Politechnika Warszawska

Wydział Mechatroniki

PROJEKT

Podstawy Machine Learning w R

Prowadzący:
dr inż. Marcel Młyńczak

Jarosław Affek

Warszawa 2019/2020

1. Dane

W projekcie wykorzystano dane z bazy *Bank Marketing*, zawierającej decyzje o subskrypcji bankowych informacji marketingowych oraz szereg cech środowiskowych, stworzonej przez Sérgio Moro, Raula Laureano oraz Paulo Corteza.

Opis kolumn wejściowych:

- (1) age - wiek - [integer]
- (2) job - zawód - [STRING: admin., blue-collar, entrepreneur, housemaid, management, retired, self-employed, services, student, technician, unemployed, unknown]
- (3) marital - stan cywilny - [STRING: divorced, married, single, unknown]
- (4) education - wykształcenie - [STRING: primary, secondary, tertiary, unknown]
- (5) default - zaległy/niespłacony kredyt - [STRING: no, yes, unknown]
- (6) balance - średnie roczne saldo w Euro - [INTEGER]
- (7) housing - posiada kredyt mieszkaniowy - [STRING: no, yes, unknown]
- (8) loan - posiada kredyt osobisty - [STRING: no, yes, unknown]
- (9) contact - forma kontaktu - [STRING: unknown, telephone, cellular]
- (10) day - dzień miesiąca - [INTEGER: 1 - 31]
- (11) month - miesiąc - [STRING: jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec]
- (12) duration - czas trwania ostatniej rozmowy/kontaktu - [INTEGER: value in seconds]
- (13) campaign - liczba wykonanych rozmów w ramach danej kampanii - [INTEGER]
- (14) pdays - liczba dni jaka upłynęła od ostatniego kontaktu z klientem w ramach poprzedniej kampanii - [FLOAT: -1 dla braku wcześniejszego kontaku]
- (15) previous - liczba nawiązanych kontaktów z danym klientem w ramach poprzednich kampanii - [INTEGER]
- (16) poutcome - wynik poprzedniej kampanii - [STRING: unknown, other, failure, success]

Brak wartości brakujących w bazie danych.

Klasy wyjściowe:

- yes
- no

Łącznie w bazie jest 4521 wyników, a ich rozkład jest następujący:

- yes - 521 - 12%
- no - 4000 - 88%

2. Cel projektu i zadania

- wybór, opis i wczytanie danych do analizy
- przeprowadzenie wizualnej analizy eksploracyjnej na podstawie wybranych wykresów
- zastosowanie metod radzenia sobie z niezbalansowaniemn danych
- klasyfikacja klasy wyjściowej, przy podziale na podzbiór trenujący i testowy w proporcji 70%/30% z wykorzystaniem różnych modeli
- próby poprawienia wyników klasyfikacji

3. Wczytanie danych

Za pomocą poniższego kodu, wczytano bazę danych w formacie .csv do środowiska RStudio. Baza została spreparowana w taki sposób, że wszystkie dane typu *string* zostały przedstawione jako całkowite wartości liczbowe w określonym zakresie. Ustawiono przecinek jako separator kolumn, zgodnie z formatem bazy danych.

```
separator = ","
dec = ”.”
data <- read.csv("C:/User/dane.csv", header=TRUE, sep=separator,
dec=dec, stringsAsFactors=FALSE)
```

4. Przygotowanie danych do analizy

Przygotowanie danych do analizy obejmowało stworzenie nagłówków dla poszczególnych kolumn

```
colnames(data) <- c("age", "job", "marital", "education",
"has_credit", "balance", "housing_loan", "personal_loan",
"contact", "day", "month", "duration", "campaign", "pdays",
"previous", "poutcome", "Diagnosis")
```

A następnie zamianę danych wyjściowych na typ faktorowy:

```
data <- data.frame(lapply(data,as.numeric))
data$Diagnosis <- as.factor(data$Diagnosis)
levels(data$Diagnosis) <- c("no","yes")
```

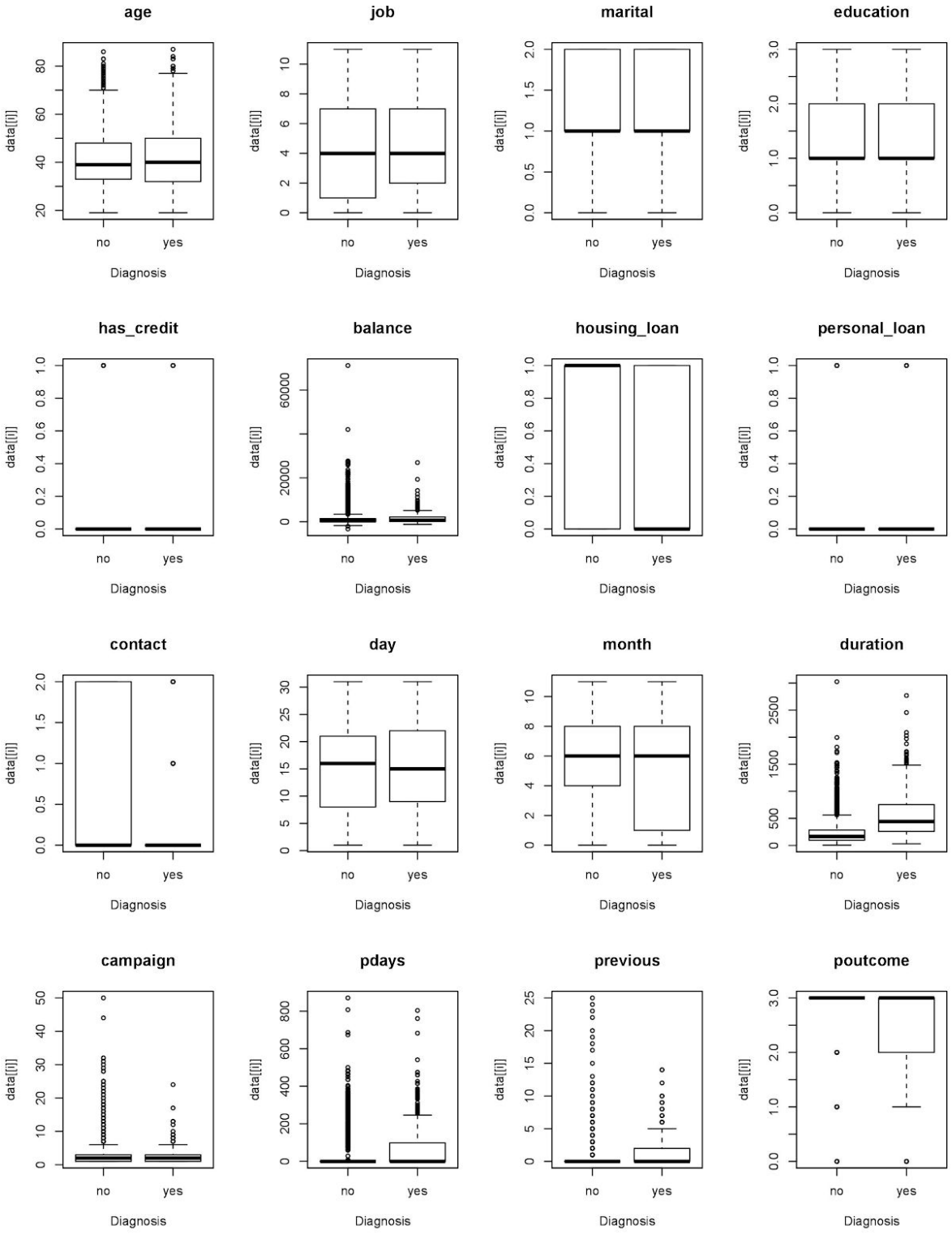
5. Wizualna analiza eksploracyjna

Aby zbadać analizowane dane i podstawowe zależności między nimi wykonano kilka typów wykresów.

Wizualizacja w formie box-plot

```
par(mfrow=c(4,4))
for(i in 1:16){
boxplot(data[[i]] ~ Diagnosis, data = data,
main=names(data)[i])}
rm(i)
```

Wykonano wykres typu box-plot dla wszystkich atrybutów wejściowych, jednak analiza tego typu wykresu ma sens tylko dla cech reprezentowanych liczbowo (mniejszość w przypadku rozważanych danych). Cechami takimi w przypadku wybranej bazy danych są: *age, balance, duration, campaign, pdays, previous*. Z powyższych cech widać, że największe różnice pomiędzy klasami wyjściowymi występują dla atrybutu *duration*, dla którego można zaobserwować również dużo obserwacji odstających, które w późniejszym etapie mogą wpływać na wyniki analizy. Dla pozostałych cech liczbowych różnice dla klasy wyjściowej są bardzo małe.

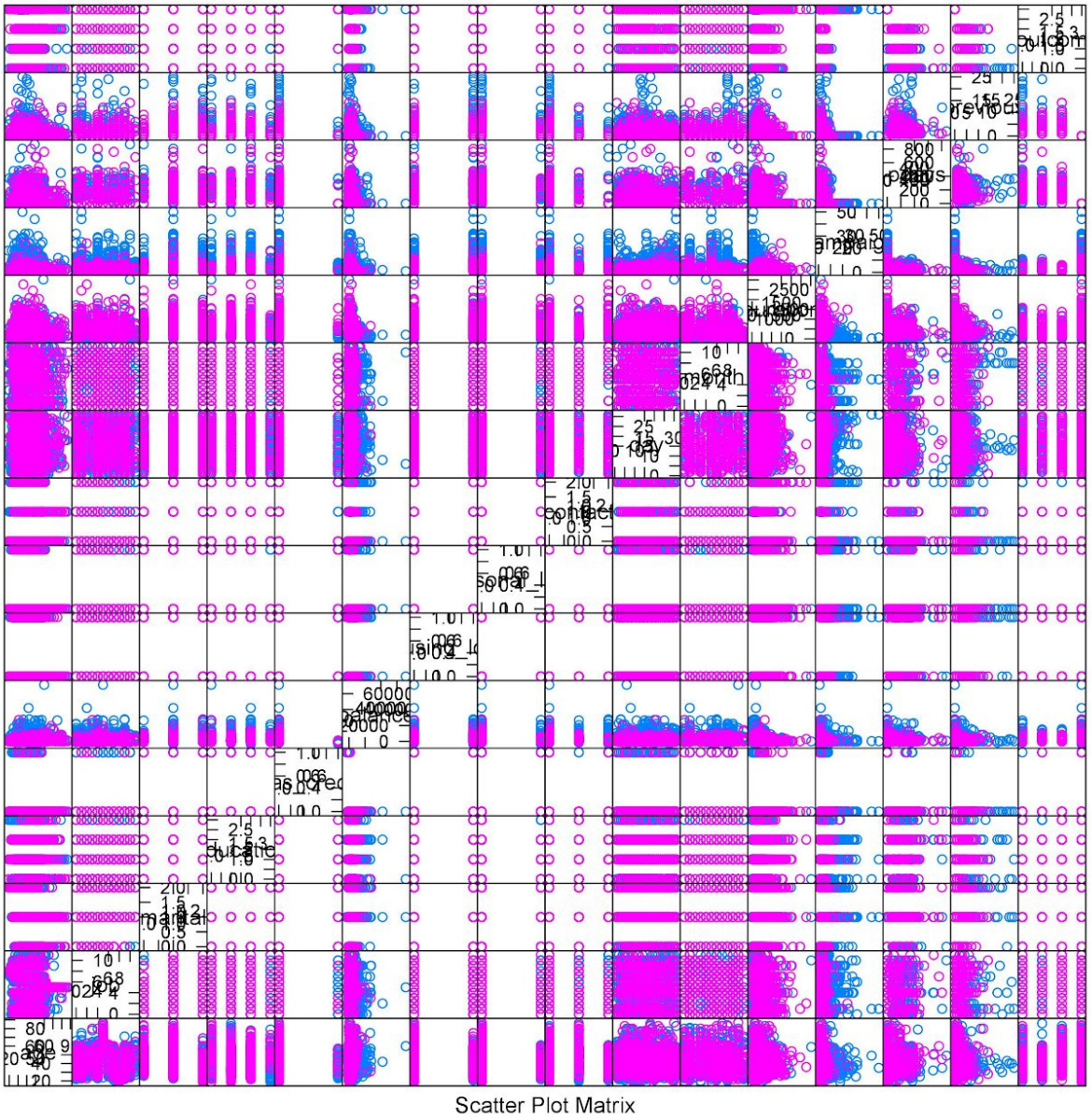


Rys. 1 Box-plot

Wykres korelogramu, podobnie jak wykres box-plot ma większe zastosowanie w przypadku danych liczbowych dlatego nie był on analizowany.

Wizualizacja z wykorzystaniem funkcji featurePlot

```
par(mfrow=c(4,4))
for(p in 1:16){
caret::featurePlot(data[,p],data$Diagnosis,plot="density")}
rm(p)
```



Rys. 2 Scatter Plot

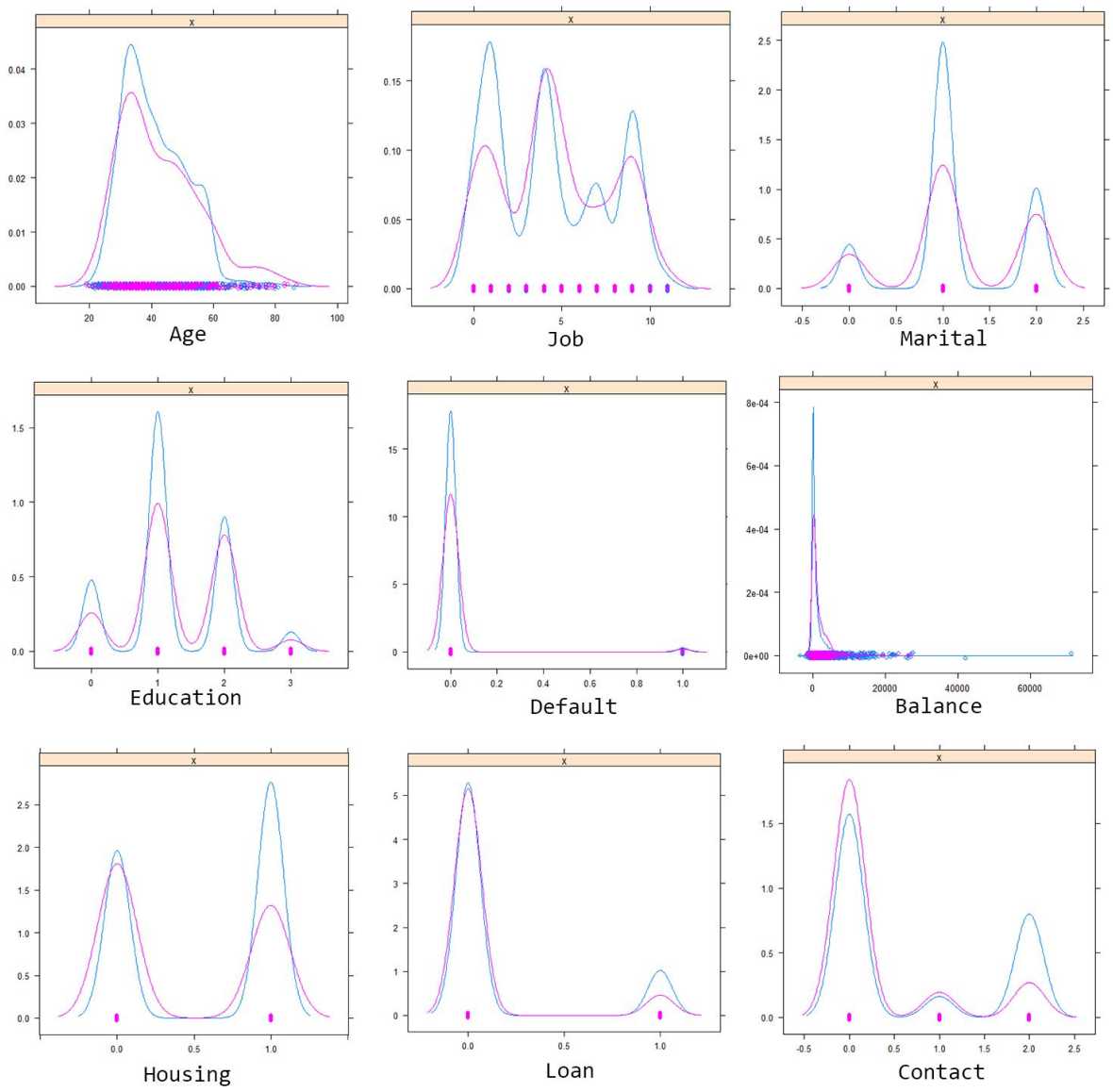
Kolejnym stworzonym wykresem był tzw. *scatter plot*, pokazujący zależności między każdymi dwoma atrybutami wejściowymi oraz klasą wyjściową. Ze względu

na dużą liczność atrybutów wejściowych wykres nie jest zbyt czytelny, jednak kolejność atrybutów w osi X, pokrywa się z kolejnością przedstawioną w punkcie pierwszym projektu (*Dane*). Ten wykres również lepiej odzwierciedla dane liczbowe niż dane w postaci kategorii. Jednak z powyższego wykresu można wyciągnąć wnioski takie, że największa rozbieżność stanów wyjściowych występuje dla atrybutów *campaign*. Jednakże analiza takiego wykresu nie przyniesie dużo konstruktywnych wniosków, gdyż liczba próbek jest na tyle duża, że przykrywają się one wzajemnie, co uniemożliwia rzetelną ocenę rozłożenia stanów wyjścia.

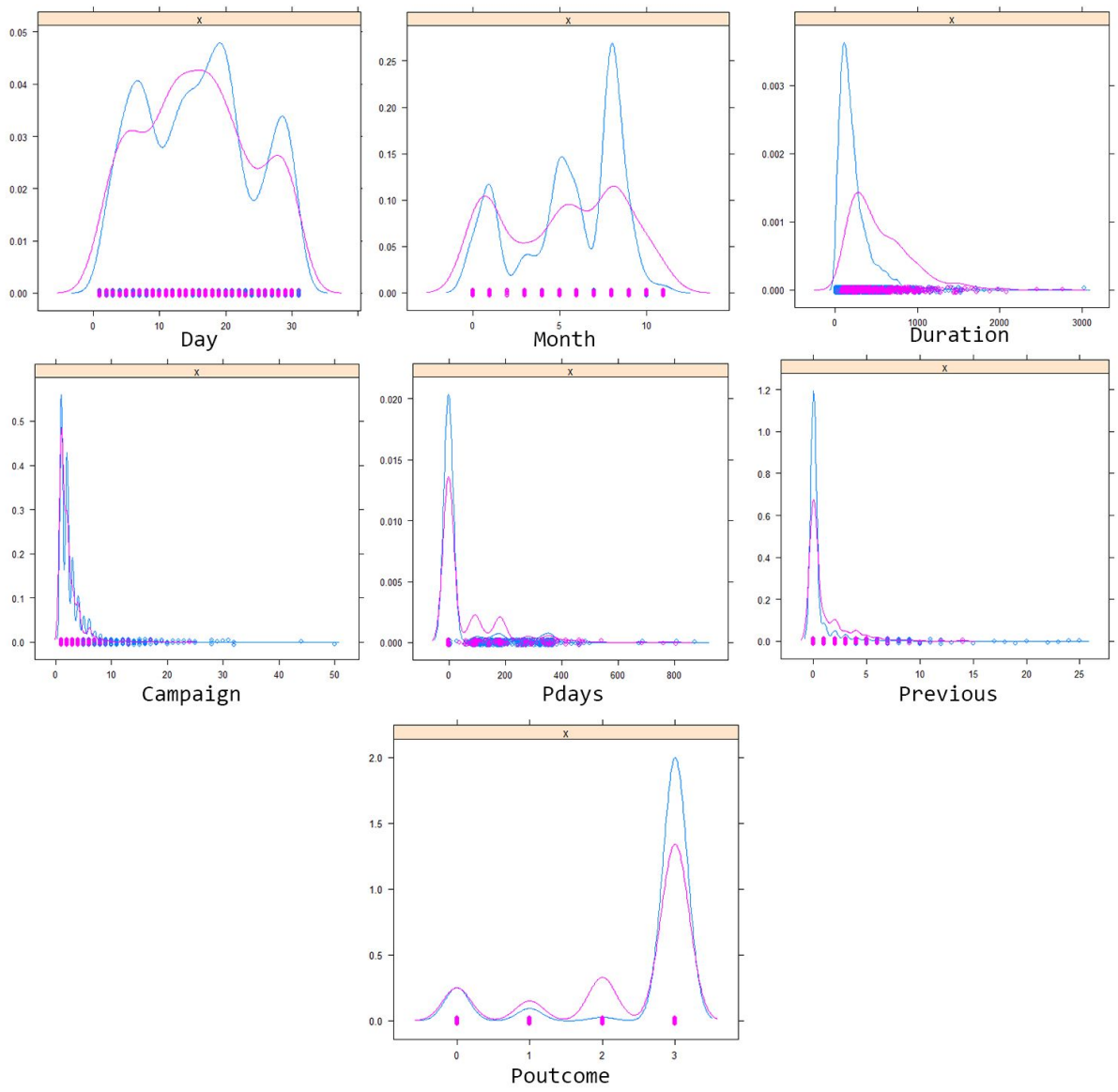
Wykonano też oddzielne wykresy gęstości dla każdego parametru wejściowego. Atrybuty opisane wartościami typu *string* zostały zamienione na kolejne liczby całkowite (widoczne na osiach X).

Wizualizacja z wykorzystaniem funkcji featurePlot - przykład kodu dla jednego wykresu

```
caret::featurePlot(data%job,data[,17],plot="pairs")
```


x
Age
Job
Marital
Education
Default
Balance
Housing
Loan
Contact

Rys. 3



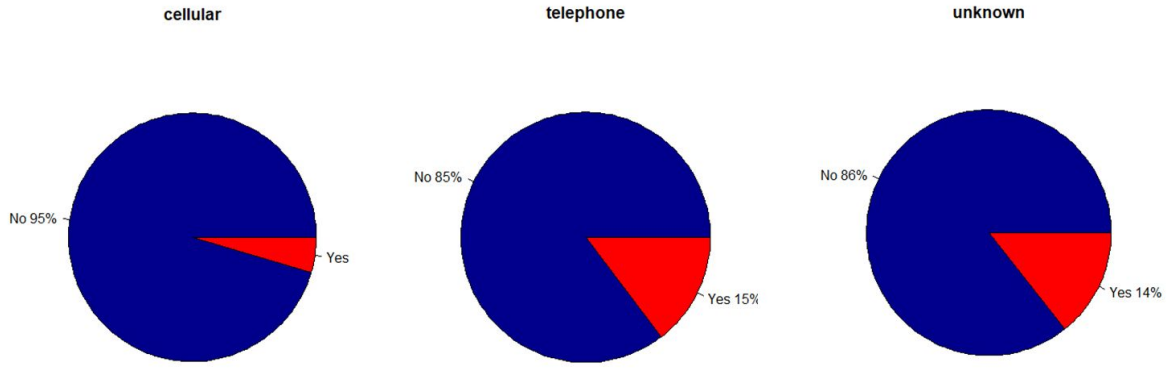
Rys. 4

Najkorzystniejszą sytuacją z punktu widzenia predykcji wartości klasy wyjściowej jest sytuacja, w której pik wykresu gęstości odpowiadający za dany stan wyjściowy (np. niebieski - yes) byłby znacznie “rozsunięty” w osi X względem piku wykresu odpowiadającego za przeciwny stan wyjściowy (fioletowy - no). Jak widać taka klarowna sytuacja nie występuje przy żadnym atrybucie. Widoczne są za to różnice w wysokości pików pomiędzy stanami *yes* i *no* w niektórych parametrach.

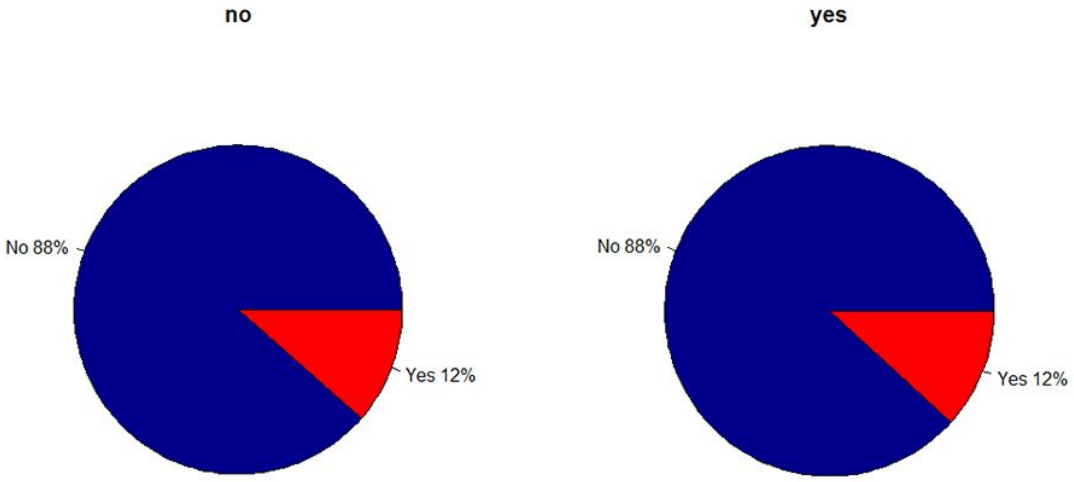
Aby zwizualizować większość atrybutów i ich rozkład w funkcji klasy wyjściowej, które nie są określone przez zakres liczbowy wykorzystano wykresy typu kołowego.

Wykresy kołowe - przykład kodu dla jednego wykresu

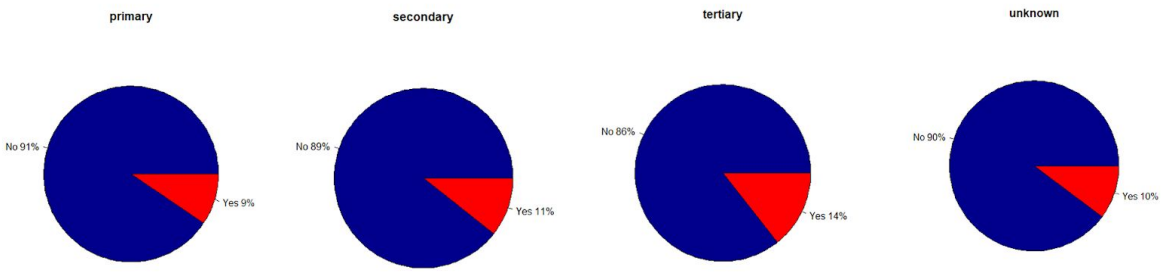
```
counts <- table(data$Diagnosis, data$job)
numbers <- c(counts[7],counts[8])
lbls <- c("No","Yes")
pct <- round(numbers/sum(numbers)*100)
lbls <- paste(lbls, pct)
lbls <- paste(lbls,"%",sep="")
colors = c("darkblue", "red")
pie(numbers, labels = lbls,col=colors,main="admin.")
```



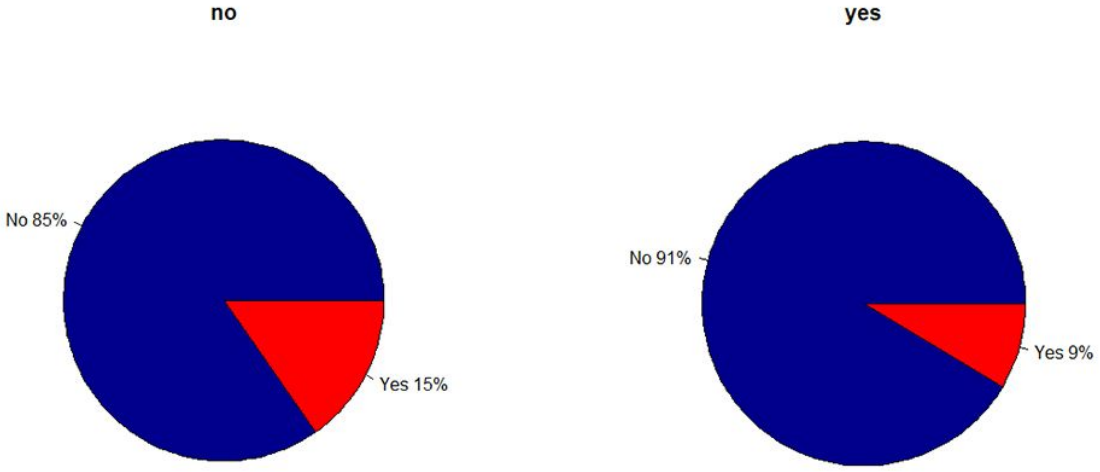
Rys. 5 Wykresy dla atrybutu "contact"



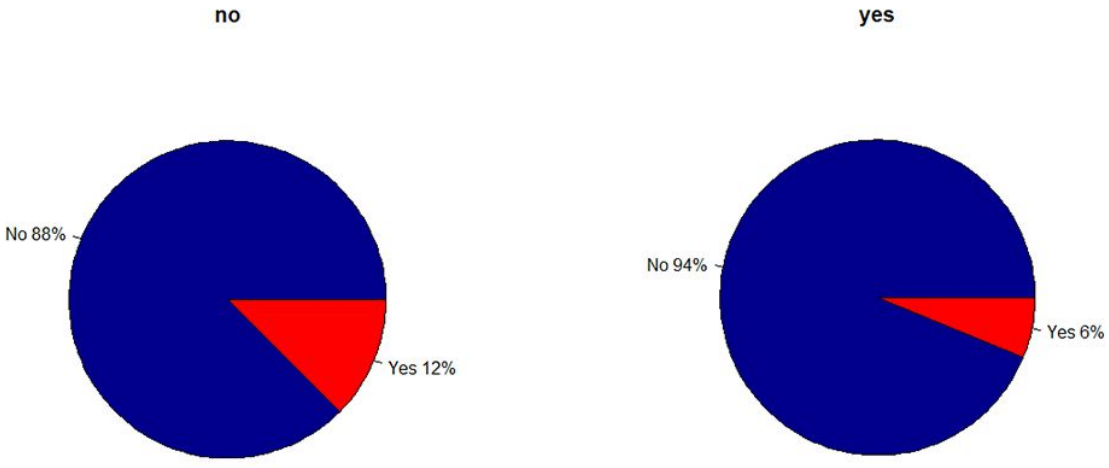
Rys. 6 Wykresy dla atrybutu "default"



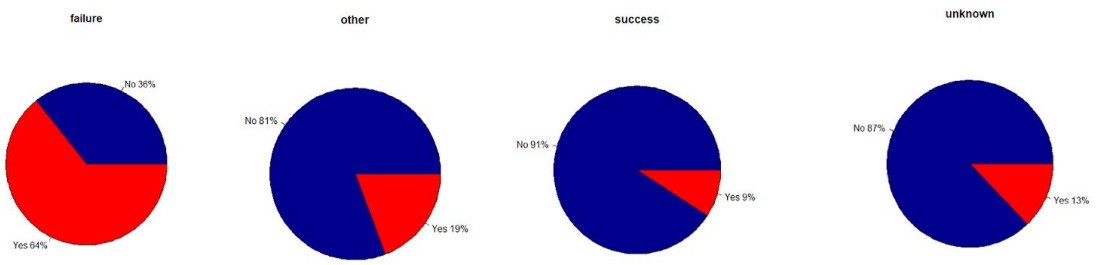
Rys. 7 Wykresy dla atrybutu “education”



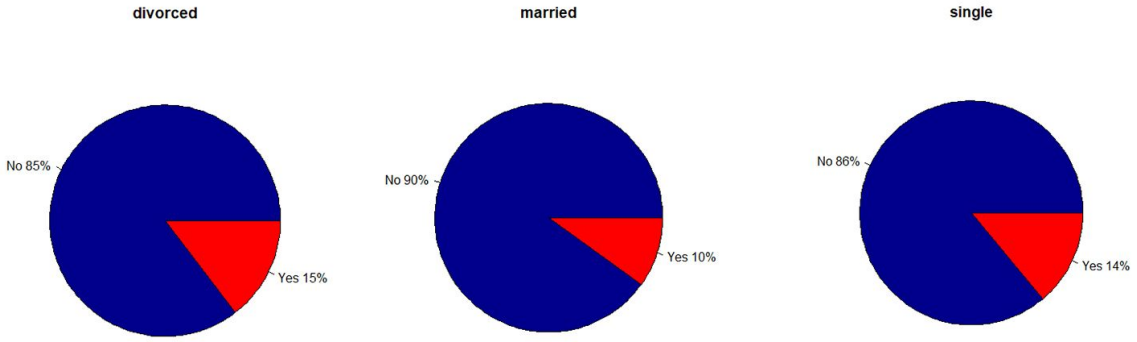
Rys. 8 Wykresy dla atrybutu “housing”



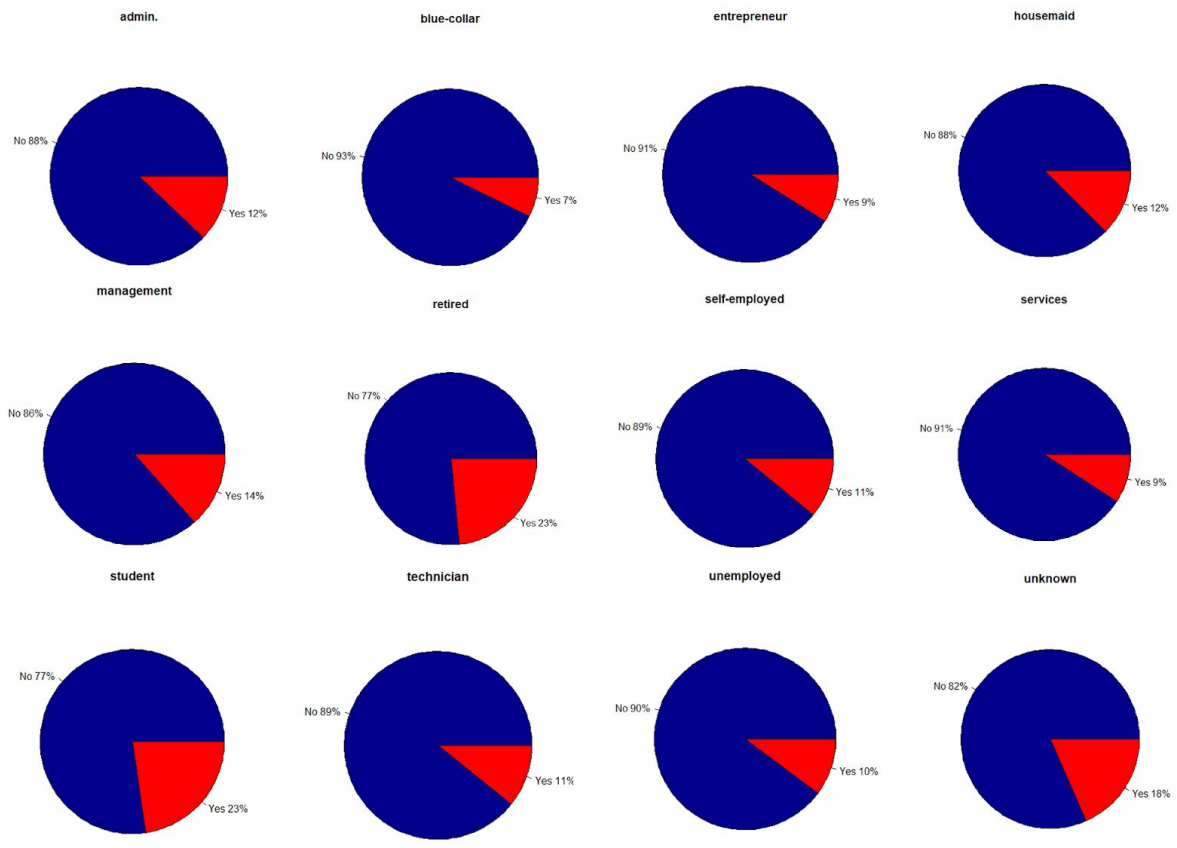
Rys. 9 Wykresy dla atrybutu “loan”



Rys. 10 Wykresy dla atrybutu “poutcome”



Rys. 11 Wykresy dla atrybutu “marital”



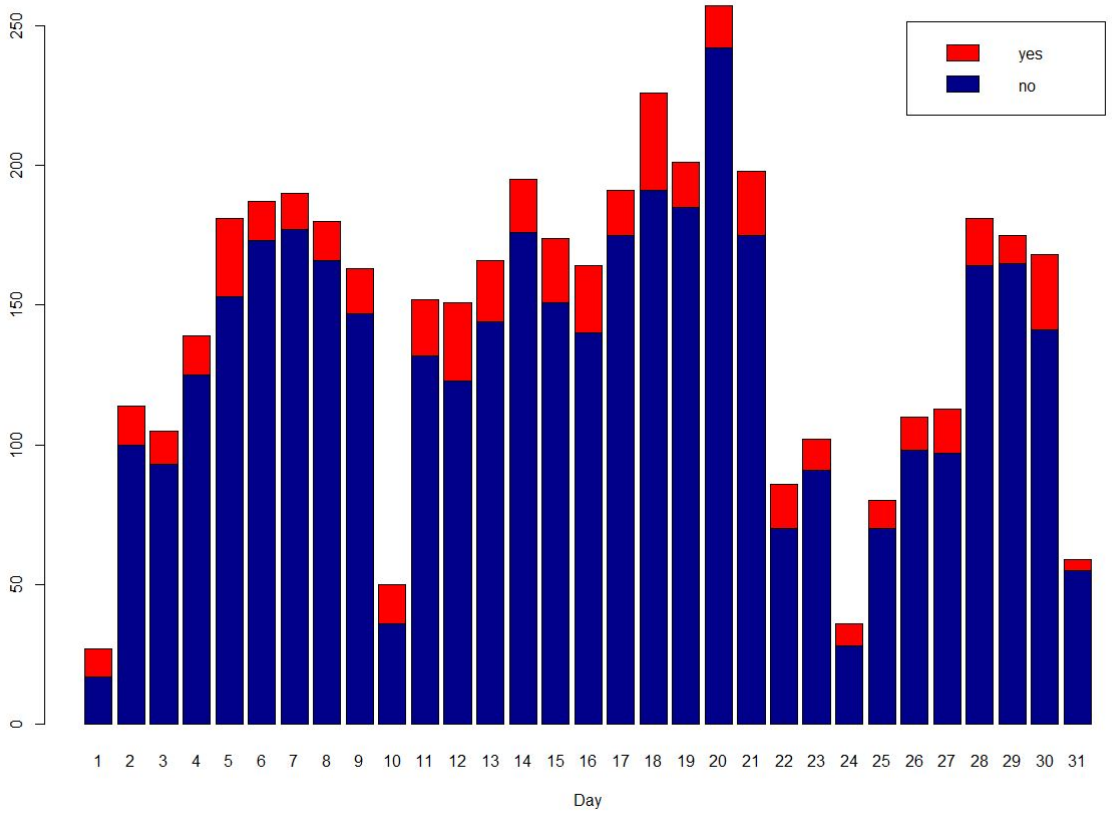
Rys. 12 Wykresy dla atrybutu "job"

Analizując wizualnie powyższe wykresy i biorąc pod uwagę duże niezbalansowanie danych, można zauważyć, że przy większości atrybutów poszczególne ich wartości nie odbiegają od siebie w bardzo znaczący sposób stosunkiem wartości w klasie wyjściowej. Wyjątkiem jest atrybut *poutcome* oraz wartość *failure*, oznacza to, że prawdopodobieństwo decyzji na subskrypcję jest znacząco wyższe przy niepowodzeniu w poprzedniej kampanii.

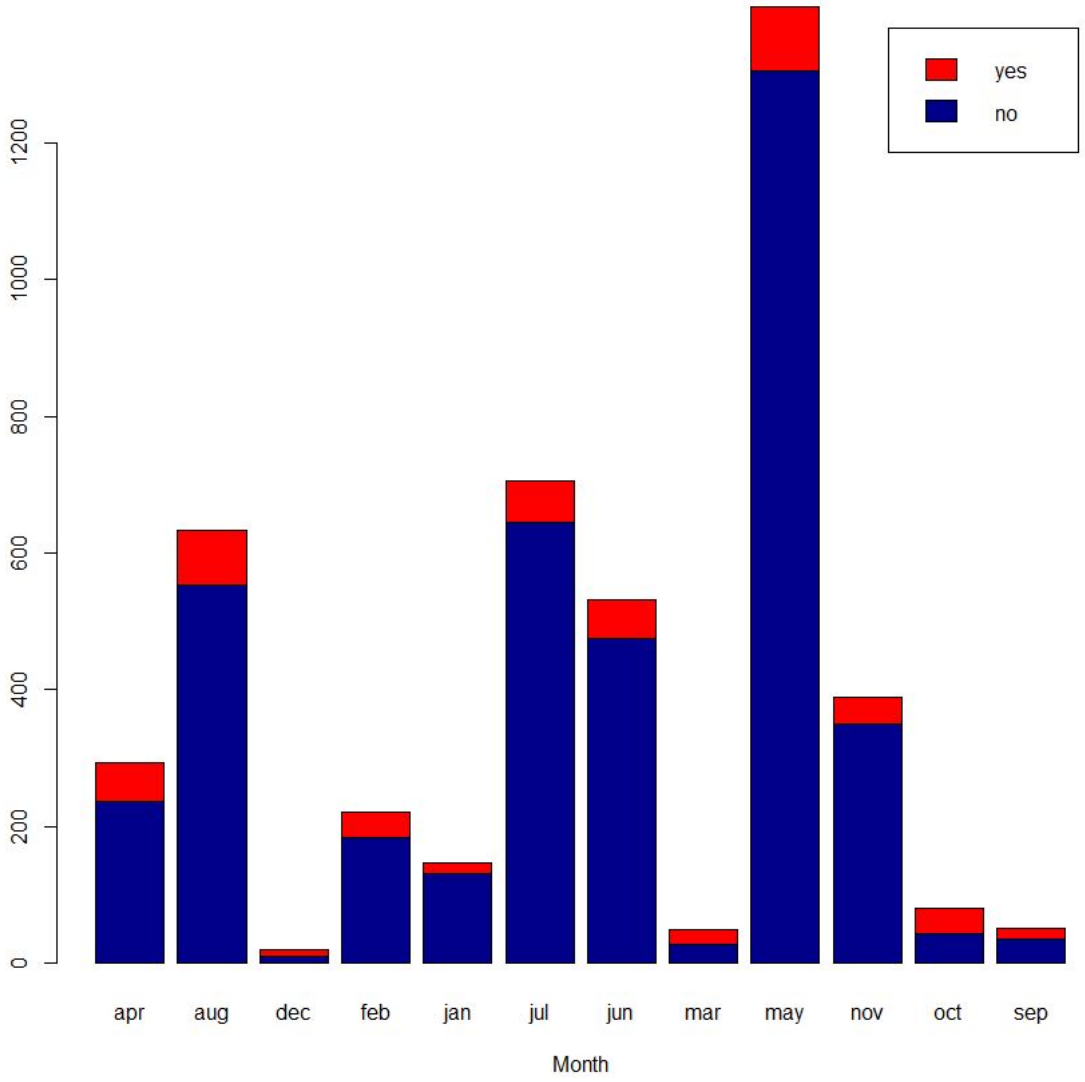
W związku z dużą liczbą wartości atrybutów *month* i *day* zwizualizowano ich zależność od klasy wyjściowej za pomocą tzw. *barplotów*.

Kod dla przykładowego *barplotu*

```
counts <- table(data$Diagnosis, data$month)
barplot(counts, xlab="Month", col=c("darkblue","red"), legend
= rownames(counts))
```



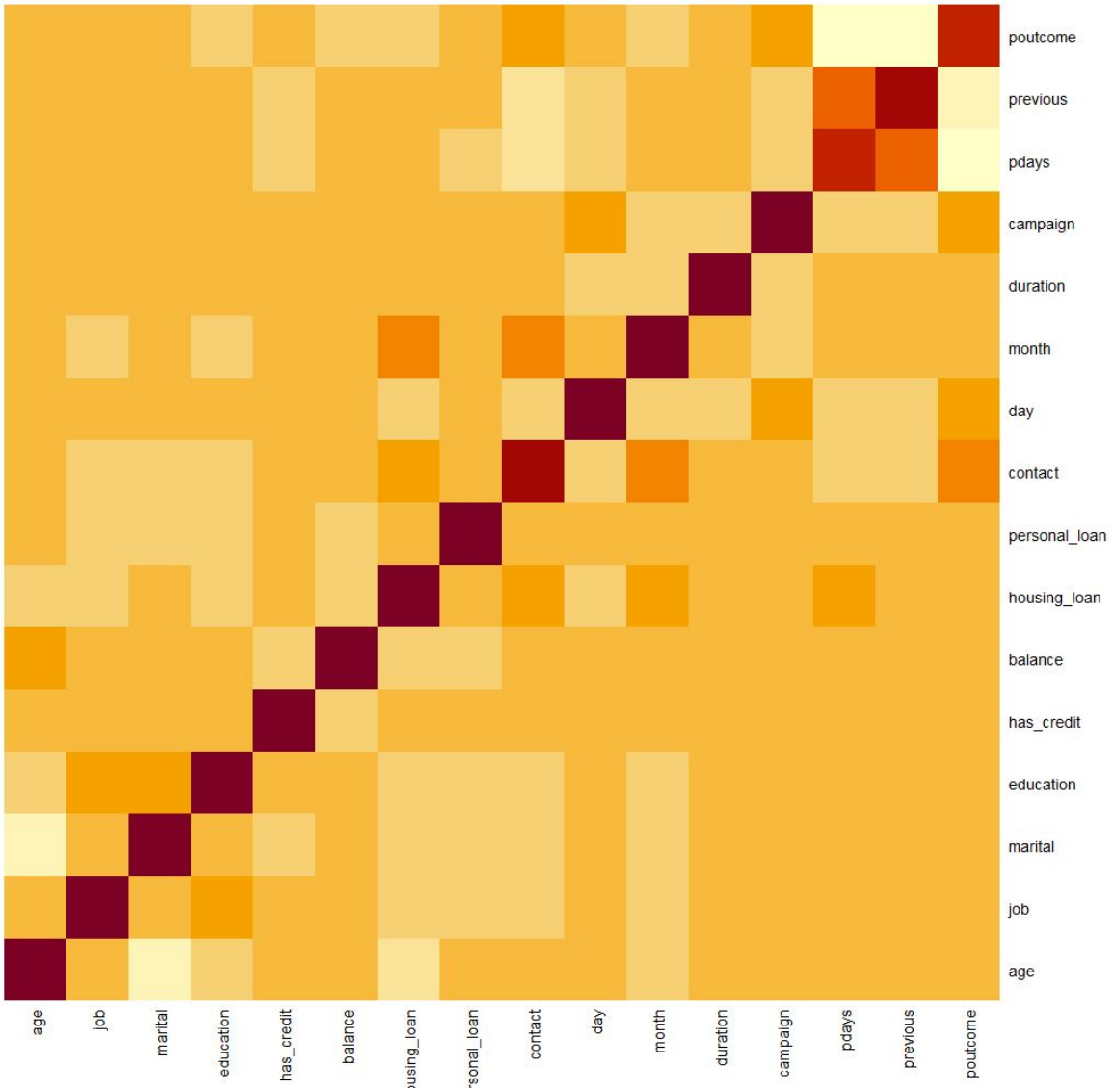
Rys. 13 Wykres dla atrybutu “day”



Rys. 14 Wykres dla atrybutu “month”

W przypadku rozkładu dni w miesiącu nie widać aby jakieś dni (lub okresy) w miesiącu miały znacznie inny rozkład od średniego. Natomiast wykres miesięczny pokazuje duże różnice w liczbie decyzji subskrypcyjnych w poszczególnych miesiącach (jednak nie ma to znaczenia dla analizy i wynika prawdopodobnie z tego, że kontakty z klientami były podejmowane w określonych miesiącach częściej). Rozkład decyzji subskrypcyjnych jest zbliżony, z wyjątkiem np. października, marca czy grudnia, jednak w te miesiące próbka badanych jest nieliczna i nie powinna być traktowana jako stała zależność.

```
Wykres typu heatmap
heatmap(data55[,-17],scale="column", Rowv=NA, Colv=NA)
```



Rys. 15 Wykres dla atrybutu “month”

Wykonano wykres typu *heatmap*, który służy do określenia korelacji między atrybutami (ciemniejszy odcień oznacza większą korelacją, jaśniejszy mniejszą). Parametry skorelowane są dodatnio, nie ma atrybutów, które byłyby zupełnie lub prawie zupełnie nieskorelowane z pozostałymi.

Podczas wizualnej analizy eksploracyjnej zdecydowano się ilustrować pełny zbiór danych, jednak z uwagi na to, że jest on silnie niezbalansowany to utrudniona jest ocena, czy drobne różnice pomiędzy rozkładem wartości klasy wyjściowej są wystarczające do określenia danego atrybutu jako jednego z bardziej znaczących w kontekście automatycznej predykcji wyniku na podstawie danych wejściowych. Kolejnym wnioskiem jest to, że predykcja stanu klasy wyjściowej może być

utrudniona i prawdopodobnie nie uda się osiągnąć pożądanych rezultatów (ponad 90% skuteczności predykcji na zbiorze testowym).

6. Analiza PCA

Aby sprawdzić możliwość zmniejszania rozmiaru zbioru danych statystycznych wykonano analizę PCA. Jako że wartości zmiennych nie są porównywalne wykonaliśmy analizę z normalizacją - tak, aby każda zmienna miała na wejściu identyczną wariancję.

```
pca1 <- prcomp(data[,1:16], center = TRUE, scale. = TRUE)
summary(pca1)
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     1.595 1.3007 1.21246 1.09038 1.06836 1.01725 0.98591
Proportion of Variance 0.159 0.1057 0.09188 0.07431 0.07134 0.06467 0.06075
Cumulative Proportion  0.159 0.2647 0.35662 0.43093 0.50227 0.56694 0.62769

    PC8     PC9    PC10    PC11    PC12    PC13    PC14    PC15    PC16
0.96480 0.95700 0.92198 0.91012 0.87101 0.73829 0.73169 0.67653 0.36754
0.05818 0.05724 0.05313 0.05177 0.04742 0.03407 0.03346 0.02861 0.00844
0.68587 0.74311 0.79624 0.84801 0.89542 0.92949 0.96295 0.99156 1.00000
```

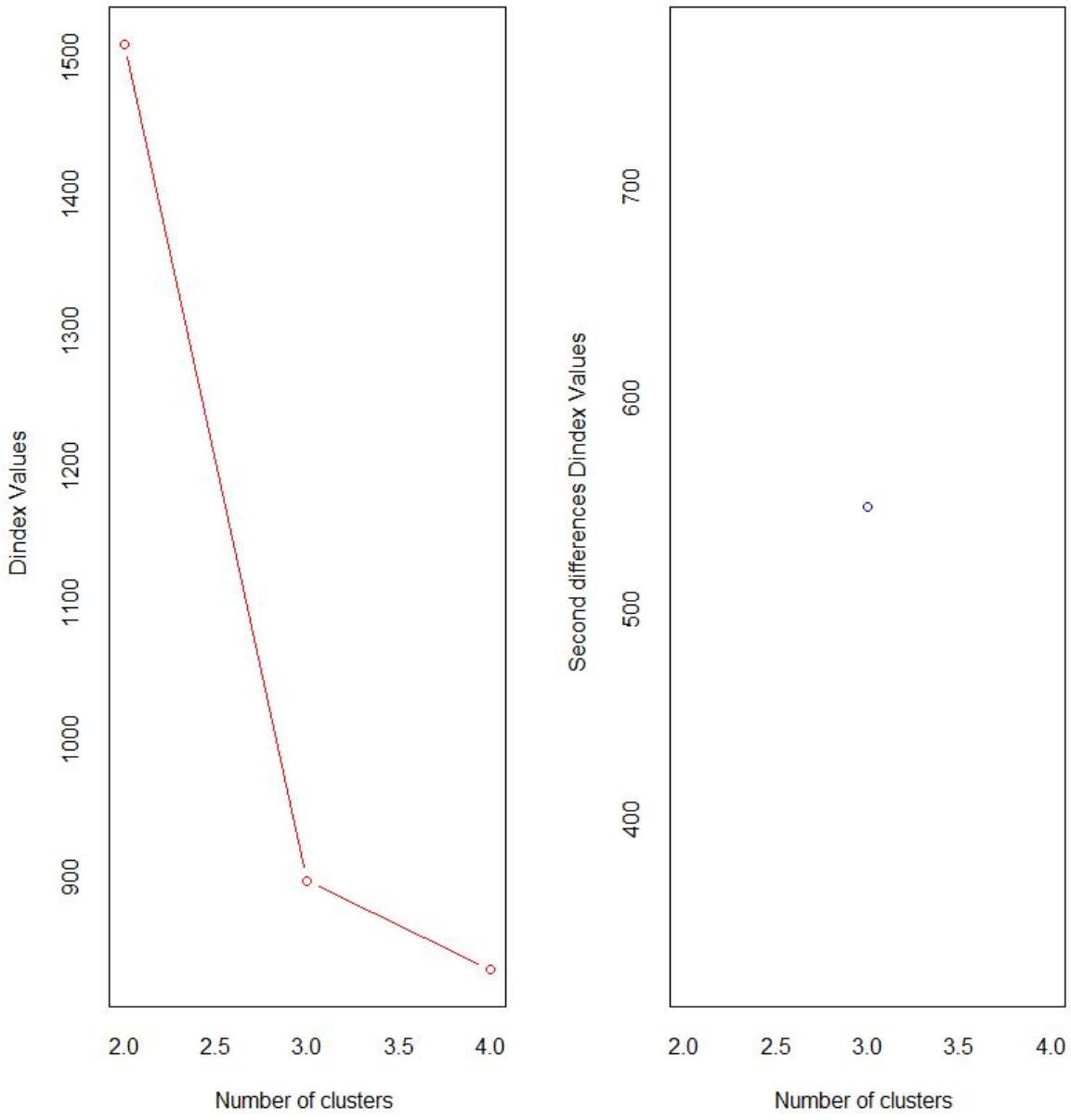
Wyniki analizy PCA są zgodne z oczekiwaniami, czyli nie ma atrybutu, który wpływałby na stan wyjściowy w zdecydowanie dominujący sposób. Do poprawnego opisu 95% obserwacji potrzeba aż 14 z 16 atrybutów. Natomiast najbardziej *znaczący* parametr odpowiada jedynie za 16%.

7. Grupowanie

Wykorzystano funkcję NbClust, aby sprawdzić na ile grup algorytm proponuje podzielić dane, bazując jedynie na atrybutach wejściowych.

```
NbClust::NbClust(data, distance="euclidean", min.nc=2,
max.nc=4, method="complete", index="all")
```

```
*****************************************************************
$All.index
       KL        CH  Hartigan      CCC    Scott      Marriot       TrCovW     TraceW Friedman  Rubin Cindex
2  0.8262  629.3253 1511.1683 -18.2045 5196.056 6.062597e+67 1.242536e+18 4600684723  76.6520 2.1398 0.1156
3  5.4300 1525.9975  388.0559 -21.1059 6195.498 5.227386e+67 1.887045e+17 1875498437  80.2190 5.2490 0.0780
4 12.9074 1525.1773   56.9281 -52.3673 6570.861 6.482086e+67 9.442673e+16 1365498632  82.6501 7.2094 0.0930
      DB Silhouette   Duda  Pseudot2   Beale Ratkowsky       Ball Ptbiserial    Frey McClain   Dunn Hubert
2 0.2855     0.8866 0.3956 1578.4409 16.8664    0.0494 2300342361     0.6875 14.8882  0.0014 0.1516      0
3 0.6555     0.6495 0.6954  109.0680  4.8203    0.0627  625166146     0.5525 -0.9066  0.1348 0.0065      0
4 0.6422     0.6444 0.2285   16.8854 31.0938    0.0578  341374658     0.5654 -0.6992  0.1302 0.0080      0
  SDindex   Dindex   SDbw
2  0.0010 1510.356 1.2558
3  0.0013  897.023 1.0481
4  0.0011  831.964 0.8509

$All.CriticalValues
  CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
2         0.9259            82.6150            0
3         0.8906            30.5767            0
4         0.5436             4.1985            0

$Best.nc
                     KL       CH Hartigan      CCC    Scott      Marriot       TrCovW     TraceW Friedman   Rubin
Number_clusters  4.0000    3.000    3.000   2.0000   3.0000 3.000000e+00 3.000000e+00          3    3.000  3.0000
Value_Index     12.9074 1525.997 1123.112 -18.2045 999.4418 2.089912e+67 1.053831e+18 2215186481    3.567 -1.1488
                 Cindex     DB Silhouette Duda PseudoT2 Beale Ratkowsky       Ball PtBiserial    Frey McClain
Number_clusters  3.000 2.0000     2.0000    NA       NA    NA     3.0000          3     2.0000  2.0000  2.0000
Value_Index      0.078 0.2855     0.8866    NA       NA    NA     0.0627 1675176216     0.6875 14.8882  0.0014
                  Dunn Hubert SDindex Dindex   SDbw
Number_clusters 2.0000      0   2.000      0 4.0000
Value_Index     0.1516      0   0.001      0 0.8509


*****************************************************************************
* Among all indices:
* 8 proposed 2 as the best number of clusters
* 11 proposed 3 as the best number of clusters
* 2 proposed 4 as the best number of clusters

                    ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3


*****************************************************************************
```



Rys. 15

Większość kryteriów sugeruje podział na 3 grupy, co nie jest zgodne z rzeczywistością, może to powodować duże trudności w późniejszej predykcji na zbiorze testowym, gdyż wybrane dane wejściowe nie posiadają wartości, które jednoznacznie sugerowałyby dopasowanie wyjścia do jednego z 2 stanów.

8. Niezbalansowanie danych i podział

Ze względu na wystarczająco dużą ilość danych wejściowych zdecydowano się na zbalansowanie danych za pomocą funkcji *DownSample()*, która pomniejsza

zbiór danych, dla grupy z przeważającą liczbą stanów wyjść, tworząc dwie równoliczne grupy - o ilości danych grupy mniej licznej.

```
dataB <- caret::downSample(x = data[,-17], y = data$Diagnosis)
dataB1 <- dataB
colnames(dataB) <-
c("age","job","marital",
"education","has_credit","balance",
"housing_loan","personal_loan","contact","day","month","durati
on","campaign","pdays","previous","poutcome", "Diagnosis")
```

Podzielono dane na testowe i treningowe w stosunku 70% do 30%.

```
trainIndex<-caret::createDataPartition (dataB$Diagnosis,p =
0.7, list=FALSE)
train1 <- dataB[trainIndex,]
test1 <- dataB[-trainIndex,]
```

9. Zastosowanie różnych metod modelowania

Następnie wykonano 4 przykładowe modele, powstałe na podstawie różnych metod: *eXtreme Gradient Boosting* z drzewem jako atomowym klasyfikatorem (nowsza i szybsza implementacja algorytmu *gradient descent boosting*), *Lasy losowe* (kombinacja wielu drzew decyzyjnych z wykorzystaniem *baggingu*), *SVM z jądrem radialnym, SVM z jądrem polimianowym*.

```
models1 <- lapply(c( 'xgbTree', 'rf', 'svmRadial', 'svmPoly'),
function (met) { caret::train(Diagnosis~., method=met,
data=train1)})
```

10. Ocena dokładności

Sprawdzenie dokładności klasyfikacji z wykorzystaniem wyznaczonych modeli składało się z dwóch etapów: przewidywania wartości dla zbioru testowego, z wykorzystaniem funkcji *predict()* oraz wyznaczenia podstawowych statystyk w porównaniu z wartościami referencyjnymi, przy użyciu funkcji *confusionMatrix()* z pakietu *caret*.

```
pred1a <- predict(models1[[1]], test1)
pred1b <- predict(models1[[2]], test1)
pred1c <- predict(models1[[3]], test1)
pred1d <- predict(models1[[4]], test1)

caret::confusionMatrix(pred1a, test1$Diagnosis)
caret::confusionMatrix(pred1b, test1$Diagnosis)
caret::confusionMatrix(pred1c, test1$Diagnosis)
caret::confusionMatrix(pred1d, test1$Diagnosis)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  119  25
       yes  37 131

               Accuracy : 0.8013
                 95% CI : (0.7526, 0.8441)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6026

 Mcnemar's Test P-Value : 0.1624

            Sensitivity : 0.7628
            Specificity : 0.8397
         Pos Pred Value : 0.8264
         Neg Pred Value : 0.7798
             Prevalence : 0.5000
         Detection Rate : 0.3814
   Detection Prevalence : 0.4615
      Balanced Accuracy : 0.8013

       'Positive' Class : no
```

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  123  22
       yes  33 134

               Accuracy : 0.8237
                 95% CI : (0.7768, 0.8644)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6474

 Mcnemar's Test P-Value : 0.1775

            Sensitivity : 0.7885
            Specificity : 0.8590
         Pos Pred Value : 0.8483
         Neg Pred Value : 0.8024
             Prevalence : 0.5000
         Detection Rate : 0.3942
   Detection Prevalence : 0.4647
      Balanced Accuracy : 0.8237

       'Positive' Class : no
```

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  112  36
       yes  44 120

               Accuracy : 0.7436
                 95% CI : (0.6914, 0.7911)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.4872

 Mcnemar's Test P-Value : 0.4338

            Sensitivity : 0.7179
            Specificity : 0.7692
         Pos Pred Value : 0.7568
         Neg Pred Value : 0.7317
             Prevalence : 0.5000
         Detection Rate : 0.3590
   Detection Prevalence : 0.4744
      Balanced Accuracy : 0.7436

       'Positive' Class : no
```

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  112  34
       yes  44 122

               Accuracy : 0.75
                 95% CI : (0.6981, 0.7971)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5

 Mcnemar's Test P-Value : 0.3082

            Sensitivity : 0.7179
            Specificity : 0.7821
         Pos Pred Value : 0.7671
         Neg Pred Value : 0.7349
             Prevalence : 0.5000
         Detection Rate : 0.3590
   Detection Prevalence : 0.4679
      Balanced Accuracy : 0.7500

       'Positive' Class : no
```

Najlepszy model z wybranych okazał się model oparty na metodzie *lasów losowych* z 82% dokładnością predykcji. 55 przypadków zostało źle sklasyfikowanych dla najlepszego obliczonego modelu. Rozbieżność między specyficznością i czułością nie jest zbyt duża.

Po analizie tych wyników zastosowano walidację krzyżową, która poprawia sztuczny podział na dane testowe i treningowe. W tym celu skorzystano z metody *LOOCV*, czyli *Leave one out* i wykonano powtórnie model z wykorzystaniem metody ‘rf’.

```
train_cont <-  caret::trainControl(method = "LOOCV")
models2 <- lapply(c('rf'), function (met) {
caret::train(Diagnosis~., method=met, data=dataB,trControl =
train_cont)})
```

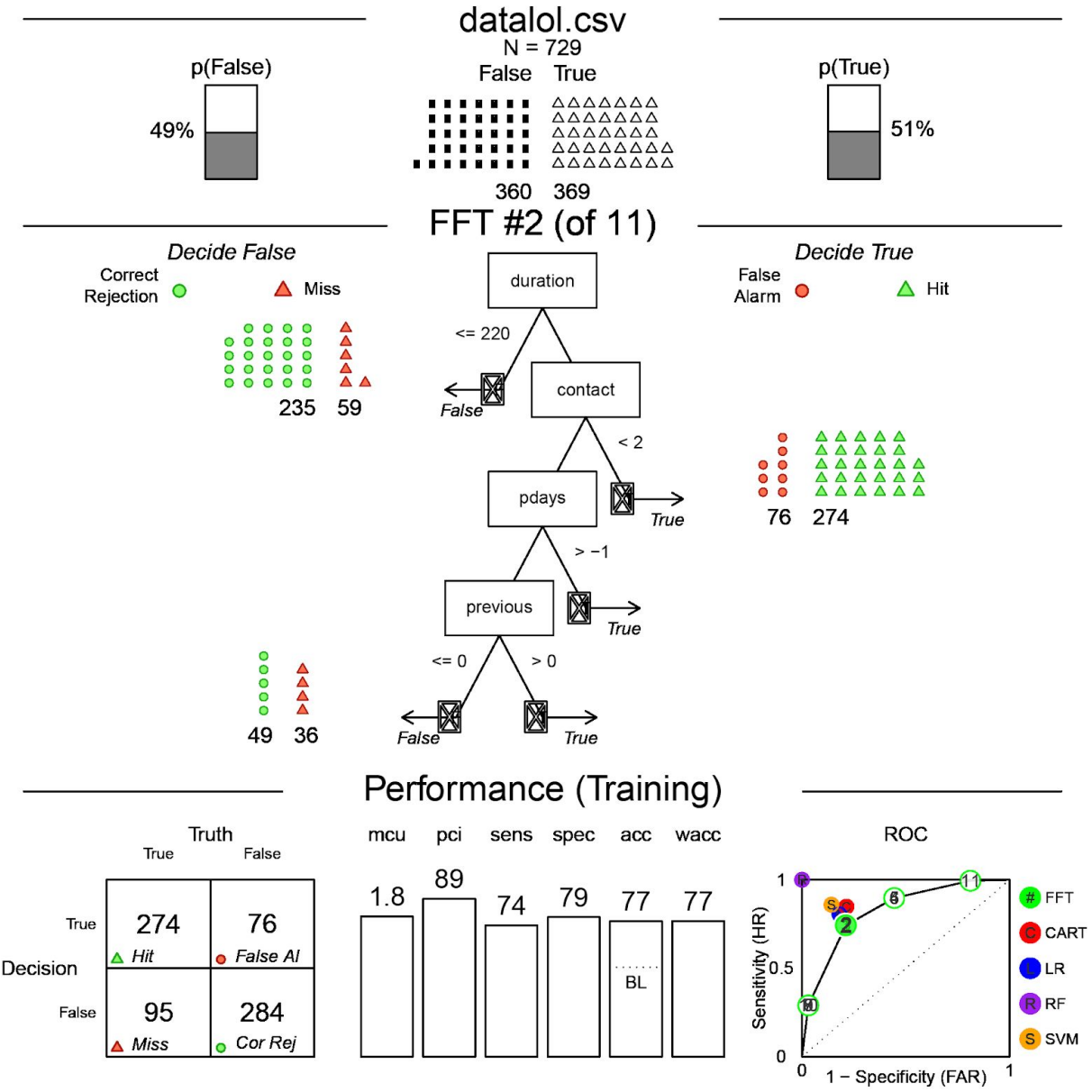
Wynik przedstawiono poniżej:

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  1217  211
       yes  346 1352

               Accuracy : 0.8218
                 95% CI : (0.8079, 0.8351)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6436

 Mcnemar's Test P-Value : 1.365e-08

            Sensitivity : 0.7786
            Specificity : 0.8650
         Pos Pred Value : 0.8522
         Neg Pred Value : 0.7962
             Prevalence : 0.5000
         Detection Rate : 0.3893
   Detection Prevalence : 0.4568
      Balanced Accuracy : 0.8218

       'Positive' Class : no
```

Zastosowanie walidacji krzyżowej nie wpłynęło znacząco na uzyskane wartości dokładności. Zwiększyła się nieco różnica między specyficznością i czułością oraz zwiększyły parametry ufności 95% CI.

11. Fast and Frugal Trees

Do analizy danych wykorzystano również metodę *Fast and frugal trees*, której efekt w postaci wygenerowanego w aplikacji *Shiny* drzewa wraz z jego parametrami przedstawiono poniżej.



Rys. 16

Jak widać metoda ta nie przyniosła lepszych rezultatów, wartość *accuracy* na poziomie 77% jest zbliżona (a nawet niższa) do przedstawionych wyżej modeli.

12. Automatyczny tuning jednej z metod

Wykonano także automatyczny tuning dla metody *random forest*, by sprawdzić czy metoda znajdzie wartości parametrów, które dadzą lepsze rezultaty. Zastosowano losowe przeszukiwanie parametrów.

```
control <- caret::trainControl(method='repeatedcv', number=10,
repeats=3, search='random')
set.seed(1)
rf_random <- caret::train(Diagnosis ~ ., data = train1, method
= 'rf', metric = 'Accuracy', tuneLength  = 15, trControl =
control)
print(rf_random)
```

Otrzymany rezultat przedstawiony jest poniżej:

```
Random Forest

730 samples
 16 predictor
  2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 657, 658, 657, 657, 656, 658, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   1    0.7578075  0.5156320
   2    0.8167603  0.6334661
   3    0.8204516  0.6408579
   4    0.8186123  0.6371963
   5    0.8195444  0.6390672
   6    0.8145025  0.6289780
   7    0.8145654  0.6291463
   8    0.8127137  0.6254322
   9    0.8104551  0.6208819
  10    0.8095232  0.6190338
  11    0.8049442  0.6098711
  12    0.8063579  0.6126838
  13    0.8072837  0.6145460
  14    0.8035740  0.6071114
  15    0.8040559  0.6081164

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 3.
```

Jak widać powyżej najwyższa wartość *accuracy* to 0,820 dla parametrów *mtry* = 3 i *kappa* = 0,641 i nie jest ona wyższa niż uzyskana wcześniej podczas modelowania.

13. Podsumowanie

Najdokładniejsze wyniki otrzymano stosując model przy pomocy metody *Random Forest*. Dokładność jej przewidywania nie jest jednak bardzo wysoka, co

wynika prawdopodobnie z charakterystyki badanych danych. Już w trakcie automatycznego podziału na grupy (*NbClust*) oraz analizy wykresów można było się spodziewać, że wyniki predykcji nie osiągną poziomu dokładności powyżej 90%.