K-Nearest Neighbors and Cross-Validation

CH5. Cross Validation

The Auto was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition. The original dataset has 397 observations, of which 5 have missing values for the variable "horsepower". These rows are removed here.

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
```

Use the following approaches to determine the number of nearest neighbors (K) to use in the KNN regression method:

- Validation set method
- 10-fold Cross-Validation

You may get some inspiration for the steps to follow from "CrossValidation_RCode.Rmd".

Validation-set approach

Step 1: Getting a train/test split.

```
# Set seed for reproducibility
set.seed(7304)

#  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, 10)
```

```
## $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
```

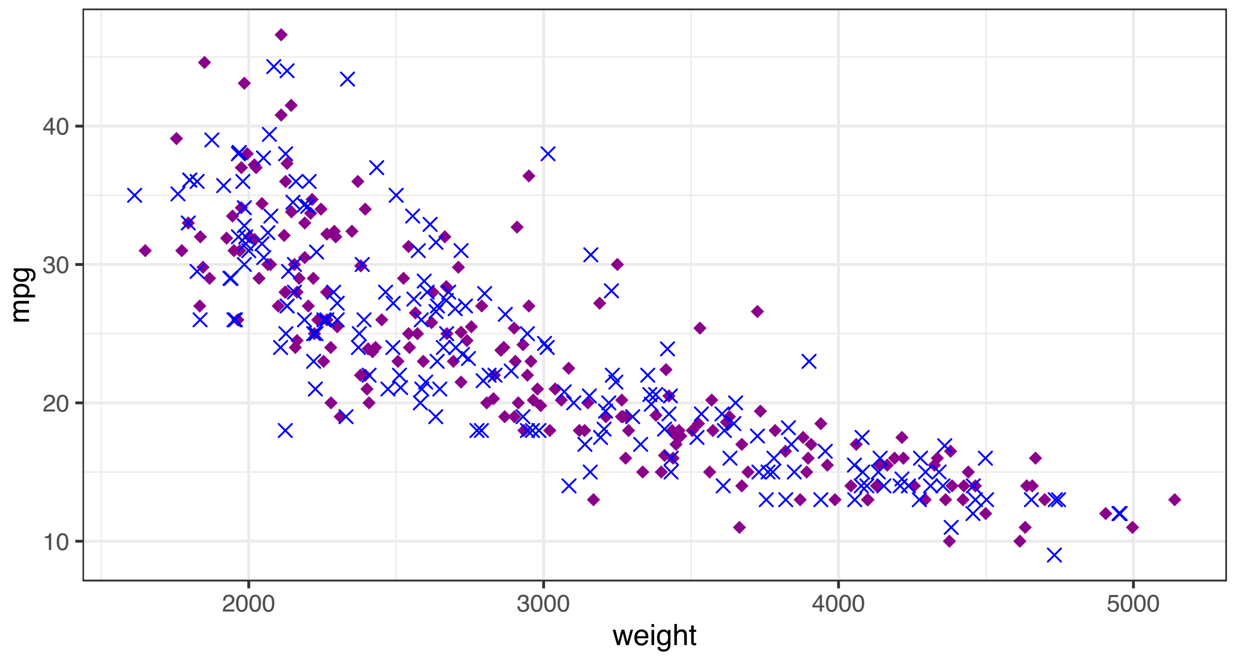
```r
nrow(cars_train)
```

```
## [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()
```


mpg
40
30
20
10
2000
3000
4000
5000
weight

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:

```r
K_neighbors <- c(1, 3, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 100)
length(K_neighbors)
```

```
## [1] 13
```

```r
results_mse <- data.frame(
  K_neighbors = K_neighbors,
  train_mse = NA,
  test_mse = NA
)

# For loops:
for(i in 1:length(K_neighbors)) {

  # fit to training set
  knn_fit <- train(form = mpg ~ weight,
                   data = cars_train,
                   method = "knn",
                   trControl = trainControl(method = "none"),
                   tuneGrid = data.frame(k = K_neighbors[i]))

  # by default, predictions are for training set
  # get residuals and mse for training set
  train_resids <- cars_train$mpg - predict(knn_fit)
  results_mse$train_mse[i] <- mean(train_resids^2)
  # get residuals and mse for test set
  test_resids <- cars_test$mpg - predict(knn_fit, newdata = cars_test)
  results_mse$test_mse[i] <- mean(test_resids^2)
}
data.frame(K_neighbors, results_mse$test_mse)
```

```
##    K_neighbors results_mse.test_mse
## 1            1             33.54798
## 2            3             23.81516
## 3            5             21.19182
## 4           10             19.51819
## 5           15             18.26038
## 6           20             18.20265
## 7           25             18.17735
## 8           30             18.29190
## 9           35             18.49757
## 10          40             18.54106
## 11          45             18.53618
## 12          50             18.49696
## 13         100             23.69861
```

$K = 25$ has the lowest MSE_{te}.

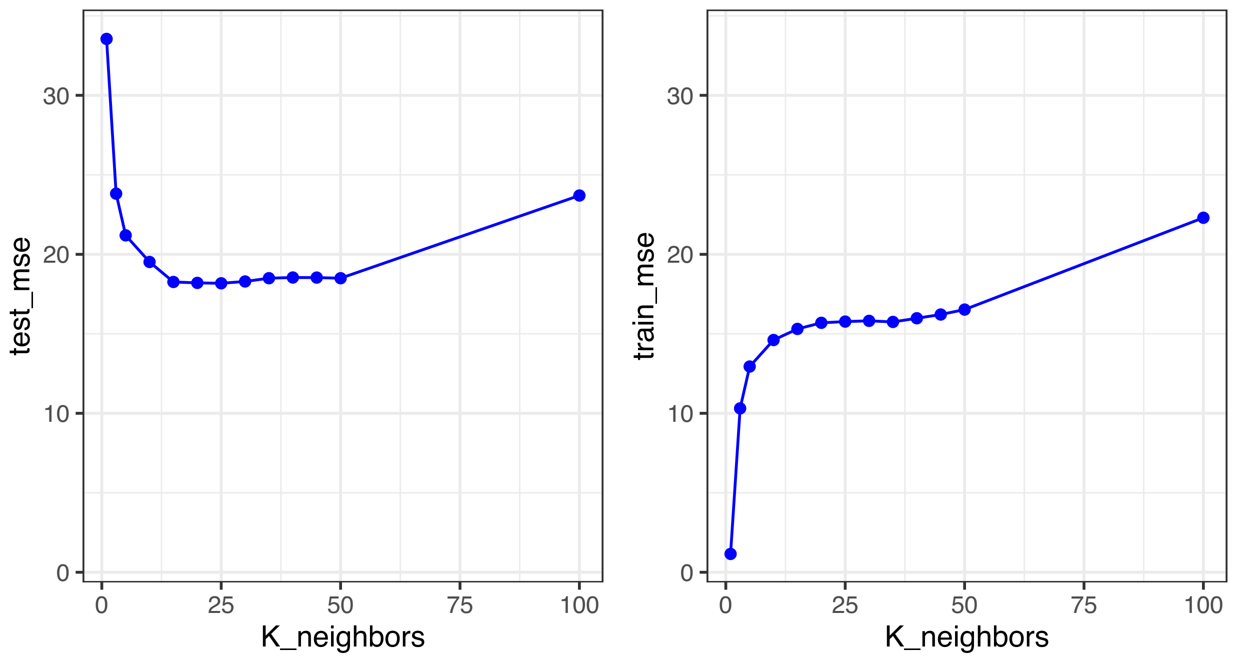
Here's a plot of the results:

```r
# 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 = K_neighbors, y = test_mse)) +
  geom_line(color="blue") +
  geom_point(color="blue") +
  ylim(plot_ylim) +
  theme_bw()

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

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



For KNN method, the smaller number of neighbors make the model to be more complex than the larger number of neighbors.

10-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 10 folds
# The result is a list of length 10 with indices of observations to include in each fold.
num_crossval_folds <- 10
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(
  K_neighbors = K_neighbors,
  fold_num    = seq_len(num_crossval_folds),
  train_mse = NA,
  test_mse = NA
)
# For loops:
#    13 different number of neighbors (outside loop)
#    10 model fits for the 10 folds (inside loop)

for(i in 1:length(K_neighbors))  { # K neighbors
  for(fold_num in seq_len(num_crossval_folds)) { # folds

    # Index where to store results
    results_index <- which(
      results_mse$K_neighbors == K_neighbors[i] &
      results_mse$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)
    knn_fit2 <- train(form = mpg ~ weight, data = cars_train,
                      method = "knn",
                      trControl = trainControl(method = "none"),
                      tuneGrid = data.frame(k = K_neighbors[i]))

    # get residuals and mse for training set
    train_resids <- cars_train$mpg - predict(knn_fit2)
    results_mse$train_mse[results_index] <- mean(train_resids^2)
    # get residuals and mse for test set
    test_resids <- cars_test$mpg - predict(knn_fit2, cars_test)
    results_mse$test_mse[results_index] <- mean(test_resids^2)
  }
}
head(results_mse)
```

```
##   K_neighbors fold_num train_mse test_mse
## 1           1        1  2.073407 37.96669
## 2           3        1 10.828603 37.18885
## 3           5        1 13.063618 31.86305
## 4          10        1 14.863174 30.45482
## 5          15        1 14.906983 30.99602
## 6          20        1 15.485910 28.34440
```

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

```
## # A tibble: 13 x 2
##    K_neighbors crossval_mse
##          <dbl>        <dbl>
##  1           1         32.4
##  2           3         23.2
##  3           5         20.1
##  4          10         19.1
##  5          15         18.5
##  6          20         18.2
##  7          25         18.0
##  8          30         17.9
##  9          35         17.5
## 10          40         17.4
## 11          45         17.5
## 12          50         17.5
## 13         100         18.3
```

$K = 40$ has the lowest MSE_{te}.