RMarkdown/ HTML skripta 'Mašinsko učenje u R-u' za moje studente na smeru Biomedicinsko inženjerstvo, generacija 2016/ 2017. Materijal ce biti vremenom dopunjavan.

Igor Hut

December 5, 2016

Contents

Uvodna analiza podataka

- `data.frame` - primarna forma za smestanje, analizu i manipulaciju podacima
- Uvoz podataka - licna preporuka `readr` (Hadley Wickham)
- Prvi korak - upoznavanje sa podacima
- Dimenzije - broj promenljivih (features) i opservacija (observations)
- Ciscenje i uredjivanje podataka - licna preporuka `dplyr` i `tidyr`(Hadley Wickham)
- Vizualizacija - `base` i `ggplot2` (Hadley Wickham)

Primer inicijalne provere i analize podataka

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
# broj promenljivih i broj opservacija
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 ...
```

```r
dim(iris)
```

```
## [1] 150   5
```

```r
# Nekoliko prvih i poslednjih vrsta iz `iris` baze
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
```

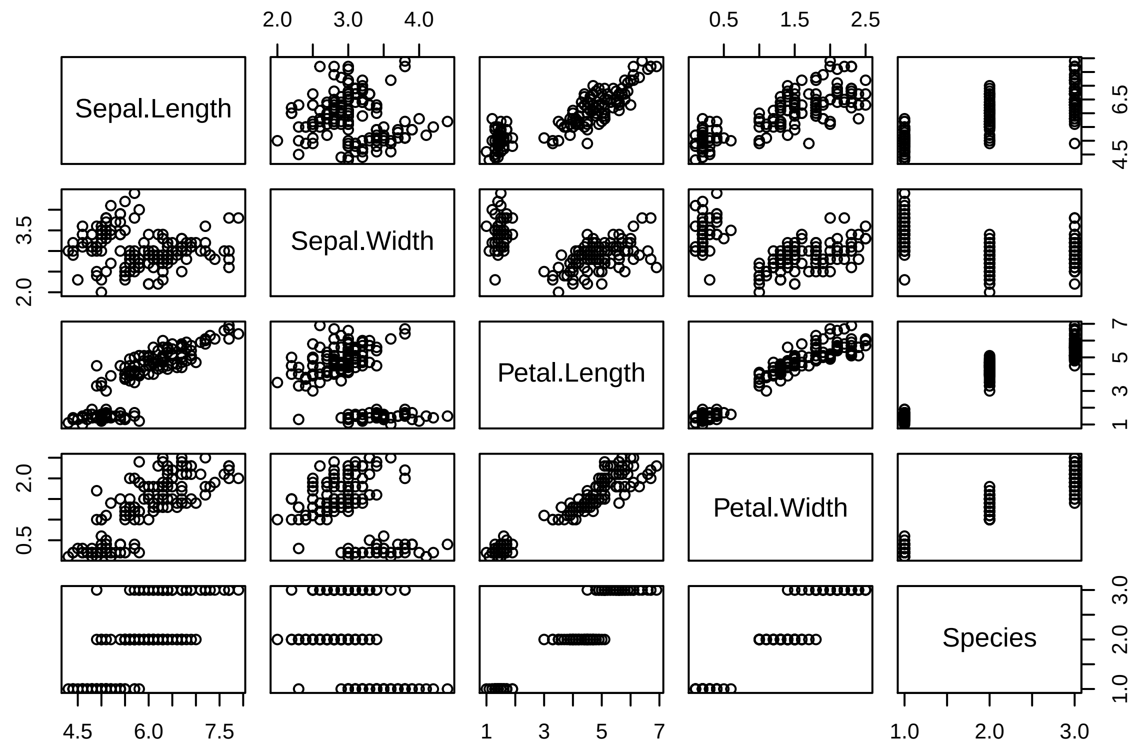
```r
tail(iris)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

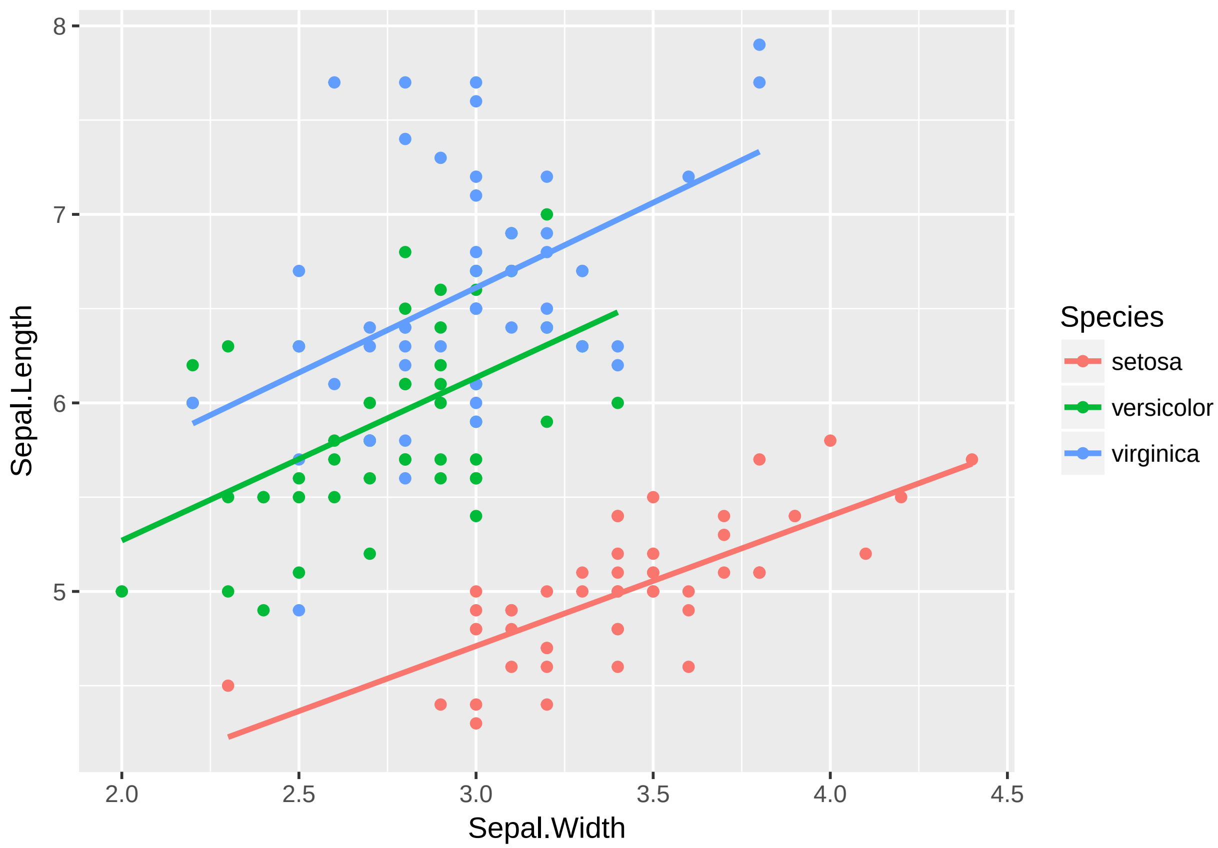
```r
# sumarna statistika za podatke u `iris` bazi
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
##        Species  
##  setosa    :50  
##  versicolor:50  
##  virginica :50  
##                 
##                 
##                 
```

```r
plot(iris) #rezultat ce biti isti kao da smo upotrebili `pairs` funkciju iz `base` bilioteke
```




```
ggplot(iris, aes( x = Sepal.Width, y = Sepal.Length, col = Species)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



Regresija, klasifikacija, klasterizacija - Uvod

Linearna regresija - uvodni primer 1

```
# Ucitavamo "Wage" bazu iz "ISLR" paketa koja sadrzi neke opste podatke o radnicima u srednje-Atlanstko
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.3.2
```

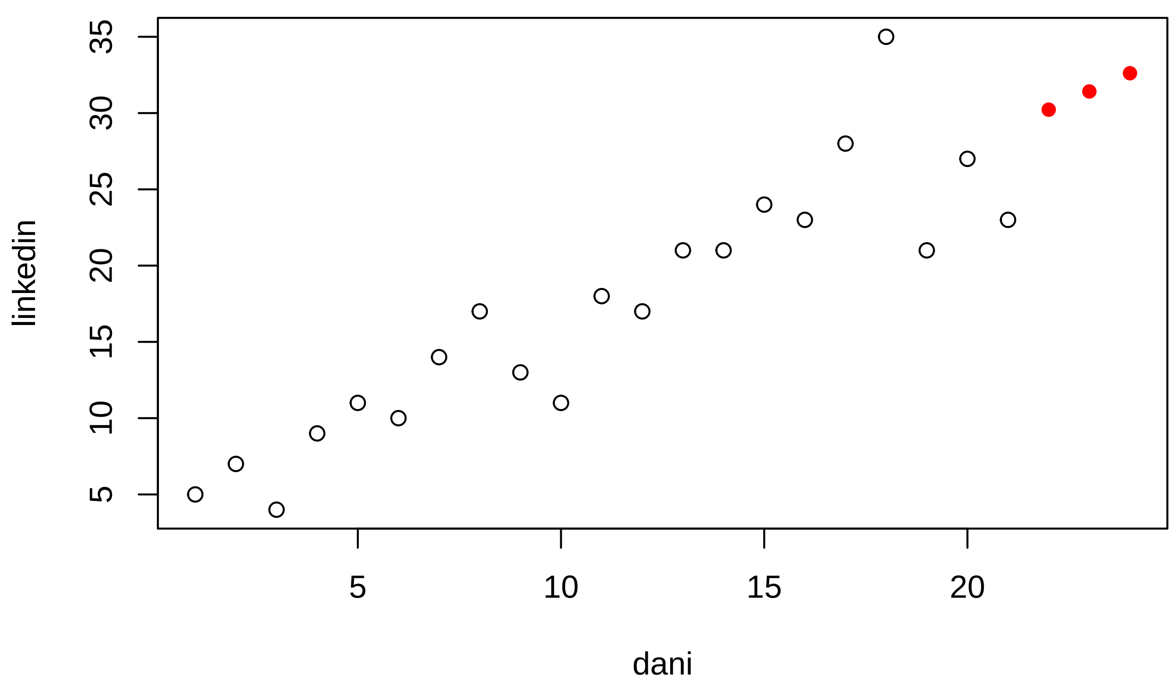
```
data("Wage")


# Generisemo linearni model "lm_wage"

lm_wage <- lm(wage ~ age, data = Wage)

# ?lm - kako se funkcija "lm()" koristi

# Definisemo data.frame sa novim vrednostima, koje nisu koriscenje za sintezu modela: "unseen"
unseen <- data.frame(age = 60)


# Na osnovu modela "lm_wage" predvidjamo koliko iznosi plata 60-togodisnjeg radnika
predict(lm_wage, unseen)
```

```
##        1
```

```
## 124.1413
```

Regresija - uvodni primer 2

```r
# Broj pregleda vased LinkedIn profila u periodu od tri nedelje
linkedin <- c(5, 7,  4,  9, 11, 10, 14, 17, 13, 11, 18, 17, 21, 21, 24, 23, 28, 35, 21, 27, 23)

# Vektor koji sadrzi korespodentne dane: "dani"
dani <- 1:21

# Linearni model - broj pregleda po danima: linkedin_lm

linkedin_lm <- lm(linkedin ~ dani)

# Predvidjamo broj pregleda u sledeca tri dana: linkedin_pred
buduci_dani <- data.frame(dani = 22:24)
linkedin_pred <- predict(linkedin_lm, buduci_dani)

# Plotujemo "istorijske" podatke i predvidjanje
plot(linkedin ~ dani, xlim = c(1, 24))
points(22:24, linkedin_pred, col = "red", pch = 16)
```



Klasifikacija - uvodni primer

Primer neadekvatnog klasifikatora - krajnje overfitovanje podataka iz trening seta!

```r
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.3.2
```

```r
if (!"emails" %in% ls()) {
    emails <- read_csv("data/emails_small.csv")
}
```

```
## Parsed with column specification:
## cols(
##   avg_capital_seq = col_double(),
##   spam = col_integer()
## )
```

```r
# Proveravamo strukturu seta podataka
str(emails)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    13 obs. of  2 variables:
##  $ avg_capital_seq: num  1 2.11 4.12 1.86 2.97 ...
##  $ spam           : int  0 0 1 0 1 0 1 0 0 1 ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 2
##   .. ..$ avg_capital_seq: list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ spam           : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```r
# Definisemo funkciju spam_classifier()
# 1 - spam, 0 - ham
spam_classifier <- function(x){
  prediction <- rep(NA,length(x))
  prediction[x > 4] <- 1
  prediction[x >= 3 & x <= 4] <- 0
  prediction[x >= 2.2 & x < 3] <- 1
  prediction[x >= 1.4 & x < 2.2] <- 0
  prediction[x > 1.25 & x < 1.4] <- 1
  prediction[x <= 1.25] <- 0
  return(prediction)
}

# Primenimo nas klasifikator na kolonu "avg_capital_seq": "spam_pred"
spam_pred <- spam_classifier(emails$avg_capital_seq)

# Uporedimo "spam_pred" i  "emails$spam"
spam_pred == emails$spam
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```r
identical(spam_pred, as.numeric(emails$spam))
```

```
## [1] TRUE
```

Klasterovanje - uvodni primer

```r
# Da bi smo obezbedili reproduktibilnost
set.seed(1)

# Proveravamo strukturu podataka
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 ...
```

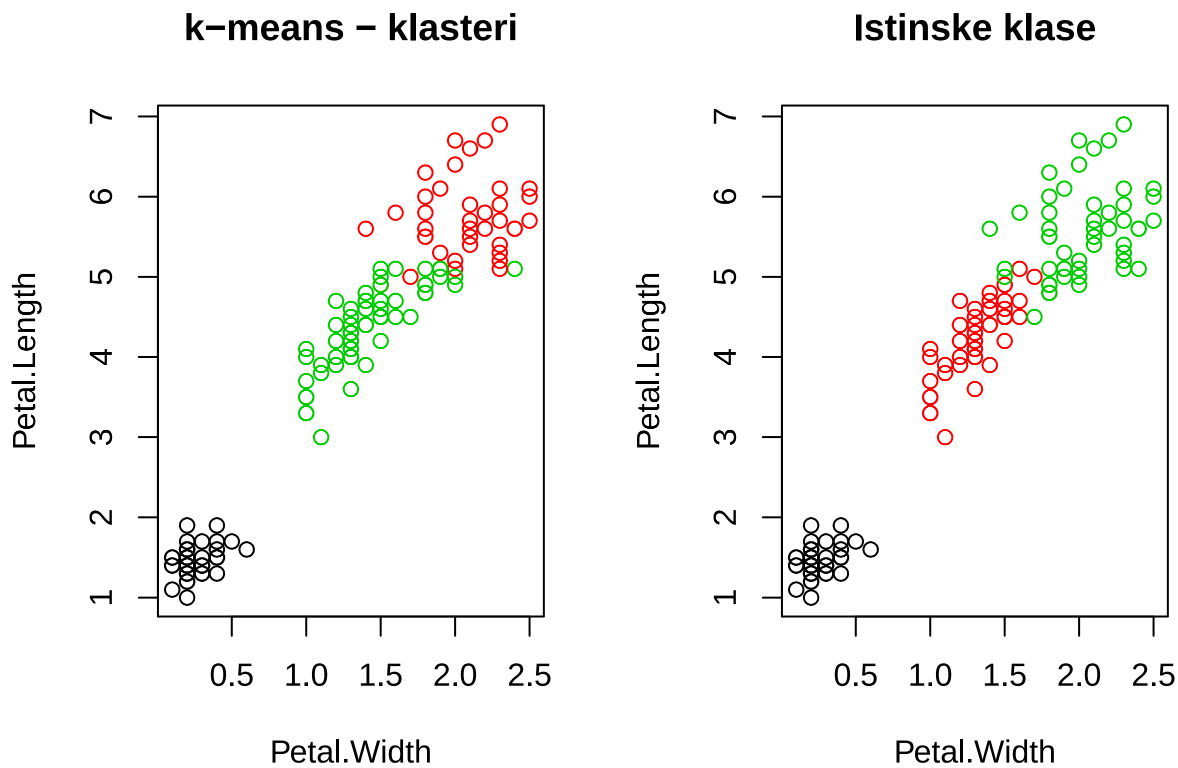
```r
head(iris)
```

```
##   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
```

```r
# Delimo "iris" na dva seta: "my_iris" i "species""
my_iris <- iris[-5]
species <- iris$Species

# Vrsimo k-means klasterizaciju za "my_iris", pretpostavljamo da postoje tri klase: "kmeans_iris"
kmeans_iris <- kmeans(my_iris,3)

# Poredimo dobijene klastere sa istinskim klasama (kategorijama)
table(species, kmeans_iris$cluster)
```

```
##
## species       1  2  3
##   setosa     50  0  0
##   versicolor  0  2 48
##   virginica   0 36 14
```

```r
# Plotujemo "Petal.Width" vs "Petal.Length", bojimo po klasterima odn. postojecim kategorijama
par(mfrow = c(1,2))
plot(Petal.Length ~ Petal.Width, data = my_iris, col = kmeans_iris$cluster)
title("k-means - klasteri")
plot(Petal.Length ~ Petal.Width, data = my_iris, col = iris$Species)
title("Istinske klase")
```



Ocena modela

Konfuziona matrica - Primeri

- Objasnicemo ocenu klasifikacionog modela (u ovom slucaju korisceno je stablo odlucivanja - Decision Tree), na osnovu matrice konfuzije, koristeci za primer `titanic` set podataka u kome se nalaze podaci o putnicima na Titaniku.
  - Verzija koju cemo mi koristiti moze da se preuzme sa Kaggle stranice: Titanic: Machine Learning from Disaster

Primer 1:

```
library(rpart)
library(readr)
library(purrr)
```

```
##
## Attaching package: 'purrr'

## The following objects are masked from 'package:dplyr':
##
##     contains, order_by
```

```
# Import podataka
if (!"titanic" %in% ls()) {
```

```r
    titanic <- read_csv("data/train.csv")
}
```

```
## Parsed with column specification:
## cols(
##   survived = col_integer(),
##   pclass = col_integer(),
##   name = col_character(),
##   sex = col_character(),
##   age = col_double(),
##   sibsp = col_integer(),
##   parch = col_integer(),
##   ticket = col_character(),
##   fare = col_double(),
##   cabin = col_character(),
##   embarked = col_character()
## )
```

```r
# Da obezbedimo reproduktibilnost
set.seed(33)

# Proveravamo strukturu data seta
str(titanic)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    891 obs. of  11 variables:
##  $ survived: int  0 1 1 1 0 0 0 0 1 1 ...
##  $ pclass  : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ name    : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)" "He
##  $ sex     : chr  "male" "female" "female" "female" ...
##  $ age     : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ sibsp   : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ parch   : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ ticket  : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ fare    : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ cabin   : chr  NA "C85" NA "C123" ...
##  $ embarked: chr  "S" "C" "S" "S" ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 11
##   .. ..$ survived: list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ pclass  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ name    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ sex     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ age     : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ sibsp   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ parch   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ ticket  : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ fare    : list()
```

```
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ cabin   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ embarked: list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```r
# Koristicemo samo kolone 'survived', 'pclass', 'sex' i 'age'
titanic <- titanic[, c(1, 2, 4, 5)]
str(titanic)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    891 obs. of  4 variables:
##  $ survived: int  0 1 1 1 0 0 0 0 1 1 ...
##  $ pclass  : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ sex     : chr  "male" "female" "female" "female" ...
##  $ age     : num  22 38 26 35 35 NA 54 2 27 14 ...
```

```r
# Prve tri promenlive bi evidentno trebalo da budu tretirane kao kategoricke promenljive - faktori
titanic[-4] <- map(titanic[-4], as.factor)

str(titanic)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    891 obs. of  4 variables:
##  $ survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ pclass  : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ sex     : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ age     : num  22 38 26 35 35 NA 54 2 27 14 ...
```

```r
table(titanic$survived)
```

```
##
##   0   1
## 549 342
```

```r
# Odnos broja prezivelih i poginulih
prop.table(table(titanic$survived))
```

```
##
##         0         1
## 0.6161616 0.3838384
```

```r
# Generisemo klasifikacioni model (drvo odlucivanja - decision tree) na osnovu datih podataka:
tree <- rpart(survived ~ ., data = titanic, method = "class")

# Koristimo predict() funkciju da predvidimo klase
pred <- predict(tree, newdata = titanic, type = "class")


# Konstruisemo konfuzionu matricu koristeci "table()":
conf_t <- table(titanic$survived, pred)
conf_t
```

```
##    pred
##       0   1
##   0 479  70
##   1  94 248
```

Primer 2:

```r
#Isto to sa "pima" bazom podataka
library(faraway)
```

```
## Warning: package 'faraway' was built under R version 3.3.2

##
## Attaching package: 'faraway'

## The following object is masked from 'package:rpart':
##
##     solder
```

```r
data(pima)

head(pima)
```

```
##   pregnant glucose diastolic triceps insulin  bmi diabetes age test
## 1        6     148        72      35       0 33.6    0.627  50    1
## 2        1      85        66      29       0 26.6    0.351  31    0
## 3        8     183        64       0       0 23.3    0.672  32    1
## 4        1      89        66      23      94 28.1    0.167  21    0
## 5        0     137        40      35     168 43.1    2.288  33    1
## 6        5     116        74       0       0 25.6    0.201  30    0
```

```r
str(pima)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ pregnant : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose  : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ diastolic: int  72 66 64 66 40 74 50 0 70 96 ...
##  $ triceps  : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ insulin  : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ bmi      : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ diabetes : num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age      : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ test     : int  1 0 1 0 1 0 1 0 1 1 ...
```

```r
# Da bismo obezbedili reproduktibilnost
set.seed(33)

# Generisemo klasifikacioni model (drvo odlucivanja - decision tree) na osnovu datih podataka:
tree <- rpart(test ~ ., data = pima, method = "class")

# Koristimo predict() funkciju da predvidimo klase
pred <- predict(tree, newdata = pima, type = "class")


# Konstruisemo konfuzionu matricu koristeci "table()":
conf_p <- table(pima$test, pred)
conf_p
```

```
##    pred
##       0   1
##   0 449  51
##   1  72 196
```

Tacnost, preciznost, senzitivnost (recall), specificnost - Primer

```
# Izracunajmo parametre za ocenu valjanosti modela "tree" za "titanic" skup podataka

# Formiramo TP, FN, FP i TN na osnovu "conf_t"

TP <- conf_t[2,2]

FP <- conf_t[1,2]

FN <- conf_t[2,1]

TN <- conf_t[1,1]


# Tacnost (Accuracy)
acc <- (TP + TN)/sum(conf_t)
acc
```

```
## [1] 0.8159371
```

```
# Preciznost (Precision)
prec <- TP/(TP + FP)
prec
```

```
## [1] 0.7798742
```

```
# Senzitivnost (Sensitivity, Recall)
sens <- TP/(TP + FN)
sens
```

```
## [1] 0.7251462
```

```
# Specificnost (Specificity)
spec <- TN/(TN + FP)
spec
```

```
## [1] 0.8724954
```

Zadatak za vezbanje na casu:

Izracunajte ove vrednosti za "tree" model generisan na osnovu "pima" seta podataka.

Kvalitet regresije

- Srednja kvadratna greska
- U nasem slucaju mozemo smatrati da se poklapa sa standardnom devijacijom
- `sqrt((1/nrow(truth)) * sum( (truth$col - pred)^2))`

Primer:

```
# Koristicemo "pima" bazu

# Struktura seta podataka
str(pima)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ pregnant : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ glucose  : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ diastolic: int  72 66 64 66 40 74 50 0 70 96 ...
##  $ triceps  : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ insulin  : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ bmi      : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ diabetes : num  0.627 0.351 0.672 0.167 2.288 ...
##  $ age      : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ test     : int  1 0 1 0 1 0 1 0 1 1 ...
```

```r
# Multivarijabilna linearna regresija - prostiji model (ukljucen manji broj promenljivih)
fit_1 <- lm(diabetes ~ bmi + triceps + age + glucose, data = pima)

# Predvidjanje na osnovu modela: pred_1
pred_1 <- predict(fit_1)

# RMSE na osnovu "pima$diabetes" (tacne vrednosti) i "pred_1" (vrednosti na osnovu modela fit_1)
rmse_1 <- sqrt(1/nrow(pima)*sum((pima$diabetes - pred_1) ^ 2))

rmse_1
```

```
## [1] 0.3222776
```

```r
# Multivarijabilna linearna regresija - kompleksniji model (ukljucen veci broj promenljivih)
fit_2 <- lm(diabetes ~ bmi + triceps + age + glucose + diastolic + insulin + pregnant, data = pima)

# Predvidjanje na osnovu modela: pred_1
pred_2 <- predict(fit_2)

# RMSE na osnovu "pima$diabetes" (tacne vrednosti) i "pred_1" (vrednosti na osnovu modela fit_1)
rmse_2 <- sqrt(1/nrow(pima)*sum((pima$diabetes - pred_2) ^ 2))

rmse_2
```

```
## [1] 0.3205351
```

Procena valjanosti klasterizacije: WSS vs BSS

```r
# Da bi smo obezbedili reproduktibilnost
set.seed(33)

# Proveravamo strukturu podataka
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 ...
```

```r
head(iris)
```

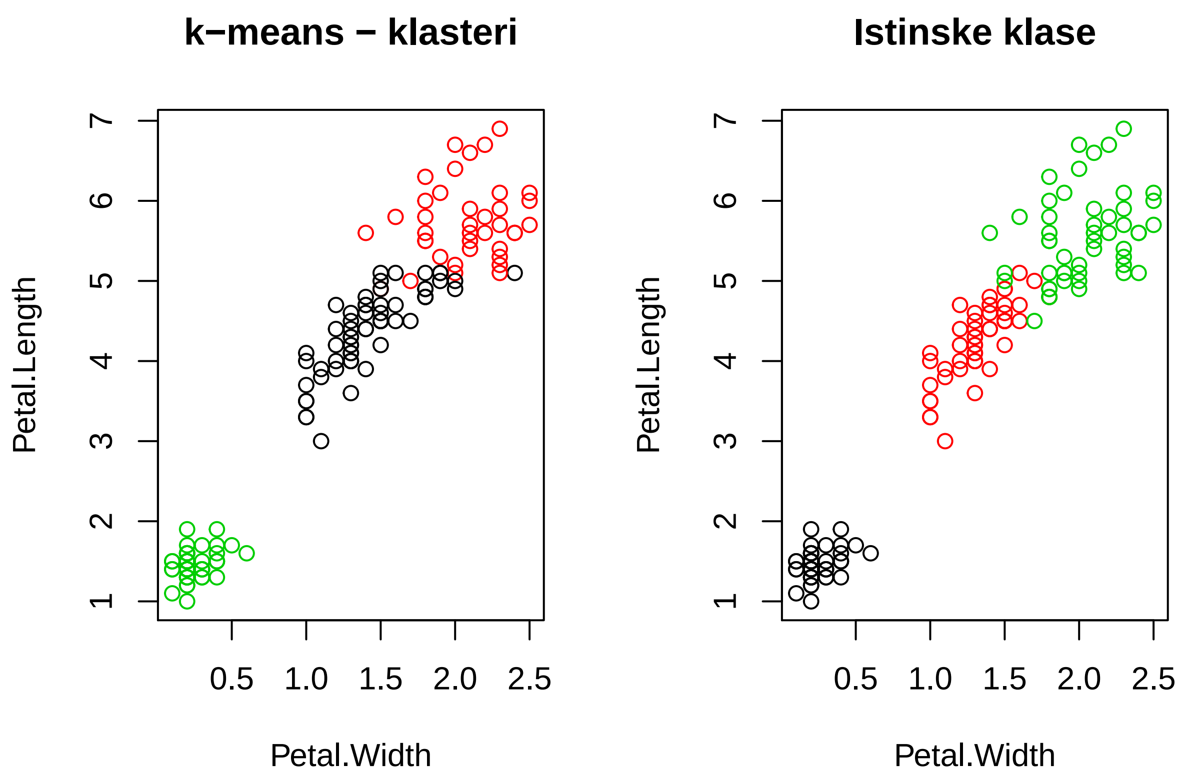
```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
```

```
## 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
```

```r
# Delimo "iris" na dva seta: "my_iris" i "species""
my_iris <- iris[-5]
species <- iris$Species

# Vrsimo k-means klasterizaciju za "my_iris" uz pretpostavku da postoje tri klase: "kmeans_iris"
kmeans_iris <- kmeans(my_iris,3)

# Poredimo dobijene klastere sa istinskim klasama (kategorijama)
table(species, kmeans_iris$cluster)
```

```
##
## species       1  2  3
##   setosa      0  0 50
##   versicolor 48  2  0
##   virginica  14 36  0
```

```r
# Plotujemo "Petal.Width" vs "Petal.Length", bojimo po klasterima odn. postojecim kategorijama
par(mfrow = c(1,2))
plot(Petal.Length ~ Petal.Width, data = my_iris, col = kmeans_iris$cluster)
title("k-means - klasteri")
plot(Petal.Length ~ Petal.Width, data = my_iris, col = iris$Species)
title("Istinske klase")
```




```
kmeans_iris$tot.withinss/kmeans_iris$betweenss
```

```
## [1] 0.1308696
```

Trening set i test set

- Cilj implementacije algoritma **nadgledanog** ucenja jeste dobijanje "dovoljno" dobrog prediktivnog modela na osnovu raspolozivog seta podataka.

- Set podataka koji se koristi za formiranje modela - **trening set**
- Set podatak koji se koristi za procenu valjanosti modela - **test set**
- Trening set i test set ne smeju imati/deliti zajednicke elemente tj. opservacije
- Samo testiranjem modela na podacima koji nisu korisceni za ucenje mozemo izvesti adekvatnu estimaciju ocena valjanosti modela - generalizacija.
- Opste prihvacena praksa je da se rasploziv skup podataka podeli na sledeci nacin:
  - Trening set 70% ili 75%
  - Test set 30% ili 35%
- Prilikom podele raspolozivog skupa podataka treba strogo voditi racuna da zastupljenost, odn. distribucija, klasa (ovo se odnosi na algoritme za klasifikaciju) bude slicna u trening i test setu
  - ne bi smelo da se dogodi da jedan ili drugi set uopste ne sadrze ni jednu opservacuju koja pripada odredjenoj klasi
- Dobra praksa je da se poredak opservacija randomizuje (slucajno odabrana permutacija) pre deljenja skupa podataka na trening i test set
  - Ovo vazi i za klasifikaciju i za regresiju
- Odabiranje (semplovanje) opservacija za trening i test set moze ponekad i znacajno uticati na procenjene vrednosti ocena valjanosti datog modela

- Da bi se ovaj efekat minimizovao koristi se **unakrsna validacija** (cross-validation)

Primer

```
# Koristicemo "titanic" set podataka formiran u jednom od prethodnih primera
str(titanic)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    891 obs. of  4 variables:
##  $ survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ pclass  : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ sex     : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ age     : num  22 38 26 35 35 NA 54 2 27 14 ...
```

```
table(titanic$survived)
```

```
##
##   0   1
## 549 342
```

```
# Odnos prezivelih i poginulih
prop.table(table(titanic$survived))
```

```
##
##         0         1
## 0.6161616 0.3838384
```

```
# Da bismo omogucili reproduktibilnost
set.seed(33)

# Prvo napravimo jednu slucajno odabranu permutaciju celog skupa podataka (dataset shuffle)
n <- nrow(titanic)
shuffled <- titanic[sample(n),] #f-a 'sample' vrsi slucajno odabiranje elemenata zadatog vektora

# Delimo skup podataka na trening i test set (70% i 30%)
train_indicies <- 1:round(0.7 * n)
train <- shuffled[train_indicies, ]
test <- shuffled[-train_indicies, ]

# Generisemo klasifikacioni model (drvo odlucivanja - decision tree) na osnovu trening seta:
tree <- rpart(survived ~ ., data = train, method = "class")

# Koristeci dobijeni model "tree" vrsimo klasifikaciju podataka iz test seta:
pred <- predict(tree, newdata = test, type = "class")

# Racunamo matricu konfuzije
conf_t <- table(test$survived, pred)

# Prikaz matrice konfuzije
conf_t
```

```
##    pred
##       0   1
##   0 128  28
##   1  36  75
```

```
# Formiramo TP, FN, FP i TN na osnovu "conf_t"
```

```r
TP <- conf_t[2,2]

FP <- conf_t[1,2]

FN <- conf_t[2,1]

TN <- conf_t[1,1]


# Tacnost (Accuracy)
acc <- (TP + TN)/sum(conf_t)
acc
```

```
## [1] 0.7602996
```

```r
# Preciznost (Precision)
prec <- TP/(TP + FP)
prec
```

```
## [1] 0.7281553
```

```r
# Senzitivnost (Sensitivity, Recall)
sens <- TP/(TP + FN)
sens
```

```
## [1] 0.6756757
```

```r
# Specificnost (Specificity)
spec <- TN/(TN + FP)
spec
```

```
## [1] 0.8205128
```

Zadatak za vezbanje na casu:

Ponovite pokazanu proceduru koristeci "pima" skup podataka.

Upotreba unakrsne validacije (cross-validation)

Radi demonstracije cemo rucno formirati algroritam koji koristi unakrsnu validaciju za procenu tacnosti modela:

```r
# Da bismo obezbedili reproduktibilnost
set.seed(33)

# Koristicemo prethodno formirani "shuffled" skup podataka

# Inicijalizujemo vektor accs - popunjavamo nulama
accs <- rep(0,9)


# Treniramo model koristeci kros-validacione intervale vrednosti i vrsimo estimaciju tacnosti modela ka
for (i in 1:9) {
  # Ovi indeksi ukazuju na trenutni interval test seta koji koristimo za treniranje modela
    indices <- (((i - 1) * round((1/9)*nrow(shuffled))) + 1):((i*round((1/9) * nrow(shuffled))))
```

```
   # Iskljucujemo ove intervale iz trening seta
   train <- shuffled[-indices,]

  # Ukljucimo ih u test set
  test <- shuffled[indices,]

  # Treniramo model sa svakim od dobijenih trening setova po iteracijama
  tree <- rpart(survived ~ ., train, method = "class")

  # Predvidjamo klase za tekuci test set u svakoj od iteracija
  pred <- predict(tree, test, type = "class")

  # Formiramo odgovarajucu konfuzionu matricu
  conf <- table(test$survived, pred)

  # Dodeljujemo vrednost za tacnost tekuceg modela i-tom indeksu u vektoru accs
  accs[i] <- sum(diag(conf))/sum(conf)
}

# Srednja vrednost za accs
mean(accs)
```

```
## [1] 0.7833895
```

Pitanje: Recimo da primenjujemo unakrsnu validaciju na skupu podataka koji sadrzi 22680 opservacija. Zelite da vas trening set sadrzi 21420 unosa (opservacija). Koliko iteracija moze da sadrzi kros-validacioni algoritam?

Bajas i varijansa (Bias and Variance)

Primer

Koristicemo *Spambase Data Set* koji mozete naci na https://archive.ics.uci.edu/ml/datasets/Spambase

```
if (!"emails_full" %in% ls()) {
    emails_full <- read.csv("data/spambase.data", header = FALSE)
}

# Proveravamo strukturu seta podataka
str(emails_full)
```

```
## 'data.frame':    4601 obs. of  58 variables:
##  $ V1 : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ V2 : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ V3 : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ V4 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V5 : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
##  $ V6 : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ V7 : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...
##  $ V8 : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...
##  $ V9 : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.06 ...
##  $ V10: num  0 0.94 0.25 0.63 0.63 0 0.64 0 0.76 0 ...
##  $ V11: num  0 0.21 0.38 0.31 0.31 0 0.96 0 0.76 0 ...
##  $ V12: num  0.64 0.79 0.45 0.31 0.31 0 1.28 0 0.92 0.64 ...
##  $ V13: num  0 0.65 0.12 0.31 0.31 0 0 0 0 0.25 ...
```

```
##  $ V14: num  0 0.21 0 0 0 0 0 0 0 0 ...
##  $ V15: num  0 0.14 1.75 0 0 0 0 0 0 0.12 ...
##  $ V16: num  0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...
##  $ V17: num  0 0.07 0.06 0 0 0 0 0 0 0 ...
##  $ V18: num  1.29 0.28 1.03 0 0 0 0.32 0 0.15 0.12 ...
##  $ V19: num  1.93 3.47 1.36 3.18 3.18 0 3.85 0 1.23 1.67 ...
##  $ V20: num  0 0 0.32 0 0 0 0 0 3.53 0.06 ...
##  $ V21: num  0.96 1.59 0.51 0.31 0.31 0 0.64 0 2 0.71 ...
##  $ V22: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V23: num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
##  $ V24: num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
##  $ V25: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V26: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V27: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V28: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V29: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V30: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V31: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V32: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V33: num  0 0 0 0 0 0 0 0 0.15 0 ...
##  $ V34: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V35: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V36: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V37: num  0 0.07 0 0 0 0 0 0 0 0 ...
##  $ V38: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V39: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V40: num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ V41: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V42: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V43: num  0 0 0.12 0 0 0 0 0 0.3 0 ...
##  $ V44: num  0 0 0 0 0 0 0 0 0 0.06 ...
##  $ V45: num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ V46: num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ V47: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V48: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V49: num  0 0 0.01 0 0 0 0 0 0 0.04 ...
##  $ V50: num  0 0.132 0.143 0.137 0.135 0.223 0.054 0.206 0.271 0.03 ...
##  $ V51: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ V52: num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...
##  $ V53: num  0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...
##  $ V54: num  0 0.048 0.01 0 0 0 0 0 0.022 0 ...
##  $ V55: num  3.76 5.11 9.82 3.54 3.54 ...
##  $ V56: int  61 101 485 40 40 15 4 11 445 43 ...
##  $ V57: int  278 1028 2259 191 191 54 112 49 1257 749 ...
##  $ V58: int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Na osnovu dokumentacije...

emails_full <- emails_full[, c(55, 58)]

str(emails_full)
```

```
## 'data.frame':    4601 obs. of  2 variables:
##  $ V55: num  3.76 5.11 9.82 3.54 3.54 ...
##  $ V58: int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
colnames(emails_full) <-  c("avg_capital_seq", "spam")

str(emails_full)
```

```
## 'data.frame':    4601 obs. of  2 variables:
##  $ avg_capital_seq: num  3.76 5.11 9.82 3.54 3.54 ...
##  $ spam           : int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
emails_full$spam <- as.factor(emails_full$spam)

str(emails_full)
```

```
## 'data.frame':    4601 obs. of  2 variables:
##  $ avg_capital_seq: num  3.76 5.11 9.82 3.54 3.54 ...
##  $ spam           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```r
# Definisemo funkciju spam_classifier()
# 1 - spam, 0 - ham
spam_classifier <- function(x){
  prediction <- rep(NA,length(x))
  prediction[x > 4] <- 1
  prediction[x >= 3 & x <= 4] <- 0
  prediction[x >= 2.2 & x < 3] <- 1
  prediction[x >= 1.4 & x < 2.2] <- 0
  prediction[x > 1.25 & x < 1.4] <- 1
  prediction[x <= 1.25] <- 0
  return(factor(prediction, levels = c("0","1")))
}

# Primenimo spam_classifier na emails_full: pred_full
pred_full <- spam_classifier(emails_full$avg_capital_seq)

# Konfuziona matrica za emails_full: conf_full
conf_full <- table(emails_full$spam, pred_full)

# Racunamo tacnost na osnovu conf_full: acc_full
acc_full <- sum(diag(conf_full))/sum(conf_full)
acc_full
```

```
## [1] 0.6561617
```

```r
# Uproscen model za klasifikaciju
spam_classifier <- function(x){
  prediction <- rep(NA,length(x))
  prediction[x > 4] <- 1
  prediction[x <= 4] <- 0
  return(factor(prediction, levels = c("0","1")))
}

# Tacnost predikcije sa uproscenim modelom za emails data set
conf_small <- table(emails$spam, spam_classifier(emails$avg_capital_seq))
acc_small <- sum(diag(conf_small)) / sum(conf_small)
acc_small
```

```
## [1] 0.7692308
```

```r
# Primenimo uprosceni model i na "emails_full" i sracunamo konfuzionu matricu
conf_full <- table(emails_full$spam, spam_classifier(emails_full$avg_capital_seq))

# Izracunamo tacnost
acc_full <- sum(diag(conf_full)) / sum(conf_full)
acc_full
```

```
## [1] 0.7259291
```

Linearna regresija

- "Jednostavan" pristup problemu "nadgledanog" ucenja.
- **Osnovna prepostavka:** Zavisno promenljiva Y se moze izracunati kao linearna funkcija nezavisno promenljivih $X_1, X_2, ..., X_p$.
- Literatura: Regression Models for Data Science in R

Prosta linearna regresija

Koriscenje funkcije `lm()` - Primeri

Primer 1:

U ovom primeru cemo koristiti set podataka "Boston" iz paketa `MASS`. Ovaj set podataka sadrzi podatke o trzisnoj vrednosti nekretnina u predgradjima Bostona, SAD, zajedno sa razlicitim parametrima koji uticu na formiranje ove vrednosti.

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(ISLR)
library(ggplot2)

?Boston
```

```
## starting httpd help server ...

##  done
```

```r
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
```

```
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```r
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12
##   lstat medv
## 1  4.98 24.0
## 2  9.14 21.6
## 3  4.03 34.7
## 4  2.94 33.4
## 5  5.33 36.2
## 6  5.21 28.7
```

```r
# Proverimo kako izgleda promena "medv" (median value of owner-occupied homes in \$1000s) sa
# "lstat" (lower status of the population (percent)
plot(medv~lstat,Boston)

# Kao sto vidimo postoji jasan trend opadanja vrednosti nekretnina sa porastom procenta
# siromasnijih stanovnika (ovakva korelacija je naravno i ocekivana). Ovakvi slucajevi su dobri
# kandidati za modelovanje prostom linerarnom regresijom.

fit_1 = lm(medv ~ lstat, data = Boston)

# Hajde da vidimo kako izgleda nas model
fit_1
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

```r
# Detaljniji uvid
summary(fit_1)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

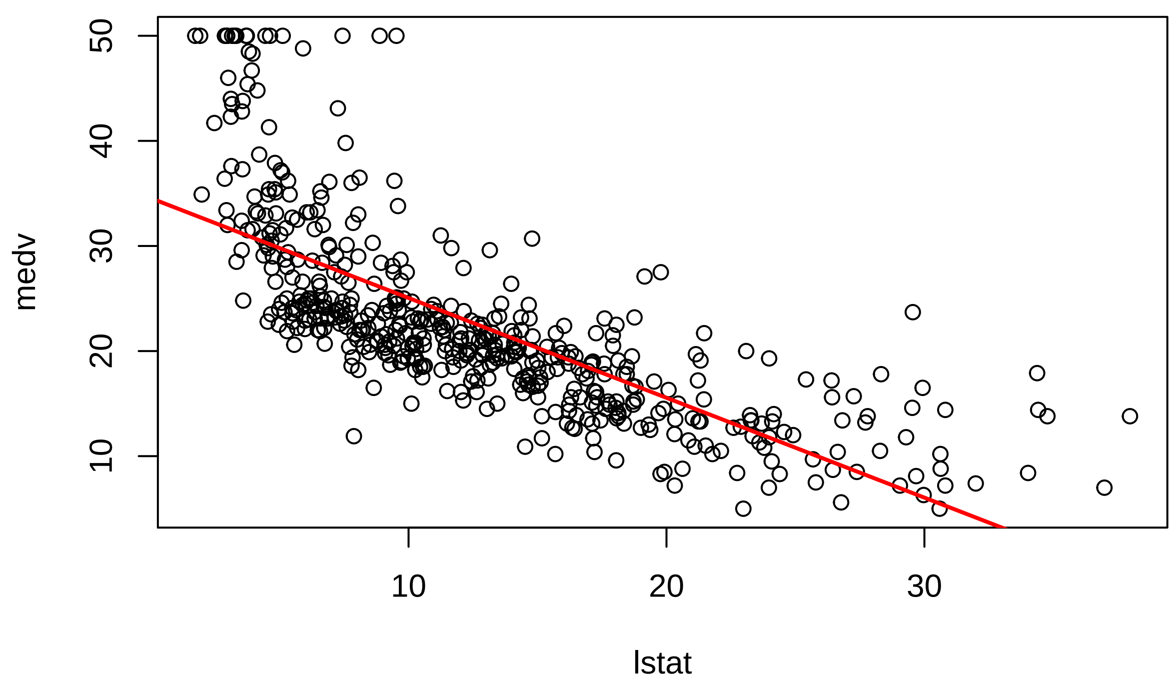
```r
# Sta sve model sadrzi
names(fit_1)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

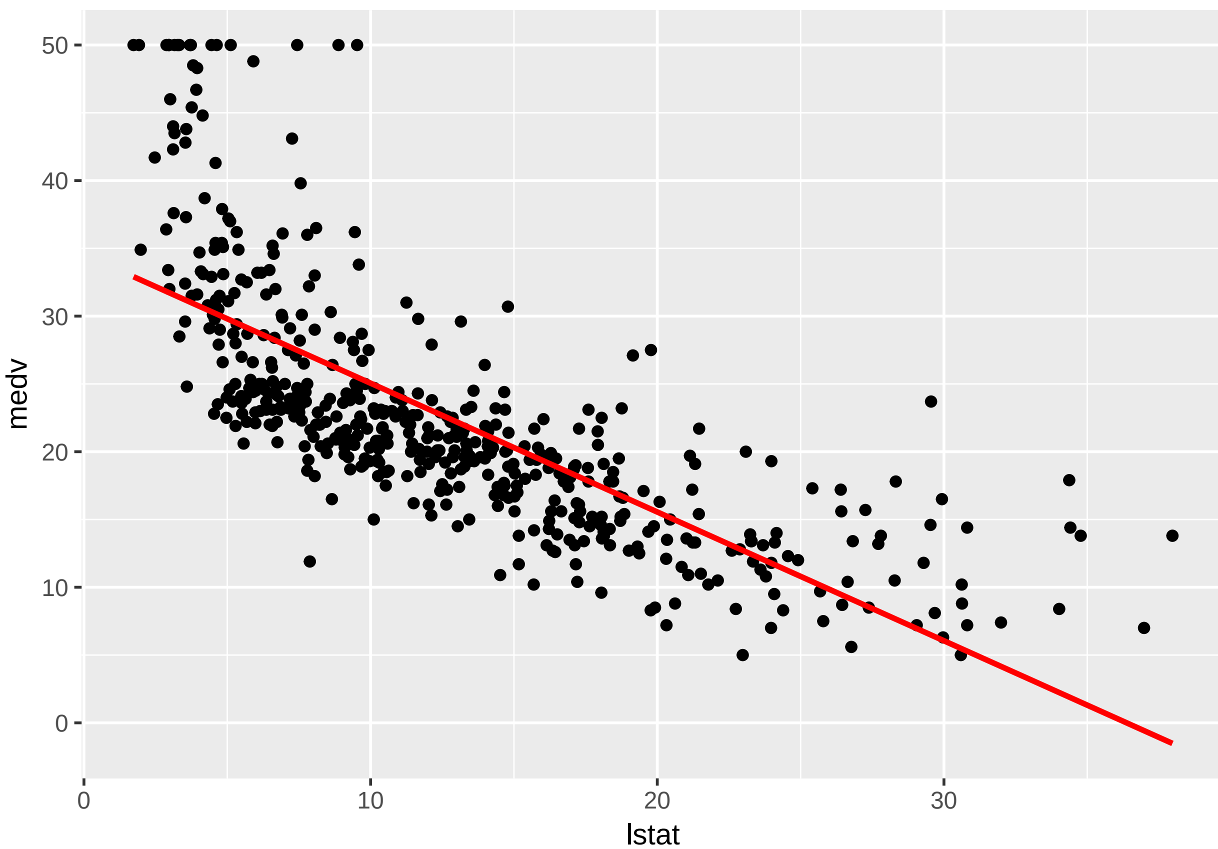
```r
# Samo koeficijenti
fit_1$coefficients
```

```
## (Intercept)       lstat
##  34.5538409  -0.9500494
```

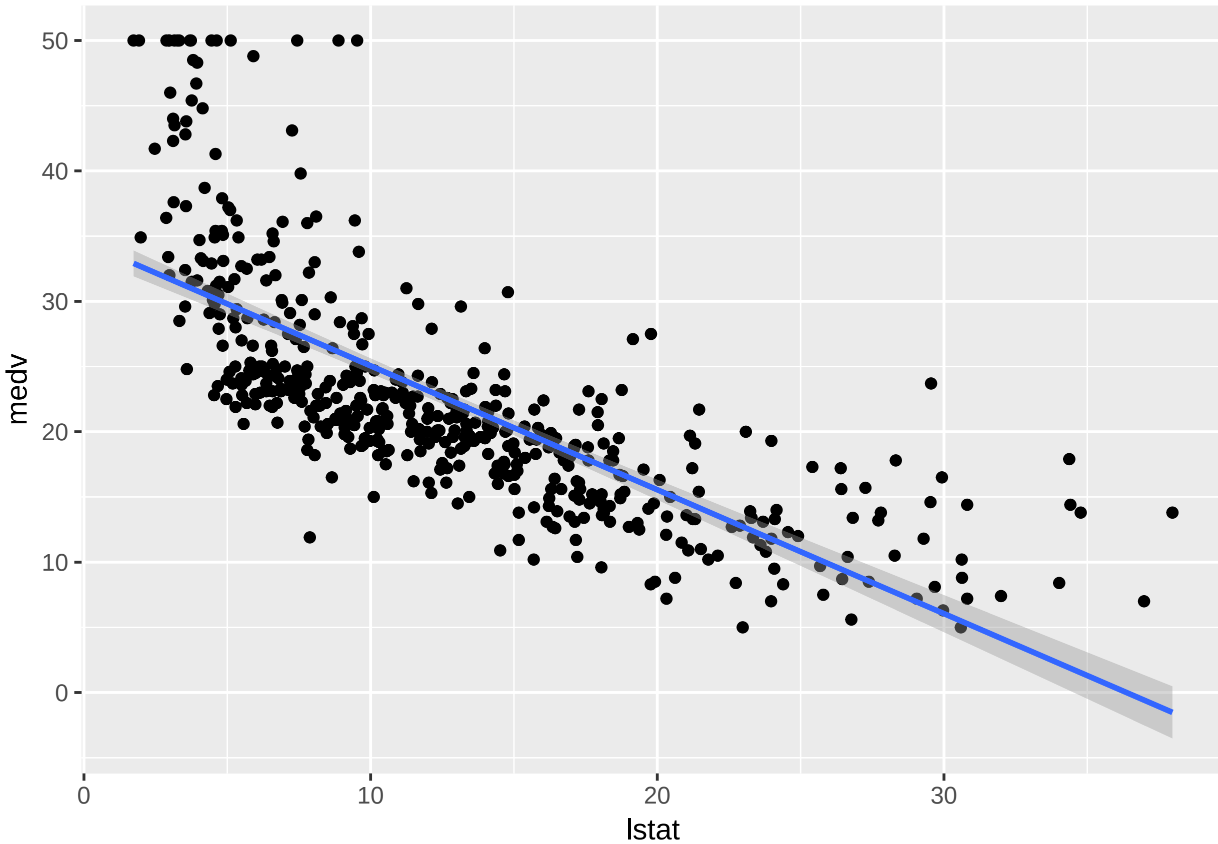
```r
# Ucrtajmo regresionu pravu na pocetni scatter plot
abline(fit_1$coefficients, col = "red", lwd = 2)
```




```r
# Ili sve zajedno koristerci "ggplot2"
ggplot(Boston, aes( x = lstat, y = medv)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, colour = "red")
```




```
# Ako zelimo i interval pouzdanosti sam izostavimo parametar "se" (podrazumevano se = TRUE)
ggplot(Boston, aes( x = lstat, y = medv)) +
    geom_point() +
    geom_smooth(method = "lm")
```




```r
# Interval pouzdanosti
confint(fit_1)
```

```
##                  2.5 %     97.5 %
## (Intercept) 33.448457 35.6592247
## lstat       -1.026148 -0.8739505
```

```r
# Da predvidimo vrednosti "medv" za dati vektor vrednosti "lstat" promenljive, uz
# proracun intervala pouzdanosti
predict(fit_1,data.frame(lstat = c(5,10,15)),interval = "confidence")
```

```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

Multivarijabilna linearna regresija

Primer 1

```r
library(readr)
library(tidyr)
library(purrr)
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 3.3.2
```

```r
# Uvoz i sredjivanje podataka
shop_data <- read_csv("data/shop_data.csv")
```

```
## Parsed with column specification:
## cols(
##   `"sales","sq_ft","inv","ads","size_dist","comp"` = col_character()
## )
```

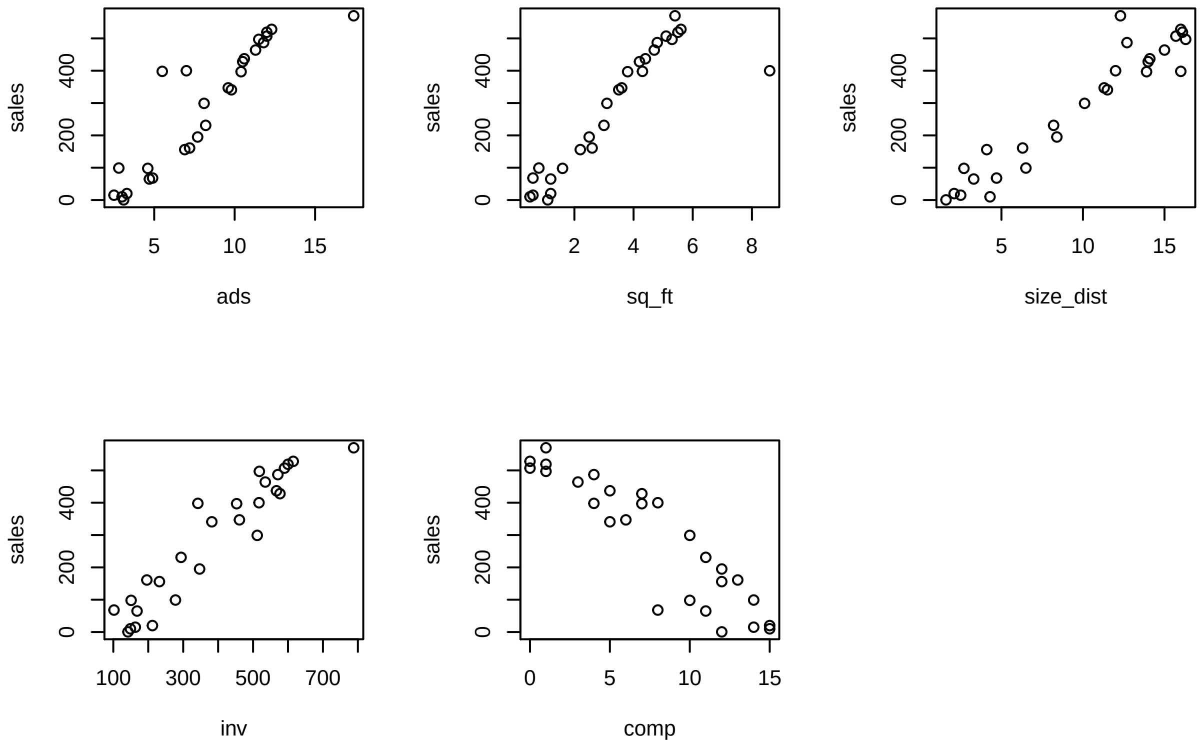
```r
shop_data <- separate(shop_data, '"sales","sq_ft","inv","ads","size_dist","comp"',
                      c("sales","sq_ft","inv","ads","size_dist","comp"), sep = ",")
shop_data <- as.data.frame(map(shop_data, as.numeric))

str(shop_data)
```

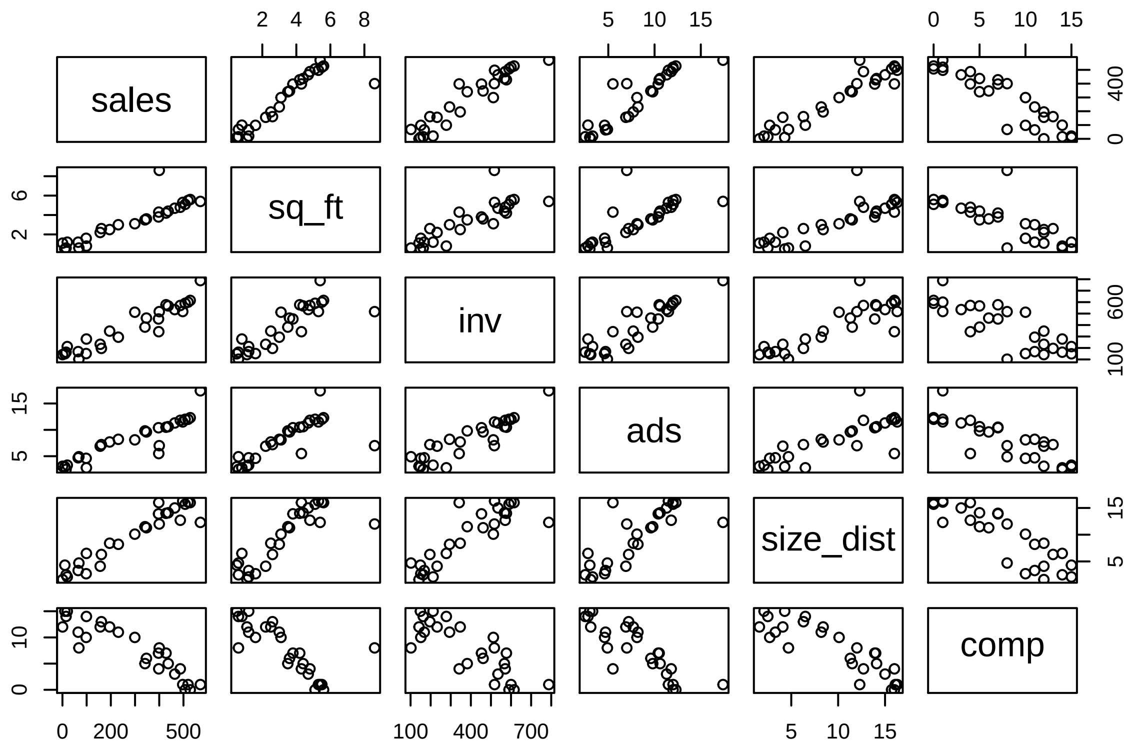
```
## 'data.frame':    27 obs. of  6 variables:
##  $ sales    : num  231 156 10 519 437 487 299 195 20 68 ...
##  $ sq_ft    : num  3 2.2 0.5 5.5 4.4 ...
##  $ inv      : num  294 232 149 600 567 571 512 347 212 102 ...
##  $ ads      : num  8.2 6.9 3 12 10.6 ...
##  $ size_dist: num  8.2 4.1 4.3 16.1 14.1 ...
##  $ comp     : num  11 12 15 1 5 4 10 12 15 8 ...
```

```r
head(shop_data)
```

```
##   sales sq_ft inv  ads size_dist comp
## 1   231   3.0 294  8.2       8.2   11
## 2   156   2.2 232  6.9       4.1   12
## 3    10   0.5 149  3.0       4.3   15
## 4   519   5.5 600 12.0      16.1    1
## 5   437   4.4 567 10.6      14.1    5
## 6   487   4.8 571 11.8      12.7    4
```

```r
# Hajde da proverimo kako se podaci ponasaju i mogu li se uociti relacije
# izmedju distribucija promenljivih koje bi ukazivale na opravdanost uvodjenja
# linearnog modela:

par(mfrow = c(2,3))
plot(sales ~ ads, shop_data)
plot(sales ~ sq_ft, shop_data)
plot(sales ~ size_dist, shop_data)
plot(sales ~ inv, shop_data)
plot(sales ~ comp, shop_data)

#Ili:
pairs(shop_data)
```


sales
0
200
400
5
10
15
ads
sales
0
200
400
2
4
6
8
sq_ft
sales
0
200
400
5
10
15
size_dist
sales
0
200
400
100
300
500
700
inv
sales
0
200
400
0
5
10
15
comp

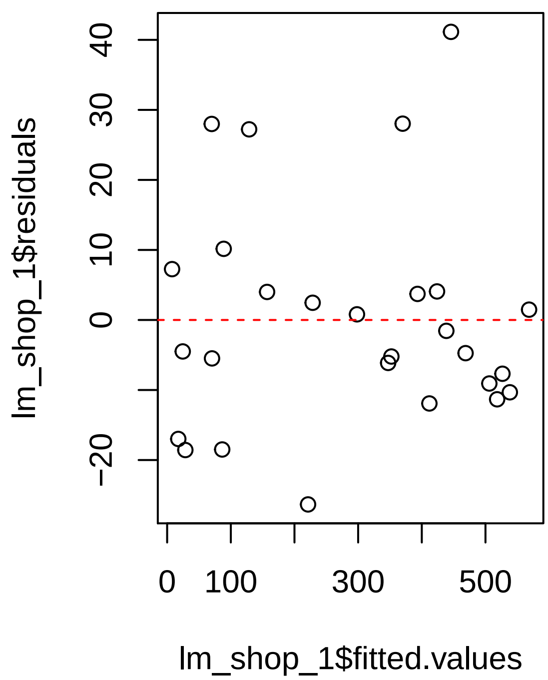


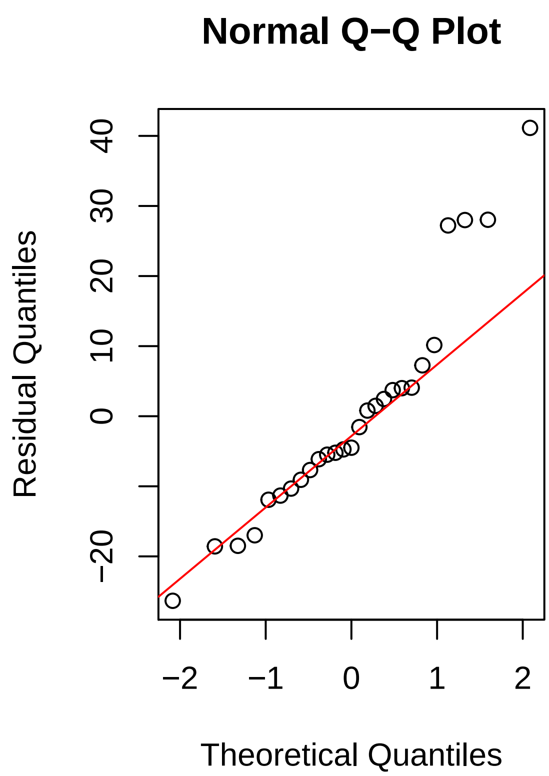

```r
# Linearni model za "sales" koji ukjucuje sve prediktore (sve preostale promenljive)
lm_shop_1 <- lm( sales ~., data = shop_data)

# Proverimo parametre valjanosti modela i koliko su pojedini prediktori znacajni u modelu:
summary(lm_shop_1)
```

```
##
## Call:
## lm(formula = sales ~ ., data = shop_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.338  -9.699  -4.496   4.040  41.139
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -18.85941   30.15023  -0.626 0.538372
## sq_ft        16.20157    3.54444   4.571 0.000166 ***
## inv           0.17464    0.05761   3.032 0.006347 **
## ads          11.52627    2.53210   4.552 0.000174 ***
## size_dist    13.58031    1.77046   7.671 1.61e-07 ***
## comp         -5.31097    1.70543  -3.114 0.005249 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.65 on 21 degrees of freedom
```

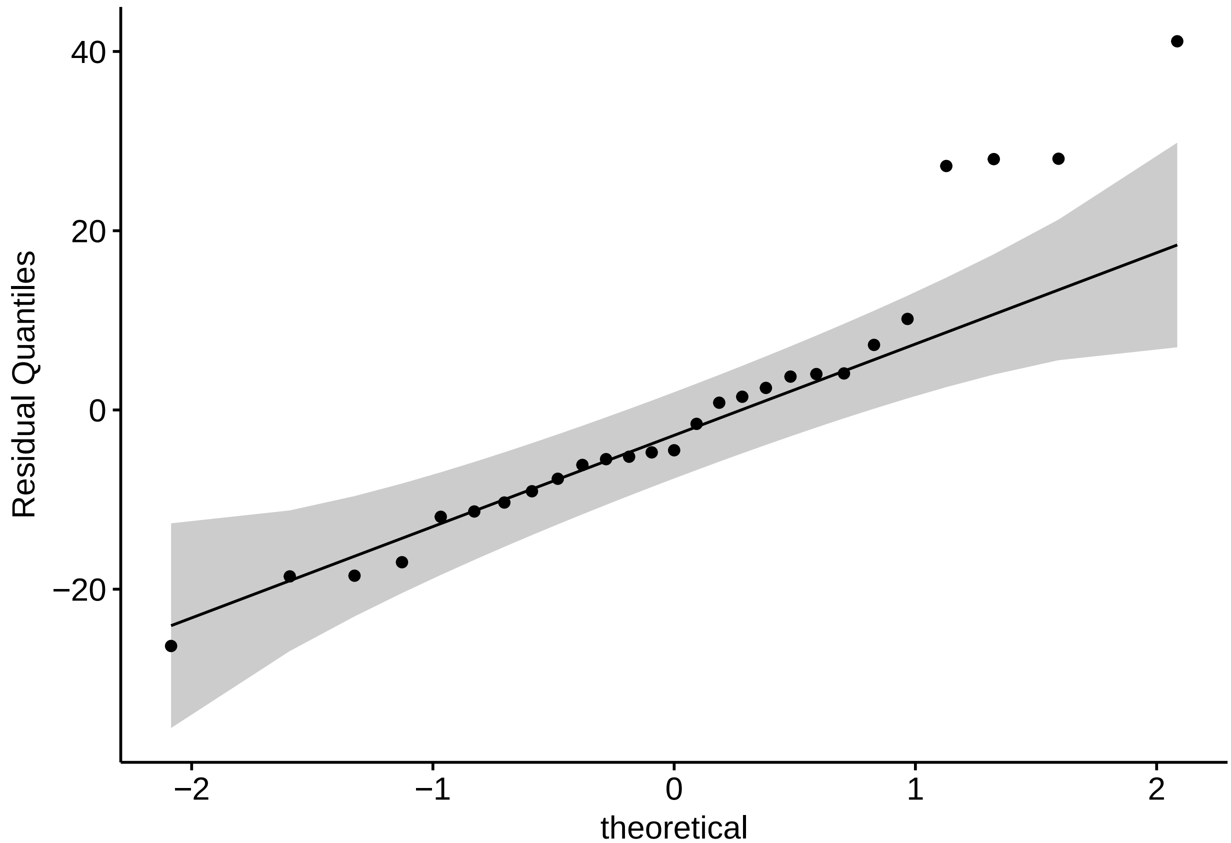
```
## Multiple R-squared:  0.9932, Adjusted R-squared:  0.9916
## F-statistic: 611.6 on 5 and 21 DF,  p-value: < 2.2e-16
```

```r
# Da bismo uopste mogli da koristimo p-vrednosti u ovom kontekstu treba prvo da proverimo
# da li je zadovoljena pretpostavka o normalnoj distribuciji reziduala!

par(mfrow = c(1,2))
# Plotujemo reziduale u funkciji fitovanih vrednosti za pojedinacje opservacije
plot(lm_shop_1$fitted.values, lm_shop_1$residuals)
abline(0,0, col = "red", lty = 2)

# Napravimo  Q-Q plot kvantila reziduala
qqnorm(lm_shop_1$residuals, ylab = "Residual Quantiles")
qqline(lm_shop_1$residuals, col = "red")
```






```r
par(mfrow = c(1,1))

# Mozemo i da upotrebimo f-ju "ggqqplot" iz paketa "ggpubr" koji sadrzi funkcije za
# plotovanje "lepih" grafika:

ggqqplot(lm_shop_1$residuals, ylab = "Residual Quantiles")
```




```r
# Me moze se uociti nikakav jasan "pattern" u distribucij reziduala, sta vise kvantili
# reziduala su uglavnom na liniji koja odgovara teorijskoj - normalnoj distribuciji

# Proverimo ponovo summary
summary(lm_shop_1)
```

```
##
## Call:
## lm(formula = sales ~ ., data = shop_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.338  -9.699  -4.496   4.040  41.139
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -18.85941   30.15023  -0.626 0.538372
## sq_ft        16.20157    3.54444   4.571 0.000166 ***
## inv           0.17464    0.05761   3.032 0.006347 **
## ads          11.52627    2.53210   4.552 0.000174 ***
## size_dist    13.58031    1.77046   7.671 1.61e-07 ***
## comp         -5.31097    1.70543  -3.114 0.005249 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.65 on 21 degrees of freedom
```

```
## Multiple R-squared:  0.9932, Adjusted R-squared:  0.9916
## F-statistic: 611.6 on 5 and 21 DF,  p-value: < 2.2e-16
```

```r
# Iskoristimo sada dobijeni model da predvidimo neto prodajnu vrednost na osnovu novog
# skupa prediktora:
shop_new = data.frame("sq_ft" = 2.3, "inv" = 420, "ads" = 8.7,
                      "size_dist" = 9.1, "comp" = 10)
predict(lm_shop_1, shop_new)
```

```
##        1
## 262.5006
```

Primer 2

Za ovaj primer cemo ponovo koristiti set podataka "Boston" iz paketa `MASS`.

```r
# Linearni model za "medv" na osnovu dva prediktora: "lstat" i "age"
fit2 = lm(medv ~ lstat + age, data = Boston)
summary(fit2)
```

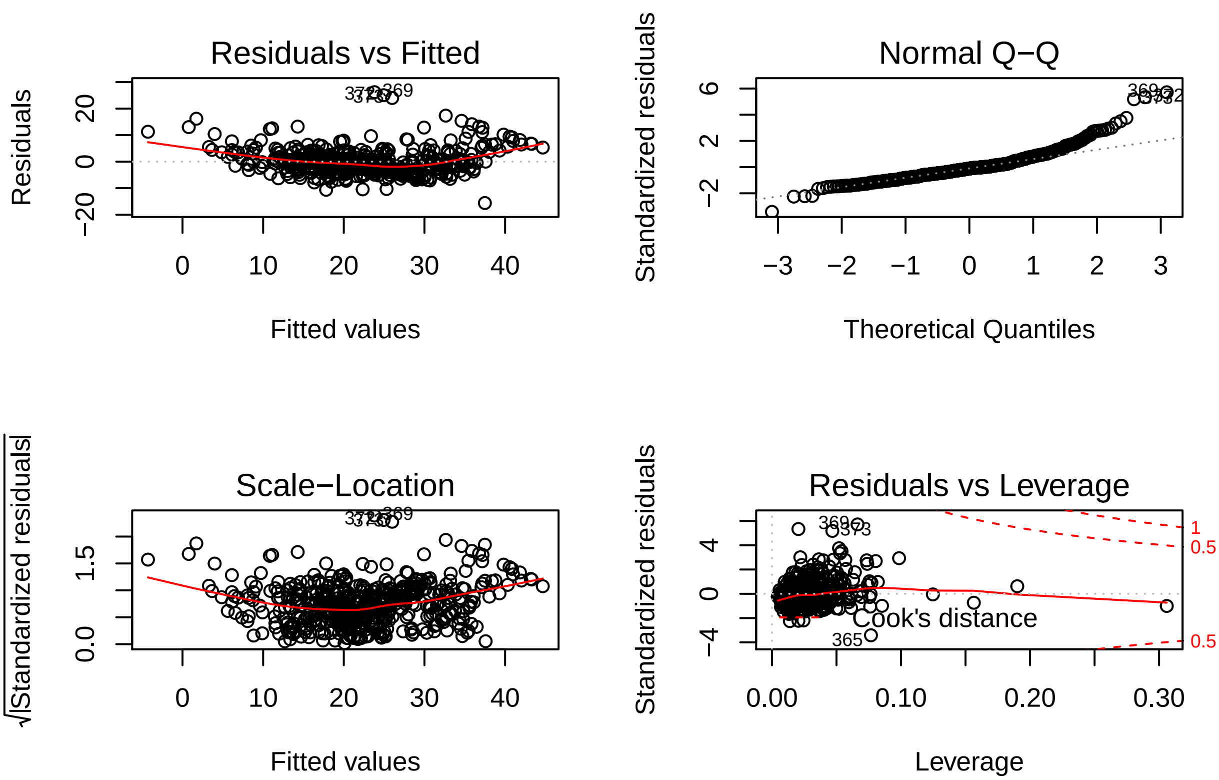
```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

```r
#Linearni model za "medv" na osnovu svih raspolozivih prediktora
fit3 = lm(medv ~.,Boston)
summary(fit3)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
```

```
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

```r
# Jos jedan nacin da se iscraju grafici koji se koriste za procenu valjanosti i opravdanosti linearnog
par(mfrow = c(2,2))
plot(fit3)
```




```r
# Na osnovu "summary" za model fit3 videli smo da promenljive "indus" i "age" ne igraju bitnu ulogu, te
fit4 = update(fit3,~.-age-indus)
summary(fit4)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad +
##     tax + ptratio + black + lstat, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.5984  -2.7386  -0.5046   1.7273  26.2373
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.341145   5.067492   7.171 2.73e-12 ***
## crim         -0.108413   0.032779  -3.307 0.001010 **
## zn            0.045845   0.013523   3.390 0.000754 ***
## chas          2.718716   0.854240   3.183 0.001551 **
## nox         -17.376023   3.535243  -4.915 1.21e-06 ***
## rm            3.801579   0.406316   9.356  < 2e-16 ***
## dis          -1.492711   0.185731  -8.037 6.84e-15 ***
## rad           0.299608   0.063402   4.726 3.00e-06 ***
## tax          -0.011778   0.003372  -3.493 0.000521 ***
## ptratio      -0.946525   0.129066  -7.334 9.24e-13 ***
## black         0.009291   0.002674   3.475 0.000557 ***
## lstat        -0.522553   0.047424 -11.019  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.736 on 494 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7348
## F-statistic: 128.2 on 11 and 494 DF,  p-value: < 2.2e-16
```

Nelinearna regresija

Polinomijalna regresija

- Ovaj segment ce biti dopunjen u buducnosti

KNN (K-Nearest Neighbors) u regresionoj analizi

- Ovaj segment ce biti dopunjen u buducnosti

Stablo odlucivanja (Decision Tree), Random Forest i Boosting za regresiju

Uvod

- Ovaj segment ce biti dopunjen u buducnosti

Klasifikacija

- Za razliku od regresionog modela, koji koristimo kako bismo predvideli neku vrednost iz kontiunualnog skupa brojnih vrednosti, klasifikacioni model odn. algoritam koristimo da predvidimo klasu odn., uslovno receno, vrednost iz nekog diskretnog skupa tj. kategoricku promenljivu.

- Ne primer: **objekat_na_slici** $\in$ **{"automobil", "bicikl", "avion"}** ili **email** $\in$ **{"spam", "ham"}**
- U ovom slucaju nas zadatak je da na osnovu vektora prediktora X i vektora odziva Y koji sadrzi kvalitativne tj. kategoricke vrednosti iz skupa C generisemo funkciju $C(X)$ (trening, odn. ucenje algoritma) koja kao ulaz prima novi (onaj koji nije koriscen u treningu) vektor prediktora X a kao odziv daje vrednosti (vrsi predvidjanje) za Y, $C(X) \in C$.
- Neretko nas interesuje procena verovatnoce da X pripada nekoj od kategorija iz C.

Logisticka regresija

Bice dodato naknadno.

Stablo odlucivanja - Decision Tree

- Metode ovog tipa pocivaju na principima stratifikacije i segmentacije prediktorskog hiperprostora u veci broj manjih, prostih, regiona.
- Istorijski gledano prva dva algoritma ovog tipa su:
  - **CART** (Classification And Regression Tree), Leo Breiman et al.
  - **ID3** (Iterative Dichotomiser 3), Ross Quinlan
- Stablo odlucivanja je algoritam jednostavan za upotrebu i interpretaciju rezultata, medjutim sklon je overfit-ovanju i uglavnom ne postize tacnost uporedivu sa nekim modernijim algoritmima.
- Metode kao sto su *bagging, random forest i boosting* koje se baziraju na iterativnoj agregaciji pojedinacnih stabala odlucivanja obezbedjuju znacajno bolje performanse ali po cenu interpretabilnosti rezultata.
- Objasnicemo primenu stabla odlucivanja za klasifikaciju na primeru, koristeci dobro poznati `titanic` set podataka koji smo, prethodno, vec uvezli i adekvatno modifikovali za potrebe nase analize i modelovanja.
- DataCamp-ov tutorial: Kaggle R Tutorial on Machine Learning

```r
# Ucitajmo prvo pakete koji ce biti korisceni u ovom primeru
library(rpart)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.3.2
```

```r
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.3.2
```

```r
# Da se podsetimo kako izgleda skup podataka koji cemo koristiti u ovom
# primeru:
str(titanic)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    891 obs. of  4 variables:
##  $ survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ pclass  : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ sex     : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ age     : num  22 38 26 35 35 NA 54 2 27 14 ...
```

```r
# Odnos broja prezivelih i poginulih: "0" - nije preziveo/la; "1" - preziveo/la
table(titanic$survived)
```

```
##
##   0   1
## 549 342
```

```r
prop.table(table(titanic$survived))
```

```
##
##         0         1
## 0.6161616 0.3838384
```

```r
# Da obezbedimo reproduktibilnost
set.seed(333)

# Za potrebe ovog primera cemo na osnovu "titanic" data set-a formirati
# "train" i "test" set podataka koje cemo, respektivno, koristiti za trening naseg
# klasifikacionog algoritma (Decision Tree) i njegovo testiranje.
# Podelu cemo izvrsiti tako da "train" set sadrzi 75% podataka,
# a "test" preostalih 25%.
train_ind <- sample(1:nrow(titanic), round(0.75*(nrow(titanic))))
train <- titanic[train_ind, ]
test <- titanic[-train_ind, ]

# Proverimo da li "train" i "test" data set izgledaju kako bi trebalo
str(train)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    668 obs. of  4 variables:
##  $ survived: Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 2 2 2 ...
##  $ pclass  : Factor w/ 3 levels "1","2","3": 2 3 2 1 2 1 1 3 2 1 ...
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 1 2 2 1 1 2 1 1 ...
##  $ age     : num  34 25 42 NA NA 24 22 25 21 35 ...
```

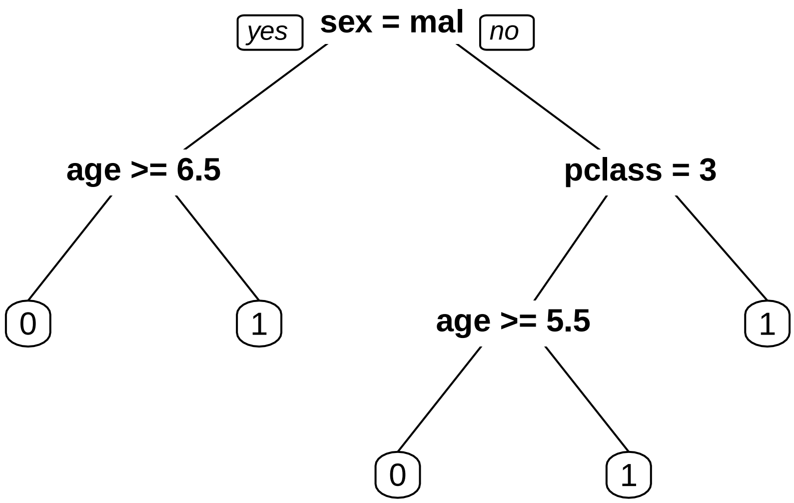
```r
str(test)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    223 obs. of  4 variables:
##  $ survived: Factor w/ 2 levels "0","1": 1 2 2 1 2 1 1 2 1 1 ...
##  $ pclass  : Factor w/ 3 levels "1","2","3": 3 1 3 3 3 3 3 3 3 3 ...
##  $ sex     : Factor w/ 2 levels "female","male": 2 1 1 2 1 2 2 1 2 2 ...
##  $ age     : num  22 38 26 2 27 20 39 NA NA NA ...
```

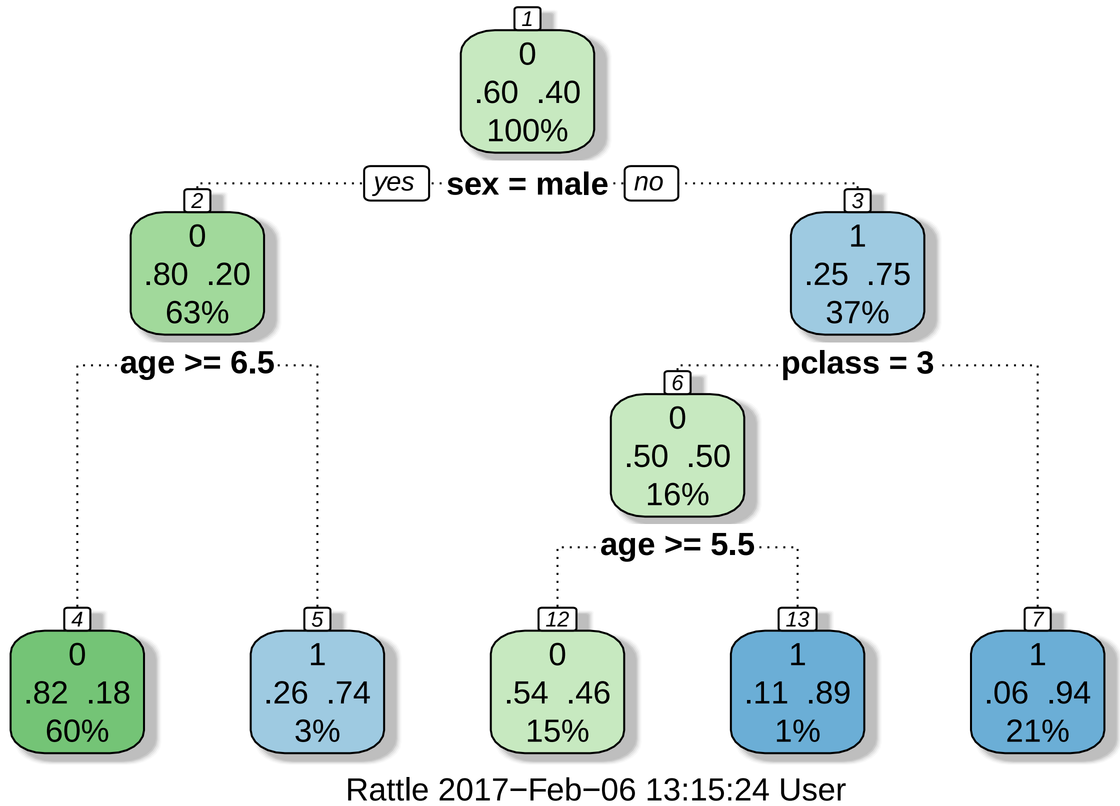
```r
# Proverimo da li je zastupljenost klasa u trening setu dobro reprezentovana, tj.
# da li se poklapa onom u polaznom setu podataka ("titanic").
prop.table(table(train$survived))
```

```
##
##         0         1
## 0.5958084 0.4041916
```

```r
# Generisemo klasifikacioni model (drvo odlucivanja - decision tree) na osnovu datih podataka, tj. vrsi
tree <- rpart(survived ~ ., data = train, method = "class")

# Hajde da vidimo kako nase stablo odlucivanja izgleda
prp(tree)
```




```r
# Korisniji i lepsi dijagram
fancyRpartPlot(tree)
```




```r
# Koristimo predict() funkciju da predvidimo klase na osnovu podataka iz
# "test" data set-a
pred <- predict(tree, newdata = test, type = "class")

# Provera tacnosti predvidjanaja na "test" setu
# Konstruisemo konfuzionu matricu: conf
conf <- table(test$survived, pred)

# Racunamo tacnost
sum(diag(conf))/sum(conf)
```

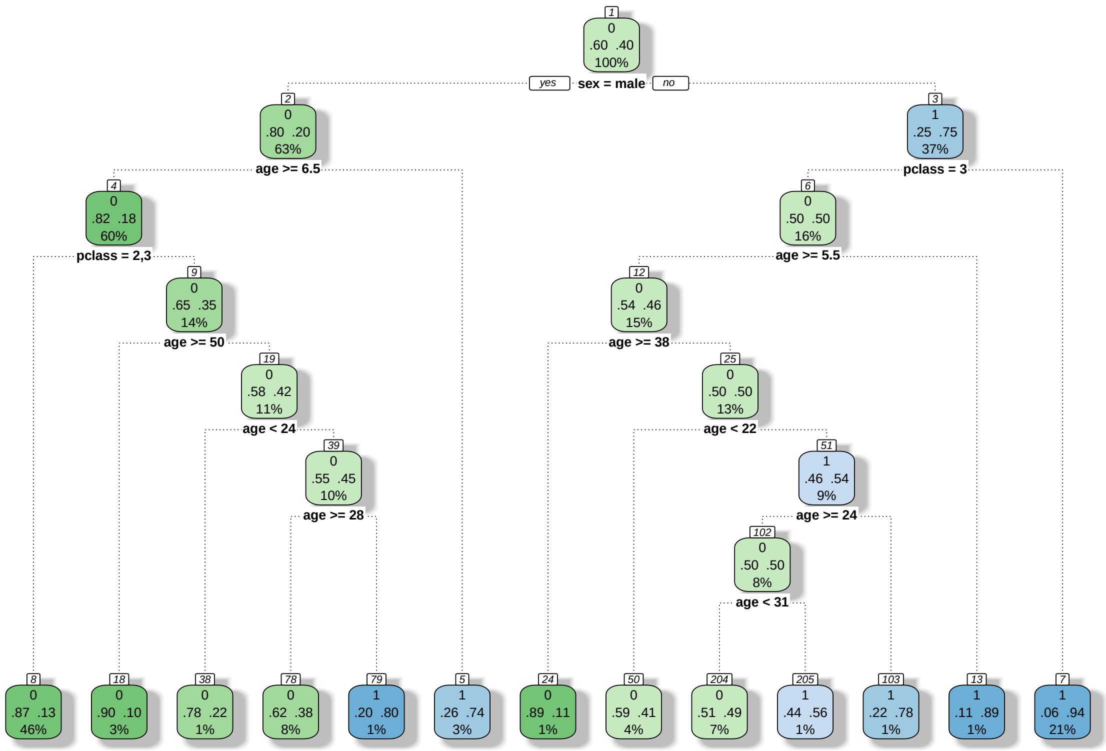
```
## [1] 0.8071749
```

Kao sto se moze videti, procenjena tacnost ovako dobijenog modela, na test setu, je nesto preko 80%, sto se, u ovom konkretnom slucaju, smatra solidnim rezultatom.

Demonstracije radi, u nastavku cemo pokazati kako se kontrolise kompleksnost modela podesavanjem parametra `cp`. Uprosceno receno, ovaj parametar odredjuje koji ce cvorovi (nodes) biti uklonjeni iz finalnog modela, jer ne doprinose, u dovoljnoj meri, razdvajanju klasa. Generalno govoreci, podrazumevano ponasanje `rpart` algoritma je u prilicnoj meri optimalno u pogledu odabira modela odgovarajuce kompleksnosti. Prilikom formiranja modela treba imati na umu da su kompleksniji modeli skloniji overfit-ovanju podataka! Ovo naravno ne znaci da prostiji model obavezno obezbedjuje bolju generalizaciju, tj. bolju klasifikaciju (sa vecom tacnoscu) podataka koji nisu korisceni tokom treninga.

```r
#Prvo cemo generisati nepotrebno kompleksan model
complex_tree <- rpart(survived ~ ., train, method = "class",
                      control = rpart.control(cp = 0.00001))
```

```r
fancyRpartPlot(complex_tree)
```



Rattle 2017−Feb−06 13:15:24 User

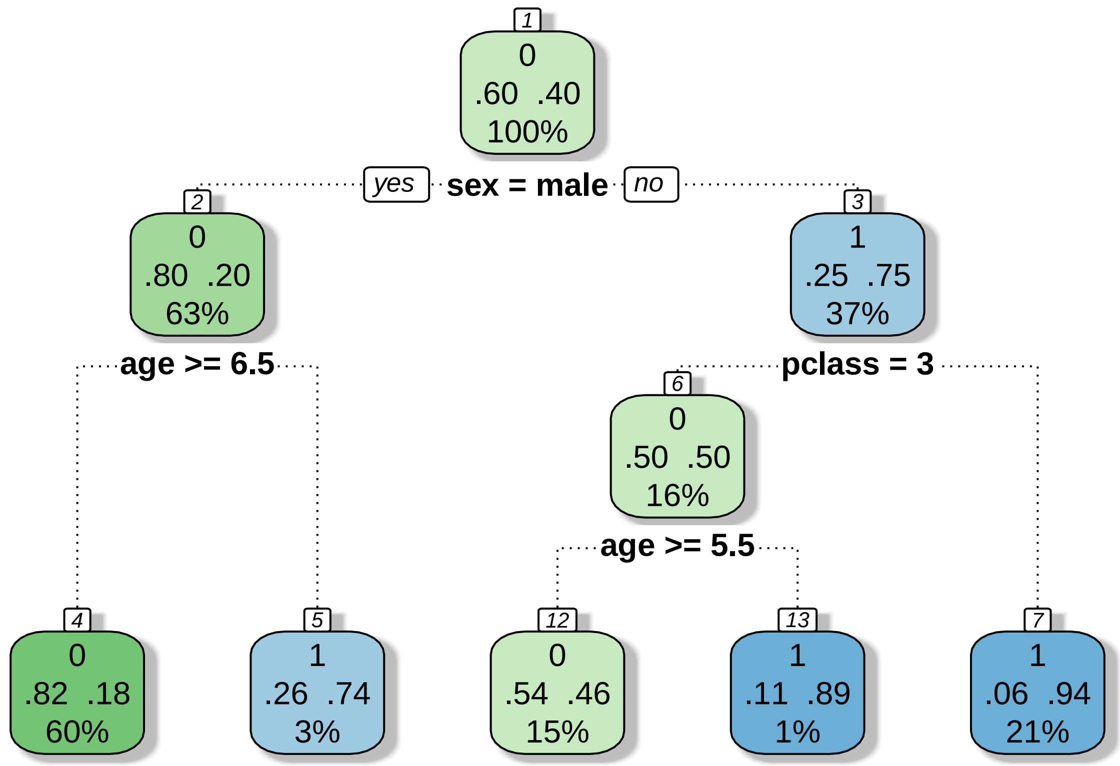
```r
# Provera tacnosti predvidjanja na "trening" setu
# Prvo vrsimo estimaciju klasa u training setu na osnovu modela "complex_tree"
pred <- predict(complex_tree, train, type = "class")


# Konstruisemo konfuzionu matricu: conf
conf <- table(train$survived, pred)
# Racunamo tacnost
sum(diag(conf))/sum(conf)
```

```
## [1] 0.8203593
```

```r
# Proverimo tacnost ovako dobijenog modela na test setu.
# Prvo vrsimo estimaciju klasa u test setu na osnovu modela "complex_tree"
pred <- predict(complex_tree, newdata = test, type = "class")

# Provera tacnosti predvidjanaja na "test" setu
# Konstruisemo konfuzionu matricu: conf
conf <- table(test$survived, pred)

# Racunamo tacnost
sum(diag(conf))/sum(conf)
```

```
## [1] 0.8026906
```

```r
# Sada cemo ovom slozenum drvetu "skresati" grane
pruned <- prune(complex_tree, cp = 0.01)
fancyRpartPlot(pruned)
```



Rattle 2017−Feb−06 13:15:24 User

```r
# Proverimo sada za model "pruned" kako stojimo sa tacnoscu
# Provera tacnosti predvidjanja na "trening" setu
# Prvo vrsimo estimaciju klasa u training setu na osnovu modela "pruned"
pred <- predict(pruned, train, type = "class")


# Konstruisemo konfuzionu matricu: conf
conf <- table(train$survived, pred)
# Racunamo tacnost
sum(diag(conf))/sum(conf)
```

```
## [1] 0.8023952
```

```r
# Proverimo tacnost ovako dobijenog modela na test setu.

# Prvo vrsimo estimaciju klasa u test setu na osnovu modela "pruned"
pred <- predict(pruned, newdata = test, type = "class")

# Provera tacnosti predvidjanaja na "test" setu
# Konstruisemo konfuzionu matricu: conf
conf <- table(test$survived, pred)

# Racunamo tacnost
```

```r
sum(diag(conf))/sum(conf)
```

```
## [1] 0.8071749
```

Kao sto se moze videti, na osnovu rezultata, prostiji model obezbedjuje bolju generalizaciju. Na ovo ukazuje manja razlika u procenjenoj tacnosti klasifikacije na *test* i *trening* setu podataka.

Jos o proceni valjanosti modela za binarnu klasifikaciju: ROC (Receiver Operator Characteristic) krive

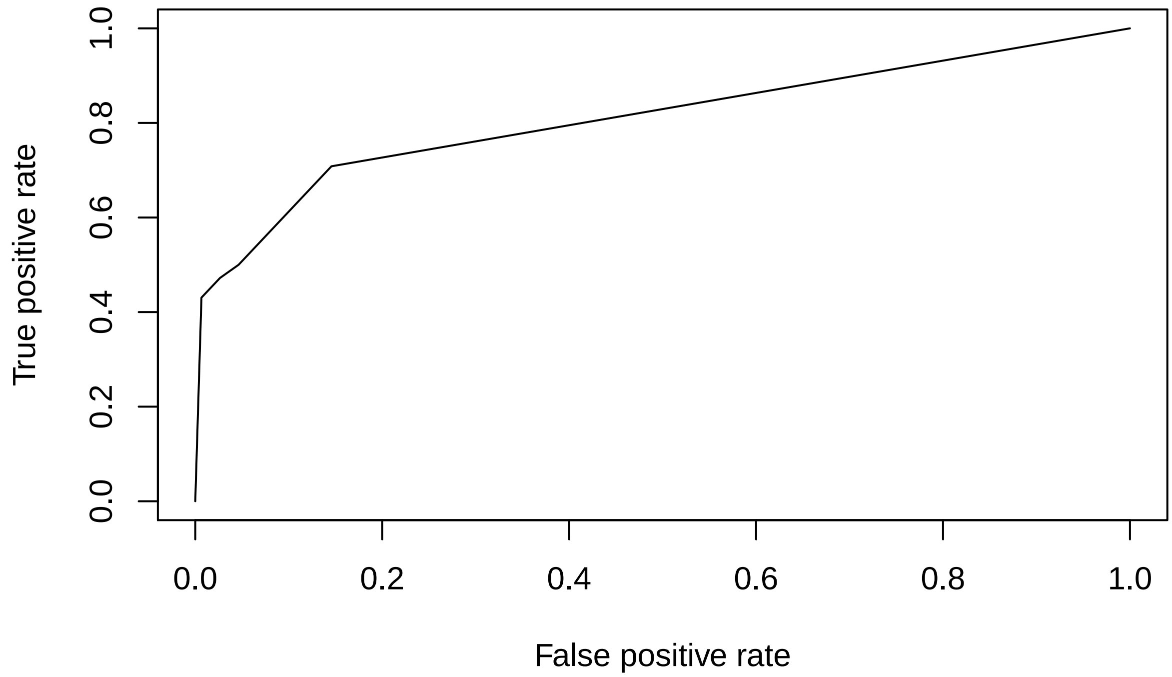
- Izuzetno mocan alat za procenu perfomansi binarnog klasifikatora
- Receiver Operator Characteristic Curve - ROCC
- Graficki prikaz promene odnosa True Positive Rate (Recall) vs False Positive Rate, sa promenom vrednosnog praga (treshold) za verovatnocu na osnovu koga se odredjuje pripadanost jednoj od dve klase.
- AUC (Area Under the Curve) > 0.9 - dobar klasifikator
- R paketi koji se mogu koristiti za ROC analizu: "ROCR" i "pROC"

Primer

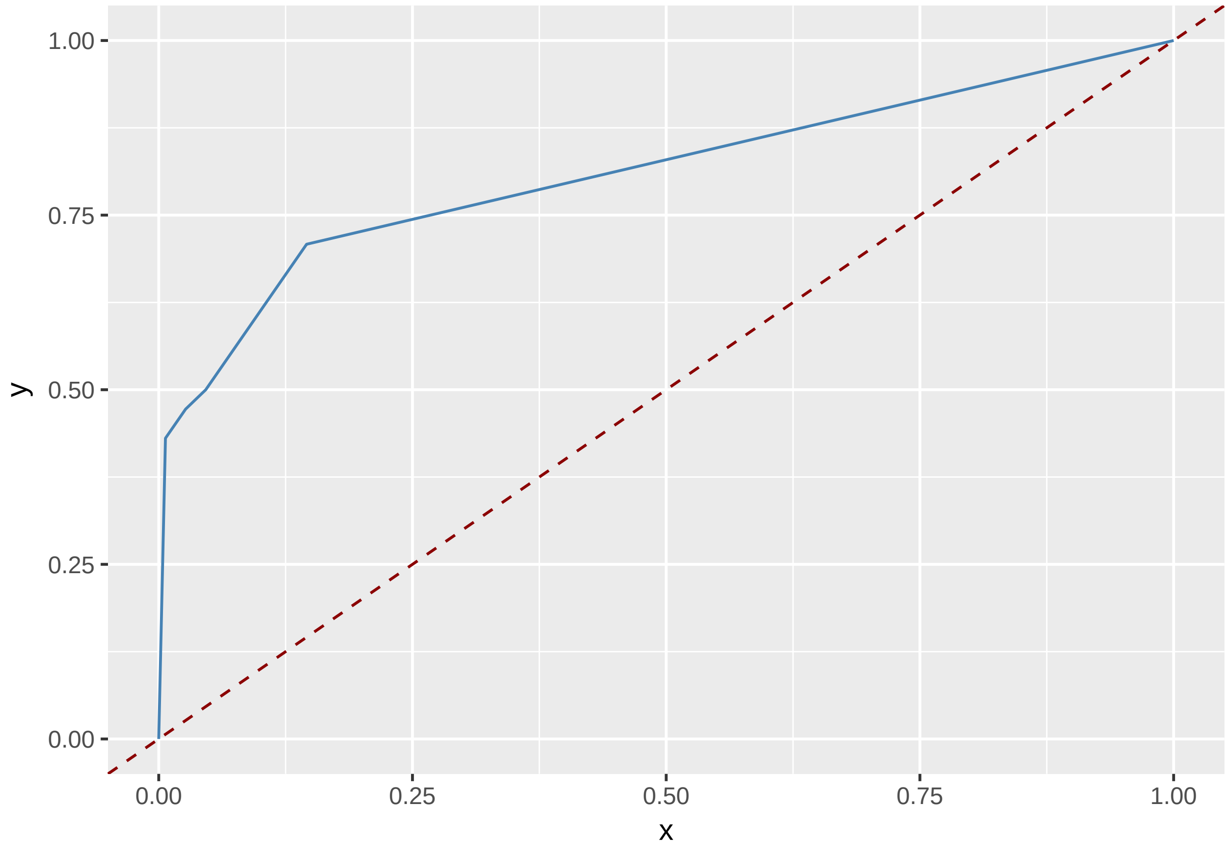
```r
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.3.2

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.3.2

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
# Generisanje ROC krive nam trebaju verovatnoce - type = "prob"
probs <- predict(tree, test, type = "prob")[,2]

# Generisemo "prediction" objeka: pred
pred <- prediction(probs, test$survived)

# Generisemo "performance" objekat: perf
perf <- performance(pred, "tpr", "fpr")

# Crtamo ROC krivu
plot(perf)
```




```
#Ili
df <- data.frame(x = perf@x.values[[1]], y = perf@y.values[[1]])
ggplot(df, aes(x,y)) +
  geom_line(color = "steelblue") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "darkred")
```




```r
# Hajde da vidimo kolika je tacno povrsina ispod krive - AUC
# Ponovo generisemo adekvatan "performance"" objekat: perf
perf <- performance(pred, "auc")

# Ispisujemo vrednost za AUC
perf@y.values[[1]]
```

```
## [1] 0.8097866
```

Random Forest

Krajnje uprosceno, **Random Forest** algoritam radi po sledecem principu: generise se veliki broj razgranatih stabala odlucivanja, koja se zatim "uprosecuju" kako bi se smanjila varijansa i izbeglo over fit-ovanje.

Primer 1

```r
library(randomForest)
```

```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
```

```
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
# Da obezbedimo reproduktibilnost
set.seed(333)

# Za potrebe ovog primera cemo na osnovu "titanic" data set-a formirati
# "train" i "test" set podataka koje cemo, respektivno, koristiti za trening naseg
# klasifikacionog algoritma (Decision Tree) i njegovo testiranje.
# Podelu cemo izvrsiti tako da "train" set sadrzi 75% podataka,
# a "test" preostalih 25%.
train_ind <- sample(1:nrow(titanic), round(0.75*(nrow(titanic))))
train <- titanic[train_ind, ]
test <- titanic[-train_ind, ]

# Proverimo da li "train" i "test" data set izgledaju kako bi trebalo
str(train)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    668 obs. of  4 variables:
##  $ survived: Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 2 2 2 ...
##  $ pclass  : Factor w/ 3 levels "1","2","3": 2 3 2 1 2 1 1 3 2 1 ...
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 1 2 2 1 1 2 1 1 ...
##  $ age     : num  34 25 42 NA NA 24 22 25 21 35 ...
```

```r
str(test)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    223 obs. of  4 variables:
##  $ survived: Factor w/ 2 levels "0","1": 1 2 2 1 2 1 1 2 1 1 ...
##  $ pclass  : Factor w/ 3 levels "1","2","3": 3 1 3 3 3 3 3 3 3 3 ...
##  $ sex     : Factor w/ 2 levels "female","male": 2 1 1 2 1 2 2 1 2 2 ...
##  $ age     : num  22 38 26 2 27 20 39 NA NA NA ...
```

```r
# Proverimo da li "train" i "test" sadrze NA vrednosti. Ako ih ima one se moraju ili
# imutirati na adekvatan nacin, ili se opservacije koje ih sadrze brisu, pre primene
# "randomForest" funkcije.

# Zamena NA vrednosti odgovarajucim brojnim vrdnostima upotrebom "rfImpute" funkcije.
titanic_imputed <- rfImpute(survived ~ ., train)
```

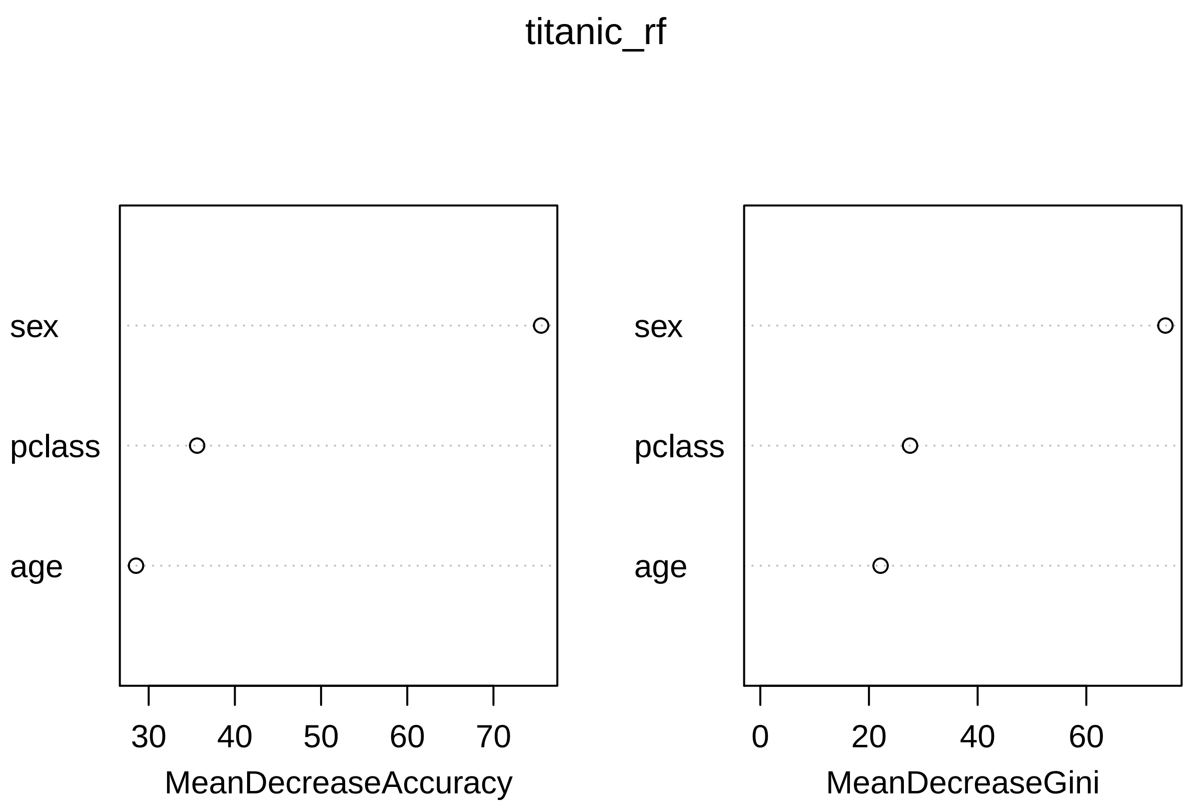
```
## ntree      OOB      1      2
##   300:  22.01% 12.31% 36.30%
## ntree      OOB      1      2
##   300:  21.56% 10.80% 37.41%
## ntree      OOB      1      2
##   300:  20.96% 12.56% 33.33%
## ntree      OOB      1      2
##   300:  21.71% 11.81% 36.30%
## ntree      OOB      1      2
##   300:  21.71% 13.57% 33.70%
```

```r
# "Treniranje" RF modela
titanic_rf <- randomForest(survived ~ ., titanic_imputed, importance = TRUE, ntree = 1000)

# Osnovni podaci o modelu
```

```r
titanic_rf
```

```
##
## Call:
##  randomForest(formula = survived ~ ., data = titanic_imputed,      importance = TRUE, ntree = 1000)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 21.56%
## Confusion matrix:
##     0   1 class.error
## 0 347  51   0.1281407
## 1  93 177   0.3444444
```

```r
# Vaznost pojedinacnih prediktora u izgradnju modela
varImpPlot(titanic_rf)
```




```r
# Proverimo sada za model "titanic_rf" kako stojimo sa tacnoscu
# Provera tacnosti predvidjanja na "trening" setu
# Prvo vrsimo estimaciju klasa u training setu na osnovu modela "titanic_rf"
pred <- predict(titanic_rf, train, type = "class")

# Konstruisemo konfuzionu matricu: conf
conf <- table(train$survived, pred)
# Racunamo tacnost
sum(diag(conf))/sum(conf)
```

```
## [1] 0.8144712
```

```r
# Proverimo tacnost ovako dobijenog modela na test setu.

# Prvo vrsimo estimaciju klasa u test setu na osnovu modela "titanic_rf"
pred <- predict(titanic_rf, newdata = test, type = "class")

# Provera tacnosti predvidjanaja na "test" setu
# Konstruisemo konfuzionu matricu: conf
conf <- table(test$survived, pred)

# Racunamo tacnost
sum(diag(conf))/sum(conf)
```

```
## [1] 0.8057143
```

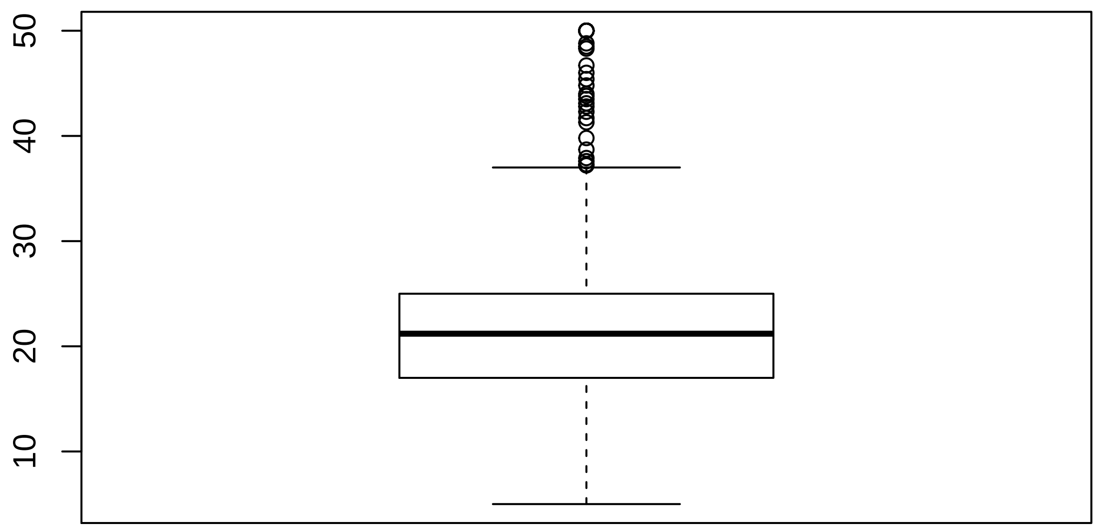
Primer 2

Za ovaj primer cemo koristiti vec posnati set podataka "Boston" iz `MASS` paketa. Ovaj set podataka sadrzi podatke o trzisnoj vrednosti nekretnina u predgradjima Bostona, SAD, zajedno sa razlicitim parametrima koji uticu na formiranje ove vrednosti. Sa obzirom da nas interesuje problem klasifikacije za pocetak cemo prevesti numericku promenljivu "medv" u kategoricku promenljivu sa tri nivoa koje cemo oznaciti sa "cheap", "average", "expensive". Dakle za razliku od prethodnog slucaja binarne klasifikacije (dve klase) ovde cemo imati tri klase.

```r
summary(Boston)
```

```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm             age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

```r
boxplot(Boston$medv)
```



```r
# Prevodimo "medv" u kategoricku promenljivu sa tri nivoa koje cemo oznaciti sa
# "cheap", "average", "expensive".
Boston$medv <- cut(Boston$medv, c(0,18,35,50), labels = c("cheap", "average", "expensive"))

summary(Boston$medv)
```

```
##     cheap   average expensive
##       149       309        48
```

```r
train <- sample(1:nrow(Boston), 300)

rf_boston <-  randomForest( medv ~., data = Boston, subset = train, importance = TRUE)
rf_boston
```

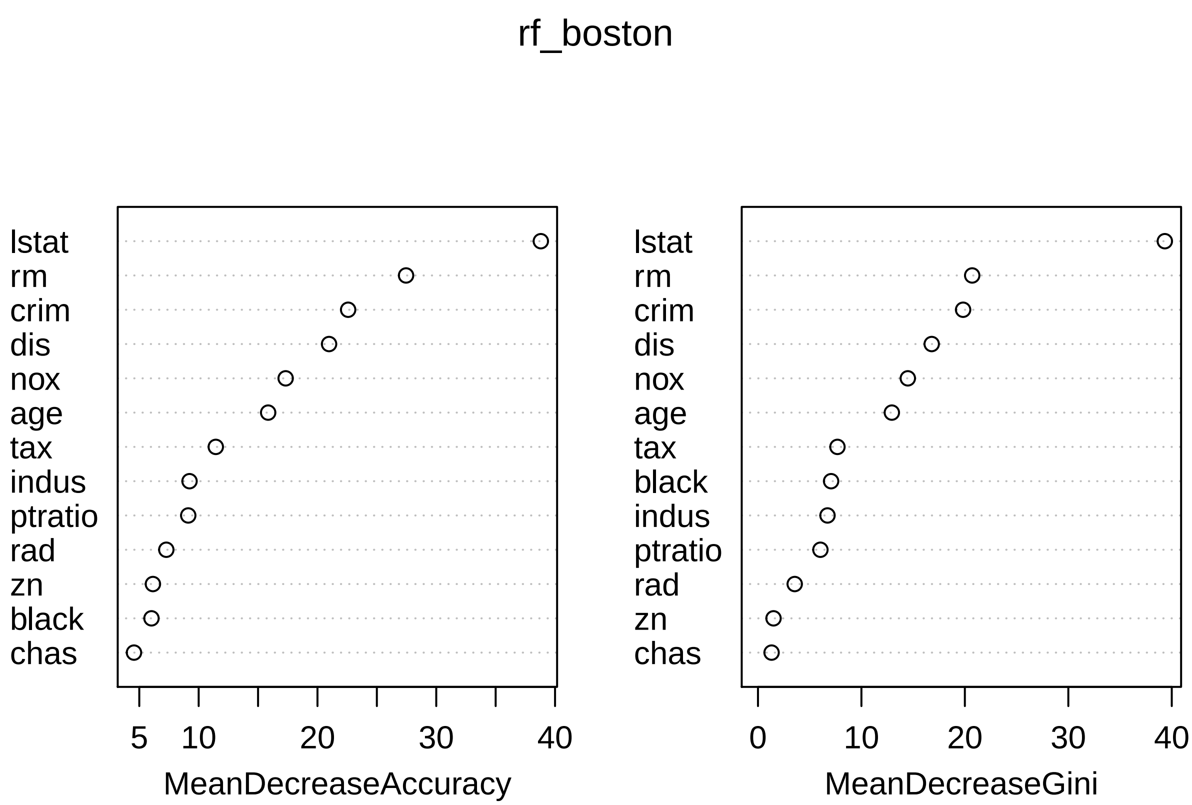
```
##
## Call:
##  randomForest(formula = medv ~ ., data = Boston, importance = TRUE,      subset = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 12.67%
## Confusion matrix:
##           cheap average expensive class.error
## cheap        76      14         0  0.15555556
```

```
## average        8     170         4  0.06593407
## expensive      0      12        16  0.42857143
```

```r
# Koliko je koja od promenljivih bitna za tacnost klasifikacije?
importance(rf_boston)
```

```
##               cheap    average  expensive MeanDecreaseAccuracy
## crim    18.191719 13.1441088  4.9825077            22.582272
## zn       1.406847  6.8792578 -0.6039329             6.148881
## indus    7.060638  3.1271680  4.8373658             9.229572
## chas     4.176063  0.7552713  3.5676006             4.554523
## nox     14.607432  6.9802331  7.1724002            17.323410
## rm       6.080570 17.1029218 30.4540344            27.457536
## age     13.720684  7.3773416  4.5181856            15.849805
## dis     13.199411 13.7297646  5.9654803            20.976262
## rad      6.252519  3.2819572  0.6593230             7.275247
## tax      7.996489  5.8452043  7.3337809            11.441162
## ptratio  8.003876  3.7401993  3.2647282             9.122871
## black    5.419059  3.0864593  3.1949336             6.027324
## lstat   38.214008 16.9906971 18.9829450            38.802556
##         MeanDecreaseGini
## crim           19.827327
## zn              1.502342
## indus           6.720518
## chas            1.314849
## nox            14.484855
## rm             20.707431
## age            12.939507
## dis            16.793916
## rad             3.551704
## tax             7.682348
## ptratio         6.033365
## black           7.067260
## lstat          39.329312
```

```r
varImpPlot(rf_boston)
```



Boosting

Ponovo, krajnje uprosceno, **Boosting** algoritam generise veliki broj malih (plitkih) stabala odlucivanja koja se inkrementalno pridodaju u cilju povecanja tacnosti odn. boljeg razdvajanja klasa.

U primeru koji sledi cemo koristiti paket **`gbm`** (Generalized Boosted Regression Models) i set podataka "Boston" koji je vec modifikovan na odgovarajuci nacin, za resavanje problema klasifikacije, u prethodnom primeru koji se odnosio na primenu "Random Forest" algoritma.

Primer

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.3.2

## Loading required package: survival

## Warning: package 'survival' was built under R version 3.3.2

##
## Attaching package: 'survival'

## The following object is masked from 'package:faraway':
##
##     rats

## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'

## The following object is masked from 'package:faraway':
##
##     melanoma

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1
```

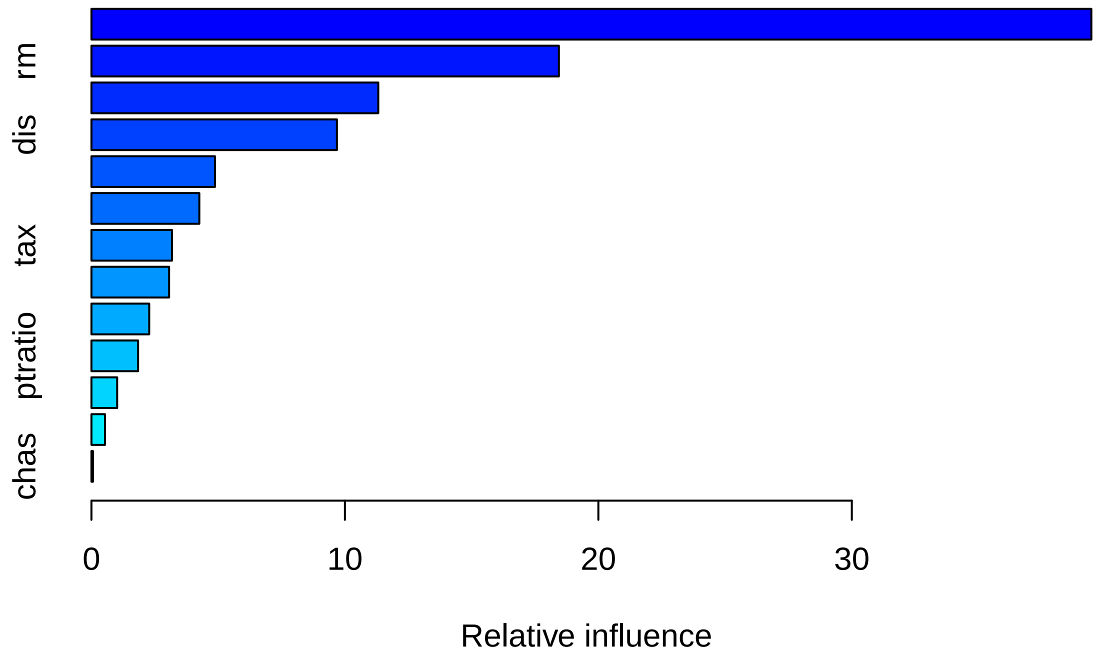
```
# Trening modela
boost_boston <- gbm(medv ~ .,
                    data = Boston[train,],
                    n.trees = 5000,
                    shrinkage = 0.01,
                    interaction.depth = 3)
```

```
## Distribution not specified, assuming multinomial ...
```

```
boost_boston
```

```
## gbm(formula = medv ~ ., data = Boston[train, ], n.trees = 5000,
##     interaction.depth = 3, shrinkage = 0.01)
## A gradient boosted model with multinomial loss function.
## 5000 iterations were performed.
## There were 13 predictors of which 13 had non-zero influence.
```

```
# Koliko je koja od promenljivih bitna za tacnost klasifikacije?
summary(boost_boston)
```




```
##             var     rel.inf
## lstat     lstat 39.45054445
## rm           rm 18.44259972
## crim       crim 11.31706304
## dis         dis  9.68494230
## age         age  4.87357251
## nox         nox  4.25663733
## tax         tax  3.17790865
## black     black  3.06514656
## indus     indus  2.27970086
## ptratio ptratio  1.84086177
## rad         rad  1.01479104
## zn           zn  0.53625481
## chas       chas  0.05997698
```

KNN (K-Nearest Neighbors) za klasifikaciju

Bice dodato naknadno.