Modelovanje ponašanja klijenata u banci

eng. Churn Modelling

Aleksandra Zdravković Ognjen Lazić Kosta Ljujić Mihajlo Srbakoski

Uvod

Banke i osiguravajuće kompanije često koriste analizu odliva kupaca (*eng. churn analysis*) i stope odliva klijenata kao jednu od svojih ključnih poslovnih pokazatelja, jer su troškovi zadržavanja postojećih kupaca daleko manji od sticanja novog.

Ova analiza se fokusira na ponašanje bankarskih klijenata za koje je veća verovatnoća da će napustiti banku (tj. zatvoriti svoj bankovni račun). Cilj je otkrivanje najupečatljivijih ponašanja kupaca kroz istraživačku analizu podataka, kao i upotreba tehnika prediktivne analize kako bi se utvrdili kupci koji će najverovatnije napustiti banku.

Pretprocesiranje podataka (*eng. Data Preprocessing*)

```
data <- read_csv("data_with_NA.csv")
data <- as.data.frame(data)
head(data)
```

```
##   RowNumber CustomerId  Surname CreditScore Geography Gender Age Tenure
## 1         1   15634602 Hargrave         619    France Female  42      2
## 2         2   15647311     Hill         608     Spain Female  41      1
## 3         3   15619304     Onio         502    France Female  42      8
## 4         4   15701354     Boni          NA    France Female  39      1
## 5         5   15737888 Mitchell         850     Spain Female  43      2
## 6         6   15574012      Chu         645     Spain   Male  44      8
##     Balance NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited
## 1      0.00             1         1             NA       101348.88      1
## 2  83807.86             1         0              1             NA      0
## 3 159660.80             3        NA              0       113931.57      1
## 4      0.00             2         0              0        93826.63      0
## 5 125510.82            NA         1              1        79084.10      0
## 6 113755.78             2         1              0       149756.71      1
```

```
glimpse(data)
```

```
## Rows: 10,000
## Columns: 14
## $ RowNumber       <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...
## $ CustomerId      <dbl> 15634602, 15647311, 15619304, 15701354, 15737888, 1...
```

```
## $ Surname         <chr> "Hargrave", "Hill", "Onio", "Boni", "Mitchell", "Ch...
## $ CreditScore     <dbl> 619, 608, 502, NA, 850, 645, 822, 376, 501, 684, 52...
## $ Geography       <chr> "France", "Spain", "France", "France", "Spain", "Sp...
## $ Gender          <chr> "Female", "Female", "Female", "Female", "Female", "...
## $ Age             <dbl> 42, 41, 42, 39, 43, 44, 50, 29, 44, 27, 31, 24, 34,...
## $ Tenure          <dbl> 2, 1, 8, 1, 2, 8, 7, 4, 4, 2, 6, 3, 10, 5, 7, 3, 1,...
## $ Balance         <dbl> 0.00, 83807.86, 159660.80, 0.00, 125510.82, 113755....
## $ NumOfProducts   <dbl> 1, 1, 3, 2, NA, 2, 2, NA, 2, 1, 2, 2, NA, 2, 2, 2, ...
## $ HasCrCard       <dbl> 1, 0, NA, 0, 1, 1, 1, 1, 0, NA, 0, 1, 1, 0, 1, 0, N...
## $ IsActiveMember  <dbl> NA, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, NA...
## $ EstimatedSalary <dbl> 101348.88, NA, 113931.57, 93826.63, 79084.10, 14975...
## $ Exited          <dbl> 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...
```

RowNumber	Redni broj reda (od 1 do 10 000)
CustomerId	Jedinstveni identifikacioni broj klijenta banke
Surname	Prezime klijenta
CreditScore	Kreditni skor klijenta
Geography	Zemlja porekla klijenta
Gender	Pol klijenta (muško ili žensko)
Age	Godine klijenta
Tenure	Broj godina koliko je dugo klijent u banci
Balance	Stanje na racunu
NumOfProducts	Broj proizvoda banke koje klijent koristi
HasCrCard	Indikator da li klijent poseduje kreditnu karticu banke
IsActiveMember	Indikator da li je klijent aktivan u banci
EstimatedSalary	Procenjena plata klijenta (u dolarima)
Exited	Indikator da li je klijent napustio banku

NA vrednosti

Proverava se da li postoje NA vrednosti:

```
sapply(data, function(x) mean(is.na(x)))
```

```
##       RowNumber      CustomerId         Surname     CreditScore       Geography
##          0.0000          0.0000          0.0000          0.1045          0.0000
##          Gender             Age          Tenure         Balance   NumOfProducts
##          0.0000          0.0000          0.0000          0.1009          0.1052
##       HasCrCard  IsActiveMember EstimatedSalary          Exited
##          0.0958          0.0968          0.0968          0.0000
```

Dakle, u sledećim kolonama se pojavljuju NA vrednosti:

- *CreditScore*
- *Balance*
- *NumOfProducts*
- *HasCrCard*
- *IsActiveMember*
- *EstimatedSalary*

Imputacija podataka. Algoritam *miss forest*

Miss forest je algoritam koji uz pomoć algoritma slučajna šuma (*eng. random forest*) imputira nedostajuće podatke.

Inicijalno, nedostajući podaci se dopunjavaju koristeći srednju vrednost/modu obeležja, a zatim se za svaku kolonu sa vrednostima koje nedostaju kreira model algoritmom *random forest* koji predviđa nedostajuću vrednost na osnovu ostalih. Ovaj proces se ponavlja dok se ne dostigne maksimalan broj iteracija.

Iskoristimo ovaj algoritam u imputaciji datih podataka.

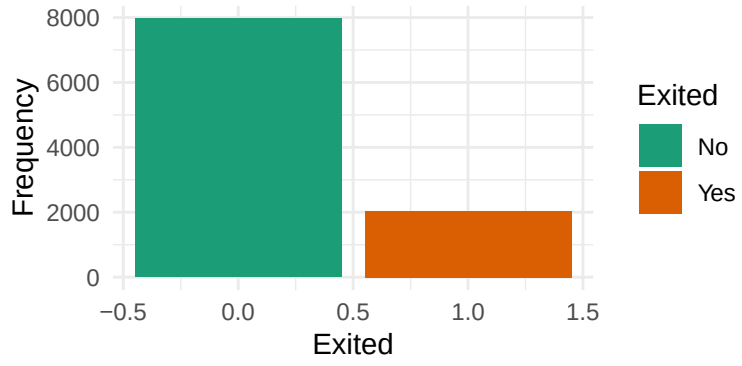
```
# kategoričke prediktore transformišemo u tip factor
# kako se ne bi desilo da se kod njih pojave decimalne vrednosti
# nakon što algoritam dodeli npr. srednje vrednosti na nedostajućim
# mestima
data[, c(10, 11, 12)] <- lapply(data[, c(10, 11, 12)], as.factor)
data.imp <- missForest::missForest(data[, c(4, 9, 10, 11, 12, 13)], maxiter = 6)
```

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
##   missForest iteration 4 in progress...done!
##   missForest iteration 5 in progress...done!
##   missForest iteration 6 in progress...done!
```

```
data[, c(4, 9, 10, 11, 12, 13)] <- data.imp$ximp
data[, c(10, 11, 12)] <- lapply(data[, c(10, 11, 12)], as.numeric) # vracamo u tip numeric
# proveravamo da li je sada procenat NA vrednosti 0%
sapply(data, function(x) mean(is.na(x)))
```

```
##       RowNumber      CustomerId         Surname     CreditScore       Geography
##               0               0               0               0               0
##          Gender             Age          Tenure         Balance   NumOfProducts
##               0               0               0               0               0
##       HasCrCard  IsActiveMember EstimatedSalary          Exited
##               0               0               0               0
```

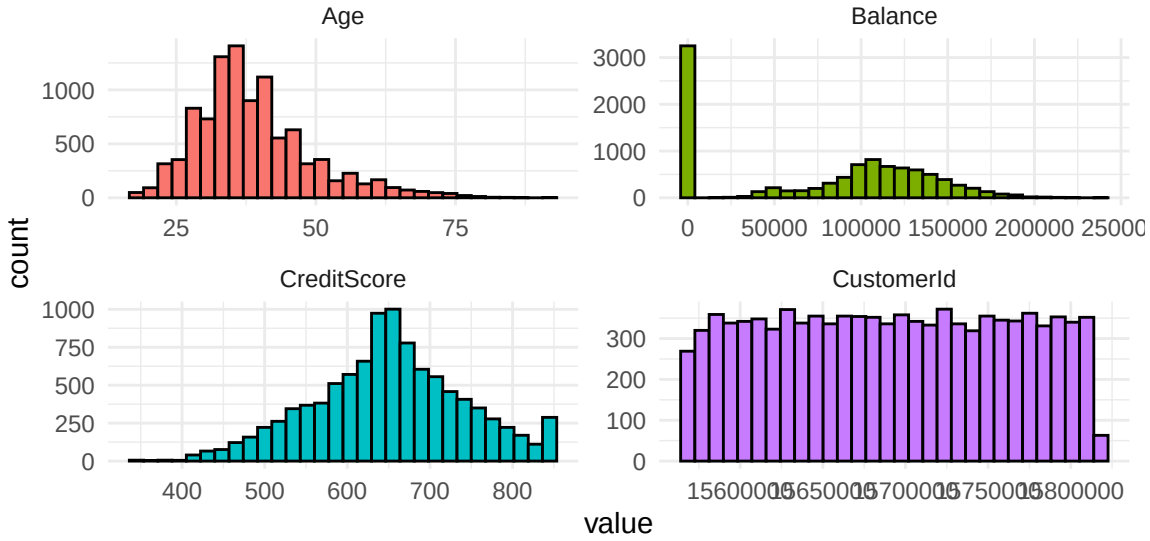
Zavisna promenljiva



Vidi se da većina korisnika nije napustila banku.

Analiza prediktora

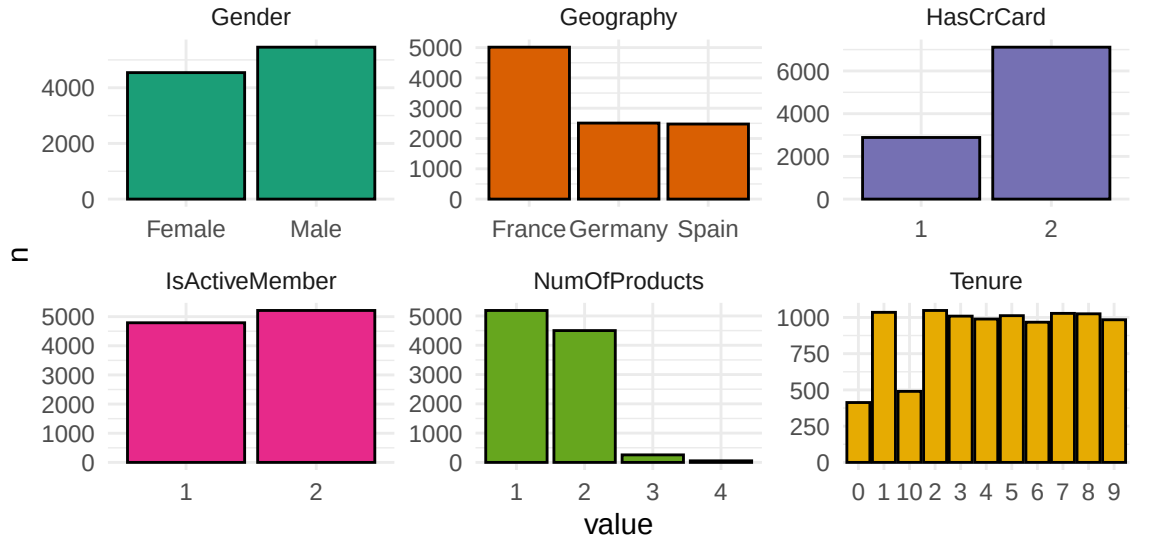
Pogledajmo prvo raspodele neprekidnih prediktora.



Zaključujemo:

- Raspodela prediktora *Age* je pomerena udesno.
- Prediktor *Balance* je blizu normalno raspodeljen.
- Većina predikotra *Credit score* je veća od 600. Moguće je da će baš ovi klijenti napustiti banku.

Pogledajmo sada raspodele kategoričkih prediktora.

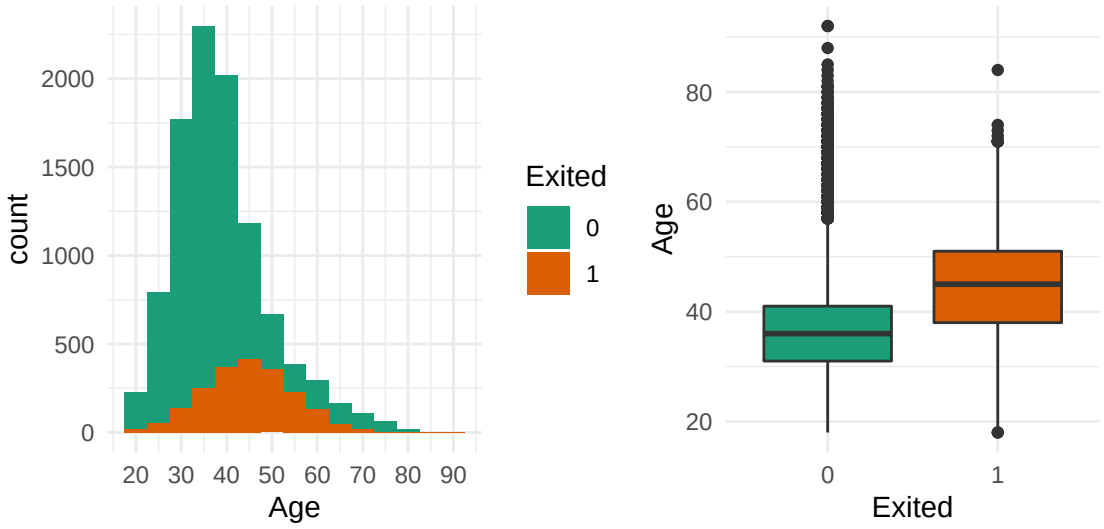


Zaključujemo:

- Veći broj klijenata je muškog pola.
- Klijenti su većinski iz Francuske.
- Većina klijenata ima kreditnu karticu.

- Broj aktivnih i neaktivnih članova je veoma sličan.
- Većina klijenata koristi 1 do 2 proizvoda banke, dok jako malo klijenata koristi 3 i 4 proizvoda.
- Broj klijenata koji su članovi banke $1, 2, ..., 9$ godina je približno isti.

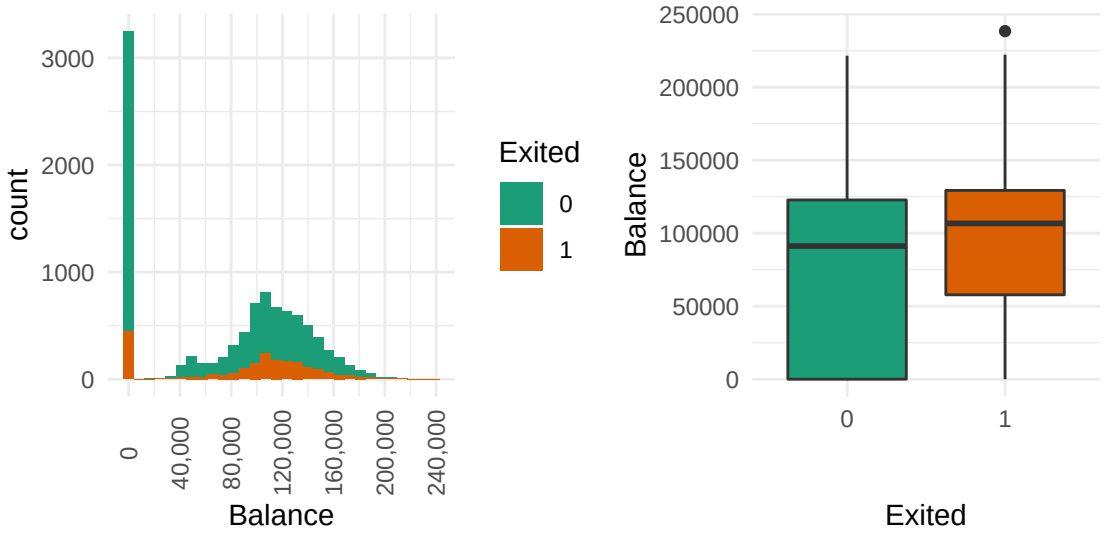
Prediktor *Age*



Zaključujemo:

- Klijenti koji su ostali u banci imaju tendenciju da budu mlađi.
- Veliki broj klijenata koji su napustili banku ima između 40 i 50 godina.
- Klijenti starosti između 60 i 80 godina imaju tendenciju da ne napuštaju banku.

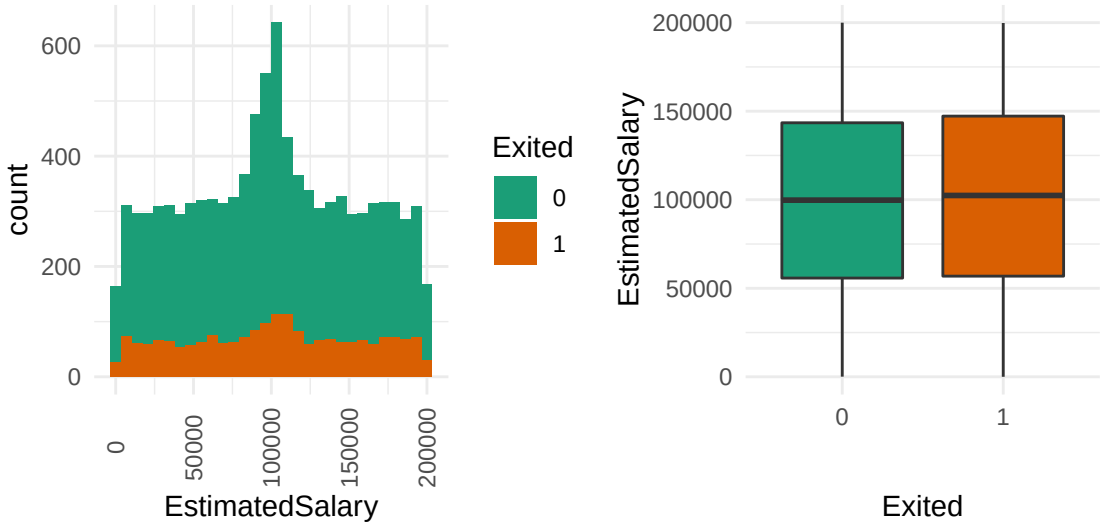
Prediktor *Balance*



Zaključujemo:

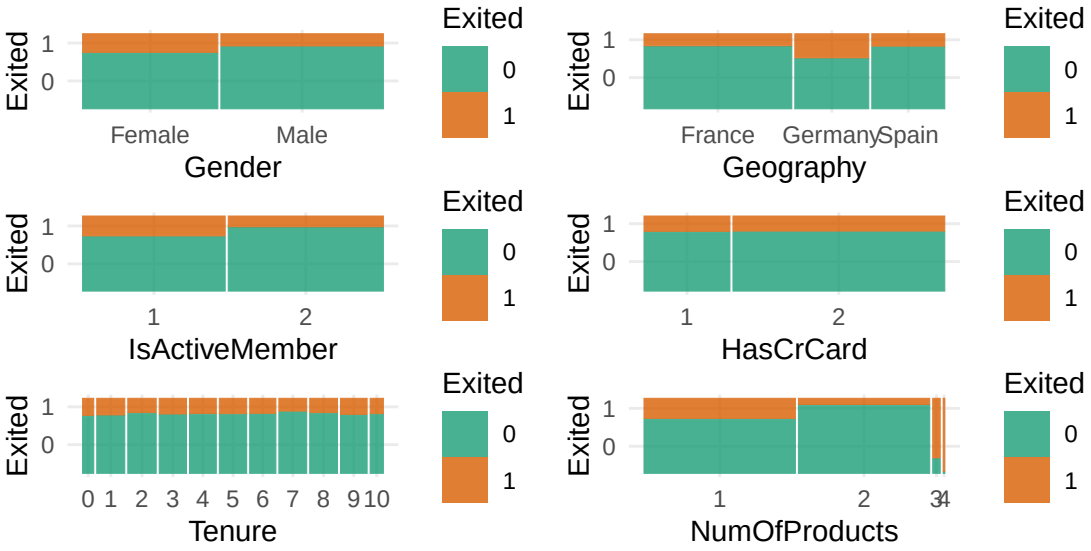
- Klijenti koji ostaju u banci imaju manje sredstava na računu od onih koji napuštaju banku.

Prediktor *Estimated Salary*



Zaključujemo:

- Ne postoji vidna razlika u zaradi između klijenata koji napuštaju/ne napuštaju banku.



Zaključujemo:

- Klijenti koji ostaju u banci koriste manje proizvoda od onih koji napuštaju banku.
- Ostali prediktori nemaju značajan uticaj na napuštanje banke

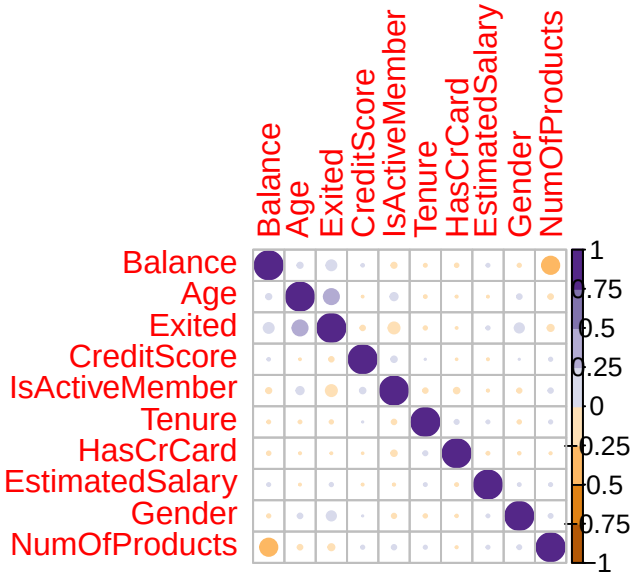
Čišćenje podataka (*eng. Data Cleaning*)

Kako smatramo da *RowNumber, CustomedId, Surname, Geography* nisu značajni prediktori, nećemo ih posmatrati u daljem radu. Kategorički prediktor *Gender* ćemo kodirati binarno.

```
# odbacujemo navedene kolone
data <- data[, -which(names(data) %in% c("RowNumber",
                                         "CustomerId",
                                         "Surname",
                                         "Geography"))]
data$Gender <- ifelse(data$Gender=="Male", 0, 1)
```

Korelacija

```
corrplot(cor(data.matrix(data)), order = "hclust",
         col = brewer.pal(n = 8, name = "PuOr"))
```



Vidimo da između preostalih prediktora ne postoji značajna korelacija.

Kreiranje modela

Delimo podatke na trening i test skup u odnosu 4:1.

```
set.seed(11)
index_train <- sample(nrow(data), 0.8 * nrow(data))
train <- data[index_train, ]
test <- data[-index_train, ]
```

Podaci su nebalansirani, pa ćemo ih balansirati koristeći funkciju *mwmote*. Ova funkcija na slučajan način bira tačku A iz nedominantne kategorije, a potom posmatra njenih k najbližih suseda koji imaju istu kategoriju. Od tih suseda se na slučajan način bira tačka B. Nova tačka C, koju će funkcija generisati i dodeliti

joj nedominantnu kategoriju, nalazi se između A i B, tj. dobijena je kao njihova konveksna kombinacija: $C = cA + (1-c)B$, za neko $c \in [0,1]$.

```
n <- 8000 - 2 * sum(train$Exited)
newSamples <- mwmote(dataset = train, numInstances = n, classAttr = "Exited")
train1 <- rbind(train, newSamples)
train2 <- train1 # pravimo kopiju
```

Kao meru kvaliteta modela koristimo kombinaciju *fbeta_score* i *recall*. Kako nam je najbitnije otkrivanje klijenata koji će napustiti banku, posmatraćemo zbir $\frac{9}{10} fbeta\_score + \frac{1}{10} recall$. Na taj način *recall* dobija veću težinu nego *precision*, što smo i hteli da postignemo.

Neuronska mreža

Prvo pravimo model koristeći potpuno povezanu neuronsku mrežu sa dva skrivena sloja od po 10 i 6 čvorova. Postupak ponavljamo 5 puta i čuvamo najbolji rezultat.

```
nc <- length(train1)
trainLabels <- to_categorical(train1$Exited)
testLabels <- to_categorical(test$Exited)
training1 <- as.matrix(train1[, -which(names(data) %in% c("Exited"))])
test1 <- test[, -which(names(data) %in% c("Exited"))]
test2 <- test1 # pravimo kopiju
```

```
thresholds <- seq(0.1, 0.9, 0.02)
l <- length(thresholds)
# max <- 0
# pred_max <- c()
# for(i in 1:5){
#   model_nn <- keras_model_sequential()
#   model_nn %>%
#         layer_dense(units = 10, activation = 'relu', input_shape = c(nc - 1)) %>%
#         layer_dense(units = 6, activation = 'relu', input_shape = c(10)) %>%
#         layer_dense(units = 2, activation = 'softmax')
#   model_nn %>% compile(loss = 'categorical_crossentropy',
#                        optimizer = 'adam',
#                        metrics = 'accuracy')
#   model_nn %>% fit(data.matrix(training1),
#                    trainLabels,
#                    epochs = 12,
#                    batch_size = 128,
#                    validation_split = 0.2,
#                    verbose = 0)
#   pred <- model_nn %>% predict(as.matrix(test1))
#   pred <- apply(pred, 1, which.max) - 1
#   fbeta <- fbeta_score(test$Exited, pred)
#   recall <- recall(test$Exited, pred)
#   if((9*fbeta+recall) > max){
#     max <- 9*fbeta+recall
#     pred_max <- pred
#   }
# }
```

```
# recall(test$Exited, pred_max)
# fbeta_score(test$Exited, pred_max)
# (recall(test$Exited, pred_max) + 9 * fbeta_score(test$Exited, pred_max))/10
```

Slučajne šume (*eng. Random Forest*)

Pokušaćemo da nadmašimo ovaj rezultat koristeći *randomForest*. I ovde ponavljamo postupak 5 puta.

```
# max <- 0
# pred_max <- c()
# for(i in 1:5){
#   model1 <- randomForest(Exited ~ ., data=train1, proximity=TRUE)
#   for(i in 1:l){
#     pred <- predict(model1, as.matrix(test1)) > thresholds[i]
#     fbeta <- fbeta_score(test$Exited, pred)
#     recall <- recall(test$Exited, pred)
#     # uporedjujemo da li je preciznost veća od dosadašnje maksimalne
#     if((9*fbeta+recall) > max){
#       max <- 9*fbeta+recall
#       pred_max <- pred
#     }
#   }
# }
# recall(test$Exited, pred_max)
# fbeta_score(test$Exited, pred_max)
# (recall(test$Exited, pred_max) + 9 * fbeta_score(test$Exited, pred_max))/10
```

Logistička regresija

Sada pravimo model pomoću logističke regresije.

```
max <- 0
pred_max <- c()
model_glm1 <- glm(Exited ~ ., family = binomial(link = 'logit'), data = train1)
probs <- predict(model_glm1, test, type = 'response')
for (i in 1:l) {
  glm.pred <- ifelse(probs > thresholds[i], 1, 0)
  recall <- recall(test$Exited, glm.pred)
  fbeta <- fbeta_score(test$Exited, glm.pred)
  if ((9 * fbeta + recall) > max) {
    max <- 9 * fbeta + recall
    pred_max <- glm.pred
  }
}
recall(test$Exited, pred_max)
```

```
## [1] 0.6551724
```

```
fbeta_score(test$Exited, pred_max)
```

```
## [1] 0.3856362
```

```
(recall(test$Exited, pred_max) + 9 * fbeta_score(test$Exited, pred_max)) / 10
```

```
## [1] 0.4125898
```

Modeli nakon modifikacije parametara

Sada konstruišimo modele sa transformisanim prediktorima *NumOfProducts* i *Age*. Prediktor *NumOfProducts* uzima vrednosti 1, 2, 3 i 4, pri čemu smo videli da klijenti koji imaju vrednosti 1 ili 2 imaju tendenciju da ostanu u banci, a oni sa 3 ili 4 je uglavnom napuštaju. Stoga ćemo prediktor *NumOfProducts* transformisati tako da uzima vrednost 0 umesto 1 i 2, a vrednost 1 umesto 3 i 4. Takođe, videli smo da najmlađi i najstariji klijenti češće ostaju u banci nego oni srednjih godina, pa ćemo prediktor *Age* podeliti u dve kategorije, i to tako da u jednoj kategoriji budu klijenti mlađi od 30 i stariji od 60 godina, a ostali u drugoj.

```
train2$Age <- ifelse(train2$Age <= 30 | train2$Age >= 60, 1, 0) # transformacija prediktora
train2$NumOfProducts <- ifelse(train2$NumOfProducts > 2, 1, 0)
test2$Age <- ifelse(test2$Age <= 30 | test2$Age >= 60, 1, 0)
test2$NumOfProducts <- ifelse(test2$NumOfProducts > 2, 1, 0)
```

Neuronska mreža

Prvo pravimo model pomoću neuronskih mreža.

```
# training2 <- as.matrix(train2[, -which(names(data) %in% c("Exited"))])
# max <- 0
# pred_max <- c()
# for (i in 1:5)
# {
#   model_nn2 <- keras_model_sequential()
#   model_nn2 %>%
#     layer_dense(units = 10,
#                 activation = 'relu',
#                 input_shape = c(nc - 1)) %>%
#     layer_dense(units = 6,
#                 activation = 'relu',
#                 input_shape = c(10)) %>%
#     layer_dense(units = 2,
#                 activation = 'softmax')
#   model_nn2 %>% compile(loss = 'categorical_crossentropy',
#                         optimizer = 'adam',
#                         metrics = 'accuracy')
#   model_nn2 %>% fit(training2,
#                     trainLabels,
#                     epochs = 12,
#                     batch_size = 128,
#                     validation_split = 0.2,
#                     verbose = 0)
#   pred <- model_nn2 %>% predict(as.matrix(test2))
#   pred <- apply(pred, 1, which.max) - 1
#   fbeta <- fbeta_score(test$Exited, pred)
#   recall <- recall(test$Exited, pred)
#   if ((9 * fbeta + recall) > max) {
```

```
#     max <- 9 * fbeta + recall
#     pred_max <- pred
#   }
# }
# recall(test$Exited, pred_max)
# fbeta_score(test$Exited, pred_max)
# (recall(test$Exited, pred_max) + 9 * fbeta_score(test$Exited, pred_max)) / 10
```

Slučajne šume (*eng. Random Forest*)

Pravimo *randomForest* model.

```
# max <- 0
# pred_max <- c()
# for (i in 1:5) {
#   model2 <- randomForest(Exited ~ ., data = train2, proximity = TRUE)
#   for (i in 1:l) {
#     pred <- predict(model2, as.matrix(test2)) > thresholds[i]
#     fbeta <- fbeta_score(test$Exited, pred)
#     recall <- recall(test$Exited, pred)
#     if ((9 * fbeta + recall) > max) {
#       max <- 9 * fbeta + recall
#       pred_max <- pred
#     }
#   }
# }
# recall(test$Exited, pred_max)
# fbeta_score(test$Exited, pred_max)
# (recall(test$Exited, pred_max) + 9 * fbeta_score(test$Exited, pred_max)) / 10
```

Logistička regresija

Pravimo model pomoću logističke regresije.

```
test3 <- test2
test3$Exited <- test$Exited
max <- 0
pred_max <- c()
model_glm2 <- glm(Exited ~ ., family = binomial(link = 'logit'), data = train2)
probs <- predict(model_glm2, test3, type = 'response')
for (i in 1:l) {
  glm.pred2 <- ifelse(probs > thresholds[i], 1, 0)
  recall <- recall(test$Exited, glm.pred2)
  fbeta <- fbeta_score(test$Exited, glm.pred2)
  if ((9 * fbeta + recall) > max) {
    max <- 9 * fbeta + recall
    pred_max <- glm.pred2
  }
}
recall(test$Exited, pred_max)
```

```
## [1] 0.8992042
```

```
fbeta_score(test$Exited, pred_max)
```

```
## [1] 0.3446873
```

```
(recall(test$Exited, pred_max) + 9 * fbeta_score(test$Exited, pred_max)) / 10
```

```
## [1] 0.400139
```