

Raport 4

Eksploracja danych

Mikołaj Langner, Marcin Kostrzewa
nr albumów: 255716, 255749

2021-05-28

Spis treści

1	Wstęp	1
2	Zadanie 1	2
2.1	a)	2
2.2	b)	6
3	Zadanie 2	11

1 Wstęp

Niniejszy raport zawiera rozwiązania zadań z listy 4.

W zadaniu pierwszym zastosujemy zaawansowane metody klasyfikacji:

- bagging,
- boosting,
- random forest,
- metodę wektorów nośnych (SVM),

W zadaniu drugim badamy jakość

2 Zadanie 1

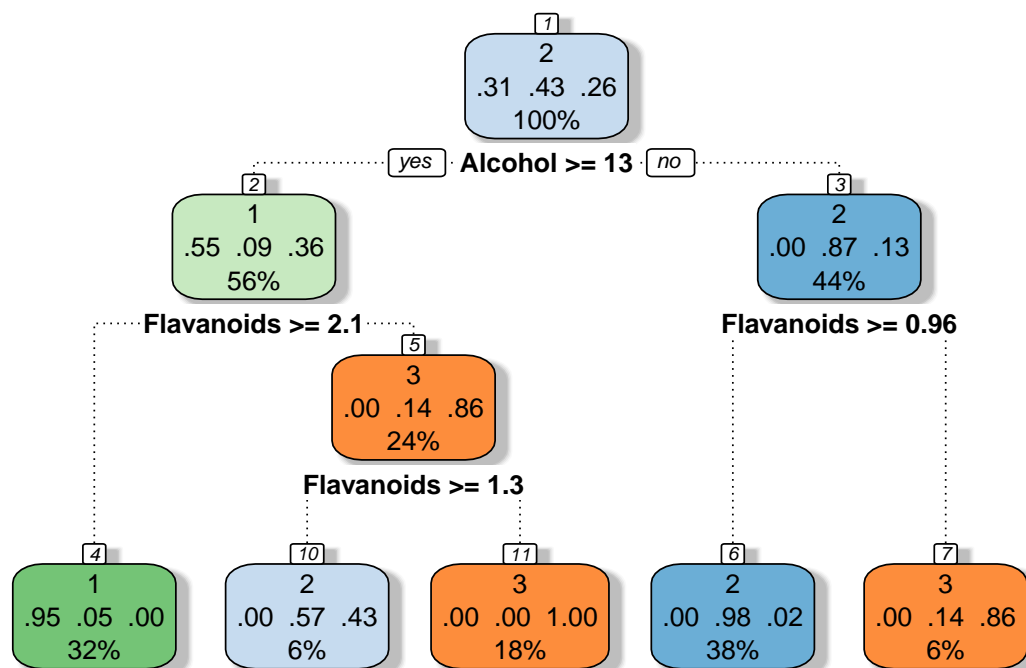
2.1 a)

2.1.1 Pojedyncze drzewo decyzyjne

Przypomnijmy najpierw jak radziła sobie metoda drzewa klasyfikacyjnego.

```
tree.model <- rpart(Type ~ ., data = train.subset, cp=0)
```

Wyglądało ono następująco — rysunek (??).



Rysunek 1: Pojedyncze drzewo decyzyjne.

Przypomnijmy także jak wyglądały błędy klasyfikacji dla drzewa.

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

decision.tree.predictor <- function(formula, data)
{ rpart(formula, data = data, cp = 0) }

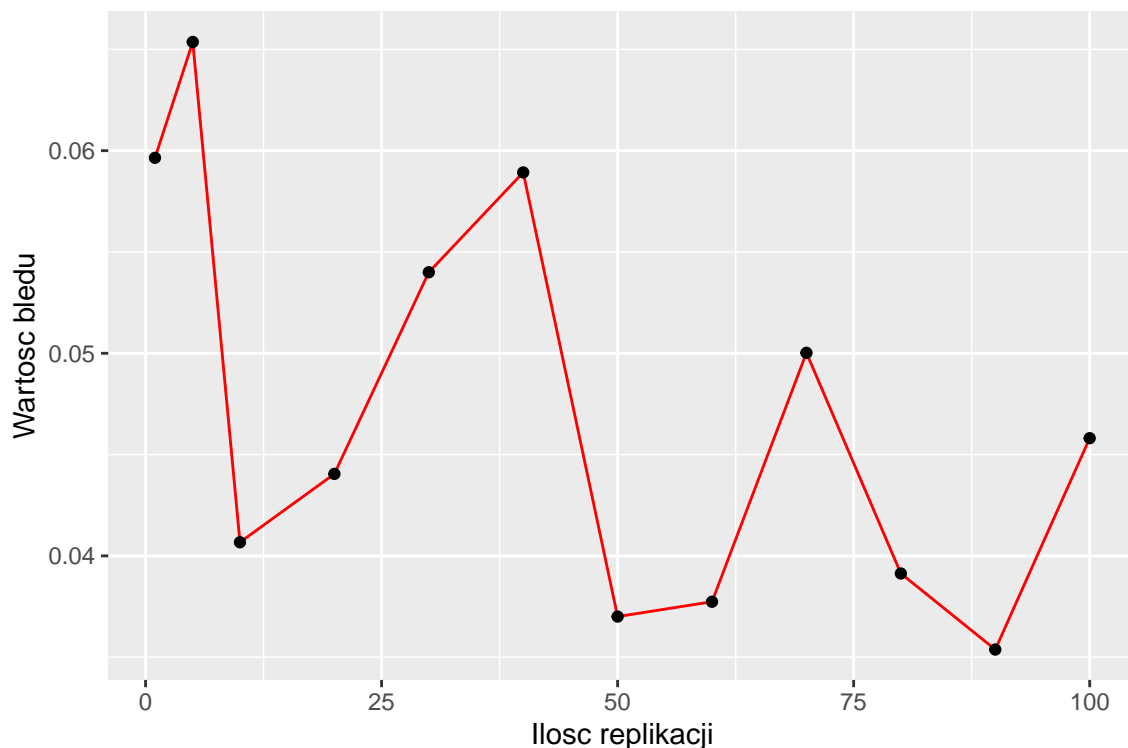
decision.tree.error.cv <- errorest(Type~., wine[, c(1, 2, 8)],
                                   model=decision.tree.predictor,
                                   predict=predictor, estimator="cv",
                                   est.para=control.errorest(k = 5))
```

Wyniósł on 0.1179775.

2.1.2 Bagging

Najpierw skorzystamy z algorytmu bagging. Znajdziemy optymalną wartość dla parametru nbagg.

```
B.vector <- c(1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
bagging.error.rates <- sapply(B.vector, function(b)
  {errorest(Type~., data=train.data, model=bagging,
    nbagg=b, estimator="632plus",
    est.para=control.errorest(nboot = 20))$error})
choice <- B.vector[which.min(bagging.error.rates)]
```



Rysunek 2: Wpływ ilości replikacji na błąd klasyfikacji.

Jak widać, najlepiej zbudować model dla nbagg równego 90.

```
bagging.model <- bagging(Type~., data=train.data, nbagg=choice,
  minsplit=1, cp=0)
bagging.train.pred <- predict(bagging.model, train.data)
bagging.test.pred <- predict(bagging.model, test.data)
```

Wyznamy dla tego modelu macierze pomyłek i wartości błędów klasyfikacji.

Błędy klasyfikacji to kolejno 0 i 0.05.

	1	2	3		1	2	3
1	36	0	0	1	21	0	0
2	0	51	0	2	2	19	0
3	0	0	31	3	0	1	17

(a) Zbiór uczący (b) Zbiór testowy

Tabela 1: Macierze pomyłek dla algorytmu bagging.

Wyznamy teraz dla tego modelu klasyfikacyjnego błąd predykcji — skorzystamy z 5-krotnej walidacji krzyżowej, metody bootstrap oraz .632+.

```

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

bagging.predictor <- function(formula, data)
{bagging(formula, data = data, nbagg = choice, cp = 0)}

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

bagging.error.boot <- errorest(Type~., wine,
                              model=bagging.predictor,
                              predict=predictor, estimator="boot",
                              est.param=control.errorest(nboot = 25))

bagging.error.632 <- errorest(Type~., wine,
                              model=bagging.predictor,
                              predict=predictor, estimator="632plus",
                              est.param=control.errorest(nboot = 25))

```

Błędy wyniosły kolejno 0.0561798, 0.0584677 oraz 0.0402642.

2.1.3 Boosting

2.1.4 Random Forest

Teraz wykorzystamy algorytm random forest.

Postaramy się odpowiednio dobrać parametry `ntree` (ilość drzew) i `mtry` (ilość losowo wybieranych cech).

```

ntree.vector <- seq(10, 500, by=20)
ntree.error.rates <- sapply(ntree.vector, function(b)
  {errorest(Type~., data=train.data, model=randomForest,
            ntree=b, estimator="632plus",
            est.param=control.errorest(nboot = 20))$error})

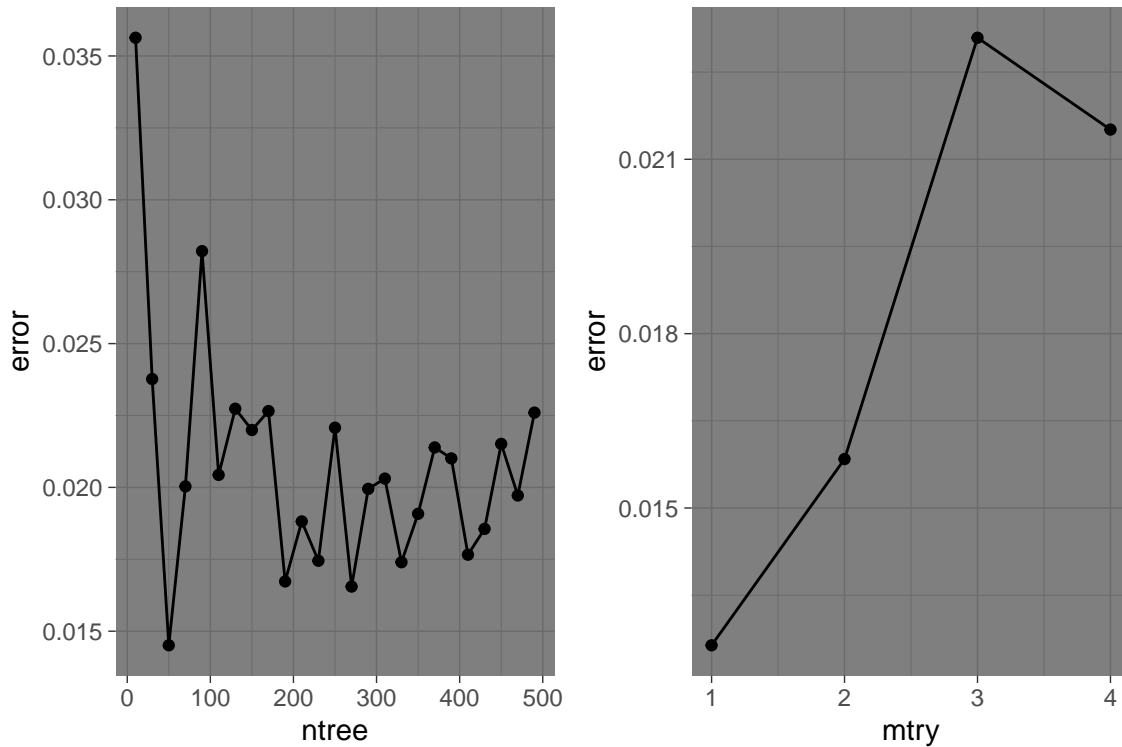
```

```

ntree.choice <- ntree.vector[which.min(ntree.error.rates)]

mtry.vector <- seq(1, sqrt(ncol(wine))+1, by=1)
mtry.error.rates <- sapply(mtry.vector, function(m)
  {errorest(Type~., data=train.data, model=randomForest, ntree = ntree.choice,
    mtry=m, estimator="632plus",
    est.param=control.errorest(nboot = 20))$error})
mtry.choice <- mtry.vector[which.min(mtry.error.rates)]

```



Rysunek 3: Wykresy zależności błędów klasyfikacji od parametrów mtry i ntree.

Podobnie jak wcześniej wyznaczamy za pomocą modelu etykiety klas i wyznaczamy macierze pomyłek i błędów klasyfikacji.

```

rf.model <- randomForest(Type ~., data = train.data, ntree=ntree.choice,
  mtry=mtry.choice, importance=TRUE)
rf.test.pred <- predict(rf.model, test.data)
rf.train.pred <- predict(rf.model, train.data)

```

Błędy klasyfikacji to kolejno 0 i 0.

Tak jak wcześniej wyznaczymy dla tego modelu błędy predykcji.

```

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

```

	1	2	3		1	2	3
1	36	0	0	1	23	0	0
2	0	51	0	2	0	20	0
3	0	0	31	3	0	0	17

(a) Zbiór uczący (b) Zbiór testowy

Tabela 2: Macierze pomyłek dla algorytmu randomForest.

```
rf.predictor <- function(formula, data)
{ randomForest(formula, data = data, ntree = ntree.choice, mtry = mtry.choice)}

rf.error.cv <- errorest(Type~., wine, model=rf.predictor,
                        predict=predictor, estimator="cv",
                        est.param=control.errorest(k = 5))
rf.error.boot <- errorest(Type~., wine, model=rf.predictor,
                          predict=predictor, estimator="boot",
                          est.param=control.errorest(nboot = 25))
rf.error.632 <- errorest(Type~., wine, model=rf.predictor,
                        predict=predictor, estimator="632plus",
                        est.param=control.errorest(nboot = 25))
```

Błędy wyniosły kolejno 0.005618, 0.0227064 oraz 0.0125766.

Wykorzystamy teraz algorytm random forest do wyznaczenia rankingu cech (*variable importance*).

Widzimy, że ...

2.1.5 Wnioski

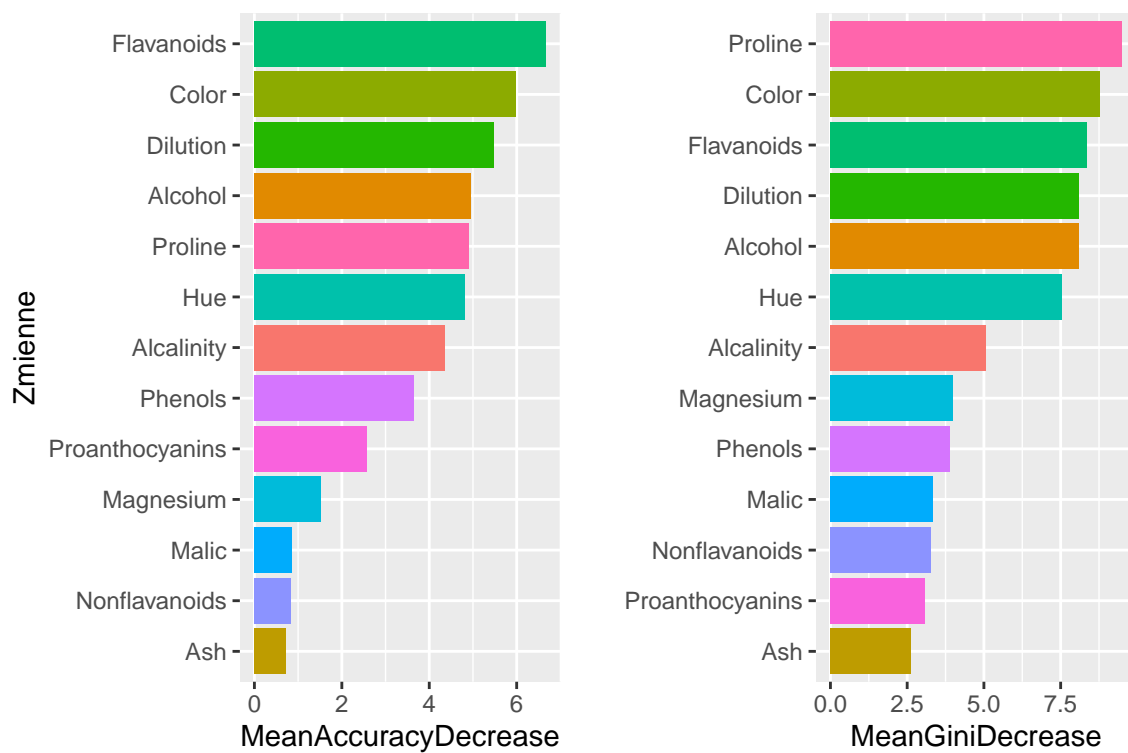
2.2 b)

```
wine <- wine %>% select(c(Type, Alcohol, Flavanoids))
```

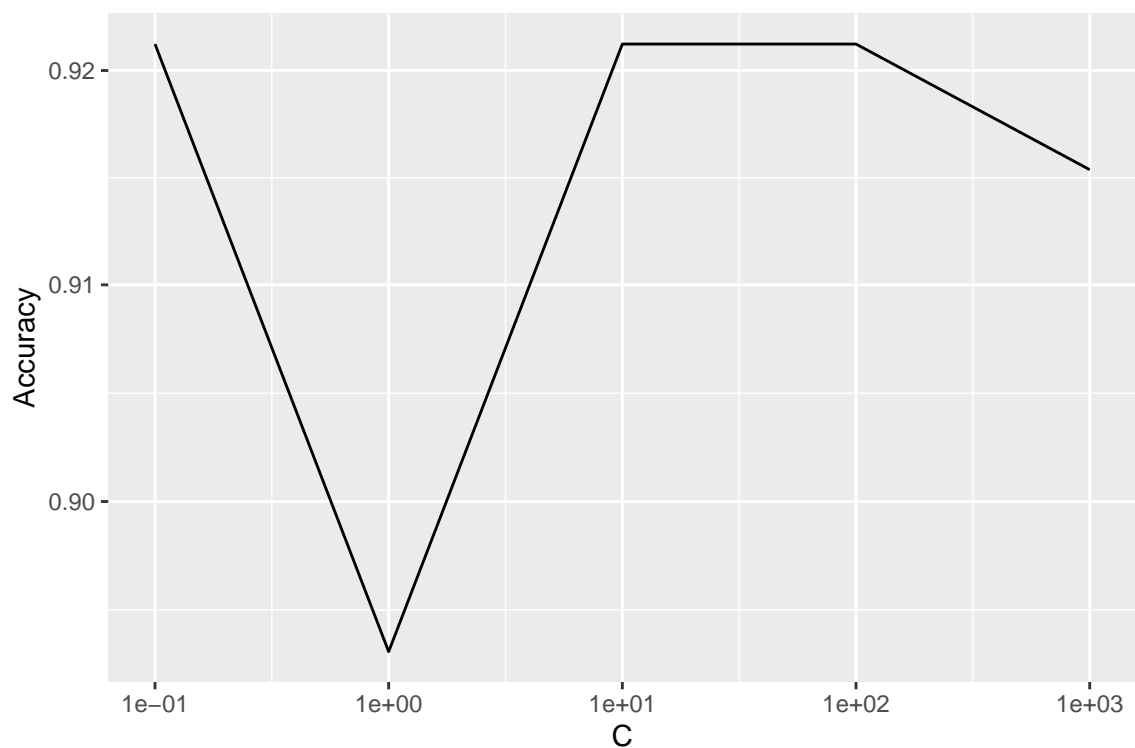
```
## Setting default kernel parameters
## Setting default kernel parameters
## Setting default kernel parameters
## Setting default kernel parameters
## Setting default kernel parameters
```

% latex table generated in R 4.1.0 by xtable 1.8-4 package % Sat Jun 19 15:24:45 2021

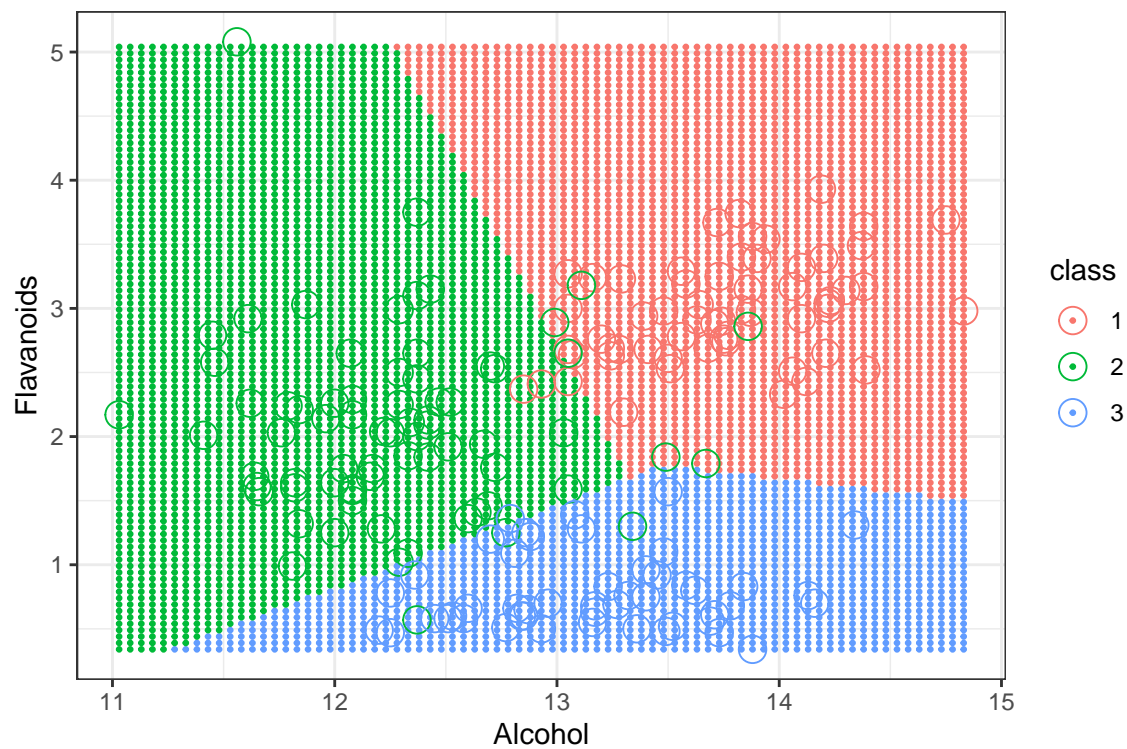
% latex table generated in R 4.1.0 by xtable 1.8-4 package % Sat Jun 19 15:24:48 2021



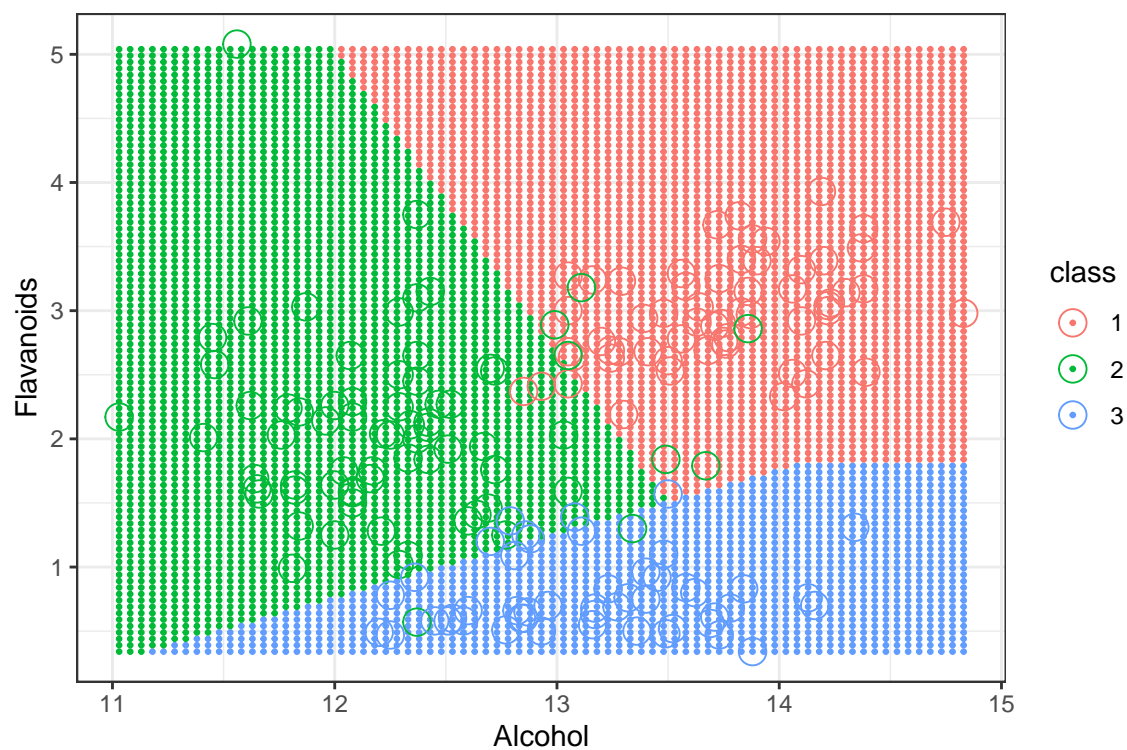
Rysunek 4: Wykres waznosci zmiennych.



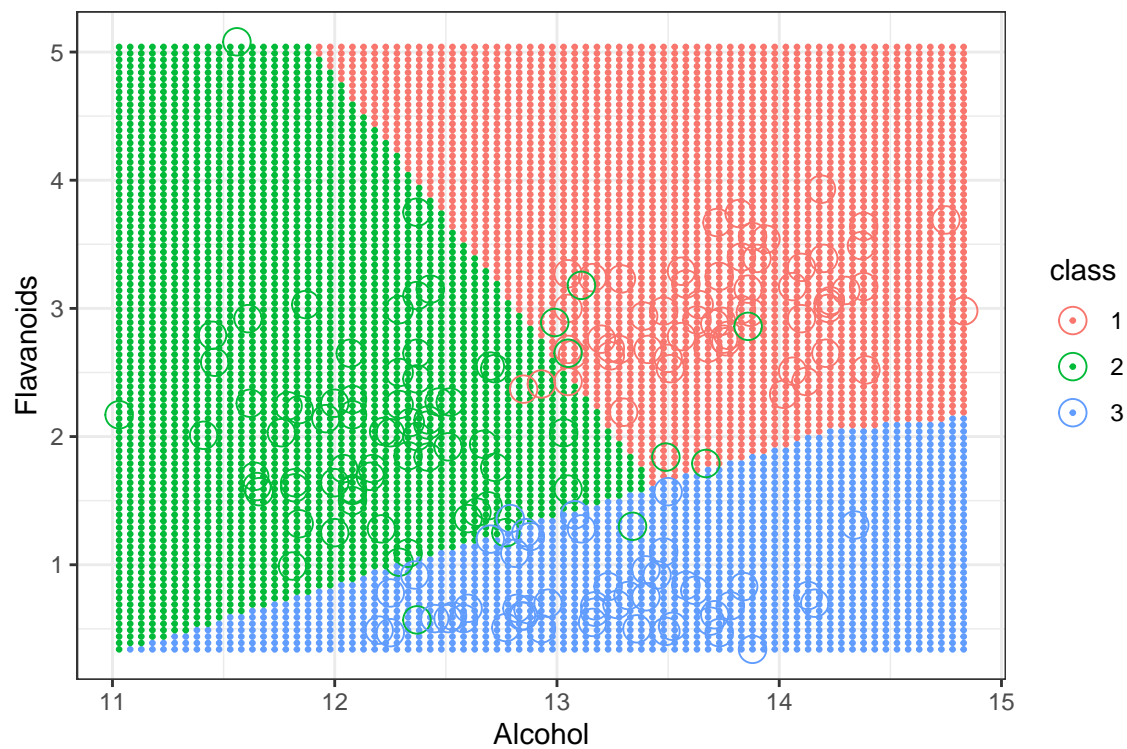
Rysunek 5: Dokładność klasyfikatora od parametru kosztu



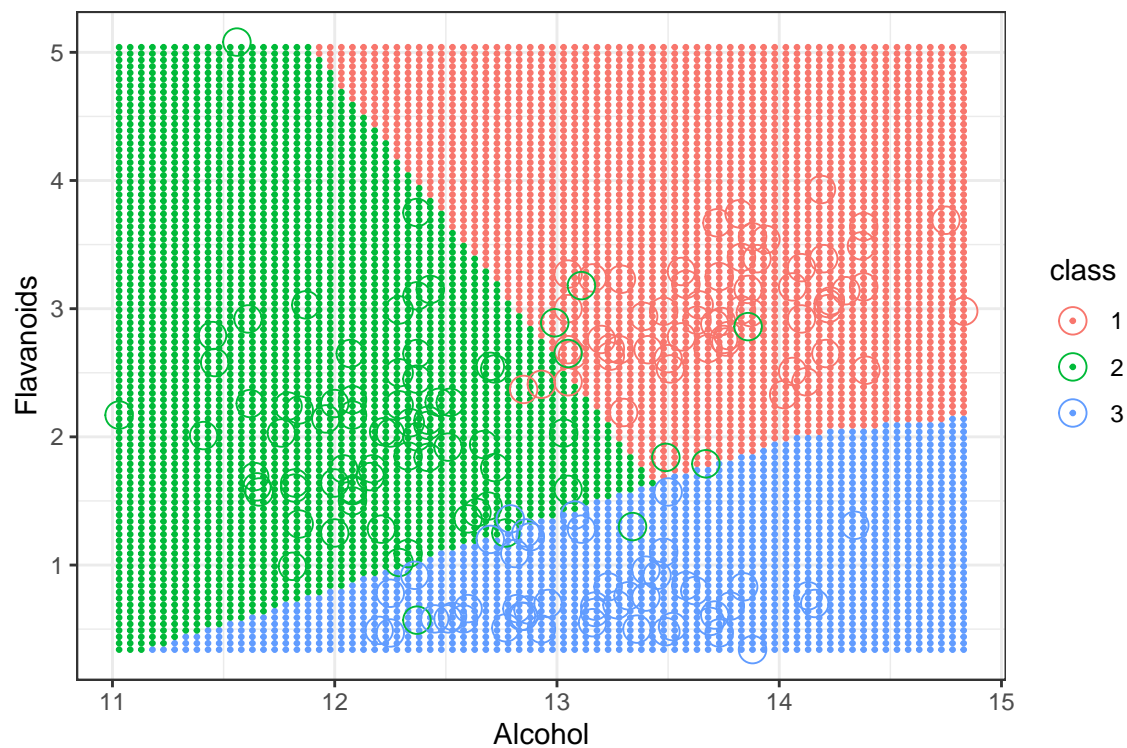
Rysunek 6: Obszary decyzyjne dla $C = 0.1$



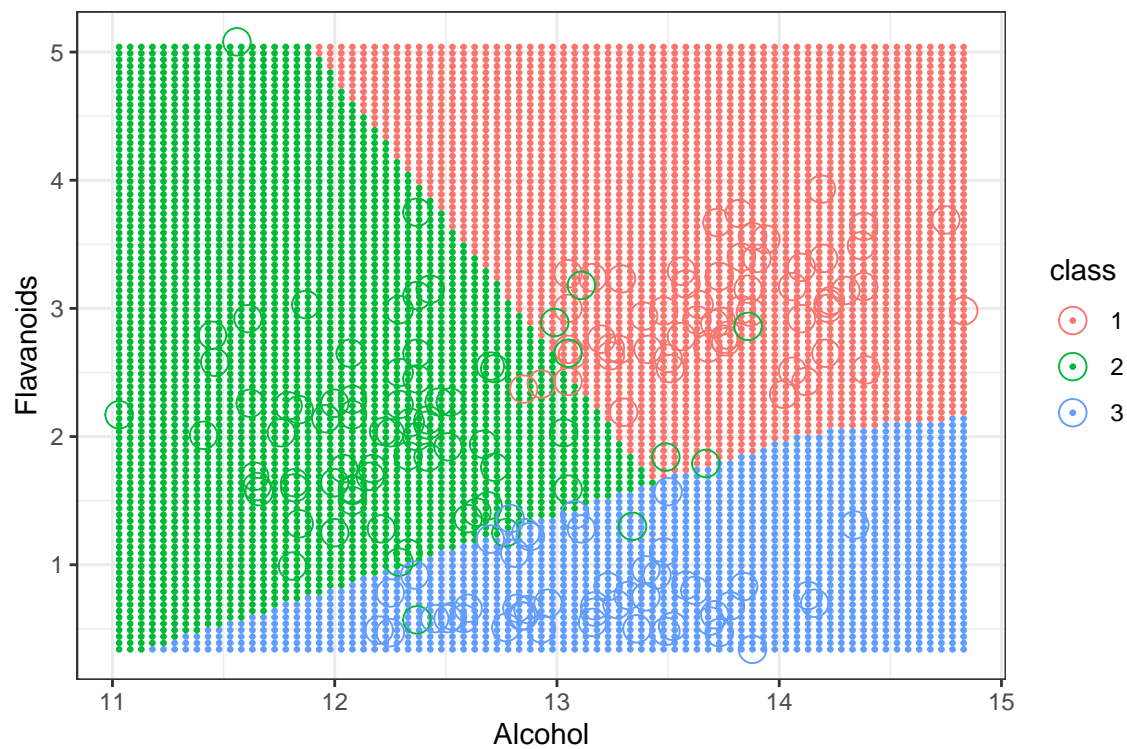
Rysunek 7: Obszary decyzyjne dla $C = 1$



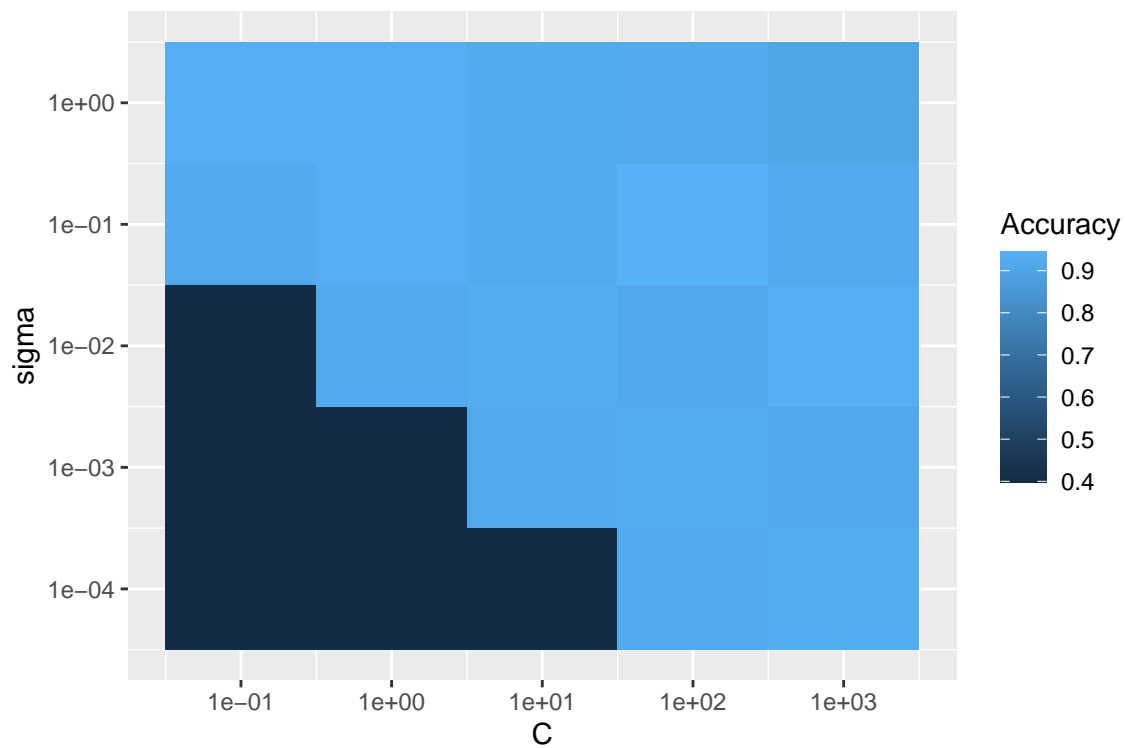
Rysunek 8: Obszary decyzyjne dla $C = 10$



Rysunek 9: Obszary decyzyjne dla $C = 100$



Rysunek 10: Obszary decyzyjne dla $C = 1000$



Rysunek 11: Mapa ciepła dokładności klasyfikatora

linear	polynomial	radial
0.928	0.933	0.938

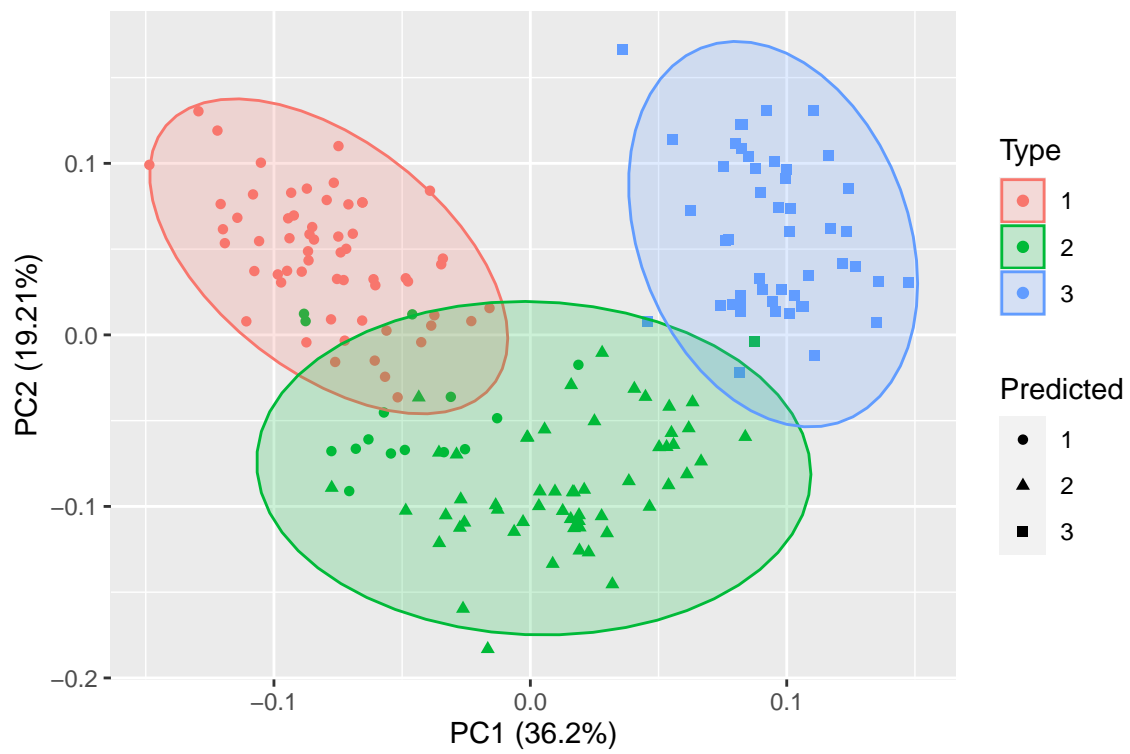
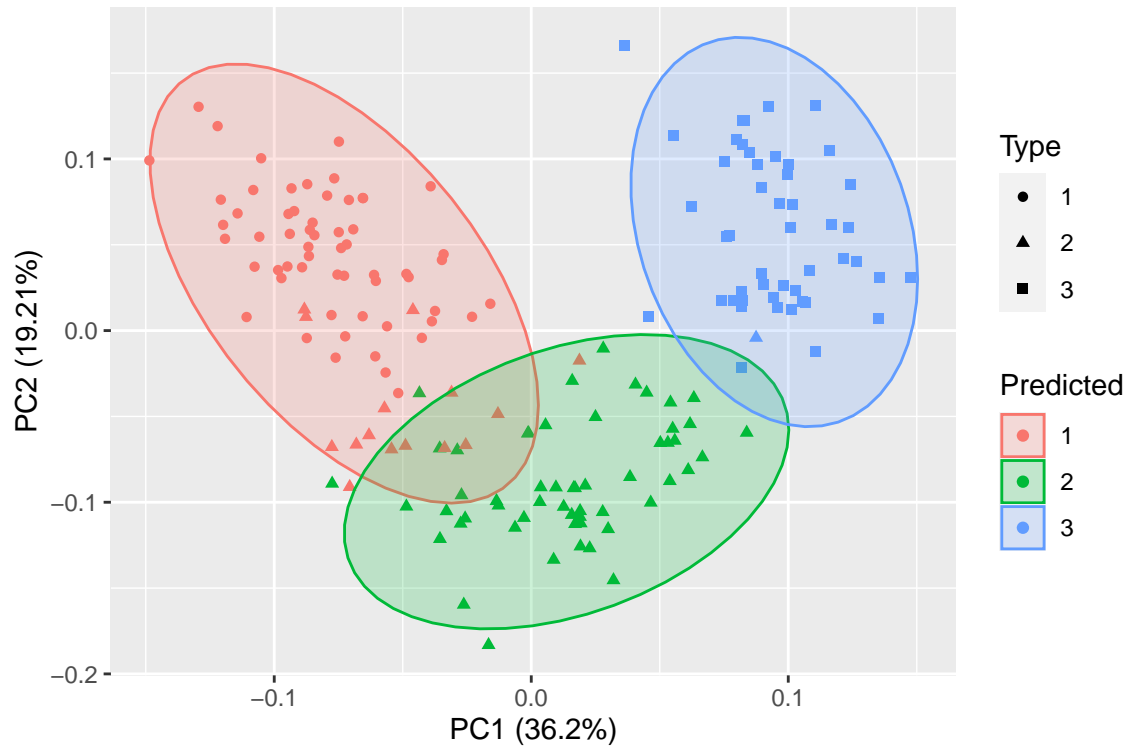
Tabela 3: Porównanie klasyfikatorów dla różnych jąder

sigma	C
0.10	100.00

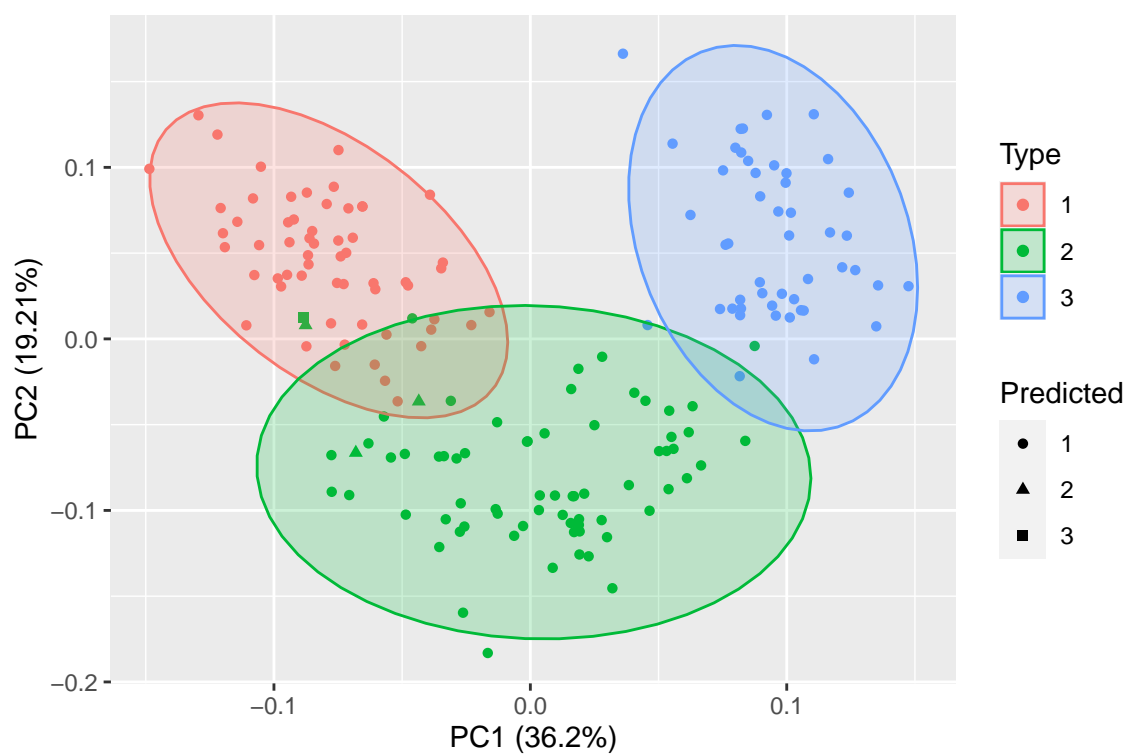
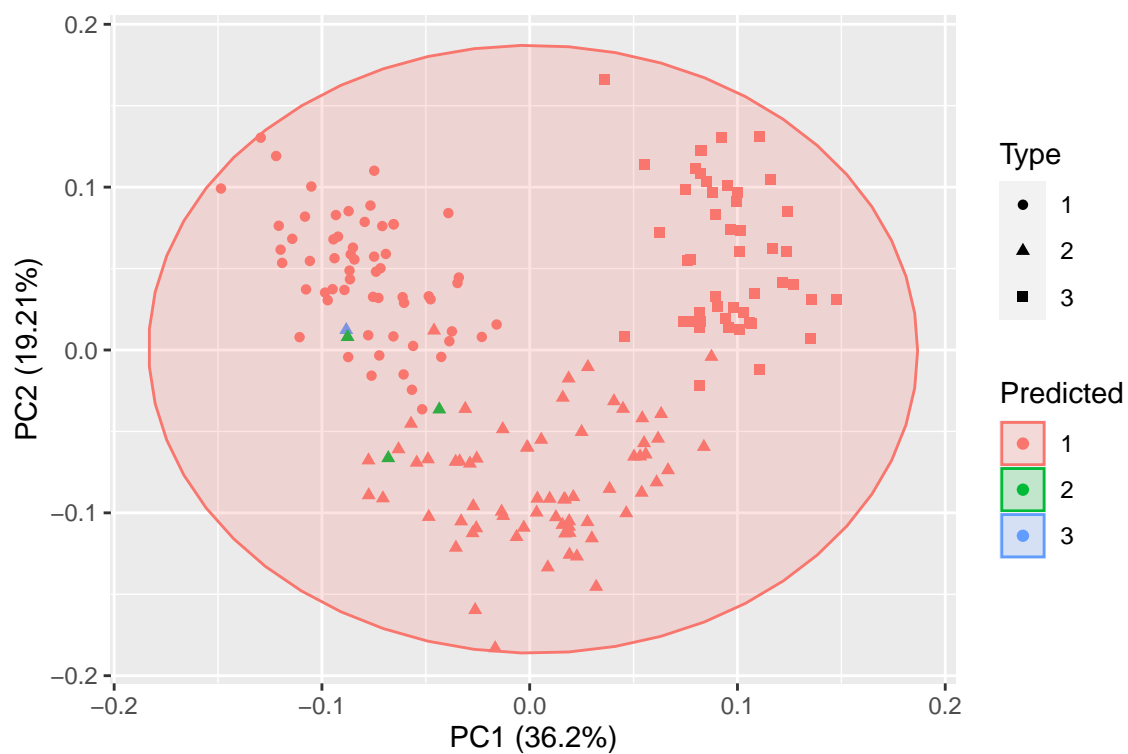
Tabela 4: Parametry dla najlepszego klasyfikatora

3 Zadanie 2

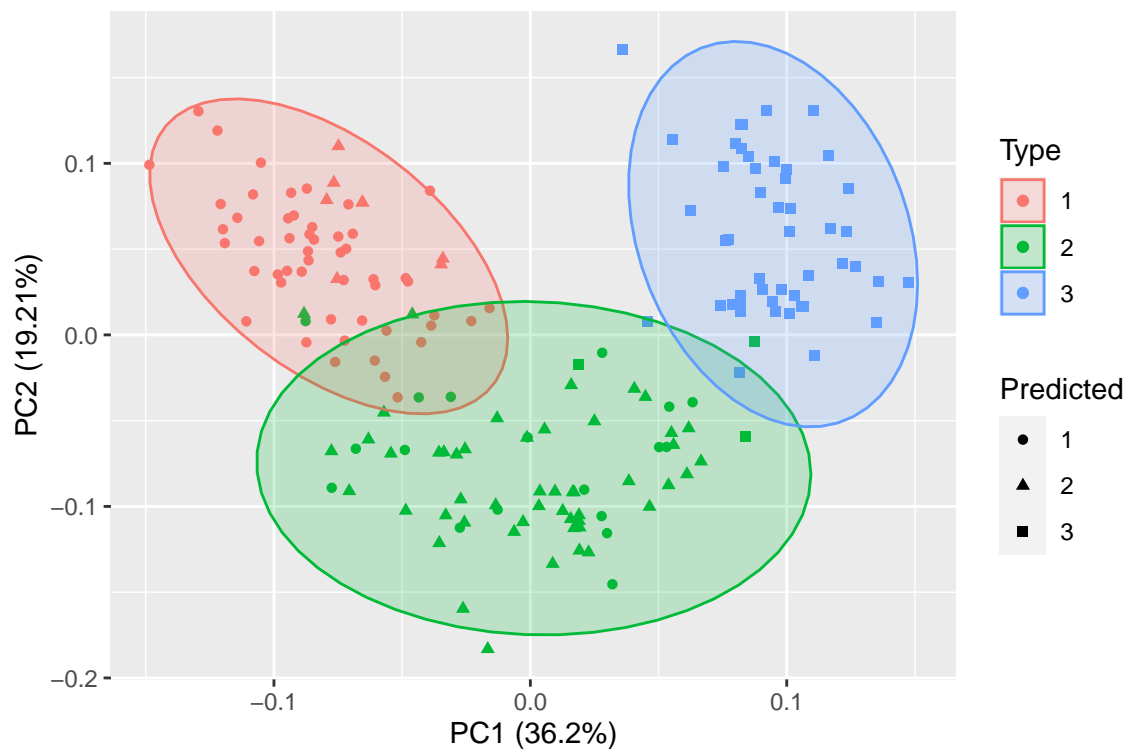
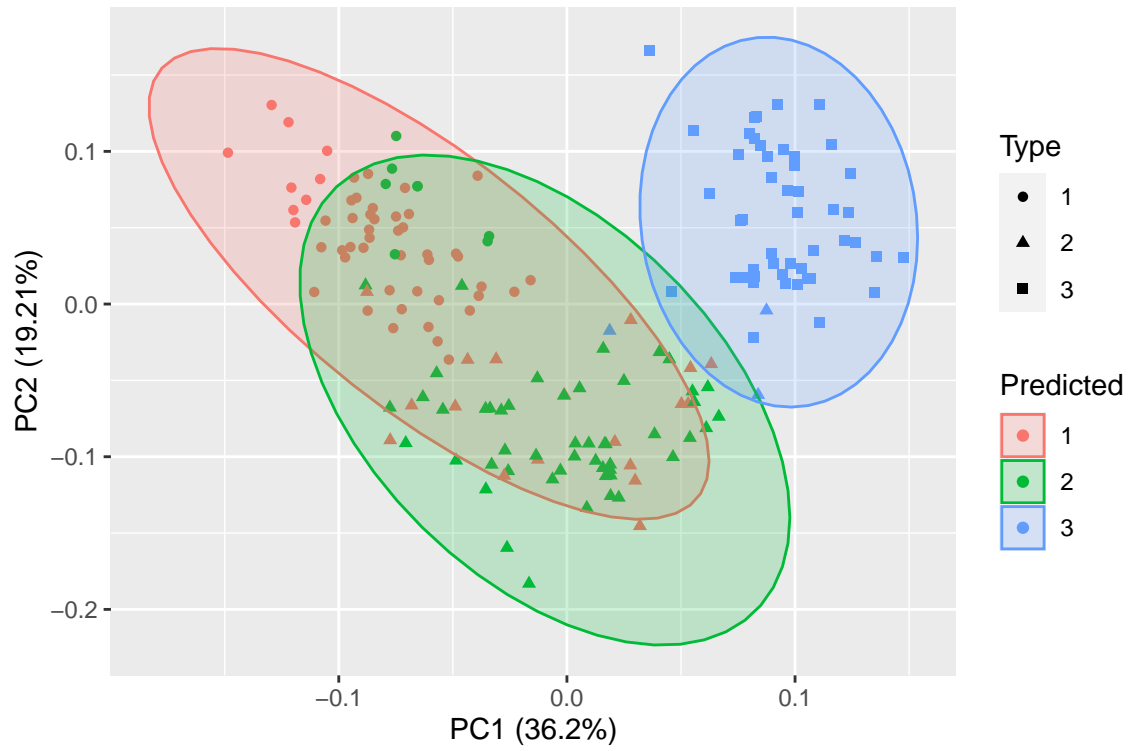
```
##
## Clustering Methods:
## agnes pam
##
## Cluster sizes:
## 2 3 4 5 6 7 8 9 10
##
## Validation Measures:
##          2          3          4          5          6          7          8          9
##
## agnes Connectivity 4.4925 8.0972 12.8210 17.7913 21.4591 22.9877 25.8044 30.6730 3
##      Dunn         0.0374 0.0227 0.0417 0.0347 0.0368 0.0544 0.0561 0.0656
##      Silhouette    0.6413 0.5419 0.5336 0.4806 0.4824 0.5075 0.5055 0.5024
## pam  Connectivity 1.5286 5.1048 16.2798 20.0643 23.1155 27.8393 31.0163 33.5841 3
##      Dunn         0.0434 0.0229 0.0340 0.0340 0.0233 0.0502 0.0478 0.0359
##      Silhouette    0.6494 0.5708 0.5620 0.5469 0.5414 0.5622 0.5401 0.5353
##
## Optimal Scores:
##
##          Score Method Clusters
## Connectivity 1.5286 pam      2
## Dunn         0.0776 agnes   10
## Silhouette   0.6494 pam      2
##
##          Score Method Clusters
## Connectivity 1.52857143    pam      2
## Dunn         0.07755693    agnes    10
## Silhouette   0.64936476    pam      2
##
## Using cluster as id variables
## Using cluster as id variables
## Using cluster as id variables
## Cases in matched pairs: 80.9 %
```



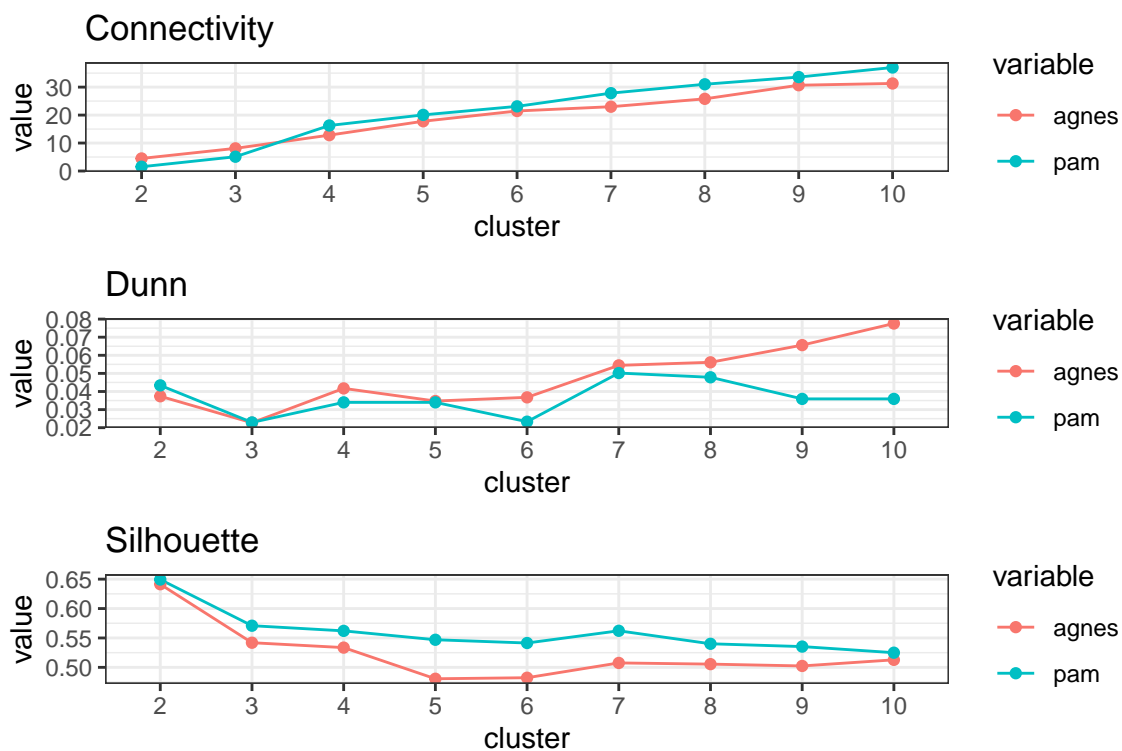
Rysunek 12: Skupienia dla metody PAM



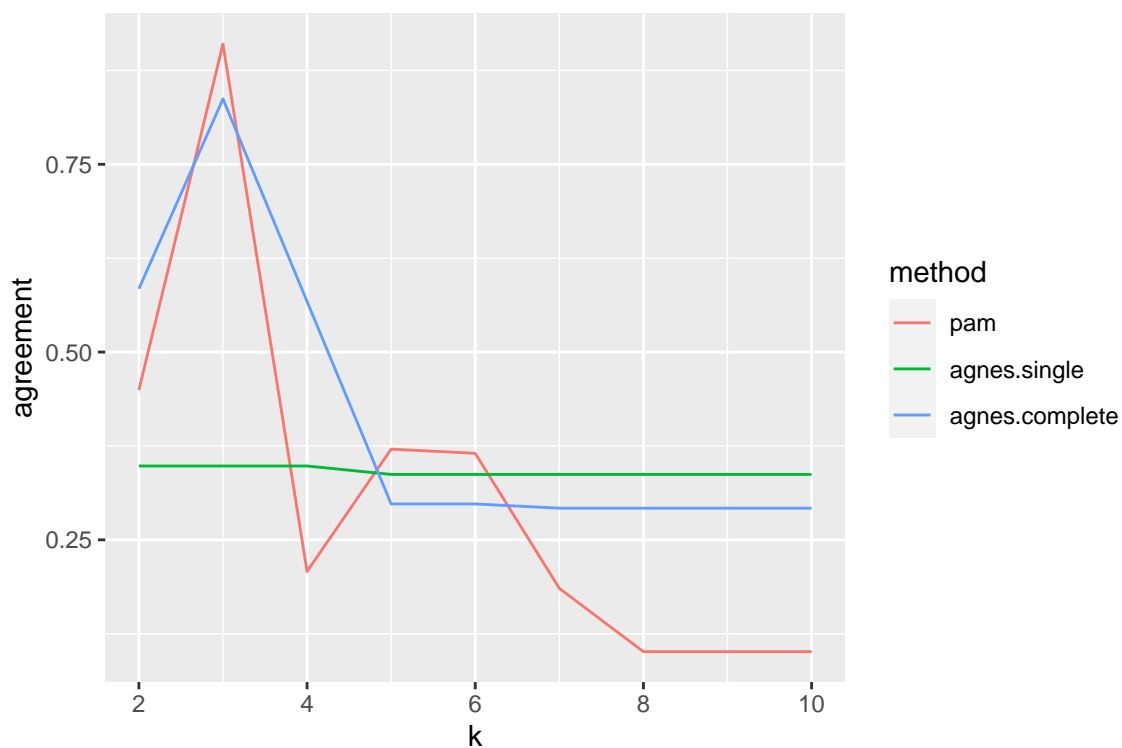
Rysunek 13: Skupienia dla metody AGNES z single-linkage



Rysunek 14: Skupienia dla metody AGNES z complete-linkage

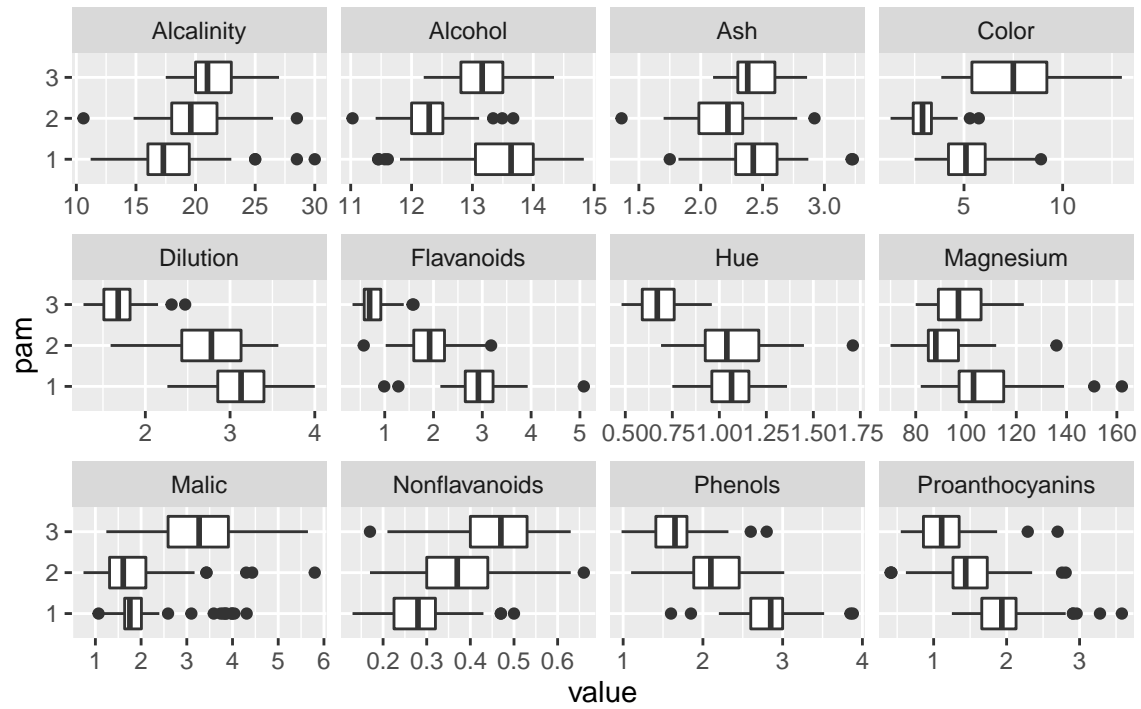


Rysunek 15: Wskaźniki wewnętrzne dla PAM i AGNES z complete-linkage

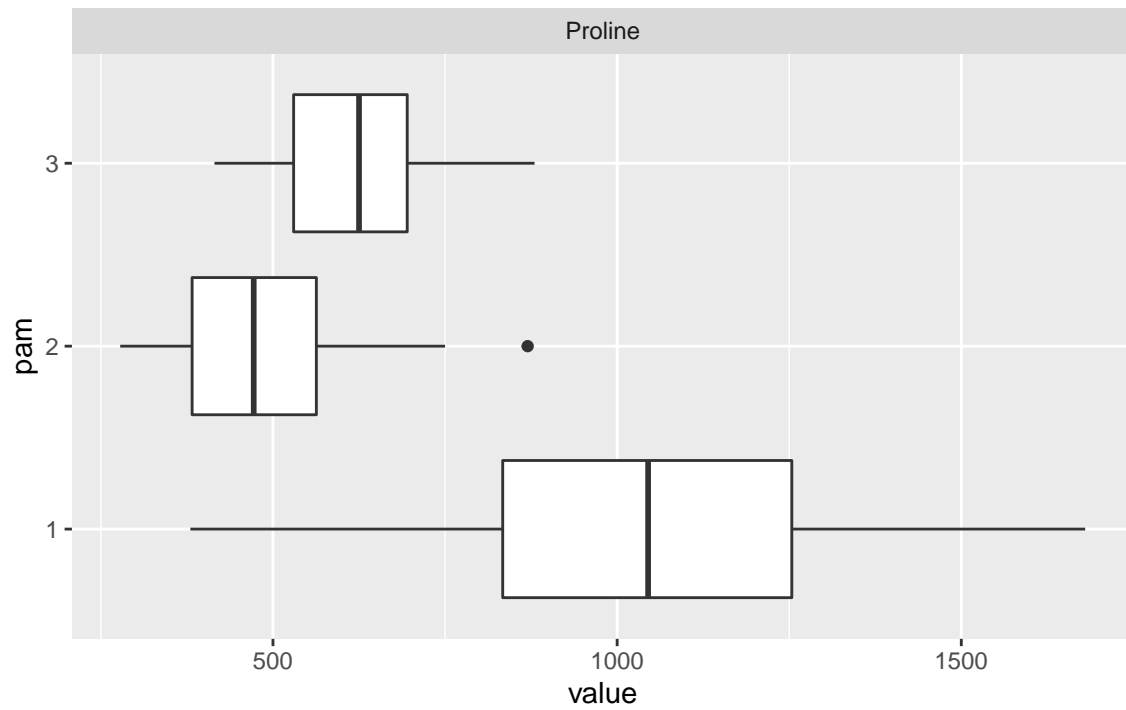


Rysunek 16: Porównanie wskaźników zewnętrznych

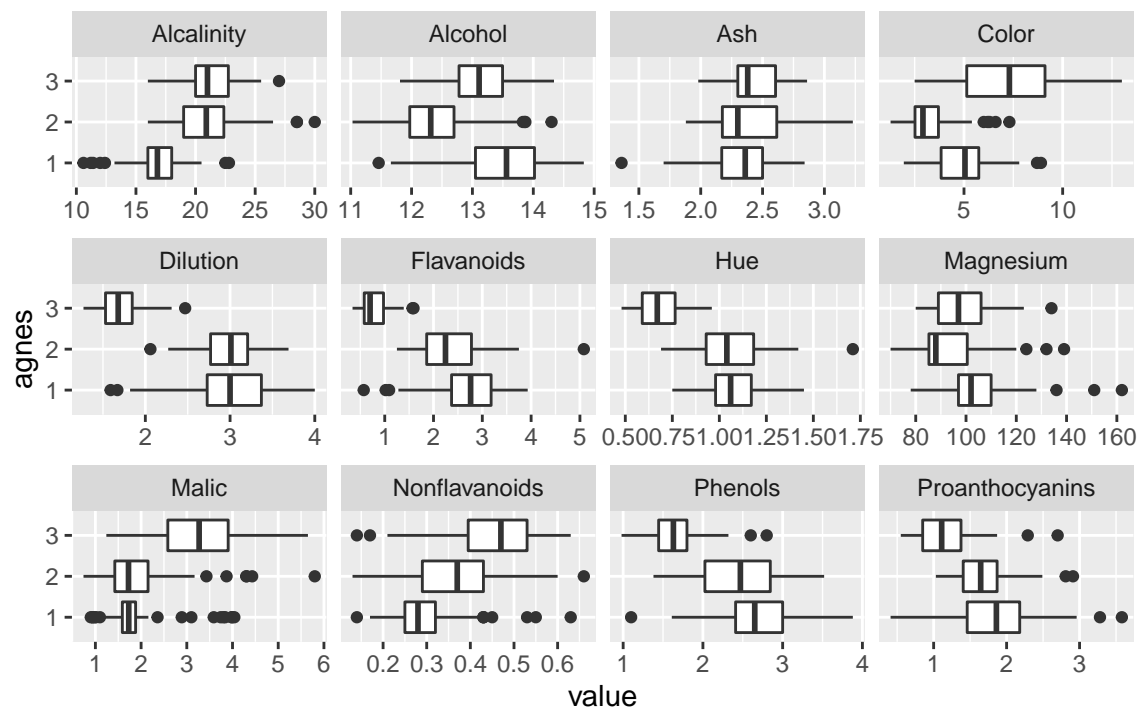
1 2 3
1 2 3



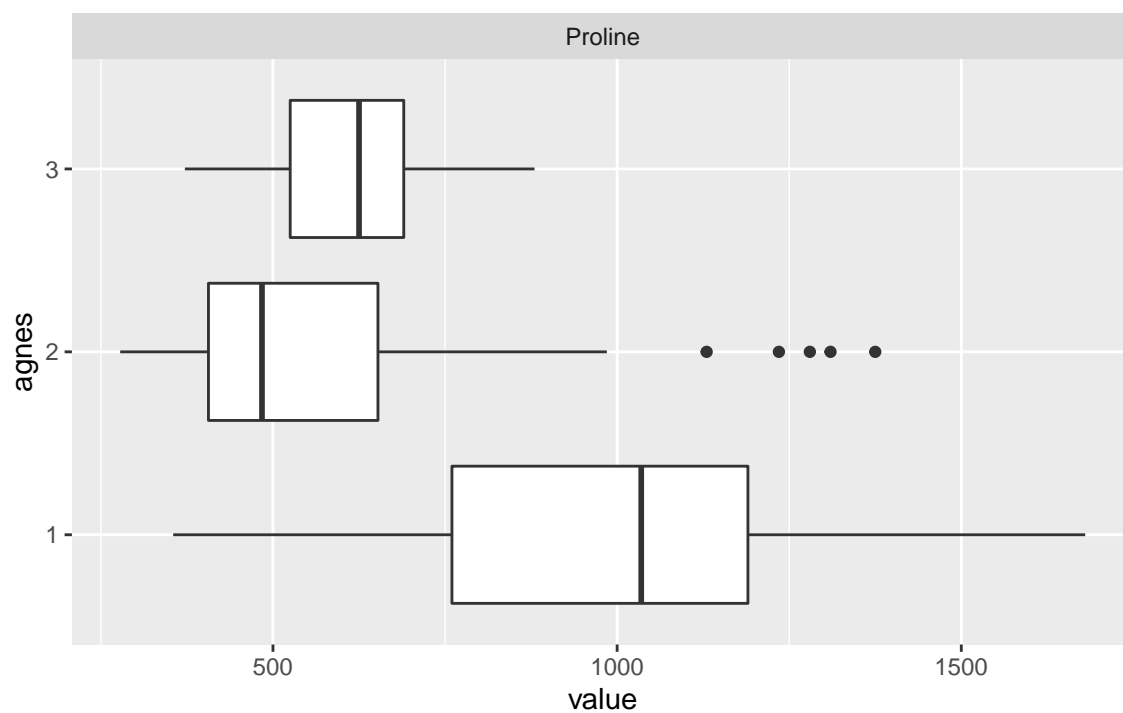
Page 1



Page 2



Page 1



Page 2

% latex table generated in R 4.1.0 by xtable 1.8-4 package % Sat Jun 19 15:24:53 2021

	1	2	3
Alcohol	0.59	-0.92	0.39
Malic	-0.47	-0.54	0.81
Ash	0.16	-0.90	0.05
Alcalinity	0.30	-0.15	0.60
Magnesium	0.02	-1.38	-0.54
Phenols	0.65	-1.03	-0.58
Flavanoids	0.95	0.00	-1.27
Nonflavanoids	-0.82	0.07	0.71
Proanthocyanins	0.47	0.07	-0.60
Color	0.02	-0.72	1.45
Hue	0.36	0.19	-1.78
Dilution	1.21	0.79	-1.40
Proline	0.55	-0.75	-0.31

Tabela 5: Medoidy dla metody PAM przy K=3