Seminarska naloga 1

Umetna inteligenca
FRI BVS

Domen Koščak, Jan Leskovec

November 2020

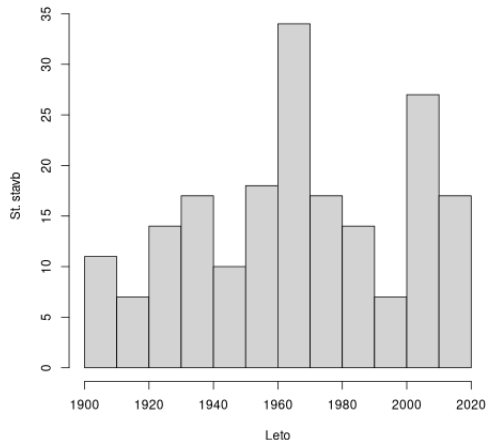
Kazalo

1 Uvod

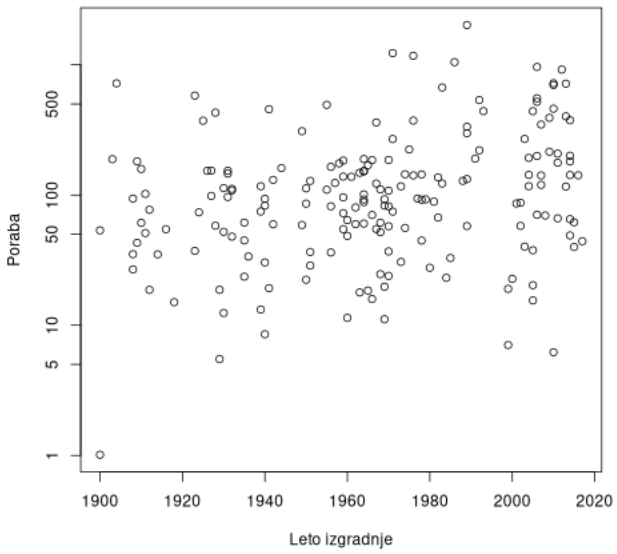
V seminarski nalogi bova predstavila uporabo strojnega učenja za gradnjo modelov za napovedovanje, v najinem primeru za napovedovanje porabe električne energije. Začela bova z analizo podanih podatkov in prikazom različnih grafov. Nadaljevala bova z ocenjevanjem že obstoječih atributov in konstruiranjem novih, s katerimi bi sam model lahko izboljšala. Sledi modeliranje na različne načine. Na koncu bova dobljene modele še ovrednotila tako, da bova podatke razdelila na 12 podmnožic glede na mesec zajema in regije, nato pa bova uporabila podatke ene množice za učenje modela, podatke druge množice pa kot testne za pridobljen model.

2 Vizualizacija podatkov

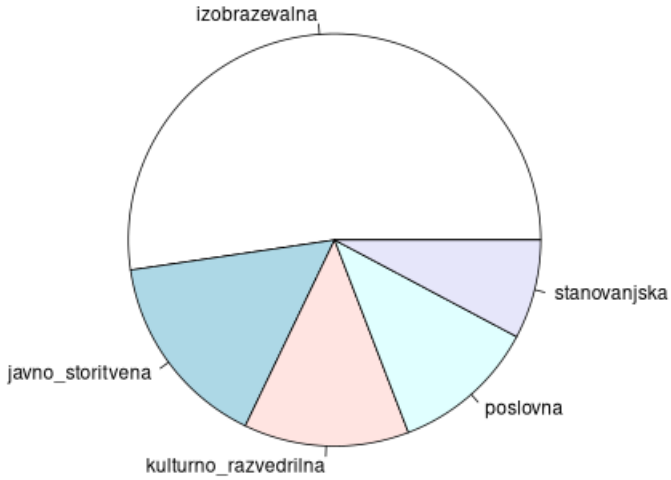
Naloge sva se lotila z analiziranjem podatkov in risanjem nekaterih grafov. Predstavila bova nekatere izmed njih.



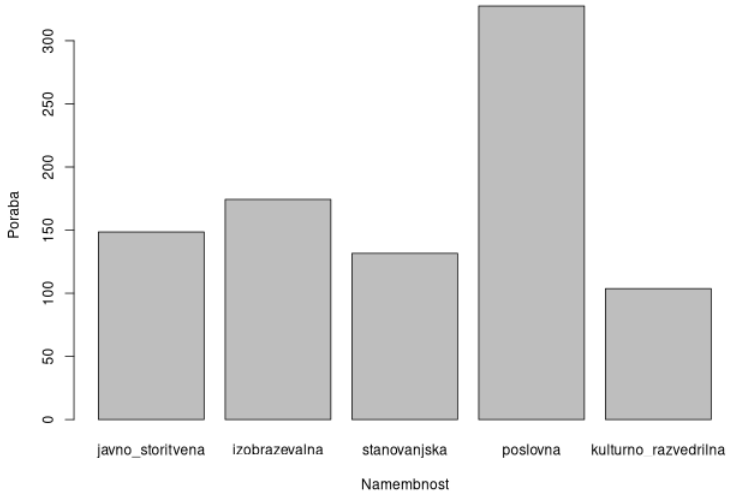
Na zgornjem grafu lahko vidimo, da je največ izmed opazovanih zgradb zgrajenih med letoma 1960 in 1970. Sledijo zgradbe zgrajene med letoma 2000 in 2010, ostale pa so približno enakomerno razporejene.



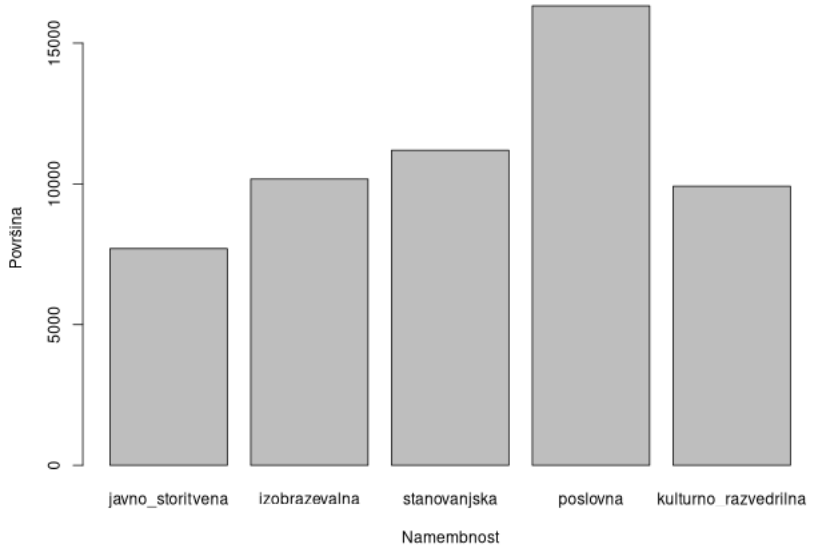
Na zgornjem grafu lahko vidimo, da električna poraba ni pretirano odvisna od leta izgradnje same stavbe. Večina stavb ima povprečno porabo manjšo od 500 kWh, nekatere pa imajo porabo tudi višjo.



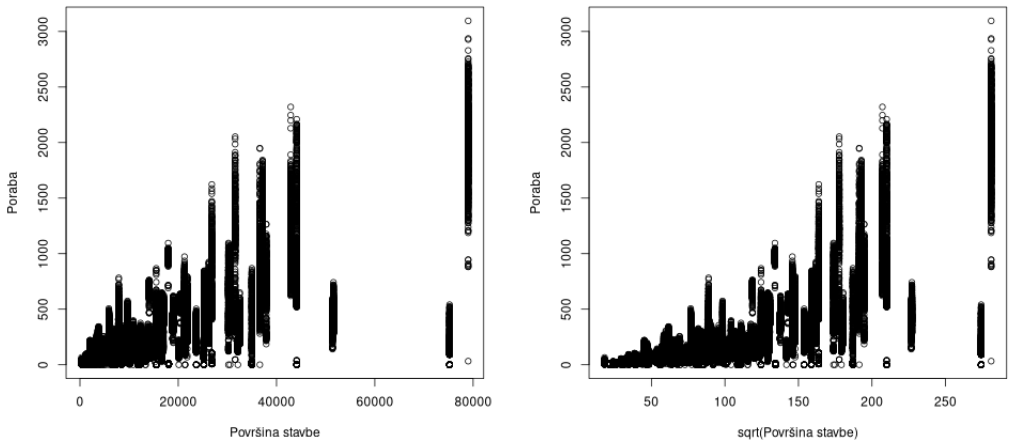
Na zgornjem grafu je prikazan delež stavb, ki pripadajo določeni namembnosti. Več kot polovica stavb je izobraževalnih, sledijo javno storitvene, delež kulturno razvedrilnih in poslovnih je približno enak, najmanj pa je stanovanjskih stavb.



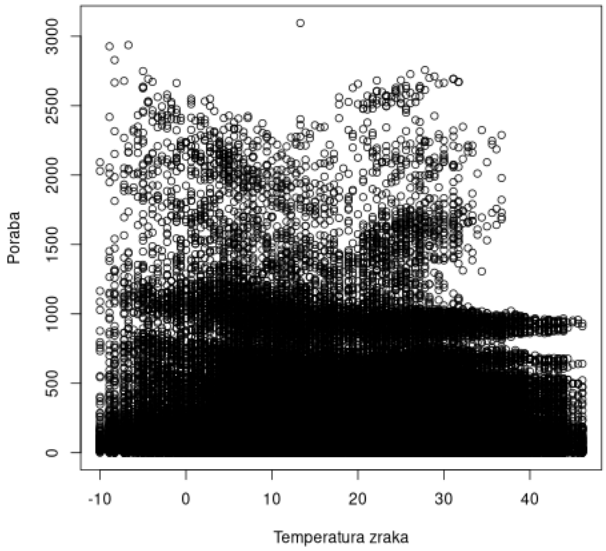
Na zgornjem grafu je prikazana povprečna poraba električne energije glede na namembnost stavbe. Vidimo lahko, da imajo najvišjo povprečno porabo poslovne stavbe. Javno storitvene, izobraževalne in stanovnjske stavbe imajo približno enako porabo, najmanjšo pa imajo kulturno razvedrilne.



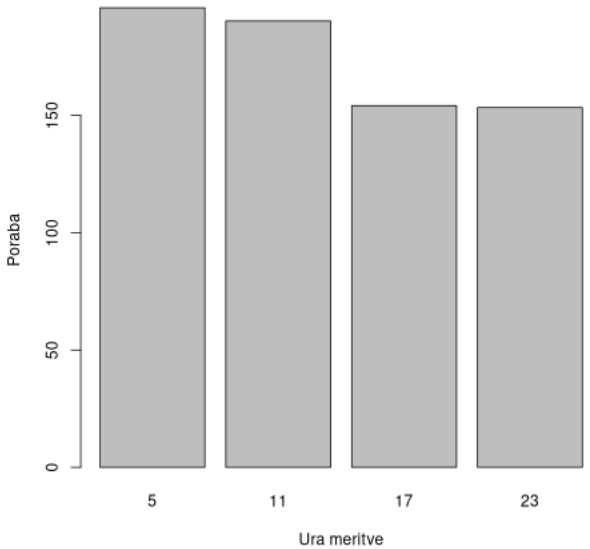
Na zgornjem grafu je prikazana povprečna površina stavbe za določeno namembnost. Vidimo lahko, da imajo največjo povprečno površino poslovne stavbe. Iz tega lahko sklepamo, da imajo zaradi tega tudi najvišjo povprečno porabo (prejšnji graf). Ostale stavbe imajo povprečno površino približno enako.



Na zgornjih dveh grafih lahko vidimo, da povprečna poraba narašča enakomerno s korenom površine. Na levem grafu so podatki narisani takšni kot so, na desnem pa je skala površine (x os) spremenjena, in sicer z uporabo kvadratnega korena.



Na zgornjem grafu lahko vidimo, da je poraba električne energije večja, kadar je temperatura zraka manjša od 10°C ali večja od 20°C. Iz tega lahko sklepamo, da je v primeru, ko je temperatura višja od 20°C v stavbah vklopljena klimatska naprava, v obratnem primeru pa je vklopljeno gretje prostorov.



Na zgornjem grafu je prikazana povprečna poraba električne energije glede na uro. Vidimo lahko, da je poraba višja ob 5. in 11. uri dopoldan in nižja ob 17. in 23. uri. Iz tega lahko sklepamo, da stavbe obratujejo ob 5. in 11. uri dopoldan, ob 17. in 23. uri pa so stavbe zaprte.

3 Vrednotenje atributov in kreacija novih

3.1 Obstoječi atributi

Med podanimi podatki sva imela podane atribute datum, ura, regija, stavba, namembnost, površina, leto izgradnje, temperatura zraka, temperatura rosišča, padavine, pritisk, smer vetra in hitrost vetra. Predvidava, da bosta na kreacijo klasifikacijskega modela najbolj vplivala atributa površina stavbe in temperatura zraka, najmanj pa bosta vplivala atributa smer vetra in hitrost vetra.

Predvidevava, da se bo težava pojavila pri ustvarjanju regresijskega modela, natančneje pri atributu padavine. Težava se pojavi, saj je v primeru pršenja vrednost atributa enaka -1, to pa pomeni, da je vrednost padavin večja od 0, a hkrati v atributu piše da ni.

3.2 Kreirani atributi

3.2.1 Vikend

Atribut vikend sva pridobila tako, da sva glede na datum določila, ali je bil dan sobota ali nedelja. To sva storila s pomočjo funkcije *is.weekend*, ki vrne *TRUE*, če je datum vikend, in *FALSE*, če datum ni vikend.

```
#install.packages("chron")
library(chron)
df$vikend <- is.weekend(df$datum)
summary(df)
```

3.2.2 Tekoče povprečje 7 dni

Atribut tekoče povprečje 7 dni sva pridobila tako, da sva za 7 dni nazaj pogledala porabo ob določeni uri za vsako stavbo posebej in izračunala povprečje.

```
print("Tekoce povprecje ma7")
n <- nrow(df)
min_search <- 1 # optimizacija
for (i in 1:nrow(df)) { #1:nrow(df)
    if (i %% as.integer(n/200) == 0) {
        print(sprintf("%.1f%%", (i/n)*100)) # proggress (traja nekaj minut)
    }
    window <- df[min_search:i,]
    # filtriraj po datumu (7 dni nazaj, brez trenutnega dneva)
    datum_sel <-
        window$datum >= df$datum[i]-7 &
        window$datum < df$datum[i]
    if (any(datum_sel)) min_search <- min_search + (min(which(datum_sel))-1)
    sel <-
        datum_sel &
```

```
        window$ura == df$ura[i] &
        window$stavba == df$stavba[i]
    # povprecje zadnjih 7 dni (kolikor pac je podatkov v tem obdobju)
    df$ma7[i] <- mean(window$poraba[sel])
    if (is.nan(df$ma7[i])) {
        df$ma7[i] <- 0
    }
}
print("ma7 done")
df[ df$ura == 23 & df$stavba == 1,]$ma7[1:21]
df[ df$ura == 23 & df$stavba == 1,]$poraba[1:21]
summary(df)
```

3.2.3 Letni čas

Atribut letni čas sva določila tako, da sva primerjala mesece meritev. Če je bil mesec meritve enak decembru, januarju ali februarju, je to pomenilo da je letni čas zima, če je bil mesec enak marcu, aprilu ali maju je bila to pomlad, junij, julij, avgust predstavljajo poletje, september, oktober in november pa predstavljajo jesen.

```
for (i in 1:nrow(df)) {
    mesec <- as.numeric(format(df$datum[i], "%m"))
    if (mesec == 12 || mesec == 1 || mesec == 2){
        df$letni_cas[i] <- "zima"
    } else if (mesec == 3 || mesec == 3 || mesec == 5){
        df$letni_cas[i] <- "pomlad"
    } else if (mesec == 6 || mesec == 7 || mesec == 8){
        df$letni_cas[i] <- "poletje"
    } else if (mesec == 9 || mesec == 10 || mesec == 11){
        df$letni_cas[i] <- "jesen"
    }
}
df$letni_cas <- as.factor(df$letni_cas)
summary(df)
```

3.3 Povzetek atributov

Po kreaciji vseh treh atributov sva pognala ukaz *summary(df)*, ki nama je vrnil rezultate, kot so prikazani spodaj. Vidimo lahko, da imamo za datum napisan najnižjo in najvišjo vrednost, mediano, srednjo vrednost ter prvi in tretji kvartil. Enake podatke dobimo za vse ostale atribute razen regije, stavbe, namembnosti, normalizirane porabe, vikend in letni čas.

Za regijo lahko vidimo, da je bilo za stavbe v vzhodni regiji opravljenih 93555 meritev, za zahodno pa 113230. Pri atributu stavba vidimo, da je za vsako stavbo bilo opravljenih 1348 meritev, pri namembnosti vidimo koliko stavb pripada določeni namembnosti, pri normalizirani porabi so meritve razdeljene glede na porabo. Pri atributu vikend vidimo, da je bilo 147311 meritev opravljenih med tednom, 59474 pa med vikendom. Vidimo pa lahko tudi, da je bilo 52221 meritev opravljenih jeseni, 42405 poleti, 29523 pomladi in 82636 pozimi.

```
> summary(df)
     datum                  ura            regija           stavba
 Min.   :2015-12-31   Min.   : 5.00   vzhodna: 93555   1      :  1348
 1st Qu.:2016-03-09   1st Qu.: 5.00   zahodna:113230   2      :  1348
 Median :2016-07-02   Median :11.00                    3      :  1348
 Mean   :2016-06-29   Mean   :13.98                    4      :  1348
 3rd Qu.:2016-10-16   3rd Qu.:23.00                    5      :  1348
 Max.   :2016-12-31   Max.   :23.00                    6      :  1348
                                                       (Other):198697
                namembnost        povrsina       leto_izgradnje
 izobrazevalna       :105293   Min.   :  329.3   Min.   :1900
 javno_storitvena    : 27219   1st Qu.: 3445.1   1st Qu.:1949
 kulturno_razvedrilna: 29293   Median : 6619.1   Median :1968
 poslovna            : 26228   Mean   :10575.4   Mean   :1968
 stanovanjska        : 18752   3rd Qu.:12733.8   3rd Qu.:1993
                               Max.   :79000.4   Max.   :2017

  temp_zraka      temp_rosisca       oblacnost        padavine
 Min.   :-10.00   Min.   :-22.800   Min.   :0.000   Min.   :-1.0000
 1st Qu.: 10.00   1st Qu.: -2.800   1st Qu.:0.000   1st Qu.: 0.0000
 Median : 19.40   Median :  2.800   Median :4.000   Median : 0.0000
 Mean   : 18.91   Mean   :  3.877   Mean   :3.397   Mean   : 0.2634
 3rd Qu.: 27.80   3rd Qu.: 10.600   3rd Qu.:6.000   3rd Qu.: 0.0000
 Max.   : 46.10   Max.   : 25.000   Max.   :9.000   Max.   :56.0000

   pritisk         smer_vetra    hitrost_vetra        poraba
 Min.   : 991.9   Min.   :  0.0   Min.   : 0.000   Min.   :   0.00
 1st Qu.:1009.8   1st Qu.: 90.0   1st Qu.: 2.100   1st Qu.:  38.22
 Median :1014.1   Median :170.0   Median : 3.100   Median :  92.70
 Mean   :1015.1   Mean   :170.3   Mean   : 3.432   Mean   : 173.57
 3rd Qu.:1019.9   3rd Qu.:270.0   3rd Qu.: 4.600   3rd Qu.: 185.61
 Max.   :1040.9   Max.   :360.0   Max.   :14.900   Max.   :3095.44

    norm_poraba      vikend             ma7            letni_cas
 NIZKA     :48335   Mode :logical   Min.   :   0.00   jesen  :52221
 SREDNJA   :76435   FALSE:147311    1st Qu.:  38.29   poletje:42405
 VISOKA    :38660   TRUE :59474     Median :  92.24   pomlad :29523
 ZELONIZKA :15945                   Mean   : 171.07   zima   :82636
 ZELOVISOKA:27410                   3rd Qu.: 182.20
                                    Max.   :2658.73
```

Pognala sva tudi funkcijo *attrEval*, ki nama je atribute ocenila po pomembnosti, nato pa sva jih po pomembnosti padajoče uredila in izpisala.

```
> library(CORElearn)
> sort(attrEval(norm_poraba ~ ., df, "InfGain"), decreasing = TRUE)
       stavba         poraba            ma7 leto_izgradnje     namembnost
 1.0122433832   0.1353764256   0.1075041557   0.0576120974   0.0548670491
     povrsina            ura   temp_rosisca         vikend         regija
 0.0240757138   0.0235681488   0.0111544847   0.0065996882   0.0059742970
    letni_cas          datum     temp_zraka        pritisk     smer_vetra
 0.0051871838   0.0036369087   0.0026924803   0.0017944931   0.0014230198
    oblacnost  hitrost_vetra       padavine
 0.0005750612   0.0001852129   0.0001569170
```

4 Modeliranje

Za modeliranje poznamo več različnih algoritmov. Nekatere izmed njih sva preizkusila in izmed njih kasneje izbrala tri, ki so se nama zdeli najučinkovitejši.

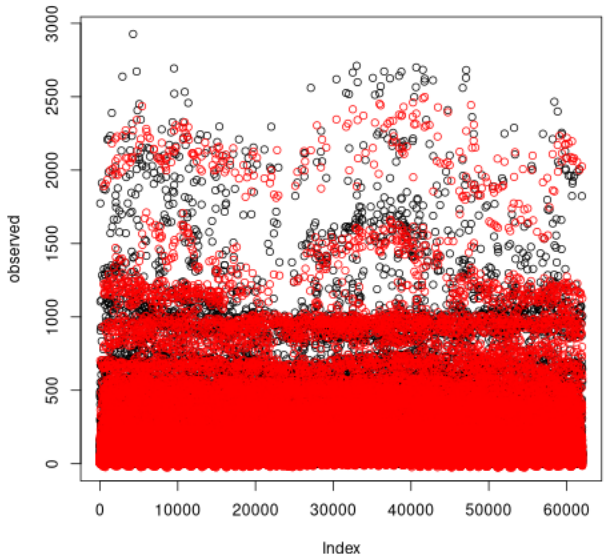
Preizkusila sva algoritem linearne regresije, regresijskega drevesa, naključnega gozdu, svm, k-najbližjih sosedov in nevronske mreže. Pri vsakem izmed njih sva, če je bil algoritem dovolj hiter, izpisala relativno srednjo kvadratno napako (rmse). Glede na rmse vrednost, sva nato algoritme primerjala med seboj in izmed njih izbrala tri, ki so imeli rmse vrednost najnižjo. To so bili linearna regresija, regresijsko drevo in nevronska mreža.

Modeliranja sva se lotila tako, da sva najprej napisala štiri funkcije za izračun napak. Napisala sva funkcije za izračun srednje absolutne napake (mae), srednje kvadratne napake (mse), relativne srednje absolutne napake (rmae) in relativne srednje kvadratne napake (rmse).

4.1 Regresija

4.1.1 Linearna regresija

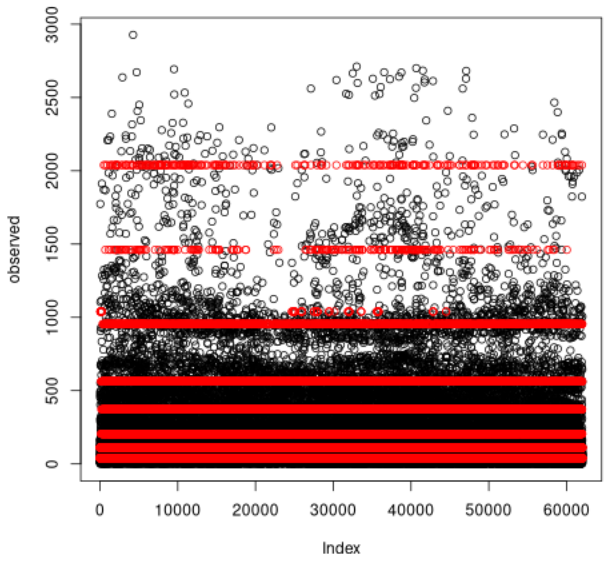
Za izračun linearne regresije sva uporabila funkcijo *lm*, ki sva ji podala uteži, ki jih upošteva pri računanju, ter podatke, ki so shranjeni v spremenljivki *train*. Po tem sva v spremenljivko *predicted* napovedala rezultate s pomočjo prej izračunane linearne regresije in to narisala na graf, ki je prikazan spodaj. S črno so narisani izmerjeni podatki, z rdečo pa so narisani predvideni rezultati.



Vidimo lahko, da se napovedani rezultati dokaj lepo prilegajo izmerjenim. Za test sva izračunala tudi rmse vrednost, ki v tem primeru znaša 0.0612.

4.1.2 Regresijsko drevo

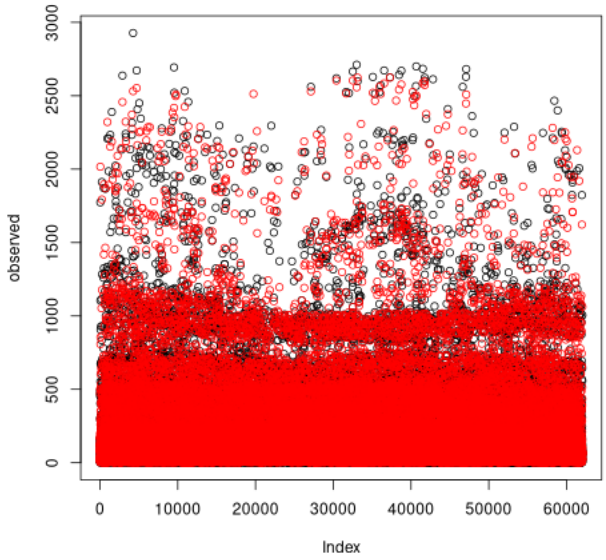
S pomočjo knjižnice *rpart* sva izračunala regresijsko drevo. Uporabila sva funkcijo *rpart* in ji podala uteži, ki jih upošteva pri izračunu in učne podatke, shranjene v spremenljivki *train*. Enako kot pri linearni regresiji sva v spremenljivko *predicted* shranila predvidene vrednosti in jih narisala na spodnji graf. Tudi na tem grafu so s črno barvo označeni izmerjeni podatki, z rdečo pa predvideni.



V primerjavi z linearno regresijo vidimo, da je ujemanje napovedanih in izmerjenih podatkov v regresijskem drevesu manjše, torej je rmse vrednost za regresijsko drevo večja. Rmse vrednost za regresijsko drevo je 0.0799.

4.1.3 K-najbližjih sosedov

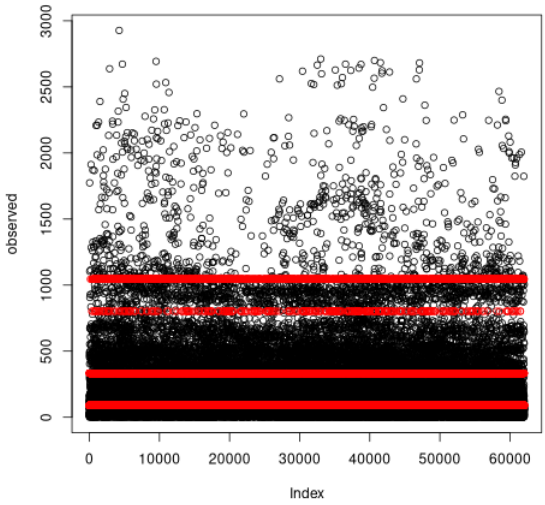
Za izračun k-najbližjih sosedov sva uporabila knjižnico *kknn* in njeno funkcijo *kknn*. Funkcijo sva optimizirala tako, da sva jo pognala večkrat, z različnimi vrednostmi k. Primerjala sva vrednosti rmse in izbrala tako vrednost k, kjer je bil rmse najnižji. V spremenljivko *predicted* sva shranila napovedane rezultate in jih kasneje, skupaj z izmerjenimi podatki izrisala na grafu.



Vidimo lahko, da se je do sedaj izmerjenim podatkom približal algoritem k-najbližjih sosedov. Ker je ujemanje največje, je tudi rmse vrednost najnižja, v tem primeru enaka 0.0387.

4.1.4 Nevronska mreža

S pomočjo knjižnice *nnet* sva izračunala nevronsko mrežo. Najprej sva *seed* nastavila na 0, nato pa s pomočjo funkcije *nnet* izračunala mrežo. Izmed atributov sva morala izločiti stavbo zaradi prevelikega števila atributov, s tem pa se je natančnost nevronske mreže zmanjašala toliko, da je mreža za ta problem neuporabna. Enako kot pri prejšnjih sva napovedala podatke in jih izrisala na grafu.



Vidimo lahko, da je nevronska mreža v primeru napovedovanja porabe neuporabna, saj se niti ne približa izmerjenim vrednostim, prav tako je rmse vrednost previsoka (0.2962).

4.2 Klasifikacija

Klasifikacije sva se lotila tako, da sva na začetku definirala nekaj funkcij, ki jih bova uporabila v naslednjih delih kode. Definirala sva funkcije *CA*, *brier.score* in *inf.score*.

4.2.1 Odločitveno drevo

S pomočjo knjižnice *CORElearn* in njeno funkcijo *CoreModel* sva izračunala odločitveno drevo. Funkciji *CoreLearn* sva podala parameter *model="tree"* in ji s tem povedala, da želiva odločitveno drevo. V spremenljivko *predicted* sva shranila napovedane rezultate in jih nato primerjala z izmerjenimi. S pomočjo funkcije *CA* sva izmerila natančnost drevesa in ugotovila, da je bila napoved v 83.4% pravilna.
Izmerila sva tudi brierjevo mero, ki v tem primeru znaša 0.25. Informacijska vsebina odgovora pa je enaka 1.64.

4.2.2 Naivni bayes

Izmed algoritmov za klasifikacijo sva izbrala tudi naivnega bayesa. Izračunala sva ga s pomočjo funkcije *CoreLearn* in nato napovedala rezultate. Natančnost napovedi v tem primeru je bila enaka 61.73%, brierjeva mera pa 0.497. Informacijska vsebina odgovora pri naivnem bayesu je bila enaka 1.017.

4.2.3 K-najbližjih sosedov

Za napovedovanje z algoritmom k-najbližjih sosedov sva uporabila funkcijo *CoreLearn*. Za vrednost k sva vzela število 5, saj je z večjim k računanje trajalo predolgo. Po napovedi sva rezultate primerjala z izmerjenimi in ugotovila, da je bila natančnost 59.5%, brierjeva mera pa je bila 0.549. Informacijska vsebina odgovora je 0.957.

4.2.4 Nevronska mreža

Nevronsko mrežo sva izračunala s funkcijo *nnet*. Ugotovila sva, da bi s skaliranjem atributov lahko pridobila boljše rezultate. Brez skaliranja je bila natančnost enaka 74.79%. Nato sva podatke skalirala na intervalu od 0 do 1. Po ponovnem izračunu nevronske mreže, se je natančnost izboljšala za 4%. Pri obeh sva iz atributov izločila atribut stavba, saj je z njim vse skupaj preveč zamudno.

5 Izbira atributov

Uporabila sva *wrapper.r*, ki iz atributov izbere tiste, ki so za modeliranje najpomembnejši.

Program deluje tako, da začne s funkcijo, ki prejme formulo, set podatkov, funkcijo za treniranje, funkcijo za napovedovanje in funkcijo za evalvacijo. Nato pripraviva podatke za obdelavo in se sprehodiva čez vse atribute. Nato za vsak atribut z dodatnimi preveri, kateri set atributov vrne najboljše rezultate. Dobljene rmse vrednosti primerja in v spremenljivko *local.best* shrani set atributov, ki ima 1 - CA vrednost najnižjo. V terminal nakoncu izpiše predvideno najboljšo 1 - CA vrednost in set atributov, ki je za napovedovanje najboljši.

```
> wrapper(norm_poraba ~ ., train, trainfn_tree, predictfn, evalfn, cvfolds=10)
selected attribute:  stavba
selected attribute:  ma7
selected attribute:  vikend
[1]    28131 killed     R
# cvfolds sva morala zmanjšati, saj je z vrednostjo 10 potrebovala preveč RAM-a

> wrapper(norm_poraba ~ ., train, trainfn_tree, predictfn, evalfn, cvfolds=2)
best model: estimated error =  0.1741497 ,
selected feature subset =  norm_poraba ~ stavba + ma7 + vikend + ura + temp_zraka
+ povrsina + leto_izgradnje + temp_rosisca

> wrapper(norm_poraba ~ ., train, trainfn_bayes, predictfn, evalfn, cvfolds=10)
best model: estimated error =  0.3318365 , selected feature subset =  norm_poraba
~ stavba + ura + vikend + letni_cas + smer_vetra + padavine

> wrapper(norm_poraba ~ ., train, trainfn_bayes, predictfnprob, evalbrier, cvfolds=10)
best model: estimated error =  0.4820053 , selected feature subset =  norm_poraba
~ stavba + ma7 + ura + vikend + povrsina + temp_rosisca + temp_zraka + oblacnost
+ namembnost + letni_cas + smer_vetra
```

Po tem, sva vsako učenje pognala ponovno, z atributi, ki nama jih je vrnila funkcija *wrapper*. Dobljene rezultate sva primerjala s prejšnimi in ugotovila naslednje. V primeru odločitvenega drevesa, razlika med prvim učenjem in drugim, z optimalnimi atributi, ni bila velika (0.1%). Največja razlika je bila pri naivnem bayesu, kjer se je CA vrednost spremenila za kar 5.1%. Poizkusila sva bayesa še izboljšati z optimizacijo brierja, vendar razlike v končnem rezultatu ni bilo.

Za zanimivost sva poizkusila ročno določiti optimalne atribute. V primeru odločitvenega drevesa sva dosegla malenkost nižjo CA vrednost, pri naivnem bayesu pa 3% nižjo končno CA vrednost.

6 Kombiniranje modelov

V tem delu, sva poizkusila kombinirati različne modele strojnega učenja. Poizkusila sva kombinirati s pomočjo uteženega glasovanja in glasovanja.

6.1 Uteženo glasovanje

Napisala sva funkcije *voting_train*, vrne model z naučenimi algoritmi za napovedovanje, *voting_predict*, ki vrne rezultat uteženega glasovanja treh učnih algoritmov (nevronska mreža, odločitveno drevo, naivni bayes) in pa *voting_prob*, ki vrne matrico verjetnosti uteženega glasovanja.

Nato sva agregirala podatke tako, da sva pognala vse tri funkcije in primerjala CA vrednost s CA vrednostjo odločitvenega drevesa (CA(UG): 0.8344348, CA(OD): 0.8342092), saj je odločitveno drevo najboljše. Opazila sva, da je CA vrednost pri uteženem glasovanju malenkost boljša kot pri odločitvenem drevesu, birerjeva mera pa je celo slabša, zato iz tega sklepava, da se kombiniranje v tem primeru ne izplača, saj so razlike minimalne.

6.2 Glasovanje

Napisala sva funkcijo *voting_simple_predict*, ki stori enako kot funkcija pri uteženem glasovanju, le da v tem primeru kombinira algoritme na način prepostega glasovanja.

Ponovno sva rezultate primerjala z rezultati odločitvenega drevesa in v tem primeru je bila CA vrednost slabša (CA(OD): 0.8342092, CA(G): 0.8192824) od odločitvenega drevesa in zato se tudi glasovanje ne izplača.

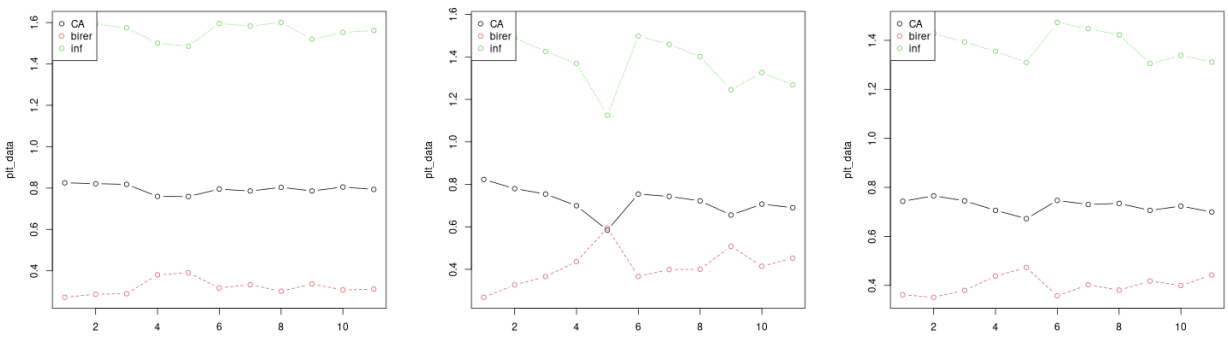
7 Evalvacija modelov

Kot zadnji del seminarske naloge sva uspešnosti učnih množic primerjala, najprej glede na mesece, nato pa še po regiji.

7.1 Meseci

Vse podatke sva razdelila na 12 podmnožic, glede na mesec zajema. Nato sva januarske podatke porabila za učenje modela, ki sva ga preizkusila na februarskih podatkih. Naslednji korak je bil, da sva januarske in februarske podatke porabila za učenje modela, katerega sva kasneje preizkusila na marčevskih podatkih. Tako sva nadaljevala, dokler nisva to storila za vse mesece.

Spodaj so prikazani grafi uspešnosti algoritma pri testiranju s podatki naslednjega meseca (od leve proti desni: odločitveno drevo, naivni bayes, k-najbližjih sosedov):



7.2 Regije

Podatke sva razdelila v tri učne množice (vse, vzhod, zahod). Nato sva iz vsake množice naredila svoj model in ga testirala s testno množico, ki je enaka za vse.

To sva testirala z uporabo odločitvenega drevesa in naivnega bayesa in dobila naslednje CA vrednosti:

```
Odločitveno drevo:
    CA general: 0.836127409891031
    CA vzhod: 0.514733380617706
    CA zahod: 0.60609968405442

Naivni bayes:
    CA general: 0.78252949900058
    CA vzhod: 0.474369720807273
    CA zahod: 0.572570765362048
```

Ugotovila sva, da so rezultati pri obeh algoritmih najboljši, če so v učni množici podatki iz obeh regij. Opazila sva tudi, da je rezultat boljši, kadar model učimo s podatki iz zahodne regije in ga testiramo na obeh regijah, kot če so v učni množici podatki iz vzhoda.

8 Zaključek

S to seminarsko nalogo sva predstavila različne algoritme za učenje modelov za napovedovanje. Ugotovila sva, da so včasih preprosti algoritmi zadostni za problem, ki ga rešujemo, še posebej v primerih kot je ta, ko za kompleksnejše algoritme (primer: nevronska mreža) nimamo zadostne strojne opreme za hiter in učinkovit izračun.