

SPRAWOZDANIE

Zajęcia: Analiza procesów uczenia

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium 5

14 czerwca 2020

Temat: Modelowanie procesów uczenia maszynowego w pakiecie mlr. Trenowanie, ocena i porównywanie modeli w pakiecie mlr

Wariant: 1

Adres repozytorium: <https://github.com/Konradbor/APU/tree/master/5>

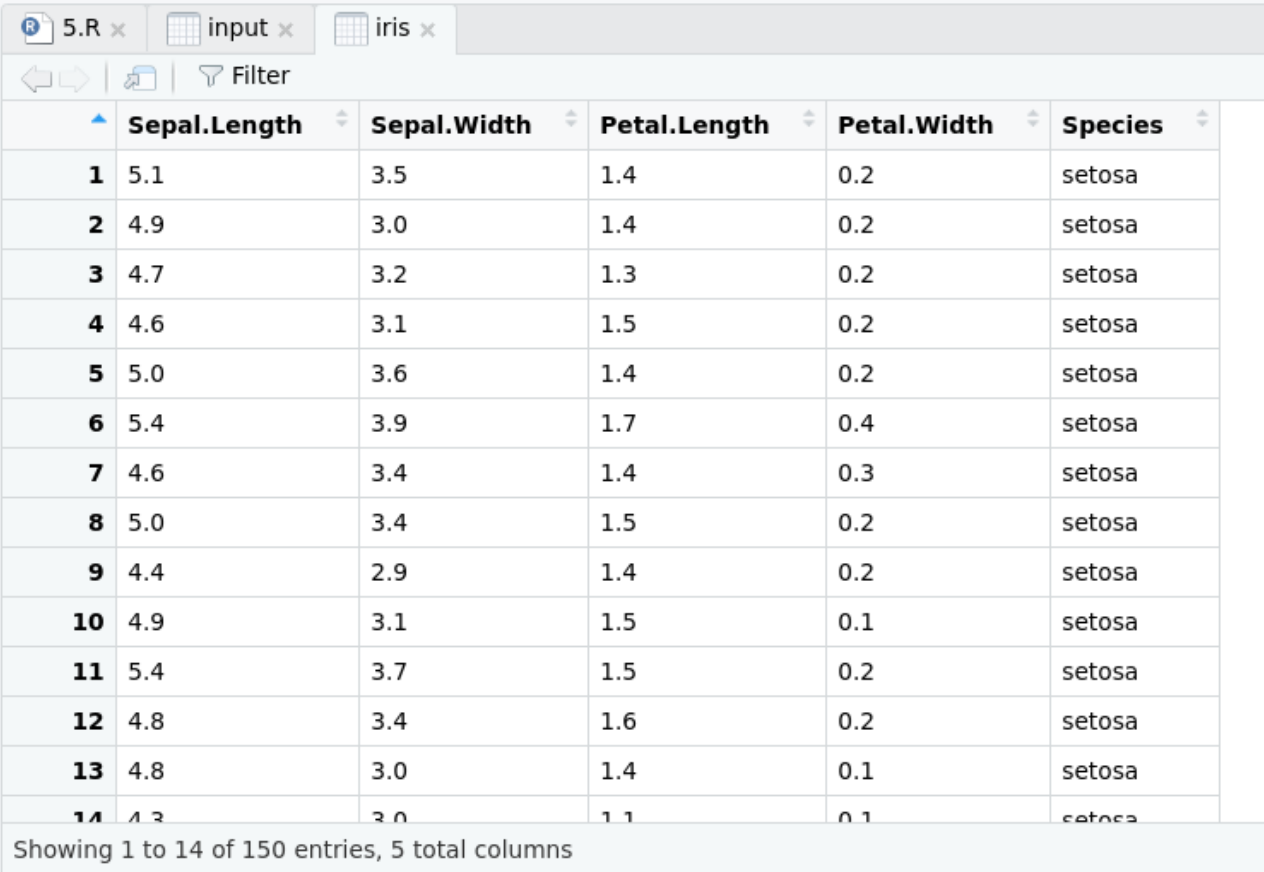
Konrad Boroń
Informatyka II stopień,
stacjonarne,
7 semestr,
Gr. 1A

1. Polecenie:

Zadanie dotyczy konstruowania drzew decyzyjnych oraz reguł klasyfikacyjnych na podstawie zbioru danych (library(MASS lub datasets)). Warianty zadania 1. iris

2. Wprowadzane dane:

Wczytany zbiór danych jest dostarczany razem z R. Jest to zbiór pomiarów długości różnych elementów kwiatów irysa, dostępny jako `iris`. Każdy pomiar jest opatrzony adnotacją dotyczącą przynależności gatunkowej.



| | 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 |
| 7 | 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 8 | 5.0 | 3.4 | 1.5 | 0.2 | setosa |
| 9 | 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 10 | 4.9 | 3.1 | 1.5 | 0.1 | setosa |
| 11 | 5.4 | 3.7 | 1.5 | 0.2 | setosa |
| 12 | 4.8 | 3.4 | 1.6 | 0.2 | setosa |
| 13 | 4.8 | 3.0 | 1.4 | 0.1 | setosa |
| 14 | 4.3 | 3.0 | 1.1 | 0.1 | setosa |

Showing 1 to 14 of 150 entries, 5 total columns

3. Wykorzystane komendy:

a) kod źródłowy A

```
if (!require("mlr")){ install.packages("mlr");  
  ↪ library("mlr")}  
if (!require("rpart")){ install.packages("rpart");  
  ↪ library("rpart")}  
if (!require("partykit")){ install.packages("partykit");  
  ↪ library("partykit")}  
if (!require("C50")){ install.packages("C50");  
  ↪ library("C50")}  
  
rtree <- rpart(Species ~ ., iris)
```

```

par(mar = rep(0,4))
plot(rtree)
text(rtree)

ctr <- ctree(Species ~ ., iris)
plot(ctr)

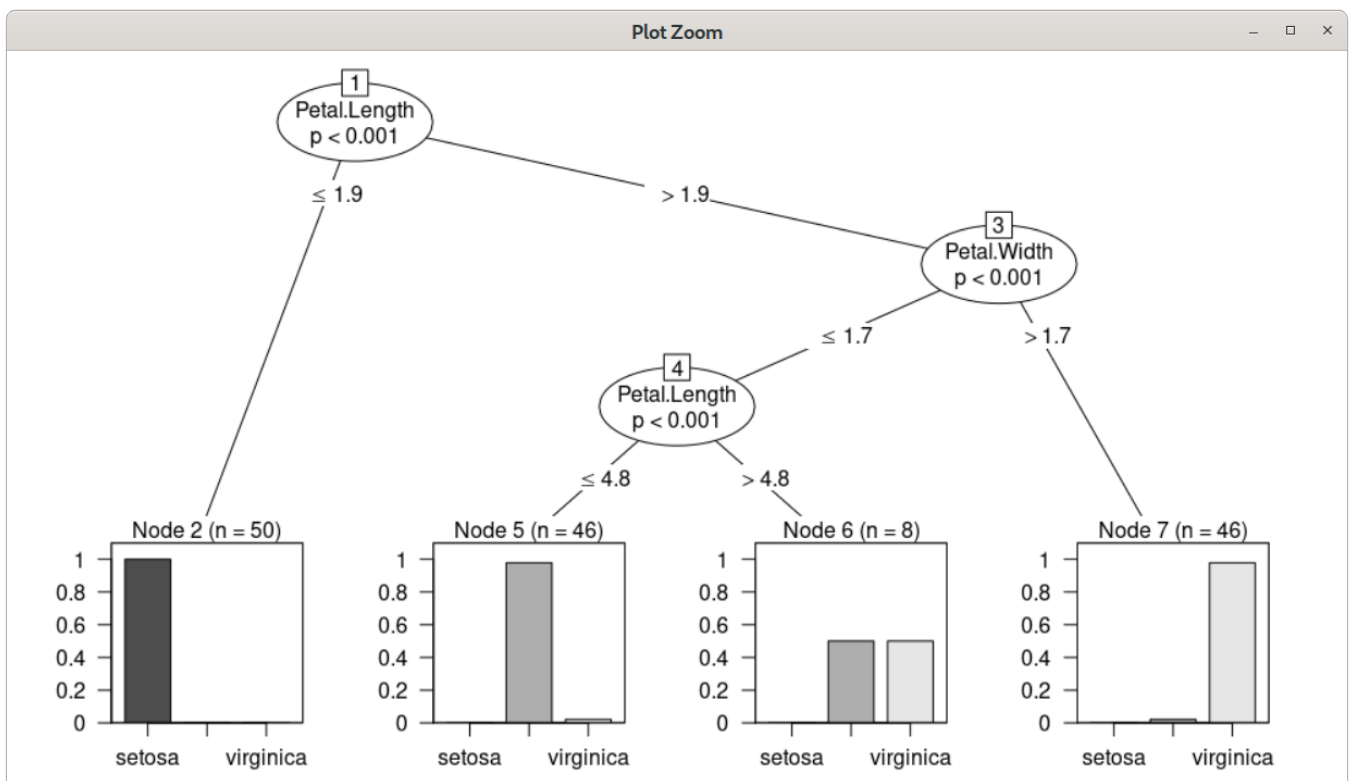
input <- subset(iris, select = -c(Species))
tree <- C5.0.default(x=input, y=iris$Species)

summary(tree)
plot(tree)

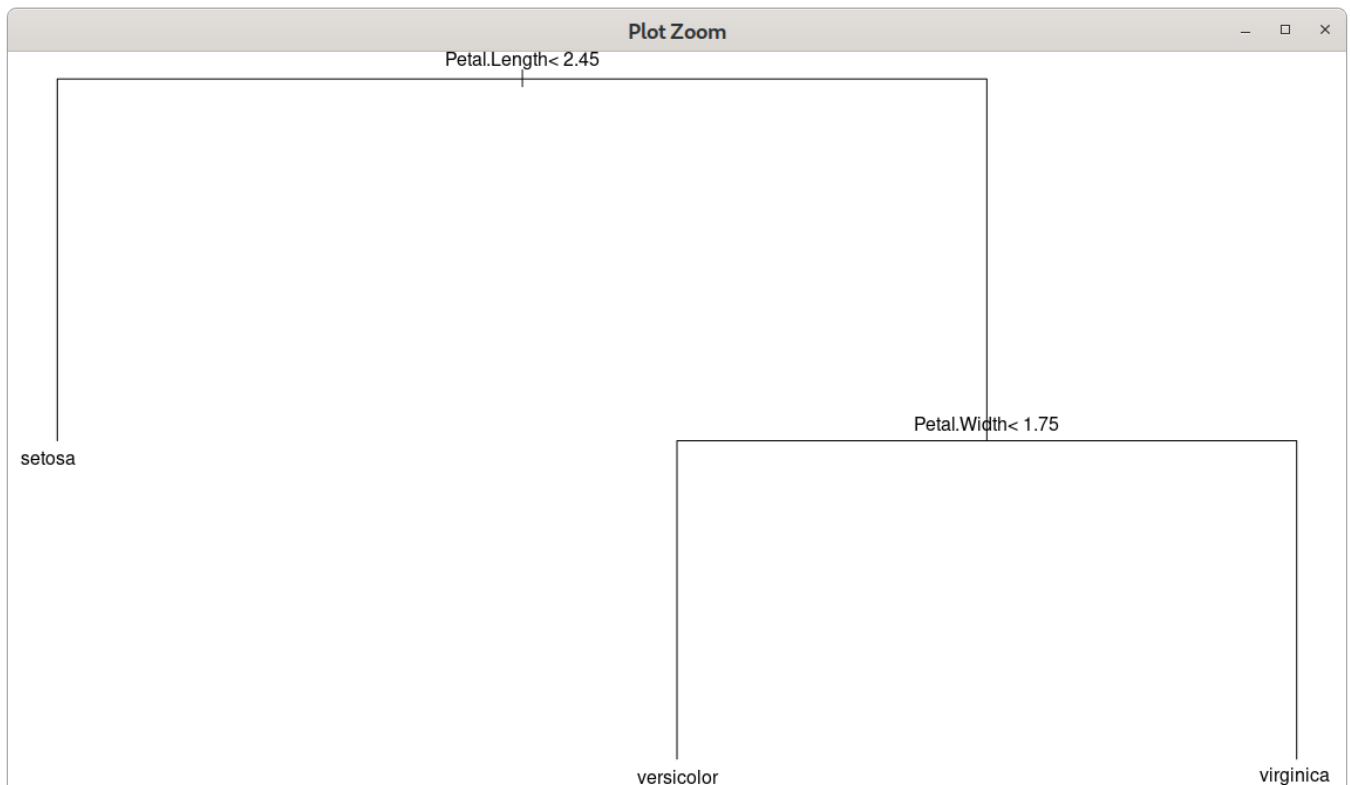
classif <- C5.0(Species ~ ., data = iris, rules = T)
summary(classif)

```

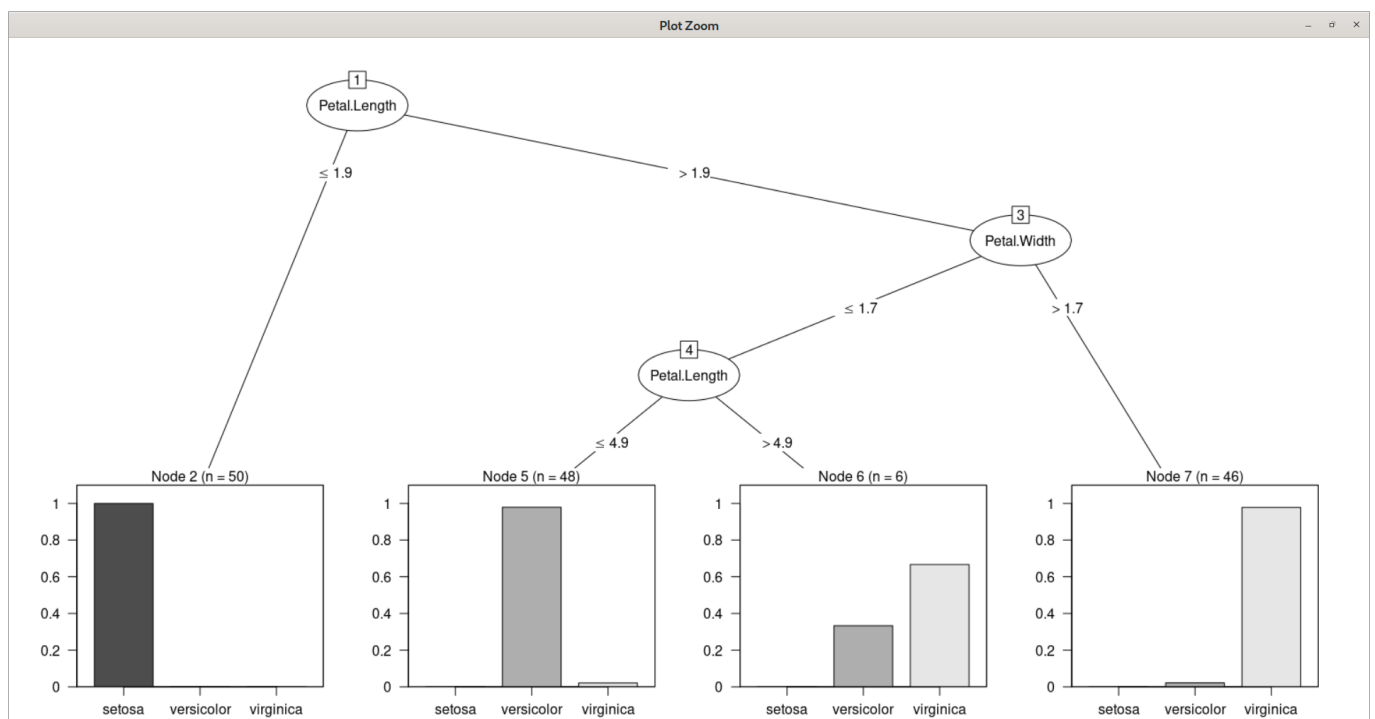
4. Wynik działania:



Rysunek 1: Drzewo decyzyjne z pakietu partykit(ctree())



Rysunek 2: Drzewo decyzyjne z pakietu rpart



Rysunek 3: Drzewo decyzyjne z pakietu C5.0

Reguły klasyfikacyjne z pakietu C5.0:

```
> summary(classif)
```

Call:

```
C5.0.formula(formula = Species ~ ., data = iris, rules = T)
```

C5.0 [Release 2.07 GPL Edition]

Sun Jun 14 18:18:07 2020

Class specified by attribute `outcome`

Read 150 cases (5 attributes) from undefined.data

Rules:

Rule 1: (50, lift 2.9)

Petal.Length <= 1.9

-> class setosa [0.981]

Rule 2: (48/1, lift 2.9)

Petal.Length > 1.9

Petal.Length <= 4.9

Petal.Width <= 1.7

-> class versicolor [0.960]

Rule 3: (46/1, lift 2.9)

Petal.Width > 1.7

-> class virginica [0.958]

Rule 4: (46/2, lift 2.8)

Petal.Length > 4.9

-> class virginica [0.938]

Default class: setosa

Evaluation on training data (150 cases):

| Rules | | |
|-------|----------|----|
| ----- | | |
| No | Errors | |
| 4 | 4(2.7%) | << |

| (a) | (b) | (c) | <-classified as |
|------|------|------|-----------------------|
| ---- | ---- | ---- | |
| 50 | | | (a): class setosa |
| | 47 | 3 | (b): class versicolor |
| | 1 | 49 | (c): class virginica |

Attribute usage:

96.00% Petal.Length
62.67% Petal.Width

Time: 0.0 secs

5. Wnioski:

Silnym parametrem klasyfikującym jest długość płatków kwiatu `Petal.length`. Wszystkie badane osobniki z gatunku *Setosa* miały długość mniejszą niż lub równą 1,9.

Biblioteka C5.0 tworzy minimalne drzewo decyzyjne nawet z dużej liczby osobników. Dzięki temu można znaleźć cechy charakterystyczne dla danej klasy.

Reguły klasyfikacyjne pokazują w jaki sposób można sklasyfikować dany gatunek, za pomocą zebranych parametrów. Dodatkowo jeśli jakaś reguła powoduje błędną klasyfikację, zostanie ona wyszczególniona, oraz jej błąd zostanie obliczony w procentach.

1. Polecenie:

Zadanie 2. Zadanie dotyczy prognozowania oceny klientów (w skali 5-punktowej, Error < 5%) urządzeń RTV AGD, określonych na Zajeciu 1. Rozwiązanie polega na użyciu pakietu mlr. Należy wybrać najlepszą metodę wśród 5 możliwych z punktu widzenia precyzyjności. Wyniki porównywania precyzyjności metod należy przedstawić w postaci graficznej.

2. Wprowadzane dane:

Lista smartfonów z zadania 1.

| | ↑ nazwa ↓ | wyświetlacz ↓ | pamięć_RAM ↓ | pamięć_wbudowana ↓ | aparat_foto ↓ | cena ↓ | liczba_opinii ↓ | ocena ↓ | status_opinii ↓ |
|----|-------------------------|---------------|--------------|--------------------|---------------|--------|-----------------|---------|-----------------------|
| 1 | Galaxy J2 Core (2020) | 5.00 | 1 | 16 | 8 | 80 | 17 | 3 | mniej niż 50 opinii |
| 2 | Galaxy Xcover FieldPro | 5.10 | 4 | 64 | 12 | 1020 | 48 | 5 | mniej niż 50 opinii |
| 3 | Galaxy A2 Core | 5.00 | 1 | 8 | 5 | 120 | 36 | 5 | mniej niż 50 opinii |
| 4 | Galaxy View2 | 17.30 | 3 | 64 | 0 | 660 | 50 | 4 | 50-100 opinii |
| 5 | Galaxy M30 | 6.40 | 3 | 32 | 13 | 300 | 316 | 4 | więcej niż 100 opinii |
| 6 | Galaxy M20 | 6.30 | 3 | 32 | 13 | 300 | 358 | 4 | więcej niż 100 opinii |
| 7 | Galaxy M10 | 6.22 | 2 | 16 | 13 | 135 | 107 | 4 | więcej niż 100 opinii |
| 8 | Galaxy Tab Advanced2 | 10.10 | 3 | 32 | 8 | 200 | 8 | 5 | mniej niż 50 opinii |
| 9 | Galaxy Tab A 8.0 (2018) | 8.00 | 2 | 32 | 8 | 130 | 40 | 4.5 | mniej niż 50 opinii |
| 10 | Galaxy A6s | 6.00 | 6 | 64 | 12 | 300 | 86 | 5 | 50-100 opinii |
| 11 | Galaxy A9 (2018) | 6.30 | 6 | 64 | 24 | 359 | 320 | 4.5 | więcej niż 100 opinii |
| 12 | Galaxy A7 (2018) | 6.00 | 4 | 64 | 12 | 309 | 223 | 4 | więcej niż 100 opinii |
| 13 | Galaxy Note9 | 6.40 | 6 | 128 | 12 | 820 | 1243 | 5 | więcej niż 100 opinii |
| 14 | Galaxy J6+ | 6.00 | 3 | 32 | 10 | 230 | 198 | 4 | więcej niż 100 opinii |
| 15 | Galaxy J4 Core | 6.00 | 1 | 16 | 5 | 150 | 87 | 3.5 | 50-100 opinii |

3. Wykorzystane komendy:

a) kod źródłowy A

```
if (!require("mlr")){ install.packages("mlr"); library("mlr")}
if (!require("rFerns")){ install.packages("rFerns");
  ↪ library("rFerns")}
if (!require("randomForestSRC")){
  ↪ install.packages("randomForestSRC")}
```

```
load("../1/ramka_smartfony")
```

```
input <- data.frame(name = ramka$nazwa,
  display = ramka$wyświetlacz,
  RAM = ramka$pamięć_RAM,
  ROM = ramka$pamięć_wbudowana,
  camera = ramka$aparat_foto,
  price = ramka$cena,
  grade = ramka$ocena)
```

```
task = makeClassifTask(id = deparse(substitute(input)), input,
  ↪ "grade",
  weights = NULL, blocking = NULL, coordinates =
  ↪ NULL,
  positive = NA_character_, fixup.data = "warn",
  ↪ check.data = TRUE)
```

```
lrns <- makeLearners(c("rpart", "C50","rFerns", "randomForestSRC"),
  ↪ type = "classif")
resamp <- makeResampleDesc(method = "CV", iters = 2)
porownanie <- benchmark(learners = lrns,
  tasks = task,
  resampling = resamp)
```

```
porownanie
```

```
plotBMRSummary(porownanie)
plotBMRBoxplots(porownanie)
```

4. Wynik działania:

```
> porownanie <- benchmark(learners = lrns,
+                          tasks = task,
+                          resampling = resamp)
Task: input, Learner: classif.rpart
Resampling: cross-validation
Measures:          mmce
[Resample] iter 1:  0.6250000
[Resample] iter 2:  0.5714286
```

Aggregated Result: mmce.test.mean=0.5982143

```
Task: input, Learner: classif.C50
Resampling: cross-validation
Measures:          mmce
[Resample] iter 1:  0.7500000
[Resample] iter 2:  0.5714286
```

Aggregated Result: mmce.test.mean=0.6607143

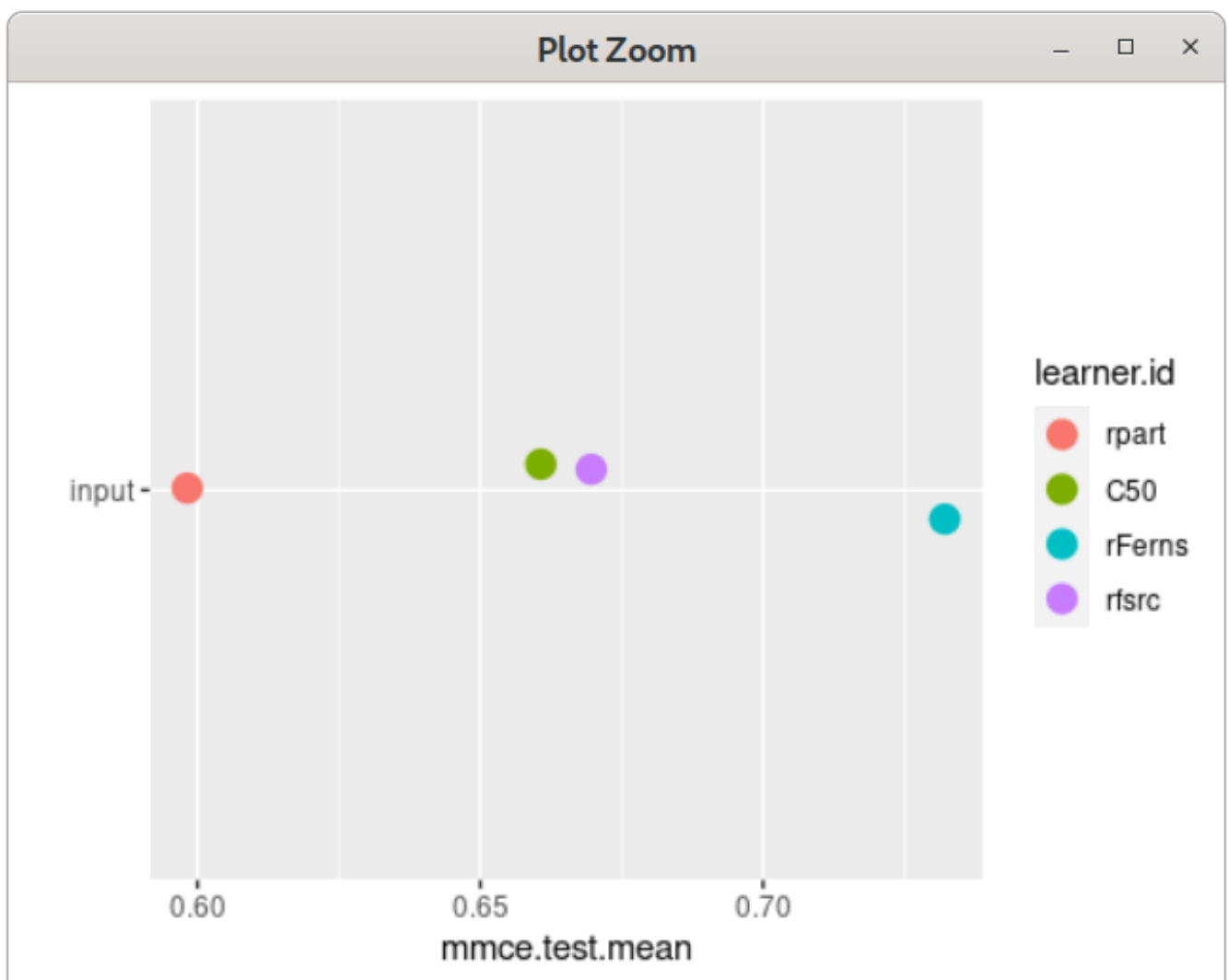
```
Task: input, Learner: classif.rFerns
Resampling: cross-validation
Measures:          mmce
[Resample] iter 1:  0.7500000
[Resample] iter 2:  0.7142857
```

Aggregated Result: mmce.test.mean=0.7321429

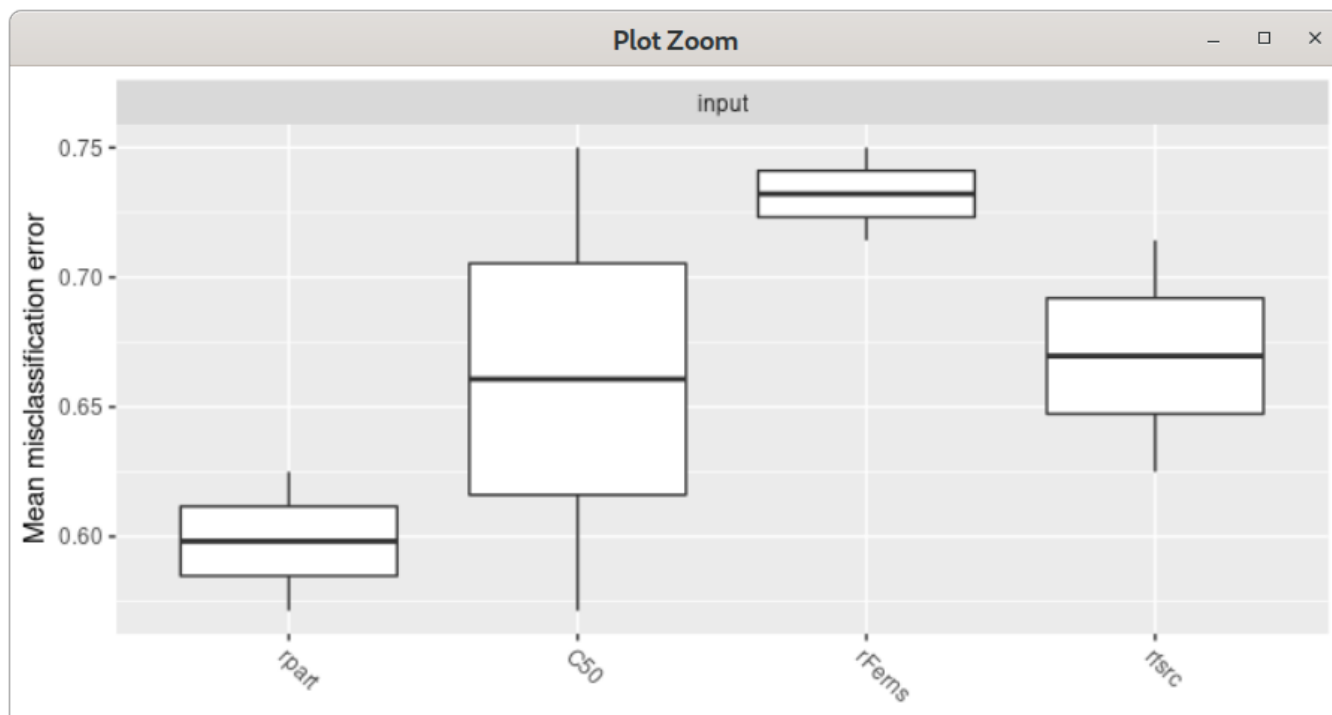
```
Task: input, Learner: classif.randomForestSRC
Resampling: cross-validation
Measures:          mmce
[Resample] iter 1:  0.6250000
[Resample] iter 2:  0.7142857
```


Aggregated Result: mmce.test.mean=0.6696429

```
> porownanie
  task.id      learner.id mmce.test.mean
1  input      classif.rpart      0.5982143
2  input      classif.C50       0.6607143
3  input      classif.rFerns    0.7321429
4  input      classif.randomForestSRC 0.6696429
```



Rysunek 4: Wykres średniego błędu



Rysunek 5: Wykres skrzynkowy średniego błędu

5. Wnioski:

Na podstawie otrzymanego wyniku można stwierdzić, że metodą, która klasyfikuje oceny klientów z najmniejszym średnim błędem jest `rpart`.