# Multiple Regression for Predicting a Quantitative Target Variable

Shiju Zhang

2026-01-18

## Contents



### Introduction

This document demonstrates a machine learning workflow for multiple regression using a 70/30 train-test split.

### Load Required Libraries

```
library(tidyverse)
library(caret)
library(kableExtra)
```

### Load and Explore Data

For this example, we'll use the built-in mtcars dataset. In practice, you would load your own dataset.

```
cat("Dataset dimensions:", dim(mtcars)[1], "rows and", dim(mtcars)[2], "columns\n")
```

```
## Dataset dimensions: 32 rows and 11 columns
```

```
cat("Target variable: mpg (Miles per Gallon)\n")
```

```
## Target variable: mpg (Miles per Gallon)
```

```
cat("Predictor variables:", paste(names(mtcars)[-1], collapse = ", "), "\n")
```

```
## Predictor variables: cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb
```

## Data Preparation

```r
# Set seed for reproducibility
set.seed(123)

# Create 70/30 train-test split
train_index <- createDataPartition(mtcars$mpg, p = 0.7, list = FALSE)
train_data <- mtcars[train_index, ]
test_data <- mtcars[-train_index, ]

cat("Training set size:", nrow(train_data), "observations\n")
```

```
## Training set size: 24 observations
```

```r
cat("Test set size:", nrow(test_data), "observations\n")
```

```
## Test set size: 8 observations
```

## Train Multiple Regression Model

```r
# Train linear regression model
model <- lm(mpg ~ ., data = train_data) # The dot represents all other variables

# Model summary
summary(model)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8335 -1.2145 -0.0044  1.1104  4.1136
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14.72655   27.33755  -0.539    0.599
## cyl           1.33019    1.41806   0.938    0.365
## disp          0.01218    0.01887   0.645    0.530
## hp           -0.01756    0.02588  -0.678    0.509
## drat          2.67080    2.51575   1.062    0.308
## wt           -3.65531    2.37988  -1.536    0.149
## qsec          1.02957    0.86524   1.190    0.255
## vs           -0.37848    3.00566  -0.126    0.902
## am            1.82482    2.74395   0.665    0.518
## gear          3.74182    2.99973   1.247    0.234
## carb         -1.42190    1.42397  -0.999    0.336
##
## Residual standard error: 2.641 on 13 degrees of freedom
## Multiple R-squared:  0.8971, Adjusted R-squared:  0.818
## F-statistic: 11.33 on 10 and 13 DF,  p-value: 6.948e-05
```

## Make Predictions

```
# Predict on test data
predictions <- predict(model, newdata = test_data)

# Create results data frame
results <- data.frame(
  Actual = test_data$mpg,
  Predicted = predictions
)
head(results)
```

```
##                   Actual Predicted
## Mazda RX4 Wag       21.0 21.806487
## Duster 360          14.3 13.367084
## Chrysler Imperial   14.7  9.796747
## Dodge Challenger    15.5 17.409006
## Porsche 914-2       26.0 29.354589
## Lotus Europa        30.4 29.017943
```

## Calculate Performance Metrics

```
# Calculate metrics
rmse <- sqrt(mean((results$Actual - results$Predicted)^2))
mae <- mean(abs(results$Actual - results$Predicted))
r2 <- cor(results$Actual, results$Predicted)^2
mape <- mean(abs((results$Actual - results$Predicted)/results$Actual)) * 100

# Create metrics table
metrics_table <- data.frame(
  Metric = c("RMSE", "MAE", "R-squared", "MAPE (%)"),
  Value = c(
    round(rmse, 3),
    round(mae, 3),
    round(r2, 3),
    round(mape, 3)
  ),
  Description = c(
    "Root Mean Square Error",
    "Mean Absolute Error",
    "Coefficient of Determination",
    "Mean Absolute Percentage Error"
  )
)

# Display metrics table
kable(metrics_table, caption = "Model Performance Metrics on Test Set") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 1: Model Performance Metrics on Test Set

| Metric | Value | Description |
|---|---|---|
| RMSE | 3.989 | Root Mean Square Error |
| MAE | 2.844 | Mean Absolute Error |

| | | |
|---|---|---|
| R-squared | 0.658 | Coefficient of Determination |
| MAPE (%) | 16.631 | Mean Absolute Percentage Error |

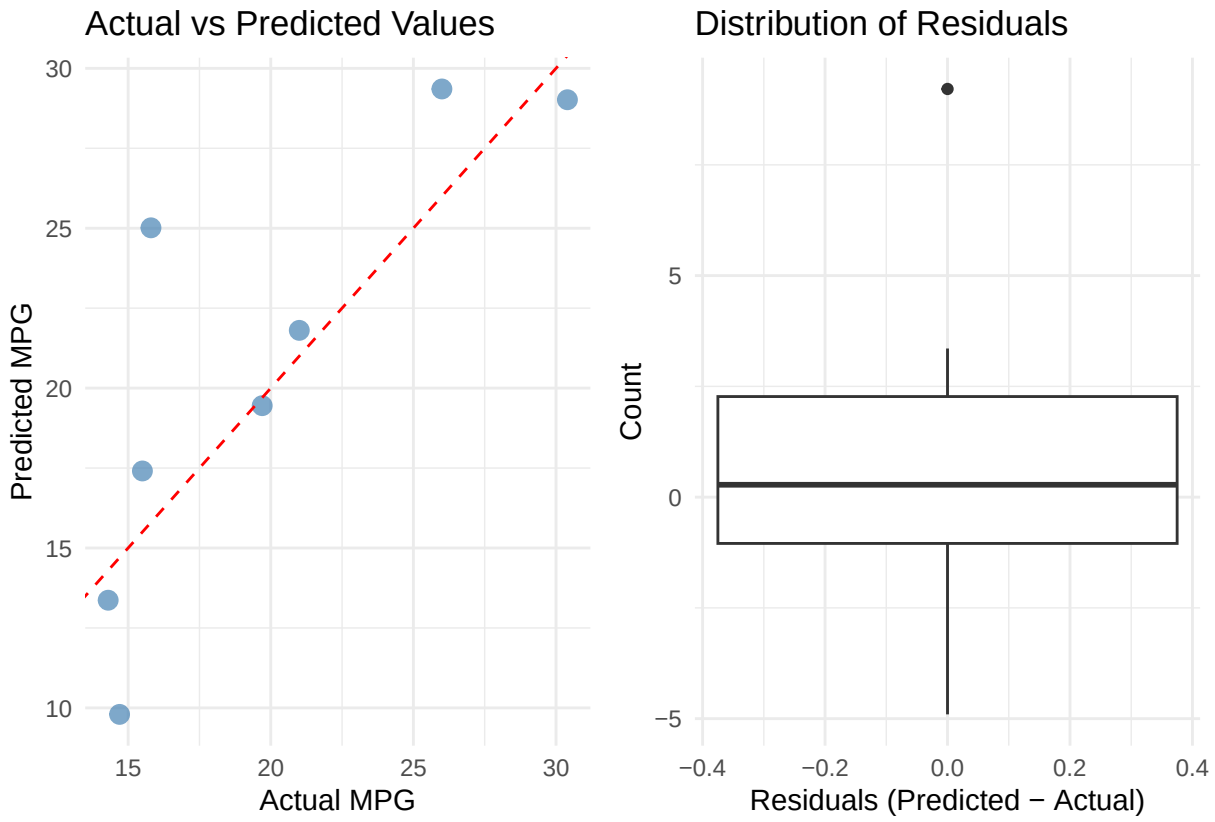
### Visualize Results

```
# Create visualization
library(patchwork)

p1 <- ggplot(results, aes(x = Actual, y = Predicted)) +
  geom_point(color = "steelblue", size = 3, alpha = 0.7) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(title = "Actual vs Predicted Values",
       x = "Actual MPG",
       y = "Predicted MPG") +
  theme_minimal()

p2 <- ggplot(results, aes(y = Predicted - Actual)) +
  geom_boxplot() +
  labs(title = "Distribution of Residuals",
       x = "Residuals (Predicted - Actual)",
       y = "Count") +
  theme_minimal()

p1 + p2
```



## Cross-Validation Results

```
# Perform k-fold cross-validation on training data
set.seed(123)
cv_model <- train(mpg ~ .,
                  data = train_data,
                  method = "lm",
                  trControl = trainControl(method = "cv", number = 5))

cv_results <- cv_model$results
kable(cv_results, caption = "5-Fold Cross-Validation Results") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2: 5-Fold Cross-Validation Results

| intercept | RMSE | Rsquared | MAE | RMSESD | RsquaredSD | MAESD |
|---|---|---|---|---|---|---|
| TRUE | 4.390669 | 0.6540267 | 3.528197 | 1.852619 | 0.4033106 | 1.64065 |

## Conclusion

The multiple regression model achieved an R-squared of 0.658 on the test set, indicating that approximately 65.8% of the variance in MPG is explained by the model. The RMSE of 3.989 suggests the average prediction error.

## Critical Thinking: Why Cross-Validation?

The Problem with Simple Train/Test Split: if you use a 70/30 split (8 vehicles for training, 24 for testing), you might get unlucky.

- What if your test set has mostly vehicles with higher horse power?
- What if your test set has mostly heavy vehicles?
- The model performance could vary wildly depending on which vehicles end up in the test set!

The Solution: Cross-Validation

Cross-validation helps solve this by systematically testing your model on different parts of your data. We creates $k$ folds with original data. We run a model using each of the folds as a test set and the remaining folds as a training set. The performance of each model is averaged to get the final performance metric. The standard deviation of the $k$ separate performance metrics can also be calculated. If we use the RMSE as a performance metric, the final CV performance metric can be expressed as $AVG.RMSE \pm SD$ (say $4.39 \pm 0.35$), which tells you that

- The typical error is about 4.39 miles pewr gallon.
- The error varies between 4.04 and 4.74 depending on the data.
- Your model is fairly consistent across different types of vehicles.

## Appendix: How Cross-Validation Works

CV takes 3 steps.

- Step 1: Partition the Data We divide the original dataset into $k$ equally sized subsets, called "folds." For instance, with 100 data points and $k = 5$, we would create 5 folds, each containing 20 data points.

- Step 2: Iterative Training and Testing The algorithm performs $k$ separate modeling iterations. In each iteration:

  Test Set: One fold serves as the test set

  Training Set: The remaining $(k-1)$ folds combine to form the training set

  This process rotates through all folds, ensuring each data point appears in the test set exactly once.

- Step 3: Performance Aggregation After completing all $k$ iterations, we calculate performance metrics (such as RMSE or accuracy) for each model. The final performance estimate is obtained by averaging these $k$ individual metrics, providing a more robust and reliable assessment of the model's generalization capability.

This methodology offers several advantages:

- (Reduced Variance) By averaging across multiple test sets, we minimize the impact of any particularly favorable or unfavorable data split.
- (Maximized Data Utilization) Every data point contributes to both training and testing phases.
- (Reliable Performance Estimation) The averaged metrics offer a more trustworthy indicator of how the model will perform on unseen data.