How to Evaluate Your R Model’s Performance with Cross-Validation

Authored by

stats writer

March 4, 2026

Cross-validation is a statistical technique used to evaluate the performance of a model in R. It involves dividing the available data into multiple subsets, training the model on one subset, and testing it on the remaining subsets. This process is repeated multiple times, with different subsets being used for training and testing each time. The results are then averaged to provide a more reliable assessment of the model's performance. By using cross-validation, potential issues such as overfitting and underfitting can be identified and addressed, leading to a more accurate evaluation of the model's predictive power. Ultimately, cross-validation helps determine the effectiveness and robustness of a model in accurately predicting outcomes on unseen data.

Perform Cross Validation for Model Performance in R

In statistics, we often build models for two reasons:

To gain an understanding of the relationship between one or more predictor variables and a response variable.To use a model to predict future observations.

Cross validation is useful for estimating how well a model is able to predict future observations.

For example, we may build a mulitple linear regression model that uses *age* and *income* as predictor variables and *loan default status* as the response variable. In this case, we may wish to fit the model to a dataset and then use that model to predict, based on a new applicant's income and age, the probability that they will default on a loan.

To determine if the model has strong predictive ability, we need to use the model to make predictions on data that it has not seen before. This will allow us to estimate the prediction error of the model.

Using Cross Validation to Estimate Prediction Error

Cross validation refers to different ways we can estimate the prediction error. The general approach of cross-validation is as follows:

1. Set aside a certain number of observations in the dataset - typically 15-25% of all observations.
2. Fit (or "train") the model on the observations that we keep in the dataset.
3. Test how well the model can make predictions on the observations that we did not use to train the model.

Measuring the Quality of a Model

When we use the fitted model to make predictions on new observations, we can use several different metrics to measure the quality of the model, including:

Multiple R-squared: This measures the strength of the linear relationship between the predictor variables and

the response variable. A multiple R-squared of 1 indicates a perfect linear relationship while a multiple R-squared of 0 indicates no linear relationship whatsoever. The higher the multiple R-squared, the better the predictor variables are able to predict the response variable.

Root Mean Squared Error (RMSE): This measures the average prediction error made by the model in predicting the value for a new observation. This is the average distance between the true value of an observation and the value predicted by the model. Lower values for RMSE indicate a better model fit.

Mean Absolute Error (MAE): This is the average absolute difference between the true value of an observation and the value predicted by the model. This metric is generally less sensitive to outliers compared to RMSE. Lower values for MAE indicate a better model fit.

Implementing Four Different Cross-Validation Techniques in R

Next, we will explain how to implement the following cross validation techniques in R:

1. Validation Set Approach

2. k-fold Cross Validation

3. Leave One Out Cross Validation

4. Repeated k-fold Cross Validation

To illustrate how to use these different techniques, we will use a subset of the built-in R dataset *mtcars*:

```
#define dataset
data <- mtcars

#view first six rows of new data
head(data)

# mpg disp hp drat
#Mazda RX4 21.0 160 110 3.90
#Mazda RX4 Wag 21.0 160 110 3.90
#Datsun 710 22.8 108 93 3.85
#Hornet 4 Drive 21.4 258 110 3.08
#Hornet Sportabout 18.7 360 175 3.15
#Valiant 18.1 225 105 2.76
```

We will build a multiple linear regression model using *disp*, *hp*, and *drat* as predictor variables and *mpg* as the response variable.

Validation Set Approach

The validation set approach works as follows:

1. Split the data into two sets: one set is used to train the model (i.e. estimate the parameters of the model) and the other set is used to test the model. Typically the training set is generated by randomly selecting 70-80% of the data, and the other remaining 20 - 30% of the data is used as the test set.

2. Build the model using the training data set.
3. Use the model to make predictions on the data in the test set.
4. Measure the quality of the model using metrics like R-squared, RMSE, and MAE.

Example:

The following example uses the dataset we defined above. First, we split the data into
a training set and test set, using 80% of the data as the training set and the remaining 20% of the data as the test set. Next, we build the model using the training set. Then, we use the model to make predictions on the test set. Lastly, we measure the quality of the model using

R-squared, RMSE, and MAE.

```
#load dplyr library used for data manipulation
library(dplyr)

#load caret library used for partitioning data into training and test set
library(caret)

#make this example reproducible
set.seed(0)

#define the dataset
data <- mtcars

#split the dataset into a training set (80%) and test set (20%).
training_obs <- data$mpg %>% createDataPartition(p = 0.8, list = FALSE)

train <- data
test <- data

# Build the linear regression model on the training set
model <- lm(mpg ~ ., data = train)
```

```
# Use the model to make predictions on the test set
predictions <- model %>% predict(test)

#Examine R-squared, RMSE, and MAE of predictions
data.frame(R_squared = R2(predictions, test$mpg),
RMSE = RMSE(predictions, test$mpg),
MAE = MAE(predictions, test$mpg))

# R_squared RMSE MAE
#1 0.9213066 1.876038 1.66614
```

When comparing different models, the one that produces the lowest RMSE on the test set is the preferred model.

Pros & Cons of this Approach

The advantage of the validation set approach is that it's straightforward and computationally efficient. The disadvantage is that the model is built only using a portion of the total data. If the data that we happen to leave out of the training set contains interesting or valuable information, the model will not take this into account.

k-fold Cross Validation Approach

The k-fold cross validation approach works as follows:

1. Randomly split the data into k "folds" or subsets (e.g. 5 or 10 subsets).
2. Train the model on all of the data, leaving out only one subset.
3. Use the model to make predictions on the data in the subset that was left out.
4. Repeat this process until each of the k subsets has been used as the test set.
5. Measure the quality of the model by calculating the average of the k test errors. This is known as the cross-validation error.

Example

In this example, we first split the data into 5 subsets. Then, we fit the model using all of the data except one subset. Next, we use the model to make predictions on the subset that was left out and record the test error (using R-squared, RMSE, and MAE). We repeat this process until each subset has been used as the test set. Then we simply compute the average of the 5 test errors.

```
#load dplyr library used for data manipulation
library(dplyr)

#load caret library used for partitioning data into training and test set
library(caret)

#make this example reproducible
set.seed(0)

#define the dataset
data <- mtcars

#define the number of subsets (or "folds") to use
train_control <- trainControl(method = "cv", number = 5)

#train the model
model <- train(mpg ~ ., data = data, method = "lm",
trControl = train_control)

#Summarize the results
print(model)

#Linear Regression
#
#32 samples
```

 # 3 predictor

#

#No pre-processing

#Resampling: Cross-Validated (5 fold)

#Summary of sample sizes: 26, 25, 26, 25, 26

#Resampling results:

#

 # RMSE Rsquared MAE

 # 3.095501 0.7661981 2.467427

#

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

Pros & Cons of this Approach

The advantage of the k-fold cross validation approach over the validation set approach is that it builds the model several different times using different chunks of data each time, so we have no chance of leaving out important data when building the model.

The subjective part of this approach is in choosing what value to use for k, i.e. how many subsets to split the data into. In general, lower values of k lead to higher bias but lower variability, while higher values of k lead

to lower bias but higher variability.

In practice, k is typically chosen to be 5 or 10, as this number of subsets tends to avoid too much bias and too much variability simultaneously.

Leave One Out Cross Validation (LOOCV) Approach

The LOOCV approach works as follows:

1. Build the model using all observations in the dataset except for one.
2. Use the model to predict the value of the missing observation. Record the test error of this prediction.
3. Repeat this process for every observation in the dataset.
4. Measure the quality of the model by calculating the average of the all of the prediction errors.

Example

The following example illustrates how to use perform LOOCV for the same dataset that we used in the previous examples:

#load *dplyr* library used for data manipulation

```
library(dplyr)

#load caret library used for partitioning data into training and test set
library(caret)

#make this example reproducible
set.seed(0)

#define the dataset
data <- mtcars

#specify that we want to use LOOCV
train_control <- trainControl(method = "LOOCV")

#train the model
model <- train(mpg ~ ., data = data, method = "lm", trControl = train_control)

#summarize the results
print(model)

#Linear Regression
#
#32 samples
# 3 predictor
```

#

#No pre-processing

#Resampling: Leave-One-Out Cross-Validation

#Summary of sample sizes: 31, 31, 31, 31, 31, 31, ...

#Resampling results:

#

 # RMSE Rsquared MAE

 # 3.168763 0.7170704 2.503544

#

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

Pros & Cons of this Approach

The benefit of LOOCV is that we use all data points, which generally reduces potential bias. However, since we use the model to predict the value for each observation, this could lead to higher variability in prediction error.

Another drawback of this approach is that it has to fit so many models that it can become computationally inefficient and cumbersome.

Repeated k-fold Cross Validation Approach

We can perform repeated k-fold cross validation by simply performing k-fold cross validation several times. The final error is the mean error from the number of repeats.

The following example performs 5-fold cross validation, repeated 4 different times:

```
#load dplyr library used for data manipulation
library(dplyr)

#load caret library used for partitioning data into training and test set
library(caret)

#make this example reproducible
set.seed(0)

#define the dataset
data <- mtcars

#define the number of subsets to use and number of times to repeat k-fold CV
train_control <- trainControl(method = "repeatedcv",
```

```
number = 5, repeats = 4)

#train the model
model <- train(mpg ~ ., data = data, method = "lm",
trControl = train_control)

#summarize the results
print(model)

#Linear Regression
#
#32 samples
# 3 predictor
#
#No pre-processing
#Resampling: Cross-Validated (5 fold, repeated 4 times)
#Summary of sample sizes: 26, 25, 26, 25, 26, 25, ...
#Resampling results:
#
# RMSE Rsquared MAE
# 3.176339 0.7909337 2.559131
#
#Tuning parameter 'intercept' was held constant at a
value of TRUE
```

Pros & Cons of this Approach

The benefit of the repeated k-fold cross validation approach is that for each repeat, the data will be split up into slightly different subsets, which should give an even more unbiased estimate of the prediction error of the model. The drawback of this approach is that it can be computationally intensive since we have to repeat the model-fitting process several times.

How to Choose the Number of Folds in Cross Validation

The most subjective part of performing cross validation is n deciding how many folds (i.e. subsets) to use. In general, the smaller number of folds, the more biased the error estimates, but the less variable they will be. Conversely, the higher number of folds, the less biased the error estimates, but the higher variable they will be.

It's also important to keep in mind computational time. For each fold, you will have to train a new model, and if this is a slow process then it could take a long time if you choose a high number of folds.

In practice, cross validation is typically performed with 5 or 10 folds because this allows for a nice balance

between variability and bias, while also being computationally efficient.

How to Choose a Model After Performing Cross Validation

Cross validation is used as a way to assess the prediction error of a model. It can help us choose between two or more different models by highlighting which model has the lowest prediction error (based on RMSE, R-squared, etc.).

Once we have used cross validation to pick the best model, we then use *all* of the data available to fit the chosen model. We don't use the actual model instances we trained during cross validation for our final model.

For example, we may use 5-fold cross validation to determine which model is best to use between two different regression models. However, once we identify which model is best to use, we then use *all* of the data to fit the final model. In other words, we don't leave out one of the folds when building the final model.