Vaja 6: Vrednotenje modela

`R` paketi, ki jih bomo uporabili na vajah:

```
library(ISLR2) # datasets
library(ggplot2) # nice plots (ggplot)
library(knitr)  # for markdown
library(leaps) # best subset
library(mgcv) # gam
```

Vrednotenje napovednega modela je ključno za ocenitev natančnosti napovedi v populaciji, za katero je model namenjen. Korektno vrednotenje je pomembno, saj lahko slab napovedni model v praksi naredi precej škode.

Vrednotenje napovednega modela poteka v naslednjih fazah:

1. **Notranje vrednotenje** (*internal validation*) poteka še v fazi razvijanja modela in model vrednoti na podatkih, ki izhajajo iz istega podatkovnega vira kot podatki, na katerih je bil model zgrajen. Model lahko ovrednotimo:

   - tako, da podatke po principu slučajnosti razdelimo na učni in testni del, pri čemer se model, razvit na učnem delu podatkov, ovrednoti na testnem delu. Čeprav se zdi, da je v primeru naključne razdelitve vzorca na dva dela, testni del podatkov povsem neodvisen, temu ni tako, saj oba izhajata iz istega podatkovnega vira. Problematičnost tega pristopa je, da ustvarimo dva manjša podatkovna seta, kar je še posebej problematično v primerih, ko so vzorci majhni. To ima za posledico nestabilnost modela, kar vodi v večjo variabilnost in manj natančne napovedi.

   - metode ponovnega vzorčenja (K-kratno navzkrižno preverjanje, bootstrap), ki namesto specifičnega modela ovrednotijo sam postopek gradnje modela. Prednost teh pristopov je v tem, da se za vrednotenje uporabi vse podatke, ki so na voljo v fazi razvijanja modela.

2. **Zunanje vrednotenje** (*external validation*) je proces vrednotenja obstoječega napovednega modela na novih podatkih, pridobljenih na isti populaciji.

Napovedne modele vrednotimo na podlagi različnih kriterijev za ovrednotenje napovedne kakovosti modela. V prejšnji vaji smo imeli na voljo le učni del podatkov za gradnjo modela.

Podatke smo razdelili na učni in testni del po principu slučajnosti:

```
data("Hitters")
#str(Hitters)

Hitters <- na.omit(Hitters)

set.seed(123)
train <- sample(1:nrow(Hitters), round(2/3*nrow(Hitters)), replace=F)

train_set <- Hitters[train, ]
test_set <- Hitters[-train, ]
```

Odzivno spremenljivko smo logaritmirali:

```
train_set$Salary <- log(train_set$Salary)
test_set$Salary <- log(test_set$Salary)
```

Za primerjavo bomo modelom iz prejšnje vaje dodali še aditivni model, ki predstavlja razširitev linearnega modela, v katerem je odzivna spremenljivka v linearni odvisnosti od gladkih funkcij napovednih spremenljivk (npr. zlepki). Aditivni model številske spremenljivke modelira kot prilagodljive, gladke funkcije, ki jih lahko definiramo na podlagi zlepkov ali kakšnih drugih baznih funkcij. Da bi se izognili preprileganju takega modela, penalizacijski člen v modelu kaznuje pomanjkanje gladkosti oz. preprileganje (*wiggliness*), kar zmanjša efektivno število stopinj prostosti, ki jih porabi posamezna številska spremenljivka v modelu.

Optimalno stopnjo glajenja oz. penalizacije lahko določimo s pomočjo kriterijev za izbiro modela. Model načeloma predpostavlja aditivnost učinkov, a vanj lahko vključimo tudi interakcije z opisnimi napovednimi spremenljivkami.

Aditivni model lahko naredimo s funkcijo `gam` iz paketa `mgcv`. Gladke funkcije (v našem primeru bomo uporabili kubične zlepke) bomo v modelu dodali za tiste številske spremenljivke, pri katerih smo z grafi parcialnih ostankov detektirali nelinearnost. Nastavitev argumenta `bs='cs'` avtomatično izvede tudi izbiro modela: pred vsak nelinearni člen je dodan dodaten penalizacijski člen, ki lahko pomen posamezne spremenljivke v modelu skrči na 0 (kar pomeni, da so efektivne stopinje prostosti enake 0 in je spremenljivka odstranjena iz modela). Izbiro modela bi lahko naredili tudi z uporabo drugih gladkih funcij, pri čemer bi morali argument `select` nastaviti na `TRUE`.

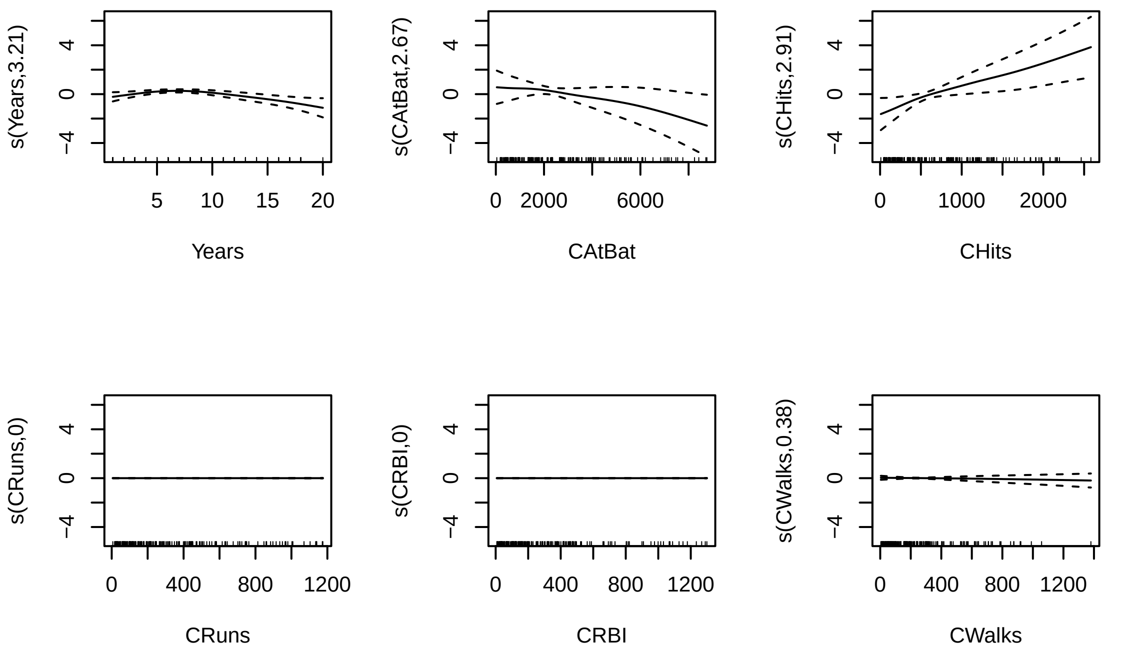
```r
gam_mod <- gam(Salary ~ AtBat + Hits + HmRun + Runs + RBI + Walks +
                 s(Years, bs = "cs") + s(CAtBat, bs = "cs") +
                 s(CHits, bs = "cs") + CHmRun +
                 s(CRuns, bs = "cs") + s(CRBI, bs = "cs") +
                 s(CWalks, bs = "cs") + League + Division + PutOuts +
                 Assists + Errors + NewLeague,
               method="REML", data=train_set)

summary(gam_mod)
```

```
Family: gaussian
Link function: identity

Formula:
Salary ~ AtBat + Hits + HmRun + Runs + RBI + Walks + s(Years,
    bs = "cs") + s(CAtBat, bs = "cs") + s(CHits, bs = "cs") +
    CHmRun + s(CRuns, bs = "cs") + s(CRBI, bs = "cs") + s(CWalks,
    bs = "cs") + League + Division + PutOuts + Assists + Errors +
    NewLeague

Parametric coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.6133334  0.1661477  33.785   <2e-16 ***
AtBat       -0.0008248  0.0011762  -0.701   0.4842
Hits        -0.0033848  0.0044169  -0.766   0.4447
HmRun       -0.0018330  0.0099910  -0.183   0.8547
Runs         0.0095901  0.0047102   2.036   0.0435 *
RBI          0.0022382  0.0042247   0.530   0.5970
Walks        0.0028906  0.0030666   0.943   0.3474
CHmRun       0.0028452  0.0011069   2.570   0.0111 *
LeagueN      0.0416654  0.1655141   0.252   0.8016
DivisionW   -0.0947078  0.0736967  -1.285   0.2007
PutOuts      0.0001838  0.0001384   1.328   0.1863
Assists      0.0006702  0.0004389   1.527   0.1289
Errors      -0.0118960  0.0088158  -1.349   0.1792
NewLeagueN   0.1358364  0.1647907   0.824   0.4111
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                edf Ref.df     F  p-value
s(Years)  3.208e+00      9 2.564 1.51e-05 ***
```

```
s(CAtBat) 2.667e+00      9 0.824 0.002885 **
s(CHits)  2.908e+00      9 1.258 0.000283 ***
s(CRuns)  1.739e-04      9 0.000 0.893474
s(CRBI)   6.101e-05      9 0.000 0.481242
s(CWalks) 3.816e-01      9 0.056 0.232193
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =   0.75   Deviance explained = 78.2%
-REML = 179.48  Scale est. = 0.20676   n = 175
```

Funkcija `plot.gam` omogoča vizualizacijo nelinearne povezanosti napovednih spremenljivk, ki smo jih modelirali z gladkimi funkcijami.

```r
par(mfrow=c(2,3))
plot(gam_mod)
```



Slika 1: Modeliranje nelinearnosti v kontekstu GAM za model `gam_mod`.

Nelinearna je zveza z `log(Salary)` in spremenljivkami `Years`, `CatBat` in `CHits`, med tem ko sta spremenljivki `CRuns` in `CRBI` odstranjeni iz modela.

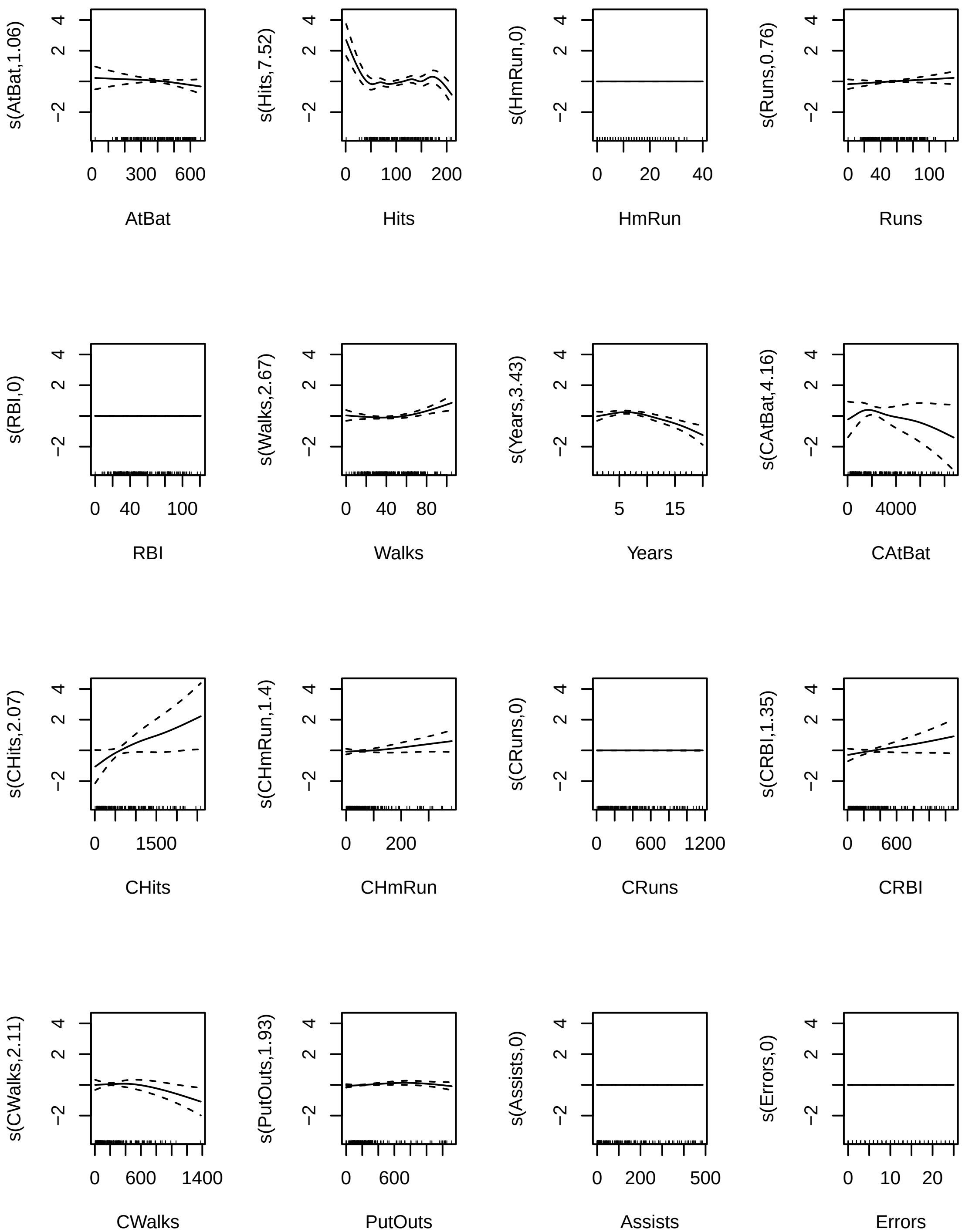
Za primerjavo naredimo še aditivni model, kjer popolnoma avtomatiziramo izbiro spremenljivk v model.

```r
gam_mod_2 <- gam(Salary ~ s(AtBat, bs = "cs") + s(Hits, bs = "cs") +
                 s(HmRun, bs = "cs") + s(Runs, bs = "cs") +
                 s(RBI, bs = "cs") + s(Walks, bs = "cs") +
                 s(Years, bs = "cs") + s(CAtBat, bs = "cs") +
                 s(CHits, bs = "cs") + s(CHmRun, bs= "cs") +
                 s(CRuns, bs = "cs") + s(CRBI, bs = "cs") +
```

```r
                 s(CWalks, bs = "cs") + League + Division +
                 s(PutOuts, bs = "cs") + s(Assists, bs = "cs") +
                 s(Errors, bs = "cs") + NewLeague,
               method="REML", data=train_set)

summary(gam_mod_2)
```

```
Family: gaussian
Link function: identity

Formula:
Salary ~ s(AtBat, bs = "cs") + s(Hits, bs = "cs") + s(HmRun,
    bs = "cs") + s(Runs, bs = "cs") + s(RBI, bs = "cs") + s(Walks,
    bs = "cs") + s(Years, bs = "cs") + s(CAtBat, bs = "cs") +
    s(CHits, bs = "cs") + s(CHmRun, bs = "cs") + s(CRuns, bs = "cs") +
    s(CRBI, bs = "cs") + s(CWalks, bs = "cs") + League + Division +
    s(PutOuts, bs = "cs") + s(Assists, bs = "cs") + s(Errors,
    bs = "cs") + NewLeague

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.91211    0.04679 126.344   <2e-16 ***
LeagueN      0.03774    0.12625   0.299    0.765
DivisionW   -0.08392    0.05641  -1.488    0.139
NewLeagueN   0.08896    0.12476   0.713    0.477
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                 edf Ref.df     F  p-value
s(AtBat)   1.057e+00      9 0.258  0.07769 .
s(Hits)    7.519e+00      9 8.723  < 2e-16 ***
s(HmRun)   1.282e-05      9 0.000  0.46399
s(Runs)    7.603e-01      9 0.176  0.09811 .
s(RBI)     7.758e-06      9 0.000  0.54304
s(Walks)   2.669e+00      9 1.948 4.37e-05 ***
s(Years)   3.428e+00      9 3.446 5.60e-07 ***
s(CAtBat)  4.164e+00      9 1.529 6.77e-05 ***
s(CHits)   2.074e+00      9 0.598  0.00786 **
s(CHmRun)  1.400e+00      9 0.394  0.00881 **
s(CRuns)   2.181e-04      9 0.000  0.43209
s(CRBI)    1.352e+00      9 0.329  0.01671 *
s(CWalks)  2.114e+00      9 1.118  0.00188 **
s(PutOuts) 1.935e+00      9 0.550  0.05677 .
s(Assists) 5.581e-06      9 0.000  0.49955
s(Errors)  7.243e-06      9 0.000  0.32968
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.862   Deviance explained = 88.7%
-REML =  109.8  Scale est. = 0.11415   n = 175
```

```r
par(mfrow=c(4,4))
plot(gam_mod_2)
```



Slika 2: Modeliranje nelinearnosti v kontekstu GAM za model `gam_mod_2`.

Dani model poleg spremenljivk `Years`, `CatBat` in `CHits` upošteva nelinearnost zveze z `log(Salary)` še pri spremenljivkah `Hits`, `Walks`, `CHmRun`, `CRBI`, `CWalks` in `PutOuts`. Za modeliranje zveze s `Hits` se npr. porabi kar 7,5 stoping prostosti! Vidimo pa, da je zveza nelinearna predvsem v repih - vprašanje je, če se dani model morda ne preprilega.

Primerjajmo oba modela na podlagi AIC:

```
AIC(gam_mod, gam_mod_2)
```

```
                df      AIC
gam_mod   26.77278 249.4950
gam_mod_2 40.27519 161.4735
```

Drugi model je glede na AIC kriterij precej boljši.

Za vrednotenje napovedne kakovosti modelov moramo najprej izračunati napovedi za enote v testnem vzorcu.

```
m0 <- lm(Salary~., data=train_set)

best_subset = regsubsets(Salary ~. , data = train_set, nvmax = 19)

bwd_sel = regsubsets(Salary ~., data = train_set, nvmax = 19, method = "backward")

fwd_sel = regsubsets(Salary ~., data = train_set, nvmax = 19, method = "forward")
```

Uporabili bomo funkcijo s prejšnje vaje, ki vrne napovedi:

```
predict.regsubsets <- function(object, newdata, id){
  form = as.formula(object$call[[2]]) # formula modela
  mat = model.matrix(form, newdata) #modelska matrika
  coefi = coef(object, id=id) #ocenjeni parametri modela
  xvars = names(coefi)
  mat[,xvars]%*%coefi
}
```

```
preds_m0 <- predict(m0, newdata = test_set)

preds_best_subset_cp <- predict(best_subset, newdata = test_set, id = 7)
#izbira najboljše podmnožice s Cp kriterijem je dala model s 7 spremenljivkami

preds_best_subset_adjr2 <- predict(best_subset, newdata = test_set, id = 11)
#izbira najboljše podmnožice s AdjR2 kriterijem je dala model z 11 spremenljivkami

preds_bwd_sel <- predict(bwd_sel, newdata = test_set, id = 6)
#izbira nazaj je dala model s 6 spremenljivkami

preds_fwd_sel <- predict(fwd_sel, newdata = test_set, id = 7)
#izbira naprej je dala model s 7 spremenljivkami

preds_cv5 <- predict(best_subset, newdata = test_set, id = 2)
#izbira najboljše podmnožice s CV MSE kriterijem je dala model z 2 spremenljivkama

#napovedi na podlagi gam:
preds_gam <- predict(gam_mod, test_set)

#napovedi na podlagi gam_2:
preds_gam_2 <- predict(gam_mod_2, test_set)
```

```r
preds <- list(preds_m0, preds_best_subset_cp, preds_best_subset_adjr2,
              preds_bwd_sel, preds_fwd_sel, preds_cv5, preds_gam, preds_gam_2)

modeli <- c("Polni model", "Best subset (Cp)", "Best subset (Adj R2)",
            "Izbira nazaj (Cp)", "Izbira naprej", "Best subset (5x CV)", "Moj GAM", "Auto GAM")
```

V regresiji imamo na voljo dejanske vrednosti odzivne spremenljivke, ki jih lahko med sabo primerjamo. Tako lahko linearni napovedni model vrednotimo na podlagi naslednjih kriterijev (*performance measures*):

1. srednja abolutna napaka napovedi:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|;$$

2. povprečna kvadratna napaka napovedi:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2;$$

   oz. povprečna napaka napovedi:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2};$$

3. koeficient determinacije:

$$R^2 = 1 - \frac{SS_{residuals}}{SS_{yy}};$$

4. kalibriranosti modela, tj. ujemanjem med dejanskimi vrednostmi in napovedmi. Kalibracija se ocenjuje grafično tako, da na x-osi prikažemo napovedi, na y-os pa dejanske vrednosti, čemur dodamo še gladko kalibracijsko krivuljo. Numerično pa kalibracijo modela ocenimo s:

   - splošno kalibracijo oz. presečiščem kalibracije (*calibration-in-the-large*, idealna vrednost = 0): ocenjuje povprečno (splošno) kalibracijo in kvantificira morebitno sistematično precenjevanje ali podcenjevanje napovedi, tako da primerja povprečje napovedi na testnih podatkih s povprečjem dejanskih vrednosti. V primeru, da je povprečje napovedi večje od povprečja dejanskih vrednosti, model na splošno precenjuje napovedi, v nasprotnem primeru pa model podcenjuje napovedi.

   - naklonom kalibracije (*calibration slope*, idealna vrednost = 1): je kar ocenjeni naklon b v modelu, ki ocenjuje odvisnost dejanskih vrednosti od napovedi, dobljenih na testnem vzorcu:

$$Y_{test} = a + b\hat{Y}.$$

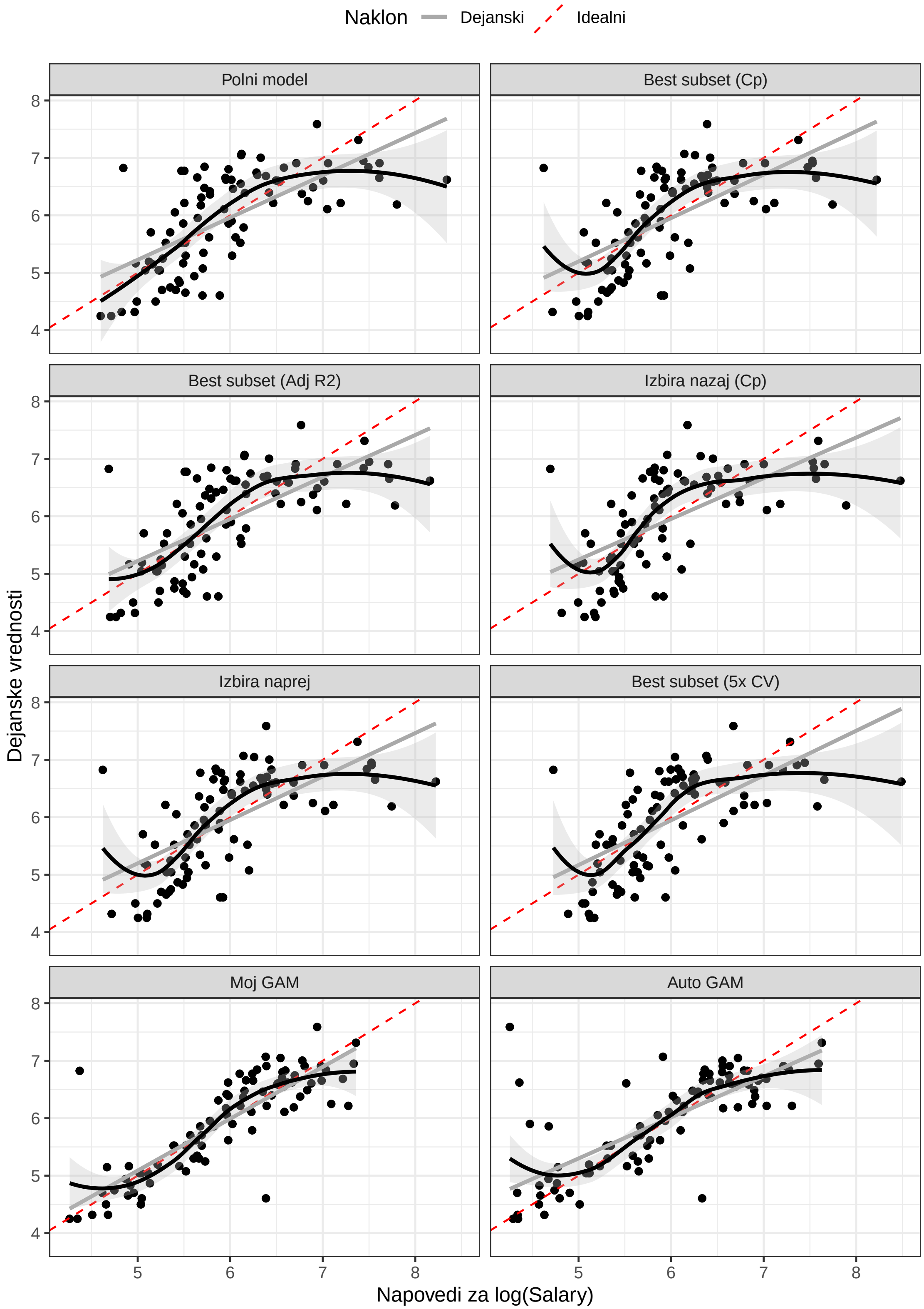
   Kalibracijski naklon < 1 nakazuje na preprileganje modela učnim podatkom. Preprileganje nastane, kadar model zajame preveč naključnega šuma v podatkih (ima torej slabo sposobnost generalizacije) in je prekompleksen glede na razpoložljivo količino podatkov (npr. preveliko število napovednih spremenljivk, izbira napovednih spremenljivk na podlagi statistične značilnosti, uporaba zelo fleksibilnih algoritmov). Na splošno je za preprileganje značilno, da so ocenjene napovedi preveč ekstremne (prenizke za nizke dejanske vrednosti in previsoke za visoke dejanske vrednosti). Nasprotno kalibracijski naklon > 1 nakazuje na podprileganje modela učnim podatkom, torej bo variacijski razmik ocenjenih napovedi preozek.

> Kadar ocenjujemo kalibracijo modela, moramo vedno upoštevati tako naklon kalibracijske kot splošno kalibracijo, saj kalibracijski naklon, ki je blizu 1 sam po sebi še ne pomeni dobre kalibriranosti modela na testnem setu podatkov.

Modele bomo med seboj najprej primerjali grafično.

```
#iz lista v dolgi format
preds_long <- data.frame(dejanske=rep(test_set$Salary, length(modeli)),
                         napovedi=unlist(preds),
                         method=rep(modeli, each=nrow(test_set)))

preds_long$method <- factor(preds_long$method, levels=modeli)
```

```
ggplot(preds_long, aes(y = dejanske, x = napovedi)) +
  geom_point() +
  theme_bw() +
  geom_abline(aes(colour="Idealni", slope=1, intercept=0), linetype = "dashed") +
  geom_smooth(method = "lm", se = FALSE, aes(colour = "Dejanski")) +
  geom_smooth(method = "loess", color = "black", se = TRUE, fill = "gray", alpha = 0.3) +
  scale_colour_manual(name="Naklon", values=c("darkgrey", "red")) +
  ylab("Dejanske vrednosti") +
  xlab("Napovedi za log(Salary)") +
  facet_wrap(~method, ncol=2) +
  theme(legend.position = "top")
```

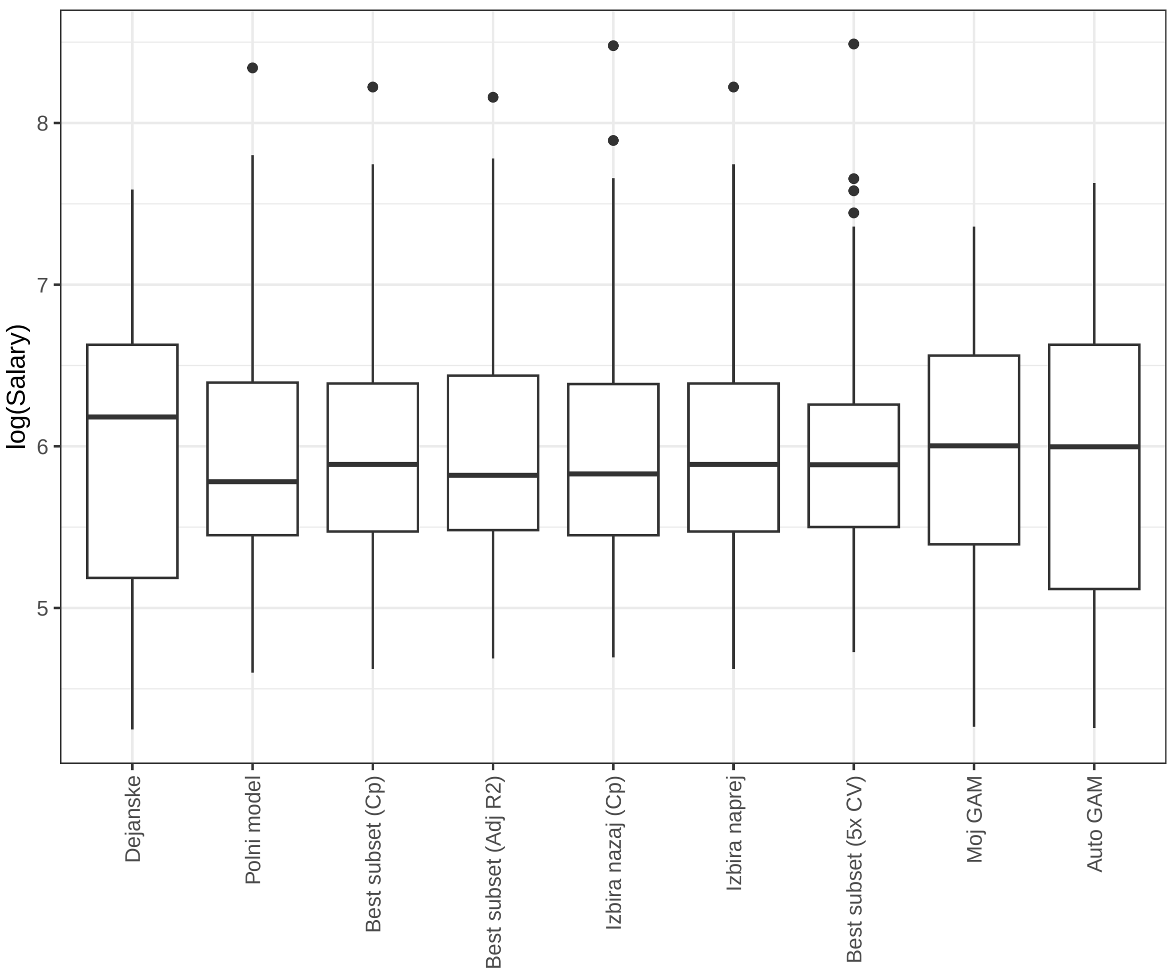


Slika 3: Kalibracijski naklon za različne napovedne modele.

Poglejmo še porazdelitev napovedi za posamezne modele:

```r
#iz lista v dolgi format
preds_long_2 <- data.frame(preds=c(test_set$Salary, unlist(preds)),
                           method=c(rep("Dejanske", nrow(test_set)),
                                    rep(modeli, each=nrow(test_set))))

preds_long_2$method <- factor(preds_long_2$method, levels=c("Dejanske", modeli))
```

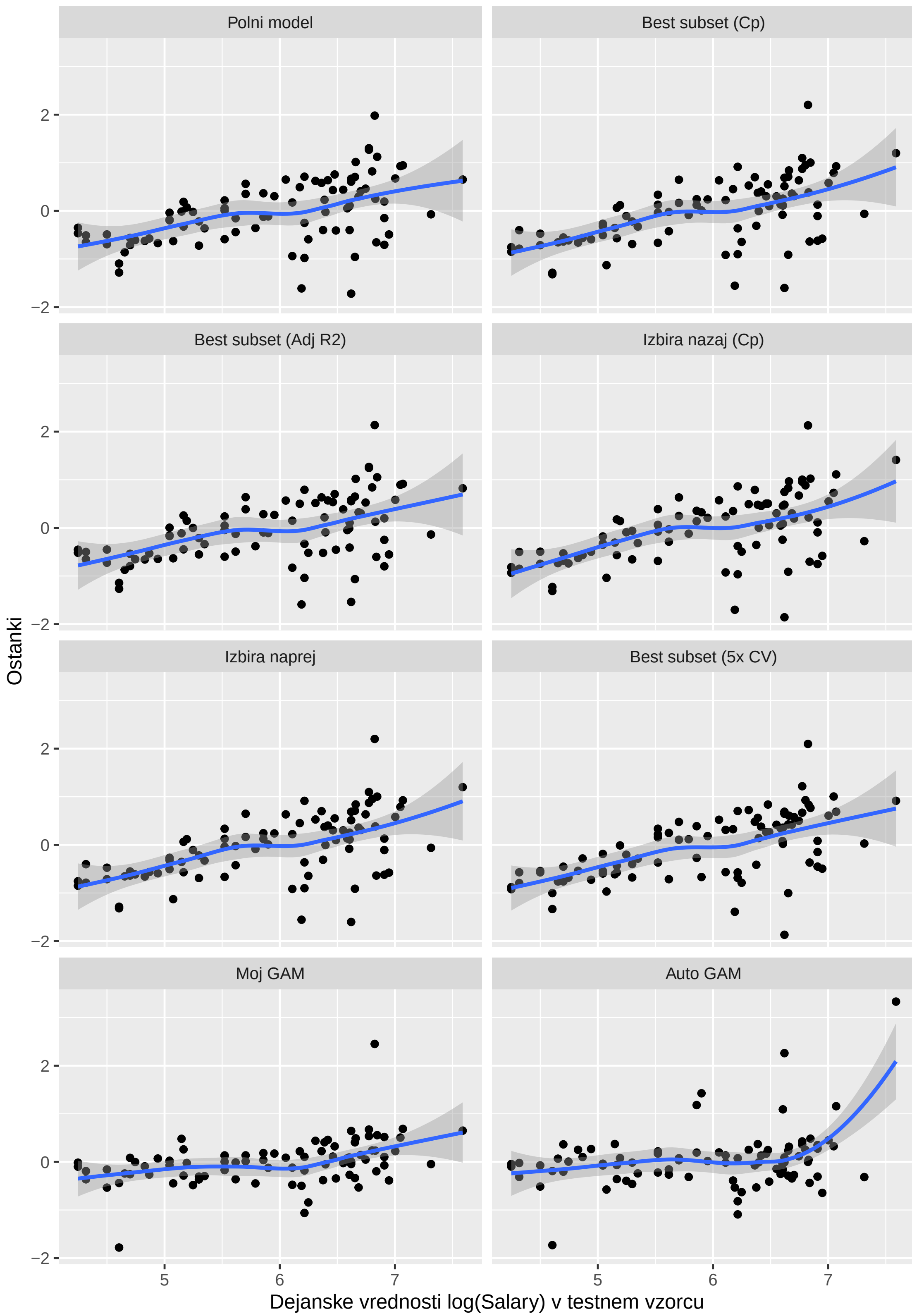
```r
ggplot(preds_long_2, aes(x = method, y = preds)) +
  geom_boxplot() +
  theme_bw() +
  ylab("log(Salary)") +
  xlab("") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



Slika 4: Porazdelitev dejanskih vrednosti `log(Salary)` na testnem vzorcu in napovedi za `log(Salary)` za različne napovedne modele.

Grafično si poglejmo še pristranskost napovedi, tj. porazdelitev ostankov glede na dejanske vrednosti, ter lokalno napako napovedi, tj. vrednosti absolutnih ostankov glede na dejanske vrednosti odzivne spremenljivke.

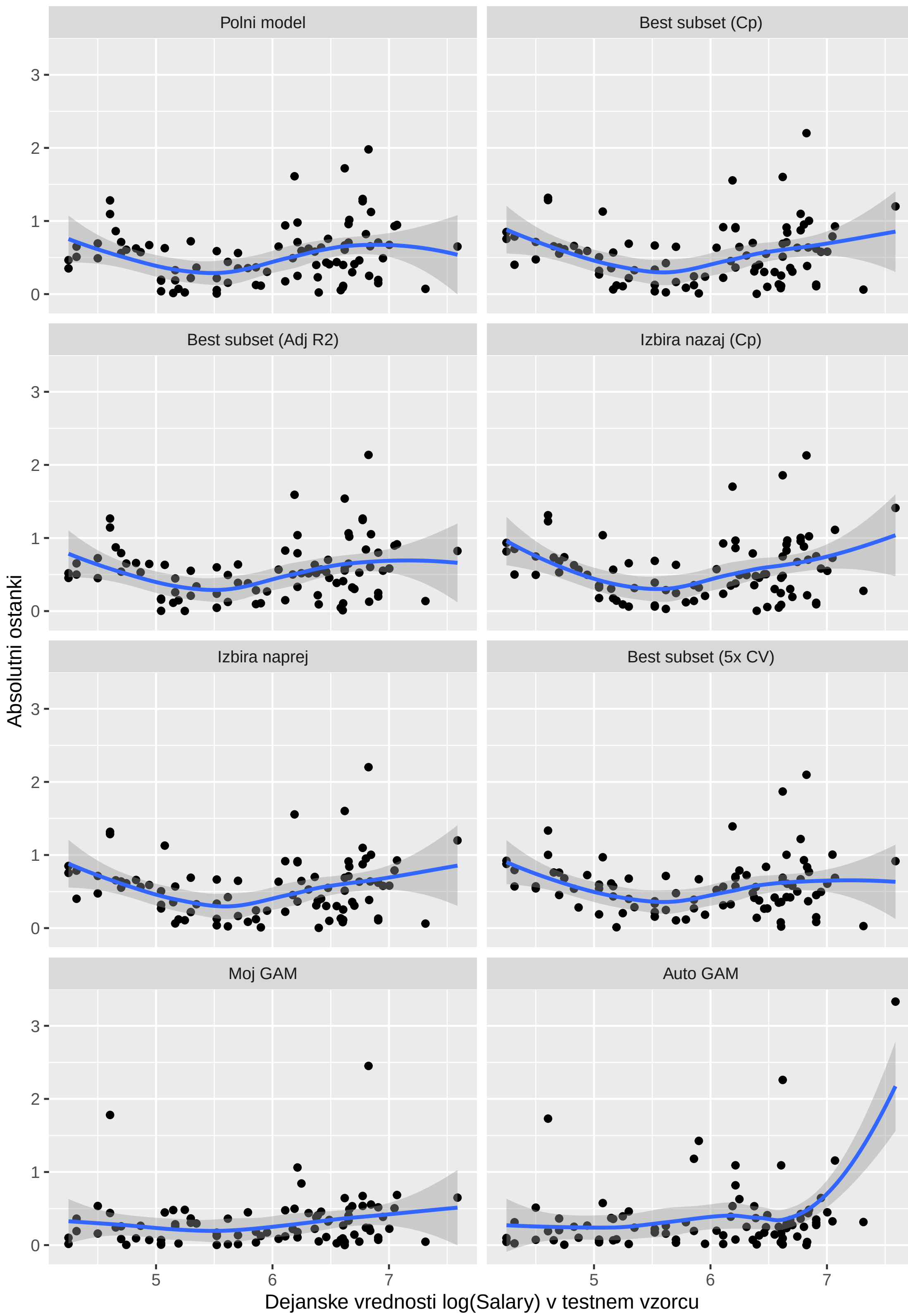
```r
ggplot(preds_long, aes(x=dejanske, y=dejanske-napovedi)) +
  geom_point() +
  facet_wrap( ~ method, scale = "fixed", nrow=4) +
  geom_smooth(method="loess") +
  ylab("Ostanki")+
  xlab("Dejanske vrednosti log(Salary) v testnem vzorcu")
```



Slika 5: Lokalna pristranskost napovedi za različne napovedne modele.

Graf ostakov v odvisnosti od dejanskih vrednosti v testnem delu podatkov kaže na to, da modeli nepristransko ocenjujejo plače za igralce s povprečno `log(Salary)`, medtem ko za igralce pod povrečjem `log(Salary)` precenijo, za igralce nad povprečjem pa podcenijo. Izjema je model `Moj GAM`, pri katerem je pristranskost majhna po celem razponu dejanskih vrednosti. Podobno lahko opazimo tudi, če prikažemo absolutno napako napovedi v odvisnosti od dejanskih vrednosti. Napaka napovedi za celotni razpon dejanskih vrednosti `log(Salary)` je izrazito najmanjša pri modelu `Moj GAM`, kjer smo modelirali nelinearnost.

```
ggplot(preds_long, aes(x=dejanske, y=abs(dejanske-napovedi))) +
  geom_point() +
  facet_wrap( ~ method, scale = "fixed", nrow=4) +
  geom_smooth(method="loess") +
  ylab("Absolutni ostanki")+
  xlab("Dejanske vrednosti log(Salary) v testnem vzorcu")
```



Slika 6: Lokalna napaka napovedi za različne napovedne modele.

Primerjajmo modele še numerično:

```
primerjava.modelov <- matrix(NA, length(modeli), 4)
colnames(primerjava.modelov) <- c("Splošna kalibracija", "Naklon kalibracije", "R2", "RMSE")
rownames(primerjava.modelov) <- modeli

for(i in 1:length(preds)){

  primerjava.modelov[i, "Splošna kalibracija"] <- mean(test_set$Salary - preds[[i]], na.rm=TRUE)
  # coef(lm(Salary~offset(preds[[i]]), data=test_set))
  primerjava.modelov[i, "Naklon kalibracije"] <- coef(lm(test_set$Salary ~ preds[[i]]))[2]
  primerjava.modelov[i, "R2"] <- cor(test_set$Salary, preds[[i]], use="complete.obs")^2
  primerjava.modelov[i, "RMSE"] <- sqrt(mean((test_set$Salary - preds[[i]])^2, na.rm=TRUE))

}

kable(primerjava.modelov, digits = 3, caption = "Kakovost posameznega napovednega modela.")
```

Tabela 1: Kakovost posameznega napovednega modela.

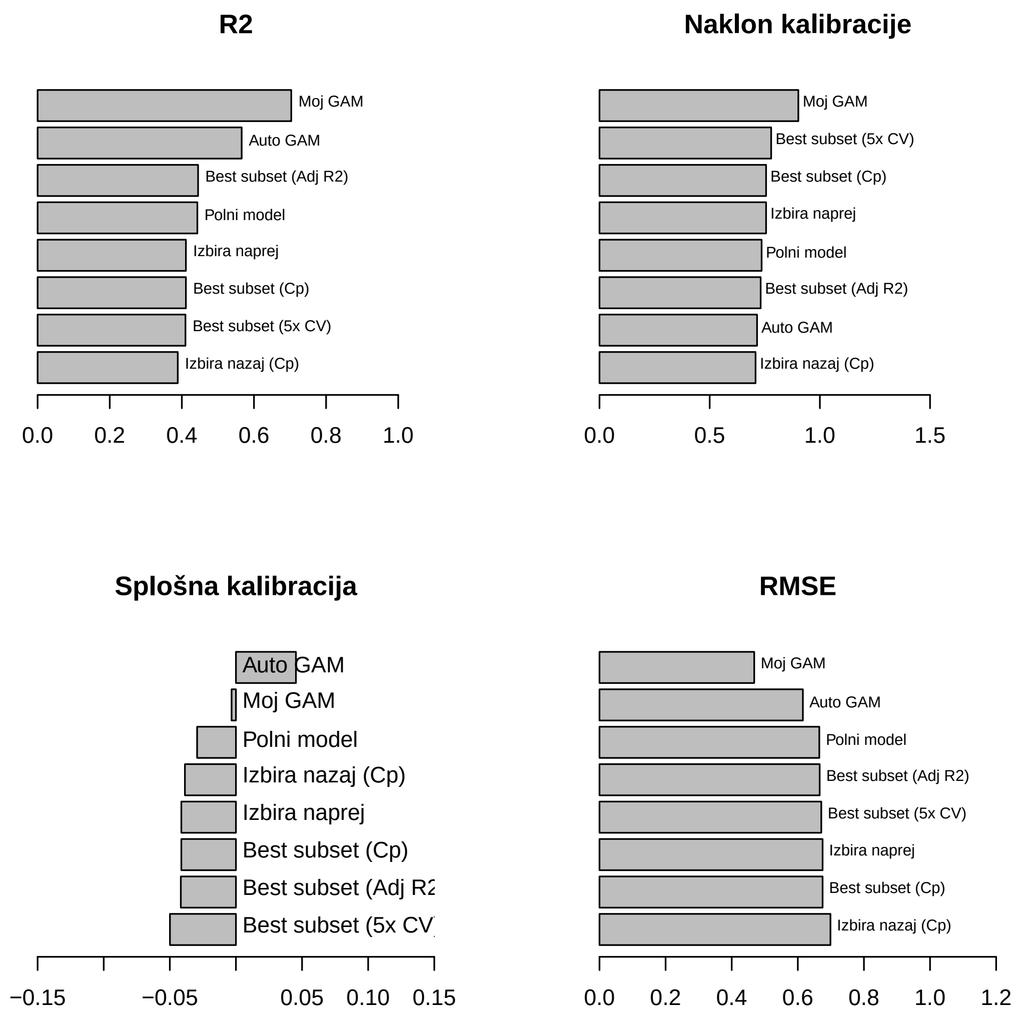
	Splošna kalibracija	Naklon kalibracije	R2	RMSE
Polni model	-0.029	0.736	0.443	0.665
Best subset (Cp)	-0.041	0.756	0.411	0.675
Best subset (Adj R2)	-0.042	0.731	0.445	0.666
Izbira nazaj (Cp)	-0.039	0.708	0.389	0.699
Izbira naprej	-0.041	0.756	0.411	0.675
Best subset (5x CV)	-0.050	0.779	0.410	0.671
Moj GAM	-0.003	0.902	0.703	0.468
Auto GAM	0.045	0.715	0.566	0.615

Z ozirom na vse kriterije je zmagovalec model `Moj GAM`, kar kaže na to, da lahko z ustreznim modeliranjem nelinearnosti napovedno moč modela izrazito izboljšamo, a moramo biti previdni, da model hkrati ni preveč fleksibilen (tako kot `Auto GAM`).

```
#funkcija za grafični prikaz rezultatov
plotcomp <- function(kriterij="R2", main=kriterij, xrange=c(0,1),
                     legendx=NULL, fun=function(x) x, negative=FALSE){

  sta <- fun(primerjava.modelov[,kriterij])
  sta <- sta[order(sta)]
  if(negative) sta <- sta[order(-sta)]
  barplot(sta, horiz=TRUE, xlim=xrange, ylab="", axes=F, axisnames=F)
  axis(1)
  if(is.null(legendx)) for(i in 1:length(sta)) text(sta[i]+0.02, y=i*1.2-0.4,
                                                    names(sta)[i], adj=0, cex=0.7)
  else for(i in 1:length(sta)) text(legendx, y=i*1.2-0.4, names(sta)[i], adj=0)
  title(main=main)

}
```

```
par(mfrow=c(2,2))

plotcomp(kriterij="R2", xrange=c(0, 1.1))
```

```r
plotcomp(kriterij="Naklon kalibracije", xrange=c(0,1.8))

plotcomp(kriterij="Splošna kalibracija", xrange=c(-0.15, 0.15), legendx=0.005)

plotcomp(kriterij="RMSE", negative=TRUE, xrange=c(0, 1.2))
```



Slika 7: Ocenjena kakovost napovednih modelov glede na različne kriterije.

Notranje vrednotenje na podlagi bootstrapa

V nadaljevanju bomo pokazali, kako bi modele ovrednotili na podlagi bootstrapa. Študije so pokazale, da je ta strategija v praksi boljša, saj imamo za gradnjo modela na voljo vse podatke. Bootstrap ovrednoti strategijo modeliranja, tako da oceni optimizem, tj. v kolikšni meri je vrednotenje modela na učnih podatkih preoptimistično. Ko smo ocenili optimitem, lahko izračunane vrednosti kriterijev na učnih podatkih popravimo, tako da od njih odštejemo optimizem.

Koraki za izračun za optimizem popravljenega kriterija s pomočjo bootstrapa so:

1. Razvijemo napovedni model z uporabo celotnega prvotnega nabora podatkov in izračunamo navidezno učinkovitost modela na podlagi nekega kriterija (*apparent performance*).
2. Generiramo bootstrap vzorec (enake velikosti kot prvotni podatki) z vzorčenjem enot s ponavljanjem iz prvotnega nabora podatkov.
3. Z uporabo istih metod modeliranja in izbire napovednih spremenljivk v model kot v 1. koraku naredimo bootstrap model na bootstrap vzorcu.

   - Izračunamo navidezno učinkovitost modela na podlagi nekega kriterija na bootstrap vzorcu (*bootstrap performance*).

   - Izračunamo testno učinkovitost modela na podlagi nekega kriterija na prvotnih podatkih (*test performance*).
4. Izračunamo optimizem kot razliko med navidezno učinkovitostjo na bootstrap vzorcu in testno učinkovitostjo na prvotnih podatkih.
5. Korake 2 do 4 ponovimo večkrat (npr. 500-krat).
6. Izračunamo povprečno vrednost optimizma iz koraka 5.
7. Od navidezne učinkovitosti na prvotnih podatkih (iz koraka 1) odštejemo povprečni optimizem (iz koraka 6), da dobimo popravljeno vrednost kriterija za vrednotenje modela.

Za primer se bomo osredotočili na strategijo izbire nazaj.

```
Hitters$Salary <- log(Hitters$Salary)
```

```
bwd_sel_vsi = regsubsets(Salary ~., data = Hitters, nvmax = 19, method = "backward")
bwd_sel_vsi_summary <- summary(bwd_sel_vsi)
preds_vsi <- predict(bwd_sel_vsi, newdata = Hitters, id = which.min(bwd_sel_vsi_summary$cp))
```

```
(navidezni_c_spl <- mean(Hitters$Salary - preds_vsi, na.rm=TRUE))
```

```
[1] -7.946318e-15
```

```
(navidezni_c_nakl <- coef(lm(Hitters$Salary ~ preds_vsi))[2])
```

```
preds_vsi
        1
```

```
(navidezni_R2 <- cor(Hitters$Salary, preds_vsi, use="complete.obs")^2)
```

```
          [,1]
[1,] 0.5473898
```

```
(navidezni_rmse <- sqrt(mean((Hitters$Salary - preds_vsi)^2, na.rm=TRUE)))
```

```
[1] 0.5970775
```

Komentar: Model bo na učnih podatkih vedno popolno kalibriran!

```
set.seed(23345)
B=500
opt_c_spl <- opt_c_nakl <- opt_R2 <- opt_rmse <- numeric(B)

for(b in 1:B) {
  ind <- sample(1:nrow(Hitters), replace=TRUE) # indikator enot v bootstrap vzorcu
  my.data.boot <- Hitters[ind, ] # bootstrap vzorec
  my.mod.boot <- regsubsets(Salary ~., data = my.data.boot, nvmax = 19,
```

```
                              method = "backward") # izbira nazaj na bootstrap vzorcu

  #napovedi na boot učnem vzorcu
  preds_boot <- predict(my.mod.boot, newdata = my.data.boot,
                        id = which.min(summary(my.mod.boot)$cp))

  #napovedi na originalnih podatkih
  preds_orig <- predict(my.mod.boot, newdata = Hitters,
                        id = which.min(summary(my.mod.boot)$cp))

  #bootstrap performance
  c_spl.boot <- mean(my.data.boot$Salary - preds_boot, na.rm=TRUE) #=0
  c_nakl.boot <- coef(lm(my.data.boot$Salary ~ preds_boot))[2] #=1
  R2.boot <- cor(my.data.boot$Salary, preds_boot, use="complete.obs")^2
  rmse.boot <- sqrt(mean((my.data.boot$Salary - preds_boot)^2, na.rm=TRUE))

  #test performance
  c_spl.test <- mean(Hitters$Salary - preds_orig, na.rm=TRUE)
  c_nakl.test <- coef(lm(Hitters$Salary ~ preds_orig))[2]
  R2.test <- cor(Hitters$Salary, preds_orig, use="complete.obs")^2
  rmse.test <- sqrt(mean((Hitters$Salary - preds_orig)^2, na.rm=TRUE))

  #optimism za b bootstrap vzorec
  opt_c_spl[b]=c_spl.boot-c_spl.test
  opt_c_nakl[b]=c_nakl.boot-c_nakl.test
  opt_R2[b]=R2.boot-R2.test
  opt_rmse[b]=rmse.boot-rmse.test
}

popravljeni_c_spl <- navidezni_c_spl - mean(opt_c_spl)
popravljeni_c_nakl <- navidezni_c_nakl - mean(opt_c_nakl)
popravljeni_R2 <- navidezni_R2 - mean(opt_R2)
popravljeni_rmse <- navidezni_rmse - mean(opt_rmse)

primerjava.vrednotenja <- data.frame(Navidezni = c(navidezni_c_spl, navidezni_c_nakl,
                                                   navidezni_R2, navidezni_rmse),
                                     Popravljeni = c(popravljeni_c_spl, popravljeni_c_nakl,
                                                     popravljeni_R2, popravljeni_rmse),
                         Testni = primerjava.modelov["Izbira nazaj (Cp)", ])

kable(primerjava.vrednotenja, digits = 3,
      caption = "Primerjava načinov vrednotenja za strategijo izbire modela z izbiro nazaj.")
```

Tabela 2: Primerjava načinov vrednotenja za strategijo izbire modela z izbiro nazaj.

	Navidezni	Popravljeni	Testni
Splošna kalibracija	0.000	-0.006	-0.039
Naklon kalibracije	1.000	0.924	0.708
R2	0.547	0.477	0.389
RMSE	0.597	0.653	0.699

Domača naloga: Oglaševanje

Zamislimo si, da smo zaposleni kot statistiki. Najame nas stranka, ki jo zanima, kakšna je zveza med oglaševanjem in prodajo nekega izdelka. Na razpolago imamo podatke *Advertising.csv* o prodaji tega izdelka na 200 tržiščih, skupaj z oglaševalskim proračunom na vsakem od teh tržišč za tri različne medije: TV, radio in časopis. Stranka ne more neposredno vplivati na prodajo izdelka, lahko pa vpliva na višino oglaševalskega proračuna pri vsakem od treh medijev. Stranka nas prosi, da izdelamo načrt trženja za prihodnje leto, ki bo privedel do visoke prodaje izdelkov. Pri analizi poskušajte odgovoriti na naslednja vprašanja, ki bi utegnila zanimati vašo stranko:

- Ali obstaja povezanost med oglaševanjem ter prodajo?
- Če obstaja, kako močna je povezanost?
- Kateri oglaševalski mediji so povezani s prodajo?
- Kako močna je povezava med posameznim medijem in prodajo?
- Ali obstaja interakcija med posameznimi oglaševalskimi mediji?
- Ali je povezanost linearna?
- Kako natančno lahko na podlagi modela napovemo prodajo za nove enote?

```r
data <- read.csv("Advertising.csv", header=T)
str(data)
```

```
'data.frame':	200 obs. of  5 variables:
 $ X        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ TV       : num  230.1 44.5 17.2 151.5 180.8 ...
 $ Radio    : num  37.8 39.3 45.9 41.3 10.8 48.9 32.8 19.6 2.1 2.6 ...
 $ Newspaper: num  69.2 45.1 69.3 58.5 58.4 75 23.5 11.6 1 21.2 ...
 $ Sales    : num  22.1 10.4 9.3 18.5 12.9 7.2 11.8 13.2 4.8 10.6 ...
```