STAT167 HW6 - Spring 2025

Ethan Choi

Contents

Homework #6 instructions

Review the ISLR textbook Chapter 4.3 “Logistic Regression”, Chapter 5.1 “Cross-Validation”, Chapter 6.1 “Subset Selection”, and Chapter 7.1 “Polynomial Regression”, as well as the lecture notes on nonlinear regressions, polynomial regression, model evaluation, and logistic regression before answering the homework questions.

This homework contains 3 question, each with multiple parts, 100 points in total.

Replace **INSERT_YOUR_ANSWER** with your own answers.

- First open this `rmd` file in RStudio and click `Knit -> Knit to PDF` to render it to PDF format. You need to have `LaTex` installed on the computer to render it to PDF format. If not, you can also render it to HTML format.
- It is best to read this `rmd` file and the rendered `pdf/html` file side-by-side, while you are working on this homework.
- If the question asks you to write some R code, remember to put your code into a **R code chunk**. Make sure both your R code chunk and its output are visible in the rendered `pdf/html` file.
- For this homework, use **ggplot2** to visualize your data.
- **Please comment your R code thoroughly, and follow the R coding style guideline (https://google.github.io/styleguide/Rguide.xml). Partial credit will be deducted for insufficient commenting or poor coding styles.**
- If you have any question about this homework assignment, we encourage you to post it on **Piazza**.

Homework submission guideline

- **This homework is DUE at *11:59 PM* on *Sunday May 25, 2025*.**
- Late submission penalties.
  - Submissions up to 24 hours late will incur a 10% deduction.
  - Submissions up to 48 hours late will incur a 30% deduction.
- **If you are using one or both of your free late days, please state here: INSERT_YOUR_ANSWER**
- After you complete all questions, save your `rmd` file to `FirstnameLastname-SID-HW6.rmd` and save the rendered pdf file to `FirstnameLastname-SID-HW6.pdf`. If you can not knit it to pdf, knit it to html first and then print/save it to pdf format.
- Submit **BOTH your source `rmd` file and the knitted `pdf` file** to **GradeScope**. Do NOT create a zip file. For the `pdf` submission, please tag specific pages that correspond with each question in the assignment.
- You can submit multiple times, you last submission will be graded.

Acknowledgements

Please list all the help you have received for completing this homework.

Some geeksforgeeks

Install necessary packages

Note that you only need to install each package once. Then you can comment out the following installation lines.

```r
#install.packages("MASS")
```

Load necessary packages

```r
library(tidyverse) # for `ggplot2`, `dplyr`, and more
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- 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
```

```r
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

```r
# set the random seed so the analysis is reproducible
set.seed(167) # do NOT change this number
```

Question 1 [55pt] Model Evaluation

Recall the `Boston` data set contains housing values in suburbs of Boston.

```r
?Boston # full documentation
## starting httpd help server ... done
dim(Boston)
## [1] 506  14
glimpse(Boston)
## Rows: 506
## Columns: 14
## $ crim    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829,~
## $ zn      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
## $ indus   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87, 7.~
## $ chas    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ nox     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524,~
## $ rm      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631,~
## $ age     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
## $ dis     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## $ rad     <int> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4,~
## $ tax     <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311, 31~
## $ ptratio <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## $ black   <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## $ lstat   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## $ medv    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~
```

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

```r
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
```

(a) [10pt] Variable/feature selection: forward selection

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

In Section 3.2.2 of the ISLR textbook, the authors described several strategies for selecting important/influential predictors. 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.

Write your R code to pick the top 3 most important variables using the *forward selection* strategy. Thoroughly comment your code.

Note that you are expected to write your own R code rather than using the `step()` function or other package(s).

If your code can automatically select the top 3 variables (which means you do not need to manually inspect the regression results and select the variables yourself), you will receive **EXTRA credit**.

Save the resulting model which uses only the top 3 variables to be `lm.fwd3`. What is the adjusted R^2 value of `lm.fwd3`?

```r
remaining_vars <- setdiff(names(Boston), "medv")
selected_vars <- c()
rss_values <- c()

for (i in 1:3) {
  rss_list <- c()

  for (var in setdiff(remaining_vars, selected_vars)) {
    current_vars <- c(selected_vars, var)
    formula <- as.formula(paste("medv ~", paste(current_vars, collapse = " + ")))
    model <- lm(formula, data = Boston)
    rss <- sum(residuals(model)^2)
    rss_list <- c(rss_list, rss)
  }

  best_var <- setdiff(remaining_vars, selected_vars)[which.min(rss_list)]
  selected_vars <- c(selected_vars, best_var)
  rss_values <- c(rss_values, min(rss_list))
}

formula_final <- as.formula(paste("medv ~", paste(selected_vars, collapse = " + ")))
lm.fwd3 <- lm(formula_final, data = Boston)
summary(lm.fwd3)$adj.r.squared
```

```
## [1] 0.6767036
```

(b) [20pt] Model evaluation - Validation set approach.

(i) [5pt] Compute **mean squared error (MSE)**

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

Recall in Homework 5 Question 2(c), you have written your own code to calculate RSE. Now write your own function to calculate the MSE.

Your function `calculateMSE` should take 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:

Your function should return one numeric value of the computed MSE.

INSERT_YOUR_ANSWER

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

After you complete the function, run the following code. You should expect a return value of 21.89483.

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

(ii) [5pt] Compute **adjusted R^2 value**

$$\begin{aligned} \text{TSS} &= \sum_{i=1}^{n}(y_i - \bar{y})^2 \\ \text{RSS} &= \sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2 \\ R^2 &= \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}} \\ R^2_{\text{adjusted}} &= 1 - \frac{(n-1)(1-R^2)}{n-p-1} \end{aligned}$$

Recall in Homework 5 Question 2(c), you have written your own code to calculate the R^2 statistics. Now write your own function to calculate the adjusted R^2 statistics.

Your function `calculateR2adj` should take 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:

Your function should return one numeric value of the computed adjusted R^2 value.

INSERT_YOUR_ANSWER

```r
calculateR2adj <- function(y, yhat, p) {
  n <- length(y)
  rss <- sum((y - yhat)^2)
  tss <- sum((y - mean(y))^2)
  r2 <- 1 - rss / tss
  r2_adj <- 1 - ((n - 1) / (n - p - 1)) * (1 - r2)
  return(r2_adj)
}
```

After you complete the function, run the following code. You should expect a return value of 0.7337897.

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

(iii) [10pt] Now let's evaluate `lm.full` and `lm.fwd3` using the validation set approach.

Complete the following code. For each model, report the training MSE, the test MSE, the training adjusted R^2 value, and the test adjusted R^2 value.

INSERT_YOUR_ANSWER

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

# split the data 50/50 into training set and test set
set.seed(167) # set the seed so the analysis is reproducible
train.idx <- sample(506, 253) # random sample the training data index
train <- Boston[train.idx, ] # training set
test <- Boston[-train.idx, ] # validation/test set

set.seed(167)
train.idx <- sample(1:nrow(Boston), nrow(Boston) / 2)
train <- Boston[train.idx, ]
test <- Boston[-train.idx, ]

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

mse.full.train <- calculateMSE(train$medv, pred.full.train)
mse.full.test <- calculateMSE(test$medv, pred.full.test)

r2adj.full.train <- calculateR2adj(train$medv, pred.full.train, length(coef(lm.full)) - 1)
r2adj.full.test <- calculateR2adj(test$medv, pred.full.test, length(coef(lm.full)) - 1)

formula_fwd3 <- as.formula(paste("medv ~", paste(selected_vars, collapse = " + ")))
lm.fwd3 <- lm(formula_fwd3, data = train)
pred.fwd3.train <- predict(lm.fwd3, newdata = train)
pred.fwd3.test <- predict(lm.fwd3, newdata = test)
```

```r
mse.fwd3.train <- calculateMSE(train$medv, pred.fwd3.train)
mse.fwd3.test <- calculateMSE(test$medv, pred.fwd3.test)

r2adj.fwd3.train <- calculateR2adj(train$medv, pred.fwd3.train, length(coef(lm.fwd3)) - 1)
r2adj.fwd3.test <- calculateR2adj(test$medv, pred.fwd3.test, length(coef(lm.fwd3)) - 1)

results <- data.frame(
  Model = c("Full", "Forward Selection"),
  Train_MSE = c(mse.full.train, mse.fwd3.train),
  Test_MSE = c(mse.full.test, mse.fwd3.test),
  Train_Adj_R2 = c(r2adj.full.train, r2adj.fwd3.train),
  Test_Adj_R2 = c(r2adj.full.test, r2adj.fwd3.test)
)
print(results)
##               Model Train_MSE Test_MSE Train_Adj_R2 Test_Adj_R2
## 1              Full  22.55266 23.36402    0.7146450   0.7118181
## 2 Forward Selection  29.77163 25.26567    0.6384329   0.7008779
```

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

Full is better. It has a lower Test MSE and higher Test R2

(c) [25pt] Model evaluation - Cross Validation (CV) approach

(i) [10pt] Complete the code below to perform a 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.

Note that you are expected to write your own R code rather than calling the `cv.glm()` function to answer this part.

INSERT__YOUR__ANSWER

```r
set.seed(167)
n <- nrow(Boston)
n.folds <- 10
folds.idx <- sample(rep(1:n.folds, length.out = n))
mse.cv <- numeric(n.folds)

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

  model <- lm(medv ~ ., data = train)
  pred <- predict(model, newdata = test)
  mse.cv[k] <- calculateMSE(test$medv, pred)
}

mean_cv_mse <- mean(mse.cv)
print(mean_cv_mse)
## [1] 23.96565
```

(ii) [10pt] Weighted CV MSE

When n is not divisible by K, it will not be possible to partition the sample into K *equally sized groups*. You should 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 you saw in class: $\text{MSE}_{\text{CV}} = \frac{1}{K}\sum_{k=1}^{K} MSE_k$

Complete the code below to calculate the weighted CV MSE.

Note that you are expected to write your own R code rather than calling the `cv.glm()` function to answer this part.

INSERT_YOUR_ANSWER

```r
set.seed(167)
n <- nrow(Boston)
n.folds <- 10
folds.idx <- sample(rep(1:n.folds, length.out = n))
mse.cv.weighted <- numeric(n.folds)
n_k <- numeric(n.folds)

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

  model <- lm(medv ~ ., data = train)
  pred <- predict(model, newdata = test)
  mse.cv.weighted[k] <- calculateMSE(test$medv, pred)
  n_k[k] <- length(test.idx)
}

weighted_cv_mse <- sum((n_k / n) * mse.cv.weighted)
print(weighted_cv_mse)
```

```
## [1] 23.97543
```

(iii) [5pt] Alternatively, you 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.

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

INSERT_YOUR_ANSWER

Question 2 [25pt] Polynomial Regressions and Generalized Additive Model (GAM)

(a) [15pt] 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}.

(i) [5pt] Repeat the CV analysis for `medv ~ poly(lstat, d_1)` and find the optimal degree of d_1 for the `lstat` predictor (using random seed 167).

You can use the `cv.glm()` function to answer this question.

INSERT_YOUR_ANSWER

(ii) [5pt] Perform similar CV analysis for `medv ~ poly(rm, d_2)` to identity the the optimal degree of d_2 for the `rm` predictor.

You can use the `cv.glm()` function to answer this question.

INSERT_YOUR_ANSWER

(iii) [5pt] Perform similar CV analysis for `medv ~ poly(ptratio, d_3)` to identity the the optimal degree of d_3 for the `ptratio` predictor.

You can use the `cv.glm()` function to answer this question.

INSERT_YOUR_ANSWER

(b) [10pt] Generalized Additive Model (GAM)

(i) [5pt] Now, let's construct a new generalized additive model:

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

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

INSERT_YOUR_ANSWER

(ii) [5pt] Next, Run 10-fold CV test on `lm.full`, `lm.fwd3`, and your new generalized additive model, separately.

Compare the weighted MSE_{CV}. Which model has the best performance? Explain your answer.

You can use the `cv.glm()` function to answer this question.

INSERT_YOUR_ANSWER

Question 3 [20pt] Logistic Regression

In the above questions, we have applied a few regression models to predict median housing values using the `Boston` data set.

In this question, we are interested in building a classification model to predict whether a given suburb in Boston has a crime rate above or below the median.

(a) [5pt] First, calculate the median crime rate. Add a binary variable to the `Boston` data set which indicates whether a given suburb has a crime rate above or below the median.

INSERT_YOUR_ANSWER

(b) [5pt] Then, build a multiple logistic regression model using a full set of 13 other predictors. Note that you should NOT use the original `crim` variable as a predictor.

INSERT_YOUR_ANSWER

(c) [5pt] Describe your findings. What can you learn from this logistic regression model?

INSERT_YOUR_ANSWER

(d) [5pt] Use the `predict(..., type="response")` function to calculate the estimated conditional probabilities of the crime rate above the median. Then use the simple Bayes rule (if the conditional probability is greater than 0.5, classify it as above the median; otherwise, below the median). What is the (training) mis-classification rate?

INSERT_YOUR_ANSWER
