Sprawozdanie

Zajęcia: Analiza procesów uczenia
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium 1
Data: 26.04.2024

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

Wariant: 1

Agnieszka Białecka
Informatyka II stopień,
stacjonarne,
1 semestr,
Gr.1a

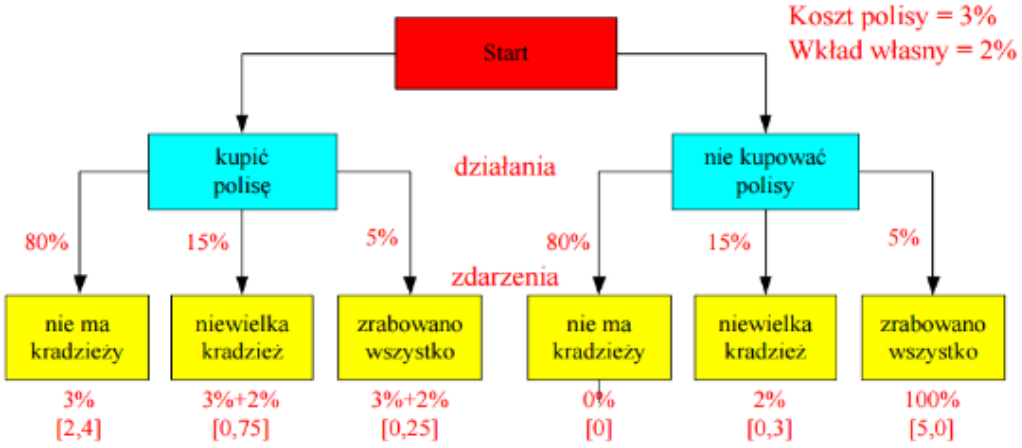
1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z konceptem uczenia maszynowego przy pomocy pakietu mlr.

2. Wstęp teoretyczny

Drzewa decyzyjne stanowią model decyzyjny, w którym w uporządkowany sposób przedstawia się jako hierarchiczne ciągi działań i zdarzeń. Graficzne przedstawienie w postaci drzewa decyzyjnego ułatwia analizę wszystkich elementów sytuacji istotnych przy podejmowaniu decyzji. W efekcie możliwe staje się określenie wariantów decyzyjnych i ich konsekwencji. W modelu tym nie występują tu w jawnej postaci warunki sztywne i elastyczne, są one uwzględniane w trakcie budowy drzewa. Dodatkowe podanie prawdopodobieństw i kosztów poszczególnych wariantów decyzyjnych prowadzi do zwiększenia racjonalności optymalizacyjnej poprzez maksymalizację funkcji użyteczności.

Celem stosowania modelu w postaci drzewa decyzyjnego jest uproszczenie oceny sytuacji decyzyjnej, model ten pozwala na jednoczesną analizę wielu wariantów decyzyjnych i kryteriów ich oceny. Model taki jest użyteczny, o ile drzewo nie staje się zbyt obszerne (nie mieści się na kartce lub ekranie). Z wykorzystaniem drzew decyzyjnych może być prowadzona analiza wielowariantowa (what-if analysis), a poprzez implementację programową możliwe jest zastosowanie tego modelu w komputerowych systemach wspomagania decyzji.



Zdjęcie 1. Drzewo decyzyjne dotyczące decyzji o zakupie polisy na ubezpieczenie mieszkania

Powyżej (Zdjęcie 2) rozważana jest sytuacja związana z ubezpieczeniem mieszkania, przy założeniu kosztów polisy w wysokości 3% oraz wkładu własnego w wysokości 2% wartości mieszkania. Możliwym zdarzeniom (brak kradzieży, kradzież niewielka – nie przekraczająca wkładu własnego oraz kradzież pełna) przypisano prawdopodobieństwa ich wystąpienia (odpowiednio 80%, 15% i 5%).

Zarówno wydatki związane z zakupem polisy, z ponoszeniem wkładu własnego, jak i rekompensatą za skradzione wyposażenie mieszkania (w przypadku rezygnacji z zakupu polisy) traktowane są jako stratą którą należy zminimalizować.

Wartość oczekiwaną straty związanej z daną decyzją można obliczyć z zależności (1.2) wprowadzonej przy okazji omawiania strategii scalania prawdopodobieństw i użyteczności, przy czym tutaj użytecznością (negatywną– strata) będzie koszt poniesiony przy danym wariancie decyzyjnym dla poszczególnych zdarzeń. Wartości iloczynów pi(sk) ui(sk) umieszczono w nawiasach kwadratowych pod zdarzeniami na Zdjęciu 3 oczekiwana strata dla poszczególnych decyzji wynosi:

$$SPU(d_i) = \begin{cases} 3.4, \text{ dla decyzji kupować polisę} \\ 5.3, \text{ dla decyzji nie kupować polisę} \end{cases}$$

a zatem właściwą decyzją będzie zakup polisy ubezpieczeniowej.

3. Przebieg ćwiczenia

Repozytorium GitHub:

https://github.com/Delisolara/APU

3.1. Zadanie 1

Zadanie dotyczy konstruowania drzew decyzyjnych oraz regul klasyfikacyjnych na podstawie zbioru danych (library (MASS lub datasets)).

Wariant: iris

```r
install.packages("C50")
library(MASS)
require(C50)

data("iris")
head(iris)

str(iris)

iris$Species <- as.factor(iris$Species)
str(iris)
table(iris$Species)

set.seed(123)
train_indices <- sample(1:nrow(iris), 0.8 * nrow(iris))
train_data <- iris[train_indices, ]
test_data <- iris[-train_indices, ]

m1 <- C5.0(Species ~ ., data=train_data)
summary(m1)

plot(m1)

predictions <- predict(m1, test_data)

confusion_matrix <- table(test_data$Species, predictions)
print(confusion_matrix)
```

Zdjęcie 4. Skrypt

```
> data("iris")
> 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
```

Zdjęcie 5. Wynik konsoli

```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> iris$Species <- as.factor(iris$Species)
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Zdjęcie 6. Wynik konsoli

```
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
```

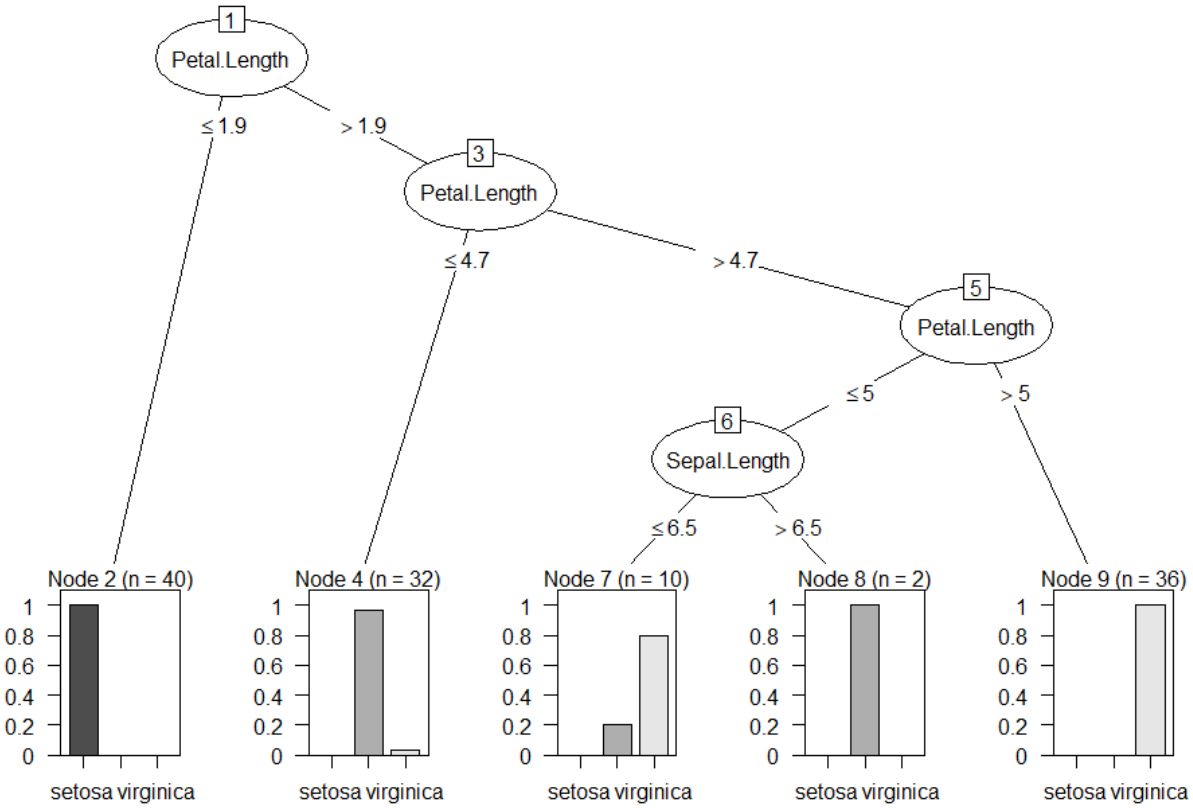
Zdjęcie 7. Wynik konsoli

```
> summary(m1)

Call:
C5.0.formula(formula = Species ~ ., data = train_data)


C5.0 [Release 2.07 GPL Edition]         Mon May 20 13:43:48 2024
-----------------------------

Class specified by attribute `outcome'

Read 120 cases (5 attributes) from undefined.data

Decision tree:

Petal.Length <= 1.9: setosa (40)
Petal.Length > 1.9:
:...Petal.Length <= 4.7: versicolor (32/1)
    Petal.Length > 4.7:
    :...Petal.Length > 5: virginica (36)
        Petal.Length <= 5:
        :...Sepal.Length <= 6.5: virginica (10/2)
            Sepal.Length > 6.5: versicolor (2)
```

Zdjęcie 8. Wynik konsoli

```
Evaluation on training data (120 cases):

          Decision Tree
        ----------------
        Size      Errors

           5    3( 2.5%)   <<


         (a)   (b)   (c)    <-classified as
        ----  ----  ----
          40                (a): class setosa
                33     2    (b): class versicolor
                 1    44    (c): class virginica


       Attribute usage:

       100.00% Petal.Length
        10.00% Sepal.Length


Time: 0.0 secs
```

Zdjęcie 9. Wynik konsoli



Zdjęcie 10. Wizualizacja

```
> predictions <- predict(m1, test_data)
> confusion_matrix <- table(test_data$Species, predictions)
> print(confusion_matrix)
            predictions
             setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         14         1
  virginica       0          0         5
```

Zdjęcie 11. Wynik konsoli

3.2. Zadanie 2

Zadanie dotyczy prognozowania oceny klientów (w skali 5-punktowej, Error < 5%) urządzeń RTV AGD, określonych na Zajęciach 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.

```
install.packages("randomForest")
install.packages("e1071")
install.packages("party")
install.packages("mlr")
install.packages("rFerns")

library(randomForest)
library(e1071)
library(party)
library(mlr)
library(rFerns)

df=read.csv("D:/Studia/APU/smartfony.csv")
df <- df [1:8]
df$nazwa = factor(df$nazwa)
df$ocena_klientow = factor(df$ocena_klientow)

summarizeColumns(df)

rdesc = makeResampleDesc("CV", iters = 10)

task = makeClassifTask(id = deparse(substitute(df)), df, "ocena_klientow",
                       weights = NULL, blocking = NULL, coordinates = NULL,
                       positive = NA_character_, fixup.data = "warn", check.data = TRUE)
lrns <- makeLearners(c("rpart", "C50", "ctree", "naiveBayes", "randomForest"), type = "classif")

bmr <- benchmark(learners = lrns, tasks = task, rdesc, models = TRUE, measures = list(acc, ber))
p = getBMRPredictions(bmr)
plotBMRSummary(bmr)
```

Zdjęcie 12. Skrypt

```
> df=read.csv("D:/Studia/APU/smartfony.csv")
> df <- df [1:8]
> df$nazwa = factor(df$nazwa)
> df$ocena_klientow = factor(df$ocena_klientow)
> summarizeColumns(df)
              name    type na      mean        disp median       mad min     max nlevs
1            nazwa  factor  0        NA   0.9500000     NA        NA   1     1.0    20
2     wyĹ.wietlacz numeric  0     6.375   0.1996708    6.4   0.22239   6     6.7     0
3       pamiec_RAM integer  0     6.500   1.2773327    6.0   0.00000   4     8.0     0
4 pamiec_wbudowana integer  0   144.000  71.5541753  128.0  47.44320  64   256.0     0
5      aparat_foto integer  0    68.600  24.3924061   64.0  23.72160  48   108.0     0
6             cena integer  0  1065.000 220.7046133 1000.0 148.26000 700  1500.0     0
7    liczba_opinii integer  0   183.900  48.1990391  185.0  37.06500  18   250.0     0
8   ocena_klientow  factor  0        NA   0.8500000     NA        NA   1     3.0    11
```

Zdjęcie 13. Wynik konsoli

```
> rdesc = makeResampleDesc("CV", iters = 10)
> task = makeClassifTask(id = deparse(substitute(df)), df, "ocena_klientow",
+                        weights = NULL, blocking = NULL, coordinates = NULL,
+                        positive = NA_character_, fixup.data = "warn", check.data = TRUE)
> lrns <- makeLearners(c("rpart", "C50", "ctree", "naiveBayes", "randomForest"), type = "classif")
> bmr <- benchmark(learners = lrns, tasks = task, rdesc, models = TRUE, measures = list(acc, ber))
Task: df, Learner: classif.rpart
Resampling: cross-validation
Measures:             acc       ber
[Resample] iter 1:    0.0000000      NaN
[Resample] iter 2:    0.0000000      NaN
[Resample] iter 3:    0.0000000      NaN
[Resample] iter 4:    0.0000000      NaN
[Resample] iter 5:    0.0000000      NaN
[Resample] iter 6:    0.0000000      NaN
[Resample] iter 7:    0.0000000      NaN
[Resample] iter 8:    0.0000000      NaN
[Resample] iter 9:    0.0000000      NaN
[Resample] iter 10:   0.0000000      NaN


Aggregated Result: acc.test.mean=0.0000000,ber.test.mean=     NaN
```

Zdjęcie 14. Wynik konsoli

```
Task: df, Learner: classif.C50
Resampling: cross-validation
Measures:             acc       ber
[Resample] iter 1:    0.5000000      NaN
[Resample] iter 2:    1.0000000      NaN
[Resample] iter 3:    0.5000000      NaN
[Resample] iter 4:    0.0000000      NaN
[Resample] iter 5:    0.0000000      NaN
[Resample] iter 6:    0.0000000      NaN
[Resample] iter 7:    0.5000000      NaN
[Resample] iter 8:    0.5000000      NaN
[Resample] iter 9:    0.5000000      NaN
[Resample] iter 10:   0.5000000      NaN


Aggregated Result: acc.test.mean=0.4000000,ber.test.mean=     NaN
```

Zdjęcie 15. Wynik konsoli

```
Task: df, Learner: classif.ctree
Resampling: cross-validation
Measures:             acc       ber
[Resample] iter 1:    0.0000000      NaN
[Resample] iter 2:    0.0000000      NaN
[Resample] iter 3:    0.0000000      NaN
[Resample] iter 4:    0.0000000      NaN
[Resample] iter 5:    0.0000000      NaN
[Resample] iter 6:    0.0000000      NaN
[Resample] iter 7:    0.0000000      NaN
[Resample] iter 8:    0.0000000      NaN
[Resample] iter 9:    0.0000000      NaN
[Resample] iter 10:   0.0000000      NaN


Aggregated Result: acc.test.mean=0.0000000,ber.test.mean=     NaN
```

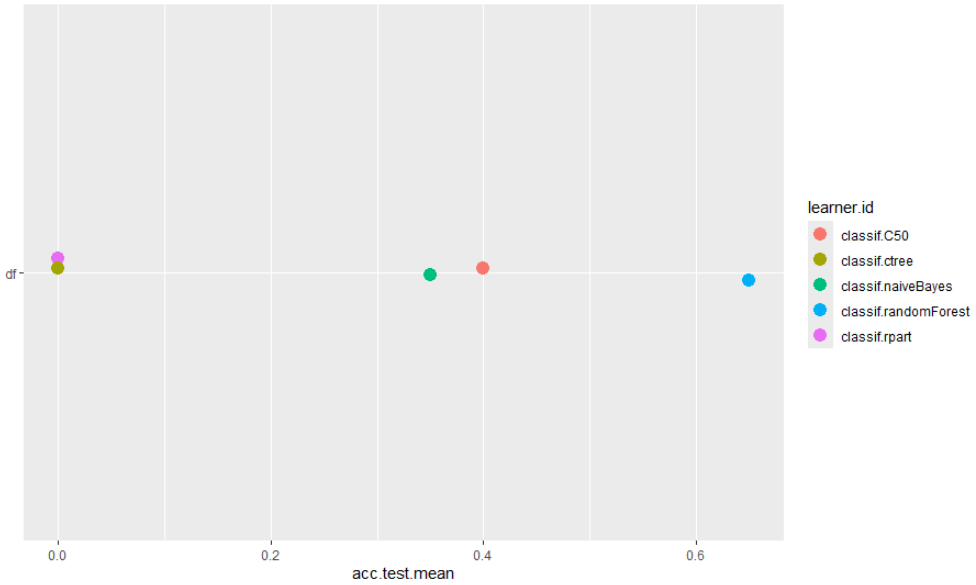
Zdjęcie 16. Wynik konsoli

```
Task: df, Learner: classif.naiveBayes
Resampling: cross-validation
Measures:             acc       ber
[Resample] iter 1:    0.0000000      NaN
[Resample] iter 2:    1.0000000      NaN
[Resample] iter 3:    0.5000000      NaN
[Resample] iter 4:    0.0000000      NaN
[Resample] iter 5:    0.0000000      NaN
[Resample] iter 6:    0.0000000      NaN
[Resample] iter 7:    0.5000000      NaN
[Resample] iter 8:    0.5000000      NaN
[Resample] iter 9:    0.5000000      NaN
[Resample] iter 10:   0.5000000      NaN


Aggregated Result: acc.test.mean=0.3500000,ber.test.mean=     NaN
```

Zdjęcie 17. Wynik konsoli

```
Task: df, Learner: classif.randomForest
Resampling: cross-validation
Measures:             acc       ber
[Resample] iter 1:    1.0000000      NaN
[Resample] iter 2:    1.0000000      NaN
[Resample] iter 3:    0.5000000      NaN
[Resample] iter 4:    1.0000000      NaN
[Resample] iter 5:    0.5000000      NaN
[Resample] iter 6:    0.0000000      NaN
[Resample] iter 7:    0.5000000      NaN
[Resample] iter 8:    0.5000000      NaN
[Resample] iter 9:    0.5000000      NaN
[Resample] iter 10:   1.0000000      NaN


Aggregated Result: acc.test.mean=0.6500000,ber.test.mean=     NaN
```

Zdjęcie 18. Wynik konsoli



Zdjęcie 19. Wizualizacja

4. Podsumowanie

Przeprowadzone ćwiczenie umożliwiło zapoznanie się z konceptami modelowania procesów uczenia maszynowego.