How can I perform K-Fold Cross Validation in R?

Authored by

stats writer

April 22, 2024

K-Fold Cross Validation is a statistical method used to evaluate the performance of a predictive model on a given dataset. It involves dividing the dataset into k subsets, or folds, and using one fold as a testing set while training the model on the remaining k-1 folds. This process is repeated k times, with each fold being used as the testing set once. By averaging the results of the k tests, we get a more accurate estimate of the model's performance. In R, this can be performed using the "cv.glm" function in the "boot" package. This function allows for the selection of the number of folds, as well as other options such as stratification and randomization. K-Fold Cross Validation is a useful tool for assessing the generalization ability of a model and can help in choosing the best model for a given dataset.

K-Fold Cross Validation in R (Step-by-Step)

To evaluate the performance of a model on a dataset, we need to measure how well the predictions made by the model match the observed data.

One commonly used method for doing this is known as k-fold cross-validation, which uses the following approach:

1. Randomly divide a dataset into *k* groups, or "folds", of roughly equal size.

2. Choose one of the folds to be the holdout set. Fit the model on the remaining k-1 folds. Calculate the test MSE on the observations in the fold that was held out.

3. Repeat this process *k* times, using a different set each time as the holdout set.

4. Calculate the overall test MSE to be the average of the *k* test MSE's.

The easiest way to perform k-fold cross-validation in R is by using the **trainControl()** function from the caret library in R.

This tutorial provides a quick example of how to use this function to perform k-fold cross-validation for a given model in R.

Example: K-Fold Cross-Validation in R

Suppose we have the following dataset in R:

```
#create data frame
df <- data.frame(y=c(6, 8, 12, 14, 14, 15, 17, 22, 24, 23),
x1=c(2, 5, 4, 3, 4, 6, 7, 5, 8, 9),
x2=c(14, 12, 12, 13, 7, 8, 7, 4, 6, 5))

#view data frame
df

y x1 x2
6 2 14
8 5 12
```

12 4 12

14 3 13

14 4 7

15 6 8

17 7 7

22 5 4

24 8 6

23 9 5

The following code shows how to fit a multiple linear regression model to this dataset in R and perform k-fold cross validation with k = 5 folds to evaluate the model performance:

```
library(caret)

#specify the cross-validation method
ctrl <- trainControl(method = "cv", number = 5)

#fit a regression model and use k-fold CV to evaluate performance
model <- train(y ~ x1 + x2, data = df, method = "lm", trControl = ctrl)

#view summary of k-fold CV
```

print(model)

Linear Regression

10 samples
2 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 8, 8, 8, 8, 8
Resampling results:

RMSE Rsquared MAE
3.018979 1 2.882348

Tuning parameter 'intercept' was held constant at a value of TRUE

Here is how to interpret the output:

No pre-processing occured. That is, we didn't scale the data in any way before fitting the models.The resampling method we used to evaluate the model was cross-validation with 5 folds.The sample size for each training set was 8.RMSE: The root mean squared error. This measures the average difference between the

predictions made by the model and the actual observations. The lower the RMSE, the more closely a model can predict the actual observations.Rsquared: This is a measure of the correlation between the predictions made by the model and the actual observations. The higher the R-squared, the more closely a model can predict the actual observations.MAE: The mean absolute error. This is the average absolute difference between the predictions made by the model and the actual observations. The lower the MAE, the more closely a model can predict the actual observations.

Each of the three metrics provided in the output (RMSE, R-squared, and MAE) give us an idea of how well the model performed on previously unseen data.

In practice we typically fit several different models and compare the three metrics provided by the output seen here to decide which model produces the lowest test error rates and is therefore the best model to use.

#view final model
model$finalModel

Call:

lm(formula = .outcome ~ ., data = dat)

Coefficients:

(Intercept) x1 x2

21.2672 0.7803 -1.1253

The final model turns out to be:

y = 21.2672 + 0.7803*(x1) - 1.12538(x2)

We can use the following code to view the model predictions made for each fold:

#view predictions for each fold

model$resample

RMSE Rsquared MAE Resample

1 4.808773 1 3.544494 Fold1

2 3.464675 1 3.366812 Fold2

3 6.281255 1 6.280702 Fold3

4 3.759222 1 3.573883 Fold4

5 1.741127 1 1.679767 Fold5

Note that in this example we chose to use k=5 folds, but

you can choose however many folds you'd like. In practice, we typically choose between 5 and 10 folds because this turns out to be the optimal number of folds that produce reliable test error rates.