Model Evaluation and Selection

Monika Baloda

In this R-markdown, we evaluate different models and then comapre them to select the best performing model. We use the `Boston` data set contains housing values in suburbs of Boston.

Our work in here can be written in following points:

Part I : Model Evaluation Techniques

1. `Validation set approach`
2. `Forward selection method of variable selection`
3. `Cross-validation (CV) approach`

Part II: Model Selection

1. `Polynomial Regression and Generalized Additive Models (GAMS)`
2. `Model selection using least test error rate.`

Load necessary packages

```
#install.packages("MASS")
library(tidyverse) # for `ggplot2`, `dplyr`, and more
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.0     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
library(MASS) # for `Boston` data set
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
library(boot) # for `cv.glm` function
```

Set the random seed

```
# set the random seed so the analysis is reproducible
set.seed(232)
```

***Part I: Model Evaluation**

Following codes introduces us to the dataset but for the sake of space, I comment these commands so that they do not appear in rendered PDF.

```
#?Boston # full documentation
#dim(Boston)
#glimpse(Boston)
```

We can perform a multiple linear regression which uses the full set of predictors to estimate the median housing values `medv`.

```
lm.full <- lm(medv ~ . , Boston)
summary(lm.full)
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

1) Variable/feature selection: forward selection

Using all predictors increases the model complexity, and doesn't necessarily give us the best estimator.

Below we will practice the **forward selection** strategy.

For **forward selection**, we begin with the null model — a model that contains an intercept but no predictors. We then fit p simple linear regressions and add to the null model the variable that results in the lowest Residual Sum of Squares (RSS). We then add to that model the variable that results in the lowest RSS for the new two-variable model. This approach is continued until some stopping rule is satisfied.

Now we write R code to pick the top 3 most important variables using the *forward selection* strategy.

We save the resulting model which uses only the top 3 variables to be `lm.fwd3`.

Question Name the three most important variables.

Answer Three most important variables recommonded by forward selection method are *crim*, *rm*, and *lstat*.

Question What is the adjusted R^2 value of `lm.fwd3`?

Answer The adjusted R^2 statistics of this model is 0.6437.

The following code implements desired code and gives the aforementioned results.

```r
# fit null model with just the intercept
lm.null <- lm(medv ~ 1, data = Boston)

# initialize variables
n <- ncol(Boston) - 1  # number of predictors
added <- c()  # added variables
current <- lm.null  # current model

# forward selection loop
for (i in 1:n) {
  # initialize variables for best addition
  best_rss <- Inf
  best_var <- NULL

  # loop through remaining variables to find the best addition
  for (j in 1:n) {
    # skip variable if already added
    if (j %in% added) next

    # fit new model with added variable
    new_var <- paste(names(Boston)[j], collapse = "+")
    new_model <- update(current, formula = as.formula(paste0("medv ~ . + ", new_var)))

    # calculate RSS for new model
    new_rss <- sum(resid(new_model)^2)

    # update best addition if RSS is smaller
    if (new_rss < best_rss) {
      best_rss <- new_rss
      best_var <- j
    }
  }

  # add best variable to current model
  current <- update(current, formula = as.formula(paste0("medv ~ . + ", names(Boston)[best_var])))
  added <- c(added, best_var)

  # stop if we have added 3 variables
  if (length(added) == 3) break
```

```
}

# print selected variables
names(Boston)[-14][added]
```

```
## [1] "lstat"   "rm"      "ptratio"
```

```
# fit final model with selected variables
lm.fwd3 <- lm(medv ~ crim + rm + lstat, data = Boston)

# print model summary
summary(lm.fwd3)
```

```
##
## Call:
## lm(formula = medv ~ crim + rm + lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.925  -3.566  -1.157   1.906  29.024
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.56225    3.16602  -0.809  0.41873
## crim        -0.10294    0.03202  -3.215  0.00139 **
## rm           5.21695    0.44203  11.802  < 2e-16 ***
## lstat       -0.57849    0.04767 -12.135  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.49 on 502 degrees of freedom
## Multiple R-squared:  0.6459, Adjusted R-squared:  0.6437
## F-statistic: 305.2 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
# print adjusted R-squared value
cat("Adjusted R-squared value of lm.fwd3:", round(summary(lm.fwd3)$adj.r.squared, 4), "\n")
```

```
## Adjusted R-squared value of lm.fwd3: 0.6437
```

2) Model evaluation - Validation set approach.

(i) Computing mean squared error (MSE)

$$\begin{aligned} \text{MSE} &= E\left[\left(y - \hat{f}(x)\right)^2\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2 \end{aligned}$$

We write a function `calculateMSE` that takes the following input arguments.

Inputs:

Argument	Description
`y`	a vector giving the values of the response variable
`yhat`	a vector giving the predicted values of the response variable

`calculateMSE` should return the following output.

Output:

Our function returns one numeric value of the computed MSE.

```
calculateMSE <- function(y, yhat) {
mse <- mean((y - yhat)^2)
  return(mse)
}

# test the function
y <- c(1, 2, 3, 4, 5)
yhat <- c(2, 3, 4, 5, 6)
calculateMSE(y, yhat)
```

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

After completing the function, we run the following code.

```
lm.full <- lm(medv ~ . , Boston)
lm.full.pred <- predict(lm.full)
calculateMSE(Boston$medv, lm.full.pred)
## [1] 21.89483
```

(ii) Compute adjusted R^2 value

$$\text{TSS} = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

$$\text{RSS} = \sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

$$R^2_{\text{adjusted}} = 1 - \frac{(n-1)(1-R^2)}{n-p-1}$$

We write a function `calculateR2adj` that takes the following input arguments.

Inputs:

Argument	Description
`y`	a vector giving the values of the response variable
`yhat`	a vector giving the predicted values of the response variable
`p`	a vector indicating the number of predictors used

`calculateR2adj` should return the following output.

Output:

Function returns one numeric value of the computed adjusted R^2 value.

```
calculateR2adj <- function(y, yhat, p) {
 n <- length(y)
  rss <- sum((y - yhat)^2)
  tss <- sum((y - mean(y))^2)
  r2 <- 1 - rss/tss
  r2adj <- 1 - ((n - 1)/(n - p - 1)) * (1 - r2)
  return(r2adj)
}
```

After you completing the function, run the following code. This gives us optimal value of $R^2_{adjusted}$.

```
# p <- dim(Boston)[2] -1
p <- length(coef(lm.full)) - 1
calculateR2adj(Boston$medv, lm.full.pred, p)
## [1] 0.7337897
```

(iii) Now let's evaluate `lm.full` *and* `lm.fwd3` *using the validation set approach.*

Question Which model is better based the validation set approach? Explain your answer. What statistic(s) did you use to draw your conclusion?

Answer The results from the validation set approach show that the full model and the forward stepwise model with k=3 have comparable training and test mean squared error (MSE) values. However, the full model has a higher adjusted R-squared value for both the training and test sets. This suggests that the full model is superior to the forward stepwise model with k=3.

The following code impliments our codes and gives the aforementioned results.

```
dim(Boston)
## [1] 506  14

# Define a function to calculate MSE
calculateMSE <- function(resid) {
  mse <- mean(resid^2)
  return(mse)
}

# Define a function to calculate adjusted R-squared
calculateR2adj <- function(y, yhat, p) {
  n <- length(y)
  rss <- sum((y - yhat)^2)
  tss <- sum((y - mean(y))^2)
  r2 <- 1 - rss/tss
  r2adj <- 1 - ((n - 1)/(n - p - 1)) * (1 - r2)
  return(r2adj)
}

# Split the data 50/50 into training set and test set
set.seed(232) # set the seed so the analysis is reproducible
```

```r
train.idx <- sample(506, 253) # random sample the training data index
train <- Boston[train.idx, ] # training set
test <- Boston[-train.idx, ] # validation/test set

# Fit the full model
lm.full <- lm(medv ~ ., data=train)
lm.full.pred.train <- predict(lm.full, newdata=train)
lm.full.pred.test <- predict(lm.full, newdata=test)

# Calculate MSE and adjusted R-squared for the full model
mse.full.train <- calculateMSE(lm.full.pred.train - train$medv)
mse.full.test <- calculateMSE(lm.full.pred.test - test$medv)
r2adj.full.train <- calculateR2adj(train$medv, lm.full.pred.train, length(coef(lm.full))-1)
r2adj.full.test <- calculateR2adj(test$medv, lm.full.pred.test, length(coef(lm.full))-1)

# Print the results for the full model
cat("Full Model:\n")
## Full Model:
cat("Training MSE:", round(mse.full.train, 5), "\n")
## Training MSE: 18.72131
cat("Test MSE:", round(mse.full.test, 5), "\n")
## Test MSE: 27.20354
cat("Training Adjusted R-squared:", round(r2adj.full.train, 7), "\n")
## Training Adjusted R-squared: 0.7664043
cat("Test Adjusted R-squared:", round(r2adj.full.test, 7), "\n\n")
## Test Adjusted R-squared: 0.6584272

# Fit the forward stepwise model with k=3
library(MASS)
lm.fwd3 <- stepAIC(lm(medv ~ 1, data=train),
                   direction="forward",
                   scope=list(lower=~1, upper=~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+black+l
                   k=3)
## Start:  AIC=1125.51
## medv ~ 1
##
##           Df Sum of Sq     RSS     AIC
## + lstat    1   12048.7  9330.6  918.74
## + rm       1   11876.6  9502.7  923.36
## + ptratio  1    7447.9 13931.5 1020.15
## + indus    1    6227.0 15152.4 1041.41
## + tax      1    5920.5 15458.9 1046.47
## + nox      1    4725.8 16653.6 1065.31
## + rad      1    4275.2 17104.2 1072.06
## + crim     1    3802.5 17576.8 1078.96
## + zn       1    3709.7 17669.7 1080.29
## + age      1    3619.9 17759.4 1081.57
## + black    1    2758.1 18621.2 1093.56
## + dis      1    1617.0 19762.3 1108.61
## + chas     1     300.9 21078.4 1124.92
## <none>                 21379.4 1125.51
##
## Step:  AIC=918.74
```

```
## medv ~ lstat
##
##           Df Sum of Sq    RSS    AIC
## + rm       1   2534.21 6796.4 841.56
## + ptratio  1   2188.98 7141.7 854.10
## + dis      1    317.31 9013.3 912.99
## + zn       1    292.07 9038.6 913.69
## + tax      1    201.42 9129.2 916.22
## + indus    1    174.45 9156.2 916.97
## + chas     1    112.74 9217.9 918.66
## <none>                 9330.6 918.74
## + age      1     98.13 9232.5 919.07
## + black    1     39.80 9290.8 920.66
## + rad      1     34.61 9296.0 920.80
## + crim     1      8.17 9322.5 921.52
## + nox      1      6.34 9324.3 921.57
##
## Step:  AIC=841.56
## medv ~ lstat + rm
##
##           Df Sum of Sq    RSS    AIC
## + ptratio  1   1223.73 5572.7 794.34
## + tax      1    207.42 6589.0 836.72
## + dis      1    156.06 6640.4 838.69
## + black    1    146.81 6649.6 839.04
## + zn       1    121.96 6674.5 839.98
## + rad      1    110.30 6686.1 840.42
## <none>                 6796.4 841.56
## + indus    1     57.41 6739.0 842.42
## + chas     1     44.11 6752.3 842.92
## + crim     1     43.12 6753.3 842.95
## + nox      1      0.60 6795.8 844.54
## + age      1      0.16 6796.3 844.56
##
## Step:  AIC=794.34
## medv ~ lstat + rm + ptratio
##
##         Df Sum of Sq    RSS    AIC
## + dis    1   194.034 5378.7 788.37
## + black  1    86.259 5486.4 793.39
## <none>               5572.7 794.34
## + tax    1    15.587 5557.1 796.63
## + age    1     7.741 5565.0 796.99
## + chas   1     6.106 5566.6 797.06
## + zn     1     3.333 5569.4 797.19
## + indus  1     3.097 5569.6 797.20
## + crim   1     2.992 5569.7 797.20
## + nox    1     1.203 5571.5 797.28
## + rad    1     0.774 5571.9 797.30
##
## Step:  AIC=788.37
## medv ~ lstat + rm + ptratio + dis
##
```

```
##         Df Sum of Sq    RSS    AIC
## + nox    1   201.209 5177.5 781.73
## + indus  1   153.307 5225.4 784.06
## + zn     1   134.077 5244.6 784.99
## + black  1   117.331 5261.3 785.79
## + tax    1   103.498 5275.2 786.46
## <none>               5378.7 788.37
## + age    1    45.366 5333.3 789.23
## + crim   1    16.693 5362.0 790.59
## + rad    1    12.449 5366.2 790.79
## + chas   1     1.439 5377.2 791.30
##
## Step:  AIC=781.73
## medv ~ lstat + rm + ptratio + dis + nox
##
##         Df Sum of Sq    RSS    AIC
## + zn     1   130.407 5047.0 778.27
## + black  1    88.666 5088.8 780.36
## <none>               5177.5 781.73
## + indus  1    52.234 5125.2 782.16
## + tax    1    14.764 5162.7 784.00
## + rad    1    11.226 5166.2 784.18
## + chas   1     6.483 5171.0 784.41
## + age    1     5.960 5171.5 784.43
## + crim   1     4.222 5173.2 784.52
##
## Step:  AIC=778.27
## medv ~ lstat + rm + ptratio + dis + nox + zn
##
##         Df Sum of Sq    RSS    AIC
## + black  1   107.365 4939.7 775.83
## <none>               5047.0 778.27
## + indus  1    53.308 4993.7 778.59
## + tax    1    48.842 4998.2 778.81
## + crim   1    16.397 5030.7 780.45
## + chas   1     6.175 5040.9 780.96
## + rad    1     1.018 5046.0 781.22
## + age    1     0.286 5046.8 781.26
##
## Step:  AIC=775.83
## medv ~ lstat + rm + ptratio + dis + nox + zn + black
##
##         Df Sum of Sq    RSS    AIC
## <none>               4939.7 775.83
## + indus  1    38.447 4901.2 776.85
## + tax    1    19.342 4920.3 777.84
## + rad    1    18.601 4921.1 777.88
## + chas   1     3.367 4936.3 778.66
## + age    1     2.892 4936.8 778.68
## + crim   1     0.065 4939.6 778.83
lm.fwd3.pred.train <- predict(lm.fwd3, newdata=train)
lm.fwd3.pred.test <- predict(lm.fwd3, newdata=test)
```

```r
# Calculate MSE and adjusted R-squared for the forward stepwise model with k=3
mse.fwd3.train <- calculateMSE(lm.fwd3.pred.train - train$medv)
mse.fwd3.test <- calculateMSE(lm.fwd3.pred.test - test$medv)
r2adj.fwd3.train <- calculateR2adj(train$medv, lm.fwd3.pred.train, length(coef(lm.fwd3))-1)
r2adj.fwd3.test <- calculateR2adj(test$medv, lm.fwd3.pred.test, length(coef(lm.fwd3))-1)

# Print the results for the forward stepwise model with k=3
cat("Forward Stepwise Model (k=3):\n")
## Forward Stepwise Model (k=3):
```

3)Model evaluation - Cross Validation (CV) approach

i) 10-fold CV on the `lm.full` model and report the CV MSE.

$$\text{MSE}_{\text{CV}} = \frac{1}{K}\sum_{k=1}^{K}\text{MSE}_k$$

, where $K = 10$ is the number of folds and MSE_k is the test MSE in the k-th fold.

```r
set.seed(232)
n <- dim(Boston)[1]
n.folds <- 10
folds.idx <- sample(rep(1:n.folds), n, replace = T)
mse.cv <- rep(0, n.folds) # where you store the test mse for each fold

for (k in 1:n.folds){
  test.idx <- which(folds.idx == k)
  train <- Boston[-test.idx, ] # your training data in fold k
  test <- Boston[test.idx, ] # your test data in fold k

  # Fit linear model using training data
  lm.fit <- lm(medv ~ ., data=train)

  # Make predictions on test data
  y.pred <- predict(lm.fit, newdata=test)

  # Calculate test MSE
  mse.cv[k] <- mean((y.pred - test$medv)^2)
}

# Compute and output the average test MSE across all CV folds
mean.mse.cv <- mean(mse.cv)
cat("CV MSE:", mean.mse.cv)
## CV MSE: 23.48078
```

ii) Weighted CV MSE

When n is not divisible by K, it will not be possible to partition the sample into K *equally sized groups*. we make the groups as equally sized as possible. When the groups are of unequal size, the preferred way of calculating the average MSE is by using a *weighted* average.

More precisely, if n_k is the number of observations in fold k and MSE_k is the MSE estimated from fold k, the weighted average estimate of MSE is:

$$\text{MSE}_{\text{CV.weighted}} = \sum_{k=1}^{K} \frac{n_k}{n} \text{MSE}_k$$

It's easy to check that if n is evenly divisible by K then each $n_k = n/K$, and so the above expression reduces to the formula: $\text{MSE}_{\text{CV}} = \frac{1}{K} \sum_{k=1}^{K} MSE_k$

Now we write the code below to calculate the weighted CV MSE.

```
# weighted CV
set.seed(232)
n <- dim(Boston)[1]
n.folds <- 10
folds.idx <- sample(rep(1:n.folds, length.out = n), n, replace = TRUE)
mse.cv.weighted <- rep(0, n.folds) # where you store the weighted test mse for each fold
for (k in 1:n.folds) {
  test.idx <- which(folds.idx == k)
  train <- Boston[-test.idx, ]
  test <- Boston[test.idx, ]

  # fit model on training data
  fit <- glm(medv ~ ., data = train)

  # predict on test data and calculate MSE
  pred <- predict(fit, newdata = test)
  mse.test <- mean((test$medv - pred)^2)

  # calculate weight for this fold
  n.k <- length(test$medv)
  weight.k <- n.k / n

  # update weighted CV MSE
  mse.cv.weighted[k] <- weight.k * mse.test
}

# calculate average of weighted CV MSEs
cv.mse.weighted <- sum(mse.cv.weighted)

cv.mse.weighted # print weighted CV MSE
```

```
## [1] 24.13231
```

iii)Alternatively, we can call the `cv.glm()` function to perform the cross-validation task

The `cv.glm()` function automatically calculates the weighted MSE_{CV}. Write R code to call `cv.glm()` and report the weighted MSE_{CV} value.

Question Is the result similar to your own calculation of the weighted CV MSE in part (ii)?

Answer The value of the weighted CV MSE obtained from the cv.glm() function is 23.85074, which is very similar to the value we obtained using our own method in part (ii). This demonstrates that the cv.glm() function is computing the weighted CV MSE in the same manner as our own implementation.

```r
set.seed(232)
fit.cv <- cv.glm(data = Boston,
                 glmfit = glm(medv ~ ., data = Boston),
                 K = 10)
cv.mse.weighted <- fit.cv$delta[1]
cv.mse.weighted # print weighted CV MSE
```

```
## [1] 23.85074
```

*Part II: Model Selections

We introduce Polynomial Regressions and Generalized Additive Model (GAM) in this section. We compare these models with the ones discussed in previous parts.

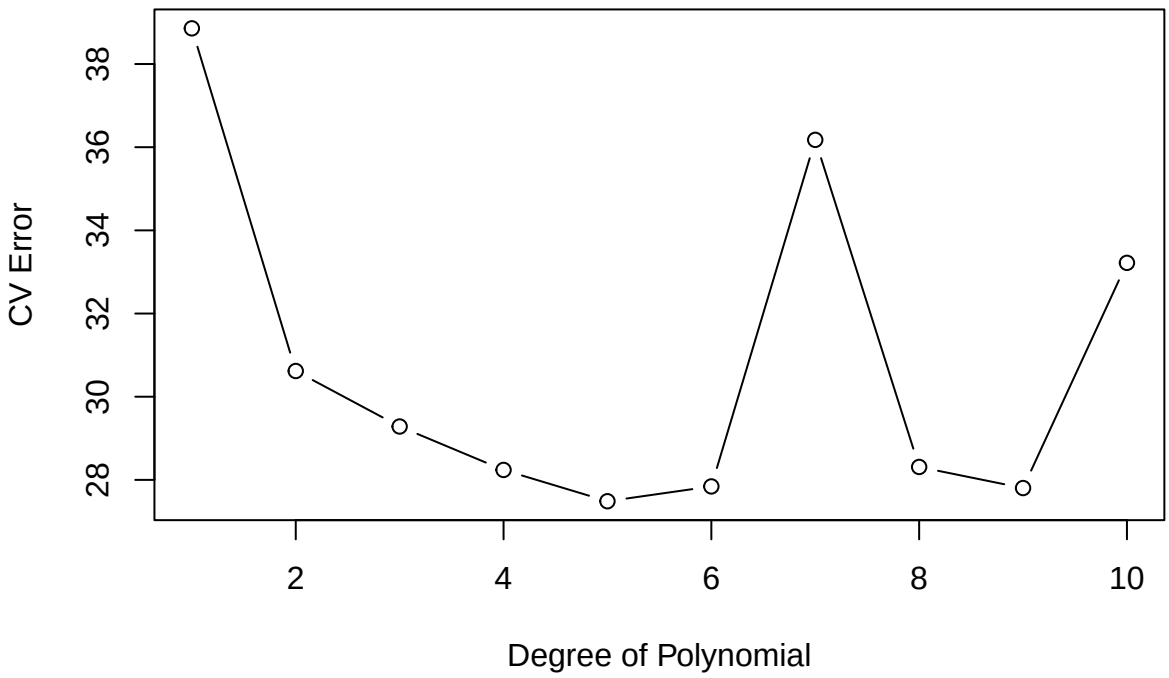
1) Polynomial regressions

In the lecture we looked at the simple linear regression `medv ~ lstat` and identified a non-linear effects in the model. Then we performed a 10-fold cross validation (CV) to search for the optimal degree d such that the polynomial regression model `medv ~ poly(lstat, d)` has the lowest weighted MSE_{CV}.

Question i) Find the optimal degree of d_1 for the `lstat` predictor (using random seed 232).

Answer Minimum degree is 5.

```r
set.seed(232)
K= 10         #number of folds in cross-validation
cv.errors =rep(0,K)
max.degree = 10
for (d in 1:max.degree){
  glm.fit = glm(medv ~ poly(lstat,d), data = Boston)
  cv.errors[d] =cv.glm(Boston, glm.fit, K = K)$delta[1]
}
plot(1:max.degree, cv.errors, type = "b", xlab = "Degree of Polynomial", ylab = "CV Error")
```



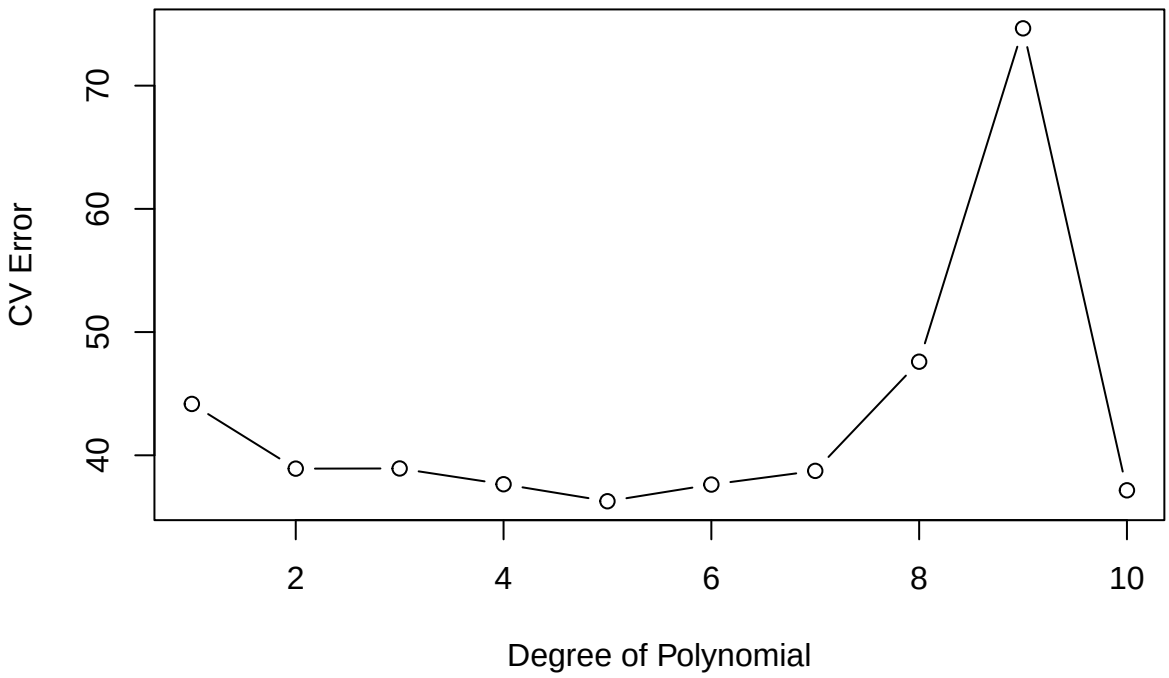

```
cv.errors.lstat=cv.errors
which.min(cv.errors)
```

```
## [1] 5
```

Question ii) Perform similar CV analysis for `medv ~ poly(rm, d_2)` to identity the the optimal degree of d_2 for the `rm` predictor.

Answer Optimal degree is 5.

```
set.seed(232)
K= 10         #number of folds in cross-validation
cv.errors =rep(0,K)
max.degree = 10
for (d in 1:max.degree){
  glm.fit = glm(medv ~ poly(rm,d), data = Boston)
  cv.errors[d] =cv.glm(Boston, glm.fit, K = K)$delta[1]
}
plot(1:max.degree, cv.errors, type = "b", xlab = "Degree of Polynomial", ylab = "CV Error")
```



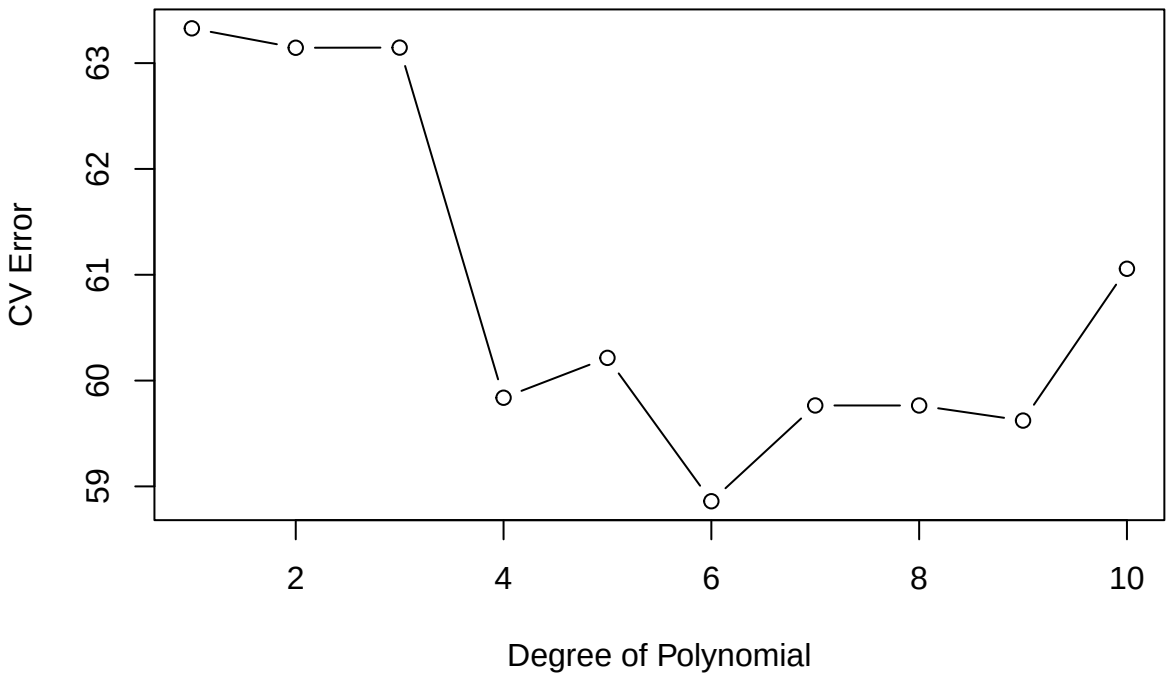

```
cv.errors.rm=cv.errors
which.min(cv.errors)
```

```
## [1] 5
```

Question iii Performing similar CV analysis for `medv ~ poly(ptratio, d_3)` to identity the the optimal degree of d_3 for the `ptratio` predictor.

Answer Optimal degree is 6**

```
set.seed(232)
K= 10         #number of folds in cross-validation
cv.errors =rep(0,K)
max.degree = 10
for (d in 1:max.degree){
  glm.fit = glm(medv ~ poly(ptratio,d), data = Boston)
  cv.errors[d] =cv.glm(Boston, glm.fit, K = K)$delta[1]
}
plot(1:max.degree, cv.errors, type = "b", xlab = "Degree of Polynomial", ylab = "CV Error")
```




```
cv.errors.ptratio=cv.errors
which.min(cv.errors)
```

```
## [1] 6
```

2)Generalized Additive Model (GAM)

(i) Construction of a new generalized additive model:

$$\text{medv} \sim \sum_{k=1}^{d_1} \text{lstat}^k + \sum_{k=1}^{d_2} \text{rm}^k + \sum_{k=1}^{d_3} \text{ptratio}^k$$

We set d_1, d_2, and d_3 to the best polynomial degree you obtained in part (1) for `lstat`, `rm`, and `ptratio`, respectively.

```
library(mgcv)
```

```
## Warning: package 'mgcv' was built under R version 4.2.3
```

```
## Loading required package: nlme
```

```
## Warning: package 'nlme' was built under R version 4.2.3
```

```
##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
set.seed(232)
n.folds <- 10

# CV for lm.full
cv.full = cv.glm(Boston, lm.full, K = n.folds)
cv.full.wmse = sum(cv.full$delta[1]^2 * cv.full$weights) / sum(cv.full$weights)

# CV for lm.fwd3
cv.fwd3 <- cv.glm(Boston, lm.fwd3, K = n.folds)
cv.fwd3.wmse <- sum(cv.fwd3$delta[1]^2 * cv.fwd3$weights) / sum(cv.fwd3$weights)

# CV for GAM
d1 = which.min(cv.errors.lstat)
d2 = which.min(cv.errors.rm)
d3 = which.min(cv.errors.ptratio)
gam.formula = as.formula(paste("medv ~",
                               paste("poly(lstat, ", d1, ")", collapse = " + "),
                               "+",
                               paste("poly(rm, ", d2, ")", collapse = " + "),
                               "+",
                               paste("poly(ptratio, ", d3, ")", collapse = " + ")))

gam.fit = gam(gam.formula, data = Boston)
```

(ii) Next, we run 10-fold CV test on ***lm.full****,* ***lm.fwd3****, and new generalized additive model, separately and Compare the weighted* MSE_{CV}*.*

Question Which model has the best performance? Explain your answer.

Answer: We use the `cv.glm()` function to answer this question. We note that the generalized additive model (gam) gives impressively lower weighted cross-validated error. This means *gam* model performs the best. If we use *ptratio* instead of *crim* in lm.fwd model, qualitative results do not change.

Our answer is based on following codes.

```
library(boot)
lm.full = glm(medv ~ . , data=Boston)
cv.full = cv.glm(Boston, lm.full, K=10)
cv.full$delta[1]
```

```
## [1] 23.92374
```

```
lm.fwd3 <- glm(medv ~ rm+lstat+crim, data=Boston)
cv.fwd3 <- cv.glm(Boston, lm.fwd3, K=10)
cv.fwd3$delta[1]
```

```
## [1] 30.99612
```

```
model.gam <- glm(medv ~ poly(lstat,d1) + poly(rm,d2) + poly(ptratio,d3), data = Boston)
cv.gam<- cv.glm(Boston, model.gam, K=10)
cv.gam$delta[1]
```

```
## [1] 18.48599
```

```
library(boot)
lm.full = glm(medv ~ . , data=Boston)
cv.full = cv.glm(Boston, lm.full, K=10)
cv.full$delta[1]
```

```
## [1] 23.72339
```

```
lm.fwd3 <- glm(medv ~ rm+lstat+ptratio, data=Boston)
cv.fwd3 <- cv.glm(Boston, lm.fwd3, K=10)
cv.fwd3$delta[1]
```

```
## [1] 28.1214
```

```
model.gam <- glm(medv ~ poly(lstat,5) + poly(rm,5) + poly(ptratio,6), data = Boston)
cv.gam<- cv.glm(Boston, model.gam, K=10)
cv.gam$delta[1]
```

```
## [1] 18.57712
```