3. zadaća – Inteligentni sustavi 2

1. Izrada boosting modela i predikcija

```
#podjela indeksa 60 %-40%
indeks_treniranje = sample(1:nrow(Auto), round(0.6 * nrow(Auto)))
#stvaranje skupa za treniranje
Auto_treniranje = Auto [indeks_treniranje,]
#stvaranje skupa za testiranje
Auto_testiranje = Auto [-indeks_treniranje,]
```

Slika 1: Podjela skupa

Prvi korak je import potrebnih biblioteka (*ISLR, gbm*) i *„attach"* dataseta *„Auto"*.
Za izradu indeksa korištena je funkcija *„sample*", u kojoj je potrebno navesti odakle biramo indekse te koliko primjeraka želimo (u ovom slučaju 60%). Zatim slijedi izrada skupa za testiranje koji sadrži 60 % opservacija dataseta *„Auto"* i skupa za testiranje koji sadrži preostale indekse navedenog skupa.

```
#izrada boosting modela gbm
Auto_model1 = gbm(mpg ~ cylinders+weight+year+displacement+acceleration,
                  distribution = "gaussian", data = Auto_treniranje,
                  n.trees = 5000, interaction.depth = 4)
```

Slika 2: Izrada gbm modela

Na slici 2. vidljiv je kod za izradu gbm (Gradient Boosting Machine) modela. Za izradu modela korištena je funckija *„gbm"* koja sadrži slijedeće parametre: varijablu prema kojoj se vrši predikcija (*mpg*) te varijable na temelju kojih će model vršiti predikciju *„mpg"* varijable (*cylinders, weight, year, displacement* i *acceleration*), distribuciju (u ovom slučaju to je Gaussova distribucija s obzirom da radimo sa kontinuiranim varijablama), podaci (skup za treniranje – *Auto_treniranje*), broj stabala (u ovom slučaju 5000) te dubina svakog stabla (4).

```
#predikcija prvog modela
predikcija1 = predict.gbm(object = Auto_model1, newdata = Auto_testiranje,
                          n.trees = 5000)
#izračun točnosti tj.stope pogreške
mean((predikcija1-Auto_testiranje$mpg)^2)
```

Slika 3: Predikcija gbm modela

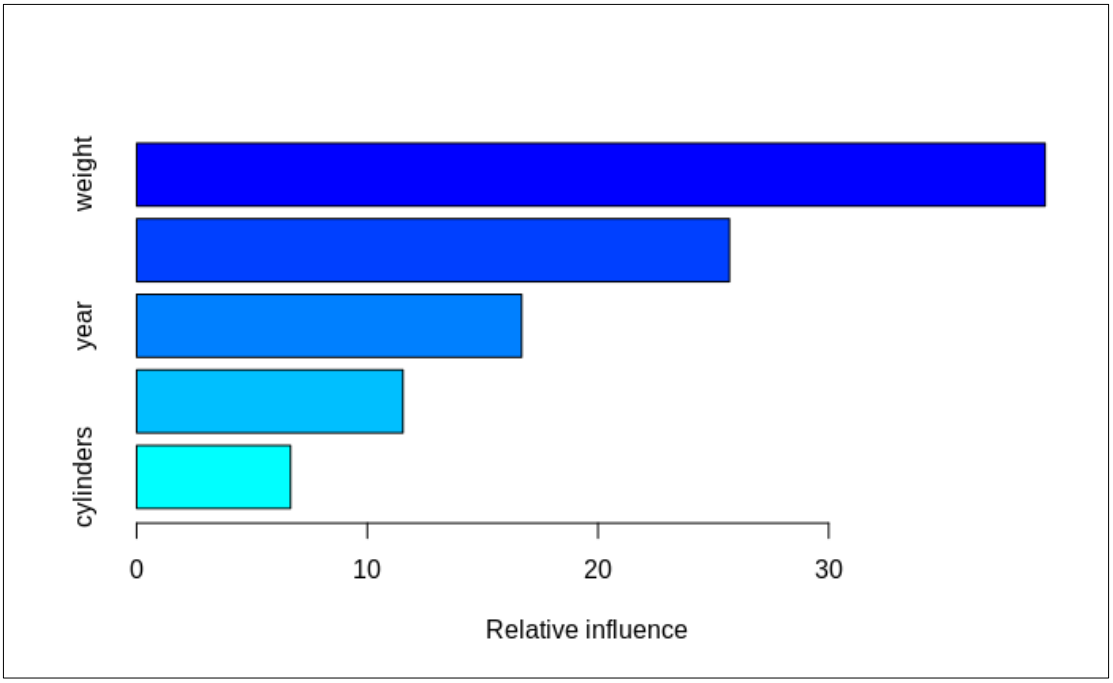
Za predikciju korištena je funkcija *„predict.gbm"* koja uzima parametre *„object"* (gbm model pomoću kojeg će se vršiti predikcija), *„newdata"* (podaci za testiranje) te *„n.trees"* (u ovom slučaju 5000). Izračun stope pogreške modela možemo izračunati pomoću funkcije *„mean"* (auc se ne može koristiti u ovom slučaju s obzirom da se radi o kontinuiranim varijablama – auc vrši izračun točnosti u slučaju da je prediktor binarnog tipa).
Pokretanjem ove funkcije dobijemo rezultat od 12.21792 koji predstavlja stopu pogreške modela (MSE – *mean squared error*), što znači da je točnost modela oko 87.79%.

2. Relativni utjecaj

```
> #graf relativne ovisnosti
> summary(Auto_model1)
                       var   rel.inf
weight              weight 33.250692
displacement displacement 30.894201
year                 year 16.278732
acceleration acceleration 11.908167
cylinders       cylinders  7.668207
```

Slika 4: Relativni utjecaj

Pomoću funkcije *„summary"* možemo vidjeti graf i statistiku relativnog utjecaja varijabli. Prema navedenoj statistici vidimo da su varijable *„weight" i „displacement"* daleko najznačajnije varijable relativnog utjecaja od 33.25 i 30.89. Na slici 5. prikazan je grafikon navedene statistike (relativnog utjecaja varijabli).



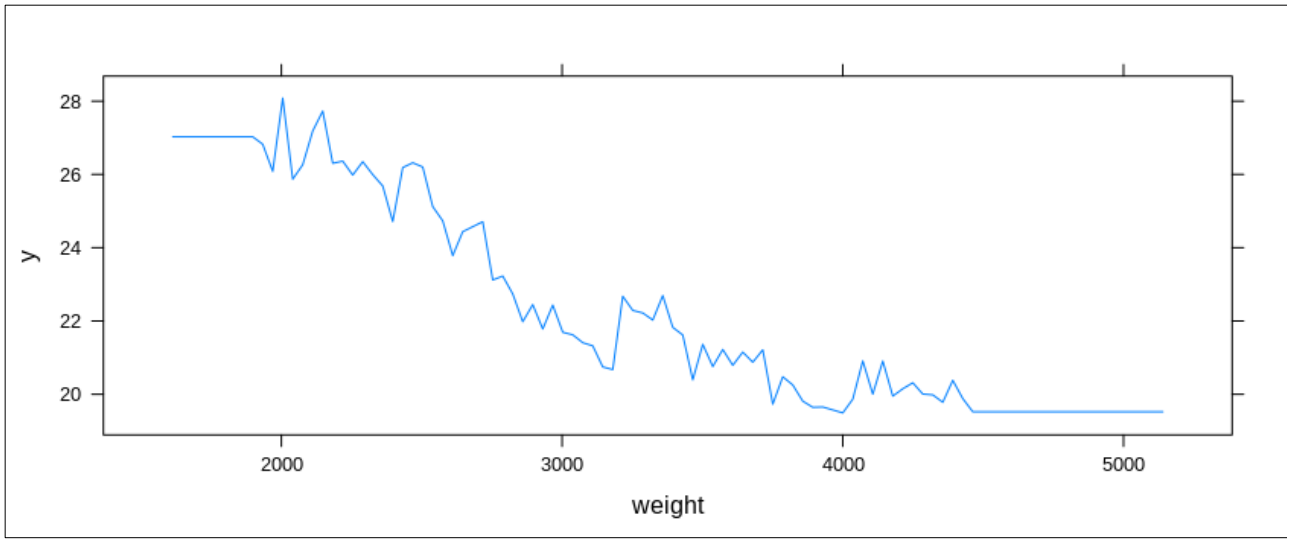
Slika 5: Graf relativnog utjecaja varijabli

3. Graf parcijalne ovisnosti

```
plot(Auto_model1, i="weight")
plot(Auto_model1, i="displacement")
```

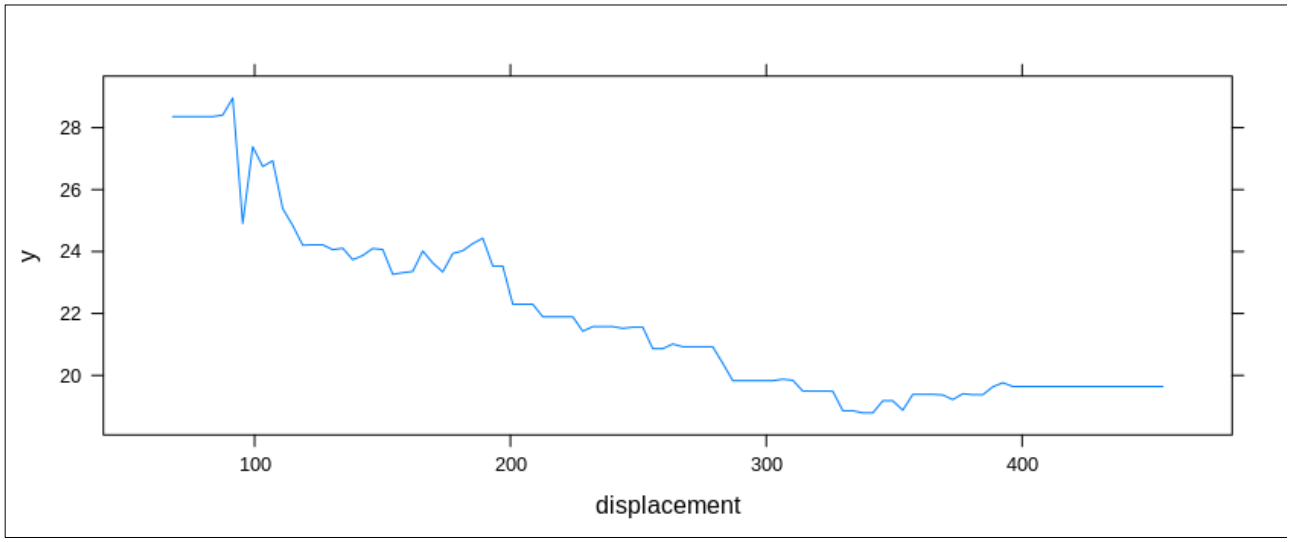
Slika 6: Kod za izradu grafova parcijalne ovisnosti

Za izradu grafova parcijalne ovisnosti izabrane su dvije varijable najvećeg relativnog utjecaja (weight i displacement). Za izradu grafova korištena je funkcija *„plot”* koja za argumente uzima model i varijablu čiji utjecaj želimo prikazati na grafikonu putem argumenta *„i”*.



Slika 7: Graf parcijalne ovisnosti za weight varijablu

Iz grafikona parcijalne vrijednosti *„weight"* varijable na slici 7. vidimo da što je auto teže, to može prijeći manje milja po galonu (što je i logično s obzirom da težina auta utječe na vožnju istog).



Slika 8: Parcijalna ovisnost za varijablu displacement

Iz grafikona parcijalne ovisnosti *„displacement"* varijable na slici 8. vidljivo je da je ukupan volumen cilindara u motoru auta (displacement) puno veći kod auta koji prijeđu manju udaljenost po galonu.

4. Poboljšani (boosted) model i predikcija

```
#izrada poboljšanog modela
opt_auto <- gbm.perf(object = Auto_model1,
                     method = "OOB",
                     oobag.curve = TRUE)
opt_auto
```

Slika 9: Kod za otkrivanje optimalnog broja stabla

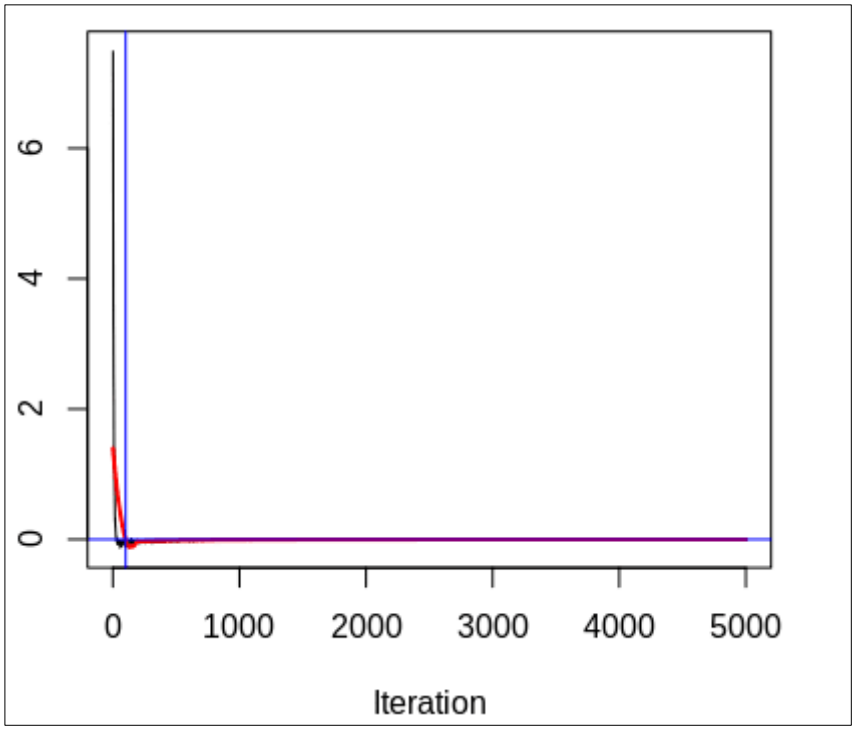
Za ugađanje gbm modela izabrala sam metodu ranog zaustavljanja. Za procjenjivanje optimalnog broja boosting iteracija korištena je funkcija *„gbm.perf"* koja za parametre uzima objekt (izvorni model), metodu kojom se određuje optimalan broj iteracija (u ovom slučaju to je *„out-of-bag*) te parametar za crtanje krivulje. Pokretanjem ovog dijela koda dobijemo slijedeći rezultat:

```
> opt_auto
[1] 102
attr(,"smoother")
Call:
loess(formula = object$oobag.improve ~ x, enp.target = min(max(4,
    length(x)/10), 50))

Number of Observations: 5000
Equivalent Number of Parameters: 39.99
Residual Standard Error: 0.16
```

Slika 10: Optimalan broj iteracija

Sa slike 10. vidimo da je optimalan broj stabala (iteracija) koji bi se trebali koristiti kako bi greška validacije ostala najmanja iznosi 102.



Slika 11: Graf funkcije gbm.perf

Na slici 11. prikazan je graf dobiven pokretanjem funkcije *„gbm.perf”.* Vertikalna plava linija nam pokazuje koliko iteracija je potrebno da bi se postigli optimalni rezultati (102).

```
Auto_model2 = gbm(mpg ~ cylinders+weight+year+displacement+acceleration,
                  distribution = "gaussian", data = Auto_treniranje,
                  n.trees = 102, interaction.depth = 4)
predikcija2 = predict.gbm(object = Auto_model2, newdata = Auto_testiranje,
                          n.trees = 102)
mean((predikcija2-Auto_testiranje$mpg)^2)
```

Slika 12: Drugi model sa optimalnim brojem iteracija

Na slici 12. prikazan je kod za izradu drugog modela sa optimalnim brojem iteracija dobivenim u prošlom bloku koda. Funkcija sadrži sve iste parametre kao i kod izrade prvog modela uz izuzetak promjene broja stabala koji je postavljen na 102. Zatim vršimo predikciju modela pomoću iste funkcije *„predict.gbm”.* Pokretanjem funckcije *„mean”* dobijemo rezultat od 12.13081 što je manja pogreška testiranja (MSE) od drugog modela (12.13 < 12.22). Razlika nije značajna s obzirom da se radi o malom skupu podataka.