

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)	9
3	Zadanie 2	13
3.1	Wizualizacja wyników grupowania ($K = 3$)	13
3.2	Ocena jakości grupowania	15

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),

do zbioru `wine` i ocenimy jakość ich działania.

W zadaniu drugim przeprowadzimy analizę skupień dla tego samego zbioru, używając metod hierarchicznych (AGNES) i grupujących (PAM).

2 Zadanie 1

2.1 a)

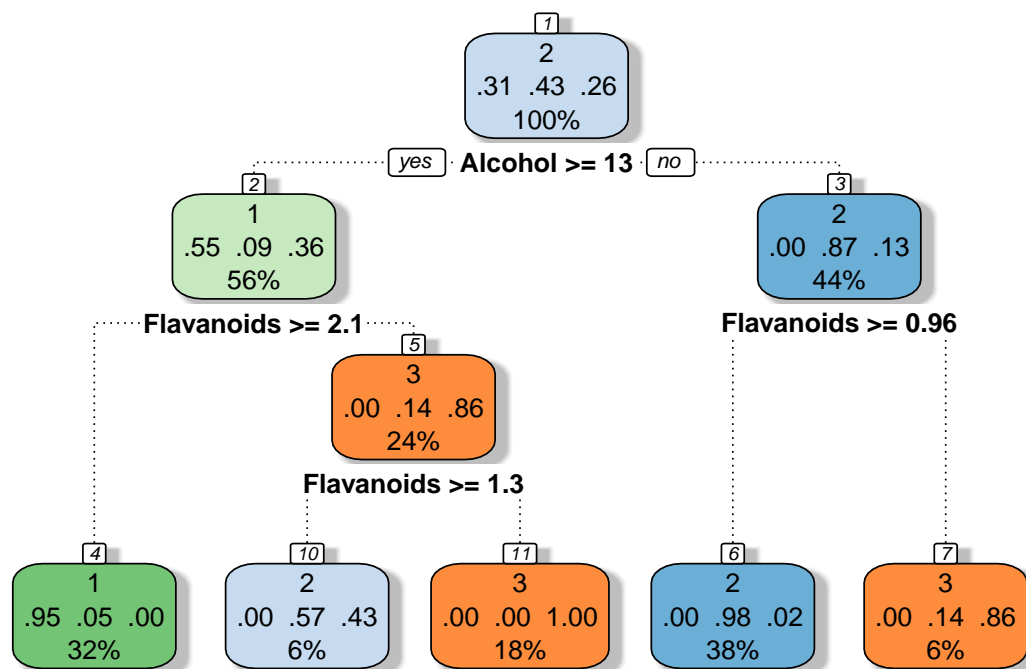
Naszym zadaniem będzie zbadanie tego jak poprawia się jakość klasyfikacji, jeżeli zamiast pojedynczego drzewa decyzyjnego, użyjemy złożonego klasyfikatora.

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 (1).



Rysunek 1: Pojedyncze drzewo decyzyjne.

Błędy predykcji wyznaczone za pomocą metod: 5-krotnej walidacji krzyżowej, bootstrap oraz .632+, wyniosły kolejno 0.1179775, 0.1180151 oraz 0.1008182.

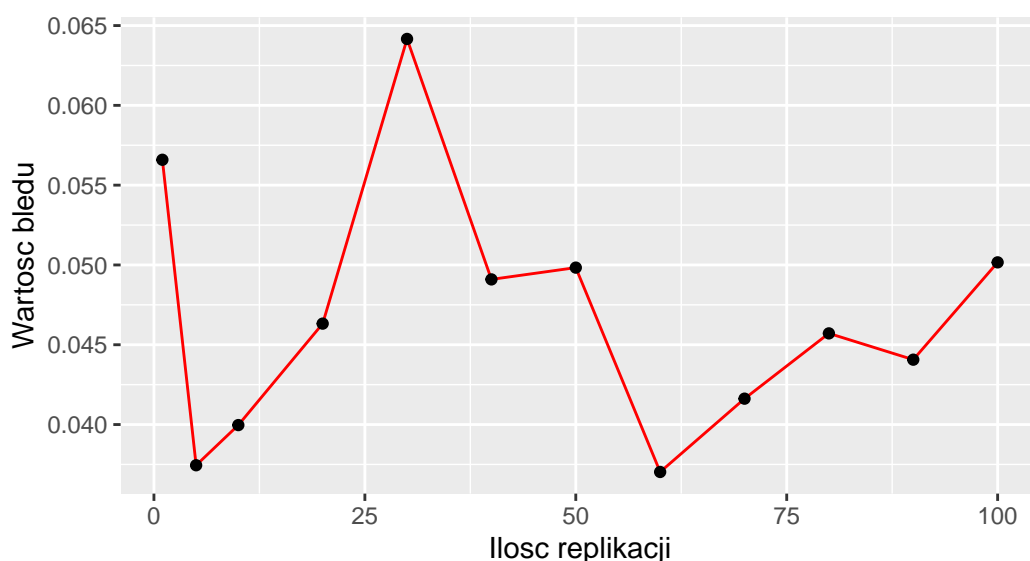
2.1.2 Bagging

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.param=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 60. Parametr złożoności `cp` przyjmujemy równy 0, tak jak w przypadku pojedynczego drzewa.

```

bagging.start <- Sys.time()
bagging.model <- bagging(Type~., data=train.data, nbagg=choice,
  minsplit=1, cp=0)
bagging.end <- Sys.time()
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.

	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.

Błędy klasyfikacji to kolejno 0 i 0.05.

Wyznamy teraz dla tego modelu klasyfikacyjnego błędy predykcji podobnie jak dla drzewa decyzyjnego.

```
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, 0.0561724 oraz 0.0298668.

2.1.3 Boosting

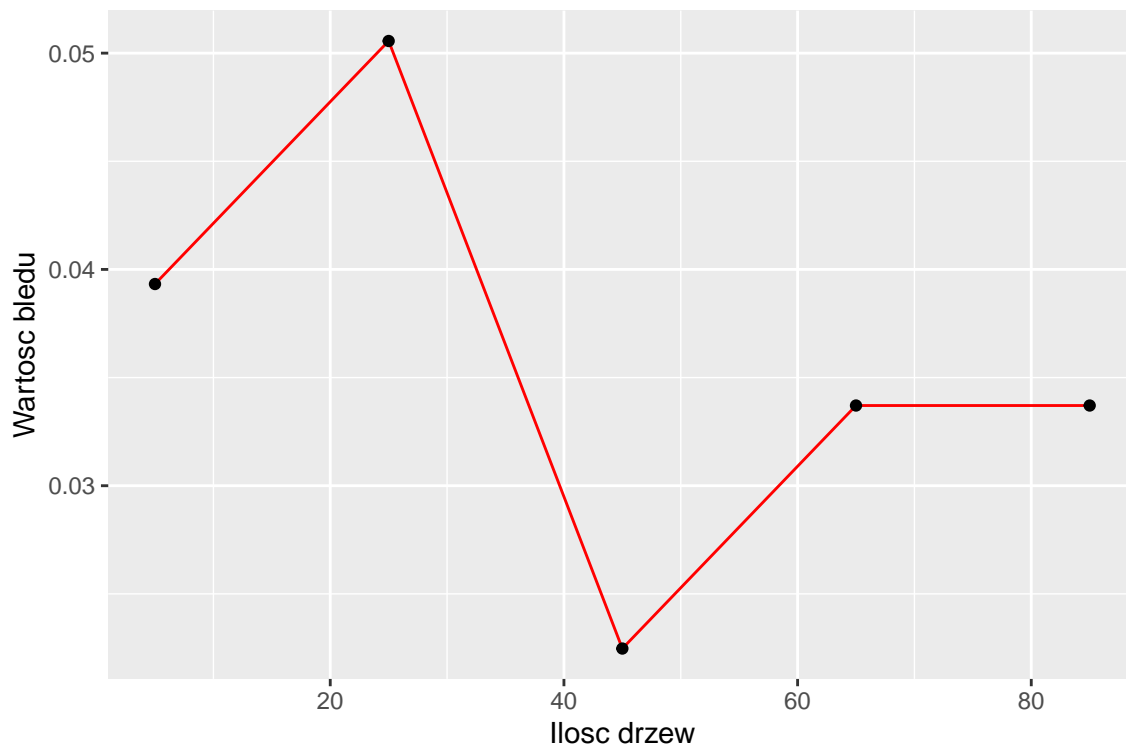
Wykorzystamy teraz algorytm boosting — skorzystamy z funkcji `boosting` z pakietu `adabag`.

Najpierw dobierzemy optymalnie wartość parametru `mfinal` — ilość wykorzystanych przez algorytm drzew.

```
mfinal.choice <- 20
mfinal.vector <- seq(5, 100, by=20)
predictor <- function(model, newdata)
{ as.factor(predict(model, newdata=newdata, type = "class"))$class }
boosting.error.rates <- sapply(mfinal.vector, function(m)
  { errorest(Type~., wine, predict=predictor,
              model=boosting, mfinal=m,
              estimator="cv",
              est.param=control.errorest(k = 5))$error })
mfinal.choice <- mfinal.vector[which.min(boosting.error.rates)]
```

Wybieramy wartość `mfinal` równą 45.

```
boosting.start <- Sys.time()
boosting.model <- boosting(Type~., data = train.data, boos = TRUE,
```



Rysunek 3: Zaleznosc bledu od ilosci drzew.

```

                                mfinal = mfinal.choice)
boosting.end <- Sys.time()
boosting.test.pred <- predict(boosting.model, test.data)
boosting.train.pred <- predict(boosting.model, train.data)

test.confusion <- boosting.test.pred$confusion
train.confusion <- boosting.train.pred$confusion

print(xtable(train.confusion), file="boost1.tex", floating=FALSE)
print(xtable(test.confusion), file="boost2.tex", floating=FALSE)

```

	1	2	3		1	2	3
1	36	0	0	1	22	0	0
2	0	51	0	2	1	19	0
3	0	0	31	3	0	1	17
(a) Zbior uczacy				(b) Zbior testowy			

Tabela 2: Macierze pomylek dla algorytmu boosting.

Błędy klasyfikacji to kolejno 0 i 0.0333333.

Wyznamy tez teraz błędy predykcji (tym razem z racji złożoności obliczeniowej i długiego

czasu wykonania tylko dla metody 5-krotnej walidacji krzyżowej).

```
boosting.predictor <- function(formula, data)
{ boosting(formula, data = data, mfinal=mfinal.choice, boos = TRUE)}

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

Błąd wyniósł 0.0224719.

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)]

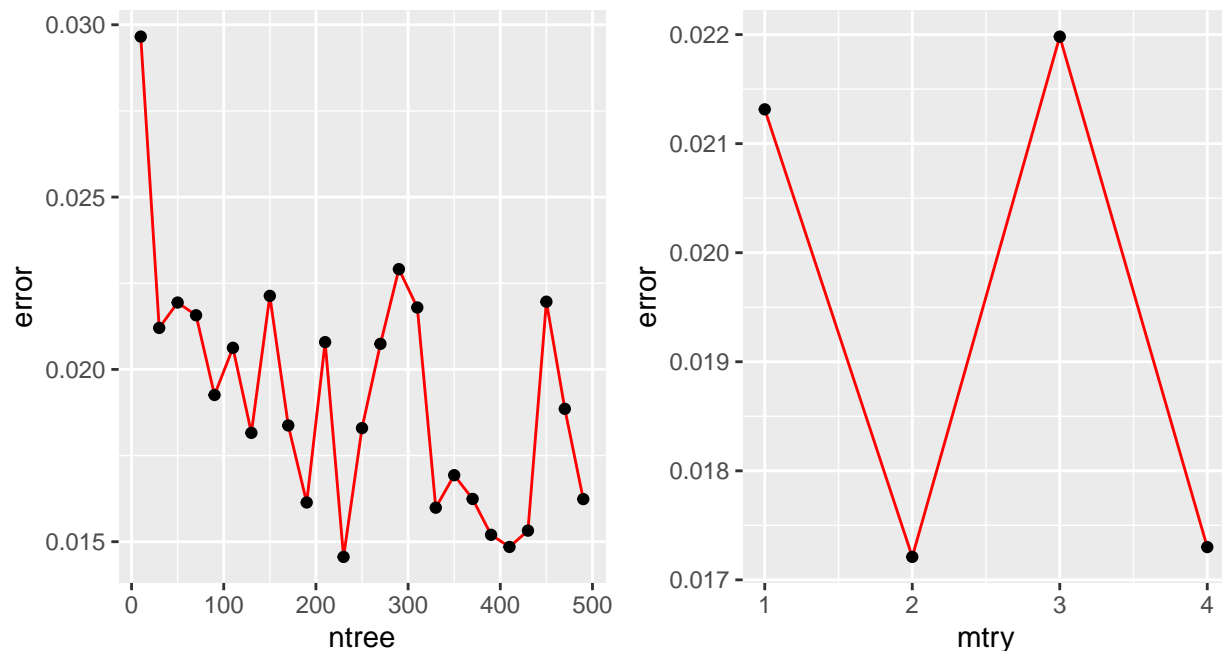
mtry.vector <- seq(1, sqrt(ncol(wine))+1, by=1)
mtry.error.rates <- sapply(mtry.vector, function(m)
  {errorest(Type~., data=train.data, model=randomForest, ntree = ntree.choice,
            mtry=m, estimator="632plus",
            est.param=control.errorest(nboot = 20))$error})
mtry.choice <- mtry.vector[which.min(mtry.error.rates)]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

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

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

Błędy klasyfikacji to kolejno 0 i 0.0166667.

Tak jak dla wcześniejszych algorytmów, wyznaczymy teraz błędy predykcji.



Rysunek 4: Wykresy zależności błędów klasyfikacji od parametrów mtry i ntree.

	1	2	3		1	2	3
1	36	0	0	1	23	0	0
2	0	51	0	2	0	19	0
3	0	0	31	3	0	1	17

(a) Zbiór uczący (b) Zbiór testowy

Tabela 3: Macierze pomyłek dla algorytmu randomForest.

```

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

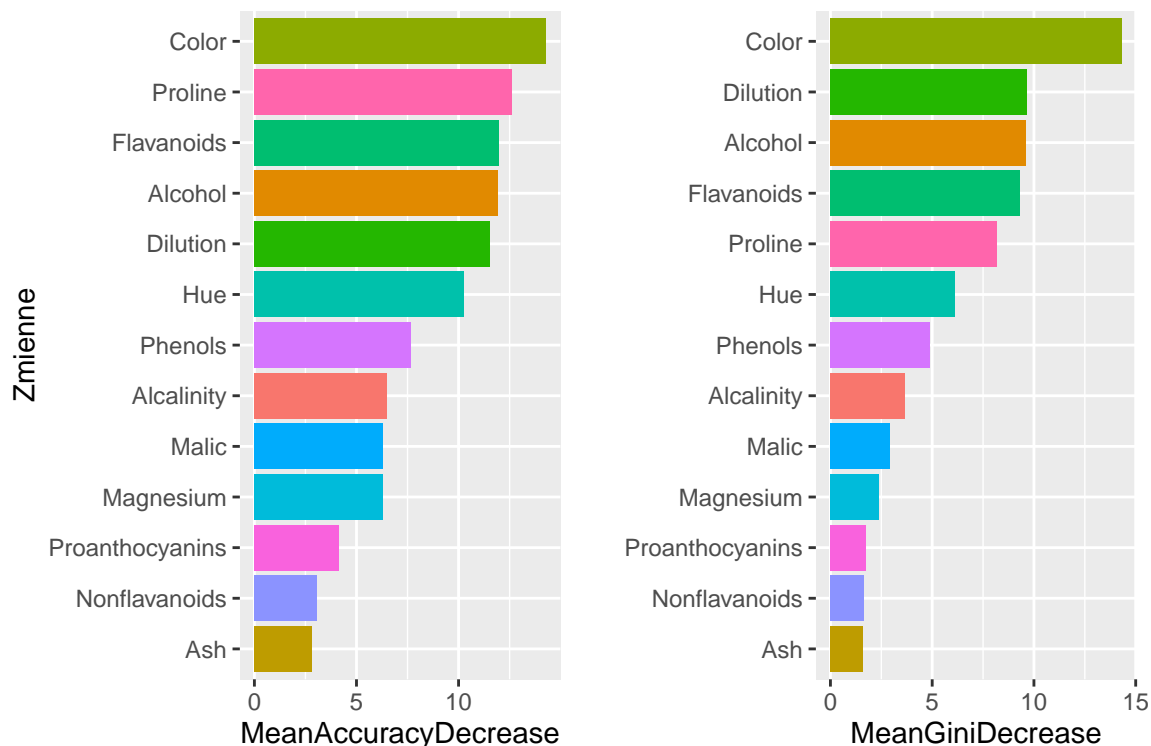
rf.predictor <- function(formula, data)
{ randomForest(formula, data = data, ntree = ntree.choice, mtry = mtry.choice) }

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))

```

Wyniosły one kolejno 0.0168539, 0.0183539 oraz 0.0106015.

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



Rysunek 5: Wykres waznosci zmiennych.

Przypomnijmy, że na ostatniej liście za zmienne istotne, takie, które dobrze dywersyfikowały klasy ze zbioru `wine`, były `Alcohol` i `Flavanoids`. Tak jak widzimy to na rysunku (5) dokonaliśmy wtedy całkiem dobrej decyzji, ponieważ zmienne te są w czołówce najważniejszych zmiennych. Wykresy wskazują, że najważniejsze są zmienne `Color` i `Proline`, które w trakcie dokonywania wyboru, odrzuciliśmy.

2.1.5 Wnioski

Podsumujmy uzyskane rezultaty w tabeli (4).

Metoda	Drzewa decyzyjne	bagging	boosting	random forest
CV	0.1179775	0.0561798	0.0224719	0.0168539
Bootstrap	0.1180151	0.0561724	NaN	0.0183539
632+	0.1008182	0.0298668	NaN	0.0106015

Tabela 4: Wartości błędów predykcji.

Jak widzimy każda rozpatrywana metoda jest znacząco lepsza od pojedynczego drzewa klasyfikacyjnego. Najlepiej z nich poradził sobie algorytm random forest.

Mieliśmy także porównać czasy potrzebne do zbudowania modelu dla kolejnych metod. Wynosiły one:

- dla bagging 0.327126026153564,
- dla boosting 39.2845530509949,
- dla random forest 0.038970947265625.

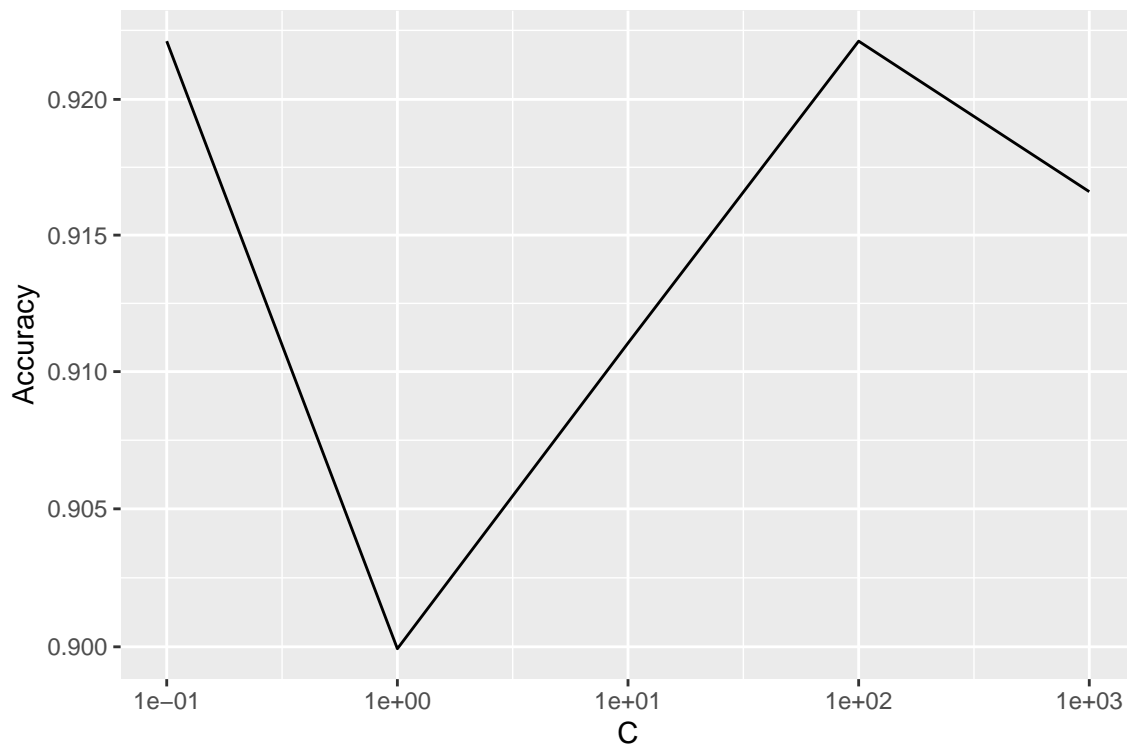
Widać, że algorytm boosting potrzebuje znacznie więcej czasu na stworzenie modelu od pozostałych algorytmów.

2.2 b)

Teraz zbudujemy klasyfikator oparty na algorytmie SVM. W celu późniejszej wizualizacji obszarów decyzyjnych oraz dla uproszczenia modelu wybierzemy tylko 2 cechy: Alcohol i Flavanoids.

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

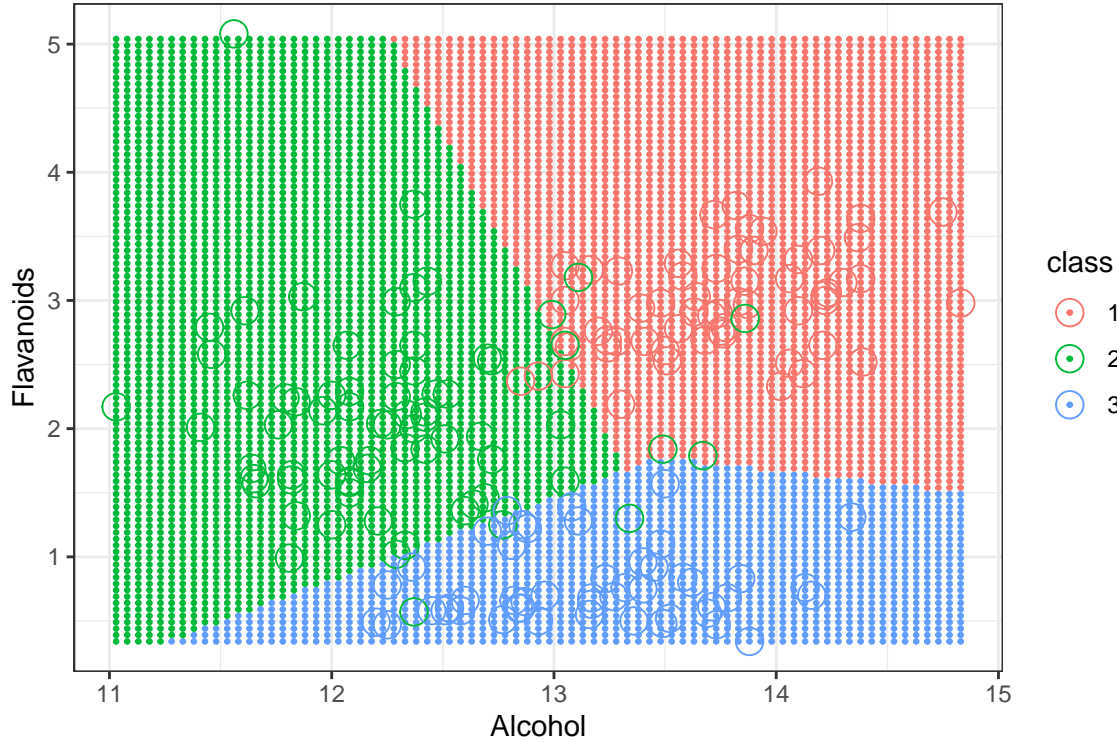
Korzystając z walidacji krzyżowej, sprawdzimy jak klasyfikator z jądrem liniowym radzi sobie przy różnych wartościach parametru kosztu C .



Rysunek 6: Dokładność klasyfikatora od parametru kosztu

Możemy zauważyć, że najlepsze wyniki uzyskujemy dla stosunkowo małej wartości parametru kosztu ($C = 0.1$).

Zbadamy teraz jak wyglądają obszary decyzyjne dla poszczególnych wartości tego parametru — wykresy (7) - (11).



Rysunek 7: Obszary decyzyjne dla $C = 0.1$

Można zaobserwować, że granica pomiędzy typami **Type=1** oraz **Type=2** się praktycznie nie zmienia. Zwiększenie parametru C wpływa jednakże na poszerzenie się obszaru dla **Type=3**.

Sprawdźmy teraz dokładność klasyfikatora przy zastosowaniu różnych jąder.

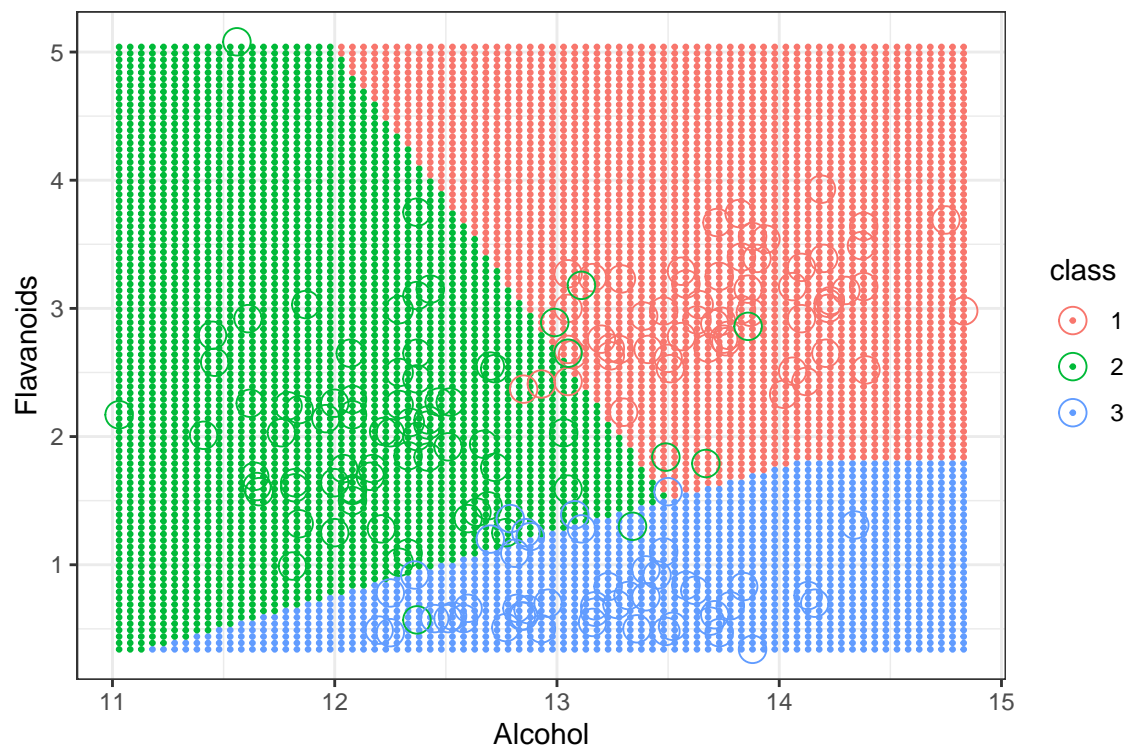
linear	polynomial	radial
0.928	0.938	0.933

Tabela 5: Porównanie klasyfikatorów dla różnych jąder

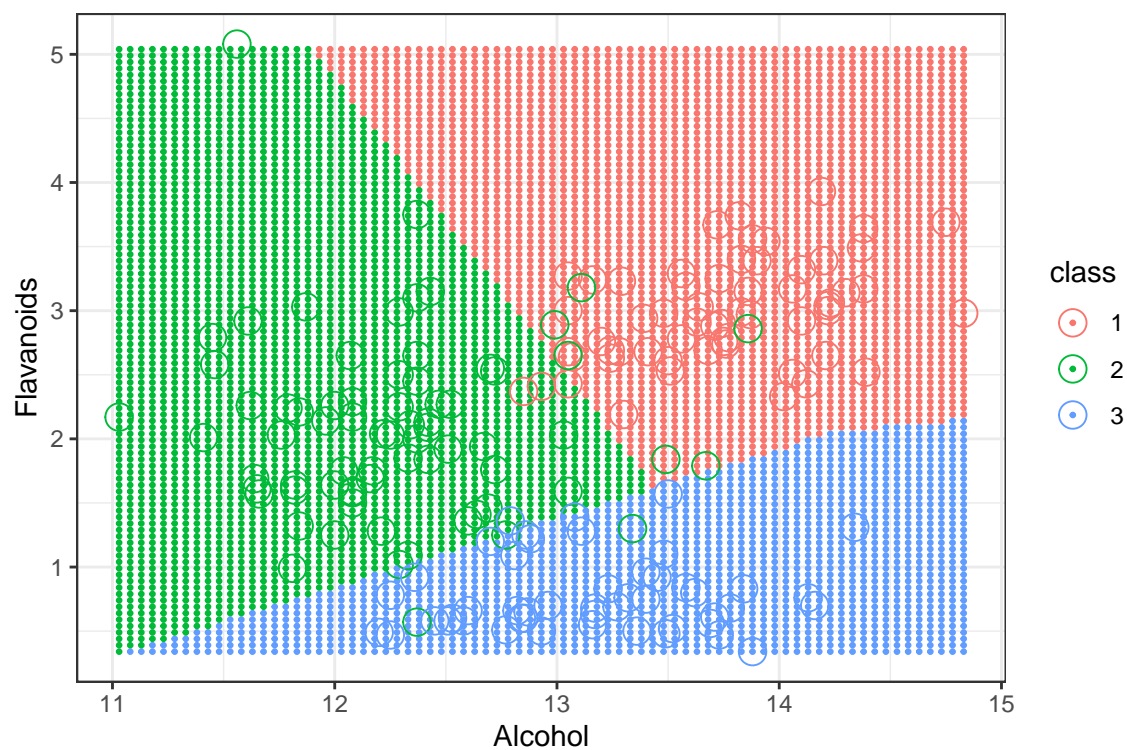
Można zauważyć (5), że zastosowanie nieliniowych jąder trochę poprawiło dokładność klasyfikatora, natomiast różnice pomiędzy jądrem wielomianowym i gaussowskim są nieznaczne.

Sprawdźmy teraz jak wygląda dokładność dla różnych parametrów przy zastosowaniu jądra gaussowskiego.

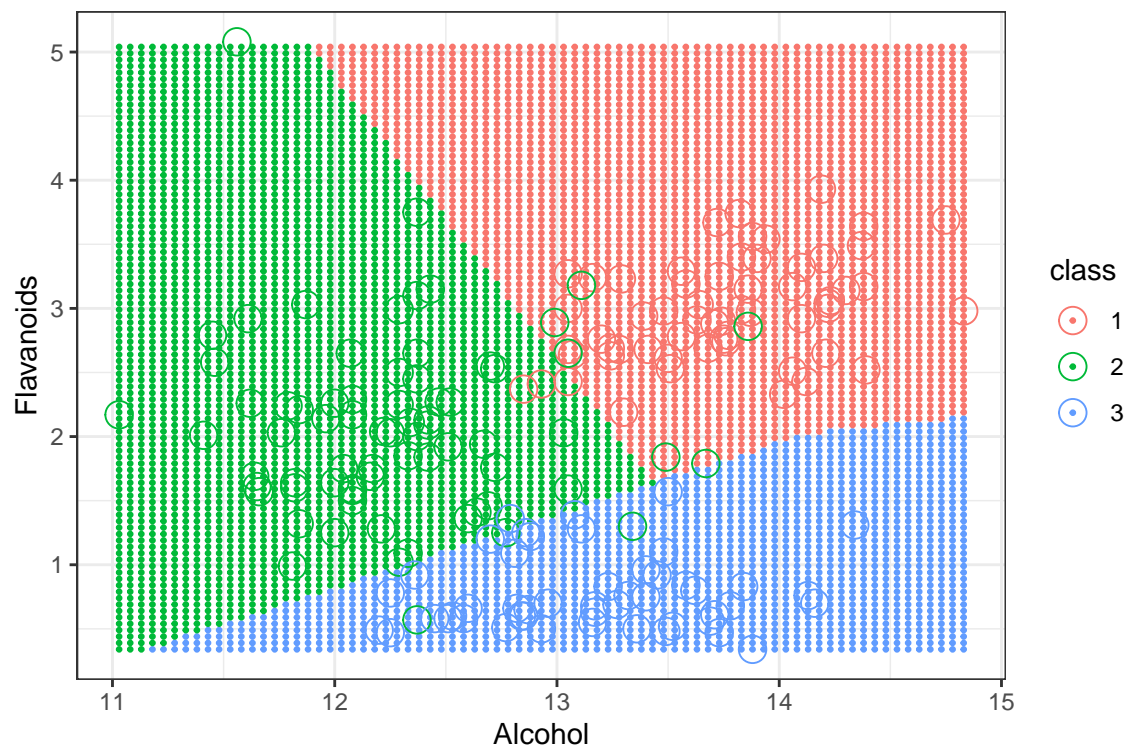
Mapa ciepła (12) pokazuje, że optymalnymi parametrami są σ 0.1 oraz $C = 100$.



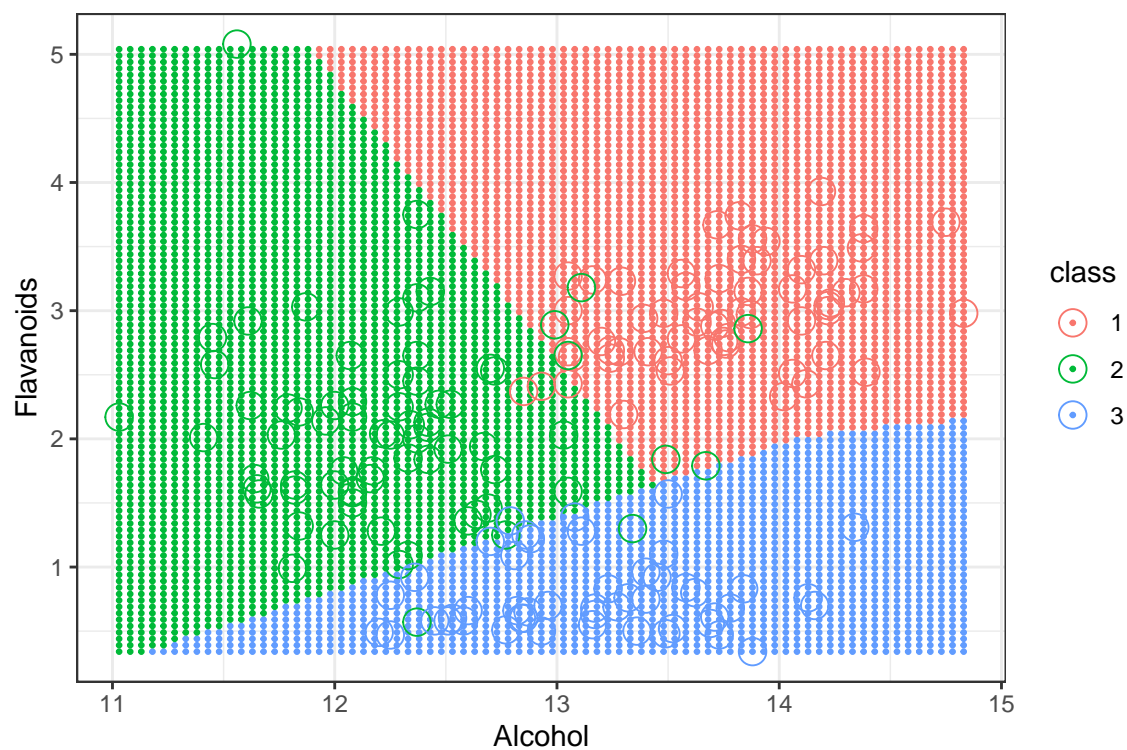
Rysunek 8: Obszary decyzyjne dla $C = 1$



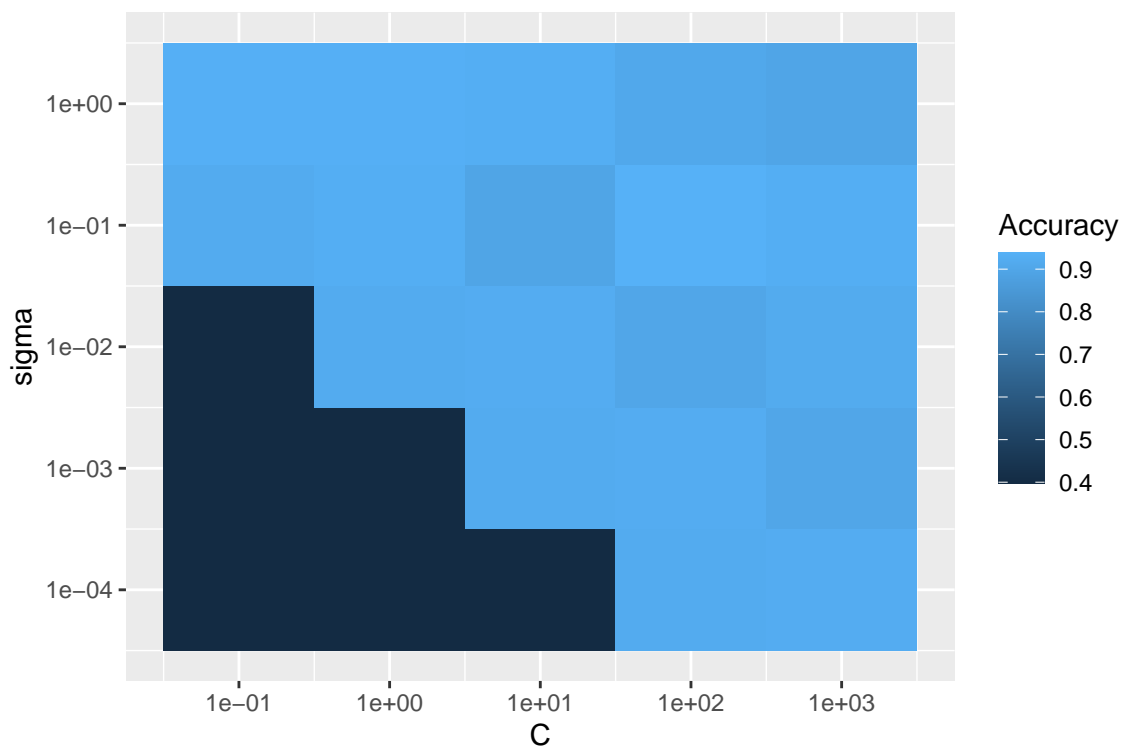
Rysunek 9: Obszary decyzyjne dla $C = 10$



Rysunek 10: Obszary decyzyjne dla $C = 100$



Rysunek 11: Obszary decyzyjne dla $C = 1000$



Rysunek 12: Mapa ciepła dokładności klasyfikatora

sigma	C
0.10	100.00

Tabela 6: Parametry dla najlepszego klasyfikatora

3 Zadanie 2

W tym zadaniu zastosujemy algorytmy analizy skupień do wyznaczenia klastrów dla zbioru *wine*, ocenimy ich skuteczność i porównamy je ze sobą. Sięgnijmy po dwa algorytmy: PAM i AGNES.

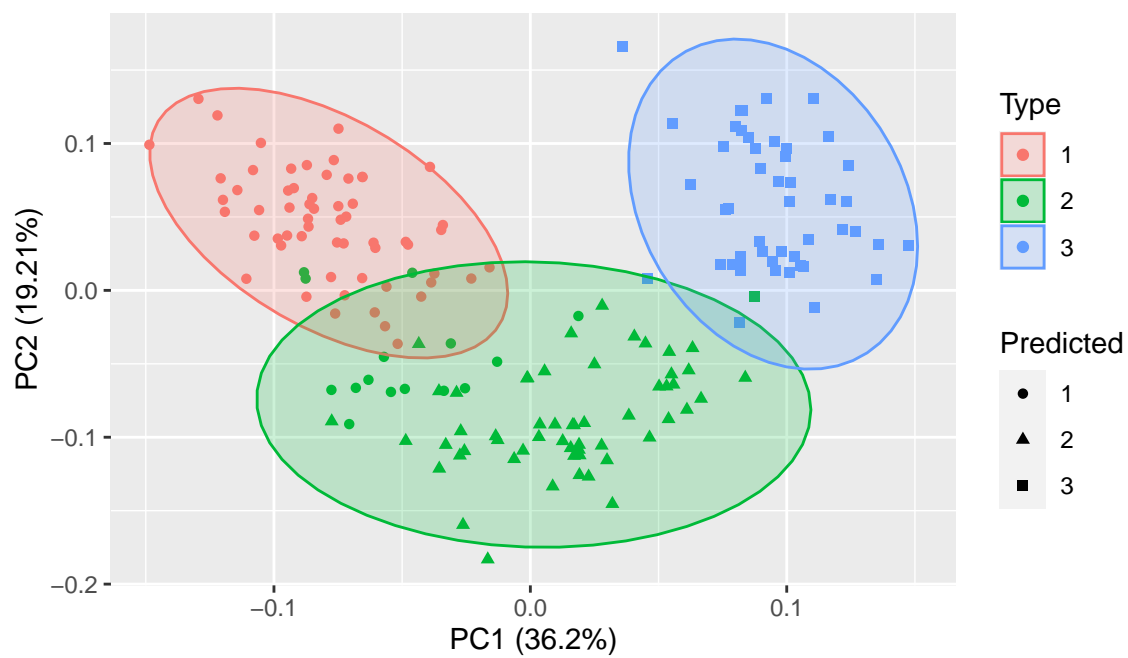
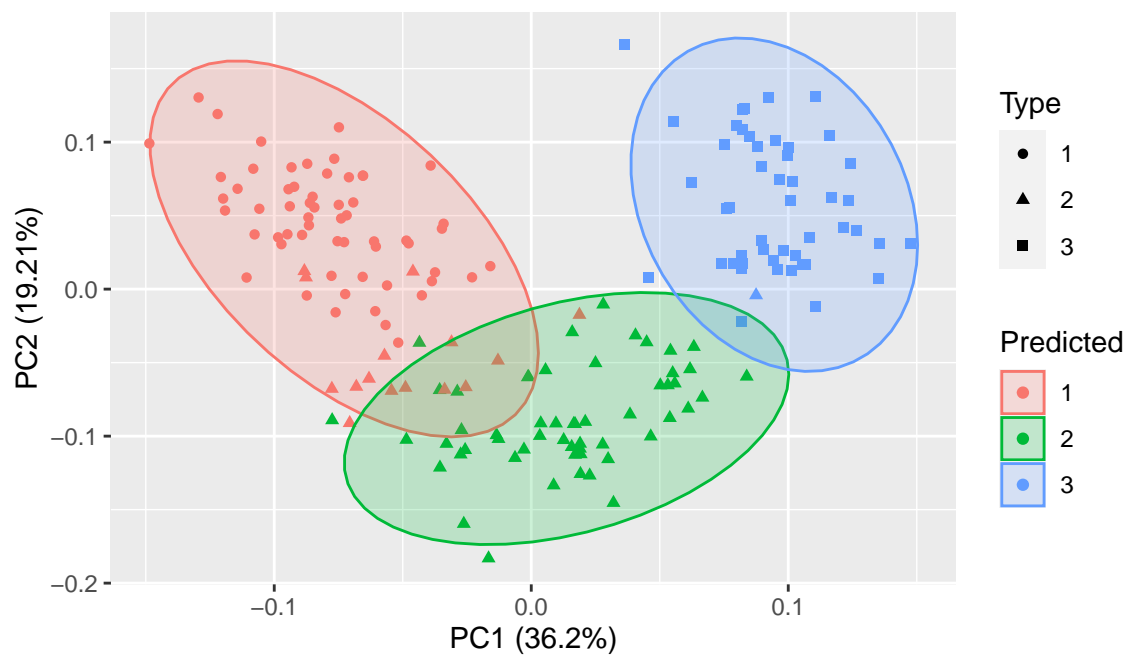
3.1 Wizualizacja wyników grupowania ($K = 3$)

Najpierw wyznaczmy macierz niepodobieństwa dla naszych danych.

```
data(wine)
wine.subset = wine[, -1]
diss.matrix <- daisy(wine.subset, stand=TRUE) %>% as.matrix
group.colors <- as.numeric(wine$Type)
```

Przyjrzyjmy się najpierw jakie wyniki daje nam zastosowanie algorytmu PAM — (13).

Dla algorytmu PAM widać zgodne z rzeczywistymi wyodrębnienie klas, dobrą separację i



Rysunek 13: Skupienia dla metody PAM

optymalną zwartość.

Zobaczmy teraz, jak poradził sobie algorytm AGNES z single-linkage (14).

Dla algorytmu AGNES z single-linkage widać, że prawie wszystkie obiekty zostały przyporządkowane do klasy pierwszej. Algorytm poradził sobie źle — nie widać wyodrębnienia klas.

Zobaczmy jak wygląda dendrogram dla tego modelu — (15).

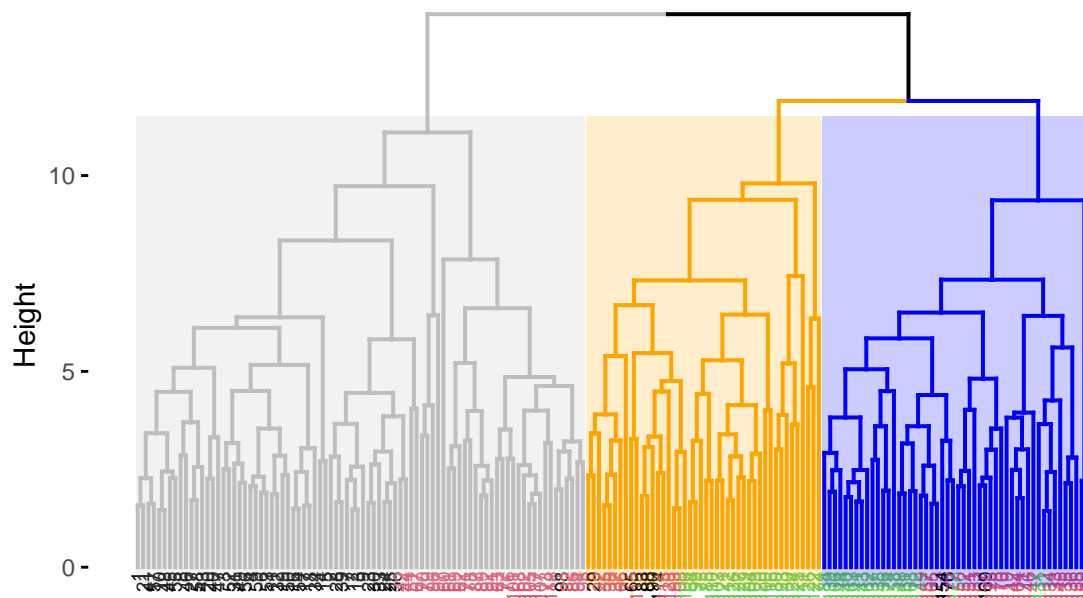
Dendrogram dla AGNES z single-linkage potwierdza wcześniejsze wnioski — widzimy, że zdecydowana większość obiektów została przyporządkowana do jednej klasy.

Poniżej wyniki dla algorytmu AGNES z complete-linkage — (16).

Dla algorytmu AGNES z complete-linkage otrzymane wyniki są lepsze niż w przypadku AGNES z single-linkage, ale odseparowanie klas jest i tak gorsze niż dla algorytmu PAM.

Zobaczmy jak wygląda dendrogram w tym przypadku — (??).

Partycja na 3 skupienia a rzeczywiste klasy – complete-linkage

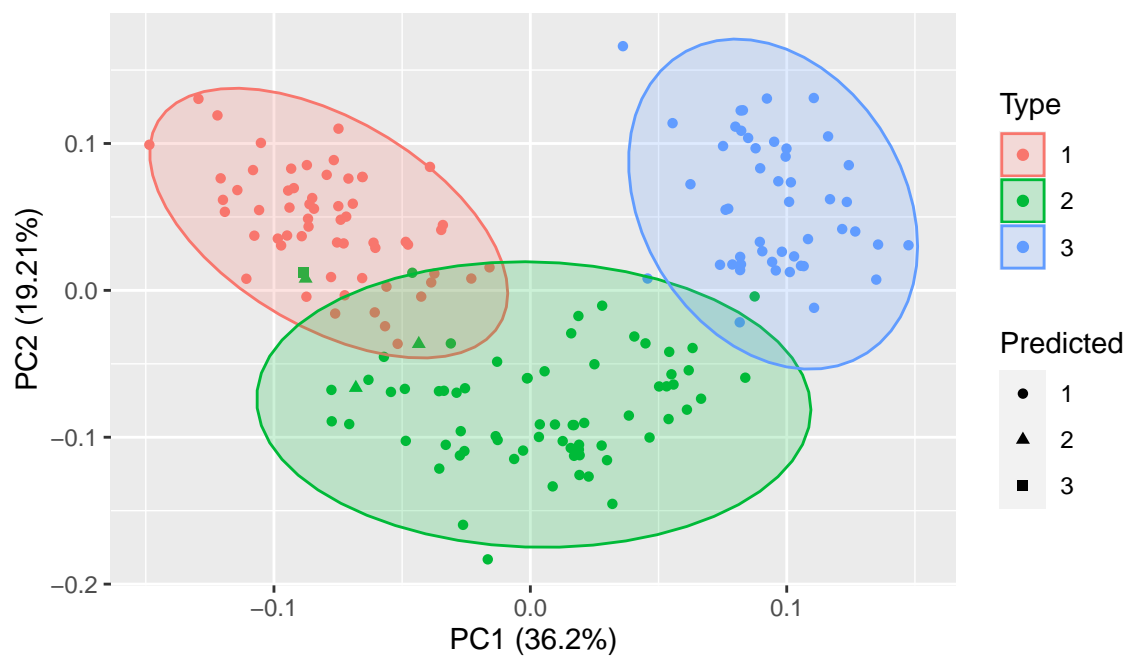
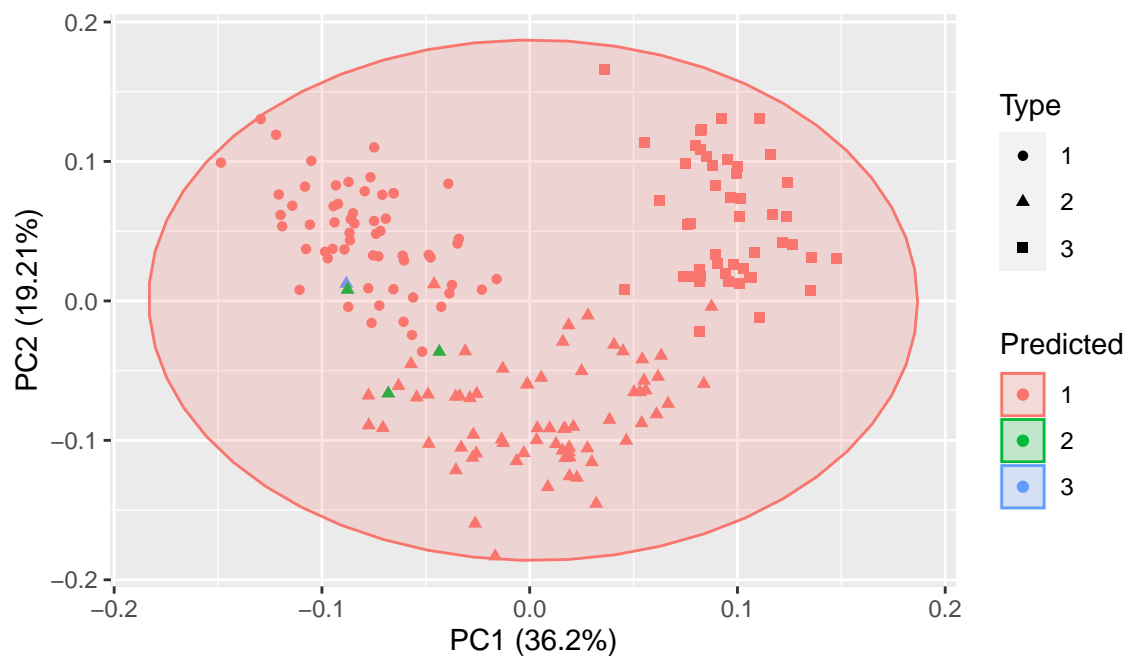


Den-

drogram potwierdza nasze wnioski.

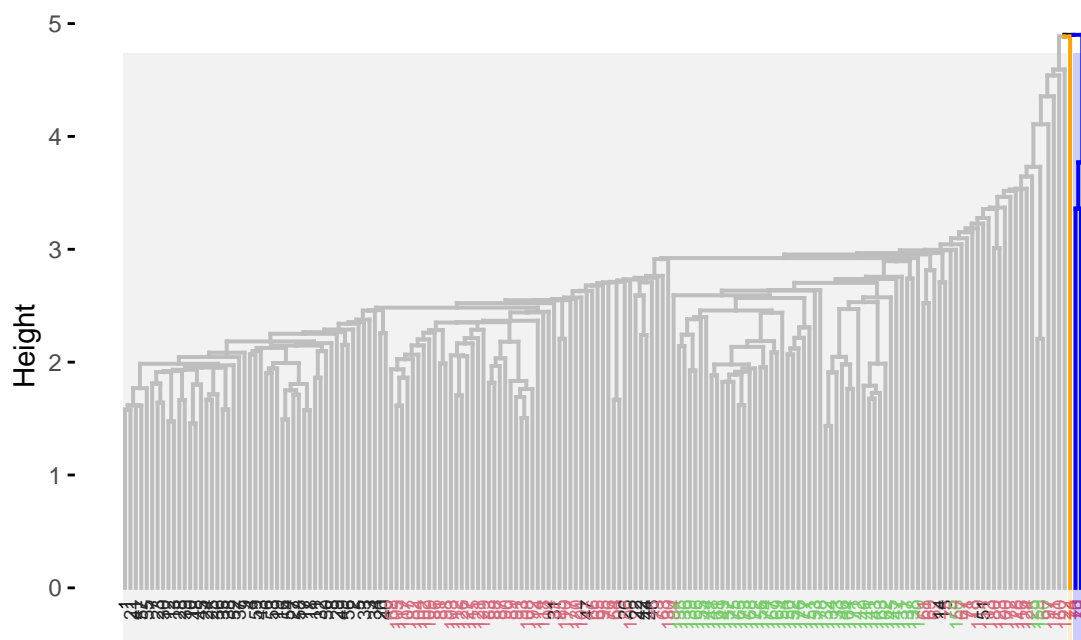
3.2 Ocena jakości grupowania

W tej części zadania, porównamy ze sobą algorytmy, jakość uzyskanego dzięki nim grupowania w zależności od przyjętej ilości skupień. Wykorzystamy wskaźniki wewnętrzne, jak i zewnętrzne.



Rysunek 14: Skupienia dla metody AGNES z single-linkage

Partycja na 3 skupienia a rzeczywiste klasy – single-linkage



Rysunek 15: Dendrogram dla single-linkage.

3.2.1 Wskaźniki wewnętrzne

Na wykresach (17) porównujemy ze sobą algorytmy PAM i AGNES z complete-linkage — plotujemy kolejne wartości wskaźników wewnętrznych dla różnej ilości skupisk K .

Tabele (7), (8), (9) zbierają informacje o średnicach, rozmiarach skupisk i separacji dla algorytmu PAM.

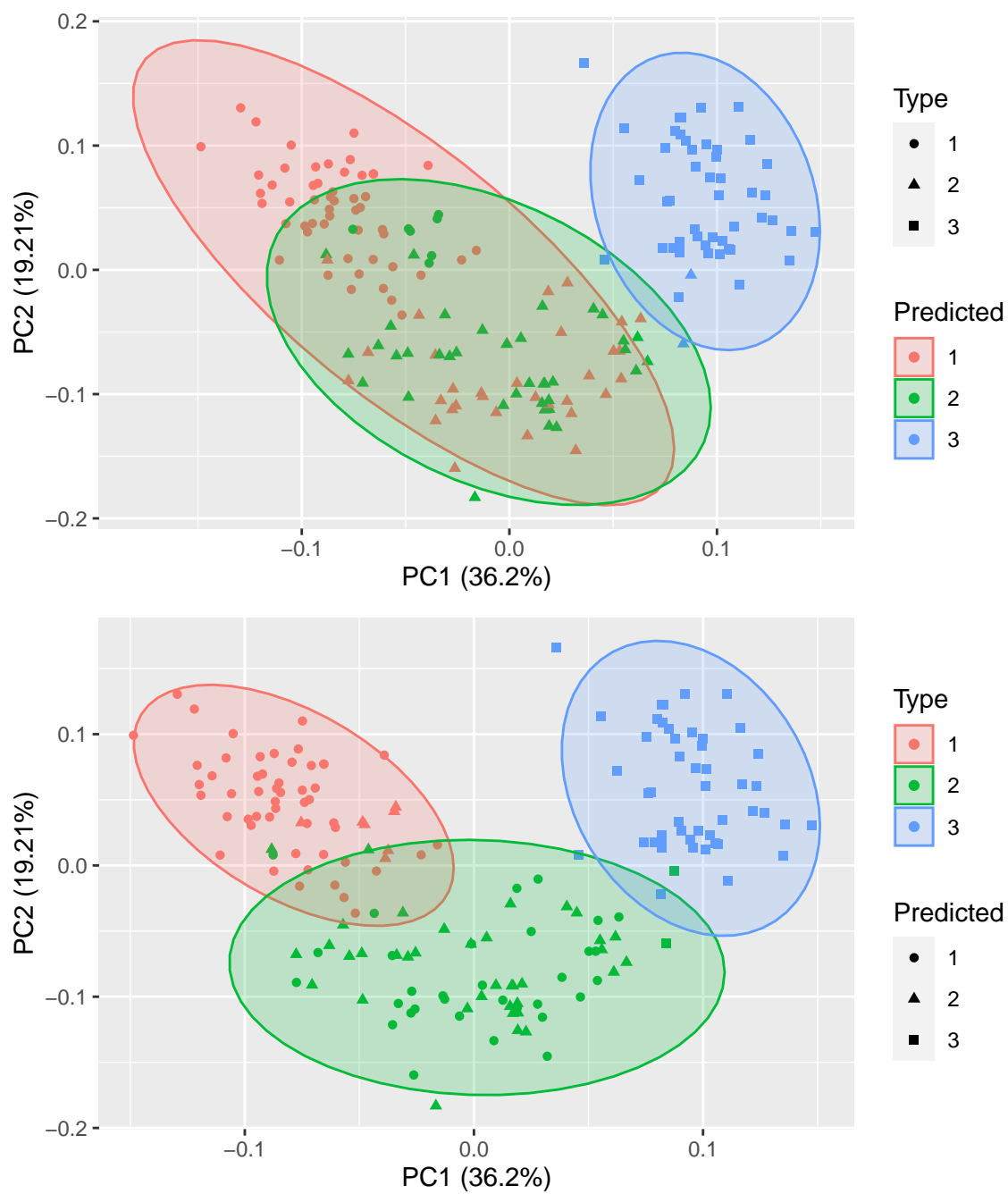
Tabele (10), (11), (12) zbierają natomiast informacje o średnicach, rozmiarach skupisk i separacji dla algorytmu AGNES z complete-linkage.

3.2.2 Wskaźniki zewnętrzne

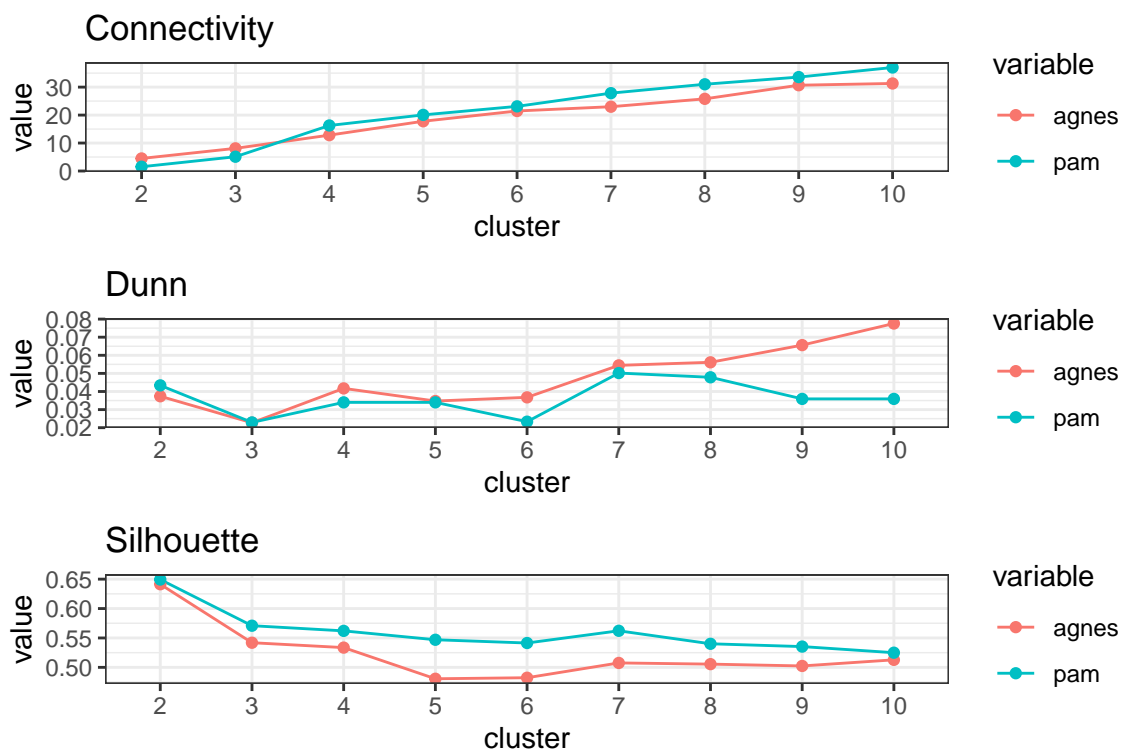
Na wykresie (18) przedstawiliśmy wartości wskaźniku **agreement** dla wszystkich trzech rozpatrywanych metod.

Podsumowując wyniki uzyskane z analizy wskaźników wewnętrznych, jak i zewnętrznych dochodzimy do wniosku, że:

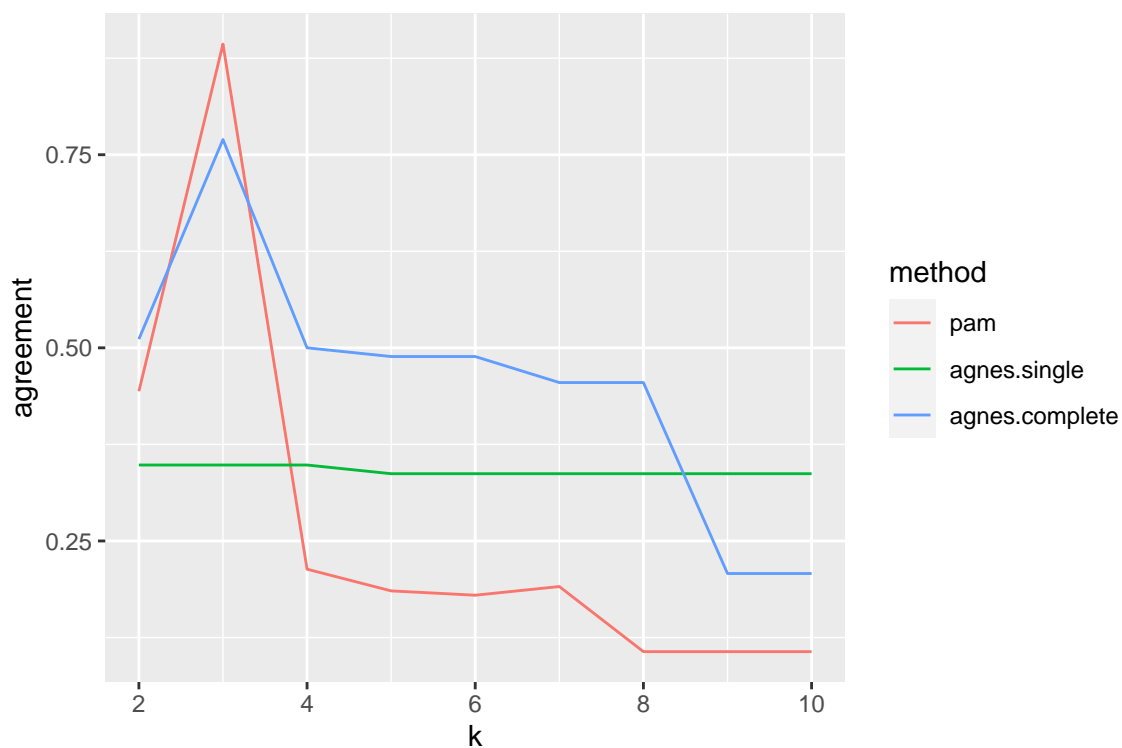
- metoda AGNES z single-linkage poradziła zdecydowanie najgorzej,
- wskaźniki wewnętrzne mówią nam, że optymalnym wyborem jest $K = 2$,
- wskaźniki zewnętrzne jednoznacznie wskazują na $K = 3$,
- wskaźniki wewnętrzne niejednoznacznie wskazują na to, który algorytm poradził sobie



Rysunek 16: Skupienia dla metody AGNES z complete-linkage



Rysunek 17: Wskaźniki wewnętrzne dla PAM i AGNES z complete-linkage



Rysunek 18: Porównanie wskaźników zewnętrznych

	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$	$K = 8$	$K = 9$	$K = 10$
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.93
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.93	1.75
3	0.00	0.00	0.00	0.00	0.00	0.00	1.93	1.75	1.75
4	0.00	0.00	0.00	0.00	0.00	1.64	1.75	1.75	2.05
5	0.00	0.00	0.00	0.00	1.64	1.64	1.75	1.94	2.03
6	0.00	0.00	0.00	1.64	1.64	1.66	1.94	1.94	2.17
7	0.00	0.00	1.64	1.64	1.71	1.94	1.94	2.35	1.93
8	0.00	2.21	1.64	1.71	1.71	2.35	2.35	1.93	2.03
9	2.15	2.14	2.21	1.71	1.93	1.93	1.93	4.36	4.90
10	2.15	2.14	2.76	2.76	1.93	1.93	1.93	1.93	1.93

Tabela 7: Seperacja w skupiskach — PAM.

	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$	$K = 8$	$K = 9$	$K = 10$
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.02
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.02	6.35
3	0.00	0.00	0.00	0.00	0.00	0.00	8.87	6.35	7.08
4	0.00	0.00	0.00	0.00	0.00	9.16	6.35	8.22	9.80
5	0.00	0.00	0.00	0.00	9.16	9.52	8.22	6.57	8.91
6	0.00	0.00	0.00	9.16	9.46	9.85	6.57	8.91	7.64
7	0.00	0.00	9.16	9.86	10.44	6.57	9.85	6.87	7.12
8	0.00	11.26	9.86	10.44	7.64	6.87	6.87	7.12	4.49
9	11.37	10.90	10.90	7.64	7.12	7.12	7.12	7.06	4.13
10	11.17	9.03	9.03	9.37	7.98	7.98	7.98	7.98	7.98

Tabela 8: Srednice skupisk — PAM.

lepiej; wskaźniki zewnętrzne mówią nam, że dla $K = 3$ najlepiej poradził sobie algorytm PAM.

3.2.3 Ocena otrzymanych rezultatów

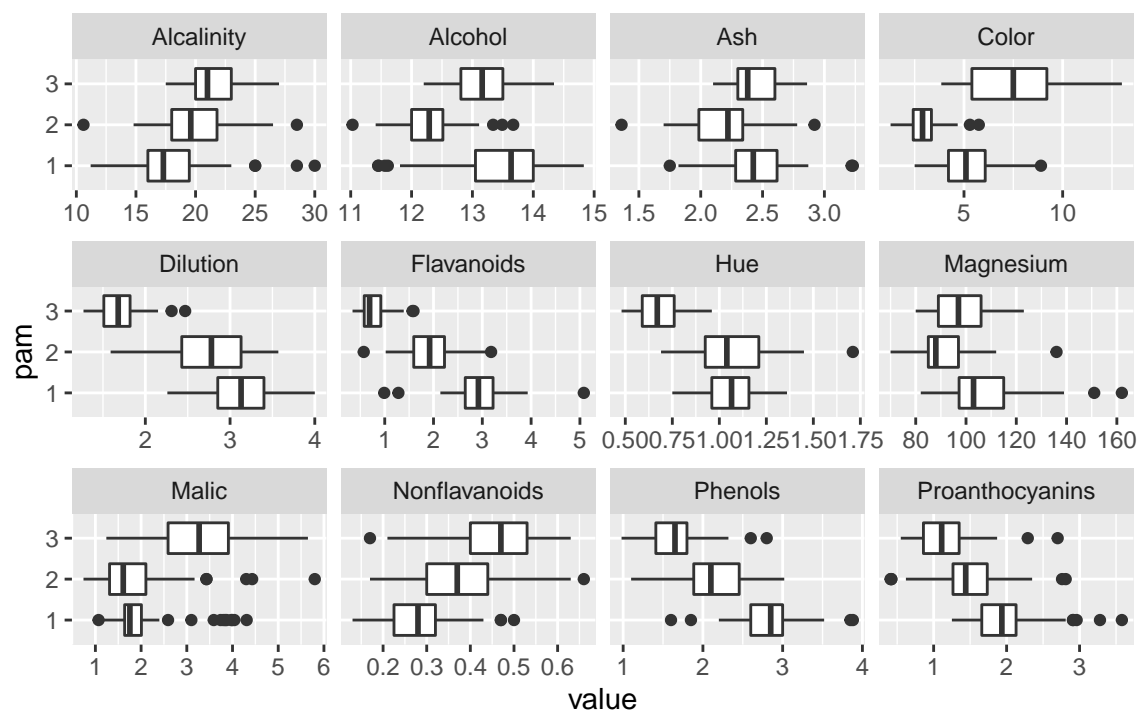
Jako optymalną ilość skupisk przyjmujemy $K = 3$. Dla wyznaczonych skupisk przedstawimy wykresy pudełkowe — osobno dla algorytmu PAM i AGNES z complete-linkage.

	K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	19.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	19.00	20.00
3	0.00	0.00	0.00	0.00	0.00	0.00	22.00	20.00	23.00
4	0.00	0.00	0.00	0.00	0.00	32.00	20.00	24.00	12.00
5	0.00	0.00	0.00	0.00	32.00	33.00	24.00	24.00	16.00
6	0.00	0.00	0.00	32.00	29.00	20.00	24.00	18.00	19.00
7	0.00	0.00	32.00	31.00	41.00	24.00	19.00	16.00	23.00
8	0.00	75.00	40.00	41.00	22.00	16.00	16.00	22.00	12.00
9	111.00	54.00	58.00	25.00	23.00	22.00	22.00	4.00	3.00
10	67.00	49.00	48.00	49.00	31.00	31.00	31.00	31.00	31.00

Tabela 9: Rozmiary skupisk — PAM.

	K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.63
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.63	0.31
3	0.00	0.00	0.00	0.00	0.00	0.00	0.63	0.31	0.31
4	0.00	0.00	0.00	0.00	0.00	0.63	0.31	0.31	0.12
5	0.00	0.00	0.00	0.00	0.63	0.31	0.31	0.12	0.12
6	0.00	0.00	0.00	0.63	0.31	0.31	0.12	0.12	0.11
7	0.00	0.00	0.63	0.31	0.31	0.12	0.12	0.11	0.11
8	0.00	0.63	0.31	0.31	0.12	0.12	0.11	0.11	0.14
9	0.63	0.31	0.31	0.12	0.12	0.11	0.11	0.14	0.13
10	0.63	0.31	0.72	0.12	0.41	0.11	0.14	0.36	0.13

Tabela 10: Seperacja w skupiskach — Agnes.

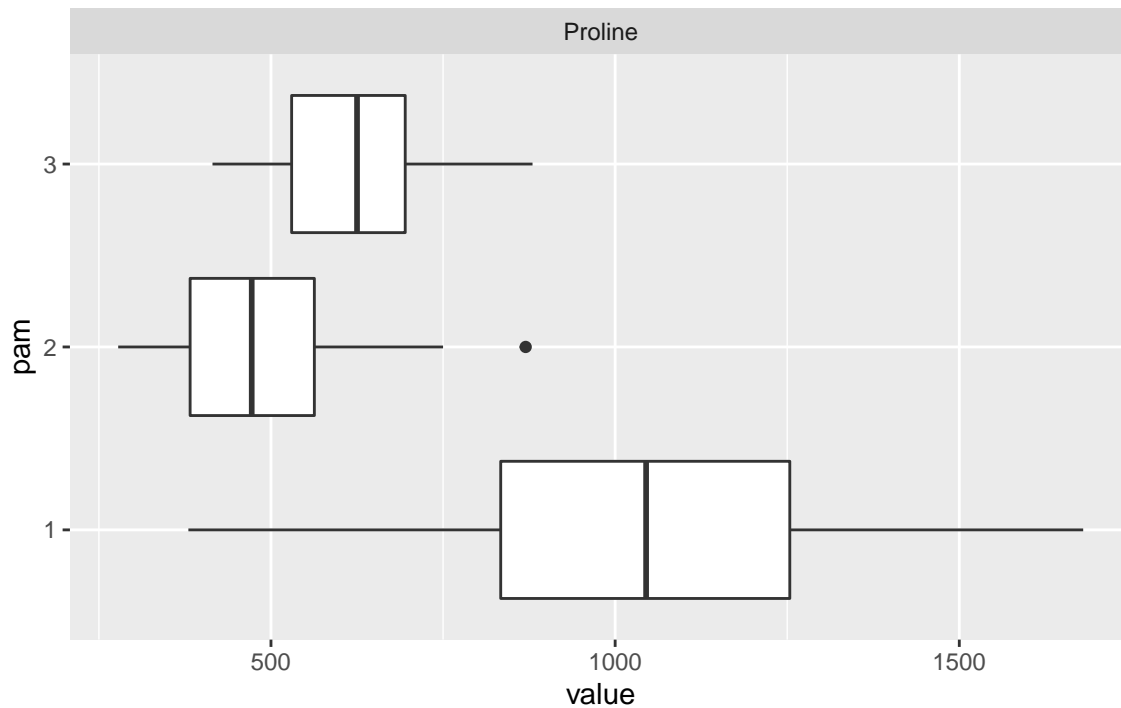


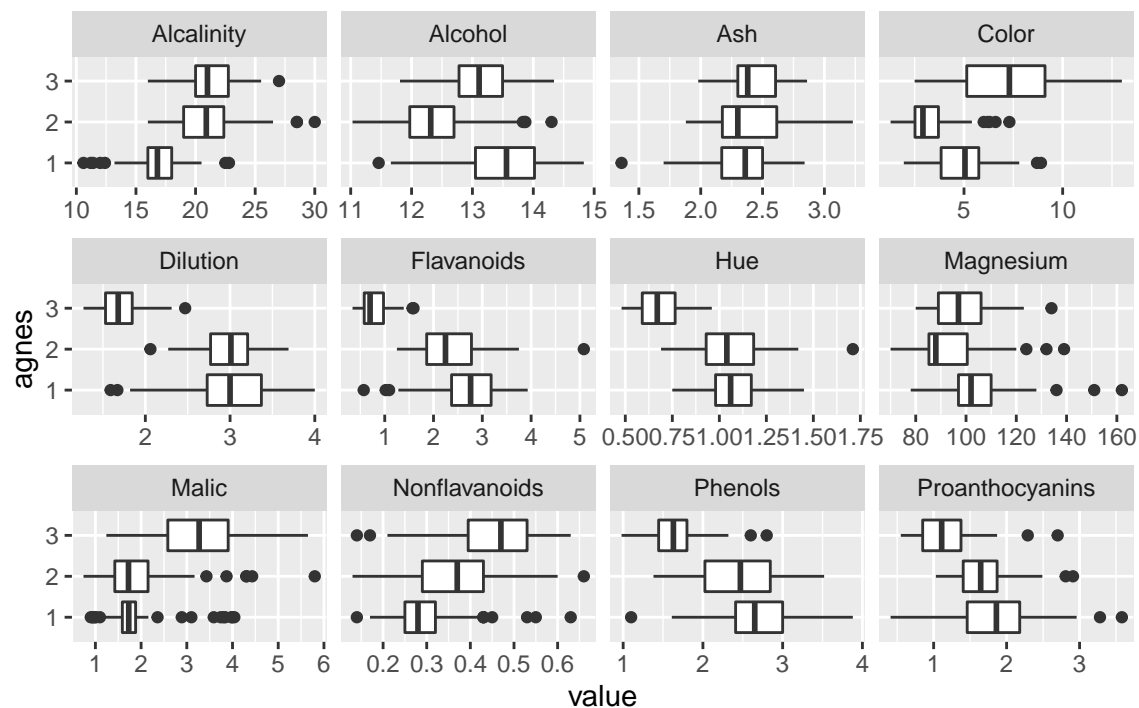
	K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.53
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.53	7.91
3	0.00	0.00	0.00	0.00	0.00	0.00	8.53	7.91	7.60
4	0.00	0.00	0.00	0.00	0.00	8.53	7.91	7.60	6.88
5	0.00	0.00	0.00	0.00	8.53	7.91	7.60	6.88	6.76
6	0.00	0.00	0.00	8.53	7.91	7.60	6.88	6.76	6.36
7	0.00	0.00	8.53	7.91	7.60	6.88	6.76	6.36	6.24
8	0.00	8.53	7.91	7.60	6.88	6.76	6.36	6.24	6.10
9	8.53	7.91	7.60	6.88	6.76	6.36	6.24	6.10	5.75
10	7.91	7.60	6.88	6.76	6.36	6.24	6.10	5.75	5.62

Tabela 11: Srednice skupisk — Agnes.

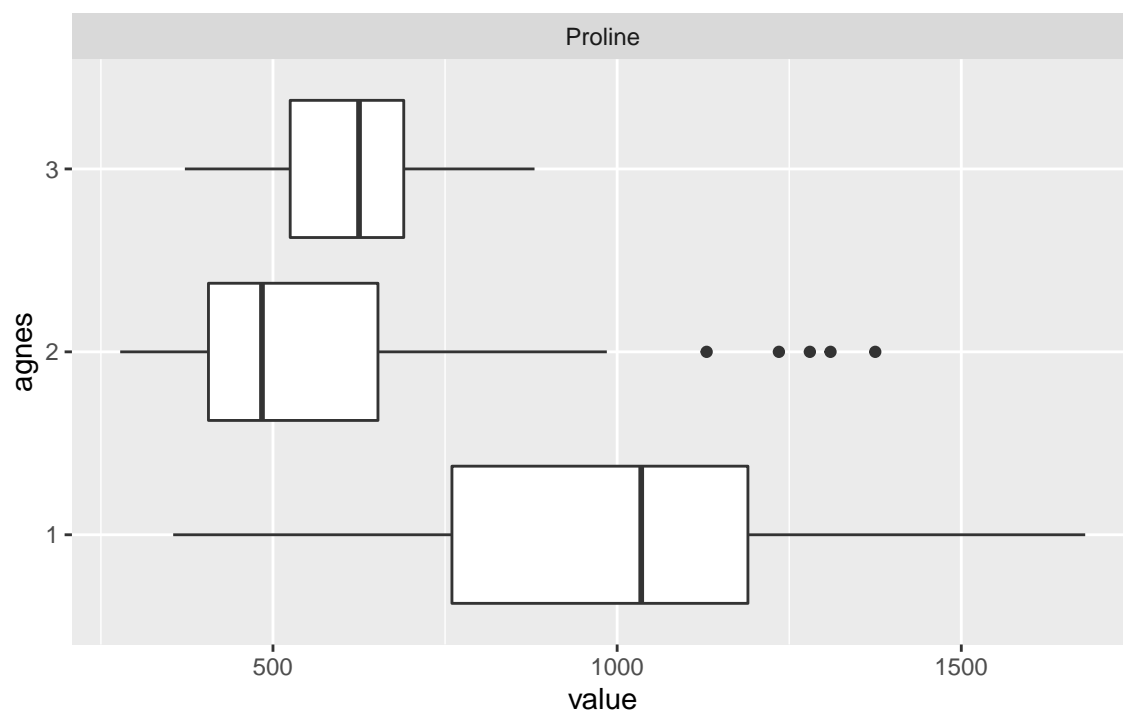
	K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	K = 9	K = 10
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	58.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	58.00	3.00
3	0.00	0.00	0.00	0.00	0.00	0.00	115.00	3.00	1.00
4	0.00	0.00	0.00	0.00	0.00	115.00	3.00	1.00	56.00
5	0.00	0.00	0.00	0.00	115.00	3.00	1.00	57.00	1.00
6	0.00	0.00	0.00	118.00	3.00	1.00	3.00	3.00	3.00
7	0.00	0.00	170.00	3.00	1.00	4.00	50.00	50.00	50.00
8	0.00	174.00	3.00	1.00	4.00	50.00	1.00	1.00	1.00
9	177.00	3.00	1.00	4.00	52.00	3.00	3.00	3.00	3.00
10	1.00	1.00	4.00	52.00	3.00	2.00	2.00	2.00	2.00

Tabela 12: Rozmiary skupisk — Agnes.





Page 1



Page 2

Widzimy, że dla algorytmu PAM uzyskujemy zmienne, które dobrze dywersyfikują klasy, np. Flavanoids czy Phenols.

Dla algorytmu AGNES natomiast, takich zmiennych nie zauważamy.

	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 13: Medoidy dla metody PAM przy K=3

Jako, że na wykresach pudełkowych zauważyliśmy, że zmienne **Phenols** i **Flavaoids** wprowadzały dobry podział na klasy, to od razu zauważamy, że różnice pomiędzy medoidami dla tych zmiennych są znaczące.