Lesson 10 k-fold Cross-Validation

Kevin Cummiskey

February 3, 2020

Review

Last lesson, we discussed three models to predicts the number of wins in a season.

$$Wpct = \beta_0 + \beta_1 RD + \epsilon \tag{1}$$

$$Wpct = \frac{R^2}{R^2 + RA^2} + \epsilon \tag{2}$$

$$Wpct = \frac{R^k}{R^k + RA^k} + \epsilon \tag{3}$$

where $Wpct$ is Win Percentage, R is Runs Scored, RA is Runs Allowed, and ϵ is the random error. Recall that we fit Model 1 to the 1997-2001 seasons.

When prediction is the goal of a statistical model, *overfitting* a model is a big concern. What is *overfitting*?

k-fold cross validation

k-fold cross validation is one method to assess how well a model will predict on new data. In this method, we randomly partition the data into k groups. We set one group (called the *test set*) aside, fit the model on the remaining data (the *training set*), and assess the performance on the test set. We repeat the process k times with each partition as the test set.

Here is a summary of the steps:

1. Randomly partition the data set into k groups.
2. Set one partition (the test set) aside.
3. Fit the model on the remaining data (the training set).
4. Calculate RMSE.
5. Repeat with each partition as the test set.

The average RMSE of the k-folds is a better measure of the model performance.

Let's look at Model 1 using 5-fold cross validation.

```
library(tidyverse)
library(Lahman)

#get the data
my_teams = Teams %>%
  filter(yearID >= 1997, yearID <= 2001) %>%
  mutate(RD = R - RA,
         Wpct = W/(W + L))

#partition the data into five groups
set.seed(100)
```

```
partition = rep(1:5, length.out = 148)
partition = sample(partition, replace = TRUE)
my_teams$partition = partition

# create empty vector to save RMSE for each fold
rmse = c()

# perform cross validation
for(i in 1:5){
  # test set
  test_teams = my_teams %>%
    filter(partition == i)

  #training set
  train_teams = my_teams %>%
    filter(partition != i)

  #fit model
  lin.fit.train = lm(Wpct ~ RD, data = train_teams)

  #get predictions and residuals on test set
  test_teams = test_teams %>%
    mutate(Wpct.pred = coef(lin.fit.train)[1] + coef(lin.fit.train)[2]*RD,
           .resid = Wpct - Wpct.pred)

  #calculate RMSE

  rmse[i] = sqrt(mean(test_teams$.resid^2))
}
```

Let's look at the RMSE for each fold and the average RMSE.

```
round(rmse,4)
```

```
## [1] 0.0209 0.0244 0.0217 0.0259 0.0217
```

```
round(mean(rmse),4)
```

```
## [1] 0.0229
```

It would also be helpful to look at this in terms of number of wins.

```
round(162*rmse,1)
```

```
## [1] 3.4 3.9 3.5 4.2 3.5
```

```
round(162*mean(rmse),1)
```

```
## [1] 3.7
```

What do we conclude from this analysis?

Using the code above as a guide, perform a 5-fold cross validation of the Pythagorean formula with exponent k. Report your results and conclusions below.