

INTELIGENTNI SISTEMI - SVI ALGORITMI

1. LINEARNA REGRESIJA

```
library(MASS)
str(Boston)
summary(Boston)
?Boston
```

1. Prvo pravimo korelacionu matricu da bismo videli zavisnosti svih varijabla u odnosu na izlaznu promenljivu

```
cor.mat <- cor(Boston)
library(corrplot)
corrplot.mixed(cor.mat, tl.cex = 0.75, number.cex = 0.75)
```

Kao znacajne varijable u odnosu na varijablu medv posmatracemo sve one koje imaju stepen korelacije veci od 0,5. To su lstat, ptratio i rm.

Sa lstat i ptratio promenljiva medv ima jaku negativnu korelaciju, a sa rm jaku pozitivnu korelaciju. Kod negativne korelacije povecanje jedne utice na smanjenje druge, dok kod pozitivne korelacije povecanje jedne utice na povecanje druge korelacije i obrnuto.

2. Zatim delimo nas dataset na train i test deo

```
library(caret)
set.seed(123)
ind<- createDataPartition(Boston$medv, p = 0.8, list = FALSE)
train.data<- Boston[ind, ]
test.data<- Boston[-ind, ]
```

3. Pravimo model u odnosu na znacajne prediktore

```
lm <- lm(medv ~ lstat + ptratio + rm, data = train.data)
summary(lm)
```

Nas model izgleda: $\text{medv} = 18.19 + (-0.56) \cdot \text{lstat} + (-0.94) \cdot \text{ptratio} + 4.62 \cdot \text{rm}$

Intercept je deo gde prava sece y osu

Koeficijenti su Intercept = 18.11824, lstat = -0.56496 sto znaci da sa jedinicnim povecanjem lstat dolazi do smanjenja medv za -0.56496, jer imamo negativnu korelaciju. Ptratio koeficijent ima vrednost -0.94082, sto znaci da sa jedinicnim povecanjem ptratia dolazi do smanjenja medv za -0.94082, jer je u pitanju negativna korelacija.

Koeficijent rm ima vrednost 4.62379 i s obzirom da je u pitanju pozitivna korelacija, sa jediničnim povecanjem rm, medv se povecava za 4.62379.

#S obzirom da je $\text{Pr}(\text{verovatnoca}) < 0.005$ za sve tri varijable, zaključujemo da su procene znacajne. Multiple R-squared iznosi 0.6935, sto znaci da nas sistem opisuje 69% varijabiliteta zavisne promenljive.

#4. Sada vrsimo predikciju modela koji smo napravili

```
lm.predict <- predict(lm, test.data)
lm.predict
```

```
head(lm.predict)
head(test.data$medv)
```

#5. Crtamo dijagnostičke plotove

```
par(mfrow = c(2,2))
plot(lm)
par(mfrow = c(1,1))
```

#PLOT 1 - Reziduali vs Predviđene vrednosti, ovaj grafik govori o pretpostavci linearnosti da su zavisna i nezavisna promenljiva međusobno u linearnoj korelaciji. Kako reziduali nisu ravnomerno raspoređeni (nije slučaj da teže nuli i da crvena linija odstupa od horizontalne linije) možemo da zaključimo da nije sasvim ispunjen uslov i da postoji indikacija nelinearnog odnosa.

Fitted values - predviđane vrednosti.

Reziduali - razlika između stvarnih i predviđanih vrednosti.

#PLOT 2 - Normal Q-Q, proverava da li je raspodela reziduala normalna. Da bi bili savršeni treba da prate isprekidanu dijagonalu. S obzirom da reziduali odstupaju od dijagonale možemo da zaključimo da nemaju normalnu raspodelu. Uslov je slabo ispunjen i predikcije koje smo dobili ne možemo uzeti za merodavne.

#PLOT 3 - Scale-Location, govori da varijabilitet reziduala treba da bude ujednačen. Varijabilitet treba da varira na prilično ujednačen način. Očekuje se da crvena linija bude horizontalna i varijabilitet ujednačen, da tackice budu oko nje. Uslov nije ispunjen, varijabilitet nije ujednačen i dobijene predikcije nisu merodavne.

#PLOT 4 - Residuals vs Leverage, koristi se za otkrivanje ekstremno visokih ili niskih vrednosti, odnosno outliera, koji mogu da uticu na nas model i da ga poremete. O tome nam govori Kukova distanca(sve vrednosti koje su iznad nje su problematyczne).

#Na osnovu ove 4 slike možemo da zaključimo da model nije pouzdan i ne možemo da se oslonimo na predikcije koje on pruža

6. Sada racunamo sve metrike da bismo videli kolika je greska koju je model napravio

```
rss <-sum((lm.predict - test.data$medv)^2)
tss <-sum((mean(train.data$medv) - test.data$medv)^2)
rsq <- 1 - rss / tss
rsq
```

Nas prediktivni model opisuje 60% varijabiliteta zavisne promenljive

```
rmse<- sqrt(rss / nrow(test.data))
Rmse
```

Model pravi ovoliko gresku prilikom predikcije. Otprilike 5.43 hiljada gresimo u procenama.

2. DRVO ODLUČIVANJA

```
library(ISLR)
?Carseats
str(Carseats)
```

#1. Pravimo novu faktorsku promenljivu

```
uslov<- quantile(Carseats$Sales, probs = 0.75)
Carseats$HighSales<- ifelse(test = Carseats$Sales> uslov, yes="Yes", no = "No")
class(Carseats$HighSales)
Carseats$HighSales<- as.factor(Carseats$HighSales)
class(Carseats$HighSales)
Carseats$Sales<-NULL
```

#2. Delimo nas dataset na train i test deo

```
library(caret)
set.seed(123)
ind <- createDataPartition(Carseats$HighSales, p = 0.8, list = FALSE)
train.data <-Carseats[ind, ]
test.data <- Carseats[-ind, ]
```

#3. Trazimo najbolje cp za nas model, OBAVEZNO učitati library(rpart)

```
library(e1071)
library(rpart)
folds = trainControl(method = "cv", number = 10)
cpGrid = expand.grid(.cp = seq(from = 0.001, to = 0.05, by = 0.001))
set.seed(10)
tr.cp<-train(HighSales ~ .,
             data = train.data,
             method = "rpart",
             control = rpart.control(minsplit = 10),
             trControl = folds, tuneGrid = cpGrid)
tr.cp
```

#4. Pravimo model sa cp koji se pokazao kao najbolji

```
tree<- rpart(HighSales ~ .,
             data = train.data,
             method = "class",
             control = rpart.control(minsplit = 10, cp = 0.013))
```

#5. Crtamo nase stablo

```
library(rattle)
library(rpart.plot)
library(RColorBrewer)
```

```
fancyRpartPlot(tree) ili fja prp(tree)
```

#6. Vrsimo predikciju modela koji smo napravili

```
tree.predict<- predict(object = tree,newdata = test.data,type = "class")
head(tree.predict)
head(test.data$HighSales)
```

#7. Pravimo matricu konfuzije koja pokazuje koliko ima tacno i netacno predvidjenih opservacija u modelu

```
tree.cm<-table(true= test.data$HighSales, predicted= tree.predict)
tree.cm
```

#Pozitivna klasa je No

#TP - vrednosti koje su stvarno pozitivne i koje su predvidjene kao pozitivne

#TN - vrednosti koje su stvarno negativne i koje su predvidjene kao negativne

#FP - vrednosti koje su negativne, a predvidjene su kao pozitivne

#FN - vrednosti koje su pozitivne,a predvidjene su kao negativne

```
compute.eval.metrics <- function(cmatrix) {
  TP <- cmatrix[1,1] # true positive
  TN <- cmatrix[2,2] # true negative
  FP <- cmatrix[2,1] # false positive
  FN <- cmatrix[1,2] # false negative
  acc <- sum(diag(cmatrix)) / sum(cmatrix)
  precision <- TP / (TP + FP)
  recall <- TP / (TP + FN)
  F1 <- 2*precision*recall / (precision + recall)
  c(accuracy = acc, precision = precision, recall = recall, F1 = F1)
}
tree.eval<-compute.eval.metrics(tree.cm)
tree.eval
```

Vrednost za tacnost predvidjenih podataka smo dobili 0.7848101. Ono sto je bilo pozitivno je svrstano kao pozitivno, a negativno kao negativno.

Vrednost za preciznost je 0.8412698 i to predstavlja vrednost stvarno pozitivnih od svih onih koje smo predvideli kao pozitivne.

Vrednost odziva je 0.8833333. Od svih onih koji stvarno pripadaju pozitivnoj klasi, koji je udeo onih koje smo mi predvideli da pripadaju negativnoj klasi.

Opsta F1 metrika objedinjuje obe metrike preciznosti i recalla i daje konkretnu vrednost koliko su one zaista dobre.

3. KNN

```
library(ISLR)
?Carseats
str(Carseats)
```

#1. Pravimo novu faktorsku promenljivu(izlaznu)

```
uslov <- quantile(Carseats$Sales, probs = 0.75)
Carseats$HighSales <- ifelse(test = Carseats$Sales > uslov, yes="Yes", no = "No")
class(Carseats$HighSales)
Carseats$HighSales <- as.factor(Carseats$HighSales)
class(Carseats$HighSales)
Carseats$Sales <- NULL
```

#2. Proveravamo da li numericke promenljive podlezu Normalnoj rasporedili

```
str(Carseats)
numer <- c(1:5,7,8)
apply(X= Carseats[, numer], MARGIN = 2, FUN = shapiro.test)
```

#3. Radimo standardizaciju za varijable koje ne podlezu normalnoj raspodeli

```
carseats.st <- Carseats[,c(2,3,4,7,8)]
carseats.st <- as.data.frame(apply(X=notnorm, MARGIN=2,
                                   FUN= function(x) scale(x, center = median(x), scale = IQR(x))))
```

#4. Radimo standardizaciju za varijable koje podlezu normalnoj raspodeli

```
carseats.st$Price <- as.vector(scale(x = Carseats$Price, center = TRUE, scale = TRUE))
carseats.st$CompPrice <- as.vector(scale(x = Carseats$CompPrice, center = TRUE, scale = TRUE))
```

#5. Sve faktorske vrednosti prebacujemo u numericke

```
carseats.st$Urban <- as.integer(Carseats$Urban)
carseats.st$US <- as.integer(Carseats$US)
levels(Carseats$ShelveLoc)
Carseats$ShelveLoc <- factor(Carseats$ShelveLoc, levels = c("Bad", "Medium", "Good"))
levels(Carseats$ShelveLoc)
carseats.st$ShelveLoc <- as.numeric(Carseats$ShelveLoc)
carseats.st$HighSales <- Carseats$HighSales
str(carseats.st)
```

#6. Delimo nas dataset na train i test deo

```
library(caret)
set.seed(123)
ind <- createDataPartition(carseats.st$HighSales, p = 0.8, list = FALSE)
train.data <- carseats.st[ind, ]
test.data <- carseats.st[-ind, ]
```

#7. Trazimo najbolji k za nas model, OBAVEZNO učitati library(class), staviti .k

```
library(e1071)
library(class)
folds = trainControl(method = "cv", number = 10)
kGrid = expand.grid(.k = seq(from = 3, to = 25, by = 2))
set.seed(1010)
tr.k<-train(HighSales ~ ., data = train.data, method = "knn", trControl = folds, tuneGrid = kGrid)
tr.k
```

#8. Pravimo predikciju sa k koji se pokazao kao najbolji

```
str(carseats.st)
library(class)
knn1 <- knn(train = train.data[,-11],
            test = test.data[,-11],
            cl = train.data$HighSales,
            k = 9)
head(knn1)
head(test.data$HighSales)
```

#9. Pravimo matricu konfuzije koja pokazuje koliko ima tacno i netacno predvidjenih opservacija u modelu

```
knn.cm<-table(true= test.data$HighSales, predicted= knn1)
knn.cm
```

#Pozitivna klasa je No

#TP - vrednosti koje su stvarno pozitivne i koje su predvidjene kao pozitivne

#TN - vrednosti koje su stvarno negativne i koje su predvidjene kao negativne

#FP - vrednosti koje su negativne, a predvidjene su kao pozitivne

#FN - vrednosti koje su pozitivne, a predvidjene su kao negativne

```
compute.eval.metrics <- function(cmatrix) {
  TP <- cmatrix[1,1] # true positive
  TN <- cmatrix[2,2] # true negative
  FP <- cmatrix[2,1] # false positive
  FN <- cmatrix[1,2] # false negative
  acc <- sum(diag(cmatrix)) / sum(cmatrix)
  precision <- TP / (TP + FP)
  recall <- TP / (TP + FN)
  F1 <- 2*precision*recall / (precision + recall)
  c(accuracy = acc, precision = precision, recall = recall, F1 = F1)
}
knn.eval<-compute.eval.metrics(knn.cm)
Knn.eval
```

Vrednost za tacnost predvidjenih podataka smo dobili 0.8607595. Ono sto je bilo pozitivno je svrstano kao pozitivno, a negativno kao negativno.

Vrednost za preciznost je 0.8656716 i to predstavlja vrednost stvarno pozitivnih od svih onih koje smo predvideli kao pozitivne.

Vrednost odziva je 0.9666667. Od svih onih koji stvarno pripadaju pozitivnoj klasi, koji je udeo onih koje smo mi predvideli da pripadaju negativnoj klasi.

Opsta F1 metrika objedinjuje obe metrike preciznosti i recalla i daje konkretnu vrednost koliko su one zaista dobre.

4. NAIVNI BAJES

```
library(ISLR)
str(Carseats)
```

#1.Pravimo faktorsku izlaznu varijablu

```
sales.3Q <- quantile(Carseats$Sales, probs = 0.75)
Carseats$HighSales <- ifelse(test = Carseats$Sales > sales.3Q, 'Yes', 'No')
Carseats$HighSales <- as.factor(Carseats$HighSales)
Carseats$Sales <- NULL
str(Carseats)
```

#NAIVNI BAJES RADI SA FAKTORSKIM VARIJABLAMA I SA NUMERIČKIM KOJE IMAJU N RASPODELU!

#sve numeričke varijable koje nemaju N raspodelu pretvaraš u faktorske (diskretizacijom)!

#2. PROCENA KOJE VARIJABLE IMAJU N RASPODELU

```
apply(X = Carseats[,c(1:5,7,8)], MARGIN = 2, FUN = shapiro.test)
```

#3. DISKRETIZACIJA NUMERIČKIH VARIJABLI KOJE NEMAJU N RASPODELU

```
library(bnlearn)
```

#Filtriramo varijable koje ćemo diskretizovati

```
to.discretize <- c("Education", "Age", "Population", "Advertising", "Income")
```

#Posmatramo opsege svih promenljivih koje ćemo diskretizovati - Kada neka promenljiva ima mali opseg može se podeliti na mali broj delova

```
summary(Carseats[,to.discretize])
```

#U ovom primeru Advertising ima mali opseg pa zato samo za nju kasnije biramo da se deli na 2 dela

#Diskretizujemo varijable

```
discretized <- discretize(data = Carseats[,to.discretize], method = 'quantile', breaks = c(5,5,5,2,5))
```

#Proveravamo da li su sve varijable prevedene u faktorske

```
str(discretized)
```

#Data set discretized ima samo one varijable koje je trebalo diskretizovati

#Potrebno je da mu dodamo preostale varijable originalnog data seta

```
Carseats2 <- cbind(discretized, Carseats[,c(1,5,6,9,10,11)])
```

#Međutim, nastao je data set kod koga su kolone izmešane u odnosu na originalni data set

#Nameštamo da bude isti redosled kolona kao u originalnom data setu

```
Carseats2 <- Carseats2[,names(Carseats)]
```


#4. Pripremljen data set delimo na trening i test

```
library(caret)
set.seed(1010)
indeksi <- createDataPartition(Carseats2$HighSales, p = 0.8, list = FALSE)
train <- Carseats2[indeksi,]
test <- Carseats2[-indeksi,]
```

#5. Pravimo model - Naivni Bajes

```
library(e1071)
nb <- naiveBayes(HighSales ~ ., data = train)
nb.pred <- predict(nb, test, type = "class")
nb.cm <- table(test$HighSales, nb.pred)
nb.cm #ovde sada dobijes da ti je NO pozitivna klasa jer se nalazi na indeksu 1,1
```

#Želimo da dobijemo verovatnoće na osnovu kojih se algoritam odlučuje da li će izlazna varijabla dobiti vrednost Yes ili No - kao type stavljamo "raw" - TREŠHOLD je po defaultu 50%

```
nb.pred.prob <- predict(nb, newdata = test, type = 'raw')
```

#6. POMOĆU ROC KRIVIH ODREĐUJEMO DA LI JE POTREBNO DA MENJAMO NAŠ TREŠHOLD

```
library(pROC)
#Kreiramo ROC krivu
(uzimas ,1 jer je tu tvoja pozitivna klasa)
nb.roc <- roc(response = as.numeric(test$HighSales), predictor = nb.pred.prob[,1], levels = c(2, 1))
#Prikazujemo prostor ispod ROC krive (AUC)
nb.roc$auc
#KOMENTARIŠEMO - ŠTO JE BLIŽE 1 TO JE BOLJE
# Area under the curve: 0.9395
```

#Crtamo ROC krivu

```
plot.roc(nb.roc,
         print.thres = TRUE,
         print.thres.best.method = "youden")
```

#Koordinate svih lokalnih maksimuma na ROC krivoj

```
nb.coords <- coords(nb.roc,
                   ret = c("accuracy", "spec", "sens", "thr"),
                   x = "local maximas")
nb.coords
```

#Kada su nam se prikazali lokalni maksimumi biramo trešhold koji maksimizira ono što mi želimo da postignemo (Trebalo bi da mi želimo da maksimiziramo i Specificity i Sensitivity, pa ćemo birati baš onaj isti trešhold koji nam se ranije prikazao na slici)

```
prob.threshold <- nb.coords[4,5] #Red je uvek isti, kolona nije (biramo trešhold iz jedne od kolona)
```

#7. Radimo predikcije na osnovu našeg trešholda

```
nb.pred2 <- ifelse(test = nb.pred.prob[,1] >= prob.threshold,  
  yes = "No",  
  no = "Yes")  
nb.pred2 <- as.factor(nb.pred)
```

#8. Matrica konfuzije za naše predikcije

```
nb.cm2 <- table(actual = test$HighSales, predicted = nb.pred2)  
nb.cm2
```

#Pozitivna klasa je No

#TP - vrednosti koje su stvarno pozitivne i koje su predviđene kao pozitivne

#TN - vrednosti koje su stvarno negativne i koje su predviđene kao negativne

#FP - vrednosti koje su negativne, a predviđene su kao pozitivne

#FN - vrednosti koje su pozitivne, a predviđene su kao negativne

```
#      predicted  
# actual  No Yes  
# No     55  5  
# Yes    3 16
```

#9. Metrike našeg predviđanja

```
compute.eval.metrics <- function(cmatrix) {  
  TP <- cmatrix[1,1] # true positive  
  TN <- cmatrix[2,2] # true negative  
  FP <- cmatrix[2,1] # false positive  
  FN <- cmatrix[1,2] # false negative  
  acc = sum(diag(cmatrix)) / sum(cmatrix)  
  precision <- TP / (TP + FP)  
  recall <- TP / (TP + FN)  
  F1 <- 2*precision*recall / (precision + recall)  
  c(accuracy = acc, precision = precision, recall = recall, F1 = F1)  
}
```

```
nb.eval <- compute.eval.metrics(nb.cm2)  
Nb.eval
```

#KOMENTARIŠEMO - Ocenjijemo kakve su metrike koje smo dobili

```
# accuracy precision recall    F1  
#0.8987342 0.9482759 0.9166667 0.9322034
```

5. KLAISTERI - KMEANS

1. Ucitavamo data set i gledamo kakva mu je struktura

```
customers <- read.csv(file = "wholesale_customers.csv", stringsAsFactors = F)
str(customers)
```

#proverimo da li u nasem datasetu ima NA vrednosti

```
which(complete.cases(customers) == F)
```

OVO NEMAMO NA KOLOK

#opredelicemo se sada za jedan od dva moguca Channel-a, npr. biramo Retail

```
retail.data <- subset(customers, Channel == 'Retail')
summary(retail.data)
```

#brisemo promenljivu Channel, jer ona vise nema smisla u novom datasetu (svi su tipa retail) NI

```
retail.data$Channel <- NULL
```

2. Sada proveravamo da li neka od varijabli iz novog dataseta ima outliere

```
View(retail.data)
apply(retail.data[, -1], MARGIN = 2, function(x) length(boxplot.stats(x)$out))
```

#Varijable Grocery i Frozen imaju outliere, tako da ih moramo srediti

#crtamo ih da bismo videli da li imaju gornje ili donje outliere (moze i fja boxplot)

```
library(ggplot2)
ggplot(retail.data, aes(x = Region, y = Grocery))+geom_boxplot() - OVO NE MORA DA SE RADI
```

3. a)Npr. prvo sredjujemo varijablu Grocery(ima samo gornje outliere)

```
sort(boxplot.stats(retail.data$Grocery)$out)
#min outlier = 39694
```

```
quantile(retail.data$Grocery, probs = seq(from = 0.9, to = 1, by = 0.025))
```

#Trazimo prvi koji je manji od 39694 i sve outliere postavljamo bas na tu vrednost

```
#to je vrednost na 95%, tj. 34731.7
```

#sada zapamtimo tu vrednost koju smo dobili kao promenljivu new.max

```
new.max <- quantile(retail.data$Grocery, 0.95)
```

#ako neka vrednost prelazi 34371.7, postavi je da ona iznosi bas toliko

```
retail.data$Grocery[retail.data$Grocery > new.max] <- new.max
```

```
boxplot(retail.data$Grocery)$out
```

#vidimo da smo uklonili sve outliere za Grocery

#3. b) sada ponavljamo postupak za Frozen

```
ggplot(retail.data, aes(x = Region, y = Frozen))+geom_boxplot() - OVO NE MORA DA SE RADI
#Frozen promenljiva takodje ima samo gornje outliere
```

```
#sada sortiramo sve te njene outliere i trazimo minimalni
```

```
sort(boxplot.stats(retail.data$Frozen)$out)
```

```
#min = 4736
```

```
quantile(retail.data$Frozen, probs = seq(from = 0.9, to = 1, by = 0.025))
```

```
#prvi manji od 4736 je 4858.55
```

```
#pamtimo tu vrednost u new.max1
```

```
new.max1 <- quantile(retail.data$Frozen, 0.925)
```

```
#sve koje imaju vecu vrednost od new.max1 postavi bas na new.max1
```

```
retail.data$Frozen[retail.data$Frozen > new.max1] <- new.max1
```

```
#crtamo da vidimo da li smo uklonili outliere iz Frozen varijable
```

```
boxplot(retail.data$Frozen)$out
```

```
summary(retail.data)
```

```
#4.PRAVIMO MODEL OD SVIH VARIJABLI i ujedno ga normalizujem
```

```
#normalizacijom cemo srediti sve opsege promenljivih
```

```
source("Utility.R") ILI F-JA ZA NORMALIZACIJU:
```

```
normalize.feature <- function( feature ) {
```

```
  if ( sum(feature, na.rm = T) == 0 ) feature
```

```
  else ((feature - min(feature, na.rm = T))/(max(feature, na.rm = T) - min(feature, na.rm = T)))
```

```
}
```

```
retail.data1.norm <- as.data.frame(apply(retail.data[, c(2:7)], MARGIN = 2, normalize.feature))
```

```
summary(retail.data1.norm)
```

```
#5. Sada trazimo optimalno K(optimalan br grupa)
```

```
novi.dataset <- data.frame()
```

```
for(k in 2:8){
```

```
  set.seed(3108)
```

```
  km.res <- kmeans(x = retail.data1.norm, centers = k, iter.max = 20, nstart = 1000)
```

```
novi.dataset <- rbind(novi.dataset, c(k, km.res$tot.withinss, km.res$betweenss/km.res$totss))
```

```
}
```

```
names(novi.dataset) <- c("cluster", "tot.within.ss", "ratio")
```

```
novi.dataset
```

```
#crtamo novi dataset kako bismo videli koje K nam je optimalno
```

```
ggplot(novi.dataset, aes(x = cluster, y = tot.within.ss))+geom_line()  
#najveci prelom je u 3, znaci da je optimalno K = 3  
#sada pravimo model sa optimalnim k  
set.seed(3108)  
retail.3k <- kmeans(x = retail.data1.norm, centers = 3, iter.max = 20, nstart = 1000)  
retail.3k
```

```
#Imamo 3 klastera velicina(redom): 43, 25, 74  
#Prva observacija nalazi se u 3.klasteru, druga u 1.klasteru, treca takodje u 1.klasteru, itd...  
#Within cluster sum of squares by cluster - Suma kvadrata odstupanja observacija od centra  
#klastera (idealno je da je ta vrednost sto manja, jer su tada observacije bolje grupisane)
```

```
#between_ss - odstupanje centra klastera od globalnog centra  
#total_ss - odstupanje svake pojedinačne observacije od globalnog centra  
#njihov odnos treba da bude sto veci(sto blizi 1), kod nas on iznosi 41.1%
```