Lab2. Cross Validation

2024-10-9

Read in data and take a look

```
cars <- Auto  # data loaded in the ISLR package
head(cars,3)  # take a look
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
```

Validation-set approach

Step 1: Getting a train/test split.

```
# Set seed for reproducibility
set.seed(7304) # generated at random.org

#  Divide into training and test sets
train_inds <- caret::createDataPartition(
  y = cars$mpg, # response variable as a vector
  p = 0.5       # approx. proportion of data used for training
)

# train_inds: indices of selected observations for training set
head(train_inds)
```

```
## $Resample1
##   [1]   1   2   3   5   9  11  16  21  22  24  26  27  31  34  37  39  44  47
##  [19]  48  50  51  52  53  55  57  59  60  63  65  66  67  70  72  73  74  76
##  [37]  78  80  82  83  84  85  86  87  88  92  97  98 100 101 103 104 106 108
##  [55] 109 110 112 114 117 118 120 121 123 124 126 127 128 129 130 131 136 138
##  [73] 139 140 141 143 144 145 147 151 152 155 156 158 159 160 162 163 165 166
##  [91] 168 169 171 172 175 176 177 178 183 185 186 187 188 193 194 195 197 199
## [109] 205 206 208 210 211 213 216 218 219 220 221 222 225 226 227 228 229 230
## [127] 231 233 235 239 242 243 249 250 253 254 259 262 271 272 273 276 280 282
## [145] 291 292 293 296 298 302 303 304 308 310 311 315 317 318 320 321 323 326
## [163] 327 328 329 330 331 332 334 337 340 344 347 348 349 350 351 352 357 358
## [181] 359 360 361 362 365 367 369 371 372 374 377 380 385 386 387 388 390 391
```

```r
# Create the training and test data sets
cars_train <- cars %>% slice(train_inds[[1]])
cars_test  <- cars %>% slice(-train_inds[[1]])
```

Summary of what we've achieved so far:

```r
# number of observations in each data sets
nrow(cars)
```

```
## [1] 392
```

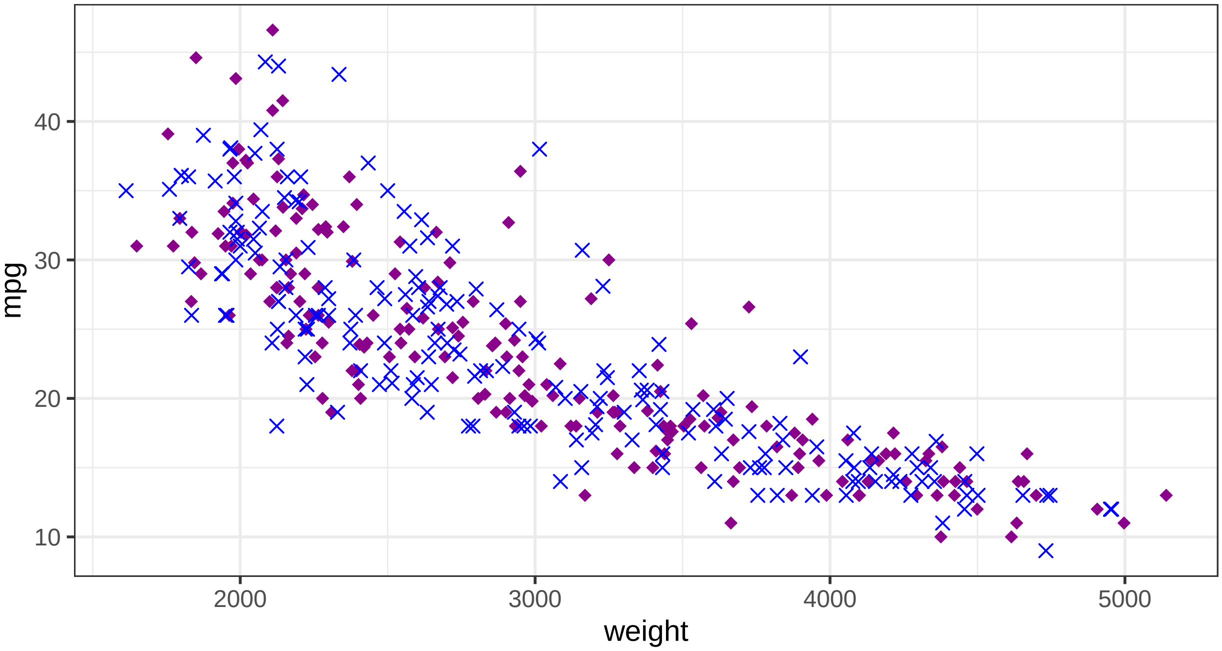
```r
nrow(cars_train)
```

```
## [1] 198
```

```r
nrow(cars_test)
```

```
## [1] 194
```

```r
# plot of the data
ggplot() +
  geom_point(data = cars_train,
             mapping = aes(x = weight, y = mpg),
             color = "magenta4", shape = 18, size =2) +
  geom_point(data = cars_test,
             mapping = aes(x = weight, y = mpg),
             color = "blue", shape = 4, size =2) +
  theme_bw()
```



Step 2: Fit all candidate models to the training data and compare performance on test data.

Here, we get validation set MSE for each candidate model:

```
results_mse <- data.frame(
  poly_degree = seq_len(7),
  train_mse = NA,
  test_mse = NA
)

for(degree in seq_len(7)) {

  # fit to training set
  fit <- lm(mpg ~ poly(weight, degree), data = cars_train)

  # by default, predictions are for training set
  # get residuals and mse for training set
  train_resids <- cars_train$mpg - predict(fit)
  results_mse$train_mse[degree] <- mean(train_resids^2)

  # get residuals and mse for test set
  test_resids <- cars_test$mpg - predict(fit, cars_test)
  results_mse$test_mse[degree] <- mean(test_resids^2)
}
```

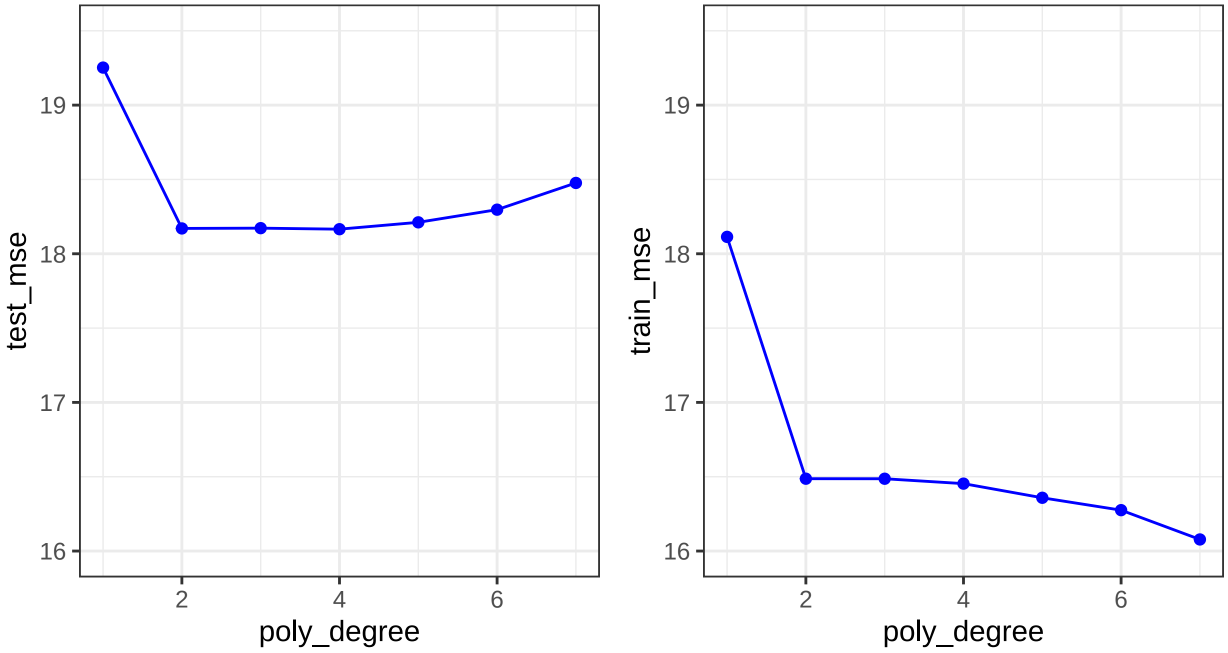
Here's a plot of the results:

```
# Code to find the limits of the y axis (not required)
mse_all   <- c(results_mse$train_mse,results_mse$test_mse)
plot_ylim <- c(floor(min(mse_all)*4)/4, ceiling(max(mse_all)*4)/4)

# Make plots of the results!
p1 <- ggplot(data = results_mse,
       mapping = aes(x = poly_degree, y = test_mse)) +
  geom_line(color="blue") +
  geom_point(color="blue") +
  ylim(plot_ylim) +
  theme_bw()

p2 <- ggplot(data = results_mse,
       mapping = aes(x = poly_degree, y = train_mse)) +
  geom_line(color="blue") + geom_point(color="blue") +
  ylim(plot_ylim) +
  theme_bw()

grid.arrange(p1,p2,ncol= 2)
```



Which model would you prefer based on this analysis? Are the shape of these plots aligned with your understanding?

2. k-fold Cross-Validation

Step 1: Split into training and test sets, obtain validation folds

```r
# Set seed for reproducibility
set.seed(7304) # generated at random.org

# Generate partition of the 5 folds
# The result is a list of length 5 with indices of observations to include in each fold.
num_crossval_folds <- 5
cross_fold_inds <- caret::createFolds(
  y = cars$mpg,     # response variable as a vector
  k = num_crossval_folds # number of folds for CV
)
```

Step 2: Get performance for each fold, using the other folds put together as a training set.

```r
# Object to store the results
results_mse <- expand.grid(
  poly_degree = seq_len(7),
  fold_num    = seq_len(num_crossval_folds),
  train_mse   = NA,
  test_mse    = NA
)
# For loops:
#    7 polynomial degrees (outside loop)
#    5 model fits for the 5 folds (inside loop)

for(poly_degree in seq_len(7)) { # degrees
  for(fold_num in seq_len(num_crossval_folds)) { # folds

    # Index where to store results
    results_index <- which(
      results_mse$poly_degree == poly_degree &
      results_mse$fold_num    == fold_num
    )

    # Training and testing sets (depends on the fold)
    cars_train <- cars %>% slice(-cross_fold_inds[[fold_num]])
    cars_test  <- cars %>% slice(cross_fold_inds[[fold_num]])

    # Fit model to training data (depends on the degree)
    fit <- lm(mpg ~ poly(weight, poly_degree), data = cars_train)

    # Get training set MSE
    train_resids <- cars_train$mpg - predict(fit)
    results_mse$train_mse[results_index] <- mean(train_resids^2)
    # Get testing set MSE
    test_resids  <- cars_test$mpg - predict(fit, cars_test)
    results_mse$test_mse[results_index]  <- mean(test_resids^2)
  }
}
head(results_mse)
```

```
##   poly_degree fold_num train_mse test_mse
## 1           1        1  19.21966 16.52778
## 2           2        1  17.49951 16.63847
## 3           3        1  17.49933 16.64132
## 4           4        1  17.45211 16.75784
## 5           5        1  17.37603 16.90866
## 6           6        1  17.34381 16.89780
```

```
# summarize the results from cross validation
# need to take the average mse for the k folds
summarized_crossval_mse_results <- results_mse %>%
  group_by(poly_degree) %>%
  summarize(
    crossval_mse = mean(test_mse)
  )
summarized_crossval_mse_results
```

```
## # A tibble: 7 x 2
##   poly_degree crossval_mse
##         <int>        <dbl>
## 1           1         18.9
## 2           2         17.6
## 3           3         17.6
## 4           4         17.6
## 5           5         17.6
## 6           6         17.8
## 7           7         17.6
```

These results suggest that polynomials of degree 2 to 5 and 7, have similar performance.

Using pre-built code from R

```r
# write our own function to add the predictions to the data set
get_pred  <- function(model, test_data){
  data  <- as.data.frame(test_data)
  pred  <- add_predictions(data, model)
  return(pred)
}
#Create the cross validation folds
cv       <- crossv_kfold(cars, k = 5)
MSE_models <- rep(NA, 7)

for (poly_degree in seq_len(7)){
  # fit the model to the k-1 training folds
  model_fit  <- map(cv$train,
                    ~lm(mpg ~ poly(weight,poly_degree),data = .))

  # get predictions for the testing fold
  pred_test  <- map2_df(model_fit, cv$test, get_pred, .id = "Run")

  #Get MSE for each k folds
  MSE_test   <- pred_test %>% group_by(Run) %>%
    summarise(MSE = mean( (mpg - pred)^2))

  #Store the results
  MSE_models[poly_degree] <- mean(MSE_test$MSE)

}
summarize_results <- data.frame(poly_degree = seq_len(7),
                                MSE = MSE_models)
summarize_results
```

```
##   poly_degree      MSE
## 1           1 19.02630
## 2           2 17.83471
## 3           3 17.99142
## 4           4 18.07091
## 5           5 17.99006
## 6           6 18.01623
## 7           7 17.80788
```

3. Leave-One-Out Cross-Validation

How could you implement the leave-one-out CV method?