STAT167 Lab #8 - Spring 2025

Ethan Choi

2025/5/23

Contents

Discussion/Lab #8 instructions

This week, we will review example code of model validation approaches, and classification method (logistic regression).

- First, download the `rmd` file from Canvas.
- 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.

- Read this `rmd` file and the rendered `pdf`/`html` file side-by-side, to see how this document was generated!
- Be sure to play with this document! Change it. Break it. Fix it. The best way to learn R Markdown (or really almost anything) is to try, fail, then find out what you did wrong.
- Read over the example code and the output. If you have any questions about certain functions or parameters, it is the time to ask!
- There are some exercises through out this document. Replace **INSERT_YOUR_ANSWER** with your own answers. Knit the file, and check your results.

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.

Lab submission guideline

- After you completed all exercises, save your file to `FirstnameLastname-SID-lab8.rmd` and save the rendered pdf file to `FirstnameLastname-SID-lab8.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.
- You can submit multiple times, you last submission will be graded.

Install necessary packages

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

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

Load necessary package

```
library(MASS) # for the `Boston` dataset
library(ISLR) # for `Default` dataset
library(tidyverse) # for `ggplot2`, `dplyr`, `tibble`, 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()
## x dplyr::select() masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
library(class) # for `knn()`
library(boot) # for cv.glm()
```

Set the random seed

```r
# set the random seed so that the analysis is reproducible
set.seed(167)
```

The Boston dataset

The `Boston` dataset (included in the `MASS` library) contains housing values in 500+ suburbs of Boston.

```r
?Boston # full documentation
## starting httpd help server ... done
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~
```

Lecture Review - Model evaluation approaches

Model evaluation approaches aim to access how well a model would perform on a new set of test data. Using model evaluation approaches, we can compare multiple models using the same benchmark dataset.

The commonly used model evaluation strategies include:

- validation set approach
- leave-one-out cross-validation (LOOCV)
- k-fold cross-validation

Validation set approach

The Validation set approach includes the following steps:

- Randomly divide the available data into two parts: a **training set** and a **validation set** or **test set**.

- Construct $\hat{f}$ by fitting your model on the **training set**
- Use $\hat{f}$ to predict responses for all points in the **validation/test set**, and calculate the resulting MSE
- Pick the simplest model that has among the lowest MSE on the **validation/test set**

Notes:

- For regression tasks, the commonly used model metric is MSE, but it can be replaced by R^2, or adjusted R^2, or other model metric.
- In the ILSR textbook, we split the data by 50/50. But it is also common to split the data 70/30 or 80/20. There is a tradeoff between the model accuracy and generalizability.

Validation set approach example - `Boston` dataset

In the lecture, we applied the validation set approach to the `Boston` data.

```
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

# fit/train the model using the training dataset
lm.train <- lm(medv ~ lstat, data = train)
#lm.train <- lm(medv ~ lstat, data = Boston, subset = train.idx)

# calculate MSE on the training set
mean((train$medv - predict(lm.train))^2)
## [1] 39.23454

# calculate MSE on the validation/test set
mean((test$medv - predict(lm.train, newdata=test))^2)
## [1] 37.74987
```

Exercise #1

(a) Now let's apply the validation set approach to evaluate the multiple linear regression model `lm(medv ~ .)`, where we use all the 13 predictors to predict the response variable `medv`. What are the training set MSE and validation/test set MSE?

```
lm.full <- lm(medv ~ ., data = train)
mean((train$medv - predict(lm.full))^2)
```

```
## [1] 22.55266
```

```r
mean((test$medv - predict(lm.full, newdata = test))^2)
```

```
## [1] 23.36402
```

Train MSE is 17.01206 Test MSE is 30.32567

(b) Next let's apply the validation set approach to evaluate the degree-6 polynomial regression model `lm(medv ~ poly(lstat, 6))`. What are the training set MSE and validation/test set MSE?

```r
lm.poly6 <- lm(medv ~ poly(lstat, 6), data = train)
mean((train$medv - predict(lm.poly6))^2)
```

```
## [1] 29.43424
```

```r
mean((test$medv - predict(lm.poly6, newdata = test))^2)
```

```
## [1] 24.38078
```

Train - 23.20324 Test - 32.63614

(c) Which regression model is better? Explain your answer. What statistic did you use to draw your conclusion?

The multiple linear regression model is better. It has a lower validation/test MSE, which is the focus variable for generalization.

Cross-validation (CV) approach

Cross-validation is the **most widely used approach** for model evaluation!!

The k-fold cross-validation (CV) approach includes the following steps

- Randomly split the data into k equal-size parts ("folds")
- Give each part the chance to be the **validation set**, treating the other $k-1$ parts (combined) as the **training set**.
  - Construct $\hat{f}$ by fitting your model on the **training set**
  - Use $\hat{f}$ to predict responses for all points in the **validation set**, and calculate the resulting MSE_{test}
- Average the MSE_{test} over all the folds
- Pick the simplest model among those with lowest MSE_{CV}
- **Final model**: Refit the model on the entire dataset.

Notes:

- For regression tasks, the commonly used model metric is MSE, but it can be replaced by R^2, or adjusted R^2, or other model metric.
- Most common choices of k: 5, 10, n. The special case $k = n$ is called leave-one-out cross-validation (LOOCV).

Cross-validation (CV) approach example - `Boston` dataset

We can use the `cv.glm()` function to calculate the cross-validation MSE. Below is the example code for calculating the 10-fold CV MSE of the full multiple linear regression model on the `Boston` data.

```
glm.full <- glm(medv ~ . , data = Boston)
summary(glm.full)
##
## Call:
## glm(formula = medv ~ ., data = Boston)
##
## 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
##
## (Dispersion parameter for gaussian family taken to be 22.51785)
##
##     Null deviance: 42716  on 505  degrees of freedom
## Residual deviance: 11079  on 492  degrees of freedom
## AIC: 3027.6
##
## Number of Fisher Scoring iterations: 2

cv.glm(Boston, glm.full, K = 10)$delta[1]
## [1] 23.72169
```

The `Default` dataset

The `Default` dataset (included in the `ISLR` library) contains 10,000 credit card customer information.

```
?Default # full documentation
glimpse(Default)
## Rows: 10,000
## Columns: 4
## $ default <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No, No, No~
## $ student <fct> No, Yes, No, No, No, Yes, No, Yes, No, No, Yes, Yes, No, No, N~
## $ balance <dbl> 729.5265, 817.1804, 1073.5492, 529.2506, 785.6559, 919.5885, 8~
## $ income  <dbl> 44361.625, 12106.135, 31767.139, 35704.494, 38463.496, 7491.55~
```

Lecture Review - Classifications

Logistic regression

In the lecture, we have learned that logistic regression aims to fit the **logit** function of the conditional probability $P(Y=1 \mid \boldsymbol{X})$ using a linear function of predictors.

$$\log \frac{P(Y=1 \mid \boldsymbol{X})}{1-P(Y=1 \mid \boldsymbol{X})} = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$$

Computationally we can call `glm()` with the argument `family = binomial()` to perform logistic regression.

```
logit.fit.all <- glm(default ~ ., family = binomial(), data = Default)
summary(logit.fit.all)
##
## Call:
## glm(formula = default ~ ., family = binomial(), data = Default)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
## studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
## balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
## income       3.033e-06  8.203e-06   0.370  0.71152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1579.5
##
## Number of Fisher Scoring iterations: 8
```

Given a data point $\boldsymbol{x} = (x_1, \cdots, x_p)$, we can predict the conditional probability $p(x) = P(Y=1 \mid \boldsymbol{X} = \boldsymbol{x})$ using the estimated model coefficients.

$$\hat{p}(\boldsymbol{x}) = \frac{e^{\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j}}{1 + e^{\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j}}$$

Then we can use $\hat{p}(\boldsymbol{x})$ to classify our test data.

$$\hat{y} = \begin{cases} 1 & \text{if } \hat{p}(\boldsymbol{x}) > \alpha \\ 0 & \text{if } \hat{p}(\boldsymbol{x}) \le \alpha \end{cases}$$

Exercise #2

To obtain the logistic regression predictions, we can use the `predict()` function.

```r
logit.fit.pred <- predict(logit.fit.all)
head(logit.fit.pred)
##         1         2         3         4         5         6 
## -6.549544 -6.791338 -4.614261 -7.724689 -6.245449 -6.217871
```

(a) Why there are negative values in `logit.fit.pred`?

Hint: Read `predict.glm` help page. What is the the `type` argument for?

logit.fit.pred returns logit values that can be negative, positive, or zero.

(b) Use `logit.fit.pred` to calculate the conditional probability $p(\boldsymbol{x}) = P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{e^{\hat{\beta}_0+\sum_{j=1}^{p}\hat{\beta}_j x_j}}{1+e^{\hat{\beta}_0+\sum_{j=1}^{p}\hat{\beta}_j x_j}}$, which should be between 0 and 1.

```r
logit.fit.manual.prob <- exp(logit.fit.pred) / (1 + exp(logit.fit.pred))
head(logit.fit.manual.prob)
```

```
##            1            2            3            4            5            6 
## 0.0014287239 0.0011222039 0.0098122716 0.0004415893 0.0019355062 0.0019895182
```

(c) You can also use the `predict(..., type = "response")` function to directly extract the conditional probability $p(\boldsymbol{x}) = P(Y = 1 \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{e^{\hat{\beta}_0+\sum_{j=1}^{p}\hat{\beta}_j x_j}}{1+e^{\hat{\beta}_0+\sum_{j=1}^{p}\hat{\beta}_j x_j}}$.

Compare the conditional probabilities you calculated in part (b) with the following `predict(..., type = "response")` output. Are they the same?

```r
logit.fit.prob <- predict(logit.fit.all, type = "response")
head(logit.fit.prob)
##            1            2            3            4            5            6 
## 0.0014287239 0.0011222039 0.0098122716 0.0004415893 0.0019355062 0.0019895182
```

```r
logit.fit.prob <- predict(logit.fit.all, type = "response")
all.equal(logit.fit.manual.prob, logit.fit.prob)
```

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

Yes, they are the same (or at least, very very similar)

(d) Once we obtain the conditional probability `logit.fit.prob`, we can use the `ifelse()` function to classify the default status with the simple Bayes rule with $\alpha = 0.5$. That is,

$$\hat{y} = \begin{cases} 1 & \text{if } \hat{p}(\boldsymbol{x}) > 0.5 \\ 0 & \text{if } \hat{p}(\boldsymbol{x}) \leq 0.5 \end{cases}$$

```
logit.fit.class <- ifelse(logit.fit.prob > 0.5, "Yes", "No") %>% as.factor()
```

Calculate the **misclassification error rate** of the logistic classification results.

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y_i})$$

```
logit.fit.class <- ifelse(logit.fit.prob > 0.5, "Yes", "No") %>% as.factor()
mean(logit.fit.class != Default$default)
```

```
## [1] 0.0268
```
