ocena jakości klasyfikacji . . . . .  | 4        |
| 2.4      | Zastosowanie regresji liniowej do modelu o rozszerzonej ilości cech . . . . . | 4        |
| <b>3</b> | <b>Zadanie 2</b>  | <b>7</b> |
| 3.1      | Wczytanie i krótka analiza danych . . . . .                                   | 7        |
| 3.2      | Metoda k-najbliższych sąsiadów . . . . .                                      | 8        |
| 3.3      | Drzewa klasyfikacyjne . . . . .   | 9        |
| 3.4      | Naiwny klasyfikator bayesowski . . . . .                                      | 12       |

## 1 Wstęp

Raport zawiera rozwiązania listy 3.

W zadaniu pierwszym budujemy klasyfikator na bazie metody regresji liniowej i oceniamy jego skuteczność i dokładność.

W zadaniu drugim . . . . Porównamy ze sobą rezultaty zastosowania:

- metoda k-najbliższych sąsiadów (*k-Nearest Neighbors*),
- drzewa klasyfikacyjne (*classification trees*),
- naiwny klasyfikator bayesowski (*naïve Bayes classifier*).

## 2 Zadanie 1

### 2.1 Wczytanie danych i podział na zbiór uczący i testowy

Wczytajmy dane o irysach i podzielmy je na zbiór uczący i testowy w proporcji 1 : 2.

```
data(iris)
n <- dim(iris)[1]

train.set.index <- sample(1:n, 2/3*n)
train.set <- iris %>% slice(train.set.index) %>% arrange(Species)
test.set <- iris %>% slice(-train.set.index) %>% arrange(Species)
```

### 2.2 Konstrukcja klasyfikatora i wyznaczenie prognoz

Stworzmy teraz macierze eksperymentu i wskaźnikową zarówno dla zbioru uczącego, jak i testowego. W tym celu wykorzystamy funkcję `dummyVars` z pakietu `Caret`.

```
dummies <- dummyVars(" ~ .", data=iris)

train.dummies <- predict(dummies, newdata = train.set)
train.X <- as.matrix(cbind(rep(1, nrow(train.dummies)),
                           train.dummies[, 1:4]))
train.Y <- train.dummies[, 5:7]

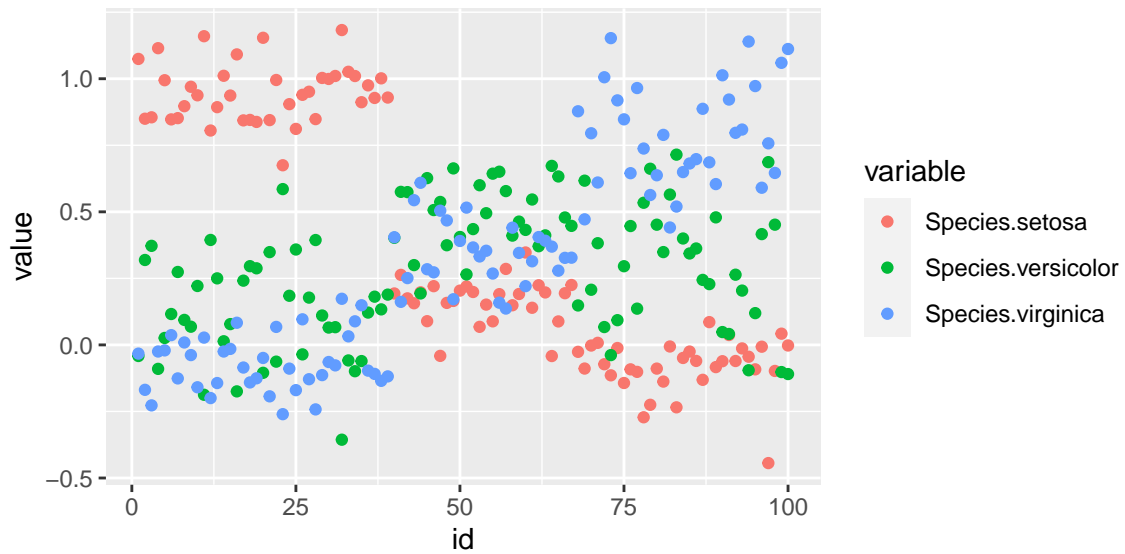
test.dummies <- predict(dummies, newdata = test.set)
test.X <- as.matrix(cbind(rep(1, nrow(test.dummies)), test.dummies[, 1:4]))
test.Y <- test.dummies[, 5:7]
```

Wykorzystując metodę najmniejszych kwadratów, wyznaczamy przewidywane prognozy klas dla obu zbiorów.

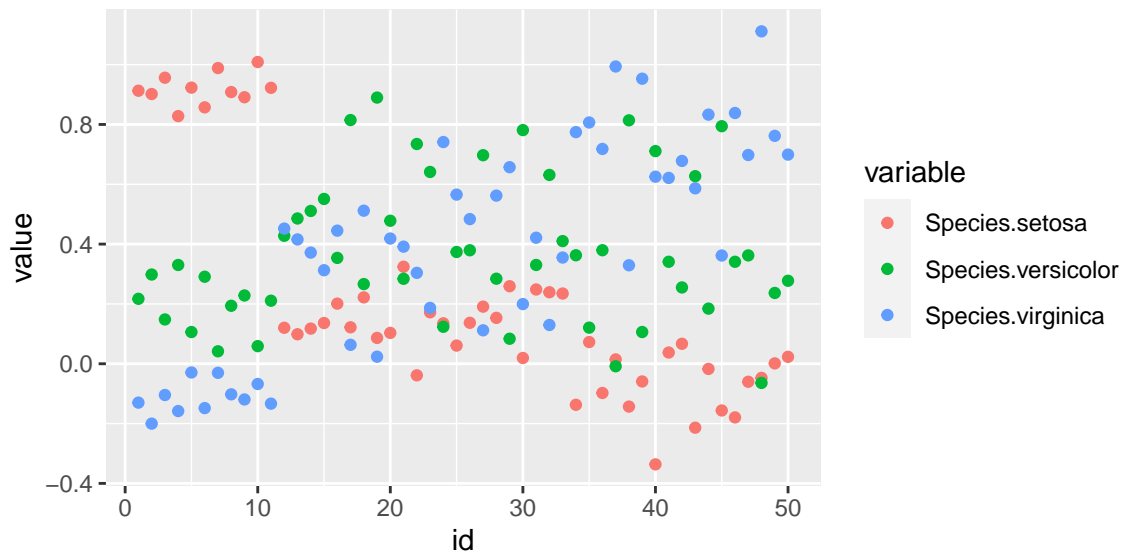
```
Y.hat <- solve(t(train.X) %*% train.X) %*% t(train.X) %*% train.Y

train.proba <- train.X %*% Y.hat
test.proba <- test.X %*% Y.hat
```

Przedstawmy prognozy klas na wykresach.



Rysunek 1: Prognozy klas dla zbioru uczącego.



Rysunek 2: Prognozy klas dla zbioru testowego.

## 2.3 Ocena jakości klasyfikacji

Wyznaczymy teraz macierz pomyłek dla zbioru uczącego.

|            | Species.setosa | Species.versicolor | Species.virginica |
|------------|----------------|--------------------|-------------------|
| setosa     | 39             | 0                  | 0                 |
| versicolor | 0              | 21                 | 7                 |
| virginica  | 0              | 4                  | 29                |

Tabela 1: Macierz pomyłek dla zbioru uczącego.

Błąd klasyfikacji to 0.11.

|            | Species.setosa | Species.versicolor | Species.virginica |
|------------|----------------|--------------------|-------------------|
| setosa     | 11             | 0                  | 0                 |
| versicolor | 0              | 12                 | 10                |
| virginica  | 0              | 4                  | 13                |

Tabela 2: Macierz pomyłek dla zbioru testowego.

Błąd klasyfikacji wynosi 0.28.

Wnioski i napomnienie o maskowaniu

## 2.4 Zastosowanie regresji liniowej do modelu o rozszerzonej ilości cech

Najpierw uzupełnijmy dane o irysach o składniki wielomianowe stopnia 2.

```
iris.quad <- (iris %>% select(-Species))^2
colnames(iris.quad) <- c("SL^2", "SW^2", "PL^2", "PW^2")
iris <- cbind(iris, combn(iris %>% select(-Species), 2,
                        FUN = Reduce, f = `*`),
            iris.quad)
```

Podobnie jak poprzednio podzielimy dane na zbiory: uczący i testowy, a następnie utworzymy macierze: eksperymentu i indykatorów.

```
train.set.index <- sample(1:n, 2/3*n)
train.set <- iris %>% slice(train.set.index) %>% arrange(Species)
test.set <- iris %>% slice(-train.set.index) %>% arrange(Species)

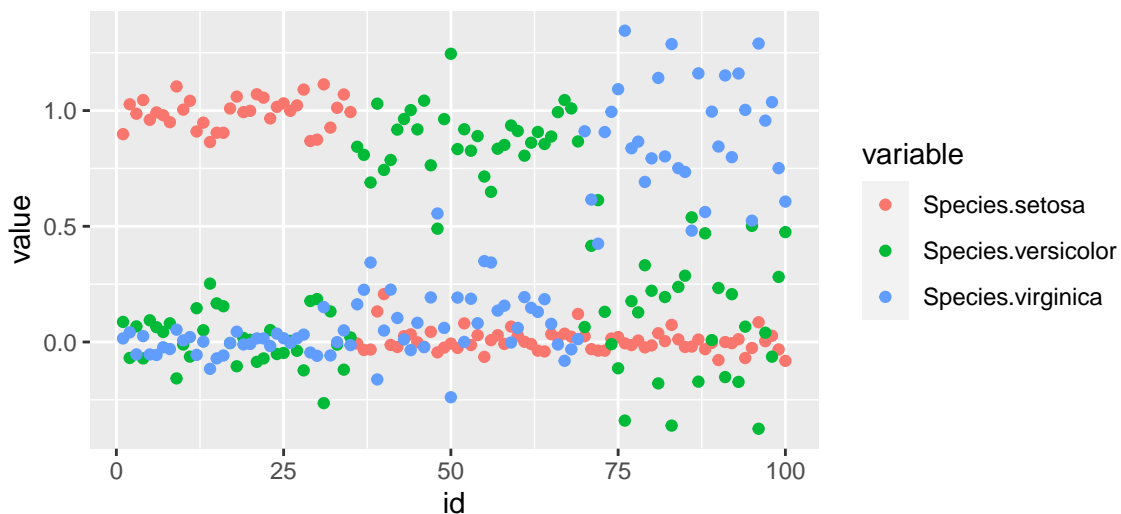
dummies <- dummyVars(" ~ .", data=iris)
train.dummies <- predict(dummies, newdata = train.set)
train.X <- as.matrix(cbind(rep(1, nrow(train.dummies)), train.dummies[, -c(5:7)]))
train.Y <- train.dummies[, 5:7]
```

```
test.dummies <- predict(dummies, newdata = test.set)
test.X <- as.matrix(cbind(rep(1, nrow(test.dummies)), test.dummies[, -c(5:7)]))
test.Y <- test.dummies[, 5:7]
```

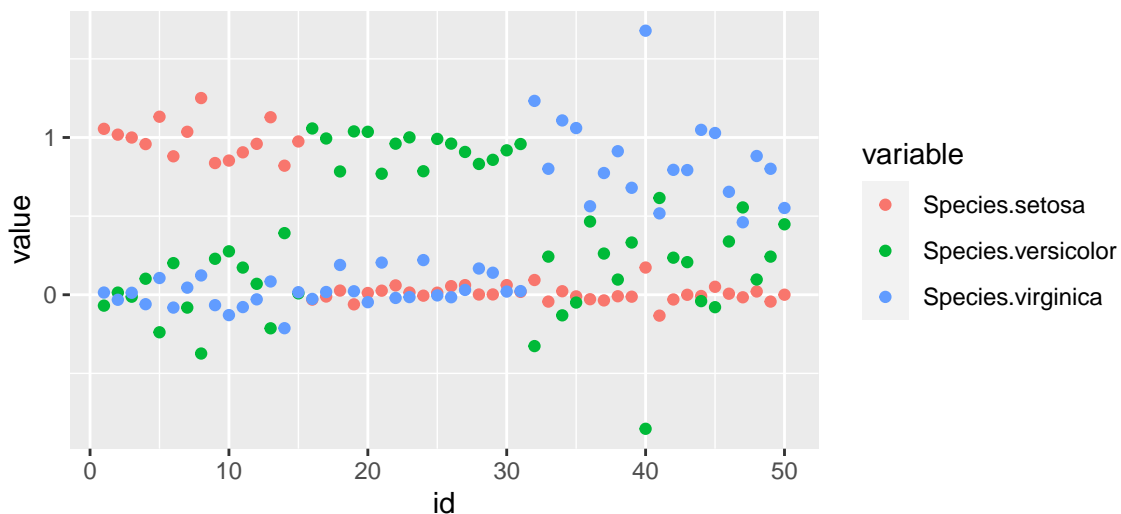
Ponownie, wyznaczmy prognozy klas i zwizualizujemy to przypisanie na wykresach.

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

train.proba <- train.X %*% Y.hat
test.proba <- test.X %*% Y.hat
```



Rysunek 3: Prognozy klas dla zbioru uczacego o rozszerzonej liczbie cech.



Rysunek 4: Prognozy klas dla zbioru uczacego o rozszerzonej liczbie cech.

Wyznaczymy także macierze pomyłek i błędy klasyfikacji.

|            | Species.setosa | Species.versicolor | Species.virginica |
|------------|----------------|--------------------|-------------------|
| setosa     | 35             | 0                  | 0                 |
| versicolor | 0              | 33                 | 1                 |
| virginica  | 0              | 2                  | 29                |

Tabela 3: Macierz pomylek dla zbioru uczacego dla przypadku o rozszerzonej liczbie cech.

Błąd klasyfikacji wynosi 0.03.

|            | Species.setosa | Species.versicolor | Species.virginica |
|------------|----------------|--------------------|-------------------|
| setosa     | 15             | 0                  | 0                 |
| versicolor | 0              | 16                 | 0                 |
| virginica  | 0              | 2                  | 17                |

Tabela 4: Macierz pomylek dla zbioru testowego dla przypadku o rozszerzonej liczbie cech.

Błąd klasyfikacji wynosi 0.04.

Wnioski i napomnienie o maskowaniu

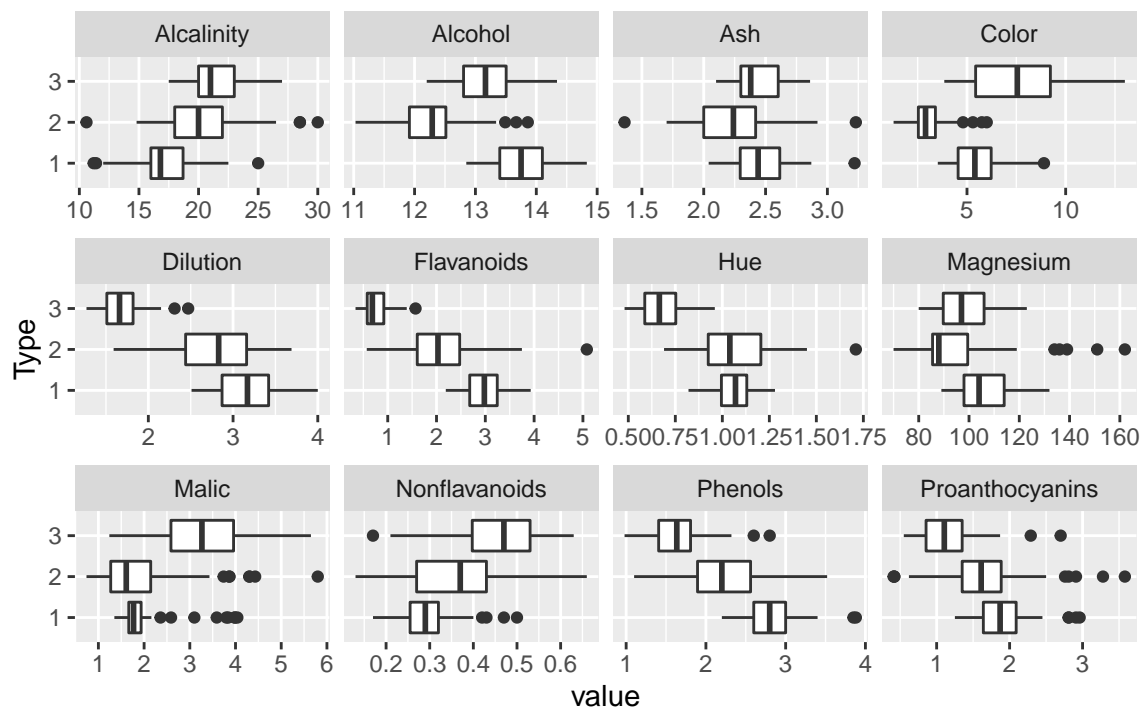
## 3 Zadanie 2

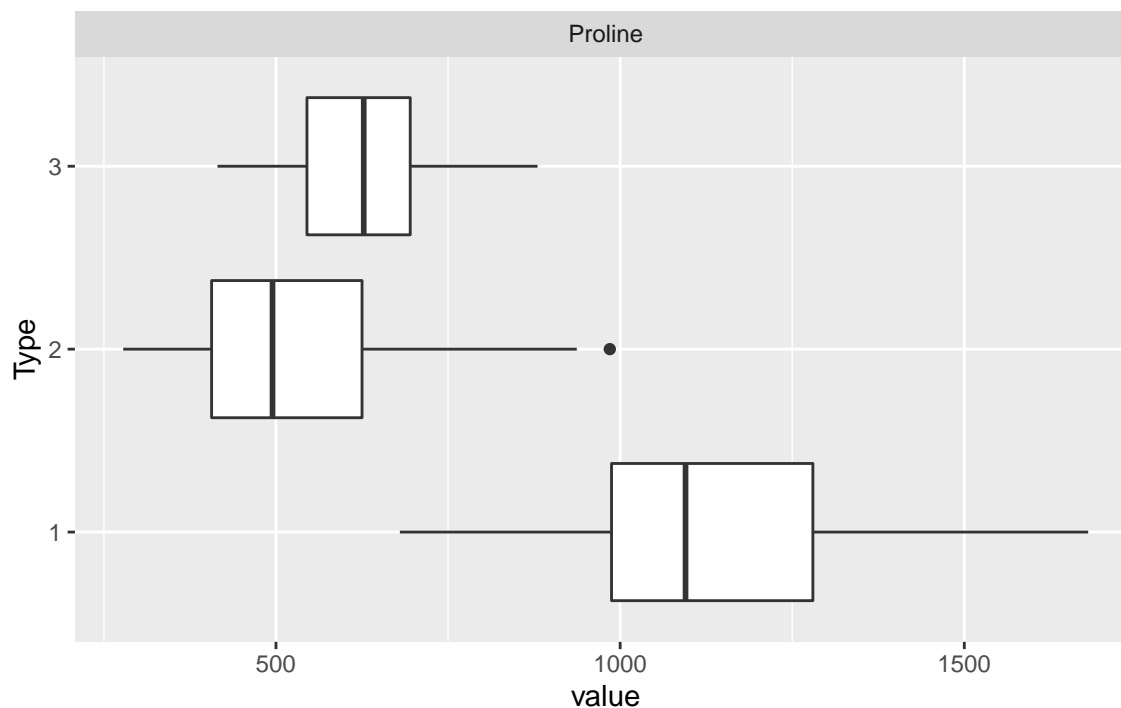
### 3.1 Wczytanie i krótka analiza danych

Wczytajmy i przygotujmy dane do dalszych analizy.

```
data(wine)
n <- dim(wine)[1]
wine <- drop_na(wine)
```

Przjrzyjmy się naszym danym na wykresach pudełkowych.





Page 2

Możemy zauważyć, że zmiennymi, które dobrze odróżniają zmienne ...

Podzielmy nasze dane na zbiór uczący i testowy w stosunku 2 : 1.

```
set.seed(42)
train.index <- sample(n, 2/3 * n)
train.data <- wine %>% slice(train.index)
test.data <- wine %>% slice(-train.index)
train.subset <- data.frame(train.data[, c(1, 2, 7, 8)])
test.subset <- data.frame(test.data[, c(1, 2, 7, 8)])
```

```
train.etiquettes <- train.data$Type
test.etiquettes <- test.data$Type
subset.train.etiquettes <- train.subset$Type
subset.test.etiquettes <- test.subset$Type
```

```
cv <- trainControl(method="cv", number=5)
```

### 3.2 Metoda k-najbliższych sąsiadów

```
model.knn.basic <- ipredknn(Type ~ ., data = train.data, k=5)

basic.knn.test.pred <- predict(model.knn.basic, test.data, type="class")
basic.knn.train.pred <- predict(model.knn.basic, train.data, type="class")
```



|   | 1  | 2  | 3  |
|---|----|----|----|
| 1 | 32 | 5  | 1  |
| 2 | 2  | 39 | 8  |
| 3 | 2  | 7  | 22 |

(a) Zbior uczacy

|   | 1  | 2  | 3 |
|---|----|----|---|
| 1 | 21 | 1  | 2 |
| 2 | 0  | 15 | 9 |
| 3 | 2  | 4  | 6 |

(b) Zbior testowy

Tabela 5: Macierze pomylek dla metody KNN — wszystkie cechy.

Błędy klasyfikacji to 0.2118644 i 0.3.

```
knn.model.subset <- ipredknn(Type ~ ., data = train.subset, k=5)

subset.knn.test.pred <- predict(knn.model.subset, test.subset, type="class")
subset.knn.train.pred <- predict(knn.model.subset, train.subset, type="class")
```

|   | 1  | 2  | 3  |
|---|----|----|----|
| 1 | 32 | 5  | 1  |
| 2 | 2  | 39 | 8  |
| 3 | 2  | 7  | 22 |

(a) Zbior uczacy

|   | 1  | 2  | 3 |
|---|----|----|---|
| 1 | 21 | 1  | 2 |
| 2 | 0  | 15 | 9 |
| 3 | 2  | 4  | 6 |

(b) Zbior testowy

Tabela 6: Macierze pomylek dla metody KNN — wybrance cechy.

Błędy klasyfikacji to 0.2118644 i 0.3.

```
model <- train(Type ~ ., data = train.data, method = "knn", trControl = cv)

tuned.knn.test.pred <- predict(model, test.data)
tuned.knn.train.pred <- predict(model, train.data)
```

|   | 1  | 2  | 3  |
|---|----|----|----|
| 1 | 33 | 3  | 1  |
| 2 | 1  | 40 | 8  |
| 3 | 2  | 8  | 22 |

(a) Zbior uczacy

|   | 1  | 2  | 3 |
|---|----|----|---|
| 1 | 21 | 1  | 3 |
| 2 | 0  | 15 | 8 |
| 3 | 2  | 4  | 6 |

(b) Zbior testowy

Tabela 7: Macierze pomylek dla metody KNN — stuningowany model.

Błędy klasyfikacji to 0.1949153 i 0.3.

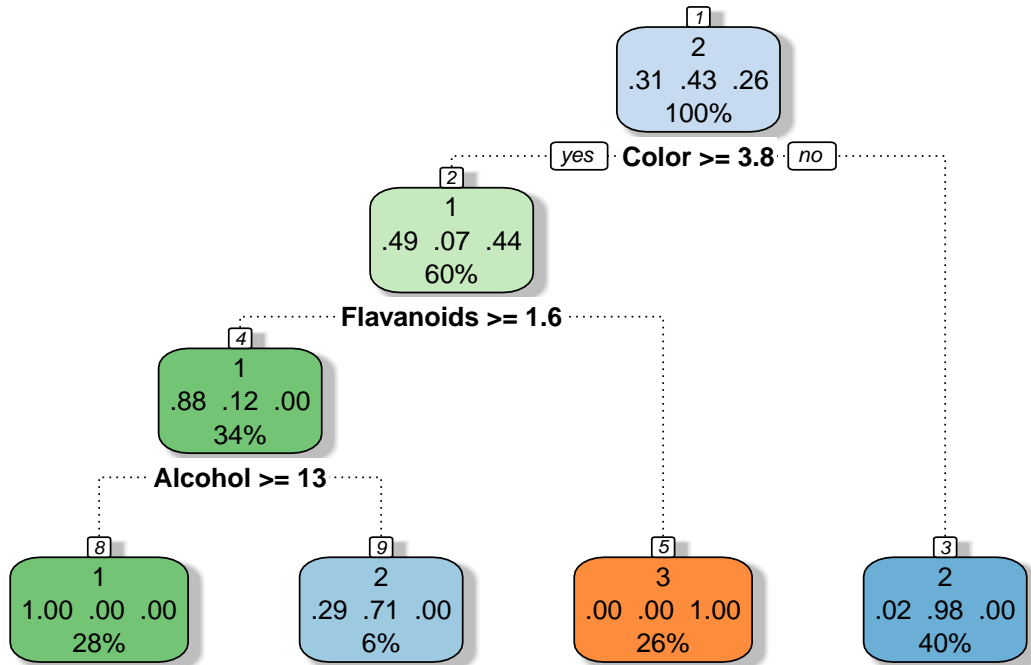
### 3.3 Drzewa klasyfikacyjne

```

basic.tree.model <- rpart(Type ~ ., data = train.data)

basic.tree.test.pred <- predict(basic.tree.model, newdata = test.data,
                                type = "class")
basic.tree.train.pred <- predict(basic.tree.model, newdata = train.data,
                                type = "class")

```



|   | 1  | 2  | 3  |
|---|----|----|----|
| 1 | 33 | 0  | 0  |
| 2 | 3  | 51 | 0  |
| 3 | 0  | 0  | 31 |

(a) Zbior uczacy

|   | 1  | 2  | 3  |
|---|----|----|----|
| 1 | 17 | 1  | 0  |
| 2 | 6  | 18 | 0  |
| 3 | 0  | 1  | 17 |

(b) Zbior testowy

Tabela 8: Macierze pomylek dla metody drzew klasyfikacyjnych — wszystkie cechy.

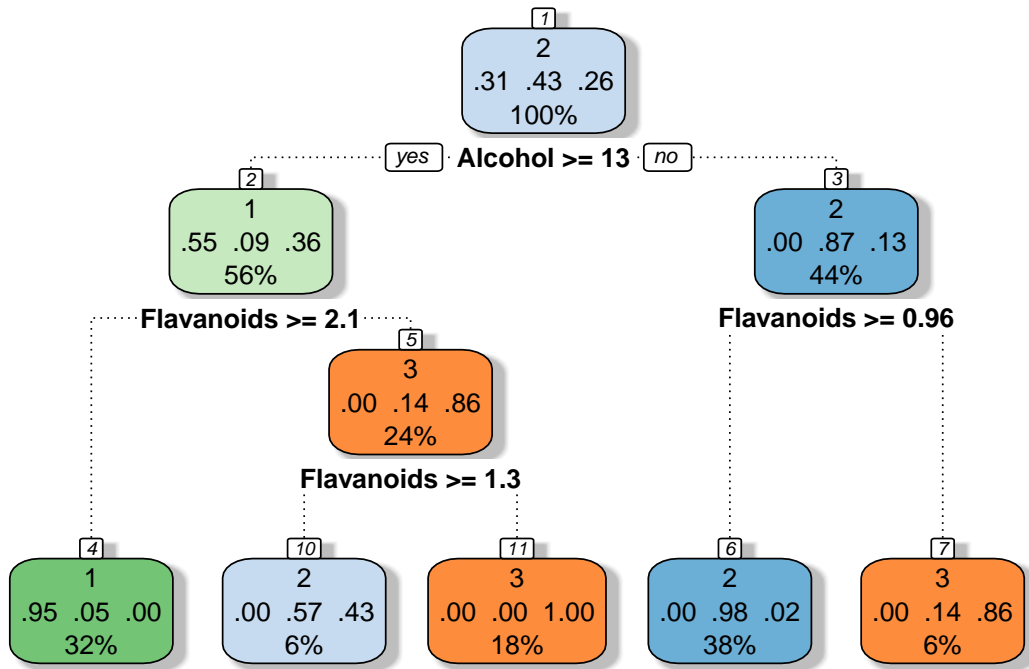
Błędy klasyfikacji to 0.0254237 i 0.1333333.

```

subset.tree.model <- rpart(Type ~ ., data = train.subset)

subset.tree.test.pred <- predict(subset.tree.model, newdata = test.subset,
                                type = "class")
subset.tree.train.pred <- predict(subset.tree.model, newdata = train.subset,
                                type = "class")

```



|   | 1  | 2  | 3  |
|---|----|----|----|
| 1 | 36 | 2  | 0  |
| 2 | 0  | 48 | 4  |
| 3 | 0  | 1  | 27 |

(a) Zbior uczacy

|   | 1  | 2  | 3  |
|---|----|----|----|
| 1 | 23 | 2  | 0  |
| 2 | 0  | 17 | 1  |
| 3 | 0  | 1  | 16 |

(b) Zbior testowy

Tabela 9: Macierze pomylek dla metody drzew klasyfikacyjnych — wybrane cechy.

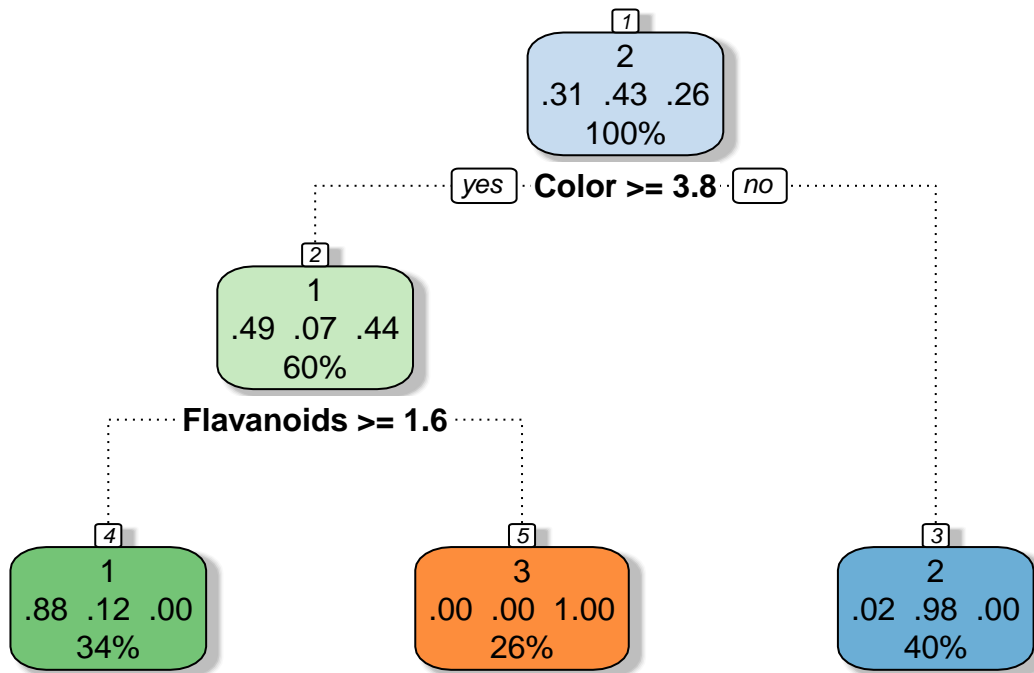
Błędy klasyfikacji to 0.059322 i 0.0666667.

```

tuned.tree.model <- train(Type ~ ., data = train.data, method = "rpart",
                           trControl = cv)

tuned.tree.test.pred <- predict(tuned.tree.model, test.data)
tuned.tree.train.pred <- predict(tuned.tree.model, train.data)

```



|   | 1  | 2  | 3  |   | 1  | 2  | 3  |
|---|----|----|----|---|----|----|----|
| 1 | 35 | 5  | 0  | 1 | 20 | 5  | 0  |
| 2 | 1  | 46 | 0  | 2 | 3  | 14 | 0  |
| 3 | 0  | 0  | 31 | 3 | 0  | 1  | 17 |

(a) Zbior uczacy                      (b) Zbior testowy

Tabela 10: Macierze pomyłek dla metody drzew klasyfikacyjnych — stuningowany model.

Błędy klasyfikacji to 0.0508475 i 0.15.

### 3.4 Naiwny klasyfikator bayesowski

```
bayes.model.basic <- naiveBayes(Type ~ ., data = train.data)

basic.bayes.train.pred <- predict(bayes.model.basic, train.data)
basic.bayes.test.pred <- predict(bayes.model.basic, test.data)
```

Błędy klasyfikacji to kolejno 0.0084746 i 0.0333333.

Powtórzmy teraz powyższe dla wybranego podzbioru naszych danych.

```
bayes.model.subset <- naiveBayes(Type ~ ., data = train.subset)
```

|                  | 1  | 2  | 3  |                   | 1  | 2  | 3  |
|------------------|----|----|----|-------------------|----|----|----|
| 1                | 36 | 0  | 0  | 1                 | 22 | 1  | 0  |
| 2                | 0  | 50 | 1  | 2                 | 0  | 19 | 1  |
| 3                | 0  | 0  | 31 | 3                 | 0  | 0  | 17 |
| (a) Zbior uczacy |    |    |    | (b) Zbior testowy |    |    |    |

Tabela 11: Macierze pomylek dla klasyfikatora bayesowskiego — wszystkie cechy.

```
basic.bayes.train.pred <- predict(bayes.model.subset, train.subset)
basic.bayes.test.pred <- predict(bayes.model.subset, test.subset)
```

|                  | 1  | 2  | 3  |                   | 1  | 2  | 3  |
|------------------|----|----|----|-------------------|----|----|----|
| 1                | 34 | 2  | 0  | 1                 | 20 | 3  | 0  |
| 2                | 2  | 47 | 2  | 2                 | 2  | 17 | 1  |
| 3                | 0  | 1  | 30 | 3                 | 0  | 0  | 17 |
| (a) Zbior uczacy |    |    |    | (b) Zbior testowy |    |    |    |

Tabela 12: Macierze pomylek dla klasyfikatora bayesowskiego — wybrane cechy.

Błędy klasyfikacji to kolejno 0.059322 i 0.1.

Model “stuningowany”

```
model <- train(Type ~ ., data = train.data, method = "naive_bayes",
               trControl = cv)
```

|                  | 1  | 2  | 3  |                   | 1  | 2  | 3  |
|------------------|----|----|----|-------------------|----|----|----|
| 1                | 36 | 0  | 0  | 1                 | 22 | 1  | 0  |
| 2                | 0  | 50 | 1  | 2                 | 0  | 19 | 1  |
| 3                | 0  | 0  | 31 | 3                 | 0  | 0  | 17 |
| (a) Zbior uczacy |    |    |    | (b) Zbior testowy |    |    |    |

Tabela 13: Macierze pomylek dla klasyfikatora bayesowskiego — model stuningowany.

Błędy klasyfikacji w tym przypadku to kolejno 0.0084746 i 0.0333333.

Powtórzymy teraz ocenę klasyfikacji przy pomocy metody 5-krotnej walidacji krzyżowej dla stuningowanego klasyfikatora bayesowskiego, który najlepiej poradził sobie na danym zbiorze treningowym i testowym.

```
predictor <- function(model, newdata)
{ predict(model, newdata=newdata) }
```

```

cv <- trainControl(method="cv", number=5)

naiveBayes.predictor <- function(formula, data)
{ train(formula, data = data, method = "naive_bayes", trControl = cv)}

error.cv <- errorest(Type~., wine, model=naiveBayes.predictor, predict=predictor,
  estimator="cv", est.param=control.errorest(k = 5))

error.cv

##
## Call:
## errorest.data.frame(formula = Type ~ ., data = wine, model = naiveBayes.predictor,
##   predict = predictor, estimator = "cv", est.param = control.errorest(k = 5))
##
##   5-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.0225

```