How to Predict Outcomes Using Logistic Regression in R

Authored by

stats writer

November 19, 2025

The core objective of building any statistical model is to generalize its findings to unseen data. When working with logistic regression models in R, the specialized function responsible for this generalization is **predict()**. This function is essential for moving beyond model fitting to practical application. It consumes the established, fitted model object and a new dataset containing predictor variables, subsequently generating a vector of predicted likelihoods for each observation.

These predicted values, which represent the probability of the positive outcome occurring, form the foundation for classifying new data points. Understanding how to correctly implement and interpret the predict() function, particularly the critical `type="response"` argument, is fundamental for robust binary classification tasks in R.

After successfully fitting a logistic regression model using the generic linear modeling function (`glm`) in **R**, we leverage the powerful **predict()** function to estimate the response variable's value for observations not included in the original training data. This mechanism ensures the model's predictive validity and utility in real-world scenarios.

Understanding the Syntax of predict() for Binary Outcomes

The structure of the **predict()** function is standardized across many different model types in R, but its application in logistic regression requires careful attention to the arguments, especially the `type` parameter. When applied to a model fitted with `family=binomial`, the function requires three primary arguments to execute the prediction successfully.

The general syntax follows this structure:

predict(object, newdata, type="response")

The specific arguments serve the following crucial roles:

object: This must be the fitted model object itself--the result generated by the `glm()` function. It contains all the necessary coefficients and statistical properties required to calculate the log-odds for the new data.
newdata: This is a data frame containing the same predictor variables that were used to train the original model. Crucially, the variable names in this new data frame must match the names used during the fitting process exactly.
type: This argument dictates the output format of the prediction. For logistic regression, setting **type="response"** is mandatory if we require the output to be the predicted **probability** (a value between 0 and 1) of the event occurring. If omitted, the default output is the linear predictor (log-odds).

The remainder of this article demonstrates a practical implementation of this function using a well-

known R dataset.

Case Study: Predicting Transmission Type Using the mtcars Dataset

To illustrate the practical application of the **predict()** function, we will utilize the familiar, built-in **R** dataset known as mtcars (Motor Trend Car Road Tests). This dataset provides characteristics for 32 automobiles and is frequently used for demonstrating statistical modeling techniques. Our goal is to model the likelihood of a car having a manual transmission based on specific engine characteristics.

First, we inspect the structure of the data to confirm the variables we will be using:

#view first six rows of *mtcars* dataset
head(mtcars)

mpg cyl disp hp drat wt qsec vs am gear carb
Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4
Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4
Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

Fitting the Logistic Regression Model

We will construct a logistic regression model where the binary response variable is **am** (transmission type: 0 = automatic, 1 = manual). We will use two key quantitative predictors: **disp** (displacement, in cubic inches) and **hp** (horsepower). The use of `family=binomial` within the `glm()` function specifies that we are performing a binomial classification task.

After fitting the model, we review the summary output, noting the coefficients that indicate the relationship between our predictors and the log-odds of having a manual transmission.

#fit logistic regression model
model <- glm(am ~ disp + hp, data=mtcars, family=binomial)

#view model summary
summary(model)

Call:
glm(formula = am ~ disp + hp, family = binomial, data = mtcars)

Deviance Residuals:
Min 1Q Median 3Q Max
-1.9665 -0.3090 -0.0017 0.3934 1.3682

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) 1.40342 1.36757 1.026 0.3048
disp -0.09518 0.04800 -1.983 0.0474 *
hp 0.12170 0.06777 1.796 0.0725 .

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 43.230 on 31 degrees of freedom
Residual deviance: 16.713 on 29 degrees of freedom
AIC: 22.713

Number of Fisher Scoring iterations: 8

The summary confirms that both displacement (**disp**) and horsepower (**hp**) contribute significantly to the prediction, as indicated by their respective p-values. The fitted model object, `model`, is now ready to be used for predicting outcomes for entirely new observations.

Generating Probabilistic Predictions on New Data

The true test of a model lies in its ability to predict outcomes for observations it has not processed previously. We construct a new data frame, `newdata`, containing eight hypothetical cars, specifying their displacement (disp) and horsepower (hp). Although the actual transmission type (am) is included here for later validation, it is not used by the **predict()** function. We ensure the predictor column names are identical to those used in the fitted model.

The essential step is invoking **predict(model, newdata, type="response")**. By setting `type="response"`, we instruct **R** to transform the calculated log-odds (linear predictor) back into the range , representing the predicted probability of the positive class (am=1, manual transmission).

#define new data frame
newdata = data.frame(disp=c(200, 180, 160, 140, 120, 120, 100, 160),
hp=c(100, 90, 108, 90, 80, 90, 80, 90),
am=c(0, 0, 0, 1, 0, 1, 1, 1))

```
#view data frame
newdata

#use model to predict value of am for all new cars
newdata$am_prob <- predict(model, newdata, type="response")

#view updated data frame
newdata
```

disp hp am am_prob
1 200 100 0 0.004225640
2 180 90 0 0.008361069
3 160 108 0 0.335916069
4 140 90 1 0.275162866
5 120 80 0 0.429961894
6 120 90 1 0.718090728
7 100 80 1 0.835013994
8 160 90 1 0.053546152

Interpreting the Probabilistic Output (am_prob)

The new column, `am_prob`, contains the predicted probability that each vehicle possesses a manual transmission (am=1). These probabilities are direct outputs of the sigmoid function applied to the linear combination of the predictor variables.

For instance, analyzing the first three rows:

Car 1 (200 disp, 100 hp) has a predicted probability of **0.0042** of being manual. This low value strongly suggests the car is automatic (am=0).
Car 2 (180 disp, 90 hp) has an even lower probability of **0.0084**, also highly likely to be automatic.
Car 3 (160 disp, 108 hp) shows a predicted probability of **0.3359**. Since this value is below the common classification threshold of 0.5, the model would classify this vehicle as automatic, despite a higher likelihood than the first two cars.

These probabilistic outputs must be converted into discrete class predictions (0 or 1) before the model's classification performance can be formally assessed.

Classifying Outcomes and Generating the Confusion Matrix

While the predicted probabilities (the output of **predict()** with `type="response"`) are informative, for classification, we must convert these continuous values into discrete binary outcomes (0 or 1).

This is typically done by setting a threshold, commonly 0.5. If the predicted probability is greater than 0.5, we classify the observation as the positive class (am=1, manual); otherwise, it is classified as the negative class (am=0, automatic).

We use this classification step, combined with the known actual values from the `newdata` frame, to construct a confusion matrix using the **table()** function. The confusion matrix is a critical tool for visualizing the performance of a classification algorithm, detailing the counts of true positives, true negatives, false positives, and false negatives.

#create vector that contains 0 or 1 depending on predicted value of am (using 0.5 threshold)
am_pred = rep(0, dim(newdata))
am_pred = 1

#create confusion matrix: predicted values (rows) vs. actual values (columns)
table(am_pred, newdata$am)

am_pred 0 1
0 4 2
1 0 2

From the output of the confusion matrix, we observe the following results for the 8 new cars: 4 True Negatives (correctly predicted automatic cars), 2 True Positives (correctly predicted manual cars), 0 False Positives, and 2 False Negatives (actual manual cars incorrectly predicted as automatic).

Calculating Overall Prediction Accuracy

The final step in evaluating the utility of the fitted model and its predictions is calculating the overall classification accuracy. This metric represents the proportion of total observations for which the model's prediction aligns with the actual outcome. We achieve this efficiently in **R** using the **mean()** function applied to a logical comparison between the predicted class vector (`am_pred`) and the actual class vector (`newdata$am`).

#calculate percentage of observations the model correctly predicted response value for
mean(am_pred == newdata$am)

0.75

The result indicates that the logistic regression model, utilizing the **predict()** function, successfully classified the transmission type (**am** value) for **75%** of the eight new vehicles in the validation data frame. This demonstrates the seamless workflow in **R** from model fitting, through prediction using **predict()**, to final performance evaluation.