Regresija nad skupom diamonds

Domina Sokol

2.6.2021.

Sadržaj

1 Uvod

Rudarenje podataka je proces otkrivanja i izvlačenja uzoraka u velikim skupovima podataka koristeći se metodama strojnog učenja, statistike i sustava baza podataka. Glavni problemi ovog podpolja računalne znanosti su: otkrivanje anomalija (eng. anomaly detection), modeliranje ovisnosti (eng. dependency modeling), grupiranje (eng. clustering), razvrstavanje (eng. classification), regresija (eng. regression) i sažimanje (eng. summarization).

Kao glavna tema ovog seminarskog rada odabran je problem regresije. Demonstrirat će se postupak rukovanja s podacima u svrhu otkrivanja znanja, što je glavna zadaća rudarenja podataka. Proces se sastoji od sljedećih postupaka: odabir podataka, priprema podataka (eng. preprocessing), transformacije podataka u prikladan oblik za daljnje rukovanje, izvlačenje znanja iz skupa podataka (najvažniji dio analize) te na kraju zaključivanje odnosno procjena rezultata.

Pojmovi koji se koriste, npr. evaluacijske metrike, hiperparametri modela i sl., neće se posebno objašnjavati jer je procijenjeno da je njihovo usvajanje ostvareno tijekom izvođenja kolegija i pisanja ovog projekta. Naglasak je stavljen na različite algoritme i njihove usporedbe.

1.1 Postavljanje problema

Jedan od osnovnih problema koji se rješavaju regresijom je problem predviđanja. Predviđanje označava smisleni postupak zaključivanja o budućim vrijednostima i/ili ponašanjima varijable (varijabli) u pitanju, na temelju trenutno dostupnih informacija koje sadrži odabrani skup podataka.

U ovom projektu su uzeti podaci ugrađeni u biblioteku "ggplot2" R programskog jezika, imena "diamonds" koji sadrži informacije o 53940 dijamanata. U daljnjem tekstu dan je detaljniji opis spomenutog skupa podataka. Definiran je problem: na temelju poznatih značajki o dijamantima želi se predvidjeti kolika će biti cijena dijamanta.

2 Analiza podataka

Analiza podataka je proces istraživanja, transformacije i modeliranja podataka u svrhu otkrivanja korisnih informacija i tvorbe smislenih zaključaka. Rudarenje podataka je dio ove discipline koji se posebno usredotočuje na statističko modeliranje i otkrivanje znanja s posebnim naglaskom na predviđanje umjesto samog opisa podataka.

U ovom projektu se koriste standardne deskriptivne statistike koje su nabrojane i objašnjene u nastavku teksta.

2.1 Osnovne značajke podataka

Skup podataka "diamonds" zapisan je R *tibble* formatu, tablici s 53940 redaka i 10 stupaca.

Stupci označavaju 10 značajki (eng. features) dijamanata:

- carat - karati
- cut - rez

```
head(diamonds)
# A tibble: 6 x 10
# carat cut       color clarity depth table price     x     y     z
# <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
#   1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
# 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
# 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
# 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
# 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
# 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

Slika 1: head(diamonds)

- color - boja
- clarity - čistoća
- depth - dubina boje; što je veća, to je dijamant kvalitetniji
- table - dimenzije "ploče" dijamanta; izbrušeni dio na vrhu dragulja
- price - cijena
- x - širina
- y - visina
- z - dužina

U numeričke podatke spadaju: carat, depth, table, price, x, y, z, a u kategoričke: cut, color, clarity. Kategoričke varijable su tipa *ordinal factor* što znači da ih je moguće poredati u niz prema vrijednosti.

```
str(diamonds)
# tibble[,10] [53,940 x 10] (S3: tbl_df/tbl/data.frame)
# $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
# $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
# $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
# $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
# $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
# $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
# $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
# $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
# $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
# $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

Slika 2: str(diamonds)

Ispitano je ima li nedostajućih vrijednosti, tj. je li potrebno izvršiti nadopunu podataka (eng. data imputation), no pokazalo se da to nije potrebno jer su sve vrijednosti prisutne - nema NA vrijednosti u tablici.

```
diamonds[is.na(diamonds) == TRUE]
# <unspecified> [0]
#nema nedefiniranih vrijednosti
```

Slika 3: nema nedostajućih vrijednosti

Također se pokazalo da nema duplikata (pojava istog dijamanta u više redaka tablice).

Daljnjom analizom podataka se pokazalo da postoji nezanemariv broj ekstremnih vrijednosti (eng. outliers), specifično prema varijabli price. Iako je logično da postoje velike razlike u cijeni dijamanata jer se radi o dovoljno velikom broju uzoraka s dovoljno različitim obilježjima, pojava velikog broja ekstremnih vrijednosti svejedno upućuje na moguće probleme u provođenju predviđanja. Ovaj problem će se detaljnjije obraditi kasnije.

2.2 Opisne statistike

Svaku varijablu će se istražiti pomoću osnovnih opisnih statistika (eng. descriptive statistics).

2.2.1 Sredine i raspon uzorka

```
statmod <- function(v){
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v,uniqv)))]
}

for(i in 1:7){
  cat(names(numerical)[i],
    "\n aritmeticka sredina: ", mean(numerical[,i]),
    "\n medijan: ", median(numerical[,i]),
    "\n mod: ", statmod(numerical[,i]))
  readline(prompt = "[enter]")
}
```

Slika 4: sredine

Kategoričke varijable su tipa *factor*, ali ordinalni. To znači da se mogu poredati. Razine koje su najmanje smatraju se najlošijima za dijamante.

Naredbom *summary()* dobiju se najosnovnije statističke informacije. Ono što se izdvaja od prethodno navedenog su kvartili. Donji i gornji kvartil, odnosno prvi i treći kvartil su vrijednosti koje govore o raspodjeli 25% i 75% iznosa podataka. Donji kvartil je vrijednost od koje je manje 25% podataka, a gornji kvartil od koje je manje 75%. Dakle, zajedno s medijanom, na neki način dijele podatke na četiri razreda prema njihovim vrijednostima (zbog toga i uvriježeno ime kvartili). Ovo je zapravo poseban slučaj općeg slučaja kvantila, gdje su postotci proizvoljni, a ne nužno 25% i 75%.

```
# carat
# aritmeticka sredina:  0.7979397
# medijan:  0.7
# mod:  0.3

# depth
# aritmeticka sredina:  61.7494
# medijan:  61.8
# mod:  62

# table
# aritmeticka sredina:  57.45718
# medijan:  57
# mod:  56

# price
# aritmeticka sredina:  3932.8
# medijan:  2401
# mod:  605

# x
# aritmeticka sredina:  5.731157
# medijan:  5.7
# mod:  4.37

# y
# aritmeticka sredina:  5.734526
# medijan:  5.71
# mod:  4.34

# z
# aritmeticka sredina:  3.538734
# medijan:  3.53
# mod:  2.7
```

Slika 5: sredine rezultati

```
range(diamonds$price)
# [1]   326 18823

range(diamonds$carat)
# [1] 0.20 5.01

range(diamonds$cut)
# [1] Fair  Ideal
# Levels: Fair < Good < Very Good < Premium < Ideal

range(diamonds$color)
# [1] D J
# Levels: D < E < F < G < H < I < J

range(diamonds$clarity)
# [1] I1 IF
# Levels: I1 < SI2 < SI1 < VS2 < VS1 < VVS2 < VVS1 < IF
```

Slika 6: raspon

2.2.2 Mjere raspršenosti

Pod mjerama raspršenosti misli se na standardnu devijaciju i varijancu. To su vrijednosti koje govore o raspršenosti podataka i zajedno s koeficijentima asimetrije i zaobljenosti daju više informacija o razdiobi podataka.

Vidi se da se varijabla price posebno izdvaja. Vrijednosti standardne devijacije i varijance su daleko najveće, a to je zato što su vrijednosti varijable price najveće. Koeficijenti zaobljenosti i asimetrije se ne razlikuju redom veličine od ostalih varijabli zato što su to mjere koje su skalirane.

```
summary(diamonds)
#  carat               cut            color        clarity          depth      
#  Min.   :0.2000   Fair     : 1610   D: 6775   SI1    :13065   Min.   :43.00  
#  1st Qu.:0.4000   Good     : 4906   E: 9797   VS2    :12258   1st Qu.:61.00  
#  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194   Median :61.80  
#  Mean   :0.7979   Premium  :13791   G:11292   VS1    : 8171   Mean   :61.75  
#  3rd Qu.:1.0400   Ideal    :21551   H: 8304   VVS2   : 5066   3rd Qu.:62.50  
#  Max.   :5.0100                     I: 5422   VVS1   : 3655   Max.   :79.00  
#                                     J: 2808   (Other): 2531                  
#
#  table          price             x                y                z          
#  Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
#  1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910  
#  Median :57.00   Median : 2401   Median : 5.700   Median : 5.710   Median : 3.530  
#  Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735   Mean   : 3.539  
#  3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040  
#  Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900   Max.   :31.800
```

Slika 7: summary(diamonds)

```
library(e1071)
for(i in 1:7){
  cat(names(numerical)[i],
      "\n varijanca: ", var(numerical[,i]),
      "\n standardna devijacija: ", sd(numerical[,i]),
      "\n koeficijent asimetrije: ", skewness(numerical[,i]),
      "\n koeficijent zaobljenosti: ", kurtosis(numerical[,i]))
  readline(prompt = "[enter]")
}
```

Slika 8: mjere raspršenosti

Svakako je najbitniji zaključak ovdje da varijabla price ima iznimno veliku varijancu i da će to utjecati na rezultate predviđanja, ako se ostavi podatke u ovakvom obliku. Velika varijanca ukazuje na velike oscilacije u vrijednostima varijable.

```
# carat
# varijanca:  0.2246867
# standardna devijacija:  0.4740112
# koeficijent asimetrije:  1.116584
# koeficijent zaobljenosti:  1.25625

# depth
# varijanca:  2.052404
# standardna devijacija:  1.432621
# koeficijent asimetrije:  -0.08228945
# koeficijent zaobljenosti:  5.738447

# table
# varijanca:  4.992948
# standardna devijacija:  2.234491
# koeficijent asimetrije:  0.7968515
# koeficijent zaobljenosti:  2.801271

# price
# varijanca:  15915629
# standardna devijacija:  3989.44
# koeficijent asimetrije:  1.618305
# koeficijent zaobljenosti:  2.177191

# x
# varijanca:  1.258347
# standardna devijacija:  1.121761
# koeficijent asimetrije:  0.3786553
# koeficijent zaobljenosti:  -0.6183029

# y
# varijanca:  1.304472
# standardna devijacija:  1.142135
# koeficijent asimetrije:  2.434031
# koeficijent zaobljenosti:  91.2025

# z
# varijanca:  0.4980109
# standardna devijacija:  0.7056988
# koeficijent asimetrije:  1.522338
# koeficijent zaobljenosti:  47.08029
```

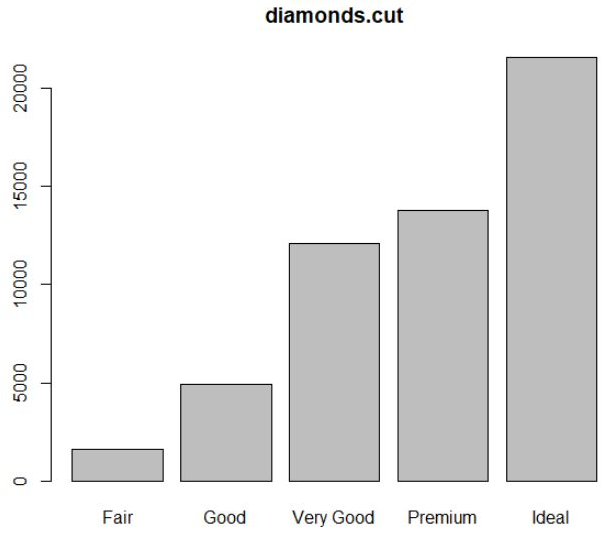
Slika 9: mjere raspršenosti - rezultati

2.2.3 Oblik razdiobe

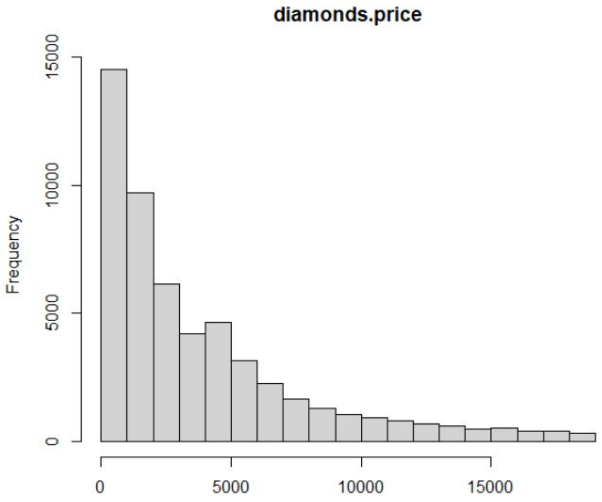
```
# stupcasti dijagram (apsolutnih) frekvencija kategorickih varijabli
for(i in 1:3){
  barplot(table(categorical[,i]), main = names(categorical)[i])
  readline(prompt = "[enter]")
}

# histogram numerickih varijabli
for(i in 1:7){
  hist(numerical[,i], main = names(numerical)[i])
  readline(prompt = "[enter]")
}

# histogram numerickih varijabli (empirijska gustoca) uz teorijsku gustocu
for(i in 1:7){
  hist(numerical[,i], main = names(numerical)[i], probability = TRUE)
  curve(dnorm(x, mean(numerical[,i]), sd(numerical[,i])),
        col= "blue", add=TRUE, lwd=2)
  readline(prompt = "[enter]")
}

# dijagram gustoce podataka uz teorijsku gustocu
for(i in 1:7){
  plot(density(numerical[,i]))
  curve(dnorm(x, mean(numerical[,i]), sd(numerical[,i])),
        col= "blue", add=TRUE, lwd=2)
  readline(prompt = "[enter]")
}
```

Slika 10: razdiobe

Kategoričke varijable prikazuju se stupčastim dijagramom (eng. barplot), a numeričke histogramom. Histogram je sličan stupčastom dijagramu, osim što



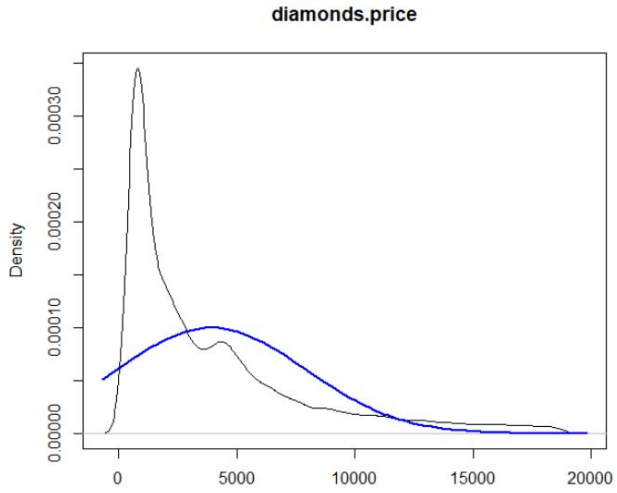
Slika 11: Kao primjer za kategoričke varijable prikazan je stupčasti dijagram varijable cut.



Slika 12: Kao primjer za numeričke varijable pokazan je histogram varijable price...

predstavlja numeričke podatke razvrstane u razrede prema nekim granicama.

Uz histograme numeričkih varijabli dodana je, osim empirijske, i teorijska gustoća zbog usporedbe. Za teorijsku gustoću uzeta je normalna (Gaussova) razdioba, jer je to najpoželjnija razdioba i na nju se podaci i testiraju. Najbitnijia je zato što većina algoritama koji se dalje koriste pretpostavlja normalnu razdiobu ulaznih varijabli. Naravno, slika nije dokaz pa se ne može zaključiti tek tako ima li neka varijabla normalnu razdiobu ili ne. Zbog toga se proveo i test normalnosti koji je potvrdio sumnju da varijable nisu normalno distribuirane i



Slika 13: ...i dijagram empirijske uz teorijsku gustoću.

```
# iz grafickih prikaza vidimo da ne mozemo zakljuciti normalnu distribuiranost
# podataka
# provjerimo ovu hipotezu statistickim testom normalnosti
# koristi se anderson-darling test jer prima vece uzorke
# (shapiro test ne preko 5000)
library(nortest)
for(i in 1:7){
  Sys.sleep(0.1)
  cat(names(numerical)[i], ad.test(numerical[,i])$p.value)
  flush.console()
  readline(prompt = "[enter]")
}
# carat 3.7e-24
# depth 3.7e-24
# table 3.7e-24
# price 3.7e-24
# x 3.7e-24
# y 3.7e-24
# z 3.7e-24

# buduci da su sve p vrijednosti znacajno manje od 0.01, na nivou znacajnosti
# od 99% ne mozemo zakljuciti da su podaci normalno distribuirani
```

Slika 14: test normalnosti

to na razini pouzdanosti od 99%.

Test normalnosti koji je odabran je Anderson-Darling, iz razloga što prima veću količinu podataka, odnosno veći skup vrijednosti.

2.2.4 Korelacija i kovarijanca

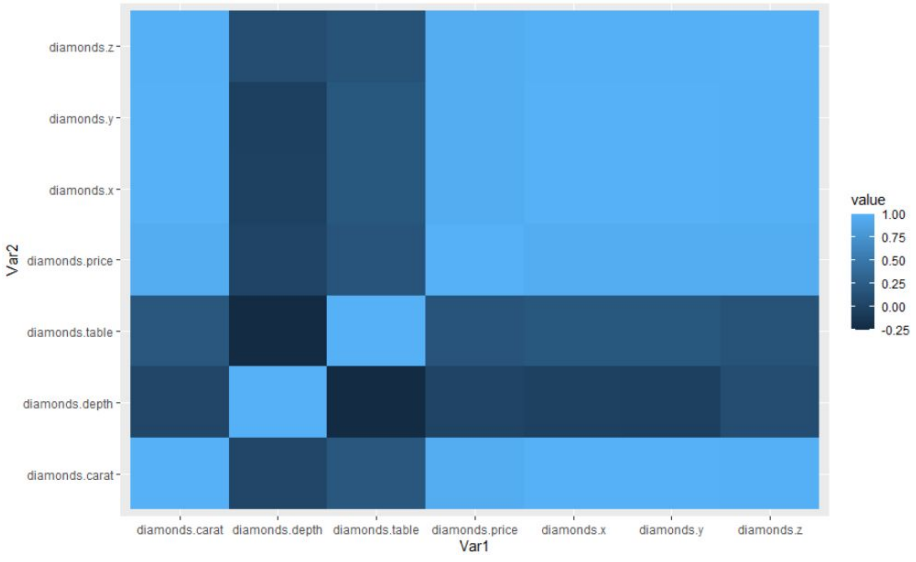
Važno je utvrditi jesu li varijable međusobno korelirane zato što međusobno korelirane ili čak kolinearne nezavisne varijable negativno utječu na moć predviđanja regresijskog modela. Ovaj problem naziva se (multi)kolinearnost varijabli i može mu se doskočiti na više načina. U ovom projektu implementirano je više vrsta rješenja za ovaj problem, s obzirom da se pokazalo potrebnim. Naime, rezultati testa korelacije i kovarijance nezavisnih varijabli ukazuju na njihovu međusobnu zavisnost.

Zbog toga što je potvrđeno da varijable nisu normalno distribuirane, koristio se Spearmanov koeficijent korelacije (umjesto Pearsonovog).

```
cormat <- round(cor(numerical, method="spearman"),2)
cormat
# carat depth table price     x     y    z
# carat  1.00  0.03  0.19  0.96  1.00  1.00 0.99
# depth  0.03  1.00 -0.25  0.01 -0.02 -0.03 0.10
# table  0.19 -0.25  1.00  0.17  0.20  0.20 0.16
# price  0.96  0.01  0.17  1.00  0.96  0.96 0.96
# x      1.00 -0.02  0.20  0.96  1.00  1.00 0.99
# y      1.00 -0.03  0.20  0.96  1.00  1.00 0.99
# z      0.99  0.10  0.16  0.96  0.99  0.99 1.00
```

Slika 15: korelacija

Toplinskom mapom se zornije prikazuju odnosi izmeu varijabli. Ovdje svjetlija boja ukazuje na veću (pozitivnu) koreliranost, a tamnija na manju.



Slika 16: toplinska mapa (eng. heatmap)

```
cov(numerical, method="spearman")
#           carat     depth     table     price         x         y         z
# carat 242339584   7294553  46793232 233404848 241459057 241326989 240744552
# depth   7294553 242287949 -58805838   2428531  -5681796  -6162426  25085006
# table  46793232 -58805838 237662225  41237114  48545402  46985908  38378172
# price 233404848   2428531  41237114 242464705 233539381 233423661 232089991
# x     241459057  -5681796  48545402 233539381 242461218 241950825 239391886
# y     241326989  -6162426  46985908 233423661 241950825 242461229 239322111
# z     240744552  25085006  38378172 232089991 239391886 239322111 242454156
```

Slika 17: kovarijanca

2.2.5 Kutijasti dijagram

Kutijasti dijagram (eng. boxplot) je često korištena grafička metoda prikaza osnovnih statističkih značajki varijable. Na dijagramu su označene najvažnije vrijednosti: dno i vrh "kutije" odgovaraju donjem i gornjem kvartilu, redom, sredina medijanu, repovi ili brkovi (eng. whiskers) vrijednostima 1.5 puta interkvartilnog raspona, a sve što se nalazi van te regije su tzv. ekstremne vrijednosti (eng. outliers).

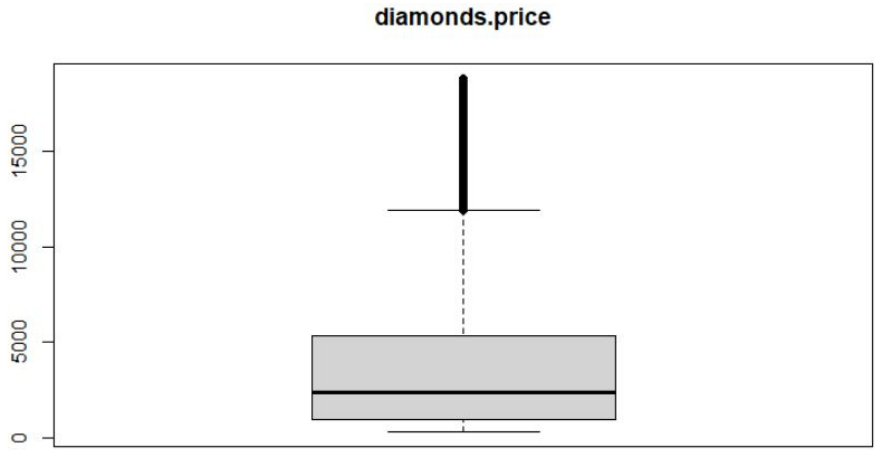
Na slici je prikazana kao primjer varijabla price. Vidi se da postoji mnogo ekstremnih vrijednosti cijene što i nije neočekivano, ako se uzme u obzir da cijene dijamanata mogu varirati jako zbog razlika u kvaliteti. Ovaj skup podataka ima dovoljan broj uzoraka da se takve pojave očituju.

Problemu ekstremnih vrijednosti može se pristupiti na više načina. Kao prvo, potrebno je ustanoviti je li pojava velikog broja ekstremnih vrijednosti uopće problem. Naime, može se dogoditi da u skupu podataka jednostavno postoje primjeri koji odskaču i logički ih se ne bi trebalo izbaciti jer su reprezentativni za svoje skupine. S druge strane, moguća je situacija da ekstremne vrijednosti ukazuju na problem pretvorbe mjernih jedinica, greški u skupu podataka i slično. Ovdje to nije slučaj, nego se jednostavno radi o prirodi cijena dijamanata koja očekivano ima jako velik raspon i oscilacije.

Zbog toga je odlučeno da se svi algoritmi provedu na potpunom skupu podataka koji je dostupan, bez izbacivanja ekstremnih vrijednosti. Kasnije je implementiran i primjer gdje se skup podataka podijelio na dva dijela gdje prvi odgovara nižim cijenama dijamanata a drugi višim. Ovime se pokušalo razdvojiti vrlo različite uzorke dijamanta u nadi da će dva modela biti točnija u procjenama cijene nego samo jedan. Budući da ne postoji generalni konsenzus u načinu pristupa ekstremnim vrijednostima, za potrebe ovog projekta je procijenjeno da je podjela skupa podataka najprikladniji način pokušaja poboljšanja modela.

```
# kutijasti dijagrami
for(i in 1:7){
  boxplot(numerical[,i], main = names(numerical)[i])
  readline(prompt = "[enter]")
}
# svi zajedno
boxplot(numerical)
# cijena ima najvecu varijabilnost u vrijednostima
```

Slika 18: kod za crtanje kutijastih dijagrama



Slika 19: kutijasti dijagram varijable price

3 Priprema podataka

Pod predprocesiranjem ili pripremom podataka podrazumijeva se više postupaka. Ovdje nije bilo potrebno primijeniti sve, npr. nije bilo potreno izbacivati duplikate i riješiti problem NA nedostajućih vrijednosti. Ono što je bilo potrebno je opisano u daljnjem tekstu.

3.1 Podjela podataka

Podaci su podijeljeni na skup za treniranje i skup za testiranje, što je standardan postupak u primjeni algoritama strojnog učenja. Smisao podjele je, u osnovi, sprječavanje prenaučenosti modela i sposobnost evaluacije, tj. procjene moći modela na sasvim novim, testnim podacima, a koji je kreiran, odnosno treniran, na temelju podataka za treniranje.

```
library(healthcareai)
split <- split_train_test(diamonds, price, percent_train = 0.8)

dim(split$train)
# [1] 43152    10
dim(split$test)
# [1] 10788    10

train = split$train
test = split$test
```

Slika 20: podjela podataka

3.2 Kodiranje kategoričkih podataka

Budući da algoritmi zahtijevaju numerički ulaz, tj. funkcijama koje primjenjuju algoritme je potrebno proslijediti brojčane parametre, kategoričke varijable potrebno je predstaviti brojevima. Za ovo postoji više metoda. Ovdje su odabrane dvije.

Prva je tzv. one-hot metoda koja za svaku razinu odnosno kategoriju varijable označava uzorak nulom ili jedinicom. Time se dobije novi, prošireni skup podataka gdje umjesto jednog stupca s oznakom kategoričke varijable stoji više stupaca (može se shvatiti i kao vektor) koji sadrže sve nule osim jedne jedinice. Ta jedinica označava kategoriju koja je prethodno bila opisana riječima. Ovaj pristup je odabran zato što je to očekivani ulaz funkcije iz paketa *neuralnet*.

Druga metoda kodiranja kategoričkih varijabli je tzv. ordinalno kodiranje (eng. ordinal encoding). Budući da se radi o varijablama koje se mogu smisleno poredati, tj. bitno je da poredak ostane uključen u model, odlučeno je da će se primijeniti jednostavno preslikavanje vrijednosti varijabli u skup $\{1, \ldots, n\}$ gdje je n broj kategorija varijable.

```
kodiraj <- function(podaci){
  coded <- cbind(podaci,
                 model.matrix( ~ 0 + podaci$cut,
                               data = podaci))
  coded <- cbind(coded,
                 model.matrix( ~ 0 + podaci$color,
                               data = podaci))
  coded <- cbind(coded,
                 model.matrix( ~ 0 + podaci$clarity,
                               data = podaci))

  coded <- coded[,-(2:4)] # uklonili stare kategoricke varijable

  # promjena imena
  for (i in 1:20){
    duljina <- nchar(names(coded)[8:27][i])
    names(coded)[8:27][i] <- substr(names(coded)[8:27][i], 8, duljina)
  }
  names(coded)[10] <- "cutVery" # izbjegavamo razmake u imenu
  return(coded)
}
train_coded <- kodiraj(train)
test_coded <- kodiraj(test)
```

Slika 21: one-hot kodiranje

```
> head(train_coded)
  carat depth table price    x    y    z cutFair cutGood cutVery cutPremium cutIdeal colorD colorE colorF
1  0.23  61.5    55   326 3.95 3.98 2.43       0       0       0          0        1      0      1      0
2  0.23  56.9    65   327 4.05 4.07 2.31       0       1       0          0        0      0      1      0
3  0.29  62.4    58   334 4.20 4.23 2.63       0       0       0          1        0      0      0      0
4  0.31  63.3    58   335 4.34 4.35 2.75       0       1       0          0        0      0      0      0
5  0.24  62.8    57   336 3.94 3.96 2.48       0       0       1          0        0      0      0      0
6  0.26  61.9    55   337 4.07 4.11 2.53       0       0       1          0        0      0      0      0
  colorG colorH colorI colorJ clarityI1 claritySI2 claritySI1 clarityVS2 clarityVS1 clarityVVS2 clarityVVS1
1      0      0      0      0         0          1          0          0          0          0          0
2      0      0      0      0         0          0          0          0          1          0          0
3      0      0      1      0         0          0          0          1          0          0          0
4      0      0      0      1         0          1          0          0          0          0          0
5      0      0      0      1         0          0          0          0          0          1          0
6      0      1      0      0         0          0          1          0          0          0          0
  clarityIF
1         0
2         0
3         0
4         0
5         0
6         0
```

Slika 22: rezultat one-hot kodiranja kategoričkih varijabli

```
library(cleandata)
kodiraj_ordinalno <- function(podaci, poredak){
  return(encode_ordinal(data.frame(podaci), out.int=T,
         order=poredak))
}

train_coded_ord <- data.frame(train)
test_coded_ord <- data.frame(test)

train_coded_ord[,2] <- kodiraj_ordinalno(train_coded_ord[,2],c("Fair","Good","Very Good","Premium","Ideal"))
train_coded_ord[,3] <- kodiraj_ordinalno(train_coded_ord[,3],c("D","E","F","G","H","I","J"))
train_coded_ord[,4] <- kodiraj_ordinalno(train_coded_ord[,4],c("I1","SI2","SI1","VS2","VS1","VVS2","VVS1","IF"))
test_coded_ord[,2] <- kodiraj_ordinalno(test_coded_ord[,2],c("Fair","Good","Very Good","Premium","Ideal"))
test_coded_ord[,3] <-kodiraj_ordinalno(test_coded_ord[,3],c("D","E","F","G","H","I","J"))
test_coded_ord[,4] <- kodiraj_ordinalno(test_coded_ord[,4],c("I1","SI2","SI1","VS2","VS1","VVS2","VVS1","IF"))
```

Slika 23: ordinalno kodiranje kategoričkih varijabli

```
> head(train_coded_ord)
  carat cut color clarity depth table price    x    y    z
1  0.23   5     2       2  61.5    55   326 3.95 3.98 2.43
2  0.23   2     2       5  56.9    65   327 4.05 4.07 2.31
3  0.29   4     6       4  62.4    58   334 4.20 4.23 2.63
4  0.31   2     7       2  63.3    58   335 4.34 4.35 2.75
5  0.24   3     7       6  62.8    57   336 3.94 3.96 2.48
6  0.26   3     5       3  61.9    55   337 4.07 4.11 2.53
```

Slika 24: rezultat ordinalnog kodiranja kategoričkih varijabli

3.3 Standardizacija

Još jedan od osnovnih postupaka pripreme podataka je standardizacija koja podrazumijeva skaliranje i centriranje podataka. S obzirom da je potrebno izbjeći tzv. curenje podataka (eng. data leakage), standardizacija podataka skupa za testiranje je odrađena koristeći aritmetičku sredinu i standardnu devijaciju podataka skupa za treniranje. Ovime se izbjegla pojava miješanja već podijeljenih podataka, tj. model neće unaprijed imati ikakav podatak o skupu za testiranje u vidu njegove sredine i standardne devijacije. Njegova moć testirat će se na dotad neviđenim podacima.

```
library(caret)
preProcValues <- preProcess(train_coded, method = c("center", "scale"))
train_standardized <- predict(preProcValues, train_coded)
test_standardized <- predict(preProcValues, test_coded)
```

Slika 25: standardizacija

3.4 Odabir značajki

Za odabir značajki (eng. feature selection) odabrano je više metoda. Prva je bila provođenje linearne regresije sa svakom varijablom posebno te u kombinacijama varijabli.

Prvo je provedena regresija sa svim mogućim varijablama, s oba načina kodiranja kategoričkih varijabli. Na prvim slikama se vide rezultati regresije sa svim varijablama gdje su kategoričke kodirane one-hot metodom. Na drugim slikama se vide rezultati regresije sa svim varijablama gdje se koristilo ordinalno kodiranje kategoričkih varijabli. Može se zaključiti da različito kodiranje nije utjecalo na kvalitetu regresije, što je i očekivano.

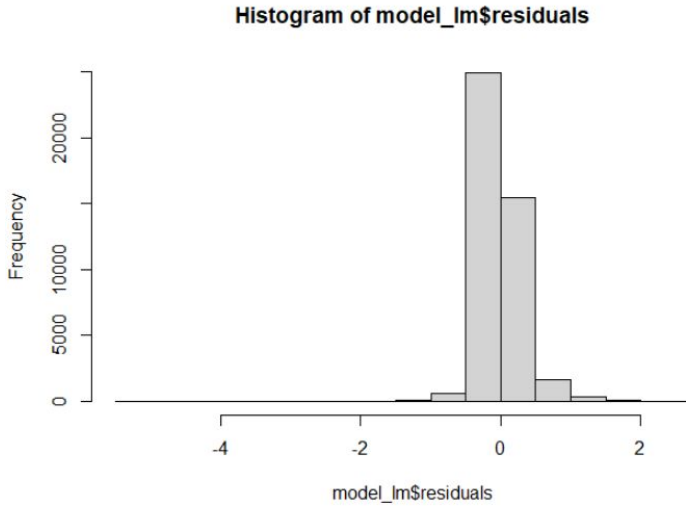
Vidi se da reziduali ne zadovoljavaju tražena svojstva:

```
model_lm <- lm(price ~ ., data=train_standardized)
summary(model_lm)
```

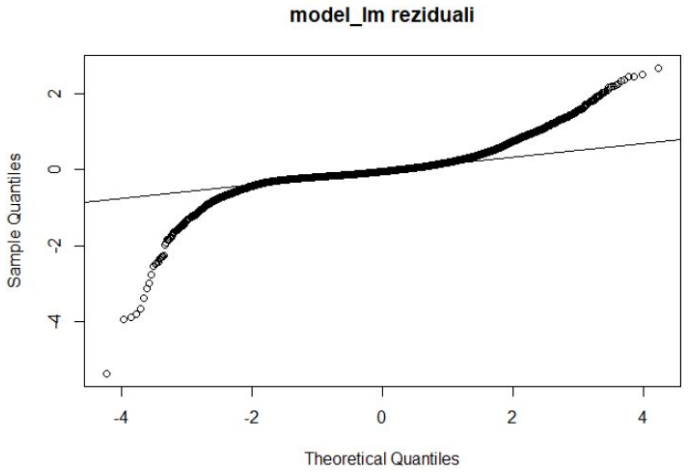
Slika 26: sve varijable, one-hot

```
# Residual standard error: 0.2835 on 43128 degrees of freedom
# Multiple R-squared:  0.9196,  Adjusted R-squared:  0.9196
# F-statistic: 2.146e+04 on 23 and 43128 DF,  p-value: < 2.2e-16
```

Slika 27: R^2 i F-test



Slika 28: reziduali



Slika 29: qqplot - kvantil-kvantil dijagram

- normalna distribuiranost
- aritmetička sredina (očekivanje u slučaju normalnosti) je 0

- nezavisnost
- homoskedastičnost (konstantnost varijance)

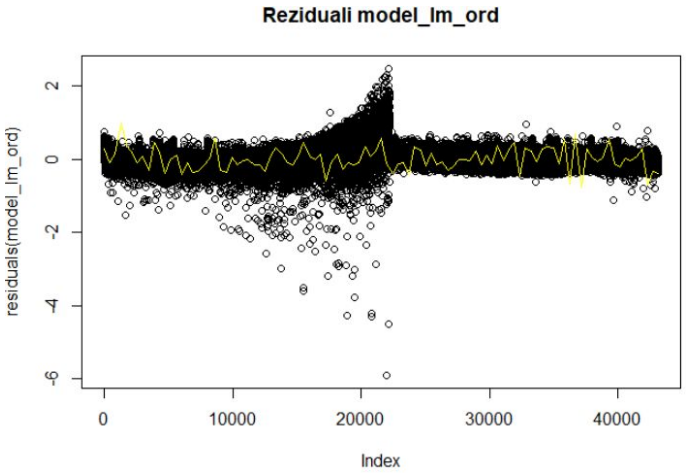
Ova činjenica je provjerena i testom normalnosti.

```
model_lm_ord <- lm(price ~ ., data=train_standardized_ord)
summary(model_lm_ord)
```

Slika 30: sve varijable, ordinalno kodiranje

```
# Residual standard error: 0.3045 on 43142 degrees of freedom
# Multiple R-squared:  0.9073,  Adjusted R-squared:  0.9073
# F-statistic: 4.693e+04 on 9 and 43142 DF,  p-value: < 2.2e-16
```

Slika 31: R^2 i F-test



Slika 32: reziduali, ordinalno kodiranje

```
ad.test(residuals(model_lm))
#
# Anderson-Darling normality test
#
# data:  residuals(model_lm)
# A = 1499.5, p-value < 2.2e-16

# zakljucujemo da reziduali nisu normalno distribuirani...
```

Slika 33: test normalnosti

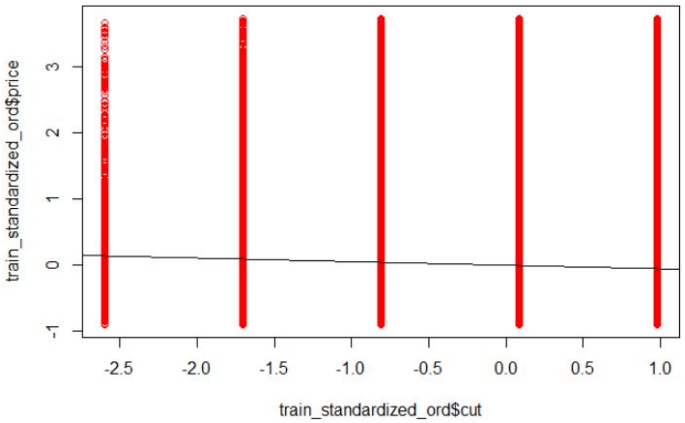
Problem predviđanja iz kategoričkih varijabli bolje ilustrira primjer regresije gdje je za procjenu cijene korištena samo jedna kategorička varijabla. Može se zaključiti da problem nije u vrsti kodiranja nego u samoj činjenici da je varijabla kategorička. Intuitivno, grafički se najbolje vidi zašto se ne može povući smisleni regresijski pravac kroz točke organizirane u stroge segmente.

```
model_lm_cutIdeal_ord <- lm(price ~ cut, data=train_standardized_ord)
summary(model_lm_cutIdeal_ord)
```

Slika 34: regresija s jednom kategoričkom varijablom

```
# Residual standard error: 0.9985 on 43150 degrees of freedom
# Multiple R-squared:  0.003021,  Adjusted R-squared:  0.002998
# F-statistic: 130.8 on 1 and 43150 DF,  p-value: < 2.2e-16
```

Slika 35: R^2 i F-test



Slika 36: regresijski pravac

Zbog ovoga je iskušana i regresija bez kategoričkih varijabli, no to se pokazalo manje uspješnim modelom nego kad se koriste sve dostupne varijable.

```
model_lm_bez_kat <- lm(price ~ carat+depth+table+x+y+z, data=train_standardized_ord)
summary(model_lm_bez_kat)
```

Slika 37: regresija bez kategoričkih varijabli

```
# Residual standard error: 0.3747 on 43145 degrees of freedom
# Multiple R-squared:  0.8596,  Adjusted R-squared:  0.8596
# F-statistic: 4.403e+04 on 6 and 43145 DF,  p-value: < 2.2e-16
```

Slika 38: R^2 i F-test

Provedena je i regresija s izbačenim varijablama koje su se testom korelacije pokazale kao nisko korelirane s predviđanom varijablom price, koja se isto pokazala kao lošija od regresije sa svim varijablama.

```
model_lm_cxyz <- lm(price ~ carat+x+y+z, data=train_standardized_ord)
summary(model_lm_cxyz)
```

Slika 39: regresija s visoko koreliranim numeričkim varijablama

```
# Residual standard error: 0.3815 on 43147 degrees of freedom
# Multiple R-squared:  0.8544,  Adjusted R-squared:  0.8544
# F-statistic: 6.332e+04 on 4 and 43147 DF,  p-value: < 2.2e-16
```

Slika 40: R^2 i F-test

Budući da je provođenje jednostavne linearne regresije u raznim kombinacijama varijabli pokazalo da je većina nezavisnih varijabli u dovoljno visokoj mjeri povezana sa zavisnom varijablom, odlučeno je da će se algoritmi provoditi nad potpunim skupom podataka.

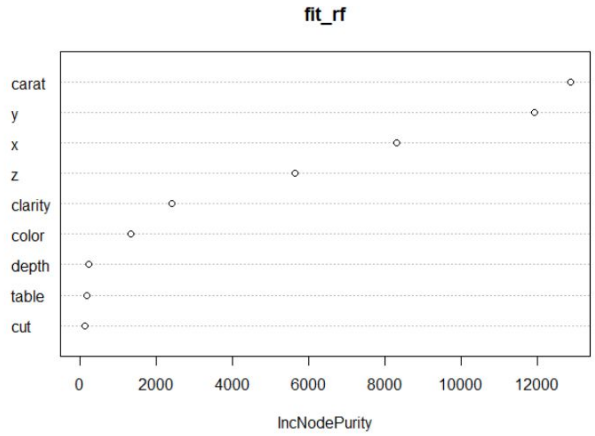
Odabir značajki (eng. feature selection) je proveden nizom jednostavnih linearnih regresija i procjenom njihovim rezultata. Jedina varijabla koja odskače po svojim rezultatima je varijabla depth. Ali, svejedno je uključena u daljnje modele zato što se pokazalo da je linearna regresija jednako uspješna ako se koriste sve dostupne varijable.

Kako bi se pokazalo koje su varijable najznačajnije za model, osim linearne regresije koristio se i model slučajne šume (eng. Random Forest) koji računa važnost značajki prema njihovom Gini indeksu nečistoće. Indeksi dani ovom metodom označavaju koliko svaka varijabla pojedino doprinosi homogenosti podataka.

Konkretno, pri svakoj podjeli čvora u stablu se računa Gini indeks korijena i listova. On predstavlja razinu homogenosti podataka (od 0 za potpunu homogenost do 1 za potpunu heterogenost). Izračuna se razlika Gini indeksa čvorova djece i čvora roditelja te se normalizira. Čistoća podataka je proporcionalna srednjem smanjenju Gini indeksa. Zbog toga je srednje smanjenje Gini indeksa najveće za najznačajnije varijable.

```
library(randomForest)
fit_rf = randomForest(price ~ ., data=train_standardized_ord)
# za stvaranje poretka znacajnosti se koristi gini indeks necistoce
importance(fit_rf)
#          IncNodePurity
# carat      12869.5472
# cut          135.8091
# color       1332.9136
# clarity     2423.5190
# depth        232.7244
# table        175.8075
# x           8298.3771
# y          11908.1023
# z           5647.1423
```

Slika 41: važnost značajki



Slika 42: vizualni prikaz

3.5 Smanjenje dimenzionalnosti

Budući da je jedna od korištenih metoda regresije ujedno i metoda smanjenja dimenzionalnosti podataka, koristit će se isti model i u ovu svrhu. Radi se o metodi parcijalnih najmanjih kvadrata koja je objašnjena kasnije, u dijelu 4.1.

4 Regresija

U ovom dijelu seminara predstavljeni su konkretni algoritmi (osim dosad viđene jednostavne linearne regresije) koji su se koristili u svrhu rješenja postavljenog problema - procjene cijene dijamanata.

4.1 PLS

Metoda parcijalnih najmanjih kvadrata (eng. Partial Least Squares) dijeli sličnosti sa poznatom metodom smanjenja dimenzionalnosti - regresijskom analizom glavnih komponenti (eng. Principal component regression). PCR (regresijska analiza glavnih komponenti je nadogradnja na analizu glavnih komponenti koja je klasifikacijski algoritam po prirodi. Regresijska nadogradnja se odnosi na činjenicu da se nađene glavne komponente koriste kao regresori, odnosno nezavisne varijable pri predviđanju zavisne varijable.

Metoda parcijalnih najmanjih kvadrata se razlikuje od regresijske analize glavnih komponenti po tome što se umjesto maksimizacijskog uvjeta na varijancu pri određivanju glavnih komponenti koristi linearna regresija. Model linearne regresije se nalazi projiciranjem predviđenih i promatranih varijabli na novi prostor.

Ovdje se koristi ovaj tip regresije zbog toga što je posebno pogodan za probleme gdje se pojavljuje multikolinearnost nezavisnih varijabli. Multikolinearnost se odnosi na činjenicu da su varijable pomoću kojih se provodi predviđanje međusobno kolinearne, odnosno zavisne. Ova pojava općenito nije poželjna zbog toga što, intuitivno govoreći, međusobno zavisne varijable nose u sebi informacije koje se preklapaju. Strože govoreći, zavisnost jedne varijable o drugoj znači da se jednu može procijeniti preko druge, i to linearno. Naravno, ovakvi "savršeni" slučajevi se pojavljuju rjeđe, umjesto kolinearnosti nezavisnih varijabli obično postoji djelomična zavisnost ili koreliranost.

U svakom slučaju, pojava kolinearnosti nezavisnih varijabli obično djeluje loše na regresiju zato što, osim numeričkih problema, stvara i problem pri određivanju koeficijenata uz nezavisne varijable. Formalnije, pri određivanju koeficijenata $\beta_1, \ldots, \beta_n$ jednadžbe

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n$$

stvaraju se poteškoće. To se događa zato što pri određivanju jednog koeficijenta, sve druge varijable se smatraju konstantnima, a to u slučaju zavisnosti nekih varijabli nije tako. Intuitivno govoreći, kolinearne nezavisne varijable X_i, X_j nose istu informaciju o promatranoj zavisnoj varijabli Y.

Još jedan problem multikolinearnosti je volatilnost modela, odnosno male promjene na ulazu mogu stvoriti velike promjene u modelu. Glavna negativna posljedica ove činjenice je prenaučenost modela (eng. overfitting). Najbolji regresijski modeli su oni gdje je svaka nezavisna varijabla X_i visoko korelirana

sa zavisnom varijablom Y, a one same su međusobno nekorelirane, ili korelirane vrlo slabo.

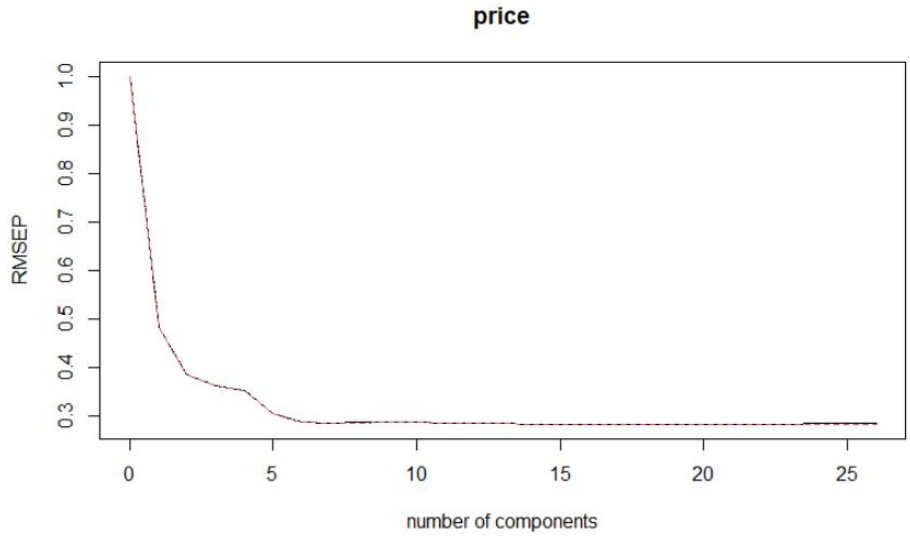
Metoda parcijalnih najmanjih kvadrata ovaj problem rješava svođenjem varijabli na novi prostor koji je razapet glavnim komponentama koje su međusobno okomite, tj. nekorelirane.

```
library(pls)

set.seed(123)
model_pls <- plsr(price ~ ., data=train_standardized, scale=F, validation="CV")

summary(model_pls)
# Data:   X dimension: 43152 26
# Y dimension: 43152 1
# Fit method: kernelpls
# Number of components considered: 26
```

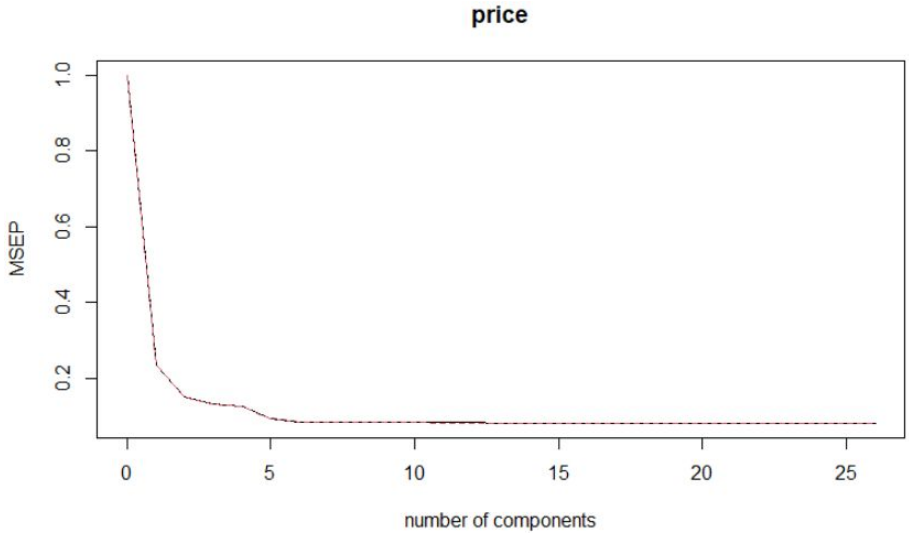
Slika 43: izgradnja PLS modela

```
validationplot(model_pls) # RMSE
validationplot(model_pls, val.type = "MSEP")

plot(model_pls, plottype="correlation")
plot(model_pls, plottype="loadings")
plot(model_pls, plottype="biplot")

predicted_pls <- predict(model_pls, test_standardized, n_comp=2)

RMSE(test_standardized$price, predicted_pls)
# [1] 0.3043098
```

Slika 44: evaluacija PLS modela



Slika 45: RMSE PLS modela



Slika 46: MSE PLS modela

U ovom modelu je također demonstriran utjecaj smanjenja dimenzionalnosti na model. U predviđanju cijene pomoću dobivenog PLS modela korištene su prve dvije glavne komponente - i one su bile dovoljne za postizanje dosta dobrih rezultata.

4.2 Ridge

Još jedan način rješenja problema multikolinearnosti nezavisnih varijabli je regularizacija. Jedan od algoritama koji je primjenjuje je tzv. Ridge (grebenasta) regresija. Ridge regresija primjenjuje poseban slučaj Tikhonove regularizacije gdje su svi parametri jednako regularizirani. Osnovna jednadžba procjenitelja (eng. estimator) je

$$\hat{\beta}_R = (X^T X + \lambda I)^{-1} X^T Y$$

gdje je Y zavisna varijabla, X nezavisna (matrica nezavisnih vektora), I identiteta, a $\lambda \geq 0$ tzv. ridge parametar koji služi kao konstanta koja permutira dijagonalu matrice momenta. Pokazalo se da je ovaj procjenitelj rješenje problema najmanjih kvadrata

$$\min_{\beta}(Y - X\beta)^T(Y - X\beta) + \lambda(\beta^T\beta - c)$$

pod uvjetom $\beta^T\beta = c$. Iz toga slijedi da je λ Lagrangeov multiplikator ovog izraza. U slučaju $\lambda = 0$, ridge procjenitelj se svodi na problem običnih najmanjih kvadrata (eng. Ordinary Least Squares).

Pojam Lagrangeovog multiplikatora se odnosi na strategiju matematičke optimizacije za pronalaženje lokalnih minimuma i maksimuma funkcija pod uvjetima koji su zadani jednadžbama, što je ovdje upravo i slučaj. Osnovna ideja je pretvoriti originalno zadani problem u takav oblik da test derivacije može i dalje biti primijenjen na problem bez uvjeta. Veza između gradijenata funkcije i jednadžbi uvjeta vodi preoblikovanju problema u tzv. Lagrangeovu funkciju.

Općenito, za pronalaženje maksimuma ili minimuma funkcije $f(x)$ pod uvjetom $g(x) = 0$ tvori se Lagrangeova funkcija

$$L(x, \lambda) = f(x) - \lambda g(x)$$

kojoj se nađu stacionarne točke pretpostavljajući da su x i λ konstante, redom. Rješenje koje odgovara originalnom problemu je sedlasta točka Lagrangeove funkcije koja se može pronaći iz ograničene Hesseove matrice, promatrajući njenu definitnost.

```
lambda_seq <- 10^seq(2, -2, by = -.1)

train_X_ridge <- train_standardized[,-4]
train_y_ridge <- train_standardized[,4]

library(glmnet)
model_ridge <- glmnet(as.matrix(train_X_ridge), as.matrix(train_y_ridge), alpha = 0, lambda  = lambda_seq)

summary(model_ridge)
# Length Class     Mode
# a0          41   -none-    numeric
# beta      1066   dgCMatrix S4
# df          41   -none-    numeric
# dim          2   -none-    numeric
# lambda      41   -none-    numeric
# dev.ratio   41   -none-    numeric
# nulldev      1   -none-    numeric
# npasses      1   -none-    numeric
# jerr         1   -none-    numeric
# offset       1   -none-    logical
# call         5   -none-    call
# nobs         1   -none-    numeric
```

Slika 47: izgradnja ridge modela

```
# najbolji lambda:
ridge_cv <- cv.glmnet(as.matrix(train_X_ridge), train_y_ridge, alpha = 0)

plot(ridge_cv)

best_lambda <- ridge_cv$lambda.min
best_lambda
# [1] 0.09217505

best_fit <- ridge_cv$glmnet.fit
head(best_fit)
# $lambda
# [1] 921.64666700 839.77011734 765.16725469 697.19190473 635.25529751
# [6] 578.82096777 527.40010833 480.54733633 437.85683545 398.95884101
# [11] 363.51643718 331.22263882 301.79773250 274.98685376 250.55777959
# [16] 228.29891704 208.01747048 189.53777174 172.69975850 157.35758795
```

Slika 48: unakrsna validacija

```
best_ridge <- glmnet(as.matrix(train_X_ridge), train_y_ridge, alpha = 0, lambda = best_lambda)

coef(best_ridge)
# 27 x 1 sparse Matrix of class "dgCMatrix"
```

Slika 49: najbolji model

Grebenasta regresija je dala odlične rezultate procjene cijene, s vrijednošću koeficijenta determinacije oko 89%. Naravno, nije uputno oslanjati se samo na jednu vrstu metrike pri evaluaciji modela. Zbog toga se promotrilo i srednju kvadratnu grešku kao i srednju apsolutnu grešku.

```
RMSE(prediction_ridge, test_standardized[,4]) # [1] 0.326149
MAE(prediction_ridge, test_standardized[,4]) # [1] 0.2330374
R2(prediction_ridge, test_standardized[,4]) # 0.8942873
```

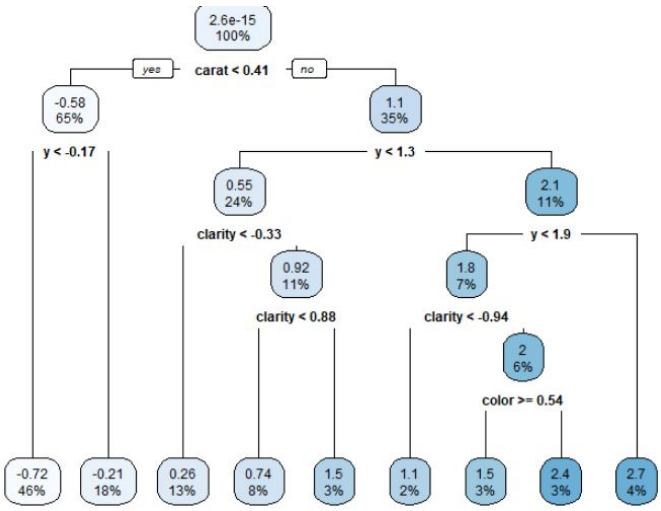
Slika 50: evaluacija ridge modela

4.3 Stablo odluke

Kao još jedna metoda regresije isprobano je regresijsko stablo odluke. Jedna od dobro poznatih mana stabala odluke je problem lake prenaučenosti. Iz tog razloga se implementiralo i slučajnu šumu kako bi se usporedilo rezultate.

```
library(rpart)
library(rpart.plot)
stablo <- rpart(price ~ ., data=train_standardized_ord,
                control=rpart.control(minsplit=10))

rpart.plot(stablo)
```

Slika 51: model stabla



Slika 52: slika modela stabla

Još jedna pogodnost modela stabla je to što daje uvid u značajnost korištenih varijabli tako da se i ono moglo koristiti kao metoda nalaženja pogodnih značajki za predviđanje.

```
stablo$variable.importance
#   carat           y           x           z     clarity
# 36258.17809 35443.97892 35124.44379 34127.07331  4559.12719
#
#   color       table         cut       depth
# 1643.83637    58.59962    42.28090    18.79831
```

Slika 53: procjena važnosti značajki

Evaluaciju modela najlakše je procijeniti iz CP tablice - popisa najvažnijih svojstava stabla: kompleknosti stabla, greške pri treniranju i greške pri unakrsnoj validaciji (unakrsna validacija se provodi prema default postavkama), poredanih prema broju grananja u stablu.

```
stablo$cptable
#           CP nsplit  rel error    xerror        xstd
# 1 0.60784715      0 1.00000000 1.0000265 0.009847608
# 2 0.18681490      1 0.39215285 0.3921776 0.004389817
# 3 0.03359112      2 0.20533795 0.2053892 0.002306157
# 4 0.02566435      3 0.17174683 0.1718600 0.002304532
# 5 0.02545006      4 0.14608248 0.1387149 0.001960198
# 6 0.01021766      5 0.12063242 0.1221254 0.001798044
# 7 0.01002529      6 0.11041476 0.1181678 0.001738577
# 8 0.01000000      8 0.09036417 0.1132136 0.001675692
```

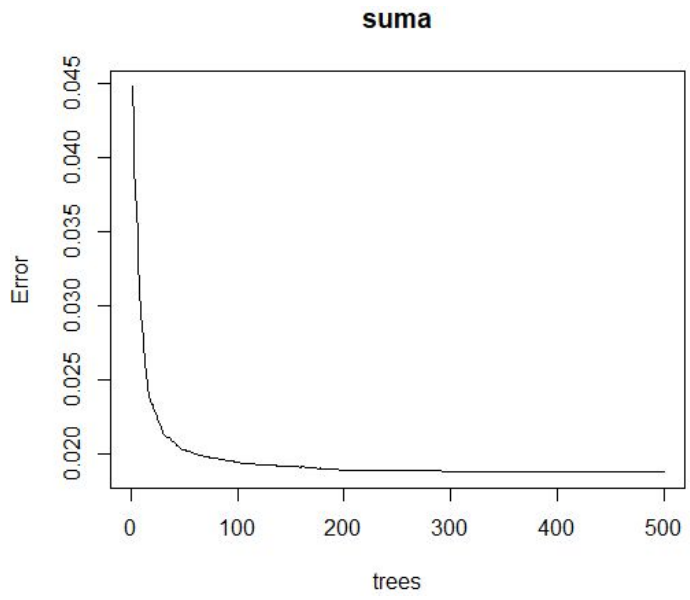
Slika 54: evaluacija modela stabla

4.4 Slučajna šuma

Slučajna šuma je jedna od najkorištenijih nadogradnji na stabla odluke koja rješava problem prenaučenosti. Pokazalo se da, u skladu s tim svojstvom, daje bolje rezultate nego samo jedno stablo odluke.

```
library(randomForest)
suma <- randomForest(price ~ ., data=train_standardized_ord)
print(suma)
# Call:
#   randomForest(formula = price ~ ., data = train_standardized_ord)
# Type of random forest: regression
# Number of trees: 500
# No. of variables tried at each split: 3
#
# Mean of squared residuals: 0.0187394
# % Var explained: 98.13

plot(suma)
```

Slika 55: izgradnja modela slučajne šume



Slika 56: greška modela slučajne šume s obzirom na broj stabala

4.5 KNN

Jedan od algoritama koji se pokazao iznenađujuće dobrim je KNN - K najbližih susjeda (eng. K nearest neighbors). Iako se obično navodi kao primarno klasifikacijski algoritam, njime se može provoditi i regresija. Jedina razlika je u izlaznim podacima. U slučaju regresije se umjesto procjene klase na temelju k najbližih susjednih podataka izračuna prosječni izlaz k najbližih susjeda i ta vrijednost se proslijedi kao konačni izlaz.

```
model_knn <- knnreg(train_standardized[-4], as.numeric(unlist(train_standardized[4])))
```

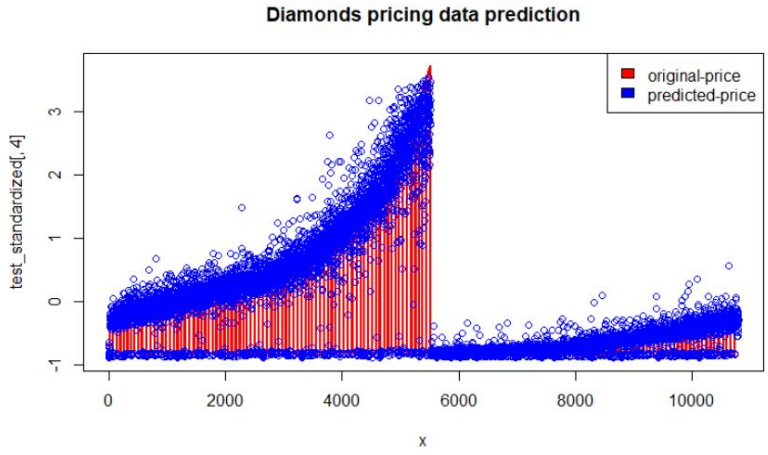
Slika 57: izgradnja knn modela

```
pred_y = predict(model_knn, test_standardized[,-4])

print(data.frame(test_standardized[,4], pred_y))

mse = mean((test_standardized[,4] - pred_y)^2)
mae = caret::MAE(test_standardized[,4], pred_y)
rmse = caret::RMSE(test_standardized[,4], pred_y)

cat("MSE: ", mse, "MAE: ", mae, " RMSE: ", rmse)
# MSE:  0.04489283 MAE:  0.1059654  RMSE:  0.2118793

model_knn$k
# [1] 5
```

Slika 58: evaluacija knn modela

```
x = 1:length(test_standardized[,4])

plot(x, test_standardized[,4], col = "red", type = "l", lwd=2,
     main = "Diamonds pricing data prediction")
points(x, pred_y, col = "blue")
legend("topright",  legend = c("original-price", "predicted-price"),
       fill = c("red", "blue"), col = 2:3,  adj = c(0, 0.6))
```

Slika 59: grafički prikaz - kod



Slika 60: grafički prikaz procjene cijene

4.6 Neuronska mreža

Neuronska mreža je implementirana pomoću dvije biblioteke - *nnet* i *neuralnet*. Osnovna razlika je u tome što neuronske mreže građene pomoću paketa *nnet* mogu imati samo jedan skriveni sloj. Nije moguće definirati višeslojni model. Iz tog razloga se napravio i model pomoću paketa *neuralnet* gdje je moguće implementirati višeslojni perceptron.

Nažalost, nedostatak biblioteke *neuralnet* je u tome što se modeli prilično sporo treniraju. To je, naravno, relativan pojam. Ali svakako, za jednostavnu

višeslojnu mrežu sa samo 8 neurona ukupno, treniranje duže od 5 minuta nije poželjno. Iz svega navedenog zaključuje se da nije uputno koristiti R ugrađene biblioteke za neuronske mreže već se poslužiti drugim programskim jezicima ili uključiti Pythonov Tensorflow modul.

```
library(nnet)

model_nnet <- nnet(price/18823 ~ ., data=train,
                   size=5, rang=0.1, decay=5e-4, maxit=1000)
# weights:  126
# initial  value 5514.767972
# iter  10 value 3831.570201
# iter  20 value 3822.900389
```

Slika 61: izgradnja nnet modela

```
# iter 990 value 37.955595
# iter1000 value 37.931580
# final  value 37.931580
# stopped after 1000 iterations

predicted_nnet <- predict(model_nnet)*18823

mean((predicted_nnet - train$price)^2)
# [1] 339428.8

plot(train$price, predicted_nnet, main="prediction vs actual", xlab="actual")

model_nnet
# a 23-5-1 network with 126 weights
```

Slika 62: evaluacija nnet modela

```
model_1 <- neuralnet(price ~ carat + depth + table + x + y + z +
                       cutFair + cutGood + cutVery +
                       cutPremium + cutIdeal + colorD + colorE + colorF +
                       colorG + colorH + colorI + colorJ + clarityI1 +
                       claritySI2 + claritySI1 + clarityVS2 + clarityVS1 +
                       clarityVVS2 + clarityVVS1 + clarityIF,
                     data = train_standardized, hidden=c(5,3),
                     linear.output=T)
```

Slika 63: izgradnja neuralnet modela

Za evaluaciju modela korištena je klasična metrika korijena srednje kvadratne greške - RMSE (eng. root mean squared error). Iznosi:582.6052, što nije baš najbolji rezultat. Model je također prikazan grafički pomoću naredbe *plot*.

Ono što je moguće napraviti je tzv. poboljšanje hiperparametara modela (eng. hyperparameter tuning). Ovaj postupak se odnosi na traženje najboljih kombinacija hiperparametara (parametara koji se prosljeđuju modelu pri pozivu), na zadanoj mreži (eng. grid). Naravno, nije lako pogoditi odmah kakve opcije hiperparametara je uopće uputno proslijediti. Iz tog razloga su provedena traženja najboljih hiperparametara na više mreža.

Svejedno, ni najbolji dobiveni rezultati (u vidu standardne metrike koeficijenta determinacije) nisu zadovoljavajući što upućuje na to da model nije moguće poboljšati do željene razine performansi samo namještanjem hiperparametara.

```
mygrid <- expand.grid(.decay=c(0.5, 0.1), .size=c(4,5,6))
nnetfit <- caret::train(train_standardized[,-4], as.numeric(train_standardized$price),
                        method="nnet",
                        maxit=10, tuneGrid=mygrid, trace=T)
print(nnetfit)
# Neural Network
#
# 43152 samples
# 26 predictor
#
# No pre-processing
# Resampling: Bootstrapped (25 reps)
# Summary of sample sizes: 43152, 43152, 43152, 43152, 43152, 43152, ...
# Resampling results across tuning parameters:
#
#   decay  size  RMSE       Rsquared   MAE
# 0.1    4     0.9569577  0.1795658  0.7495959
# 0.1    5     0.9774162  0.1894467  0.7528410
# 0.1    6     0.9842012  0.1284469  0.7632721
# 0.5    4     0.9464352  0.2771378  0.7504176
# 0.5    5     0.9377478  0.2904293  0.7499564
# 0.5    6     0.9483520  0.2309230  0.7560172
#
# RMSE was used to select the optimal model using the smallest value.
# The final values used for the model were size = 5 and decay = 0.5.
```

Slika 64: grid search 1

```
mygrid <- expand.grid(.decay=c(0.5, 0.55, 0.6, 0.7), .size=c(5,8,10))
nnetfit <- caret::train(train_standardized[,-4], as.numeric(train_standardized$price),
                        method="nnet",
                        maxit=10, tuneGrid=mygrid, trace=T)
print(nnetfit)
# Neural Network
#
# 43152 samples
# 26 predictor
#
# No pre-processing
# Resampling: Bootstrapped (25 reps)
# Summary of sample sizes: 43152, 43152, 43152, 43152, 43152, 43152, ...
# Resampling results across tuning parameters:
#
#   decay  size  RMSE       Rsquared   MAE
# 0.50    5     0.9401667  0.2415991  0.7514371
# 0.50    8     0.9375709  0.3316593  0.7512005
# 0.50   10     0.9246537  0.4088948  0.7348918
# 0.55    5     0.9303431  0.2895997  0.7503190
# 0.55    8     0.8825048  0.4377542  0.7285253
# 0.55   10     0.8864466  0.4393302  0.7191751
# 0.60    5     0.9440614  0.2911949  0.7403949
# 0.60    8     0.9433828  0.2566675  0.7344857
# 0.60   10     0.9492487  0.2725225  0.7367964
# 0.70    5     0.9181762  0.3740114  0.7275885
# 0.70    8     0.9816514  0.1624759  0.7521667
# 0.70   10     0.9457260  0.2986010  0.7390233
#
# RMSE was used to select the optimal model using the smallest value.
# The final values used for the model were size = 8 and decay = 0.55.
```

Slika 65: grid search 2

Model iz biblioteke *neuralnet* nije bilo moguće istrenirati, na svaki pokušaj izgradnje modela javlja se ista greška: *"Warning message: Algorithm did not converge in 1 of 1 repetition(s) within the stepmax."*.

5 Podjela skupa prema ekstremnim vrijednostima

Ideja koja se isprobala je podjela skupa podataka na dvije manje grupe, motivirana velikim brojem ekstremnih vrijednosti (eng. outliers). Pokazalo se da to ipak negativno utječe na rezultate regresije. Isprobane su dvije (dosad najuspješnije) metode: grebenasta regresija i metoda parcijalnih najmanjih kvadrata.

```
train_X_ridge_jeftiniji <- train_standardized_jeftiniji[,-4]
train_y_ridge_jeftiniji <- train_standardized_jeftiniji[,4]

model_ridge_jeftiniji <- glmnet(as.matrix(train_X_ridge_jeftiniji), as.matrix(train_y_ridge_jeftiniji)
                                alpha = 0, lambda  = lambda_seq_1)
```

Slika 66: izgradnja ridge modela - prvi skup

```
ridge_cv_jeftiniji <- cv.glmnet(as.matrix(train_X_ridge_jeftiniji), train_y_ridge_jeftiniji, alpha = 0)

plot(ridge_cv_jeftiniji)

best_lambda_jeftiniji <- ridge_cv_jeftiniji$lambda.min
best_lambda_jeftiniji
# [1] 0.03899967

best_fit_jeftiniji <- ridge_cv_jeftiniji$glmnet.fit
```

Slika 67: najbolji lambda - prvi skup

```
train_X_ridge_skuplji <- train_standardized_skuplji[,-4]
train_y_ridge_skuplji <- train_standardized_skuplji[,4]

model_ridge_skuplji <- glmnet(as.matrix(train_X_ridge_skuplji), as.matrix(train_y_ridge_skuplji),
                              alpha = 0, lambda  = lambda_seq_1)
```

Slika 68: izgradnja ridge modela - drugi skup

```
ridge_cv_skuplji <- cv.glmnet(as.matrix(train_X_ridge_skuplji), train_y_ridge_skuplji, alpha = 0)

plot(ridge_cv_skuplji)

best_lambda_skuplji <- ridge_cv_skuplji$lambda.min
best_lambda_skuplji
# [1] 0.06801911

best_fit_skuplji <- ridge_cv_skuplji$glmnet.fit
```

Slika 69: najbolji lambda - drugi skup

```
best_ridge_jeftiniji <- glmnet(as.matrix(train_X_ridge_jeftiniji), train_y_ridge_jeftiniji,
                               alpha = 0, lambda = best_lambda_jeftiniji)

best_ridge_skuplji <- glmnet(as.matrix(train_X_ridge_skuplji), train_y_ridge_skuplji,
                             alpha = 0, lambda = best_lambda_skuplji)

prediction_ridge_jeftiniji <- predict(best_ridge_jeftiniji, s = best_lambda_jeftiniji,
                                      newx = as.matrix(test_standardized_jeftiniji[,-4]))

RMSE(prediction_ridge_jeftiniji, test_standardized_jeftiniji[,4]) # [1] 0.7548218
MAE(prediction_ridge_jeftiniji, test_standardized_jeftiniji[,4]) # [1] 0.5979697
R2(prediction_ridge_jeftiniji, test_standardized_jeftiniji[,4]) # 0.4275972

prediction_ridge_skuplji <- predict(best_ridge_skuplji, s = best_lambda_skuplji,
                                    newx = as.matrix(test_standardized_skuplji[,-4]))

RMSE(prediction_ridge_skuplji, test_standardized_skuplji[,4]) # [1] 0.7005069
MAE(prediction_ridge_skuplji, test_standardized_skuplji[,4]) # [1] 0.5233988
R2(prediction_ridge_skuplji, test_standardized_skuplji[,4]) # 0.5642041
```

Slika 70: evaluacija oba ridge modela

Kao što se vidi na slici evaluacije standardnim metrikama, ova dva modela su znatno lošija nego globalni model istreniran nad svim dostupnim podacima.

Sljedeća metoda je metoda parcijalnih najmanjih kvadrata.

```
model_pls_jeftiniji <- plsr(price ~ ., data=train_standardized_jeftiniji, scale=F, validation="CV")
summary(model_pls_jeftiniji)
# Data:   X dimension: 40323 9
# Y dimension: 40323 1
# Fit method: kernelpls
# Number of components considered: 9
#
# VALIDATION: RMSEP
# Cross-validated using 10 random segments.
# (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
# CV             1   0.4626   0.3668    0.352   0.3450   0.3210   0.3185   0.3179   0.3163   0.3183
# adjCV          1   0.4626   0.3668    0.352   0.3449   0.3207   0.3183   0.3178   0.3163   0.3180
#
# TRAINING: % variance explained
# 1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
# X         46.98    58.71    70.10    80.56    84.67    90.49    99.39    99.89   100.00
# price     78.61    86.58    87.67    88.28    89.69    89.83    89.86    90.03    90.06
```

Slika 71: izgradnja pls modela - prvi skup

```
validationplot(model_pls_jeftiniji) # RMSE
validationplot(model_pls_jeftiniji, val.type = "MSEP")

plot(model_pls_jeftiniji, plottype="correlation")
plot(model_pls_jeftiniji, plottype="loadings")
plot(model_pls_jeftiniji, plottype="biplot")

predicted_pls_jeftiniji <- predict(model_pls_jeftiniji, test_standardized_jeftiniji, n_comp=2)

RMSE(test_standardized_jeftiniji$price, predicted_pls_jeftiniji)
# [1] 0.3435771
```

Slika 72: evaluacija pls modela - prvi skup

```
model_pls_skuplji <- plsr(price ~ ., data=train_standardized_skuplji, scale=F, validation="CV")
summary(model_pls_skuplji)
# Data:   X dimension: 2832 9
# Y dimension: 2832 1
# Fit method: kernelpls
# Number of components considered: 9
#
# VALIDATION: RMSEP
# Cross-validated using 10 random segments.
# (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
# CV               1   0.9474   0.8838   0.8727   0.8677   0.8603   0.8590   0.8627   0.9161    2.073
# adjCV            1   0.9473   0.8811   0.8692   0.8642   0.8570   0.8559   0.8589   0.9075    1.984
#
# TRAINING: % variance explained
# 1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
# X         42.51    47.85    58.56    66.40    74.03    86.75    87.93    94.67   100.00
# price     10.73    28.21    30.76    31.78    32.31    32.41    32.67    32.67    32.67
```

Slika 73: izgradnja pls modela - drugi skup

```
validationplot(model_pls_skuplji) # RMSE
validationplot(model_pls_skuplji, val.type = "MSEP")

plot(model_pls_skuplji, plottype="correlation")
plot(model_pls_skuplji, plottype="loadings")
plot(model_pls_skuplji, plottype="biplot")

predicted_pls_skuplji <- predict(model_pls_skuplji, test_standardized_skuplji, n_comp=2)

RMSE(test_standardized_skuplji$price, predicted_pls_skuplji)
# [1] 0.8980163
```

Slika 74: evaluacija pls modela - drugi skup

```
# losiji rezultati, cak i ako ukljucimo vise komponenti:
predicted_pls_skuplji <- predict(model_pls_skuplji, test_standardized_skuplji, n_comp=6)

RMSE(test_standardized_skuplji$price, predicted_pls_skuplji)
# [1] 0.8574603
```

Slika 75: evaluacija pls modela s više komponenti - drugi skup

Na zadnjem primjeru se lijepo vidi da metoda smanjenja dimenzionalnosti ne mora uvijek djelovati znatno pozitivno na rezultate modela. Naime, korijen srednje kvadratne greške se ne razlikuje mnogo kad se koristi model s više glavnih komponenti. Kod provođenja regresije, a i općenito u izgradnji modela, treba uvijek biti na oprezu.

6 Zaključak

Nakon usporedbe više modela, može se zaključiti da su najbolji: grebenasta (Ridge) regresija, algoritam parcijalnih najmanjih kvadrata (PLS), K najbližih susjeda (KNN) i slučajna šuma. Sposobnosti neuronske mreže ugrađene u R su, barem za ovaj slučaj i ove podatke, nezadovoljavajuće i daleko su najgori model.

Konačni zaključak je da je predviđanje cijene moguće uspješno provesti, zahvaljujući boljim (prijašnje navedenim) algoritmima. Ovime je postavljeni problem riješen. Uz rješenje problema, demonstrirane su brojne opcije poboljšanja modela poput smanjenja dimenzionalnosti, odabira značajki na temelju korelacije te provođenja istrage nad mrežom hiperparametara, uz unakrsnu validaciju.