Regularyzacja - Lab09

Jakub Bryl

23 12 2019

Regularyzacja

Przedstawiony zbiór danych obrazuje jakość białego wina w zależności od 11 współczynników (personalne zainteresowanie na co najlepiej patrzeć przy wyborze wina w sklepie).

Wczytanie danych i normalizacja

```
data<-read_csv("winequality-white.csv")
```

```
## Parsed with column specification:
## cols(
##   `fixed acidity` = col_double(),
##   `volatile acidity` = col_double(),
##   `citric acid` = col_double(),
##   `residual sugar` = col_double(),
##   chlorides = col_double(),
##   `free sulfur dioxide` = col_double(),
##   `total sulfur dioxide` = col_double(),
##   density = col_double(),
##   pH = col_double(),
##   sulphates = col_double(),
##   alcohol = col_double(),
##   quality = col_double()
## )
```

```
summary(data)
```

```
##  fixed acidity    volatile acidity  citric acid     residual sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides        free sulfur dioxide total sulfur dioxide
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0
##  Median :0.04300   Median : 34.00      Median :134.0
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0
##     density             pH          sulphates         alcohol
##  Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00
##  1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50
##  Median :0.9937   Median :3.180   Median :0.4700   Median :10.40
##  Mean   :0.9940   Mean   :3.188   Mean   :0.4898   Mean   :10.51
##  3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40
##  Max.   :1.0390   Max.   :3.820   Max.   :1.0800   Max.   :14.20
##     quality
##  Min.   :3.000
##  1st Qu.:5.000
##  Median :6.000
##  Mean   :5.878
##  3rd Qu.:6.000
##  Max.   :9.000
```

```
#Normalizacja
data_norm <- scale(data, center = TRUE, scale = TRUE) %>% as.data.frame()
summary(data_norm)
```

```
##  fixed acidity      volatile acidity   citric acid      residual sugar
##  Min.   :-3.61998   Min.   :-1.9668   Min.   :-2.7615   Min.   :-1.1418
##  1st Qu.:-0.65743   1st Qu.:-0.6770   1st Qu.:-0.5304   1st Qu.:-0.9250
##  Median :-0.06492   Median :-0.1810   Median :-0.1173   Median :-0.2349
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.52758   3rd Qu.: 0.4143   3rd Qu.: 0.4612   3rd Qu.: 0.6917
##  Max.   : 8.70422   Max.   : 8.1528   Max.   :10.9553   Max.   :11.7129
##    chlorides       free sulfur dioxide total sulfur dioxide
##  Min.   :-1.6831   Min.   :-1.95848    Min.   :-3.0439
##  1st Qu.:-0.4473   1st Qu.:-0.72370    1st Qu.:-0.7144
##  Median :-0.1269   Median :-0.07691    Median :-0.1026
##  Mean   : 0.0000   Mean   : 0.00000    Mean   : 0.0000
##  3rd Qu.: 0.1935   3rd Qu.: 0.62867    3rd Qu.: 0.6739
##  Max.   :13.7417   Max.   :14.91679    Max.   : 7.0977
##     density              pH             sulphates
##  Min.   :-2.31280   Min.   :-3.10109   Min.   :-2.3645
##  1st Qu.:-0.77063   1st Qu.:-0.65077   1st Qu.:-0.6996
##  Median :-0.09608   Median :-0.05475   Median :-0.1739
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.69298   3rd Qu.: 0.60750   3rd Qu.: 0.5271
##  Max.   :15.02976   Max.   : 4.18365   Max.   : 5.1711
##     alcohol            quality
##  Min.   :-2.04309   Min.   :-3.2495
##  1st Qu.:-0.82419   1st Qu.:-0.9913
##  Median :-0.09285   Median : 0.1379
##  Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.71974   3rd Qu.: 0.1379
##  Max.   : 2.99502   Max.   : 3.5252
```

Regresja liniowa - sprawdzenie p-val

```
model <- lm(quality ~. ,data_norm)
summary(model)
```

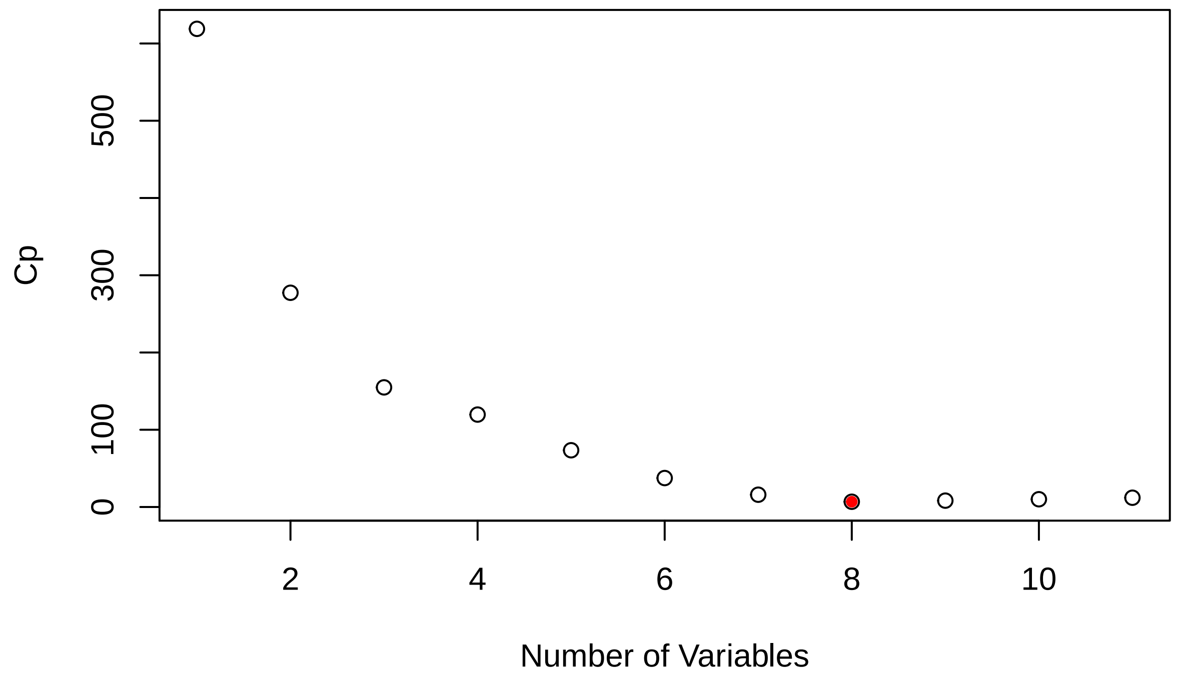
```
##
## Call:
## lm(formula = quality ~ ., data = data_norm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3299 -0.5571 -0.0428  0.5235  3.5164
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -3.668e-15  1.212e-02   0.000  1.00000
## `fixed acidity`         6.243e-02  1.989e-02   3.139  0.00171 **
## `volatile acidity`     -2.120e-01  1.295e-02 -16.373  < 2e-16 ***
## `citric acid`           3.019e-03  1.309e-02   0.231  0.81759
## `residual sugar`        4.667e-01  4.311e-02  10.825  < 2e-16 ***
## chlorides              -6.100e-03  1.348e-02  -0.452  0.65097
## `free sulfur dioxide`   7.168e-02  1.621e-02   4.422 9.99e-06 ***
## `total sulfur dioxide` -1.371e-02  1.814e-02  -0.756  0.44979
## density                -5.075e-01  6.442e-02  -7.879 4.04e-15 ***
## pH                      1.170e-01  1.797e-02   6.513 8.10e-11 ***
## sulphates               8.137e-02  1.294e-02   6.291 3.44e-10 ***
## alcohol                 2.688e-01  3.366e-02   7.988 1.70e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8484 on 4886 degrees of freedom
## Multiple R-squared:  0.2819, Adjusted R-squared:  0.2803
## F-statistic: 174.3 on 11 and 4886 DF,  p-value: < 2.2e-16
```

Znalezienie optymalnego modelu

Rozdzielenie wspolczynnikow (regsubsets -> regression sub sets)

Sposób 1 - default: Forward:

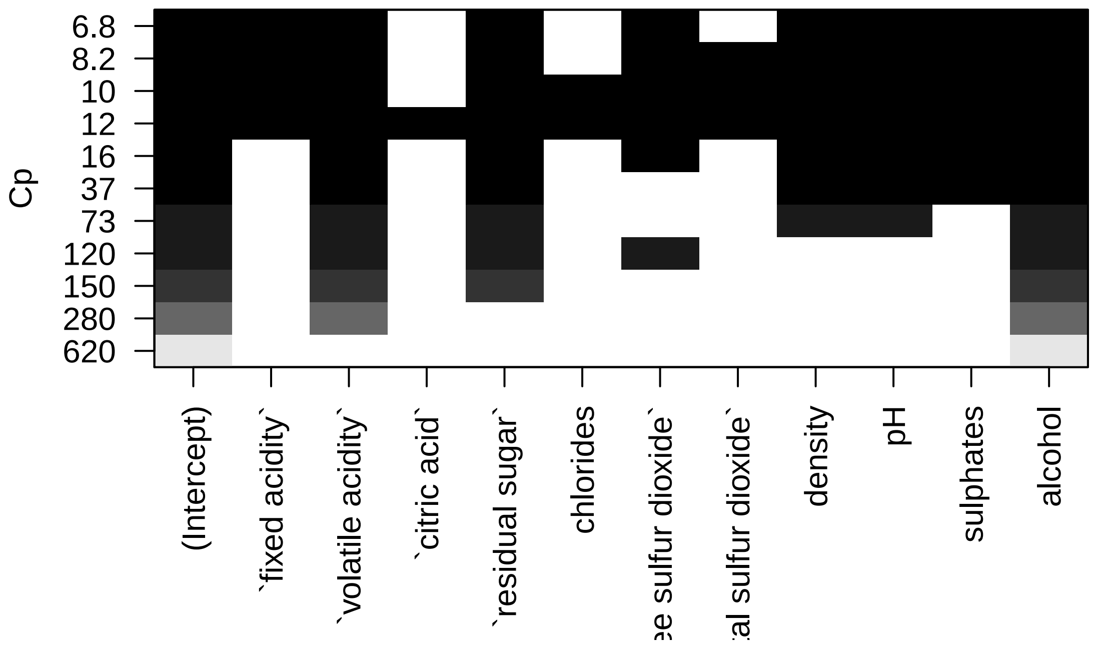
```r
library(leaps)

regfit.full=regsubsets(quality~.,data=data_norm, nvmax=11)
reg.summary=summary(regfit.full)
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp")
a=which.min(reg.summary$cp)
points(a,reg.summary$cp[a],pch=20,col="red")
```



Minimum wypada dla modelu zawierającego 8 wartości składowych, lecz różnica między modelem złożonym z 7 a 8 wartości jest znikoma, dlatego warto rozważyć model złożony z 7 współczynników (bez 'al sulfur dioxide').

Inna reprezentacja powyższego modelu Forward Stepwise Selection

```
plot(regfit.full,scale="Cp")
```

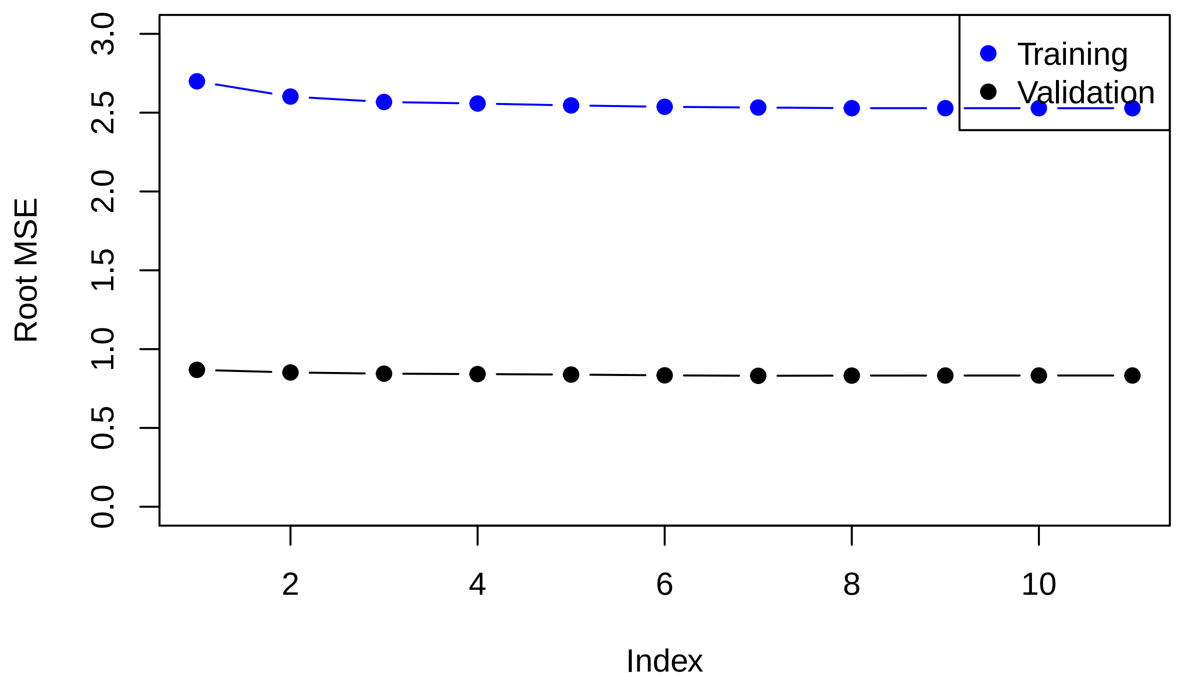


Selekcja modelu z użyciem zbioru walidacyjnego

Tworzymy zbiór treningowy w stosunku 70:30

```
set.seed(331)
train=sample(seq(4898),3500,replace=FALSE)
regfit.fwd=regsubsets(`quality`~.,data=data_norm[train,],nvmax=11,method="forward")
```

```
val.errors=rep(NA,11)
x.test=model.matrix(`quality`~.,data=data_norm[-train,])
for(i in 1:11){
  coefi=coef(regfit.fwd,id=i)
  pred=x.test[,names(coefi)]%*%coefi
  val.errors[i]=mean((data_norm$`quality`[-train]-pred)^2)
}

plot(sqrt(val.errors),ylab="Root MSE",ylim=c(0,3),pch=19,type="b")
points(sqrt(regfit.fwd$rss[-1]/400),col="blue",pch=19,type="b")
legend("topright",legend=c("Training","Validation"),col=c("blue","black"),pch=19)
```

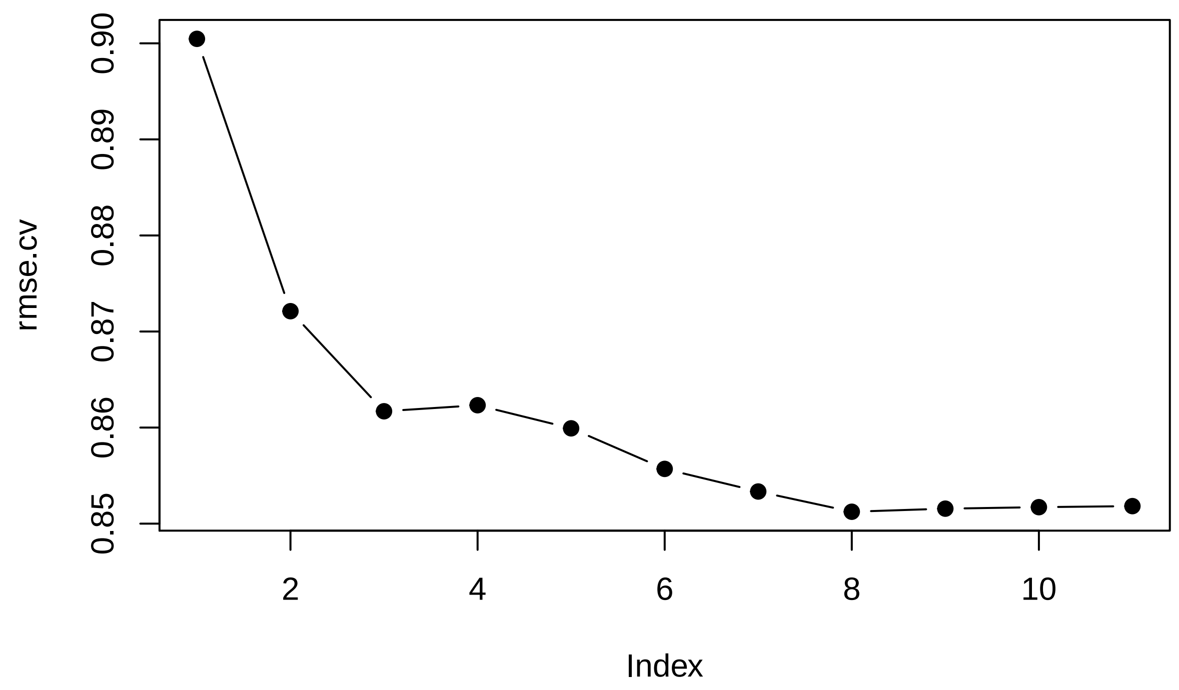


Selekcja modelu z użyciem Cross-Validacji

```
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  mat[,names(coefi)]%*%coefi
}
```

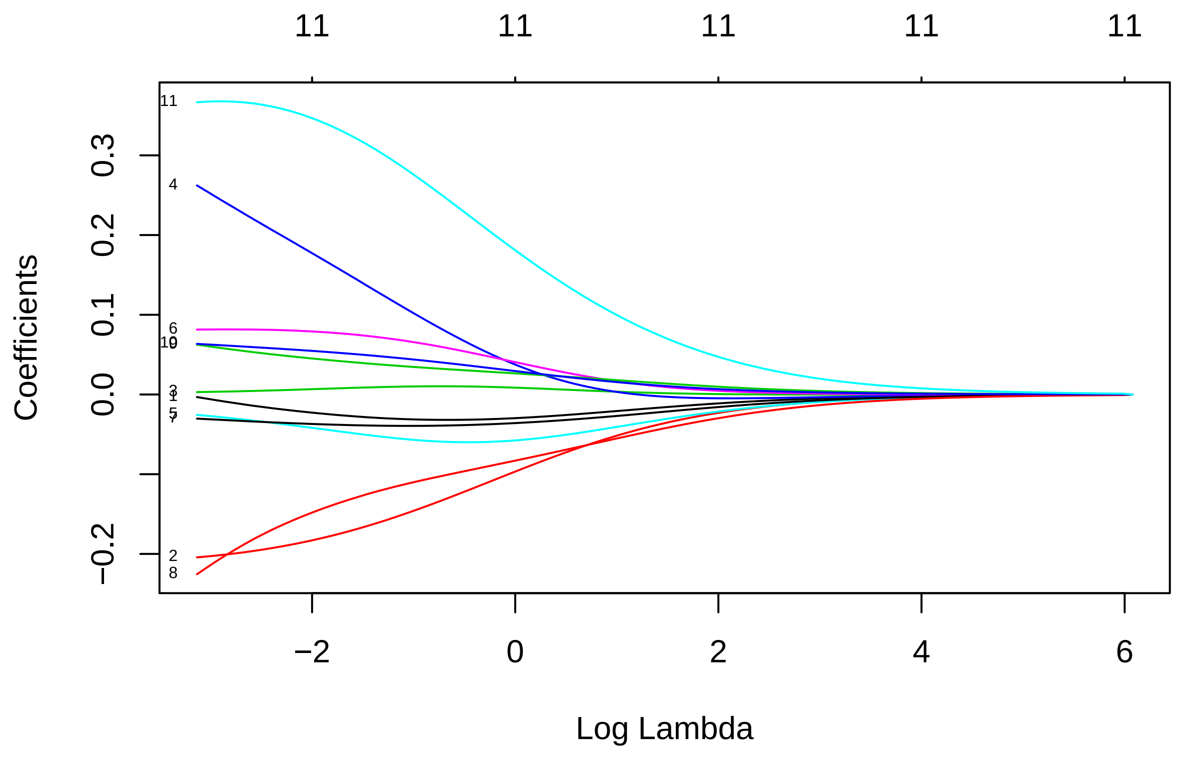
Na poniższym wykresie został przedstawiony błąd predykcji dla modelu zawierającego X liczbę zmiennych (współczynników).

```
set.seed(11)
folds=sample(rep(1:10,length=nrow(data_norm)))
cv.errors=matrix(NA,10,11)
for(k in 1:10){
  best.fit=regsubsets(quality~.,data=data_norm[folds!=k,],nvmax=11,method="forward")
  for(i in 1:11){
    pred=predict(best.fit,data_norm[folds==k,],id=i)
    cv.errors[k,i]=mean( (data_norm$`quality`[folds==k]-pred)^2)
  }
}
rmse.cv=sqrt(apply(cv.errors,2,mean))
plot(rmse.cv,pch=19,type="b")
```

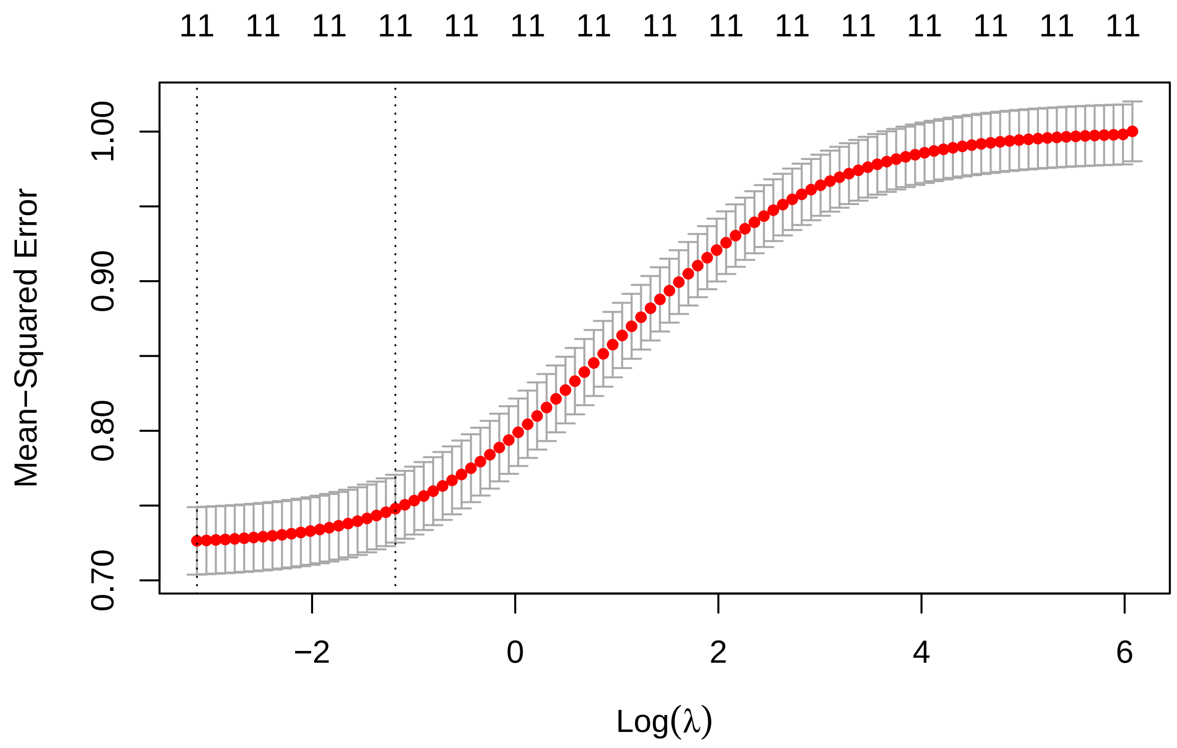


Ridge Regression

```
library(glmnet)
x=model.matrix(quality~.-1,data=data_norm)
y=data_norm$quality

#alpha=0 -> mowi ze regularyzacja l2
fit.ridge=glmnet(x,y,alpha=0)
plot(fit.ridge,xvar="lambda",label=TRUE)
```

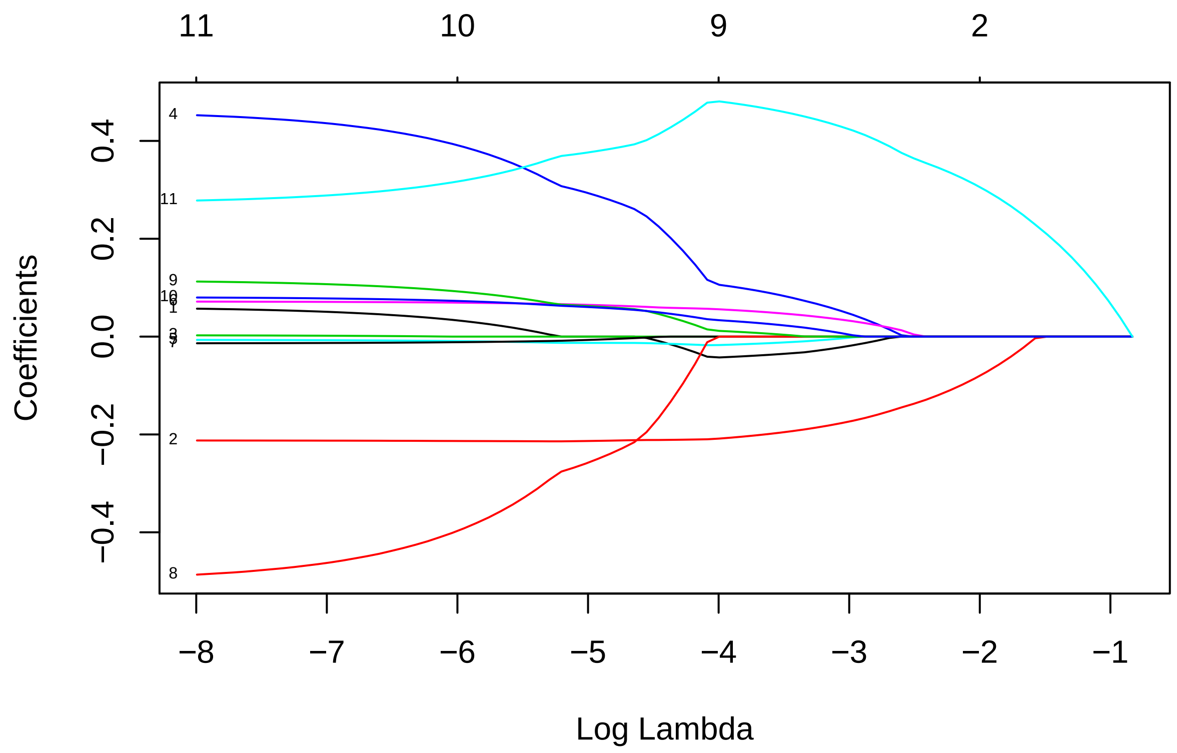


```
cv.ridge=cv.glmnet(x,y,alpha=0)
plot(cv.ridge)
```

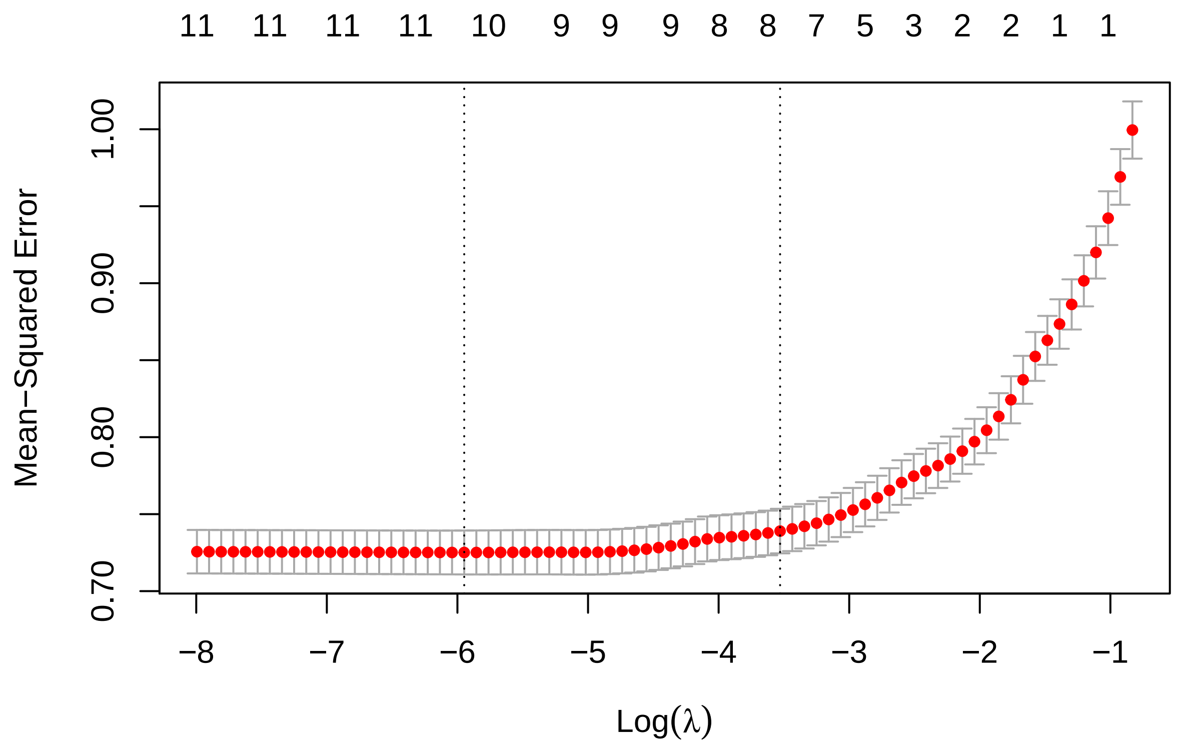


Lasso

```
fit.lasso=glmnet(x,y)
plot(fit.lasso,xvar="lambda",label=TRUE)
```



```
cv.lasso=cv.glmnet(x,y)
plot(cv.lasso)
```



```r
coef(cv.lasso)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                     1
## (Intercept)             4.933189e-16
## `fixed acidity`        -3.574513e-02
## `volatile acidity`     -1.963642e-01
## `citric acid`           .
## `residual sugar`        8.480340e-02
## chlorides              -1.234555e-02
## `free sulfur dioxide`   4.773037e-02
## `total sulfur dioxide`  .
## density                 .
## pH                      4.333538e-03
## sulphates               2.371510e-02
## alcohol                 4.608120e-01
```

Wybór parametru lambda, sprawdzamy z jaką wartością parametru lambda błąd jest najmniejszy (ponowne użycie Cross-Validacji)

```r
lasso.tr=glmnet(x[train,],y[train])
lasso.tr
```

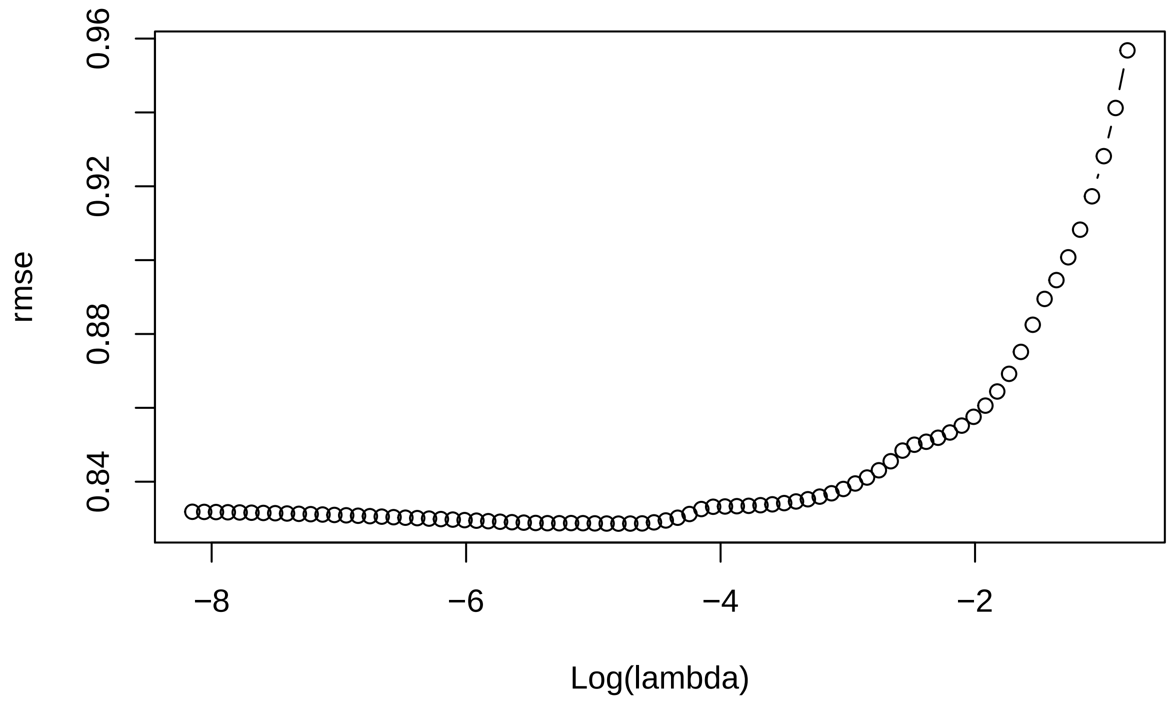
```
##
## Call:  glmnet(x = x[train, ], y = y[train])
```

```
##
##    Df    %Dev  Lambda
## 1   0 0.00000 0.44840
## 2   1 0.03303 0.40850
## 3   1 0.06044 0.37230
## 4   1 0.08321 0.33920
## 5   1 0.10210 0.30900
## 6   1 0.11780 0.28160
## 7   1 0.13080 0.25660
## 8   1 0.14160 0.23380
## 9   2 0.15670 0.21300
## 10  2 0.17270 0.19410
## 11  2 0.18610 0.17680
## 12  2 0.19710 0.16110
## 13  2 0.20630 0.14680
## 14  2 0.21400 0.13380
## 15  2 0.22030 0.12190
## 16  2 0.22550 0.11110
## 17  2 0.22990 0.10120
## 18  2 0.23350 0.09221
## 19  2 0.23650 0.08402
## 20  3 0.24090 0.07655
## 21  5 0.24710 0.06975
## 22  5 0.25250 0.06356
## 23  5 0.25700 0.05791
## 24  6 0.26080 0.05277
## 25  6 0.26430 0.04808
## 26  6 0.26730 0.04381
## 27  6 0.26970 0.03992
## 28  7 0.27180 0.03637
## 29  7 0.27350 0.03314
## 30  7 0.27500 0.03019
## 31  7 0.27620 0.02751
## 32  7 0.27720 0.02507
## 33  7 0.27800 0.02284
## 34  7 0.27870 0.02081
## 35  8 0.27920 0.01896
## 36  8 0.27970 0.01728
## 37  9 0.28080 0.01574
## 38  9 0.28290 0.01434
## 39  9 0.28460 0.01307
## 40  9 0.28600 0.01191
## 41  9 0.28720 0.01085
## 42  8 0.28800 0.00989
## 43  8 0.28850 0.00901
## 44  8 0.28890 0.00821
## 45  8 0.28920 0.00748
## 46  8 0.28950 0.00682
## 47  8 0.28970 0.00621
## 48  9 0.29000 0.00566
## 49  9 0.29050 0.00516
## 50  9 0.29090 0.00470
## 51  9 0.29130 0.00428
## 52  9 0.29150 0.00390
## 53  9 0.29180 0.00355
## 54  9 0.29200 0.00324
## 55  9 0.29220 0.00295
## 56  9 0.29230 0.00269
## 57  9 0.29240 0.00245
## 58  8 0.29250 0.00223
## 59  8 0.29260 0.00203
## 60  8 0.29270 0.00185
## 61  8 0.29270 0.00169
## 62  8 0.29280 0.00154
## 63  9 0.29280 0.00140
## 64  9 0.29290 0.00128
## 65  9 0.29290 0.00116
## 66  9 0.29290 0.00106
## 67  9 0.29290 0.00097
## 68 11 0.29300 0.00088
## 69 11 0.29300 0.00080
## 70 11 0.29300 0.00073
## 71 11 0.29300 0.00067
## 72 11 0.29300 0.00061
## 73 11 0.29300 0.00055
## 74 11 0.29300 0.00050
## 75 11 0.29300 0.00046
## 76 11 0.29300 0.00042
## 77 11 0.29300 0.00038
## 78 11 0.29300 0.00035
## 79 11 0.29300 0.00032
## 80 11 0.29300 0.00029
```

```r
pred=predict(lasso.tr,x[-train,])
dim(pred)
```

```
## [1] 1398   80
```

```r
rmse= sqrt(apply((y[-train]-pred)^2,2,mean))
plot(log(lasso.tr$lambda),rmse,type="b",xlab="Log(lambda)")
```



```
lam.best=lasso.tr$lambda[order(rmse)[1]]
lam.best
```

```
## [1] 0.008208603
```

```
coef(lasso.tr,s=lam.best)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                   1
## (Intercept)             0.001910617
## `fixed acidity`         .
## `volatile acidity`     -0.235526501
## `citric acid`           .
## `residual sugar`        0.309773045
## chlorides              -0.004989579
## `free sulfur dioxide`   0.050740802
## `total sulfur dioxide`  .
## density                -0.277472906
## pH                      0.052988320
## sulphates               0.060150632
## alcohol                 0.382461368
```