최소자승법에 의한 잔차(관측값-예측값)의 제곱합과 페널티항의 합이 최소가 되는 회귀계수를 추정

1)ridge regression analysis
모델의 설명력에 기여하지 못하는 독립변수의회귀계수 크기를 0 에 근접하도록 축소

L2-norm 페널티항으로 회귀모델에 페널티를 부과함으로써 회귀계수를 축소

$$L_{hridge}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda \sum_{j=1}^{m} w_j \hat{\beta}_j^2.$$

```
# (y= 관측값, y.hat= 예측값, n= 표본크기, lamda= 튜닝 패러미터, beta= 회귀계수, p= 독립변수)
```

2)lasso regression analysis
모델의 설명력에 기여하지 못하는 독립변수의 회귀계수 크기를 0 으로 만듬

L1-norm 페널티항으로 회귀모델에 페널티를 부과함으로써 회귀계수를 축소

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda \sum_{j=1}^{m} |\hat{\beta}_j|.$$

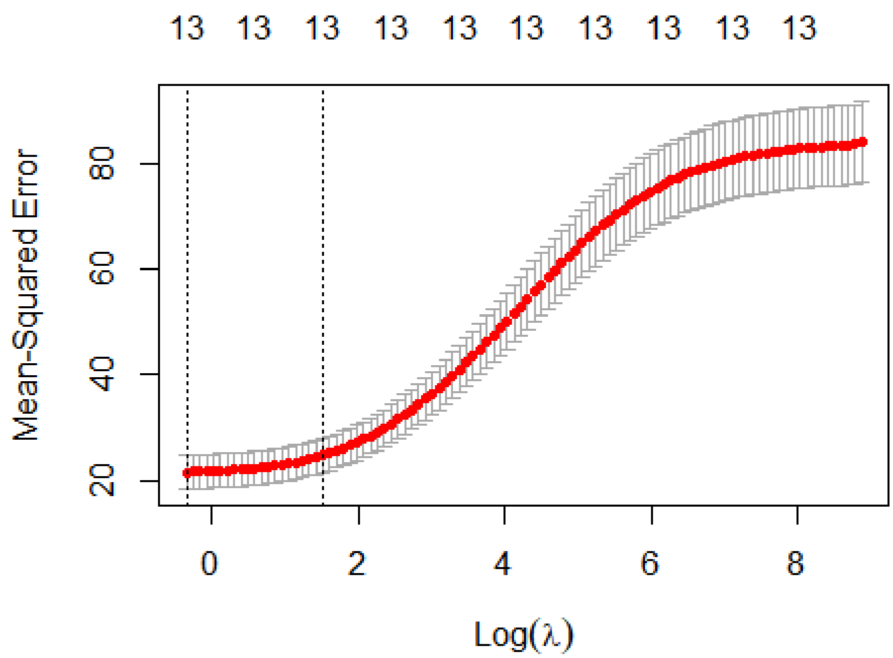
3)Elasticnet regression analysis

L1-norm L2-norm 모두를 이용하여 회귀모델에 패널티를 부과

$$L_{enet}(\hat{\beta}) = \frac{\sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2}{2n} + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j=1}^{m}|\hat{\beta}_j|\right),$$

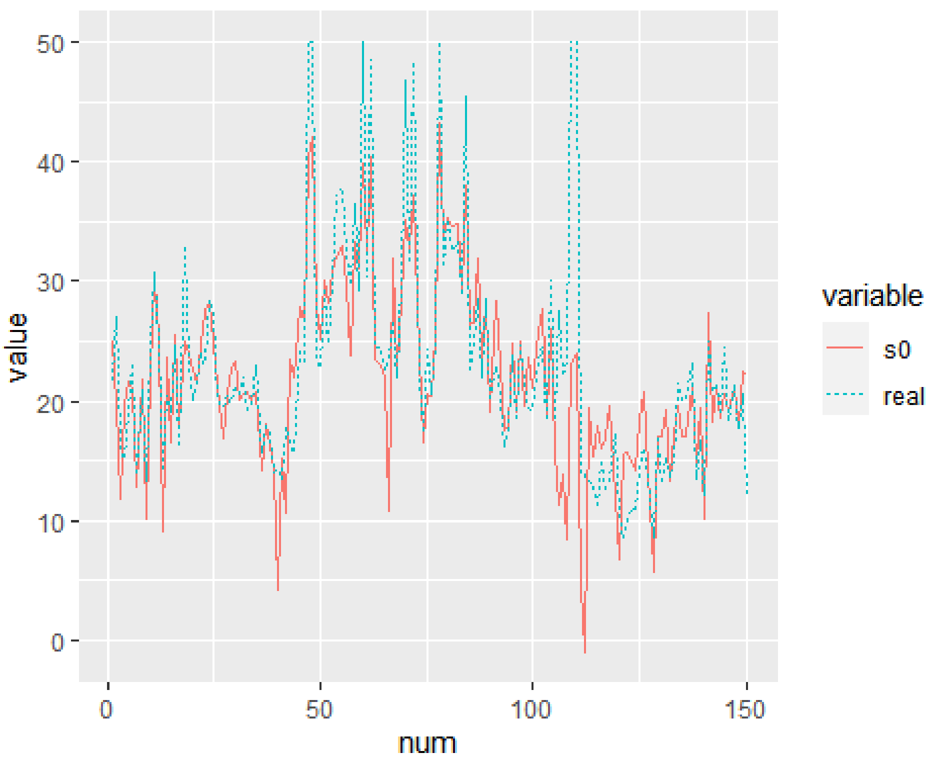
```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...

set.seed(910)
train <- createDataPartition(y=Boston$medv, p=0.7, list=FALSE)
Boston.train <- Boston[train,]
Boston.test <- Boston[-train,]
nrow(Boston.train)

## [1] 356
```

```r
nrow(Boston.test)

## [1] 150

#1)__ ridge
# 예측변수 행렬생성과정
x <- model.matrix(medv ~.,Boston.train)[,-1]
y <- Boston.train$medv

set.seed(910)
Boston.cv <- cv.glmnet(x=x, y=y, family = "gaussian", alpha=0, type.measure = "mse")
plot(Boston.cv)
```




```r
min.logL <- log(Boston.cv$lambda.min)
paste0("min(log(lamda)= ",min.logL)

## [1] "min(log(lamda)= -0.347360205272797"

Boston.rt <- glmnet(x=x, y=y, family = "gaussian",
              alpha=0, lambda= Boston.cv$lambda.min)
coef(Boston.rt)

## 14 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept)  24.264971600
## crim        -0.076292860
## zn           0.030875511
## indus       -0.015933238
## chas         2.515960237
## nox        -10.398271746
```

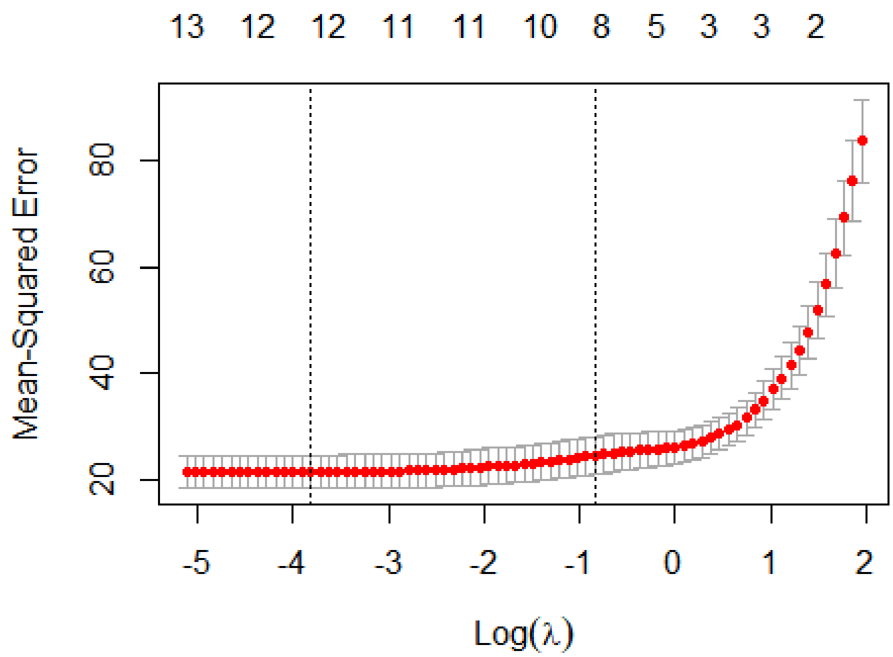
```
## rm          4.462323056
## age        -0.002801750
## dis        -1.037992066
## rad         0.122466453
## tax        -0.006205569
## ptratio     -0.812164578
## black       0.006640468
## lstat      -0.491249237

# meeve(주택가격 예측)
test.x <- model.matrix(medv ~.,Boston.test)[,-1]
Boston.pred <- predict(Boston.rt, newx = test.x)

# visualization
Boston.pred.df <- as.data.frame(Boston.pred)
Boston.pred.df <- Boston.pred.df %>% mutate(num = 1:nrow(Boston.pred.df),
                                  real = Boston.test$medv)

Boston.pred.df.v <- melt(Boston.pred.df,measure.vars = c("s0","real"))
ggplot(Boston.pred.df.v, aes(x=num,y=value))+
  geom_line(aes(col=variable, linetype=variable))
```




```
# 모델평가
postResample(pred= Boston.pred, obs= Boston.test$medv)

##     RMSE Rsquared      MAE
## 5.537079 0.649743 3.670072
```

@details

```r
lambda = 10^seq(-5,5,length=100)
set.seed(910)
ridge.f <- train(form=medv~., data=Boston.train,
            method="glmnet",
            trConrol=trainControl(method="cv",
                        number=10),
            tunGrid=expand.grid(alpha=0, lambda=lambda))

coef(ridge.f$finalModel, ridge.f$bestTune$lambda)

## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  29.112031361
## crim         -0.082895722
## zn            0.036829730
## indus         0.008724782
## chas          2.299410970
## nox         -13.516296677
## rm            4.360093415
## age           .
## dis          -1.267218011
## rad           0.193887289
## tax          -0.008990197
## ptratio      -0.863141905
## black         0.006495051
## lstat        -0.543055558

ridge.f.pred <- predict(ridge.f, Boston.test)
postResample(pred=ridge.f.pred, obs=Boston.test$medv)

##      RMSE  Rsquared       MAE
## 5.5112296 0.6558237 3.7243188

#2)__ lasso
set.seed(910)
Boston.cv <- cv.glmnet(x=x, y=y, family = "gaussian", alpha=1, type.measure = "mse")
plot(Boston.cv)
```




```
paste0("min(log(lamda)= ",log(Boston.cv$lambda.min))

## [1] "min(log(lamda)= -3.81286706561109"

#예측정확도와 간명도간의 균형점
paste0("min(log(lamda)= ",log(Boston.cv$lambda.1se))

## [1] "min(log(lamda)= -0.835787352189394"

# 회귀계수
coef(Boston.cv, Boston.cv$lambda.min)

## 14 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept)  30.559958240
## crim         -0.084622922
## zn            0.038346696
## indus         0.014101103
## chas          2.221748677
## nox         -14.255463038
## rm            4.311577710
## age           .
## dis          -1.330559789
## rad           0.215025132
## tax          -0.009867540
## ptratio      -0.873883125
## black         0.006383636
## lstat        -0.556055027

coef(Boston.cv, Boston.cv$lambda.1se)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 13.168003677
## crim        -0.017716936
## zn           .
## indus        .
## chas         1.370293410
## nox          .
## rm           4.643442669
## age          .
## dis         -0.131315661
## rad          .
## tax         -0.001070102
## ptratio     -0.740778992
## black        0.003555972
## lstat       -0.537502242
```

```
#lamda.min 을 적용했을 때와, lamda.1se 를 지정했을 때의 성능비교
# 1)각 예측모델 생성 및 평가
Boston.rt2_1 <- glmnet(x=x, y=y, family = "gaussian",
                  alpha=1, lambda= Boston.cv$lambda.min)
Boston.pred_2_1 <- predict(Boston.rt2_1, newx = test.x)
postResample(pred= Boston.pred_2_1, obs= Boston.test$medv)

##      RMSE  Rsquared       MAE
## 5.5085481 0.6569651 3.7396470

Boston.rt2_2 <- glmnet(x=x, y=y, family = "gaussian",
                  alpha=1, lambda= Boston.cv$lambda.1se)
Boston.pred_2_2 <- predict(Boston.rt2_2, newx = test.x)
postResample(pred= Boston.pred_2_2, obs= Boston.test$medv)

##      RMSE  Rsquared       MAE
## 5.9136052 0.6020505 3.9256357

laso.rts_1 <- postResample(pred= Boston.pred_2_1, obs= Boston.test$medv)
laso.rts_2 <- postResample(pred= Boston.pred_2_2, obs= Boston.test$medv)

# 성능비교 시각화
laso.r <- cbind(laso.rts_1,laso.rts_2)
laso.r <- melt(laso.r)
ggplot(data=laso.r, aes(x=Var1,y=value, fill=Var2)) +
  geom_col(position = "dodge")+
  scale_x_discrete(limits = c("RMSE","MAE","Rsquared"))
```

@details

```
set.seed(910)
lasso.f <- train(form=medv~., data=Boston.train,
            method="glmnet",
            trConrol=trainControl(method="cv",
                          number=10),
            tunGrid=expand.grid(alpha=1, lambda=lambda))

coef(lasso.f$finalModel, lasso.f$bestTune$lambda)

## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  29.112031361
## crim        -0.082895722
## zn           0.036829730
## indus        0.008724782
## chas         2.299410970
## nox        -13.516296677
## rm           4.360093415
## age          .
## dis         -1.267218011
## rad          0.193887289
## tax         -0.008990197
## ptratio     -0.863141905
## black        0.006495051
## lstat       -0.543055558

lasso.f.pred <- predict(lasso.f, Boston.test)
postResample(pred=lasso.f.pred, obs=Boston.test$medv)
```

```
##      RMSE  Rsquared       MAE
## 5.5112296 0.6558237 3.7243188
```

```
#3)__ elasticnet
# 1)__ 교차검증(k fols cross validation) 및 최적 lambda 값 추적
Boston.cv <- train(form= medv~., data=Boston.train,
                method="glmnet",
                trControl=trainControl(method="cv",
                                 number=10), tunLength=10)
Boston.cv$bestTune
```

```
##   alpha    lambda
## 2   0.1 0.1413102
```

```
Boston.rt3 <- glmnet(x=x, y=y, family = "gaussian",
                 alpha=Boston.cv$bestTune$alpha,
                 lambda= Boston.cv$bestTune$lambda)
coef(Boston.rt3)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  29.094836306
## crim         -0.083129038
## zn            0.036866735
## indus         0.008407709
## chas          2.299690296
## nox         -13.529151839
## rm            4.363305633
## age           .
## dis          -1.267388292
## rad           0.194164705
## tax          -0.008997033
## ptratio      -0.863203603
## black         0.006499956
## lstat        -0.542391953
```

```
Boston.rt3_pred <- predict(Boston.rt3, newx = test.x)
postResample(pred= Boston.rt3_pred, obs= Boston.test$medv)
```

```
##      RMSE  Rsquared       MAE
## 5.5109583 0.6558467 3.7239892
```

```
set.seed(910)
elastic.f <- train(form=medv~., data=Boston.train,
               method="glmnet",
               trConrol=trainControl(method="cv",
                                number=10),
               tuneLength=10)

coef(elastic.f$finalModel, elastic.f$bestTune$lambda)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  27.777280170
## crim         -0.079218264
## zn            0.034668167
## indus         .
## chas          2.331088928
```

```
## nox        -12.586939722
## rm           4.391500560
## age          .
## dis         -1.206775650
## rad          0.169408631
## tax         -0.007969754
## ptratio     -0.849256704
## black        0.006424659
## lstat       -0.536478021

elastic.f.pred <- predict(elastic.f, Boston.test)
postResample(pred=elastic.f.pred, obs=Boston.test$medv)

##      RMSE  Rsquared       MAE
## 5.5225524 0.6539466 3.7171759

medels <- list(ridge=ridge.f, lasso=lasso.f, elastic=elastic.f)
# 3 모델간 RMSE 값 유사
summary(resamples(medels), metric="RMSE")

##
## Call:
## summary.resamples(object = resamples(medels), metric = "RMSE")
##
## Models: ridge, lasso, elastic
## Number of resamples: 25
##
## RMSE
##             Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## ridge   3.551622 4.138042 4.666461 4.590418 5.000444 5.586917    0
## lasso   3.551622 4.138042 4.666461 4.590418 5.000444 5.586917    0
## elastic 3.536278 4.121813 4.643381 4.589322 5.008694 5.599218    0

# 3 모델간 통계적 유의성 없음
summary(diff(resamples(medels), metric="RMSE"))

##
## Call:
## summary.diff.resamples(object = diff(resamples(medels), metric = "RMSE"))
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## RMSE
##         ridge lasso    elastic
## ridge         0.000000 0.001096
## lasso   NA             0.001096
## elastic 1     1

# 결론 : 간명도를 위해 LASSO 또는 Elastic 채택
```