

## Metode uzorkovanja

### a) Validacioni skup podataka:

```
# load data
library(ISLR)
set.seed(1)
dim(Auto)

# sample train set and apply linear regression
train = sample(392, 196)
lm.fit = lm(mpg~horsepower, data=Auto, subset = train)

# MSE on test set
mean ((mpg - predict(lm.fit, Auto))[-train]^2)

# another model...
lm.fit2=lm(mpg~poly(horsepower, 2), data=Auto, subset=train)
mean((mpg - predict(lm.fit2, Auto))[-train]^2)

-----
# another seed provides different MSE result - validation set shortcoming
# test this!
set.seed(2)
train = sample(392, 196)
```

### b) LOOCV

```
# load this library
library(boot)
glm.fit = glm(mpg~horsepower, data=Auto) # generalized linear model
cv.err = cv.glm(Auto, glm.fit)
cv.err$delta # two values
```

### c) K-fold

```
set.seed(17)
cv.error.10 = rep(0 ,10)
for(i in 1:10){
  glm.fit = glm(mpg~poly(horsepower, i), data=Auto)
  cv.error.10[i]= cv.glm(Auto, glm.fit, K=10)$delta[1]
}
cv.error.10
```

## d) Bootstrap

```

set.seed(1)
alpha.fn = function(data, index){
  X = data$X[index]
  Y = data$Y[index]
  return (( var(Y)-cov(X,Y))/(var(X) + var(Y)- 2*cov(X,Y)))
}

alpha.fn(Portfolio, sample(100, 100, replace=T))

```

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

ili

```
boot(Portfolio, alpha.fn, R=1000)
```

## Regularizacija

instalirati **glmnet** paket

Note that by default, the **glmnet()** function standardizes the variables so that they are on the same scale. To turn off this default setting, use the argument **standardize=FALSE**.

```

library(glmnet)
library(ISLR)

Hitters = na.omit(Hitters)

# create dummy variables! -1 without intercept. Test a difference!
x = model.matrix(Salary~., Hitters)[,-1]
y = Hitters$Salary

# create grid for lambda
grid = 10^seq(10, -2, length=100)
ridge.mod = glmnet(x, y, alpha=0, lambda=grid)

dim(coef(ridge.mod))
# get lambda value
ridge.mod$lambda[50]
# get coefficients for a selected lambda
coef(ridge.mod)[,50]
# calculate l2 norm
sqrt(sum(coef(ridge.mod)[-1, 50]^2))

```

```
# test another lambda value
ridge.mod$lambda[60]
coef(ridge.mod)[,60]
sqrt(sum(coef(ridge.mod)[-1, 60]^2))
```

### SPLIT on Train and Test set: Why?

```
set.seed(1)
train = sample(1:nrow(x), nrow(x)/2)
test = (-train)
y.test = y[test]

ridge.mod = glmnet(x[train,], y[train], alpha=0, lambda=grid, thresh=1e-12)

# predict for lambda = 4
ridge.pred = predict(ridge.mod, s=4, newx=x[test,])
mean((ridge.pred - y.test)^2)

# predict for lambda = 10^10
ridge.pred = predict(ridge.mod, s=1e10, newx=x[test,])
mean((ridge.pred - y.test)^2)

# lambda = 0 // it is better to use lm!
ridge.pred = predict(ridge.mod, s=0, newx=x[test,], exact=T)
mean((ridge.pred - y.test)^2)
```

In general, instead of arbitrarily choosing  $\lambda$  value, it would be better to use cross-validation to choose the tuning parameter  $\lambda$ .

```
cv.out = cv.glmnet(x[train,], y[train], alpha=0)
plot(cv.out)
# get the best lambda value
bestlam = cv.out$lambda.min
bestlam

ridge.pred = predict(ridge.mod, s=bestlam, newx=x[test,])
mean((ridge.pred - y.test)^2)
```

Now, we can fit model on the entire dataset:

```
out = glmnet(x, y, alpha=0)
predict(out, type="coefficients", s=bestlam)[1:20,]
```

LASSO – Domaci 😊